Global iSeries Application Performance Analyzer

User Manual

V06M01G

TM

TM

www.giapa.com

Table of contents

Introduction: GiAPA Objectives

The name GiAPA is an acronym of the words describing the aim of the software product:
Global **i** **A**pplication **P**erformance **A**nalyzer.

Our overall goal was to create a software product that would enable the average programmer, operator, or systems analyst to cope with i.e., identify and know how to solve performance inefficiencies in applications running on the IBM i (Power Systems including AS/400, iSeries, and System i).

With GiAPA on board you seldom must consult expensive external performance experts in order to pinpoint a performance problem.

GiAPA customers tend to use GiAPA for much more than just performance analysis given that the software collects a lot of data automatically without putting any strain on the system. GiAPA is often used by operations as the daily source of information about how the system is running. This added feature is supported by GiAPA's comprehensive options for presenting the usage of the computer in graphical reports.

GiAPA is designed to analyze applications running under IBM i and provide information re:

- applications with performance problems
- identifying the problems
- location of the program(s) with problems
- how to efficiently improve the run of the application

GiAPA is not designed to identify:

- additional hardware to purchase in order to cope with (conceal?) performance problems.

GiAPA ensures that your existing computer resources are used optimally so your hardware does not have to grow faster than the number of transactions processed. Therefore, GiAPA is not just another capacity planning tool, nor an interactive (manpower eating) monitor.

Superior performance was another GiAPA objective. We could hardly believe the results ourselves: **GiAPA normally uses less than 0.1 per cent of one CPU when collecting information about all active jobs and operating system tasks!** Hence, even on a machine struggling with severe performance problems, initiating GiAPA data collection will not affect the system.

During the development of GiAPA we have tried to avoid "re-inventing the wheel". Therefore, functionality obtainable in tools within the operating system has not been included in GiAPA whereas a few user-friendly front ends to tools offered by IBM are included.

> NOTE: iPerformance welcomes any ideas that will enable GiAPA to further improve application performance. Please send us an email if you want new options included in the product!

Getting acquainted with GiAPA is not a problem: www.giapa.com contains a complete set of easy tutorials and a downloadable self-study course including a test library with a collection of hands-on exercises providing typical examples of key GiAPA features and capabilities.

Accuracy of Results Provided by GiAPA

GiAPA's primary aim is to locate applications causing performance problems, and when possible identify what these problems are, thereby often implying how they may be solved. To ensure GiAPA's remarkably low use of resources the data collection will restart automatically once every 24 hours (typically at midnight); this implies that data is not collected for a minute or two during restart. Hence, GiAPA is not a job accounting tool, and does not claim to provide data with 100% accuracy. However, the GiAPA data should reach 100% accuracy with total job statistics (CPU usage down to milliseconds) collected from the job accounting function; this only requires that *JOB is specified for the system value QACGLVL.

Background for Developing GiAPA

The IBM i is an operating system capable of delivering a remarkably high performance for the applications running on the machine if the programs are designed and coded well. Therefore, courses aimed at system analysts and programmers typically do not cover performance considerations since the machines are designed to be very efficient and the operating system extremely powerful.

The results in terms of interactive response or batch run times are quite acceptable in most cases. Occasionally the odd, but irritating situations occur, where the expectations of the end users for a rapid answer to a given request are not fully met.

Such problems are typically "solved" by ordering an upgrade. If a slow program is transferred to a new and faster machine a given performance problem is most likely concealed from the end user. This approach is like alleviating a headache with aspirins, instead of trying to identify the reason for the pain and possibly removing the cause. In the long run simply opting to upgrade the hardware adds up to become a rather expensive "solution".

In case of performance problems, it is rare for an IT department to try to optimize a slow application. Computers today are very complex with many layers of interacting software, and it is not an easy task to dive into a "haystack" of programs and operating system routines and find the "needle" causing the delay.

The inventor and key developer of GiAPA has worked with performance on the System i since the 80's where the machine was called System/38. He has taught more than 200 courses for IBM on OS/400 performance in 22 countries and has been an invited speaker at multiple COMMON conferences throughout Europe and the United States. He assisted many companies on a "No cure – no pay" basis with optimizing slow-running applications. Every time the reason behind a problem was identified, he added code to analyzer-programs, enabling them to automatically pinpoint when similar symptoms were present in future analyses. Over a 10 years' period this approach resulted in the development of more than 200 programs that over time were amalgamated into one software product – and GiAPA was born!

The average programmer not possessing special expertise within performance will with the assistance of GiAPA be able to check the performance of applications and identify all the most common performance flaws. GiAPA is easy to use – and a full GiAPA course takes less than a day. The on-line tutorials can be found on the GiAPA home page:

https://www.giapa.com/tutorials

Our motto is **"Never accept bad performance – IBM i can do better than that!"**

How does GiAPA Work – a "must read" section!

GiAPA's main performance analysis can be viewed as a three-step process:

1. Collection in compressed binary format of two types of performance data, running in batch and typically using less than 0.1% of one CPU.

2. Expansion and analysis of the collected data, where one batch job creates a number of data base file members containing the results.

3. Interactive data presentation offering many displayed or printed reports and graphics with comprehensive selection and sort options.

When starting GiAPA data collection, a command parameter (having a reasonable default value) specifies the limit in CPU per cent for when GiAPA should collect additional data for jobs that exceed this limit within a 15 sec interval. The situations where jobs exceed this user defined limit are called "**HotSpots**".

GiAPA's performance data collection obtains the data in two different ways. Understanding these two methods is mandatory for anyone wanting to interpret the exception reports produced by GiAPA. The two types of data are:

A. **Interval data** collected by job GIAPAPFCOL: statistics of 22 types of resources (CPU, I/Os, etc.) used by all active jobs and tasks. This information is received from an operating system API at 15 seconds' intervals throughout the entire GiAPA performance data collection run. These statistics are fairly accurate, but do not include jobs/tasks that both started and ended within one interval. Basic information for such small jobs is collected from QHSTLOG or job accounting. The expansion and analysis step will normally only keep records showing a certain minimum use of resources. Jobs hardly using any resources are accumulated in summary records by job type and shown in the reports as *BATCH, *INTERACT, etc., unless the user requests all data to be stored. Disk usage percentage for the system ASP is also retrieved every 15 seconds.

B. **HotSpot data** collected by GIAPAHOTSP jobs: open file statistics and program call stacks for jobs having exceeded the specified limit for CPU within a 15 sec interval. These snapshot data are true sampling, their contents are like the data seen when using command DSPJOB (Display Job), Option 11 (Call stack) and 14 (Open Files). If only a few "HotSpot" snapshots are obtained for a job, they may be insignificant. In contrast, "heavy" jobs trigger more and more HotSpots resulting in the samples becoming increasingly accurate, reliable, and statistically significant.

   For multithreaded jobs (where JAVA is the typical example), HotSpot call stack and file usage data is collected for the threads having "RUN" status and/or using the most CPU. This is in accordance with values for number of threads, thread CPU-% and "Retrieve info if thread in wait state" as specified by the user in the installation parameters.

Whilst job GIAPAPFCOL always uses minimal CPU time, the GIAPAHOTSP jobs may use more if data is collected for numerous HotSpots because the CPU limit for collecting HotSpot data is set very low, if the jobs involved have an excessive number of files opened, or if GIAPA Menu Option 12 is used for watching specific job(s) very intensively. Additional GIAPAHOTSP jobs are automatically initiated as needed.

Note that complex and longer lasting I/O functions below the MI (Machine interface) level may prohibit call stack retrieval, causing the HotSpot data collection to miss one or more samples.

Typical examples in this category include data base index generations and complex SQL / Query functions.

Before the collected performance data can be used, it must be expanded and analyzed. The user simply selects one or more input members, each containing data from a data collection. This allows e.g. an entire week to be consolidated to one set of output data providing the base for all the GiAPA reports. The expansion and analysis is made in a batch run involving a lot of rather complex processes that all are transparent to the user.

The use of CPU time in the GiAPA output based on the interval data is very accurate, because complete data are collected for all jobs and tasks by the performance collector API. The counters for I/Os are by the API supported for primary threads only. For multithreaded jobs, the I/O values reported may not reflect the total job activity.

Other results are based on HotSpot data, i.e. data was only collected for the 15 second intervals where the job exceeded the user defined limit (or if the option to watch a certain job in more details is used). To be aware that a job happened to be sending a message once (known because last program in the call stack was QMHSNSTA = Send Status Message) when a HotSpot occurred is of little interest. In contrast, if a job constantly generates HotSpots showing QMHSNSTA as the current program, it becomes interesting, since status messages are useless in the batch environment.

The usage percentages for different programs within HotSpots show percentages of the elapsed time used by the job. It does not show CPU usage. On the contrary, functions using any type of synchronous I/O (which implies wait for disk I/O completion) are generally more likely to be seen as “active” at HotSpots, compared to program code only using CPU time. For the HotSpots it is also important to observe whether they occur all the time (i.e. in the majority of the 15 seconds’ intervals), or only occasionally with longer intervals.

The expanded GiAPA performance data offers many possibilities for displaying or printing reports and graphics. These include a large variety of different analyses, each depicting various use of resources and in many cases pinpointing inefficiencies (if any): slow interactive response, batch-type work running interactively, inefficient database accesses, unintended loops gobbling up CPU time, generation of data base indexes, etc.

Included with the software product are also a few additional tools and menus facilitating the use of IBM supplied tools like PEX (Performance Explorer) and TRCJOB (Trace Job). These other options may be used to further analyze a problem area after GiAPA has identified a job as being a likely candidate for optimization. Although GiAPA in most cases can decipher what the reason is for unsatisfactory performance within a job, analysis by supplementary tools can in rare cases be useful to pinpoint a problem.

ColdSpots work the same as HotSpots, i.e. they collect file usage and call stack data. ColdSpots are triggered if an interactive job has used resources without terminating a transaction within the termination of two consecutive intervals. This feature was added based on a suggestion from a GiAPA customer, where end-users sometimes experienced minute-long response times. The ColdSpot data will in such cases reveal the program and statement number where the delay occurs – typically due to an object or a record lock.

GiAPA does not provide an analysis of the Query Optimizer's choice of access methods for SQL, as this feature is available through IBM’s SQL Performance Center within the Access Client Solutions. However, GiAPA includes a tool called “SQL Observer”, which is also available as a standalone product. This tool facilitates the automated collection of Access Plans required by IBM’s SQL Performance Center. SQL Observer offers a comprehensive overview of the Plan Cache data dumps and highlights, among other things, whether the access plan for an SQL statement has changed.

Introduction to GiAPA Graphics

GiAPA creates the input for diagrams in different ways:

Fully automated graphics only requires using F14 to create the input data to a graph. This option is available from many of GiAPA's standard reports. The call stack analysis showing a percentage for the most used "Active programs" is a typical example where a fully automated generation of the input data for a chart is available.

Semi-automatic graphics, also available from many standard reports, are close to automatic, with the exception that the user selects how many records should be included by positioning the cursor on the last record to include.

User defined graphics are created using GiAPA Menu Option 26.

Please note: Generating a new graphics data file overwrites any old file with the same name, but GiAPA will keep and reuse any changes made to graph type, "Show Values" and selected color palette.

To keep input data from previously generated charts, use GiAPA Menu Option 28 to rename the file before data for the new chart is generated.

Input data for charts showing total usage of a resource within a selected time period is available from GiAPA Menu Option 21, selection 3.

GiAPA Menu Option 16 offers comprehensive graphics options allowing detailed selection pertaining to both jobs and time.

GiAPA Menu Option 99 shows amongst others various GiAPA commands that may be used to generate various predefined diagrams from a command line or in batch runs.

The result is in all cases

1. a record in file GIAPALIB/GIAPA260P1 containing the attributes of the graph (chart type, explanatory texts, etc.), and

2. a small file serving containing the input data for the generation of the html code used to show the diagram - the file layout is described in member GIAPA265P2 of the GIAPALIB/GIAPA_QDDS source file.

Both the attribute record and the data may be edited through GiAPA Menu Option 28.

Generation of a graph usually completes within seconds, after which the result is displayed immediately. Should that not be the case, please see the last page of this manual.

Definitions and FAQs (Frequently Asked Questions)

If you want to use GiAPA and be able to interpret the GiAPA results we strongly encourage you to familiarize yourself with the terms in this section.

What is an Interval?

Performance data are received from the operating system Performance Collector APIs every 15 seconds. An interval defines the time between such two events.

What is a HotSpot – and a ColdSpot?

A HotSpot is triggered when the GiAPA performance data collection (job GIAPAPFCOL) identifies a job that within an interval exceeded the CPU-% limit defined by the user when starting the data collection. Job GIAPAPFCOL passes the identification of the job exceeding the usage limits to a GIAPAHOTSP job, which collects the HotSpot data: call stack and file usage information for the job. The HotSpot information basically corresponds to the information that can be obtained by the DSPJOB command, Option 11 (call stack) and 14 (open files).

HotSpots are samples that only are recorded whenever the CPU usage limit is exceeded. For jobs using excessive resources repeatedly and thus causing many HotSpots, the sampling becomes quite accurate, and will for any “heavy” job normally be a very reliable way of getting exact information about what the job was accomplishing.

ColdSpots collects the same data as HotSpots, but are triggered by interactive jobs that use minimal CPU without terminating a transaction within two consecutive collection intervals.

How is CPU percentage calculated by GiAPA?

The total CPU percentage includes all CPUs allocated to the LPAR. Total CPU usage per collection interval is received in seconds from the operating system, and may therefore not always exactly agree with the sum of the jobs/tasks CPU, which is received in milliseconds.

The CPU-% for a single job or task is calculated as percentage of

- total CPU capacity if less than one CPU is assigned to the LPAR
- one CPU only if the LPAR has a CPU capacity of at least 1.0.

A job with several threads using multiple CPUs in parallel could therefore display more than 100 % CPU.

What is shown by the “MAX” values in the “Job Performance Summary” report?

The lines with MAX values included in some reports pinpoint the maximum use of each resource within any interval during the performance data collection. It is used to show peaks. Note that the MAX values shown for a job do not necessarily all originate from the same collection interval.

What are physical I/Os?

Physical I/Os mean that data is transferred between external memory (typically disk) and main memory. A physical I/O operation may transfer the data in huge blocks containing many

records, described by the term physical I/O blocking. The system will never transfer less than one “page” = 4K of data in a physical I/O, but may transfer much larger blocks.

What are synchronous and asynchronous I/Os?

Physical I/Os are divided into synchronous and asynchronous I/Os.

If a program must wait for the completion of a physical I/O operation it is called a synchronous I/O. A good example is random read by key of a record. After issuing this type of read instruction, the program must wait for database management to deliver the wanted record, unless the record happens to have been used recently and therefore still exists in the main memory.

If the program process can continue while the operating system handles the I/O it is called an asynchronous I/O. These I/Os are preferable from a performance point of view because they do not delay the job. A typical example is a write instruction for a record. Database management receives the record, and control is immediately passed back to the program, which can continue running while the operating system as soon as possible asynchronously transfers the record to the disk.

When a file is read sequentially, the operating system can anticipate the coming read requests from the program and read ahead asynchronously. This can significantly expedite a job.

If possible, blocking physical I/Os will also speed up a job considerably. The operating system will attempt to use record blocking for physical I/Os e.g. when a file is read in “arrival sequence”, i.e. in the sequence in which the records are stored on the disk. Physical blocking may be forced by the NBRRCDS keyword of the OVRDBF parameter.

One disk access is (very roughly) one million times slower than one CPU cycle. Waits for (synchronous) disk I/Os are the most common reason for delays of any commercial application, interactive and batch. These waits also explain why a computer can have many jobs running actively simultaneously and still not use 100% of the CPU.

What are database and non-database I/Os?

Physical I/Os (synchronous and asynchronous) are also divided into database (DB) and non-database (NBD) I/Os. Database I/Os read or write IBM DB2 index and data. Transfers from or to a disk of all other object types (programs, data areas, queues, job descriptions, etc.) are counted as NDB I/O – this includes I/Os to the IFS.

A high non-database I/O rate could be a sign of insufficient main memory resulting in storage management often having to page objects out to disk. Machines with ample storage only need to load e.g. frequently used programs once.

What are logical I/Os?

A logical I/O transfers one (or more) records between the program work area and the database management file buffer. As such, a logical I/O basically boils down to a move internally in main memory. Although logical I/Os are performed entirely by the CPU and thus are rather fast, it involves a certain overhead, possibly even including use of an index to locate a record to be read. A job having many logical I/Os will thus inevitably always use much CPU time.

A logical I/O is caused by a read or write (or update, delete, etc.) instruction in the program. Such I/Os are never carried out by the program itself – the compiler generates a call to the appropriate operating system I/O routine. A logical I/O may also transfer a block of records. Blocking may be requested by the program automatically or through use of the OVRDBF command, where keyword SEQONLY allows specifying between 1 and 32767 records per block that is transferred between the program buffer and data management per logical I/O.

Logical I/Os may or may not trigger one or more physical I/Os (= access to the disk). If a file is being read sequentially in "arrival sequence", i.e. in the sequence in which the records are stored on the disk, the next record is likely to be within the same page or block of data; or database management may have read the next record ahead of time, or the record may happen to be in main storage because it was used recently by the same or even another job. Therefore, if the next record wanted by the program already has been fetched into main memory, a physical I/O is not required for carrying out the I/O request.

One logical I/O could also trigger several physical I/Os with the worst case probably being the random reading of a joined logical file. When the program wants to read one join file record by key, both the indexes and the data spaces of the physical files behind the join file must be accessed, probably all synchronously. This could result in a noticeable program wait time.

What are Miscellaneous I/Os?

According to the "official" definition from IBM, "Miscellaneous I/Os" contains the count for number of updates, deletes, forced end-of-data, and release operations. However, tests indicate that some below-MI-level data base accesses (typically made by SQL) also are included. **Miscellaneous I/Os are logical I/Os**, and updates and deletes will therefore in most cases immediately or eventually trigger a physical I/O.

Our tests show that FEOD (forced-end-of-data) do not cause any logical I/Os to be reported to GiAPA by the Performance Collector APIs. The same occurs for file positioning (COBOL START, RPG SETLL, etc.).

What are Permanent and Non-Permanent Writes?

To answer this question we must first define permanent and non-permanent objects.

- A permanent object is created within a library. All normal user objects are permanent objects. Any create or update of objects in libraries (including QTEMP) on the LPAR where the job runs will cause one or more writes to one or more permanent objects.
- An example of a temporary object is the Open Data Path (ODP), i.e. the operating system internal object type created when a file is opened, containing control areas, etc. needed by data management for processing a database file.

Permanent writes are writes from main storage to disk of pages belonging to permanent objects.

The performance collector APIs supplies information about the number of "Permanent writes". GiAPA automatically makes the following calculation:

(Total number of physical writes) – (Permanent writes) = "Non-permanent writes"

Non-permanent writes is a GiAPA-coined term.

A good example of writes to non-permanent objects is paging. The operating system assigns work areas on the disks, and writes pages from main storage to these work areas in order to temporarily free up main storage for other purposes.

Please note that writes to DDM-files (Distributed Data Management, i.e. files on other machines or LPARs) are not counted as “permanent writes”. Therefore, within GiAPA they will be included in what we call “Non-permanent writes” despite the data bases accessed on the other machine indeed are considered permanent objects on that machine.

Writes to the IFS (not being in a library under QSYS) are also counted as “Non-permanent”.

A high number of non-permanent writes, i.e. the number of permanent writes is considerably lower than the sum of all physical writes (data base and non-DB, synchronous and asynchronous) may therefore indicate heavy paging - - unless a lot is written to DDM-files or to the IFS. Sort criterion 15 on the GiAPA Job Summary Report shows non-permanent write percentage per job.

What is the function of CFINTnn?

CFINT is the task dispatcher that distributes resources to all other jobs. There is one CFINT task for each processor in the LPAR, and technically CFINT is the commander, running with the highest priority, which is lower than 0. The internal tasks of the Operating System are running with various negative priorities. GiAPA will add up the resources used by CFINT to one total.

What are numeric overflows / exceptions?

If a calculation gets a result larger than what the result field can accommodate (the typical example being a divide by zero), an overflow condition triggers a hardware exception. In other words, the hardware was asked to do something impossible, and is therefore forced to throw in the towel. Decades ago, that would cause an abnormal end of the entire job – and even further back the entire computer would have to be restarted.

Today the operating system can analyze, diagnose, and handle such situations. An MCH escape message is generated (MCH = Machine CHeck) and processed by the software. However, this does cost a bit of CPU time, and exception handling routines may have to be brought in from the disk. Overflows should therefore be avoided and in most cases, they are simply a sign of an error within the program.

The three types of numeric overflows, decimal (= packed), binary, and floating point, are reported by the Performance Collector APIs because they use resources. They are in GiAPA consolidated to one total value. Locating the cause of overflows could be a challenge - and is beyond the scope of this manual. Please contact your GiAPA supplier if you need assistance!

Start using GiAPA

Call the GiAPA Menu simply by using command **GIAPALIB/GIAPA** .

The menu options shown are available for installations with a valid product license code, normally L for “Licensed”, or O for “Operations only” (not covering the automated application performance analysis). An error message will appear if the security codes is invalid.

The normal full GiAPA Menu is shown below:

The GIAPA command can be prompted and accept parameters. If you know the number of the option you want to use, you can skip the GiAPA menu by using the option number as a parameter to the GIAPA command.

If you want to have your GIAPA performance analysis data stored in another library than GIAPALIB (which is the default), you can specify the library name in the lower right corner - or optionally use it as a second parameter to the GIAPA command.

Example: The command: **GIAPALIB/GIAPA 15 PFRDATALIB** will run the GiAPA Menu Option 15 “Job Performance Summary” report based on data stored in PFRDATALIB.

Many of the GiAPA Menu Options can also be run using CL-commands. The CL command names consist of “GIAPA” followed by the Menu Option number and a zero, e.g. the command GIAPA110 is the command corresponding to menu Option 11 (start performance data collection).

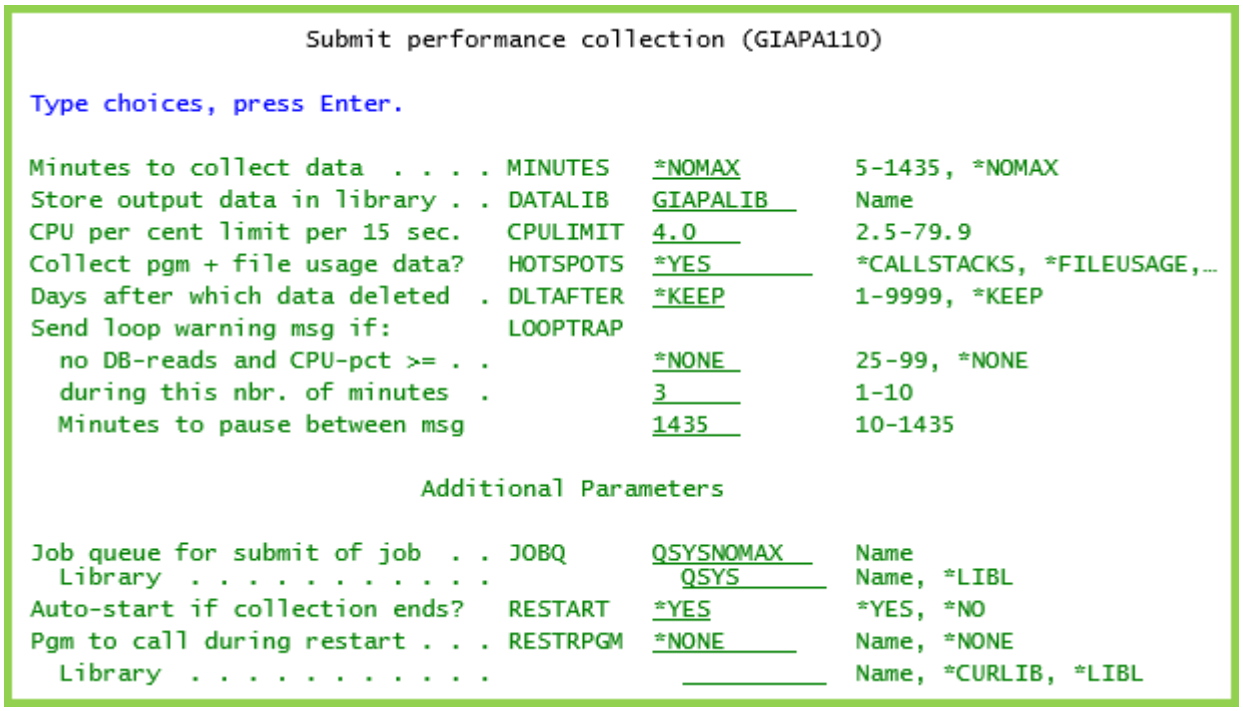
Data Library: Many prefer to have the data collected by GiAPA and the analyzed results stored in another library than GIAPALIB, which by default is suggested as library name on the above shown menu. However, by creating data area GIAPALIB/DTALIBNAM TYPE(*CHAR) LEN(10) specifying another library name for the VALUE keyword, the specified name will be used instead of GIAPALIB as data library.

GiAPA Menu Option 11: Submit performance data collection

This option uses the command GIAPA110. It may also be used directly within a CL program, e.g. in connection with a job scheduling function to start GIAPA data collection every day at a set time. Most GiAPA customers include command GIAPA110 in the QSTRUP program.

During a GiAPA data collection, job GIAPAPFCOL will collect and process the interval data every 15 seconds, and job GIAPAHOTSP will collect HotSpot data. If many HotSpots are found, additional GIAPAHOTSP jobs are submitted to assist with the collection of HotSpot data. Unless *NO is specified for the RESTART keyword, job GIAPARESTR will initiate restarts of GIAPA in case the data collection is ended. If job accounting is active, the job GIAPAACGJR is started simultaneously.

The low CPU usage for the GiAPA performance data collection (normally less than 1/10 % of one CPU) represents the total time used by all these jobs. A higher CPU usage may be seen if the HotSpot data collection limit for keyword CPULIMIT is set low enough that an unusually high number of HotSpots are generated, or if the GiAPA installation parameters are modified to request HotSpots for many threads with a job



MINUTES specifies the timeframe for collection of performance data by GIAPA. An active data collection may be terminated before this specified time by using menu Option 13. The maximum allowed value of 1435 minutes is 5 minutes short of 24 hours. Specifying *NOMAX will cause data collection to continue running until explicitly stopped, except that it will be terminated and restarted automatically at midnight.

DATALIB specifies the library you want to use for storing raw performance data.

CPULIMIT specifies a CPU limit in percent for a job. Every time a user job exceeds this limit for a 15 seconds data collection interval, the HotSpot data collection is initiated for that particular job, unless the HOTSPOTS keyword below contains *NO.

HOTSPOTS is used to indicate whether GiAPA should collect additional call stack and file usage data every time a job exceeds the above specified CPULIMIT in a 15 seconds data collection interval. The default value is *YES, causing full collection of HotSpot data.

There may be cases where collection of the HotSpot call stacks and file data is not required, e.g. if data only are used for showing trends of resource usage, and not for analyzing jobs to locate optimization potential. Therefore, more options are available:

- Specify *FILEUSAGE if only file usage data is wanted,
- Specify *CALLSTACKS if only call stacks should be retrieved during HotSpots.
- Specify *NO to suppress retrieval of both call stack and file data. *NO will provide the same result as specifying 99.9 for the CPULIMIT keyword.
- A special value of *DATATRACE used to document any input data errors to IBM should only be specified after agreement with iPerformance.

DLTAFTER should be employed if you want GiAPA to automatically delete the raw performance data collected after a given number of days. Most GiAPA users let GiAPA collect data every day to ensure that performance details are available should a problem occur. Therefore, this option specifying how many days the data should be retained on the system may be convenient.

The actual delete occurs when Option 11 is used, i.e. an automatic check for any expired data is only performed when new data collection is started. The delete is based on the member text of the members in the file GIAPA112P1 within the used DATALIB. If the keyword DLTAFTER has been used to specify a number of days, the member text field positions 36-50 contain (ExpDate YYDDD) with YYDDD indicating the expiration date in Julian data format. You may use GiAPA Menu Option 81 to enter, change or remove a delete date for existing member(s).

LOOPTRAP offers the possibility of asking GiAPA to send a message to QSYSOPR if a job appears to loop. Given that GiAPA data collection receives information about all jobs every 15 seconds the data needed for detecting jobs looping is available. The LOOPTRAP keyword requires 3 parameters, referred to below as PARM1 (percent CPU equal to or greater than), PARM2 (no. of minutes) and PARM3 (no. of minutes to pause between messages), respectively. The default value of *NONE in PARM1 causes the loop trap function to be inactive. When activated, LOOPTRAP operates in the following manner:

If a job is using CPU resources equal to or above PARM1 % within PARM2 minutes without using any data base reads or communication gets during this period, an informational message **GIA0060** with severity level 60 is sent to the QSYSOPR message queue. The message text is “GiAPA loop trap warning for job JOBNAME/USER/JOBNBR”. If the job continues looping, a new similar message will not be sent to QSYSOPR before PARM3 minutes.

The loop trap function will be reset (= restart from the beginning) for a job if:

- the job uses less than PARM1 % CPU during a 15 seconds data collection interval,
- any physical data base read is used by the job in an interval, or
- any communication get is used by the job in an interval.

Despite this command being able to trap most looping jobs, it may unfortunately also send messages for jobs not looping. If a job processes only data received from a data queue, and also delivers the results via a data queue, then this job will not use any I/Os. Many such transactions could cause the job to exceed the loop trap limits, although the job is performing as designed. Another case could be if all data required by a job is preloaded into memory (e.g. by command SETOBJACC) before the job starts. This type of jobs could exceed loop trap limits without looping given that all the data would be obtained from main memory and no reads from the disks would be needed.

In order to avoid such jobs being reported, GiAPA Loop Trap facilitates an exception list option for job names that never should be reported as looping. This job name exception list can be maintained from the GiAPA Menu Option 74.

JOBQ is used for the job queue name and library for submitting the GiAPA data collection job(s). The shown default value QSYSNOMAX is an operating system internal job queue used for jobs running in subsystem QSYSWRK. It is well suited for the GiAPA performance data collection, which uses very little resources and must run with high priority. It is recommended not to change this default value although any job queue could be selected.

RESTART The IBM Performance collector API called by GiAPA is occasionally experiencing an internal error thereby unfortunately terminating the collection of job performance data. On some servers this error never occurs whereas others have experienced several within one day. The default value is therefore *YES. The error is described in more details in the “Error recovery” section of this manual.

RESTRPGM When GiAPA restarts data collection after an IBM API error, the operating system jobs for collecting this data are terminated immediately by GIAPA prior to restart. If other data collections are running simultaneously (e.g. collecting disk performance information) these jobs will also be affected and must also be restarted. To accommodate this issue, the user must specify the program and library name of a program restarting such additional collection(s). Please note that GiAPA will not check whether the call to the program was successful or not.

Removing *MGTCOL objects

GiAPA calls the IBM Performance Collector APIs to obtain the raw performance data every 15 seconds. When active, these APIs automatically store performance data in objects type *MGTCOL, which might get somewhat voluminous. You may use iSeries Navigator (select “Configuration and Services”) or CL-command CFGPFRCOL to specify how long these objects should be retained on the system (= activate automatic deletion).

Maximum number of members in GiAPA files (default value is 499)

Note: The GiAPA files that contain raw collected performance data, and the files with the expanded and analyzed data, are shipped with MAXMBRS(499). This means that they hold up to 499 members.

Each new or re-started data collection will generate a new member. To avoid run time error message CPF7306 “Member not added” you can periodically either consolidate members (e.g. into one member per week), remove old members, or enter or change an expiration date from GiAPA Menu Option 81.

Each performance data expansion and analysis (Option 14) will create a new member in each of the involved, corresponding database files. Old members containing expanded and analyzed data can be deleted or have the expansion date changed from GiAPA Menu Option 82.

In both cases the recommended and more practical solution is to specify how many days you want the members to be retained on the system. Both the data collection (Menu Option 11 or command GIAPA110) and the expansion and analysis run (GiAPA Menu Option 14 or command GIAPA140) have a “Delete after” parameter used to define a set number of days, after which GiAPA automatically will delete the member(s). If you back up GIAPALIB every month one recommended option is to specify DLTAFTER(35).

If you insist on storing old data online it is of course also a possibility to use command CHGPF to increase the MAXMBRS parameter for the file(s) reaching the default limit of 499.

GiAPA Menu Option 12: Submit HotSpot Watch of Selected Job

As described above, GiAPA can automatically capture additional data for a job at HotSpots triggered when a job exceeds the CPU usage limit within a 15 seconds' interval. But HotSpot data collected every 15 seconds may not be enough to pinpoint all problems within interactive jobs since a transaction might take "only" four seconds. In addition, jobs delayed due to any wait will not cause HotSpots to be generated.

If more frequent HotSpot collections are needed, use GiAPA Menu Option 12. The normal HotSpot function is used to handle this option, i.e. ensure that a GiAPA data collection specifying HOTSPOTS(*YES) is active.

The Watch Job option collects HotSpot information with the user specified intervals, independent of whether the job has used resources. As usual the GiAPA Option 14 must be used to expand and analyze the data, which subsequently is available through the normal GiAPA reporting options.

```
                    Start Job Watch via HotSpots (GIAPA120)

Type choices, press Enter.

Name of job to be monitored  . . JOBID          __________     Name
  User name  . . . . . . . . . .                  *ALL        Name, *ALL
  Job number . . . . . . . . . .                  *ALL        Char. value, *ALL
Collection duration  (minutes)   COLLTIME      1             1-99
Collection intervals (seconds)   COLLINTVAL    1             0.1- 60.0
Max waittime if job not active   WAITMINUTE    60            1-999
Sec.s between check if active  . CHECKEVERY    5             1-300
```

JOBID specifies which job(s) to monitor more closely.
If the user name and job number, or the job number, is not known, specify *ALL This will cause GiAPA to wait for the first job to appear with the specified job name (and user, if specified).

COLLTIME specifies the number of minutes to monitor the job(s).

COLLINTVAL specifies in seconds down to one decimal how often the HotSpot data should be collected. HotSpot collection with extremely short intervals (fractions of a second) should not be allowed to run for very long.

WAITMAXMIN "Wait maximum minutes" indicates how long the GIAPAWATCH job should wait for the start of the job to be monitored.

CHECKEVERY specifies how many seconds the waiting watch job is delayed between each time it checks if the monitored job started.

GiAPA Menu Option 13: End Performance Data Collection

Use command GIAPALIB/GIAPA130 to terminate a GiAPA data collection, either by typing it from a command line or by selecting GiAPA Menu Option 13.

```
                    Terminate GIAPA Collection (GIAPA130)

Type choices, press Enter.

Stop GiAPA data collection?      TERMINATE      N             Y, N

                           Additional Parameters

Also stop auto-restart if any?   STOPRESTRT     Y             Y, N
Job queue for submit of job  . . JOBQ           QSYSNOMAX
  Library  . . . . . . . . . . .                  QSYS
```

TERMINATE specifies whether a running GiAPA data collection (Interval data and - if active - also HotSpot data) should be terminated. Specify Y for *YES.

Use F10=Additional parameters from the initial command prompt to get the next keywords:

STOPRESTR specifies if the automatic restart of GiAPA in case of termination before the end of the scheduled number of minutes should be stopped in order to prevent an automatic restart. The default value of STOPRESTR(Y) will cause job GIAPARESTR to terminate.

If RESTART(*YES) was specified when the data collection was started and command GIAPA130 specifies STOPRESTR(N), then GiAPA restarts automatically.

Please refrain from terminating the data collection using an ENDJOB command, or by closing down the subsystems in connection with an IPL. If GiAPA jobs terminate abnormally, at least some of the collected data will be lost, and in the worst case scenario cause the data expansion and analysis to fail.

When adding command GIAPA130 to e.g. a system close down routine, you should include a DLYJOB (Delay job) command to allow GiAPA to receive all data correctly. This could take up to four minutes due to e.g. a collection of a call stack for a resource consuming job being delayed while awaiting a time-out in the IBM API attempting to retrieve a call stack that is unavailable.

GiAPA can expand and analyze data from an ongoing collection. However, very small jobs having a total elapsed run time of e.g. 5 seconds will probably not be “seen” by GiAPA provided that data only is collected every 15 seconds. The resource usage for such tiny jobs are fetched from the history log and from job accounting (if active) in connection with data collection termination. To ensure that also all such tiny jobs are included in the data, GiAPA must be terminated. Command `GIAPALIB/GIAPA130 TERMINATE(Y) STOPRESTRT(N)` will cause the data collection to end normally and include all data. A new collection is restarted immediately provided that the command GIAPA110 (used to start GiAPA) specified RESTART(*YES).

“Current User” information for jobs like QZDASOINIT also origin from the history log. Therefore, the result of an analysis based on an active (not terminated) data collection cannot be used as input for the “Current User” report (GiAPA Menu Option 24).

GiAPA Menu Option 14: Expand Collected Performance Data

To limit the use of resources during performance data collection to an absolute minimum, GiAPA stores the binary data collected in a compressed form. The collected data must therefore be expanded and analyzed before it can be used for reporting.

Every GiAPA data collection creates a new member for the interval data. The expansion and analysis can run on one member only, or consolidate members from a number of collections into one output set.

The result of an expansion and analysis is several identically named members in the database files used as input for all the reporting. All these files have names starting with GIAPA14 and their file definitions can be found in the source file GIAPA_QDDS since they can serve as input for e.g. queries.

When using Option 14 the following panel appears:

```
GiAPA (c) by    Submit job to expand collected performance data        14/05/20
iPerformance                                                          21:14:20
                          Output member name: __________  YYMMDD Select hhmm
 1=Include in expansion   Keep detailrecords? *NO         000101  From  0000
 5=Display mbr statistics Delete output after 9999 days   991231   To   2359
                          Optional text:  Enter an output member description
Opt  Member      Date      Text
 _   PF04260005  140426    Pfr.data from 140426 at 000500     (ExpDate 14166)
 _   PF04250005  140425    Pfr.data from 140425 at 000500     (ExpDate 14165)
 _   PF04240005  140424    Pfr.data from 140424 at 000500     (ExpDate 14164)
 _   PF04230005  140423    Pfr.data from 140423 at 000500     (ExpDate 14163)
 _   PF04220840  140422    Pfr.data from 140422 at 084003     (ExpDate 14162)
 _   PF04210915  140421    Pfr.data from 140421 at 091505
 _   PF04210005  140421    Pfr.data from 140421 at 000500
 _   PF04200005  140420    Pfr.data from 140420 at 000500
 _   PF04190005  140419    Pfr.data from 140419 at 000500
 _   PF04180005  140418    Pfr.data from 140418 at 000500
 _   PF04171455  140417    Pfr.data from 140417 at 145521
 _   PF04170005  140417    Pfr.data from 140417 at 000500                  +

F2=Cmd Line    F3=Exit
```

Output member name (Ignored for Selection 5): Select an appropriate name for the performance data collection you want to expand and analyze. If you want to consolidate all data from the week starting May 16th, you could e.g. use WEEKMAY16 as output member name.

Please note: If a data expansion is re-run using the same output member name, the new data will simply replace the existing data.

Keep detail records?
In the first step of the data expansion one record is generated whenever a job or task shows use of any resource within a 15 second's interval. This results in a large data volume that is often more detailed than necessary, since it also contains all the records showing "normal" – and from a performance point of view, adequate – use of resources.

During the analysis of the data collected, GiAPA will accumulate resources used by jobs not experiencing or causing performance problems into job type summaries, thus only keeping potential problem jobs for further analysis.

If the “Keep all detailrecs?” parameter is *NO, the data base file members containing all the individual detail records are deleted at the end of the data expansion run.

In some cases, all details may be necessary to carry out special analyses, typically when reports or graphics from menu Option 16 are required, or for capacity planning purposes. Specifying *YES in these instances simply results in not deleting the detailed data members. Be aware that the detailed data probably will occupy around 20-30 times more space than the selected data stored when option *NO is used.

Delete output after nn days should be used if you want GiAPA to automatically delete the expanded results after a given number of days. The actual delete is executed at the end of expansion runs. The delete is based on the text of the members in the files named GIAPA14*P* within the used DATALIB. If the option to delete automatically was used, you will find the member text field positions 36-50 contain (ExpDate YYDDD), where YYDDD is the expiration date in Julian data format. You may use GiAPA Menu Option 82 to enter, change or remove a delete date for existing member(s).

Optional text: Specify details describing the included data, e.g. “Pfr. Data, week of May 16th.”

Select From / To YYMMDD hhmm: If all date and time default values are left unchanged then the expansion will include all the data from the members selected.

If the date(s) are altered compared to the default values in the “from” and/or “to” field:

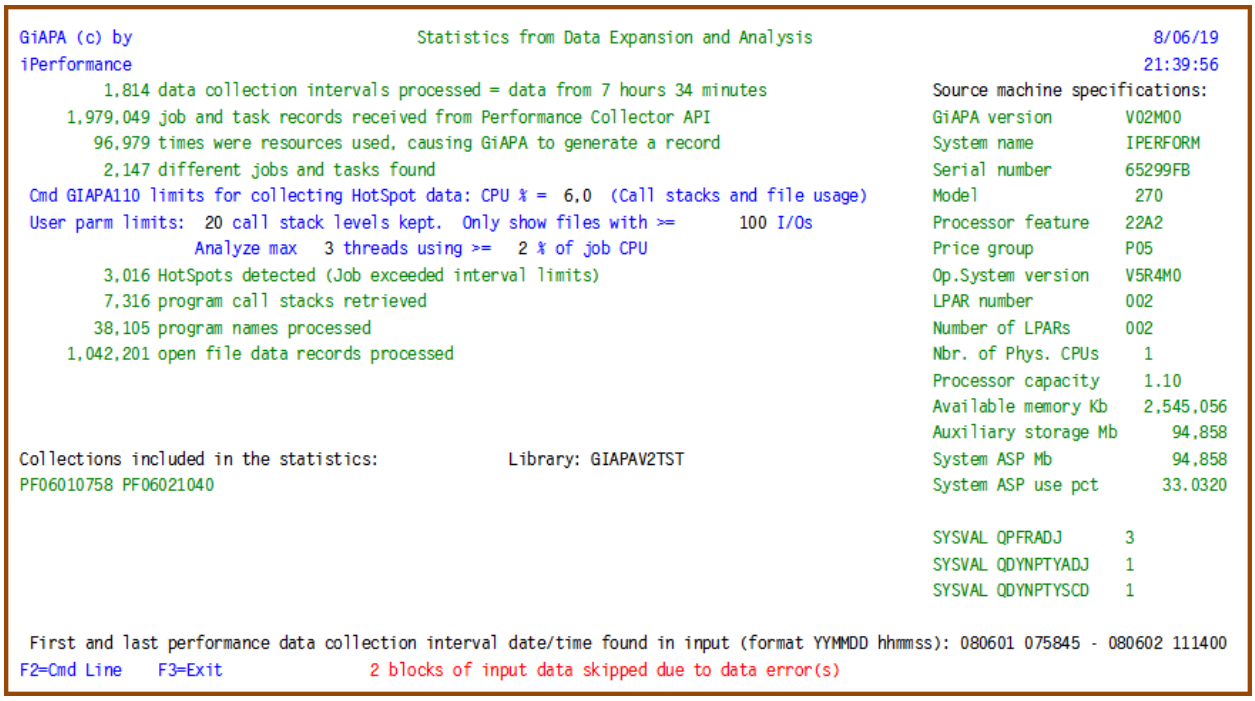
- the from date and time will be concatenated, and any data collected before this start date+time will not be included in the expansion, and
- the to date and time will be concatenated, and any data collected after this end date+time will be disregarded by the expansion.

If from and to dates are left unchanged, but a non-default value is specified for “from” time and/or “to” time, then the expansion will only include data encompassing these from and to points of time. In this scenario, data from several members / several dates could be included.

Selection 1: In the selection column of the subfile displaying the file members containing raw GiAPA performance data type the digit 1 in front of the members you want to include in the expansion and analysis run.

Pressing ENTER after having entered the above specifications will submit batch job GIAPAEXPAN to expand and analyze the performance data selected. The run time of the job is optimized by using blocked sequential process, which makes the job predominantly CPU bound. Therefore, the job will be submitted to run with priority 59.

Selection 5: Specify selection 5 for one or more members if you want to verify the validity of the data collected, and see the statistics for the data contained in the selected member(s). Selection 5 is fast and will run interactively. An example of the resulting display is shown below.



If the data collection was interrupted by an ENDJOB command instead of using the GiAPA Menu Option 13, the last blocks of input data will be incomplete. The expansion job will skip incomplete blocks of data, processing as much as possible of the available data.

To verify if any data is missing due to abnormal termination of data collection(s), use selection 5=Display statistics. Selection 5 is also available after the expansion when selecting a resulting member to view from the “F4=Prompt for member name” from the GiAPA Menu Option 15. If any data was skipped, a message in red will appear on the last line, indicating the number of blocks skipped. One block of data could contain 50 – 800 records, depending on the type of data. The exact number of lost records cannot be accounted.

Exit Program GIAPA_UE1 (User Key fields for Graphical Reports)

The data expansion and analysis run also adds a member to file GIAPA144P3, used as input for graphical reports (see menu Option 26). Before each record is written during the formation of the file member an attempt will be made to call the user exit program GIAPA_UE1 in GIAPALIB. This allows users to supply two additional sort key fields (e.g. application name, department, geographical code, etc.) to be used within the graphical reports. Please find further details in source file GIAPALIB/GIAPAEXAMP in the comments within the source code of program GIAPA_UE1.

If reporting by job accounting codes are wanted, please refer to the comments in the source code of program GIAPA_UE1F (also supplied in GIAPALIB/GIAPAEXAMP). This program will initiate USERFIELD1 with the job accounting code defined in the user profile for the Job User (or Current User, when available).

Command GIAPA140: Expansion of Pfr. Data in Scheduled Batch Job

Expansion and analysis of performance data may be scheduled to run e.g. every night or week by using the command GIAPA140, which automatically includes data collected the previous nn days. To ensure correct selection of data to include in the run, command GIAPA110 must use the default value *NOMAX for the MINUTES parameter when data collection is started.

The command can as an option send a summary data file to another LPAR using FTP, allowing e.g. usage data for all LPARS on an entire physical machine to be gathered on the "master LPAR". This can be depicted graphically to show total resource usage. Similarly, this feature opens for showing total resource usage per application in cases where the applications run on different physical or logical machines. An example of multi-LPAR graphics is included below the sections explaining the commands GIAPA140, GIAPA040, and GIAPA141.

```
                Expand and Analyze GiAPA Data (GIAPA140)

Type choices, press Enter.

Input and output library . . . . DATALIB    GIAPALIB      Name
Expand data for last NbrOfDays   NBROFDAYS  1             1-99
Mbr.name prefix (any letter) . . MBRPREFIX  E             Name
Level of details to be kept  . . DETAILLVL  *STANDARD     *ALL, *NOHOTSPOT, GRAPHDATA,
                                                            *STANDARD
Days until result is deleted . . DLTAFTER   *KEEP         1-9999, *KEEP
FTP graphics input file to . . . RMTSYSNAME *NONE         Name, *NONE
Remote system logon user name  . RMTUSRID   *NONE         Name, *NONE
Summary level for data to FTP  . SUMLEVEL   *HOURTYP      *HOUR, *HOURTYP, HOURTYPJOB,
                                                            *HOURTYPUSR, *HOURTYPJOBUSER
Minutes to retry failing FTP . . WAITFORFTP 150           1-999
FTP Port number  . . . . . . . . PORT       *DFT          1-65535, *DFT, *SECURE
Secure connection  . . . . . . . SECCNN     *DFT          *DFT, *NONE, *SSL, *IMPLICIT,
                                                            *KERBEROS
```

NBROFDAYS specifies the number of days of data to include in the expansion. The count always starts with the previous day, e.g: if the command GIAPA140 is scheduled to run on Wednesday, January 6th, 2016 using NBROFDAYS(3) and data was collected every working day (but none over the weekend Saturday/Sunday), then data from the 3 dates prior to the current (= 6th) would be selected, i.e., January 3rd, 4th, and 5th. Since no data was collected on Sunday 3rd, only data from Monday 4th and Tuesday 5th is selected.

MBRPREFIX. Each expansion of GiAPA data is given a member name, as described above in "Output mbr" under GiAPA Menu Option 14. For expansion results from command GIAPA140, the member name is generated, and will appear in the format EYYYYMMMDD, where:

- E is a user selected member name prefix letter (defaults to E for Expansion, but could also be D for daily or W for weekly)
- YYYY designates the year (e.g. 2016)
- MMM is the month abbreviated to three letters (e.g. APR for April)
- DD is the day of the month (e.g. 08 or 23).

Please recall that any existing expansion result member having the same name will be deleted.

DETAILLVL defines the level of details to be kept from the expansion and analysis. It can be seen as an extended version of the "Keep detailrecords?" specification on the panel appearing when GiAPA Menu Option 14 is selected.

- ***STANDARD** has the same function as specifying *NO in "Keep detailrecords?": GiAPA will accumulate resources used by jobs not having or causing performance problems into job type summaries. In contrast, potential problem jobs are stored individually.
- ***ALL** corresponds to entering *YES in "Keep detailrecords?": The data base file members containing all the individual detail records are stored at the end of the data expansion run.

- ***NOHOTSPOT** will not keep the call stack or file usage information, but will store the same level of performance data as *STANDARD. *NOHOTSPOT will reduce the disk space needed, but the results will not suffice for application performance analysis.
- ***GRAPHDATA** will only keep the summary data written to file GIAPA144P3. This file is primarily intended as input for various user defined graphics, but the file could also be used for queries. The file definition can be seen in source file GIAPA_QDDS in GIAPALIB. Note that the file contains two user fields, which may be initialized during expansion using the earlier mentioned user exit program GIAPA_UE1.

DLTAFTER should be used if you want GiAPA to automatically delete the expanded results after a given number of days. The actual delete occurs at the end of expansion runs. The delete is based on the text of the members in the files named GIAPA14*P* within the used DATALIB. If the option to delete automatically was used, the member text field positions 36-50 contains (ExpDate YYDDD), where YYDDD is the expiration date in Julian data format. To change or remove a delete date please use GiAPA Menu Option 82.

Please note: The remaining parameters should only be used if a summary graphics input file is sent using FTP to another machine / LPAR. Before using this option, please read about commands GIAPA040 and GIAPA141 below!

RMTSYSNAME: If you want a file of summary performance data (primarily intended to serve as input for user defined graphics) to be sent to another LPAR for consolidation this is the place to specify the “Host name” of the other machine as defined in the “Work with TCP/IP Host Table Entries”. To see this table use command CFGTCP and select Option 10. If the other machine/LPAR is not defined yet, it must be added using command ADDTCPHTE.

If this LPAR is the “master LPAR” that receives data from other LPARs, and you want data from this LPAR to be included in the consolidated graphics you must specify the system name of this LPAR. This will cause the resulting data to be copied to file GIAPA141P1 used to consolidate data from different LPARs.

RMTUSRID is the user profile name on the RMTSYSNAME machine used to logon in preparation for the FTP transfer. You will be prompted for a password.

SUMLEVEL defines the level of details transferred to the resulting data , i.e. it defines the control break key fields used to summarize. The record layout of file GIAPA140P1 used can be seen in source file GIAPALIB/GIAPA_QDDS. The key fields are USERFIELD1, USERFIELD2, SRLNBR, SYSNAME, JOBTYPE, DATEYMD and HOUR.

The fields containing machine serial number, system name, date and hour are always used as summary control break fields. Job type and the two user fields can be filled in or remain blank.

The two user fields may have been initiated during expansion by user exit program GIAPA_UE1. Please refer to the source code comments of example program GIAPA_UE1 in source file GIAPALIB/GIAPAEXAMP. Irrespective of whether the user fields have or have not been initialized by the user exit program, the contents may be overwritten depending on the value selected for the SUMLEVEL keyword.

- **SUMLEVEL(*HOUR)** will blank out JOBTYPE. If the user fields were not initialized through the user exit program, they will contain blanks.
- **SUMLEVEL(*HOURTYP)** will keep JOBTYPE. The user fields will not be altered indicating that if they are not initialized, they will contain blanks.
- **SUMLEVEL(*HOURTYPJOB)** will keep JOBTYPE and any value the user may have inserted in USERFIELD2 through the user exit program. Job name will be inserted in USERFIELD1.

- **SUMLEVEL(*HOURTYPUSR)** will keep JOBTYPE and any value the user may have inserted in USERFIELD1 through the user exit program. User name will be inserted in USERFIELD2.
- **SUMLEVEL(*HOURTYPJOBUSR)** will keep JOBTYPE. Job name will be inserted in USERFIELD1, and user name will be inserted in USERFIELD2.

If SUMLEVEL(HOUR) is specified and the two user fields are empty (= they have not been initialized by user exit program GIAPA_UE1) then:

- the "Maximum CPU percent used in LPAR" (field MAXCPUPCT) is calculated as the average of the upper quartile of CPU percentages found within all the 15 seconds' performance data collection intervals of each hour.
- the "Upper Quartile CPU Seconds Average (field UPPQUARCPU) is calculated as the average of the upper quartile of CPU seconds used within each 15 seconds' performance data collection intervals of each hour.
  The field UPPQUARCPU will be zero if SUMLEVEL was not "HOUR".

This particular way of calculating the SUMLEVEL(HOUR) results in a more correct overview of the maximum CPU usage of all LPARS on an entire serial number; the purpose being to show if free capacity is available. Using the highest value which might originate from a single 15 seconds collection interval would probably not provide a correct impression of the spare capacity available.

WAITFORFTP defines the total wait/retry time used by the sending LPAR, should the receiving LPAR not answer a PING immediately when the data is ready to be transferred. The program will PING the receiver, and only start FTP transmission if the PING was successful. The PING command will be used a maximum of six times. Example: if the WAITFORFTP value is 25 minutes, there will be 5 minutes between each attempted PING. Since this transmission in general will be scheduled to run at night time, this wait time option is included to account for periods when the receiving LPAR e.g. might be closed down for a while during backup.

PORT The port number to use for connecting to the FTP server.

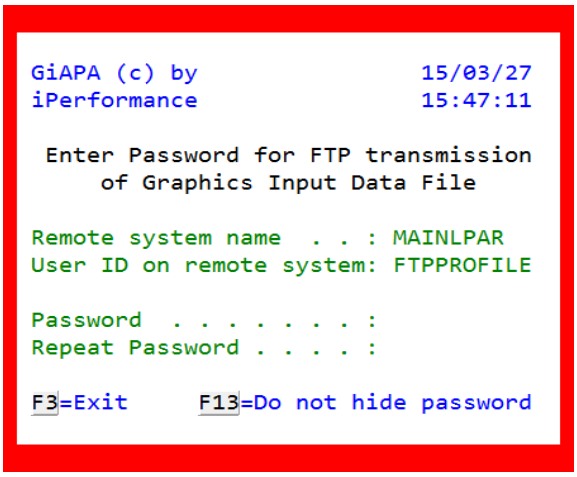
SECCNN specifies the type of security mechanism to be used for protecting information transferred on the FTP control connection. Fort details, see the FTP CL-command.

Prompt for Password

This password prompt screen will appear when command GIAPA140 is entered if the remote system name is another LPAR.

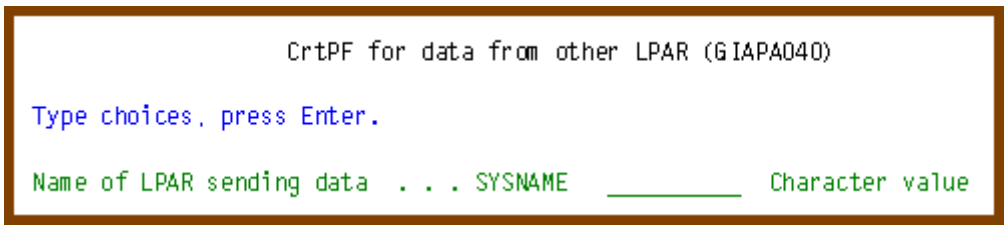
In connection with submitting a job to run, it may appear a second time.

The password is kept encrypted internally within GiAPA and is used for the FTP transfers.



Command GIAPA040: Create file to receive data from other LPARs

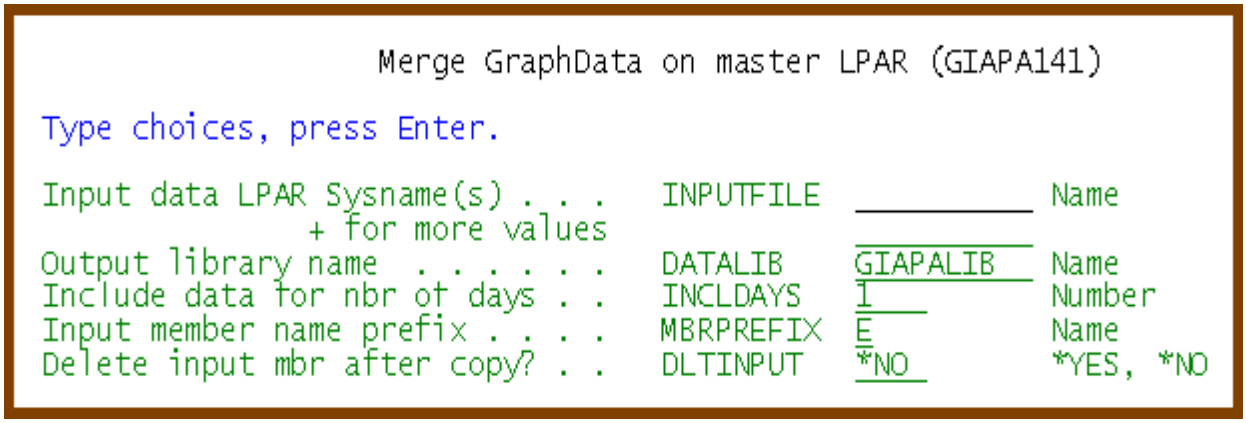
Before command GIAPA140 is used to send results to a “master LPAR”, the receiving file must be created within GIAPALIB on the LPAR to which the data will be sent. If a physical machine has defined four LPARs, and you want to consolidate data from all four on one of these in order to generate graphics showing the total picture, then command GIAPA040 must be used on the “master LPAR” once for each of the three other LPARs.



Command GIAPA040 creates the receiving file GIAPA141P1 within GIAPALIB - the file name must be the system name of the LPAR from which the data is sent. The record format is the same as in file GIAPA140P1 (see source code in GIAPALIB/GIAPA_QDDS).

Command GIAPA141: Consolidate data sent from different LPARS

After data has been received on the “master LPAR” from other LPARs, the data must be consolidated to one file (file name is GIAPALIB/GIAPA141P1) before graphics showing results from all the LPARs can be generated.



INPUTFILE name is the system name of the sending LPARs.

DATALIB will normally be GIAPALIB, but another library name can also be specified.

INCLDAYS is the number of days (run date included) to select, should the input file contain data for a longer period. Example: If INCLDAYS(3) is specified for GIAPA141 scheduled to run on May 26th, then data received on May 24th, 25th, and 26th will be selected.

Note: Since data received on a given date contains information before that date, the above example would probably include data collected on May 23rd, 24th, and 25th.
If in doubt, specify an extra day to be safe, because the program will check that records in the receiving file are not duplicated.

MBRPREFIX is the MBRPREFIX specified in command GIAPA140 – see above. This allows data with different summarization levels to be received and handled separately. We recommend using a different **DATALIB** library for each prefix used.

DLTINPUT allows GIAPA to clean the sending member from the input file after the records were successfully copied.

Example of Chart Consolidating Data from more LPARs

The column chart below shows CPU usage for three LPARs on one physical server. The purpose is to provide an overview of the total CPU usage per hour for the entire physical machine, which had four processors.

The three LPARs named ALFA, BETA, and GAMMA had respectively 0.65, 1.25, and 0.80 processors assigned. Within a 15 seconds' collection interval they have a total of 15 * 4 = 60 CPU seconds available.

The chart was produced using the following steps: Collected GiAPA performance data from a given day was expanded using command GIAPA140 on all three LPARs. Specifying RMTSYSNAME(ALFA) causes the results from all three LPARs to be sent to the "master" LPAR ALPHA. Generation of upper quartile statistics for CPU usage was requested by specifying SUMLEVEL(HOUR).

```
GIAPA140 NBROFDAYS(5) MBRPREFIX(C) DLTAFTER(35) RMTSYSNAME(ALFA) +
            RMTUSRID(KAARE) SUMLEVEL(*HOUR) WAITFORFTP(12)
```

Command GIAPA141 was then used on the ALPHA LPAR to copy the data received from the two other LPARs into the common file of data from different LPARS.

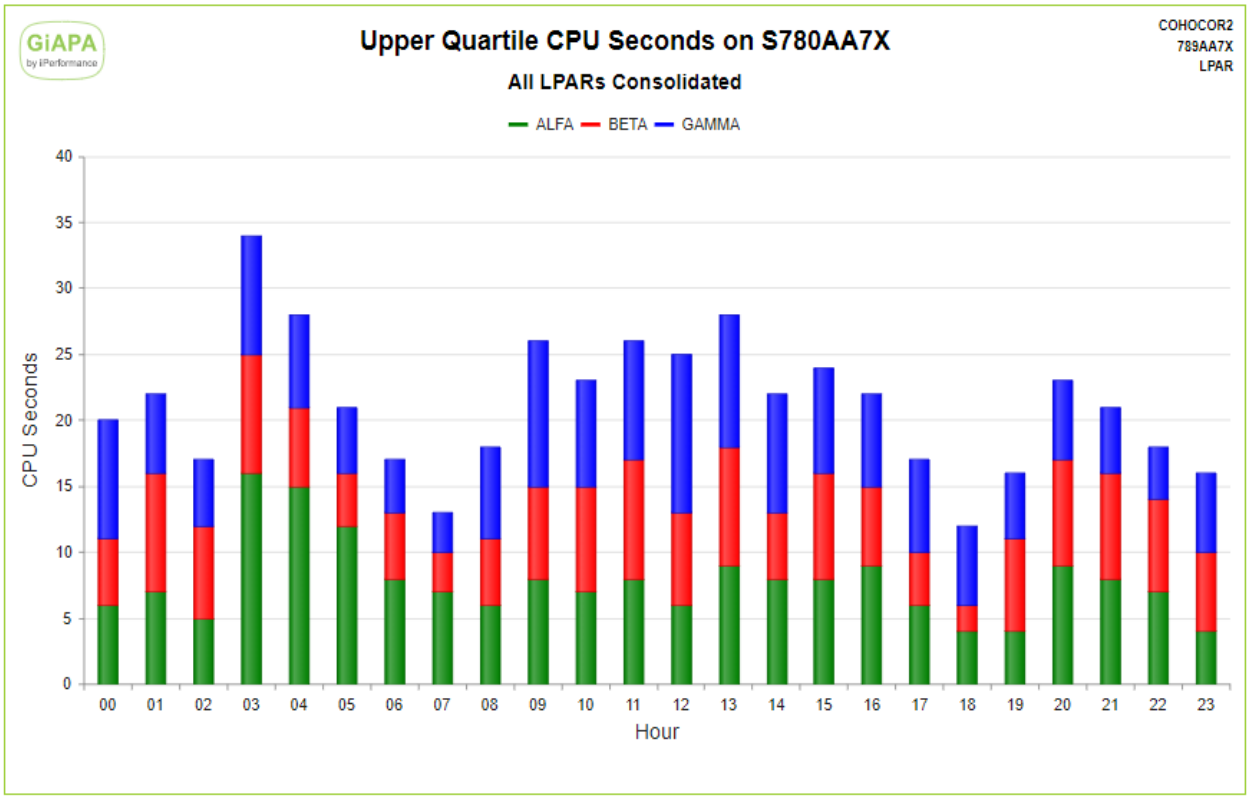
```
GIAPA141 INPUTFILE(BETA GAMMA) MBRPREFIX(C)
```

From Option 26 on the GiAPA Menu we selected the special input member MULTI_LPAR. The graph definition selected the date wanted, specified HOUR as key field 1 and SYSNAME (= LPAR) as key field 2. "Upp.Quart CPU Sec Avg" is the wanted data field, here selected by specifying an "S" to request a stacked chart.

The graph shows the average of the upper quartile of CPU seconds used within the 240 data collection intervals each hour. The LPARs are obviously defined as uncapped – e.g. the 11 seconds shown for the green ALPHA LPAR exceeds the assigned 0.65 processor (65 pct. of a 15 seconds' interval is only 9.75 seconds).

As the graph clearly shows, this machine has spare capacity with the highest average value found for a 15-seconds' interval being 34 seconds used out of the 60 seconds available.

Based on these results it should be safe to add one more LPAR. With 2.70 processors assigned already, the new LPAR could not be given more than 1.30 processor, but defined uncapped it could apparently without any problems use considerably more CPU.



Maximum CPU used per interval would result in a very different picture: most LPARs peak from time to time and typically reach close to 100 % CPU for just a few seconds. The values would therefore be too high to assess if the machine in fact was overburdened, or had spare capacity.

Average CPU time would on the other hand typically indicate that there was considerably more spare capacity. However, if an additional LPAR was added based purely on average values per hour, there would be a real danger of over committing the machine, since peaks lasting several minutes might not be visible. If such peaks occur at the same time on different LPARs, CPU resources might not be adequate.

Using three commands you can schedule this to run in nighttime batch:

- Expansion of the performance data on each LPAR including optional transfer of the results to a “master” LPAR (command GIAPA140),
- Consolidation of the data received on the “master” LPAR from different LPARs (command GIAPA141), and
- Creation of the graphics data file to use as input for the generation of the diagrams or charts (command GIAPA050)

This results in the chart being ready to view in the /GIAPA directory in the IFS – and/or the chart may be distributed attached to E-mail(s)!

Generation of a graph usually completes within seconds, after which the result is displayed immediately. Should that not be the case, please see the last page of this manual.
.

GiAPA Menu Option 15: Job Performance Summary Reports

Option 15 opens for many different types of reports, and is the most used function to obtain an overview of performance issues and identifying problem jobs. The additional HotSpot analysis reports (requested by using cursor positioning to select a job on the Job Performance Summary Report followed by hitting an F-key to select report type) provides in most cases the detailed information needed to identify how improvements may be achieved.

Using Option 15 will result in the following panel being displayed:

```
GiAPA (c) by            Selection of Job Summary Report              15-08-20
iPerformance                                                         15:16:23

 Input library GIAPALIB    Member LASTEXPAND (F4=Prompt member list)

                  Report sort criterion  05    Options:   1  Job name
 ------------------------------------------------------+    2  User name
 Sort on total or max per interval? T/M  T           I    3  Job number
 Display, print, or print max     D/P/M  D           I    4  Priority
                                                     I    5  CPU time used
 Record selection:                                   I    6  Average CPU %
  Job      Name, generic*, *ALL, *USRPGM *ALL        I    7  Physical I/Os
  User     Name, generic*, *ALL          *ALL        I    8  Logical I/Os
  Job Type A, B, I, M, R, S, V, W, X, *  *           I    9  Print lines
           or H for jobs having HotSpots             I   10  Transactions
 Single- or Multi-thread or Both? S/M/B  B           I   11  Trans/JobQ time
                                                     I   12  Average Tr.time
 Only show jobs using selected object?   *NO         I   13  Communication
(GiAPA Menu option 18, selection 1-5-6)              I   14  Overflow except.
                                                     I   15  Non-perm.write %
 Maximum number of records to select      300        I   16  Max. pages used

F2=Cmd Line   F3=Exit   F4=Member Prompt   F5=Refresh   F9=Call Stack for job
```

Input library: The name of the library containing the data to be used for input. The name appearing initially will be the library name that was passed from the GiAPA Menu.

Member: Member name of the expanded data used as input. Defaults to the last expanded member. Use F4 to obtain a subfile listing members available, and select by specifying 1 in front of the member to be used. An example of the member selection panel is shown below.

Report sort criterion: Specify one of the 16 sort criteria listed in the right-hand side of the panel. Ascending sort is used for sort criteria 1 – 4; descending sort is used for sort criteria 5 – 16. However, if the next parameter “Sort on total or max …” is set to M then descending sort is used for sort criteria 1 – 4, and ascending sort will be used for criteria 15 and 16.

Sort on total or max per interval? All the Job Performance Summary reports contain two lines of information for each job: The first line shows the total resource use for the job, the second line shows the maximum values found for one interval for the job. For sort criteria 5 – 14 the sort can be either on the value showing the T=Total use of the resource, or on the M=Maximum value for an interval, which is used to show peaks.

Display or print the result? The options will at first look render similar results, displaying the data on the work station. However, there are differences: Option D=Display produces a subfile, where cursor positioning can be used in connection with a function key (F6-F11) to request an additional level of detailed information for a job.

Option P=Print (or M=Print including max-lines) generates a report, which automatically is shown by a DSPSPLF (Display Spooled File) command. The print has the advantage that the normal F16=Find option of the operating system is available.

Record selection: Allows selection on job name, user name, job type, and/or jobs that were seen to use a certain program within a HotSpot. If several selections are used, they will be AND connected (all must be true to select a record).

Job: To limit the selection of jobs, specify a job name or a generic job name.

A special value ***USRPGM** enables a more complex selection of jobs, allowing the name of a user written selection sub-program to be specified in the **User name** field. If *USRPGM is specified in the job name field, the job type field is ignored.

The user sub-program must be stored in GIAPALIB, and will be called once for each input record with job name, user name, and job type as input parameters, and a one-byte yes/no return parameter indicating if the record should be selected. See further documentation in source file GIAPALIB/GIAPAEXAMP member **GIAPANOGRP**, which is an example causing group summaries (*BATCH, *INTERACT, etc.) to be excluded from the report.

User To limit the selection to include only jobs run by a specific (group of) user(s) specify a user name or a generic user name.

If special value *USRPGM is specified in Job name, the User name field must contain the name of a user supplied record selection program.

Job type: To limit the selection to include only jobs of a certain type, specify the job type:

A	Autostart job	M	Subsystem monitor job	V	VLIC task
B	Batch job	R	Spooled reader job	W	Spooled writer job
I	Interactive job	S	System job	X	SCPF system job

A “special value” H for Hotspots is also allowed. It will cause only jobs that have generated HotSpots to be selected. This selection will automatically exclude amongst others group totals (like *BATCH), all system tasks, and jobs being on the HotSpot Exception list (specified on GiAPA Menu Option 75).

Single- or Multithreaded Jobs or Both?

S	Single threaded only	M	Multithreaded only	B	Both

Only show jobs using selected object: To limit the selection to include only jobs that during the HotSpot collection had a given (user)program, procedure, or class name in the last called level of the call stack, specify ***YES.** Before using this option, the object to be used for the selection must have been chosen from GiAPA Menu Option 18, report selection 1, 5, or 6.

Maximum number of records to select: The reports are intended to be exception reports, consequently all sort criteria above 04 show the “worst” cases first, i.e. start with the highest use of resources. Therefore, you normally want to limit the number of lines, and not list all jobs on every report. GiAPA will automatically exclude any jobs having a zero value for the sort criteria selected. This parameter specifies the maximum number of jobs to be included in the report.

F9=Call stack for Job can only be used when selection on (generic) job name is specified. It generates a call stack report similar to the one described when using F9 from the Job Performance Summary report. This option is useful when a number of small jobs running the same application each have too few HotSpots to provide reliable call stack statistics.

Select member for Job Summary Reports

This is an example of the member selection panel, obtained by using F4=Prompt for member name from the “Selection for Job Summary Report” panel:

```
GiAPA (c) by        Select member for Job Summary Reports            07/09/23
iPerformance                                                         09:48:00

 1=Select   5=Display analysis and expansion statistics     Library: GIAPALIB

Opt  Member      Date        Text
 _   AUGUST07    070903      Consolidated performance data for August 2007
 _   SEP15MORN   070915      Performance data from the morning of September 15th
 _   WOCHE3807   070820      GiAPA Daten Woche 38 = 13 - 17 Sep 2007
 _   VECKA3907   070927      Prestanda mätning vecka 39 - 2007
      (Additional lines have been removed for brevity)
```

1=Select will select the member name and return to the “Selection for Job Summary Report” panel.

5=Display Analysis and Expansion Statistics on the above panel will supply statistics from the expansion and analysis run (GiAPA Menu Option 14) that created the member.

Below is an example of this information:

```
GiAPA (c) by              Statistics from Data Expansion and Analysis          8/06/19
iPerformance                                                                  22:17:31
     4,801 data collection intervals processed = data from 20 hours     Source machine specifications:
  26,799,529 job and task records received from Performance Collector API     GiAPA version     V02M00
   2,051,588 showed resource usage --> record generated      17 records rejected   System name      I5KNEHST
      31,918 different jobs and tasks found                      Serial number     443D9CD
Cmd GIAPA110 limits for collecting HotSpot data: CPU % =  5,5  (Call stacks and file usage)   Model          595
User parm limits:  20 call stack levels kept.  Only show files with >=    100 I/Os      Processor feature  7483
          Analyze max  5 threads using >=  2 % of job CPU                  Price group       P50
     35,346 HotSpots detected (Job exceeded interval limits)               Op.System version  V5R4M0
     35,553 program call stacks retrieved                                  LPAR number       005
    355,510 program names processed                                        Number of LPARs   008
  1,521,710 open file data records processed                               Nbr. of Phys. CPUs  28
      1,101 jobs with HotSpots kept                                        Processor capacity  10.75
     70,929 job-id summary records generated                               Available memory Kb  200,663,000
      1,162 job-id summary records kept for analysis                       Auxiliary storage Mb  10,057,203
     70,772 detailed resource usage records kept for GiAPA reporting       System ASP Mb      10,057,203
                                                                           System ASP use pct   67.7319
       69 job name summary records generated
       51 user name summary records generated                              SYSVAL QPFRADJ    2
                                                                           SYSVAL QDYNPTYADJ  1
 Library: GIAPAKNE                                                         SYSVAL QDYNPTYSCD  1
 Member:  V2TESTRUN   Text: Performance analysis of E-Server on June 14th

 First and last performance data collection interval date/time found in input (format YYMMDD hhmmss): 080614 030115 - 080614 230100
F2=Cmd Line   F3=Exit   F21=List rejected rec.    2 blocks of input data skipped due to data error(s)
```

If the statistics do not include all data, and/or the error message in red in the last line appears, then performance data collection(s) was terminated abnormally, causing some data blocks to be incomplete.

The error message in red in line 5 appears if GiAPA rejected performance collector records due to value(s) being suspiciously high. In that case F21 may be used to list the rejected errors. The records from the performance collector APIs contain very rarely values that simply cannot be valid, as shown in the example below: the two jobs shown had a four- or five-digit CPU usage percentage for a given 15-seconds data collection interval.

```
GiAPA (c) by          Performance collector records rejected due to unlikely high value in one or more fields          15-04-13
iPerformance                                Member name selected: APR08_2015                                        20:49:18
YY-MM-DD JobName    JobNbr  CPU-MSec Transact.    Syncr.    Syncr.   Asyncr.   Asyncr.   Logical Misc.I/Os    Perm.W  Comm.Put
hh:mm:ss UserName   Tp  Itv    CPU-% TransTime  DataBase    Non-DB  DataBase    Non-DB       I/Os            PrtLines  Comm.Get
15-04-08 IOSTATSTAS 000365   3984872         0         0     32288         0         0          0         0         0         0
 2:28:45            V   15   26565 %         0         0         0         0         0          0                   0         0

15-04-08 SMPLACLRTA 0004C0    369664         0         0         2         0         0          0         0         0         0
 2:33:30            V   15    2464 %         0         0   1464308         0         0          0                   0         0

15-04-08 SMPLACLRTA 0004C7    369334         0         0         2         0         0          0         0         0         0
```

Job Performance Summary Reports

This is an example of the report showing only the data line for total use of resources. Using F11 will open the subfile and show the second line per job containing the “maximum” data.

```
GiAPA (c) by          Job Performance Summary Sorted by Total CPU Usage          Input=V2TESTDATA        08-06-20
iPerformance          First / last collection interval:  08-06-14 03:01:15 / 08-06-14 23:01:00          16:37:58
JobName   UserName  RunTime  Typ Itvs ActualUsr   CPUtime   %  Prio    Logical    Physical  Trans Tr/JQ-time  Print  Ovfl.
JobNbr HotSp  RunDate from/to Pool Thrd MaxPages  H:MM:SS.s  CPU rity     I/Os       I/Os  actions h:mm:ss   lines  1000s
*BATCH    53781   00:46:09  B   0        5:28:17.7  0.0 00    56,695,258         0     5     2  976,951   311
CMPRCDCNT MIMIXOWN  11:52:44  B  271        2:13:14.0  33.3 25      427,108    1,749,081
RCV_RQS   MIMIXOWN  03:01:15? B  480        1:27:54.6  7.3 25   402,058,468    1,188,143
*CFINT**     10   03:01:15  V   0         53:13.7  0.0 00          0         0
EDITCVAGB DGWADMGB  03:07:30? B  460         46:56.2  3.9 50   224,837,462    1,537,337             35,390   60
BXFSLCUST KREZFAS   21:32:15? B   67         34:59.1  39.4 50   20,597,700    2,187,399
BXFSLCUST KREZFAS   19:36:04  B  112         29:12.0  36.4 50   16,834,559      171,011
*VLIC_TASK 14424    03:01:15  V   0        26:18.8  0.0 00          0         0
ACFSLCUST KREZFAS   19:44:14  B  118         24:54.0  37.2 50   14,105,091      134,889
*INTERACT    600    14:05:15  I   0         23:45.8  0.0 20    23,317,085         0 88,665 4:33:04   135,476   84
          (Additional lines have been removed for brevity)
DNCHJMMX  DNCHJMMX  22:37:45* I  128          11:07.1  0.1 20     6,081,798      27,603    274   3:41          +
F3=Exit    If cursor in subfile: F6=ODPs  F7=File Statistics  F8=File Analysis  F9=Call Stack  F10=Detail   F11=Show/hide max
F2=Cmd Line   F14=Create Graph Data     (Note: F6-F10 displays subfile. F18-F22 displays printed report allowing use of "Find")
```

Limitation for I/O-Counts of multithreaded jobs:

Most of the values shown in this report originate from the IBM Performance Collector API. They should for single threaded jobs be 100 % accurate.

However, the API only includes the counts for physical I/Os (and overflows) for the initial thread. Multithreaded jobs often make most I/Os within other threads, and these I/Os are not included in the I/O-counts of the API.

To compensate for this limitation GiAPA calculates the sum of physical I/Os of the thread I/O counts retrieved from the “Open List of Threads” API during HotSpots. If Job Accounting is active, GiAPA receives correct I/O counts from Job Accounting at end-of-run for the job. If Job Accounting is not running, GiAPA will retrieve I/O counts from the history log CPF1164 message – but the history log count for disk accesses does not include asynchronous I/Os.

If a multithreaded job runs over several days, GiAPA cannot provide a correct I/O count per day, because the data from Job Accounting or QHST is available only at the end of job.

Below is the first part of the same report also showing the “Max” lines. They specify the highest values found within an interval for the resource.

Headings for the “Max” line fields are shown in the second column heading line (JobNbr HotSp …).

```
GiAPA (c) by          Job Performance Summary Sorted by Total CPU Usage          Input=V2TESTDATA        08-06-20
iPerformance          First / last collection interval:  08-06-14 03:01:15 / 08-06-14 23:01:00          16:37:58
JobName   UserName   RunTime  Typ Itvs ActualUsr   CPUtime   % Prio     Logical     Physical  Trans Tr/JQ-time  Print  Ovfl.
JobNbr HotSp  RunDate from/to Pool Thrd MaxPages   H:MM:SS.s   CPU rity      I/Os        I/Os   actions h:mm:ss    lines  1000s
*BATCH    53781    00:46:09  B    0         5:28:17.7  0.0 00    56,695,258          0     5     2  976,951   311
        08-06-08 23:01:46  09   83  5,116,590 Max: 35.4 236.2 56       685,771     358,578     1     1  152,640     7

CMPRCDCNT  MIMIXOWN   11:52:44  B   271         2:13:14.0  33.3 25      427,108    1,749,081
312884   777 08-06-14 18:32:21  02    1    59,551 Max: 11.3  75.4 25      139,089      17,675

RCV_RQS    MIMIXOWN   03:01:15? B   480         1:27:54.6   7.3 25   402,058,468    1,188,143
282881  2592 08-06-14 23:00:45  02    1    81,108 Max:  2.8  18.7 25      287,956       1,304

*CFINT**          03:01:15  V    0          53:13.7   0.0 00          0          0
        08-06-14 23:01:00  01            Max:   .5   3.0 00

EDITCVAGB  DGWADMGB   03:07:30? B   460          46:56.2   3.9 50   224,837,462    1,537,337              35,390    60
386786   326 08-06-14 22:57:30  08    1  1,316,342 Max: 10.0  66.6 50     1,003,100       22,942              518
        (Additional lines have been removed for brevity)
```

Description of the report contents, and F-key options for the subfile: most of the data shown on the report is probably self-explanatory, and the exact meaning of terms like logical and physical I/Os and overflows are defined earlier in section “Definitions and FAQs”.

The job start and end time will in most cases be accurate down to the second. A job start/end time indicator immediately after the job start time will contain

- ***** (asterisk) if the shown actual job start time is before GiAPA data collection started, or before the date reported for this summary record (job running over several days).
- \+ (plus sign) if the job was continuing running the following day.
- **?** (question mark) if exact job end time is not known – start time may also be unknown (probably because job was still running when GiAPA data collection ended).

Note that GiAPA will report a job running across midnight as two jobs, one for each date.

The field under the column heading “**Itvs**” in the first line of each job indicates the number of 15 seconds’ performance collection intervals retrieved for the job.

“**Actual User**” will (for jobs started with a general user name, but used by different users) show the actual user name, if only one user was assigned to the job, and “*Various” if several users were using the job. More details can be seen by using F10=Details per interval.

The number of HotSpots for a job is found under column heading “**HotSp**” in the second line. Under the “**Thrd**” heading you find the highest number of threads for the job.

There are a few lines where the job identification has been replaced by “special names”: These lines use *CFINT**, *BATCH, *INTERACT, *AUTOSTR, *WRITER, etc., as job names, and contain summaries for a job type.

Example: On the line showing *BATCH as job name, the job user and number is replaced with the number of BATCH jobs summarized in the line. The values for use of resources on these lines represent the totals for all batch jobs that used minimal resources and therefore were not kept individually in the data expansion and analysis step.

F6, F7, F8, F9, and F10: when the cursor is positioned on a job in the subfile, use of these function keys will display other subfiles providing additional information about the job.

F18, F19, F20, F21, and F22: corresponds to using F6 – F10, but will instead of showing a subfile result in the display of a spooled file, giving the advantage that the normal F16=Find option of the operating system is available, and allowing you to print the report.

F6 and F18 = ODPs generates the ODP (Open Data Path) overview report based on HotSpot data and provides an overview per interval of the files opened. The report is described below.

F7 and F19 = File statistics generates the File statistics report specifying the detailed usage information for each file, based on HotSpot data. The report is described below.

F8 and F20 = File analysis generates the File analysis report, providing an analyzed overview of each of the files used by the job, based on HotSpot data. The report is described below.

F9 and F21 = Call stack generates the Call stack report, based on HotSpot data, including usage statistics for the most frequently used programs. The report is described below.

F10 and F22 = Details generates the “Details for job” report showing the resource usage details per interval for the job. The report is described below.

F11=Show/hide max is used to display or suppress the lines showing the maximum values on the report panel. Please note that the values shown on one maximum line do not necessarily originate from the same interval – one interval could have the highest CPU usage, another collection interval may account for the maximum transaction response time, etc.

F14=Create graph data creating a chart. The flexibility available here allows the user to select which data field should be selected, and how many records should be generated for the graphics data file.

Two fields are written to the graphics data file, one key field and one data field. The job name is used as key field, except if the cursor is positioned in the user name column, in which case the user name will be the key field for the graph. The data field is the currently selected sort criteria field. Example: if the subfile is sorted by logical I/Os, the number of logical I/Os are passed to the graphics data file. If the currently displayed data is sorted on job name, user name, job number or priority, use of F14 will have no effect.

The records for the graph are always selected from the beginning of the subfile, but the user can determine how many records should be included by positioning the cursor on a record in the subfile to indicate the last record selected. If the cursor is positioned outside the subfile, all records currently loaded in the subfile are selected (e.g. one subfile page of records if “Page down” was not used yet) – however, in either case only a maximum of 50 records are selected – since a very large number of records are not suited for creating charts.

Note that any graphics data file that was created earlier using the current subfile sort criteria as data field will be automatically overwritten, but GiAPA will keep and reuse any changes made to graph type, “Show Values” and selected color palette.

If you want to keep older graphics, use menu Option 28 to rename or move existing files before creating a new.

ODP Overview Report

```
GiAPA (c) by          ODP Overview for Job QPADEV00WM EVJIRXFB  207420      Input=SAMPLEDATA      08-06-20
iPerformance                                                                19:23:07
 Data collected at  Nbr. <------- Totals for all database files -------> Zero <------ Totals for files with few I/Os ------>
 YY-MM-DD hh.mm.ss ODPs ODPs    Writes      Reads  Upd Dlt etc.  I/Os ODPs    Writes      Reads  Upd Dlt etc.
 08-05-26 08:36:00  50   38       6     357,497       6   13   29      6      145       6
 08-05-26 08:36:15  50   38       6     455,391       6   13   29      6      145       6
 08-05-26 08:36:30  50   38       6     597,033       6   13   29      6      151       6
 08-05-26 08:37:15  50   38       6     733,536       6   13   29      6      157       6
 08-05-26 08:40:00  57   38      10      795,537      12   12   37      8      192      12
 08-05-26 08:40:15  57   38      10      870,071      12   12   37      8      192      12
 08-05-26 08:40:30  57   38      10    1,006,578      12   12   37      8      198      12
 08-05-26 08:41:15  58   38      11    1,095,994      15   12   38      9      217      15
          (Additional lines have been removed for brevity)
 08-05-26 08:49:45  84   74      45    2,512,289      46   17   58     29      596      44 +
F2=Cmd Line  F3=Exit    F7=File Statistics  F8=File Analysis  F9=Call Stack  F10=Details  ENTER=Go to Top
```

Requested by using F6 (or F18) from the Job Summary Report.

The report shows data for each 15-seconds interval where HotSpot data were collected for the job. It is important to be aware that this data was collected using sampling at HotSpots. The contents are a summary of the data also obtainable from a DSPJOB command Option 14 with F11=Display I/O details. For multithreaded jobs, the values shown represent the totals for the threads for which HotSpot data was collected.

Below are provided explanations to the various columns:

Nbr. ODPs indicates the total number of Open Data Paths = number of files opened by the job.

Totals for all database files is self-explanatory. The most interesting from a performance point of view is often the difference between two lines, if they show two consecutive intervals, since this indicates the total database activity within 15 seconds. If the number of ODPs varies heavily, it could mean that the job is wasting time by constantly opening and closing files.

Zero I/Os counts the number of database file ODPs that were opened by the job without showing any I/O values in the counts.

Totals for all files with few I/Os is a summary of scarcely used files, having so few I/Os that the I/Os – from a performance point of view – can be neglected. The limit for keeping the individual interval records for a file is specified under GiAPA Menu Option 78 (Installation parameters). Since file data below this limit are not kept individually, they cannot be shown in the reports requested with function keys F7 and F8.

F7, F8, F9, and F10 offer links to the other job-level analysis reports, i.e. they have the same function as when used from the “Job Performance Summary” report.

File Statistics Report

GiAPA (c) by iPerformance — File Statistics for Job QPADEV00VG EVJXYCU2 340113 — Input=SAMPLEDATA — 08/06/22 22:15:55

Data collected at YY-MM-DD hh.mm.ss	File nbr.	Library name	File name	Member name	Ty pe	I O	Number of writes	Number of reads	Updat delet unlock etc.	Relative record nbr	Share count
08-06-14 07:31:15	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		8,693,407		19,952	
08-06-14 07:31:30	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		9,349,813		19,952	
08-06-14 07:31:44	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		9,939,893		19,952	
08-06-14 07:31:59	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		10,600,476		19,952	
08-06-14 07:32:15	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		11,227,564		5,486	
08-06-14 07:32:30	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		11,847,305		19,952	
08-06-14 07:32:45	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		12,467,189		19,952	
08-06-14 07:33:00	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		13,051,365		19,952	
08-06-14 07:33:15	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		13,678,259		19,952	
08-06-14 07:33:30	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		14,274,611		1,538	
08-06-14 07:33:45	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		14,869,585		4,560	
08-06-14 07:34:00	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		15,546,949		12,075	
08-06-14 07:34:15	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		16,179,442		930	
08-06-14 07:34:30	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		16,796,719		2,891	
08-06-14 07:34:45	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		17,481,956		19,952	
08-06-14 07:35:00	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		18,125,763		19,952	
08-06-14 07:35:15	69	CFWGBPBD	FJBPA1L8	FJBPA1L8	LF	I		18,761,925		6,655	

(Additional lines have been removed for brevity)

F2=Cmd Line F3=Exit F4=Prompt for text F6=ODP Overview F8=File Analysis F9=Call Stack F10=Details ENTER=Go to Top

CFWGBPBD/FJBPA118 *FILE Finance: Text for general ledger acoounts 21.454 records

Requested by using F7 (or F19) from the Job Summary Report.

The File statistics report shows the exact “snapshot” usage information for each file and interval, based on HotSpot data similar to DSPJOB command Option 14 with F11=Display I/O details. If the data for a file did not change since the previously displayed interval, the line is suppressed. Details for files showing less than a certain minimum of I/Os are not kept individually, according to rules in the Installation parameters (Menu Option 78).

File nbr. is an internal GiAPA file sequence number – it is used by GiAPA to avoid mixing up data when a job has a file opened more than once without using shared open.

Type is the database file type, either LF for logical file or PF for physical file.

I O is used to tell if the file was opened for input, output, or both. It may be less efficient to open a file for I/O, if the program only uses either reads or writes, because logical blocking cannot be used for files opened for I/O, and since records read are locked.

Relative record number (RRN) has proved to be valuable information from a performance analysis point of view. When many HotSpots were taken for a job and file, RRN shows if the file is accessed in arrival sequence or randomly. It also indicates if many I/Os are used to access the same few records – in the example above, record number 19952 was accessed repeatedly.

Share count will show if the database file is re-opened with SHARE(*YES) used.

F4 = Prompt for text. When the cursor is positioned on a line in the subfile, F4 will fetch the descriptive text for the file from the object description and the number of records in the member accessed. The text will be displayed as a status message in the message line at the bottom of the screen.

F6, F8, F9, and F10 offer links to the other job-level analysis reports, i.e. they have the same function as when used from the “Job Performance Summary” report.

File Analysis Report

```
GiAPA (c) by          File Analysis for Job DBT950   BSNJSA1   423615  on 2008-06-14   Input=WEEK04_18      08-06-21
iPerformance                                                                          16:17:04
Library   File      Member    File File     Nbr. of    Nbr. of    Nbr. of    RRN Span      Reused  Reuse  Nbr.  %  Ack  RRN
Name      Name      Name      Nbr. Type      Writes     Reads  Other I/Os  (High-Low)      RRN   Count  Itvs     %  Count
LWYDDTA   DJOACC    DJOACC     42 PF O      826.875                        644.645              388
LWYDDTA   DJOACC10  DJOACC10  116 LF IO       1.350         21    647.488                    40
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I      250.164.148                    19.588     24.714   55 2647   2  2    1
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.723     46        2  4    2
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.742     224       8 12    3
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.753     46        2 14    4
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.760     172       6 21    5
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.777     89        3 24    6
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.793     66        2 26    7
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.842     70        3 29    8
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.858     176       7 36    9
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.901     99        4 39   10
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                     24.989     131       5 44   11
LWYDDTA   DBFYEL00  DBFYEL00   45 LF I                                        449      17 61   27
LWYDDTA   DBSFPL00  DBSFPL00   44 LF I        616.697                      22         10  121 2647   5  5    1
LWYDDTA   DBSFPL00  DBSFPL00   44 LF I                                       19 2500       94 99    1
LWYDDTA   DJOACC10  DJOACC10  133 LF IO   2.115.372         60       4  5.695.084              2234
LWYDDTA   DJOPRI10  DJOPRI10  211 LF I        4.211                       8         1  202  231 87 87    1
LWYDDTA   DJOPRI10  DJOPRI10  211 LF I                                       3   6       3 90    2
LWYDDTA   DJOPRI10  DJOPRI10  211 LF I                                       8   17        7 97    3
LWYDDTA   CPOBAL00  CPOBAL00  162 LF IO     10.273      3.714     3.714     10.058                3
                    214 Files  24.254.826 296.203.836   8.289.109          189.174.335 = Estimated reused   178
F2=Cmd Line  F3=Exit  F4=Prompt text  F6=ODP Overview  F7=File Statistics  F8=File Analysis Summary  F9=Call Stack  F10=Details
                                      F20=Print File Analysis Summary
```

Requested by using F8 (or F20) from the Job Summary Report.

The File analysis report is based on HotSpot data and provides an analyzed overview of the usage of each of the job's database files with data collected for more than one HotSpot. Only files showing more than a certain minimum of I/Os are included, according to rules in the Installation parameters (Menu Option 78).

The majority of the report columns are self-explanatory and the remaining are described below:

File nbr. is an internal GiAPA file sequence number.

File Type is a heading for a column telling both the file type (PF or LF), and whether the file is opened for Input, Output or both.

RRN Span (High – Low) illustrates – as the column heading implies – the RRN span of records the job has processed. It is calculated by subtracting the highest RRN from the lowest RRN and adding 1. (RRN = Relative record number.)

In the above example the file with the GiAPA internal file number 0045 shows 250.164.148 reads, but the RRN Span was only 19.588, meaning that over 250 million reads were used to access less than 20.000 records. Given that the file only was opened for input, this is an example that most likely shows waste of resources.

Reused RRN shows the relative record numbers that were found more than once in the record I/O statistics. If a file is not accessed between two intervals where HotSpot sampling was collected then the RRN will of course remain unchanged, but so will the I/O statistics.

Note that individual RRNs are only reported when the reuse count corresponds to at least 2 % of the total number of intervals. The reused RRNs not listed due to this limit will be shown in a

total on the last line for the file, as shown for file 45 above, where a total reuse count of 449 represents 16 reused RRNs. The number 16 is calculated by taking the value 27 found in column “RRN Count” and subtracting the value 11 found just above 27. In the ***FINAL TOTALS line the value in this column represents the total estimated number of reused records.

Reuse Count shows the number of HotSpots, where this RRN was accessed repeatedly.

Nbr. Itvs shown on the first report line for a file indicates the number of times HotSpot data were collected for the file. For file 45 there were samples from 2647 intervals.

% indicates the Reuse Count in percentage of the Nbr. Itvs, i.e. the percentage of HotSpots where this record was accessed repeatedly. The above example demonstrates that the file with the GiAPA internal file number 44 indicated 616.697 I/Os, and 94 % of the HotSpot intervals showed access of relative record number 19 in the file. Since data was collected at 2647 HotSpots, it is likely that record number 19 was re-read around 580.000 times (= 94 % of 616.697).

Ack % indicates the accumulated percentage. The example above demonstrates that the file with the GiAPA internal file number 0045 had 250 million reads over 2647 intervals where a HotSpot was collected. The accumulated percentage for the reappearing 11 different relative record numbers is 44. This implies that the job used 110 million reads to read 11 records.

In addition, the other RRNs used repeatedly added up to 17 % even though each were found in less than 2 % of the 2647 intervals, and therefore listed as a sum not showing individual RRNs.

RRN Count indicates the number of re-used relative record numbers. For file 0045 in the example above this column count to 11 for the 11 RRNs listed individually because they were used in => 2 % of the intervals. In the final line for file 0045 this column indirectly shows that there were 16 reused RRNs each representing less than 2 %. The reason is that the final RRN count value for this file is 27, and it was 11 in the penultimate line for the file.

Our conclusion for file 0045 is that 61 % of 250.164.148 reads (= 152.6 million read instructions) were used to read 27 records repeatedly. This is most likely correct given that the RRN Span for this file was 19588, and the RRNs hit remained around 24.000. Therefore, it was probably a rather small file that was accessed 250 million times. Obviously, much time could be saved by changing the program to avoid repeated reads of the same records.

The pink line at the bottom of the display shows job totals: the number of files found in the job, I/Os per type, in column “Reused RRN” the estimated number of reuses (= I/Os used to access records repeatedly), and how many different records were found to be accessed repeatedly. Also files not listed here (only found in one HotSpot) are included in this total.

F4 = Prompt for text. When the cursor is positioned on a line in the subfile, F4 will fetch the descriptive text for the file from the object description and the number of records in the member accessed. The text will be displayed as a status message in the message line at the bottom of the screen, as shown for the previous report “File Statistics”.

F6, F7, F9, and F10 offer links to the other job-level analysis reports, i.e. they have the same function as when used from within the “Job Performance Summary” report.

For **F7** the cursor may be positioned on a file resulting in only that selected file will be included in the report. If the selected file is read only in arrival sequence, the second title line will contain the message “Relative record numbers are ascending --> Access in arrival sequence”.

F8 = File Analysis Summary will generate the report shown below. Using F20 will direct the output to print.

File Analysis Summary

```
GiAPA (c) by        File Analysis Summary for Job QDFTJOBD  QUSER    823806  on 2008-07-25  Input=DATA080725    08-08-04
iPerformance                                                          10:50:29
Library   File     Member    File File    Nbr. of    Nbr. of   Nbr. of  Nbr. Diff Reuse % of   RRN Span   Potentially
Name      Name     Name      Nbr. Type    Writes     Reads     Other I/Os Itvs RRNs Count Itvs  (High-Low) Superfluous I/O
ITE2C0    E2PQAJ   E2PQAJ      3 PF IO        74.411.480 74.411.480 688   3  667 97      1.546   148.821.414
ITE2C0    E2POBJ   E2POBJ      1 PF I          1.696.006        333  56  288 86     14.795     1.681.211
ITE2      E2PLIB   E2PLIB      2 PF I            145.516        319  17  312 98       131       145.385
ITE2C0    E2POBJ   E2POBJ      4 PF IO             1.017       339  21   5  14 67        33           984
ITE2C0    E2PJRN   E2PJRN      5 PF I                873         21   2  21 100         2           873
ITE2C0    E2PZCENL1 E2PZCENL1   7 LF IO       128       154          6   2   4 67       67           188
ITE2      E2PLIB   E2PLIB      6 PF I              266         18   5  17 94       133          250
ITE2C0    E2PZCENL1 E2PZCENL1   8 LF IO       120        92          18   1   2 11       72           48
ITE2C0    E2PZCENL1 E2PZCENL1   9 LF IO       114        97          14   4   8 57       55           120



                      9 Files       362   76.255.501  74.411.819      95                 150.650.473
F2=Cmd Line  F3=Return  F4=Prompt text  F6=ODP overview  F7=File statistics  F8=FileName totals  F9=Call Stack  F10=Details
```

Using F8 from the previous report will generate this next report containing the same file analysis data presented in a slightly modified summary format, sorted descending on the rightmost column, “Potentially Superfluous I/Os”. Most of the columns are the same as in the above “File Analysis” – please refer to the explanation for that report.

Diff RRNs contains the number of different record numbers found. It is used for files where a few relative record numbers were found to be accessed repeatedly.

% of Itvs: The percent of the total number of intervals in which the repeated RRNs were found.

Potentially Superfluous I/Os contains the number of I/Os that could have been saved if each record used by the program from that file only had been accessed once by the job. The contents of this column can be explained using the report shown as an example:

The first file shows 74.411.480 reads and 74.411.480 other I/Os (probably updates). 688 HotSpots were collected, and 667 = 97 % of these access 3 out of totally 1.546 different RRNs (relative record numbers). If 688 samples denote a representative picture, we can calculate that 97 % of (74.411.480 + 74.411.480) = 144.358.270 I/Os were used to access only 3 records. They could have been saved, if we kept the 3 records in memory, and only updated them at end-of-job. Since the total RRN span was 1.516 indicating that the file is small it would probably be better to keep the entire file in memory and thus save even more I/Os. Reducing writing of the updates that trigger physical disk I/Os would result in significant savings.

The second file shows more than 1.6 million reads used to access at the most 14795 records, meaning that each record on average was read over 100 times. However, computers today are so fast that the savings here might not be worthwhile, unless the job runs very frequently.

It is obviously important to realize that the statistical probability of this estimate increases with the number of HotSpot intervals, and is less likely to reflect the actual situation in cases where HotSpots were collected for only a few intervals. The last column therefore only shows a value if data is available for more than five intervals.

F6, F7, F9, and F10 offer links to the other job-level analysis reports, i.e. they have the same function as when used from within the “Job Performance Summary” report.

F8=FileName totals may be useful if the job continuously opens and closes files, because this report will provide a total per file/member name. If a file is opened e.g. 785 times within a job, this option will show a total for the number of I/Os for the file.

File Name Totals

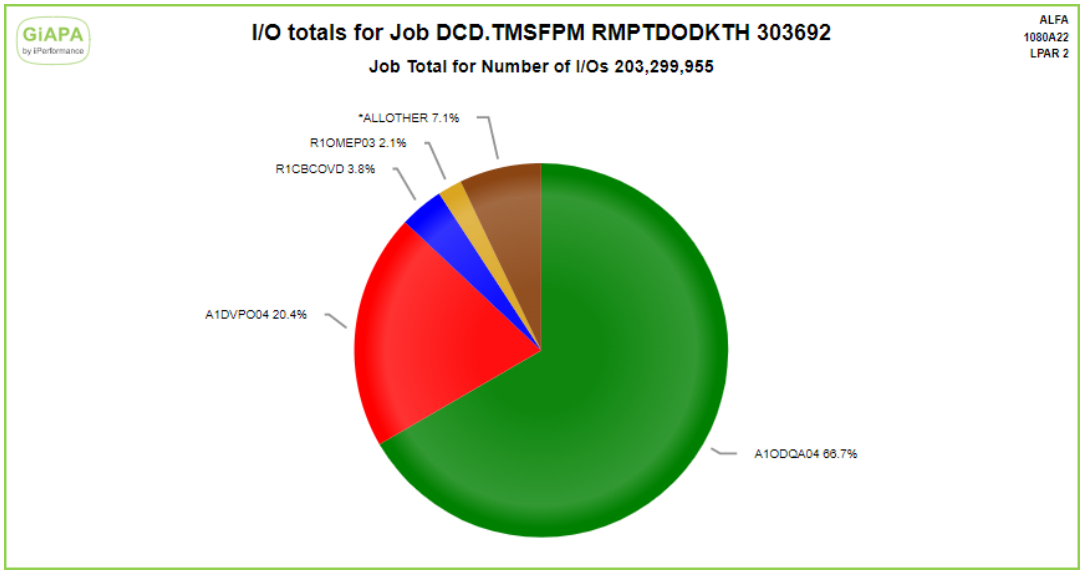
```
GiAPA (c) by       Files with over 2 % of I/Os in Job DCD.TMSFPM RMPTDODKTH 303692  on 2019-01-04   Input=TESTDATA        22-01-12
iPerformance                          (Totals of all opens per Library/File/Member)                                      15:52:56
Library    File       Member     Type Count    Log.Writes     Log.Reads     Misc.I/Os    Total I/Os  Recs. in File  Access PerCent
R1NPFEDP   A1ODQA04   A1ODQA04   LF       2                 135,580,200                135,580,200
Library or file not found
R1NPFEDP   A1DVPO04   A1DVPO04   LF       2                  41,387,640                 41,387,640
Library or file not found
R1NPFEDP   R1CBCOVD   R1CBCOVD   PF       2            44     7,700,129           238     7,700,411
Library or file not found
R1NPFEDP   R1OMEP03   R1OMEP03   LF       3                   4,201,857                  4,201,857
Library or file not found
 *** TOTAL *ALLOTHER                     46        62,473    14,350,066        17,308    14,429,847
           35 different files/members were found



 F2=Cmd Line   F3=Return   F6=ODP Overview   F7=File Statistics   F9=Call Stack   F10=Details   F11=Hide/show text   F14=Graph
```

Using F8 from the previous report will generate this report, showing I/Os summarized per file/member name for the ten files having most I/Os. Only files having more than 2 % of the total number of I/Os for the job are selected.

The report provides a very good overview in cases where many files were opened and closed frequently and therefore shown separately per open/close in the previous reports. However, the report may not provide the absolutely exact picture per file name, because

- files that did not reach the minimum number of I/Os to report separately (specified in the installation parameters) are not included, and
- the data used is collected at HotSpots and are thus samples not necessarily including the highest number of I/Os (reached just before a file is closed).

Use of F14 will generate a graph that will show the I/O totals per file/member. The pie chart below was generated based on the example described above.



Call Stack Reports

Using F9 for a job from the “Job Performance Summary” report will display different call stack reports depending on whether the job is a single- or multithreaded job. The Call Stack reports are based on HotSpot “snap shots”. The contents originate from the same data as shown by the DSPJOB command Option 11 “Display Call Stack”.

Call Stack Report for Single Threaded Job

```
GiAPA (c) by        Call stacks for Job QPADEV00W7 CJFDFSN   796264 on 08-05-23   Input=SAMPLEDATA          08-06-21
iPerformance        Analyze Call Stack Option: _ Enter * (=all) or 1-9 for number of levels to include        16:23:24
HotSpot   HotSp  Stk Initial thread Sta-  Tot.Job CPU  Non-Q    Active    Called    Called    Called    Called    Called
requested recvd  dpt  function   tus   HH:MM:SS,s  program   program   last-1    last-2    last-3    last-4    last-5
16:12:45  12:46.1  12 RUN-ACBCL01   RUN     19:38.4 X4FDYGS   QDMCLOSE  X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:13:00  13:00.2  12 RUN-ACBCL01   RUN     19:43.0 X4FDYGS   QDMCLOSE  X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:13:15  13:42.4   0 IDX-ACBTCPL5  RUN     19:47.8 RGRGZPFMCP QDBFFCPY  QDBRGZPF  QCMDEXC   RGRGZPFMCP _CL_PEP  RGRGZPFMCP
16:13:30  13:30.1   0 IDX-ACBTCPL5  RUN     19:52.6 *NotAvail
16:13:45  13:45.4  12 RUN-ACBCL01   RUN     19:58.1 X4FDYGS   QDBPUT    X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:14:00  14:00.3  12 RUN-ACBCL01   RUN     20:05.3 X4FDYGS   QDBPUT    X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:14:15  14:15.1  12 RUN-ACBCL01   RUN     20:08.5 X4FDYGS   QDMCLOSE  X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:14:30  14:30.2  12 RUN-ACBCL01   RUN     20:14.4 X4FDYGS   QDBPUT    X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:14:45  14:45.7  14 RUN-ACBCL01   RUN     20:22.7 X4FDYGS   QTNOPEN   QDBOPEN   QDMCOPEN  X4FDYGS   X4EBYGS   X4DZYGS
16:15:00  15:00.4  13 RUN-ACBCL01   RUN     20:27.6 Z3SDSTD   QCLRSLV   Z3SDSTD   X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD
16:15:15  15:15.0  14 RUN-ACBCL01   RUN     20:33.8 X4FDYGS   QDBCHKLA  QDBOPEN   QDMCOPEN  X4FDYGS   X4EBYGS   X4DZYGS
16:15:30  15:30.2  12 RUN-ACBCL01   RUN     20:38.2 X4FDYGS   QDBPUT    X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:15:45  15:45.1  12 RUN-ACBCL01   RUN     20:44.1 X4FDYGS   QDMCLOSE  X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:16:00  16:01.0  12 RUN-ACBCL01   RUN     20:51.4 X4FDYGS   QDBPUT    X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:16:45  16:45.8  12 RUN-ACBCL01   RUN     20:55.0 X4FDYGS   QDMCLOSE  X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:17:00  17:00.3  13 RUN-ACBCL01   RUN     20:59.8 Z3DRNTD   QMHRCVPM  Z3DRNTD   X4FDYGS   X4EBYGS   X4DZYGS   <LONG_NAME
16:17:15  17:15.0  12 RUN-ACBCL01   LCKW    21:02.4 X4FDYGS   QDBPUT    X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:17:30  17:30.4  12 RUN-ACBCL01   RUN     21:09.7 X4FDYGS   QDBPUT    X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS
16:18:00  18:00.1  12 RUN-ACBCL01   RUN     21:18.5 X4FDYGS   QDMCLOSE  X4FDYGS   X4EBYGS   X4DZYGS   X4D7VRD   X1O6RWS +
 QDMCLOSE=25% QDBPUT=25% QDBFFCPY=10% QDBOPEN=8% QDBGETM=5% QCLRSLV=5% Z3DRNTD=3% QWTPMSRQ=3% QTNOPEN=3% QMHRCVPM=3%
QDMCOPEN=3%
 F2=Cmd Line  F3=Exit  F4=Prompt text  F6=ODP Overview  F7=File Stat.s  F8=File Analysis  F10=Details  F11=Whole stack  F14=Graph
 QTNOPEN   OPEN FILE UNDER COMMITMENT CONTROL        F20=Exclude wait status lines    F21=Set time limits for analysis
```

Analyze Call Stack Option in the 2nd title line allows you to enter an * for all, or 1 – 9 to specify a number of call stack levels (starting from the last called) to analyze in order to produce call stack statistics. Please refer to the description of the “Analyzed Call Stack Report” below.

Stk dpt (stack depth) shows the number of levels in the call stack for the job. A user installation parameter defines a limit for the number of levels to be retrieved – shipped value is 20 levels.

Initial thread function and **Status** has the same contents as on the WRKACTJOB display.

Non-Q Program indicates the name of the last called application program in the call stack. The currently active program for a running job is most often an operating system service function, typically I/O, or (for interactive jobs) QT3REQIO meaning that the job is awaiting work station input. Therefore, GiAPA will search through the call stack, starting with the last called program, and store information about the last called program not starting with Q, i.e. not an IBM program and therefore probably the active application (= user) program.

*NotAvail is shown when the call stack could not be retrieved which typically is caused by the job temporarily running below MI-level, e.g. generating an index on a data base file member.

Active program is the name of the last called program in the call stack when the HotSpot sampling was collected. All the active programs are counted, a percentage is calculated, and a

statistic in percentage showing the most frequently found program names is displayed on the pink line at the bottom of the page.

Called-1, Called-2, etc. contains the last 5 programs, procedures, classes, or modules in the call stack (above the active program). To supply the most information in a minimum amount of space, names are not repeated. Example: If a job is running within procedure ABC33 belonging to module ABC22 in program ABC11, all three names would be shown. In contrast, if program XXYYZZ runs procedure XXYYZZ, XXYYZZ will only be shown once.

If e.g. a procedure name exceeds 10 characters and therefore does not fit within the column space available, the last 9 nine characters are shown, prefixed by < to indicate a longer name.

The pink line at the bottom of the report (right above the line with the F-Keys) contains interesting information by showing the active program names and their percentage of occurrence within all the HotSpots collected for the job (i.e. not only displayed on this page).

The example above demonstrates that the job in 39 % of all HotSpots was running a file open or close. The job was writing records (QDBPUT = unblocked database write) in 25 % of the HotSpots, and 10 % was used on copying files (QDBFFCPY). Whether this is a reasonable use of resources depends on the application, but it is far from normal and therefore most likely worth scrutinizing.

Note that since this is based on the HotSpot sampling, the statistics represent elapsed time, not CPU time used by the job.

F4 = Prompt for text. When the cursor is positioned in the subfile on a program name (including the program names in the pink line at the bottom), F4 can be used to fetch the descriptive text for the program from the object description, or (for object names starting with the letter Q) from a GiAPA table containing an explanatory description for most of the IBM programs. For names of objects from 3rd party software companies (e.g. MOVEX) a code may be added causing explanatory text to be fetched from file GIAPA092P1.

The text will be displayed as a status message in the message line at the bottom of the screen, as shown in the above example.

F6, F7, F8, and F10 offer links to the other job-level analysis reports, i.e. they have the same function as when used from within the “Job Performance Summary” report.

F11 used with the cursor on a call stack line will show the entire stack, displaying all levels stored, including statement numbers – please refer to the report “Report on All Levels Kept for one Call Stack” below.

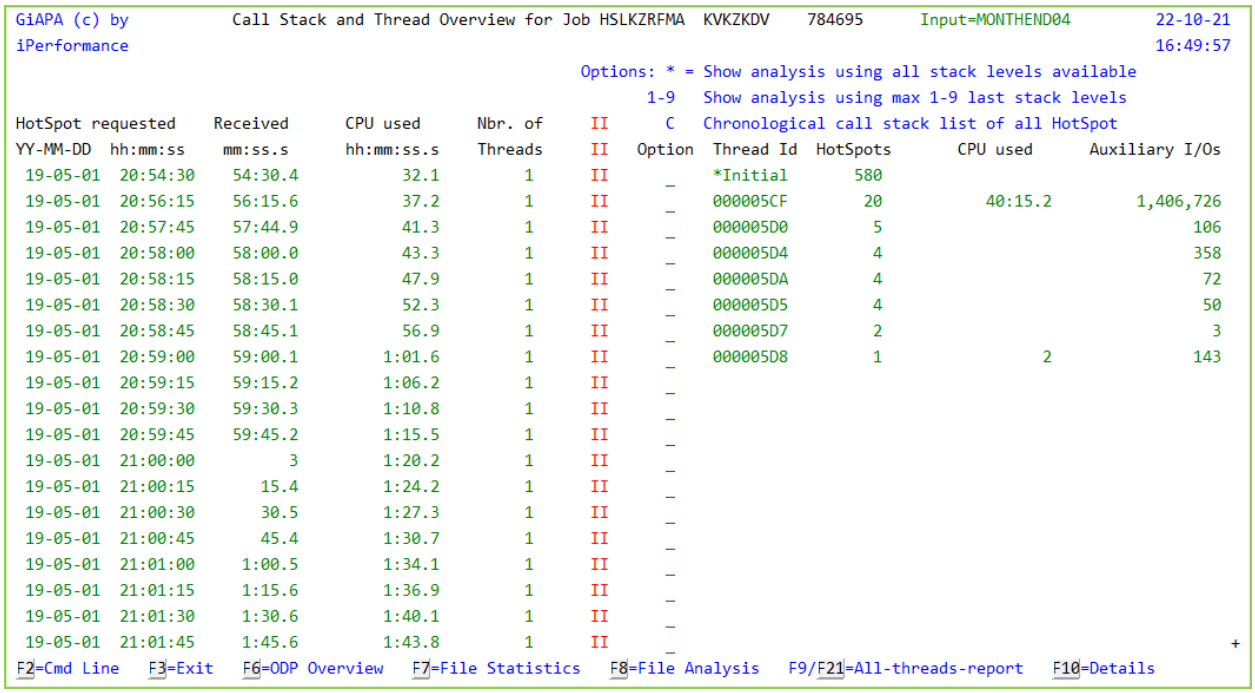
F14=Create graph data will create a pie chart reflecting the overall statistics represented in pink at the bottom of the panel. Programs using less than 2 % will be grouped together as *OTHER.

Note that any graphics data file that has been created earlier by using F14 from this panel will automatically be overwritten unless they have been renamed using menu Option 28.

F20=Exclude wait status lines change the statistics in the pink line so they only are based on data from call stacks where the job was actively running.

F21=Set time limits for analysis displays a window that may be used to limit the call stack analysis to only include HotSpots between two time intervals. This is useful to obtain call stack statistics and analysis for segments of a long running job with many HotSpots.

Call Stack and Thread Overview Report for Multithreaded Job



```
GiAPA (c) by          Call Stack and Thread Overview for Job HSLKZRFMA  KVKZKDV    784695      Input=MONTHEND04          22-10-21
iPerformance                                                                                                      16:49:57

                                                          Options: * = Show analysis using all stack levels available
                                                                1-9   Show analysis using max 1-9 last stack levels
HotSpot requested    Received      CPU used      Nbr. of     II     C   Chronological call stack list of all HotSpot
YY-MM-DD  hh:mm:ss    mm:ss.s      hh:mm:ss.s    Threads     II   Option  Thread Id  HotSpots       CPU used      Auxiliary I/Os
 19-05-01  20:54:30    54:30.4           32.1          1      II      _    *Initial       580
 19-05-01  20:56:15    56:15.6           37.2          1      II      _    000005CF        20           40:15.2        1,406,726
 19-05-01  20:57:45    57:44.9           41.3          1      II      _    000005D0         5                             106
 19-05-01  20:58:00    58:00.0           43.3          1      II      _    000005D4         4                             358
 19-05-01  20:58:15    58:15.0           47.9          1      II      _    000005DA         4                              72
 19-05-01  20:58:30    58:30.1           52.3          1      II      _    000005D5         4                              50
 19-05-01  20:58:45    58:45.1           56.9          1      II      _    000005D7         2                               3
 19-05-01  20:59:00    59:00.1         1:01.6          1      II      _    000005D8         1                 2             143
 19-05-01  20:59:15    59:15.2         1:06.2          1      II      _
 19-05-01  20:59:30    59:30.3         1:10.8          1      II      _
 19-05-01  20:59:45    59:45.2         1:15.5          1      II      _
 19-05-01  21:00:00          3         1:20.2          1      II      _
 19-05-01  21:00:15       15.4         1:24.2          1      II      _
 19-05-01  21:00:30       30.5         1:27.3          1      II      _
 19-05-01  21:00:45       45.4         1:30.7          1      II      _
 19-05-01  21:01:00     1:00.5         1:34.1          1      II      _
 19-05-01  21:01:15     1:15.6         1:36.9          1      II      _
 19-05-01  21:01:30     1:30.6         1:40.1          1      II      _
 19-05-01  21:01:45     1:45.6         1:43.8          1      II      _                                                    +
F2=Cmd Line   F3=Exit   F6=ODP Overview   F7=File Statistics   F8=File Analysis   F9/F21=All-threads-report   F10=Details
```

This report does not display a job call stack, because multithreaded jobs have a call stack for each thread. In contrast this report provides:

- an overview of when the HotSpots for the job were collected, along with the total job CPU usage per interval, and the number of threads active in the job (on the left).

- a list of threads for which call stacks were retrieved, with number of HotSpots and the total CPU and auxiliary I/Os used for each thread (on the right).

Typically for multithreaded jobs, only one (or very few) threads use any significant amount of resources. When the aim is to find the most resource consuming routines, it is not worthwhile to collect information for all the (often quite large number of) threads that use limited resources.

The idea is therefore to only request information on thread level for those threads that illustrate a significant use of resources. Note that the limits for when thread level information should be retrieved and for the number of threads to process are specified in the installation parameters.

Option input field in the subfile on the right:

- Specifying * or 1 – 9 in the option input field requests a call stack analysis for one thread for all, or for 1 – 9 call stack levels, respectively. Please refer to the description of the “Analyzed Call Stack Report” below.

- Entering C for Chronological report in the option field requests a call stack report for one thread. That report is fairly similar to the single thread call stack report mentioned earlier, and is described below.

F6, F7, F8, and F10 offer links to the other job-level analysis reports, i.e. they have the same function as when used from within the “Job Performance Summary” report.

F9/F21 displays/prints call stacks for all threads combined into one report, thus also allowing creation of a pie chart showing percentages for the active programs of the entire job.

Chronological Call Stack Report for One Thread

```
GiAPA (c) by      Call Stack for Job I.6132    M3SRVADM   271127   Thread 000019FA on 13-01-09    Input=TMHE130109   13-02-14
iPerformance                                                                                      15:22:25
Recieved Stk Sta-  CPU used Auxiliary <--- Currently Active = Last Call Stack Entry ----> <- Last User Pgm or Prev. Stack Entry -->
hh:mm:ss dpt tus  h:mm:ss.s   I/Os   Pgm+Lib or JAVA Class Mod/MthTyp Procedure or Meth. Pgm+Lib or JAVA Class  Procedure or Meth.
======== === ==== ========= ========= ==================== ========== ================== ====================== ==================
13:47:46  36 RUN   31:15.0   1221373 <db/common/GenericMDB *JAVA_MMI  READE_impl       mvx/app/pgm/SAS530    movexMain
13:48:01  37 RUN   31:17.6   1258430 unix             *PASE-32   <32>:thread_tsleep mvx/app/pgm/SAS530    movexMain
13:48:16  52 RUN   31:20.3   1296369 unix             *PASE-32   <32>:thread_tsleep mvx/app/pgm/SAS530    movexMain
13:48:39  51 RUN   31:25.0   1318813 <mData$ActualDataDISK *JAVA_JIT  getBuffer       mvx/app/pgm/SAS530    movexMain
13:48:46  52 RUN   31:28.7   1319950 unix             *PASE-32   <32>:thread_tsleep mvx/app/pgm/SAS530    movexMain
13:49:02  53 RUN   31:33.8   1319952 unix             *PASE-32   <32>:thread_tsleep mvx/app/pgm/SAS530    movexMain
                          (Additional lines have been removed for brevity)
13:52:45  36 RUN   32:05.1   1421373 <db/common/GenericMDB *JAVA_MMI  READE_impl       mvx/app/pgm/SAS530    movexMain
13:53:01  36 RUN   32:11.0   1437382 <db/common/GenericMDB *JAVA_MMI  READE_impl       mvx/app/pgm/SAS530    movexMain
13:53:17  51 RUN   32:11.8   1441835 <mData$ActualDataDISK *JAVA_JIT  getBuffer       mvx/app/pgm/SAS530    movexMain
 unix=50% <mData$ActualDataDISK=32% <db/common/GenericMDB=15% QDBGETSQ=3%
F2=Cmd Line  F3=Exit  F4=Prompt text  F6=ODP Overview  F7=File Stat.s  F8=File Analysis  F10=Details  F11=Whole stack  F14=Graph
SAS530   MOVEX Service Agrmt. Display Lines.
```

Stk dpt (stack depth) shows the number of levels in the call stack for the job. A user installation parameter defines a limit for number of levels to be retrieved – shipped value is 20 levels.

Status has the same contents as the corresponding column on a WRKACTJOB display.

CPU used displays CPU seconds since the data collection started.

Currently Active shows the names from the last called level in the call stack when the HotSpot sampling was collected. All the active programs are counted, a percentage is calculated, and the most frequently identified are displayed at the bottom of the page in pink.

Last User Pgm or Prev. Stack Entry contains the name of the last called non-IBM program or class. If all are IBM programs (starting with the letter Q) then the 2nd last call level is shown.

If a name (e.g. procedure name) exceeds the space available in the column, the last characters of the name is shown, prefixed by < to indicate a longer name.

The pink line at the bottom of the report displays the occurrence percentage for the active program names within all the HotSpots collected for the job. Note that since this is based on the HotSpot sampling, these statistics represent elapsed time, not CPU time used by the job.

F4 = Prompt for text. When the cursor is positioned on a program name, F4 can be used to fetch the descriptive text for the program or class. In the above example the class text was fetch from file GIAPA092P1 for the software product MOVEX.

F6, F7, F8, and F10 offer links to the other job-level analysis reports, i.e. they have the same function as when used from within the “Job Performance Summary” report.

F11 used with the cursor on a call stack line will show the entire stack, displaying all levels kept, including statement numbers; please see “Report on All Levels Kept for one Call Stack” below.

F14=Create graph data will create a pie chart reflecting the overall statistics represented in pink at the bottom of the panel. Programs using less than 2 % will be grouped together as *OTHER. Any earlier graphics data file created by using F14 from this panel will automatically be overwritten.

Analyzed Call Stack Report

```
GiAPA (c) by    6 Levels Stack Analysis for Job EHYCRPBFV  P59020    964364   Thread *INITIAL    Input=SCRAMBLED    13-02-14
iPerformance                        377 Call Stacks Available                              17:40:23
NbrOf Sta- <------ Currently Active = Last Entry in Call Stack ------> <--- Last User Program or Previous Entry in Call Stack --->
HotSp tus  Stmt  Pgm+Lib or JAVA Class  Mod/MthTyp Procedur or Method  Stmt  Pgm+Lib or JAVA Class  Mod/MthTyp Procedur or Method
===== ==== ====== ==================== ========== ================== ====== ==================== ========== ==================
 120 RUN  MI 3E2 QDBGETKY   QSYS                        4160 OFF2O02   AVECOJAP   OFF2O02   OFF2O02
 114 LCKW MI 3E2 QDBGETKY   QSYS                        3330 OFF2O02   AVECOJAP   OFF2O02   OFF2O02
  66 RUN  MI 3BB QDBPUT     QSYS                      3549 OFF2O02   AVECOJAP   OFF2O02   OFF2O02
  23 RUN       LZRNPMV   AVECZOAP   LZRNPMV   LZRNPMV           LZRNPMV   AVECZOAP   LZRNPMV   .QMXO.OCO.LZRNPMV
   8 RUN  MI 3E2 QDBGETKY   QSYS                        2268 OFF2O02   AVECOJAP   OFF2O02   OFF2O02
   5 RUN  MI 3E2 QDBGETKY   QSYS                        3330 OFF2O02   AVECOJAP   OFF2O02   OFF2O02
   5 RUN  MI 328 QDBGETKY   QSYS                        4160 OFF2O02   AVECOJAP   OFF2O02   OFF2O02
   4 RUN  MI 147 QDBGETKY   QSYS                        2135 OFF2O02   AVECOJAP   OFF2O02   OFF2O02
                        (Additional lines have been removed for brevity)
F2=Cmd Line  F3=Exit  F4=Prompt Txt  F5=Status+Stmt  F6=ODP Overview  F7=File Stat.  F8=File Analysis  F10=Details  F11=Whole stack
```

This report provides detailed analysis of all call stacks available for the job or thread. It offers a powerful extension of the pink line at the bottom indicating active program statistics. This option is available on the call stack reports for single threaded job, and for a selected thread.

To request this report the user specifies the number of call stack levels to include in the analysis: 1 – 9, or * (asterisk) for all levels kept. Subsequently, GiAPA will process all call stacks for the job or thread, but only include the number of levels specified. The report indicates, in descending order, a count for each combination of call stacks found.

F5 can be used to include/exclude the status of the thread and the statement number for the active program. This will lead to the count in the “Nbr.Of HotSpots” column to change if the active program was using different statement numbers at different HotSpots, or if the status for a statement number varied between e.g. “RUN“ and “LCKW” (Lock Wait). In the above example statement 3330 in user program OFF2002 is reading a file (QDBGETKY is the active program). However, the job is in LCKW (Lock wait) in 114 HotSpots, and only in “RUN” in five cases.

QCMD	QCMD
_CL_PEP	QWCCDSAC
CL_MENU	QUIMGFLW
CL_PGMAA	QUICMD
RPGLE123	_CL_PEP
	CL_MENU
	CL_PGMAA
	RPGLE123

The significance of the number of levels in the analysis is illustrated in this simplified example: suppose you want to identify how often a job is using a given application. This can be identified by program CL_MENU calling CL_PGMAA, which in turn calls program RPGLE123.

If an interactive user calls that application directly from the command processor program QCMD the call stack will resemble the left column.
In contrast, if the user calls e.g. from the command line of the WRKACTJOB command, it would resemble the 2nd column of the example shown above.

If a user opted for both calls, and the program RPGLE123 in both instances used enough resources that GiAPA HotSpots retrieved the call stack the following scenario would occur: an analysis for more than four levels would differentiate between the two ways of calling that specific application, whereas an analysis using less than five levels would result in a total count for the application.

Report on All Levels Kept for one Call Stack

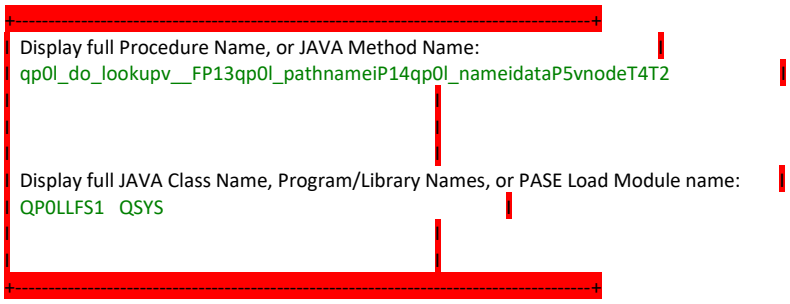
Using F11=Whole Stack with the cursor on a call stack line from one of the call stack reports described above will provide this report, illustrating all details for all the call stack levels stored.

```
GiAPA (c) by       Call Stack for Job TESTRUN5   KAARE     054132 Thread 0000003C at 08-06-14 12:08:30.7        08-06-21
iPerformance                          Input=V2TST_IPFR                                 17:18:09
Level  StmtNbr  Program and Library, or JAVA Class      Mod/MthTyp   JAVA Method Name or Procedure Name
  0       8 QSOSRV1    QSYS                  QSOSYS     oprt
 -1      37 QZBSCOMM   QSYS                    QZBSCOMM   QzbsReceiveClientReq
 -2   43970 QZDASRV    QSYS                    QZDACMDP   FO.MFREPD
 -3   40254 QZDASRV    QSYS                    QZDACMDP   QZDACMDP
 -4       1 QZDASOINIT QSYS                  QZDASOIT   nfya
 -5         QZDASOINIT QSYS                  QZDASOIT   .FGG.OCO..Bt
  (Additional lines have been removed for brevity)

F2=Cmd Line  F3=Exit  F4=Prompt for text  F6=ODP Overview  F7=File Statistics  F8=File Analysis  F10=Details  F11=Show Full Name
```

If the report is requested from the Analyzed Call Stack Report, and e.g. only four levels were requested, then only four levels will be available. In this case the date and time in the title will consist of zeroes, given that a line in the Analyzed Call Stack Report represents a summary of several HotSpots.

Details are available for a maximum of 99 call levels (number of levels retrieved and kept is determined by a GiAPA Installation parameter).

F11=Show full name. Even on this report the names might exceed the space available per line. Similar to the other reports, this will be shown by printing the last part of the lengthy name, prefixed by <. The complete object names (up to a maximum of 255 characters) will appear in a window if the cursor is positioned on the lengthy name and F11 is activated.



Also please note that for Movex / M3 threads, all Movex / M3 classes are stored and displayed. In the example below, the default value of 20 for number of call stack levels to keep was specified in the GiAPA installation parameters. The call stack contained 51 levels at this point in time (the stack depth can be seen in the earlier shown “Chronological Call Stack Report for One Thread”).

In addition to the active and the previous 19 levels, four more call stack levels were stored, because they contained class names prefixed by “mvx/app/pgm”. This means that even when M3 / Movex threads call stacks have more levels than the specified number to store you will always be able to determine which M3 / Movex classes were running.

```
GiAPA (c) by        Call Stack for Job M3AUTO    M3_ADMIN   489665 Thread 00000324 at 14-08-19 11:41:03.0          14-08-21
iPerformance                          Input=M3DATATEST                                                          10:18:09
Level  StmtNbr  Program and Library, or JAVA Class      Mod/MthTyp   JAVA Method Name or Procedure Name
Active      8   unix                          PASE-64   <syscall64>:thread_tsleep
 -1     5A4   libpthreads.a(shr_xpg5_64.o)          PASE-64   _event_sleep
 -2     2BC   libpthreads.a(shr_xpg5_64.o)          PASE-64   _event_wait
 -3     4E8   libpthreads.a(shr_xpg5_64.o)          PASE-64   _cond_wait_local
 -4      C0   libpthreads.a(shr_xpg5_64.o)          PASE-64   _cond_wait
 -5     1AC   libpthreads.a(shr_xpg5_64.o)          PASE-64   pthread_cond_wait
 -6     904   libj9thr26.so                  PASE-64   monitor_wait_original
 -7      34   libj9thr26.so                  PASE-64   j9thread_monitor_wait
 -8      F8   libj9vm26.so                   PASE-64   internalAcquireVMAccessNoMutexWithMask
 -9      74   libj9vm26.so                   PASE-64   internalAcquireVMAccess
-10     5B4   libj9vm26.so                    PASE-64   getArrayRegion
-11      C8   libi5osenv.so                   PASE-64   J9_i5OS_GetByteArrayRegion
-12           unix                          PASE-64   <callbackrtn>
-13       5   QP2USER2   QSYS                 QP2API    runpase_common__FiPvT2
-14       9   QP2USER2   QSYS                 QP2API    __Qp2CallPase2
-15       1   QP2USER    QSYS                 QP2USER   Qp2CallPase
-16       4   QXJ9VM     QJVM7064               QXJ9PASEAC <14Qxj9PaseAccessFQ2_14Qxj9PaseAccess19Xj9PaseJniFunctionsPvi
-17      15   QXJ9VM     QJVM7064               QXJ9JNI   GetByteArrayRegion__FP7JNIEnv_ilT3PSc
-18      19   MVXRLA     DEVGRID                Q105850000 Java_javarla_RLAFile_RLAGetKey
-19       8   unix                          PASE-64   <syscall64>:_ILECALLX
        7B   mvx/app/pgm/MMS920                *JAVA_JIT  RDIS
       8FD   mvx/app/pgm/customer/MMS920          *JAVA_JIT  TRAREC
       167   mvx/app/pgm/MMS920                 *JAVA_MMI  movexMain
        4D   mvx/app/pgm/MMS920CL               *JAVA_MMI  BODY
Class MMS920 at 11:41:00.0 RALUK:4. UTJAN
F2=Cmd Line  F3=Exit  F4=Prompt for text  F6=ODP Overview  F7=File Statistics  F8=File Analysis  F10=Details  F11=Show Full Name
```

Details for Job Report

```
GiAPA (c) by          Details for Job EDIRCV2GB  CFWADMGB   386786 on 2008-06-14  Jobtype B          08-06-25
iPerformance                                                                      10:57:05
HH:MM:SS Thrds Pty JobCPU    Logical  Syncr.    Syncr.  Async.    Async.  Misc.I/O Nbr.Trans PagesUsed Pag.Alloc. Curr. User
 TotCPU% Itv  Pool  CPU%      I/O    DataBase  Non-DB  DataBase  Non-DB  Perm.Wrt Transtime Num.Ovfl. Deallocat. Printlines
 7:26:00   1 50   7.9    602897     294      2       7         3386         456058    79981
  14.6% 15  08  52.5%      2056      21     43     681      70     788          136    77419      191

 7:26:15   1 50   7.9    656220     330      1       5         2882         459006    71048
  16.6% 15  08  52.9%      1697      25     47     486      58     583          120    68100      230

 7:26:30   1 50   8.6    606539      91              1         4448         465158    99429
  12.0% 15  08  57.4%      2749      23     53     918      94    1047          188    93277       60

 7:26:45   1 50   8.2    663953     293      2       1         3270         466462    76368
  10.0% 15  08  54.5%      1931      13     21     636      78     725          152    75064      191

 7:27:00   1 50   8.7    780630     192      4                 2874         467242    72038
   9.3% 15  08  57.9%      1643      10     15     514      60     587          124    71258      295

 7:27:15   1 50   8.6    588541      55      9                 4860         470838   102409
  10.0% 15  08  57.7%      3058      24     43    1042     102    1171          204    98813        4

 7:27:30   1 50   8.6    595083      62      3                 4768         478786   106250
   9.3% 15  08  57.4%      2963      38     70    1019     106    1172          212    98302       23
                                                                                  +
F2=Cmd Line  F3=Exit  F4=Prompt for text  F6=ODP Overview  F7=File Statistics  F8=File Analysis  F10=Details  F11=Show Full Name
```

Details for Job report is requested by using F10 (or F22) from the Job Summary Report.

The “Details for Job” report shows the resource usage details per interval for the job. The column headings are self-explanatory. In addition, some are also explained in the section “Definitions and Frequently Asked Questions”.

Note that this report is based on interval data and not on HotSpot data as the other reports that supply additional job-level information.

Only intervals showing a certain minimum use of resources (CPU time or I/Os) are included. The rules for keeping the individual interval records for a job are documented under GiAPA Menu Option 78 (Installation parameters).

The report is useful for looking at peaks and verifying whether a job used fluctuating resources, or was more stable throughout the run. The information provided here for the different intervals can be matched up against the reports based on HotSpot data, where the point of time can be used to align the data correctly.

If the job belongs to the type that temporarily is assigned to different users (other than the user name within the Job-ID), then the current user name is shown in the column on the right.

The function keys can be used both to return to the Job Performance Summary report and also offer links to the other job-level analysis reports.

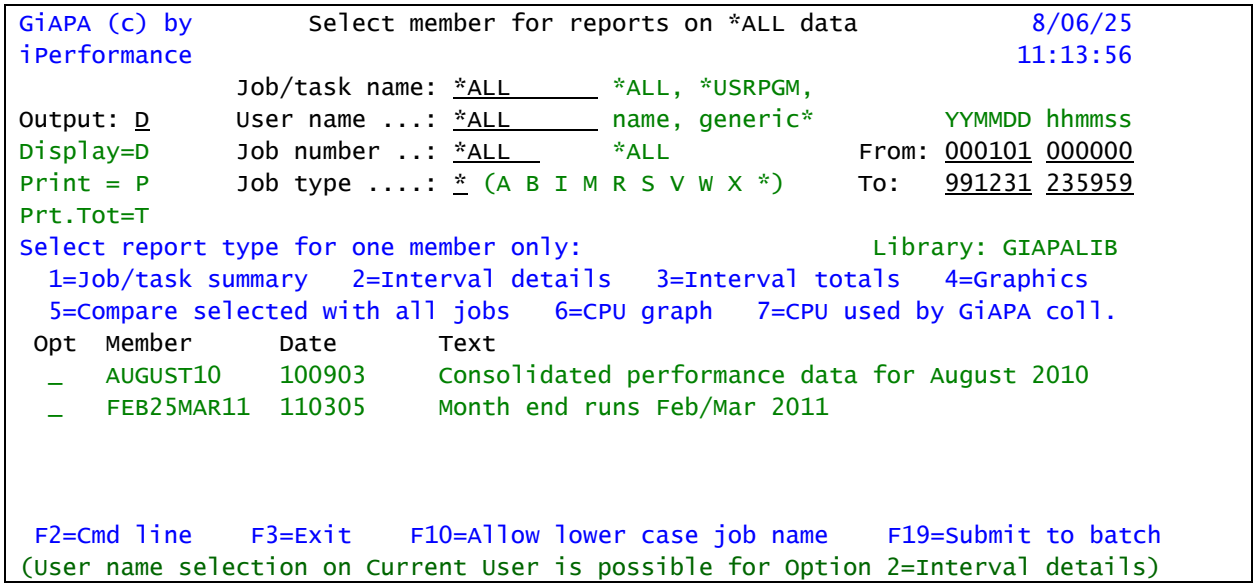
GiAPA Menu Option 16: Reports on *ALL data (when kept)

To minimize the volume of data stored (and to ensure rapid report generation) GiAPA will in the final standard output files only keep interval detail and summary records for jobs that used enough resources that they might be interesting from a performance analysis point of view.

During the expansion and analysis run (menu Option 14) detail records per interval are generated for all jobs and followed by a job summary record for every job. Subsequently, the values defined under GiAPA Menu Option 78 (Installation parameters) are used to determine which records must be selected for the final output files, after which the file members containing all details normally will be deleted.

At times, all the detailed data may be required. In such cases specify "*YES" in the "Keep all detailrecs?" parameter when submitting the data expansion run (menu Option 14). This will cause the members containing all the expanded records to be kept, in which case GiAPA Menu Option 16 can be used to generate reports based on this detailed level of data. Option 16 will cause the selection panel below to appear.

On rare occasions a user written query to extract special information from this detailed data stored in members within in the files GiAPA143P1 and GIAPA144P1 may be required.



Output can be either D for Display, P for Print, or T for Totals' page only. The last page of the printed version of report selection 1, 2, and 3 will indicate the report totals for the records selected. Please note that these totals are not generated for option D=Display. Since reports based on *ALL data could include millions of records, printing all the detail lines might not be necessary. Instead, option T can be selected, resulting in only the last page containing the final totals to be printed.

Record selection: Allows selection on date and time, job name, user name, job number, and/or job type. If several selections are used, they will be AND connected (must all be valid to select a record). If job number is not specified, selection is assumed to be generic and CPU percentage will therefore be calculated in percentage of the total CPU capacity for the LPAR. For selection 2 (Interval details) selection on user name also works on **Current User** name!

From and **To** date and time: if you leave the from/to fields unchanged, no date/time related selection is made. If you enter date and time values, only jobs that resulted in use of resources within the specified date/time range will be selected.

Selection based on time only is also possible. If you have expanded data from e.g. an entire week and you want to have totals for certain jobs between 10 and 11 in the morning for all days, simply leave the default values unchanged in the from and to date fields, and specify the wanted time interval in the from and to time fields. As an alternative you can enter zero in the date fields, and specify selections using the time fields alone.

Job/task name may contain the special value ***USRPGM** enabling a more complex selection of jobs, in which case the name of a user written selection sub-program stored in GIAPALIB must be specified in the **User name** field. The user sub-program will be called for each input record with job name, user name, and job type as input parameters, and a one-byte yes/no return parameter indicating whether the record should be selected.

The *USRPGM function is useful e.g. for capacity planning analysis, because it can be used to show total usage and peaks for a group of jobs or users. Please see further documentation of this advanced function in the source code of the example RPG subprogram supplied in source file GIAPALIB/GIAPAEXAMP member GIAPASELEC.

Important note to date/time selection: Please note that since the **Job/task summary report** is based on job summary records, the **total** use of resources for a job (spanning the time data was collected on a given day) will be included in the report, when report selection 1=Job/task summary is selected. Example: if time range 15:00:00 – 16:00:00 is specified and a batch job was running from 10:15:30 to 15:01:22 then the total resource usage for the job is included.

To obtain a “clear cut”, only showing resources used within the selected time period, use report selection 2=Interval details, or make a separate data expansion and analysis specifying the wanted from and to range.

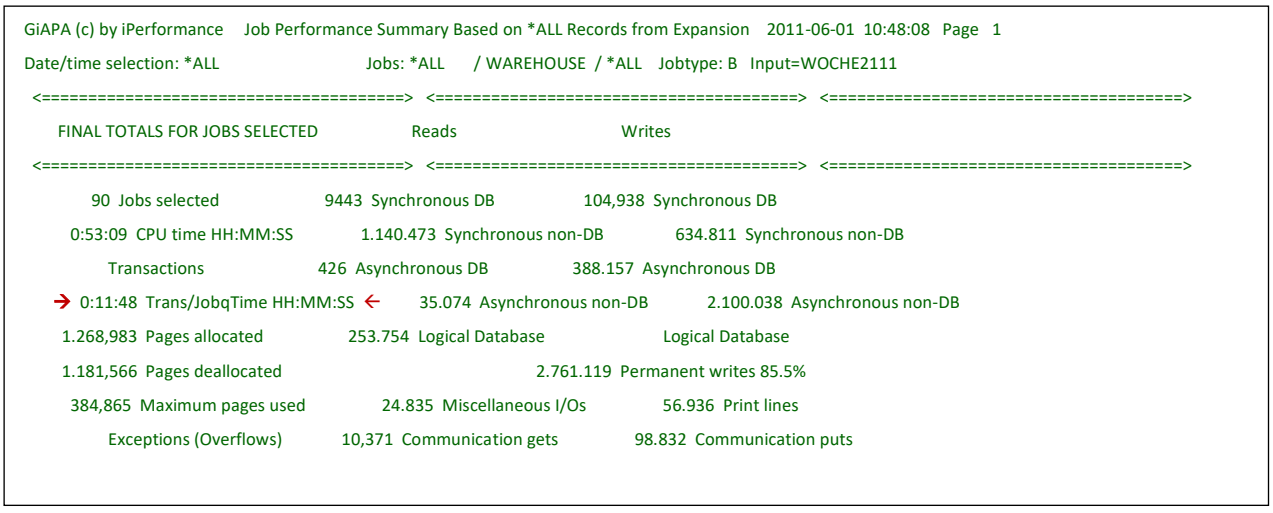
F19: Provided that the detailed data could be millions of records, F19 may be used to submit the report generation to a batch job.

Keeping *ALL details provide great many detailed analysis options: below please find one example:

We want to know total job queue wait time for jobs with user name WAREHOUSE.

Specify output type = T (totals only), user name = WAREHOUSE, and job type = B,
and select report type 1 = Job/task summary.

The result could look like this:



Job/Task Summary Based on *ALL Records (Selection 1)

This is an example of the job performance summary report for *ALL jobs. It has a similar layout as the reports obtained using Option 15, but in addition it includes jobs using minimal resources.

```
GiAPA (c) by iPerformance  Job Performance Summary Based on *ALL Records from Expansion  Input=V2TESTDATA     8/06/25 11:47:59
Date/time selection: *ALL                    Jobs: QPADEV*    Users: *ALL     Jobtype: *  Input=V2TESTDATA
JobName   UserName  RunTime  Typ Itvs ActualUsr   CPUtime   %  Prio    Logical    Physical  Trans Tr/JQ-time  Print  Ovfl.
JobNbr HotSp  RunDate from/to Pool Thrd MaxPages  H:MM:SS.s  CPU rity     I/Os       I/Os  actions h:mm:ss   lines  1000s
QPADEV00VF BRYWPDJ   14:02:58  I   60            0.2   0.0 20       849      1,216    26     4          4
QPADEV00VF BUHVAMT   16:11:51  I    4            0.9   1.4 20    11,211      2,969    22    16          5
QPADEV00VF LIVSIEP   09:41:16  I   57            4.4   5.3 20    55,864      3,905   156    39        239
QPADEV000B CPQCABS   21:41:00? I   64            0.2   0.0 20       829        360    99    10         72
QPADEV000B DUBFADL   12:03:20  I   49           12.1   1.7 20   171,949     10,727   474  1:09    2,555  2226
QPADEV000B IEVAERZ   08:27:17  I    8            7.1   6.0 20   113,378      9,485   190    57        6   601
QPADEV000B IEVAERZ   09:09:05  I   62           15.0   5.9 20   242,237     18,693   486  1:41        6  1734
(Additional lines have been removed for brevity)
F2=Cmd line    F3=Exit    F11=Show/hide lines with maximum values
```

Functions keys – F11: Show/hide lines with maximum values. Below is the first part of the same report that also shows the “Max” lines. The second column heading line (JobNbr, HotSp, ...) contains headings for line 2.

```
GiAPA (c) by iPerformance  Job Performance Summary Based on *ALL Records from Expansion  Input=V2TESTDATA     8/06/25 11:52:43
Date/time selection: *ALL                    Jobs: QPADEV*    Users: *ALL     Jobtype: *  Input=V2TESTDATA
JobName   UserName  RunTime  Typ Itvs ActualUsr   CPUtime   %  Prio    Logical    Physical  Trans Tr/JQ-time  Print  Ovfl.
JobNbr HotSp  RunDate from/to Pool Thrd MaxPages  H:MM:SS.s  CPU rity     I/Os       I/Os  actions h:mm:ss   lines  1000s
QPADEV00VF BRYWPDJ   14:02:58  I   60            0.2   0.0 20       849      1,216    26     4          4
320887      08-06-14 16:08:21  03   1     3,178 Max:  .1    .5 20       339        739     6     2          4

QPADEV00VF BUHVAMT   16:11:51  I    4            0.9   1.4 20    11,211      2,969    22    16          5
326385      08-06-14 16:20:24  03   1     8,085 Max:  .2   1.5 20     3,216        894     4     6          2

QPADEV00VF LIVSIEP   09:41:16  I   57            4.4   5.3 20    55,864      3,905   156    39        239
304766   195 08-06-14 12:00:46  03   1    11,758 Max:  .8   5.3 20    12,727        950    18     3         38
```

At the end of the printed report, or if T for totals only is selected, a total page similar to that shown on the previous page appears.

Please note that the values shown on the total’s page do not only represent the total resources used within any time interval selected, if date/time selection was used for report type 1. They represent the summary of all the total use of resources (inside and outside the time interval) for jobs that were active within (parts of) the selected interval. I.e, if a one-hour period is selected and a job started one minute before the end of that interval and ran for 3 hours, all the resources used for that job will be included in the totals.

Therefore, time selection may be relevant for restriction of jobs to be listed on the above report, but be rather misleading for the Totals’ page shown below. In this case, report type 2 would be a better choice, since it would only include data from intervals within the selected time on the summaries on the Totals’ page.

Interval Details Based on *ALL Records (Selection 2)

When requesting the “Interval Details” report data based on *ALL records (shown below), selection on time range or job id is recommended to limit the amount of output. Any job using just one millisecond or one I/O within a 15 seconds interval is included. The report can be useful for very detailed analysis of how a job performs.

```
GiAPA (c) by iPerformance      Interval Details Based on *ALL Records from Expansion   Input=V2TESTDATA   13:27:41   8/06/25
Date/time selection: 08-06-14 18:48:00 - 99-12-31 23:59:59   Jobs: *ALL      Users: *ALL      Jobtype: *   Input=V2TESTDATA
JobName   UserName   JobNbr  Tp JobMSec Nbr.Trans     LDBR     SDBR    SNDBR     ADBR    ANDBR  PagesAlc  Misc.I/O Curr.User
  Date and time   ITV CPU% Poo Pty  CPU%  HH:MM:SS     LDBW     SDBW    SNDBW     ADBW    ANDBW  PagesDlc   Perm.W  PrtLines
UKCUSTEXP  CIELADMGB 357725  B     51            72           10              1,096      4
08-06-14 18:48:00  15   4%  09 50  0.3%           106      66     159      16         1,096     229      38

QPADEV000B KWICAGS   330388  I      6      7      52      9      3                45
08-06-14 18:48:00  15   4%  03 20  0.0%       1            1                      35       1

QZDASOINIT QUSER     324748  I     50      10     201      13     33             2     632      14 AIRSQLLZ
08-06-14 18:48:00  15   4%  03 20  0.3%       2      2      22      130     54      27      632      226
                (Additional lines have been removed for brevity)
F2=Cmd line   F3=Exit   F6=Only show jobs using > 4.0 % CPU
```

F6=Only show jobs using > 4.0 % CPU should be self-explanatory. The jobs so selected will be sorted in a descending order based on CPU usage within each interval.

At the end of the printed report, or if T for totals only is selected, the page below appears. Combined with selection on time and/or selection of (group of) job(s), these summaries provide very exact information about the total amount of resources used.

```
GiAPA (c) by iPerformance      Interval Details Based on *ALL Records from Expansion      2008-06-25  13:32:36   Page   1
Date/time selection: 08-06-14 18:48:00 - 08-06-14 21:59:59   Jobs: *ALL      Users: *ALL      Jobtype: B   Input=V2TESTDATA
 <======================================>  <======================================>  <====================================
   FINAL TOTALS FOR INTERVALS SELECTED              Reads                          Writes
 <======================================>  <======================================>  <====================================
      253,203  Records selected            4,946,431  Synchronous DB            1,146,651  Synchronous DB
      9:28:34  CPU time HH:MM:SS           4,361,625  Synchronous non-DB           10,239,929  Synchronous non-DB
           Transactions                 9,569,250  Asynchronous DB             4,600,099  Asynchronous DB
           Transact. time HH:MM:SS         41,466  Asynchronous non-DB           2,639,655  Asynchronous non-DB
   802,890,049  Pages allocated            218,861,934  Logical Database            1,649,155  Logical Database
   888,753,407  Pages deallocated                                  16,355,644  Permanent writes 87.8%
                               22,172,528  Miscellaneous I/Os           1,270,561  Print lines
      32,456  Overflow exceptions             1,370  Communication gets              1,363  Communication puts
```

Interval Totals Based on *ALL Records (Selection 3)

This report can e.g. be used to demonstrate the total use of resources for a group of jobs to indicate the peak load and peak hours for an application. CPU % is in % of total CPU capacity.

```
GiAPA (c) by iPerformance      Interval Totals Based on *ALL Records from Expansion      Input=DTA       7/12/06 17:47:02
Date/time selection: *ALL                      Jobs: A206*      Users: *ALL      Jobtype: *   Input=DTA

YY-MM-DD TOTCPU JOBCPU   Logical DB   Sync.DB    Sync.non-DB   Async.DB    AsyncNonDB Communic.  Misc. I/O   Prtlines   Nbr.Trans
HH:MM:SS #CPU   #Jobs   read/write   read/write  read/write    read/write  read/write get/put    Perm.Write   Overflows  Trans.time
07-11-22 39.5% 24.4%       614        96        136       285                  1,611
11:56:45   2      9          3         2         56        24       178              116
07-11-22 27.6%  7.9%       281        43        121       91                   1,554
11:57:00   2      9          5         2         10        43       90               99
07-11-22 24.7% 14.8%       526        26        112       626                  1,518
11:57:15   2      9          7                   16        28       100              86
07-11-22 37.8% 17.5%       442        55        185       435                    946
11:57:30   2     11          6         6         24        9        114              84
```

Graphics Based on *ALL Records (Selection 4)

When requesting “Graphics” (report type 4) the following selection screen appears, allowing a selection from one to five resources for the data to be written to a file serving as input for generation of a diagram. The date/time and/or job selections specified when requesting report type 4 is displayed in the headings.

```
GiAPA (c) by      Select resource type(s) for GiAPA Graphics          17-01-06
iPerformance       Library: GIAPATOOLS  Member: M3MOVEX_T             16:57:08

From: 000101 000000  Job: *ALL       User: *ALL       Nbr: *ALL    Job Type: *
  To: 991231 235959
                            T T=Totals, A=Avg/15Sec, M=Max/15Sec, B=Both A + M
Select 1-5 resources:       2 Use 2nd Y-Axis for 0, 1 or 2 last selections?

<---Various------>  <-Physical Reads->  <-Physical writes->  <-Logical I/Os-->
4 Average CPU pct    _ DB synchronous     _ DB synchronous      _ Reads
5 Maximum CPU pct    _ DB async.          _ DB async.           _ Writes
_ CPU seconds        _ NDB synchron.      _ NDB synchron.       _ Miscell. I/Os
_ Average CPW value  _ NDB async.         _ NDB async.          1 All log. I/Os
_ Print lines        _ All DB             _ All DB
_ Transactions       _ All non-DB         _ All Non-DB          <Page Allocation>
_ Response time      _ All synchron.      _ All synchron.       _ Pages allocated
_ Overflows                               _ Permanent           _ Pages dealloc.
                                          _ Non-permanent       _ Pages used
                           2 All phys.DB I/Os
                           _ All sync. I/Os
                           3 All physical I/Os
```

You may request the following data value parameters:

- T = Totals for the job(s) within the date/time selected
- A = Average per 15 seconds interval
- M = Maximum found in any 15 seconds interval, or
- B = Both average and maximum.

In case resource types with very different values are selected (e.g. CPU percentages and number of I/Os) it will make sense to request two distinct Y-axes given that I/Os may reach millions, whereas CPU % peaks at 100. Therefore, you can specify that a secondary axis be used for the last, or for the two last resources selected.

A chart providing an overview of both logical and physical total I/Os together with both average and maximum CPU % seems to represents a frequently requested output, for which reason this selection can be specified simply by using F11 (please refer to the example above).

The left Y-axis will represent the scale for the I/Os, and the right Y-axis will display CPU %, resulting in an output as shown on the next page.

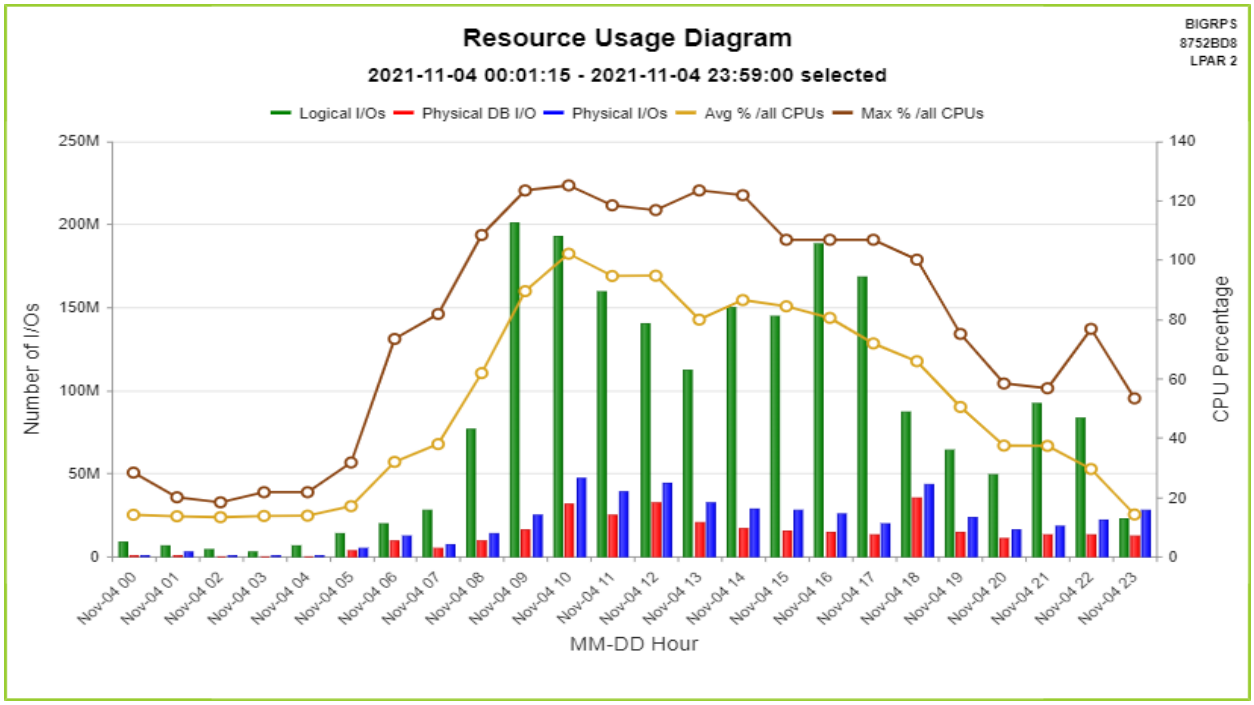
The X-axis is calculated automatically by the program, depending on the time span of the data selected. The X-axis will be as detailed as possible without exceeding 60 different values. The scale could be 15 seconds intervals, minutes, 10 minutes, hours, days, or months.

If CPU seconds are selected, the value will be converted to CPU hours if the average field value exceeds three hours.

F19 may be used to submit a batch job for the generation of the input data file for the graphics. When running interactively, the following display indicates the progress:

```
GiAPA (c) by                Selecting and Processing Records              14/02/06
iPerformance                        for GiAPA Graphics                    10:47:27

                    Input data is being read from file GIAPA143P1

                  2,310,443 records found in input member M3MOVEX_T

                  1,353,246 records processed        79,865 selected

      ****************************************

Selections:

  Job name    User name   JobNbr JobType  From date/time       To date/time

  *ALL        MIMIXOWN    *ALL       *    13-03-08 00:00:00  13-03-08 23:59:59
```

With many different selection possibilities, options and chart types the output can appear in numerous different ways. The example below is produced using F11 options like shown above.



Please note that the chart type (column-and-line, pie, donut, bar, etc.) as well as the chart titles, legends for X- and Y-axis, and color palette used may be modified from GiAPA menu Option 28.

Generation of a graph usually completes within seconds, after which the result is displayed immediately. Should that not be the case, please see the last page of this manual.

Compare Selected Jobs and *ALL Jobs (Selection 5)

GiAPA (c) by iPerformance — Compare selected job(s) and *ALL jobs, per interval — Input=T200DEC14 — 7/12/29 — 14:52:54

Date/time selection: 00-00-00 13:07:40 - 00-00-00 14:07:30 Jobs: SOCA* Users: *ALL Jobtype: B Input=T200DEC14

Date/Time	Jobs	CPU%	Sync.R+W	Async.R+W	Misc.I/O	Logic.R+W	Prm.Write	Communic.	Prtlines	Transact
07-12-14 Selected Job(s):	134	23.6%	12,668	1,237	431	156,444	799			
13:07:45 Total *ALL jobs:	195	47.4%	25,380	9,963	4,417	340,871	6,436			
07-12-14 Selected Job(s):	142	41.7%	12,912	1,613	264	278,191	1,227			
13:08:00 Total *ALL jobs:	195	72.1%	24,547	11,143	3,082	477,405	7,779			
07-12-14 Selected Job(s):	133	48.9%	13,382	1,625	471	303,709	1,039			
13:08:15 Total *ALL jobs:	189	81.7%	25,661	11,428	5,636	523,523	7,292			
07-12-14 Selected Job(s):	129	40.7%	15,320	3,061	1,124	274,807	1,286			
13:08:30 Total *ALL jobs:	186	72.5%	26,956	13,889	5,047	499,826	8,530			
07-12-14 Selected Job(s):	123	29.0%	16,916	2,210	1,122	265,409	1,195			
13:08:45 Total *ALL jobs:	183	52.5%	28,471	12,709	4,693	367,344	8,009	28		
07-12-14 Selected Job(s):	128	25.9%	17,101	1,907	1,639	204,760	1,841			
13:09:00 Total *ALL jobs:	187	45.9%	28,249	12,378	2,927	255,640	9,040	202	6	
07-12-14 Selected Job(s):	109	22.5%	16,857	1,455	3,246	149,329	881			
13:09:15 Total *ALL jobs:	168	38.6%	28,539	9,661	5,805	190,702	6,818	2		
07-12-14 Selected Job(s):	130	24.1%	16,701	1,550	261	141,030	806			
13:09:30 Total *ALL jobs:	195	42.3%	28,590	11,529	1,746	187,168	7,970	4	449	7

This report compares a group of selected jobs (shown in the first line) with the total use of resources by all jobs (including the selected) within an interval.

The report is useful in cases where the users of an application complain about periodically bad response times, since it – when the selected group of jobs use many resources of a given type - can show if other jobs already have stressed the computer close to the limits, thus resulting in a delay e.g. due to lack of available CPU resources.

Note that this is an example of a selection on time without date – the date from/to fields contain zeros and are ignored, only the from/to hour is used in the shown selection.

CPU % Statistics Based on *ALL Records (Selection 6)

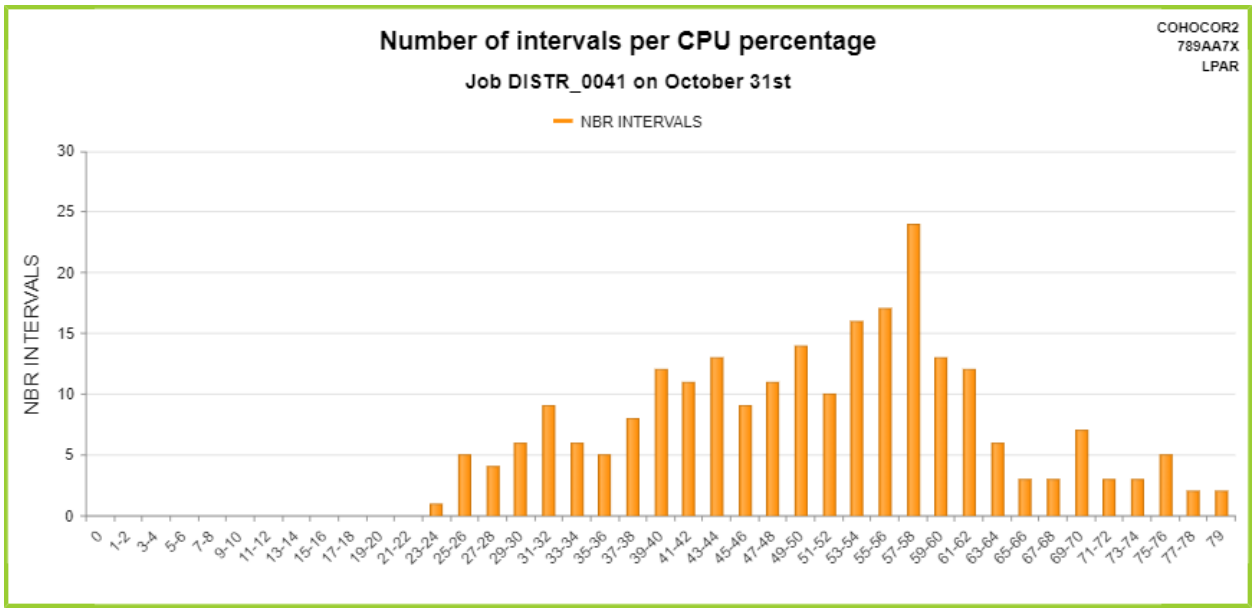
If only one job number is selected for this graphical report, then the resulting CPU % shows percent of one CPU only. Otherwise, the CPU % is calculated based on the total CPU capacity of the LPAR.

The below progress screen is shown while the report is being generated.

```
GiAPA (c) by     Processing Records for CPU Statistics Graphics        22-01-13
iPerformance                                                           18:24:38
                  Input data is being read from file GIAPA143P1

    ************************************************************************

                                                    Highest value found was
                                                        53,617 milliseconds
    Selection on generic job name DISTR_0041         at 2018-10-31 16:20:00


                                                     240 intervals selected

    All intervals in member specified selected.
    CPU % calculated of total CPU capacity.

    For details on selection of records and calculation of CPU percentage
    please refer to the GiAPA User manual.

             Data for graphics has been stored in file GIAPA16501
              Now use GiAPA Menu option 27 to generate the chart

F2=Cmd Line      F3=Exit
```

Use GiAPA Menu Option 27 to generate the graph, which provides an overview of CPU used by the selected jobs. It illustrates for each CPU % (or group of CPU %) the number of 15-seconds collection intervals where the selected job(s) were using that CPU percentage.

In the example below it is apparent that the job selected mainly uses between 40 and 60 % CPU. The peak CPU usage exceeding 65 % was only found in rather few intervals.



CPU used by GiAPA data collection (Selection 7)

GiAPA jobs will normally use somewhat less than 0,1 % CPU when collecting performance data every 15 seconds for all jobs and tasks running. The use of CPU depends primarily on the number of HotSpots generated – the retrieval of file usage and call stack data for CPU intensive jobs will often use more than the process of the resource usage data for all jobs and tasks.

CPU used for processing HotSpots will increase if

- The threshold value for when to trigger HotSpots (command GiAPA110 or GiAPA Menu option 11) is set to a lower value, causing more HotSpots to be generated.
- The limit for how many threads should be processed is set very high, or if threads in wait state should be included – both defined within the user installation parameters.

```
GiAPA (c) by          CPU used by GiAPA Data Collection              24-01-13
iPerformance                                                         09:43:38
                    Input = NIGHTBATCH in Library GIAPALIB     SrlNbr  06E84CT
                      Testing GiAPA CPU Usage Statistics       SysName POWER720
First/Last interval                                            LPAR    001
2024-01-12 21:54:15    Processor capacity   1.00 CPUs
2024-01-13 01:10:15                                    Hour:mm:ss.Msec CPUpct*)
                                                       --------------  ------
         785 data collection intervals, elapsed time      3:16:15

             Total CPU used by all jobs                   2:23:18.000  73.0191

     709,309 resource usage records were processed by
                  performance collector job GIAPAPFCOL          0.829    .0070
             Job accounting data fetched by GIAPAACGJR          0.015    .0001
             Job restarting GiAPA if needed GIAPARESTR          0.090    .0007
       1,429 HotSpots processed by   4 GIAPAHOTSP jobs          0.880    .0074

             Total CPU used by GiAPA's data collection          1.814    .0154

SQL Observer (Job Watcher, Plan Cache Dump) GIAPAJWCOL          6.927    .0588

F2=Command line     F3=Exit     *) Calculated in pct of total LPAR CPU capacity
```

GiAPA Menu Option 17: Job or User Name Summary Report

Use of Option 17 will display the following panel:

```
GiAPA (c) by    Select Member for Summary by Job or User Name          8/06/25
iPerformance                                                         11:19:08

Library: GIAPALIB         Output D D=Display      Sort by C N=Name
                                  P=Print                  C=CPU used
Select one member only:                                    O=Occurrence count
 1=Job name summary   2=User name summary                  L=Logical I/Os
                                                           P=Physical I/Os
Opt  Member      Date        Text
 _   AUGUST07    070903      Consolidated performance data for August 2007
 _   SEP15MORN   070915      Performance data from the morning of September 15th
 _   WOCHE1808   080520      GiAPA Daten Woche 18 = 28 Apr - 04 Mai 2008
 _   VECKA3907   070927      Prestanda mätning vecka 39 - 2007
```

(Additional lines have been removed for brevity)

The specifications required on this panel are explained in the concurrent text. Pressing ENTER will generate either a job name or a user name summary report. The reports are similar except for the second column, which either contains a job name or a user name.

Job Name Summary report does not include job names starting with QPADEV, since these names typically will be used for different purposes making the summary meaningless.

Only job/user names found in more than one job and showing a certain minimum use of CPU time are included. The rules for generation of job and user name summaries are defined under GiAPA Menu Option 78 (Installation parameters).

Below an example of a user name summary report – the job name summary looks the same, except that the job name has replaced the user name. Please note that tasks (internal operating system jobs) have blanks in the user name.

```
GiAPA (c) by          User Name Summary Sorted by CPU Usage     Input = V2TESTDATA          8/06/25
iPerformance                                                    11:23:35
 Count Username   Pr Typ  CPU time  Synchronous Asynchronous     Logical  Misc.I/O +  Communicat.    Print  Overflow
           io     HH:MM:SS    I/Os       I/Os       DB I/Os  Prm.write  puts+gets     lines  exceptions
24,483 MIMIXOWN      B   8:55:54  40,123,485  6,377,973  547,801,871  13,009,554                834,999
          50 Max:  87.6%    57,116     35,599     377,819     16,185                15,954
  612 KREZFAS        B   5:46:51   2,276,364   2,129,822  195,848,996  23,461,933           323,607      123
          50 Max:  66.9%    13,907     18,624     462,300     88,097           259,031        2
 1,820 ZAMZITJ       B   4:10:53   3,207,989   9,471,181      17,077   4,001,895             476
          50 Max:  68.4%    22,421     32,747       1,680     10,322               476
14,456         00 V    2:34:40  11,269,348  20,111,229           0  16,680,312
          99 Max:  27.3%    14,216    323,136           0    177,187
11,713 QMQM          A   1:42:58  23,946,220  12,332,095  119,988,742  32,875,750           508,098    181,240
          50 Max:  57.4%    31,601     26,733     739,229     40,209           84,175     1,346
 1,336 KGBADMGB      B     52:54     999,524     582,154  227,131,383   3,128,238            54,482     59,600
          50 Max:  66.6%    24,503     22,100   1,010,159     40,293            5,612       264
11,102 I2WEB         B     19:24   4,323,473   6,515,419   1,427,730   5,290,843
          50 Max:   7.0%     8,748     73,236      30,827      3,478
   12 QYPSJSVR  16 B     12:01         14          0         1,105          0
          31 Max:  33.7%        11          0         110          0
 5,801 MULTISCAN      B     10:02      64,456   1,078,308      188,716      46,449               37
          40 Max:  21.5%     1,063      7,775       1,080        433               37
                                  (Additional lines have been removed for brevity)
 F2=Cmd Line    F3=Exit    F14=Create Graphics                        Enter=Go to top of subfile
```

F14=Create graph data provides different possibilities for creating a chart. The flexibility available here allows the user to select which data field should be selected, and how many records should be generated for the graphics data file.

Two fields are written to the graphics data file, one key field and one data field. The job name or user name (depending on report selected) is always used as key field. The data field is the currently selected sort criteria field. As an example: if the subfile is sorted by logical I/Os, number of logical I/Os are passed to the graphics data file. However, if the currently displayed data is sorted on N=Name, use of F14 will have no effect.

The records for the graph are always selected from the beginning of the subfile, and the user can determine how many records should be included by positioning the cursor on a record in the subfile to indicate the last record selected. If the cursor is positioned outside the subfile, all records currently loaded in the subfile are selected (e.g. one subfile page of records if “Page down” was not used yet). In either case only a maximum of 50 records are written into the graphics data file, since a very large number of records are not suited for creating charts.

Please note that any graphics data file that was created earlier using the current subfile sort criteria as data field is automatically overwritten, unless it was renamed using menu option 28.

GiAPA Menu Option 18: HotSpot Count Summaries

These reports are based on HotSpot call stack data. The reports show in descending order by occurrence count which programs, jobs, or users were recorded in the HotSpots. A detailed report showing the job ID details within programs sorted alphabetically is also available. The four reports are selected by entering the sought report number in front of one of the data collection members.

```
GiAPA (c) by      Select member for GiAPA HotSpot count report          8/06/25
iPerformance                                                          11:34:36

Select one member only:         D=Display, P=Print: D        Library: GIAPALIB
 1=Non-Q Pgm, 2=Job, 3=User, 4=Detailed, 5=Act.Pgm/Class, 6=Act.Proc/Method

Opt  Member      Date        Text
 _   AUGUST07    070903      Consolidated performance data for August 2004
 _   SEP15MORN   070915      Performance data from the morning of September 15th
 _   WOCHE1808   080520      GiAPA Daten Woche 18 = 28 Apr - 04 Mai 2008
 _   VECKA3907   070927      Prestanda mätning vecka 39 - 2007
                      (Additional lines have been removed for brevity)
```

Below please find three examples of the HotSpots Summary reports.

HotSpot Summary by Job, User, Program, or Class

```
GiAPA (c) by        GiAPA HotSpot Summary by Program          8/06/26
iPerformance      Input = TESTDATA in Library GIAPALIB        12:07:27
    368 HotSpots without NonQ-Pgm
 HotSpots Library   Program   Program Text
    531 DGXRBP     XPREY2     Text for program XPREY2
    212 DGXRBP     XPRPE2     Object description text for XPRPE2
    165 DGXRBP     XPRSI2GE   Program XPRSI2GE description
    122 DGXRBP     XPREY3     Text for program XPREY3
     97 DGXRBP     XPRPE2GB   Object Description Text for XPRPE2GB
     96 HF71EVJ    MIM190     Program MIM190 description
     81 DJFMRBP    DPRTC2G2   Text for program DPRTC2G2
     76 HF71EVJ    EVF241     Object Description Text for EVF241
     67 DGXRBP     XPRPI2G2   *** Program not found
     58 DETGRBP    DTDHU2     Text for program DTDHU2
                      (Additional lines have been removed for brevity)
F2=Cmd Line  F3=Exit  F14=Graphics   Cursor on Pgm: F6=Select for Job Summary
```

F6=Select for Job Summary with the cursor positioned on an object name can be used for report selections 1, 5, and 6. It will cause the object name to be stored allowing it to be used for selecting only those jobs which have used the selected object. Please refer to the description of GiAPA Menu Option 15, Selection of Job Summary Reports.

F14=Create graph data creates a chart. The records for the graph are always selected from the beginning of the subfile, and the user can by positioning the cursor on a subfile record specify the last record to be included. If the cursor is positioned outside the subfile, all records

are selected. However, in either case a maximum of 50 records are written into the graphics data file, since a very large number of records are not suitable for creating charts.

Please note that any graphics data file that was created earlier by using F14 from the same subfile panel is automatically overwritten, unless it was renamed using menu option 28.

HotSpot Summary by User

```
GiAPA (c) by          GiAPA HotSpot Summary by User          7/10/21
iPerformance       Input = SAMPLEDATA in Library GIAPALIB       20:50:45

  HotSpot  User Name   User Profile Text
   1.222  EVJIRXFB    Froed Nielsing, Sales department, EMEA branch
    190  EVJIRSM     Cinda Rella, Sales department, EMEA branch
     96  IBKTUR12    Kahmel Ul Zut, Headquarter Accounting
     89  CSFFYSEI    S. Anta Claus, Toys Department, Greenland
     76  EVJXYIL     Tom Peng Pung, Invoicing section B
     64  CSFFGDJ     Michael A. L. Brecht, IT Operations Dept.
     42  SOSYAS      P. Ellesen, Marketing & Sales
     21  BCRICWH     A. Neuph, Science & Research Lab.
        (Additional lines have been removed for brevity)
F2=Cmd Line    F3=Exit    F14=Create Graphics
```

HotSpot Summary by last called Procedure or JAVA Method

```
GiAPA (c) by       GiAPA HotSpot Summary by last called Procedure or JAVA Method      8/06/26
iPerformance             Input = V2TESTDATA in Library GIAPAKNE              12:27:39
  9,678 HotSpots without active procedure or method at last call level
 HotSpots  Procedure or JAVA Method
  3.998 recv
  1.120 wait__20Qp0wPthreadConditionFP7Qp0wTcbP9Qp0wMutex
   450 sllSearch
   432 lockSlowWithInitAllowSig__9Qp0wMutexFv
   234 lvStrLenChr
   206 waittime
   117 lvWrkUxRead
    95 lvStrLen
    71 ICRTDSI
    63 waitFor__7Qp0wTcbFiT1
    51 retrieveMemberData
    47 qso_takedescriptor__FPiT1Pc
    44 GET
    43 KACIC9KN
    43 OPQRYDBM
    37 COPOH9
    33 uqDeqWithKey
    32 CALLDBMAINTFOROPENOROPTIMIZE
          (Additional lines have been removed for brevity)
    19 QDBGETMQO
F2=Cmd Line  F3=Exit  F14=Create Graphics    If cursor positioned on object name in subfile:
                            F6=Select for Job Summary
```

GiAPA Menu Option 19: Program or file performance analysis

Option 19 analyzes the data across all jobs and will in addition display/identify a program or file that only appears in a few HotSpots per job if:

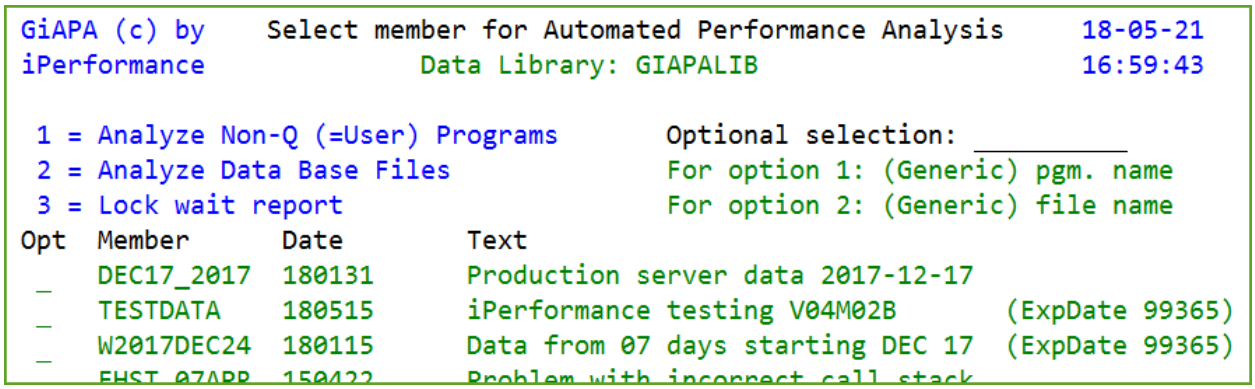
- the job is running very frequently, or
- the program or file is used by many different jobs.
-

In addition, the new analysis

- combines more information on one screen, allowing for easier identification of optimization potentials, and
- offers tips that reveal which code modifications may result in improved performance.

The overall idea is of course in line with GiAPA's original objective: to enable the average programmer or operator to become an expert in application performance optimization.

An example of the selection screen for the Automated Performance Analysis is shown below:



Selection 1: Analyze Non-Q (= User) programs

The Program Analysis requested with selection 1 displays in the title line the name and descriptive text of the last called user program. The call stack may contain additional user programs. However, this analysis does not show programs called before the last called program having a name not starting with the letter Q. In the most cases the user program shown invokes in turn the IBM program(s) actively running. The limited cases, where the user program itself is active, are of course also shown.

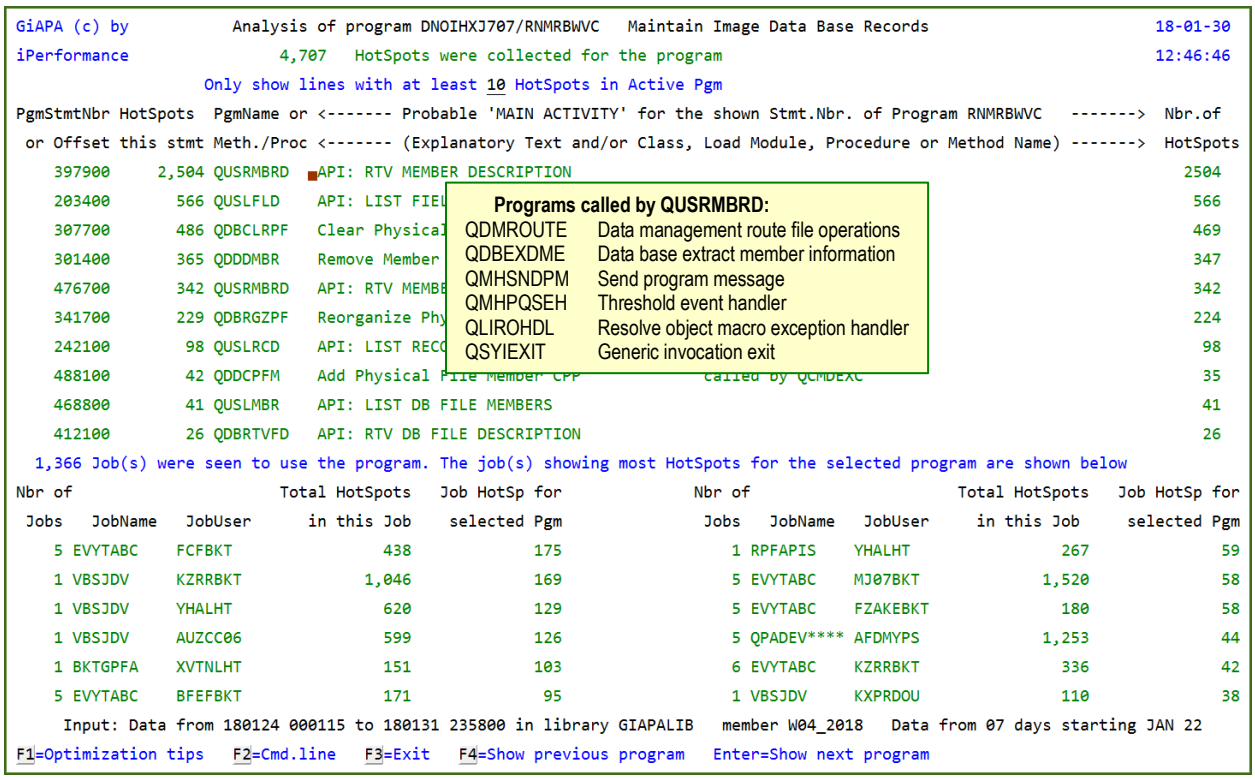
Consistent with GiAPA's core mode of action to produce exception reports, the default is set at reporting only lines with minimum 10 HotSpots. It is possible to select a lower or higher limit using the input field located in the third line.

The first subfile on the panel below contains five columns:

1. The user program statement number active when the HotSpot was collected.
2. The total number of HotSpots found for that statement number.
3. The name of the program that best describes the active function.
4. The descriptive text of the program or an explanation of the program function, if available. At times this is supplemented with the name of the class, load module, procedure and/or method.
5. The number of HotSpots for that program function.

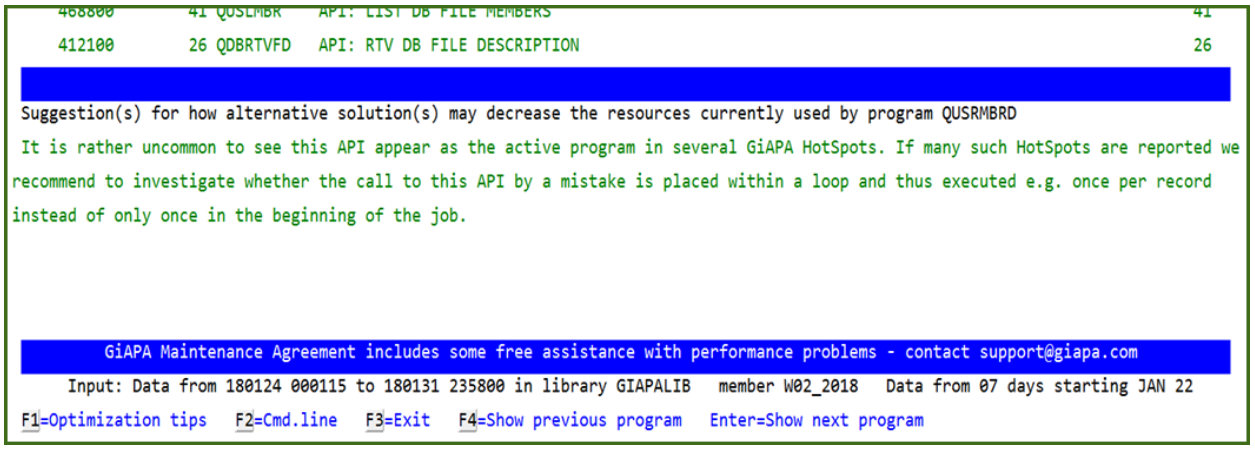
The jobs causing the most HotSpots for the analyzed program are shown in the second subfile. Please note that the program names displayed in the first subfile will often not be an actively running program. If several subprograms were invoked to deal with the function called by the user program, they are combined into one line.

Attempting to accumulate as much as possible under the main function running, GiAPA reports in the example below 2,504 HotSpots as belonging to the system API which retrieves member description data. This number includes the many cases where one of the subprograms listed in the yellow box was active. They were running under the retrieve member description API and were therefore the active program in some of the call stacks. However, showing them individually would disturb the overall picture.



In the above new example 1,366 jobs triggered 4,707 HotSpots for the program in the title line. The number of HotSpots per job was therefore on average so low that they in most cases would have gone by unnoticed. But analyzing across all jobs we see that 2,504 HotSpots, or more than half, showed different active programs called directly or indirectly by statement 3979.

F1=Optimization Tips: When the cursor is positioned on a line in the first subfile, use of F1 requests a panel that contains hints about optimization possibilities.



The above type of help texts has been defined for several programs, and more will become available in future versions as we receive feedback from our users. The most important message is maybe found in the last blue line mentioning the GiAPA support available…

At iPerformance we realize that the list of help texts is far from exhaustive. More will be added continuously to improve future versions of GiAPA and we always encourage our users to send us screen dumps of missing help texts or of text that does not correctly convey a given situation.

Selection 2: Analyze Data Base Files

This option combines a total overview of all usage collected for a file with the option to see job call stack statistics limited to the time period where the file was accessed.

```
GiAPA (c) by        Analysis of LF B.DAFAA/LJGW#3(LJGW#3)           Customer order status by class/category            18-02-01
iPerformance                      286,358 records,           0 deleted.       Journaled         Keyed access path    12:57:38

<------File opened by------> GiAPA <--Opened from/to-->  Open   Nbr.of      Logical       Logical  Miscellaneous Seq.  Rel.RecNbr
Job name   User name  JobNbr F.Nbr YYMMDD hhmmss hhmmss   for   HotSp.    DB writes      DB reads  DB operations Only     Span
     Total for     15 jobs generating HotSpots for this member    1224                1608,963,516
RPFAPIS    BCMPVDK   541100     2 180125 214945 221045 Update      66                 115,484,714                 Y        281,363
RPFAPIS    KTPMBEDJ  989368     5 180125 203515 205600 Update      84                 115,616,598                 Y        281,260
RPFAPIS    BFEFLHT   834756     3 180124 150530 154100 Update     140                 115,521,657                 Y        281,159
RPFAPIS    BFEFLHT   425878     3 180126 150500 154130 Update     147                 115,769,236                 Y        285,706
RPFAPIS    KTPMBEDJ  998340     5 180123 194500 200315 Update      74                 115,507,979                 Y        284,998
RPFAPIS    KTPMBEDJ  209480     5 180124 203200 205230 Update      82                 115,773,270                 Y        280,672
RPFAPIS    SLBSXHMA  348933     8 180122 174200 180830 Update     107                 114,849,702                 Y        280,566
RPFAPIS    BCMPVDK   221103     2 180122 212100 214100 Update      81                 115,226,839                 Y        278,670
RPFAPIS    KFBVVBXBS 271649     6 180124 221900 223815 Update      78                 115,329,765                 Y        277,447
RPFAPIS    EJPOW     130816     1 180125 000000 001145 Update      40                  71,550,513                 Y        173,919
RPFAPIS    EJPOW     130816     1 180122 235300 235945 Update      28                  42,159,348                 Y        101,682
RPFAPIS    EUDSSHSX  956780     4 180123 202545 204345 Update      73                 113,978,113                 Y        280,127
RPFAPIS    EJPOW     564640     1 180124 001830 003645 Update      74                 114,008,094                 Y        280,026
RPFAPIS    KFBVVBXBS 504605     6 180123 205200 211015 Update      74                 113,930,234                 Y        280,0 +

 I/O Statistics since           Writes           Updates           Deletes     Logical reads    Physical reads  Acc.Path Log.Reads
 IPL   18-01-07 10:52                                                         3.430,825,172            31,605     3.061,467,523

     Input: Data from 180124 000115 to 180131 235800 in library GIAPALIB   member W04_2018   Data from 07 days starting Jan 22
F2=Cmd line     F3=Exit     F4=Show previous file     F9= Show call stack info for job                        Enter=Show next file
```

Using Option 2 from the selection panel will provide a result similar to the above. Distinct basic information about the file is shown in the title lines. However, the data at the bottom of the screen may be the most relevant for assessing the total impact of accessing the given file has on the total system performance

The operating system will for all files save I/O statistics in the object description. The statistics are reset to zero at each IPL. This example shows that the last IPL was made at 10:52 AM on January 7th of 2018. Since then and until the date and time where this analysis was retrieved (shown in the lower left corner) this file was read 3,430 million times.

Overall, the data shown on this panel indicates that improvements are possible. Several jobs read records from the file around 115 million times, but despite the file being opened for updates, only reads are reported. Furthermore, the second last column indicates that all accesses seem to occur only sequentially, i.e. the records are read in the sequence in which they are stored. This would normally result in the records being read in blocks, but in this case, it is hindered by opening the file for update.

Finally, it is worth speculating why 115 million accesses are required for less than 300.000 records. However, this concern is only valid if the “sequential only” appears not to be true.

F9=Show call stack: Use of F9 with the cursor positioned on one of the job names in the subfile requests display of the call stack statistics for the job during the time span where this file was accessed. This might be much less than the total run time for the job, but in this case we want to concentrate on the impact of the I/Os of this file only.

```
GiAPA (c) by        Analysis of LF B.DAFAA/LJGW#3(LJGW#3)          Customer order status by class/category          18-02-01
iPerformance                      286,358 records,          0 deleted.       Journaled        Keyed access path    12:58:13

<------File opened by------> GiAPA <--Opened from/to-->  Open   Nbr.of      Logical       Logical  Miscellaneous Seq.  Rel.RecNbr
Job name   User name  JobNbr F.Nbr YYMMDD hhmmss hhmmss   for   HotSp.    DB writes      DB reads  DB operations Only      Span
RPFAPIS  ■ BCMPVDK    541100     2 180125 214945 221045 Update      66                115,484,714                 Y        281,363

 Summarized HotSpot call stack statistics showing the most used programs in the intervals where this job accessed the above file:
Times   % of   Active program                                  Last called user program                                Stmt.nbr.
found  HotSp.       or class                                       or class                                          or offset
   37     56   QDBGETKY   QSYS                                 LZAAVEOMWF DNODAFAWJ                                      567
               DATA BASE GET BY KEY                            Batch validation of order lines

   16     24   QSQRUN2    QSYS                                 LZAAVEOMWF DNODAFAWJ                                      592
               QSQ* = SQL functions                            Batch validation of order lines

    8     12   QDBGETMQO  QSYS                                 LZAAVEOMWF DNODAFAWJ                                      592
               SQL QUERY ENGINE (SQE) FETCHING ROWS            Batch validation of order lines

    3      4   QDBPUT     QSYS                                 LZAAVEOMWF DNODAFAWJ                                      839
               DATA BASE PUT UNBLOCKED                         Batch validation of order lines

    2      3   QSQROUTE   QSYS                                 LZAAVEOMWF DNODAFAWJ                                      592
               SQL ROUTER                                      Batch validation of order lines

F2=Cmd line     F3=Exit
```

This example clearly demonstrates that 56% of the run time is used by keyed reads of records. However, it is possible that the many I/Os partly are accessing records in other files.

Looking at the call stack analysis which we can request from the Job Performance Summary Report we would also able to see the user program statement number(s) calling the get-by-key function. The program source code would in turn reveal which file was being read.

(In the case shown above, the job CPU and elapsed time was reduced more than 50 % just by changing the file to be opened for input instead of update, and skip reading if the record wanted was the last record read.)

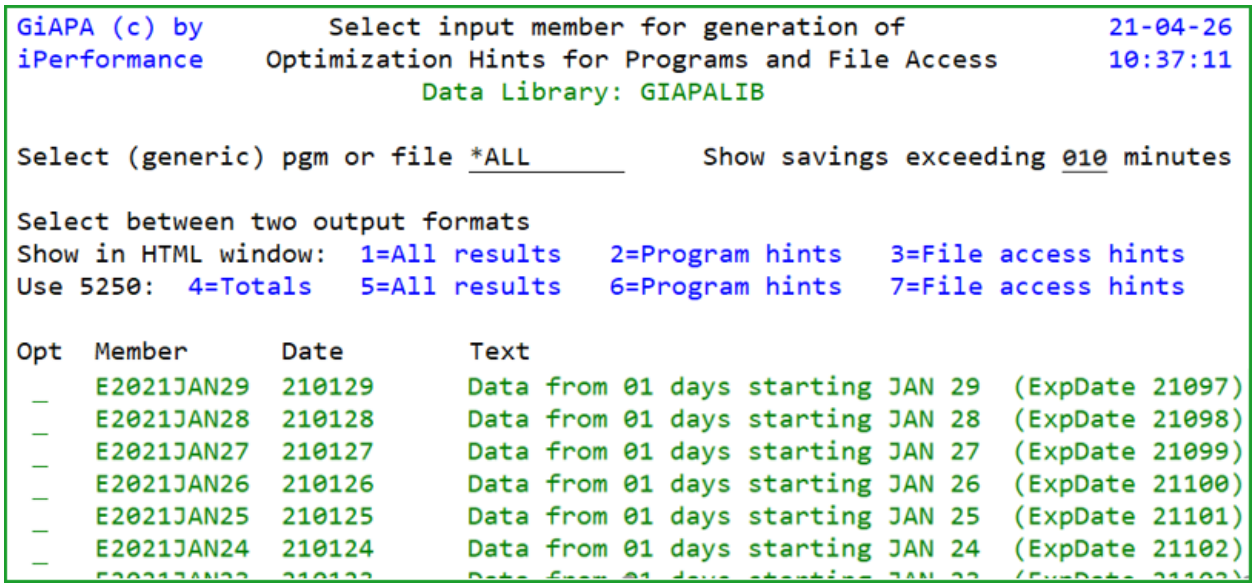
Selection 3: HotSpot call stacks with status Lock Wait.

A very straight forward report listing collected call stacks with status lock wait. Very few could be OK, but even a two-digit number of such waits might call for an explanation.

```
GiAPA (c) by          Jobs with Programs in Lock Wait Status in data member WEEK_OCT14 in library GIAPALIB          18-11-05
iPerformance                                      Member text: Week starting October 14th, 2018                  12:14:28

Locks per  User Pgm and Lib    Stmt.Nbr.   Descriptive text for job activity                    LockWait(s) found in
statement    or Class Name      or Offset     when Lock Wait Status occurred                    Job Name   User Name
     21   NCCD100    RIBTCP       18300   Copy Spooled File CPP                                NCCT071    ASINCRON
            (Same as above)                                                                    NCCT071    ASINCRON
      8   NCCD100    RIBTCP       17800   Create Physical File CPP                             WRTA02701  ASINCRON
            (Same as above)                                                                    VPCA02701  ASINCRON
```

GiAPA Menu Option 20: Program and File Optimization Hints

This is GiAPA's most advanced performance analysis, producing suggestions for how to improve program and/or file access performance 100 % automatically. Menu option 20 displays the following panel:



Select pgm or file may be used to only display optimization hints for a given program or file. Generic selection is also supported.

Show savings exceeding XX minutes specifes the minimum number of minutes of estimated saved run time that should be selected.

Select between two available formats that both show the same data by specifying:

- **1**, **2**, or **3** for HTML format, where a column chart shows the estimated savings obtainable by implementing the optimization suggested by GiAPA.
- **5**, **6**, or **7** for a 5250 green screen format providing the data in a rawer format.

Select the data collection member to be used as input by specifying

- **1** or **5** for obtaining optimization hints for both programs and file access.,
- **2** or **6** for results showing optimization potential for programs
- **3** or **7** for results showing optimization potential for accessing data bases

Or specify "**4**" to get a panel showing the total time for the potential savings found.

We trust that the optimization hint examples shown below both are self-documenting and provide sufficient information on order to

- enable management to decide if the suggested improvement should be implemented, and
- inform the programmer where to find the statement(s) to be modified, and which alternative program code may result in an improved performance.

For this task GiAPA also uses exception reporting and hence, does not report improvements where the savings are less than two seconds per job. This limit might be too tight for very tiny jobs running thousands of times. However, GiAPA Menu option 19 can be used to analyze these types of jobs.

Examples of automatically produced optimization hints for programs and file access:

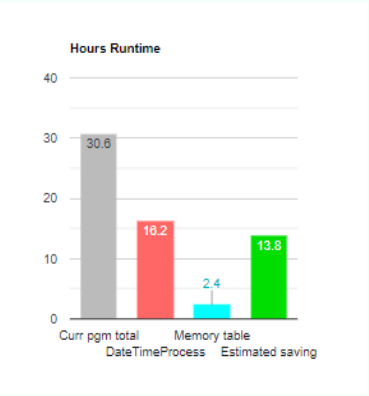
GiAPA by iPerformance

Program Optimization Hint

System: MAINSERV
781X22C LPAR 021

95.3 hours of data collected starting 2021-01-29 at 00:01

Program used	RWONMN/OMENPDHPZ	Calculate interest for outstanding invoices
Statement number	46900	
GiAPA detected	Date/time conversion or calculation found in 3907 HotSpots	
Job and user	UBSTVABZY4 KVKZKDV (4 jobs) UBSTVABZY7 KVKZKDV (4 jobs)	
Estimated saving	85 % of DATETIME = 830 minutes run time	
Effort required	Probably < 7 hours programmer time (test not included)	



Technical explanation

The process needed for format conversions or before and after time/date calculations are quite CPU intensive

Tips on how to optimize the performance

Date/Time conversions, and calculations on date and time fields may be convenient to use, but are rather CPU intensive functions. An example is interest calculation starting with finding the number of days between two dates. If this is done for each record in a batch run, the date field calculation may be responsible for around half the CPU time used by the program. Most often such routines calculate the days elapsed between an older date and today's date, in which case the results of the calculations can be stored in an array using the older date as key. Subsequent date calculations can then be replaced by much faster binary table look-ups in the array.

Print all pages Print page

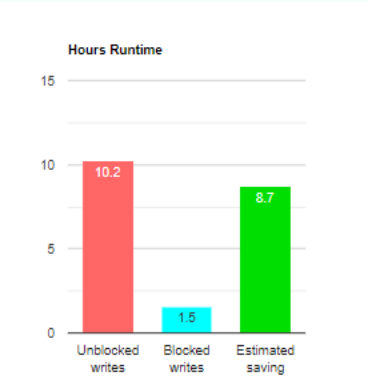
GiAPA by iPerformance

File Access Optimization Hint

System: MAINSERV
781X22C LPAR 021

95.3 hours of data collected starting 2021-01-29 at 00:01

File accessed	QTEMP/FEWXRNMP	Transactions ready for main update run
Records in file	50,513,446 (Estimate based on records accessed)	
GiAPA detected	1,765,955,117 unblocked writes of records found in 4,625 HotSpots	
Job and user	HSLAB KVKZKDV (117 jobs) HSLAX HAHXDYM (2 jobs) HSLIJ KVKZKDV (6 jobs) (More job info shown by GiAPA Menu option 19, sel. 2)	
Estimated saving	524 minutes run time (mainly CPU time)	
Effort required	Probably < 4 man-hours (test time not included)	



Technical explanation

Writing records/rows one by one is inefficient. A change to use blocking would save most of the time used by these writes.

Tips on how to optimize the performance

When QDBPUT occurs as the active program in many GiAPA HotSpots it should always be considered if the much more performance efficient blocked writes could be used. If the program logic does not necessitate forcing the records to be added to the file immediately, CL statements may be used to request blocking (please refer to GiAPA Tutorial 14, slides 4, 6, 7 and 9 for more details). Data base management will in some cases not automatically use blocked writes, e.g. if access path(s) with unique keys are defined for the data. However, if user program logic assures that duplicate key values are avoided, blocking can be forced through use of CL OVRDBF statement. Blocking could cut over 80 % of the time used for writing the records.

Print all pages Print page

An automatic performance analysis usually completes within seconds, after which the result is displayed immediately. If html format like the above example was selected and the result does not appear, please turn to the last page of this manual for assistance.

5250 format of the examples shown in HTML format on the previous page:

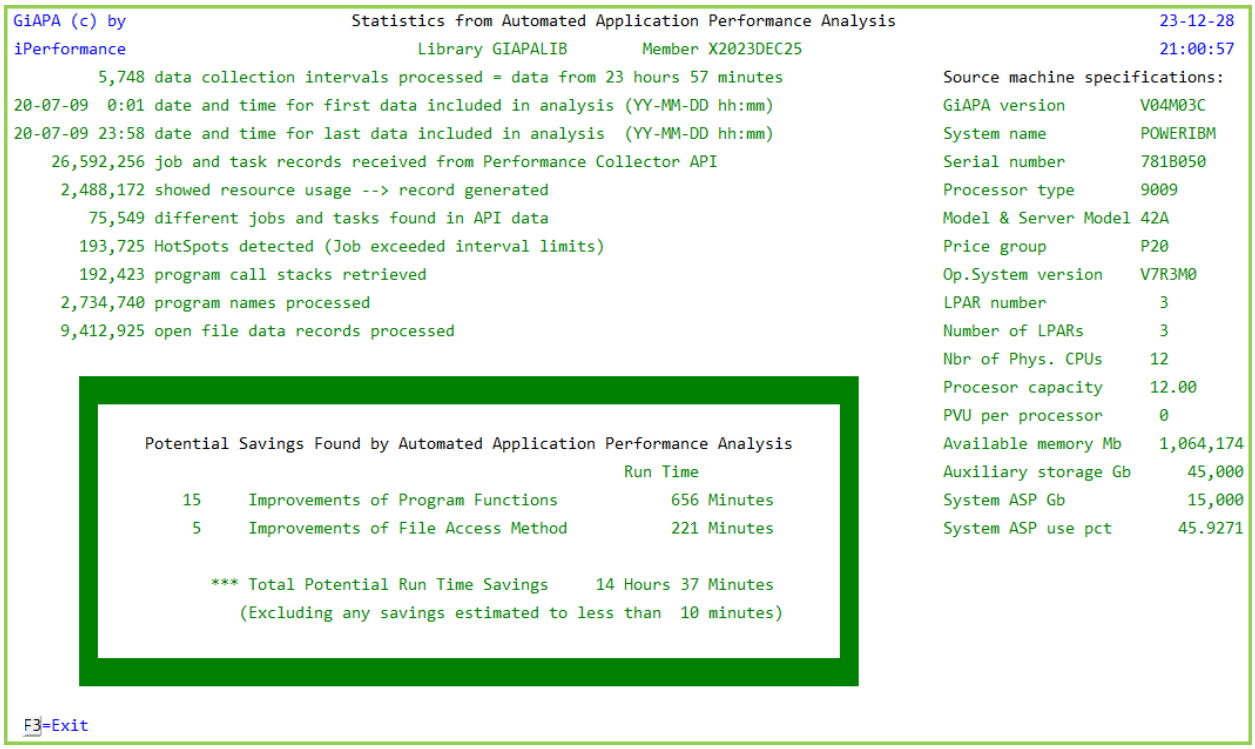
Command GIAPA200: Submit batch run of option 20, selection 1 for last date collected.

This command was suggested by a GiAPA customer who wanted to generate the automatic performance analysis html-report for the previous day in an unattended night batch job scheduled to run after command GIAPA140's expansion and analysis of collected GiAPA data.

GIAPA200 will select data for the latest analyzed date found in the library specified, but excluding any data collected on the day where the command is running.

Intended to show results for yesterday, it will only select data from one date. This would mean data from Friday if running on a Monday and no data was not collected Saturday – Sunday. This would also be true if the last expanded data covered a whole working week (Monday – Friday).

Example of selection **4** = the panel showing the total for the potential optimization savings:

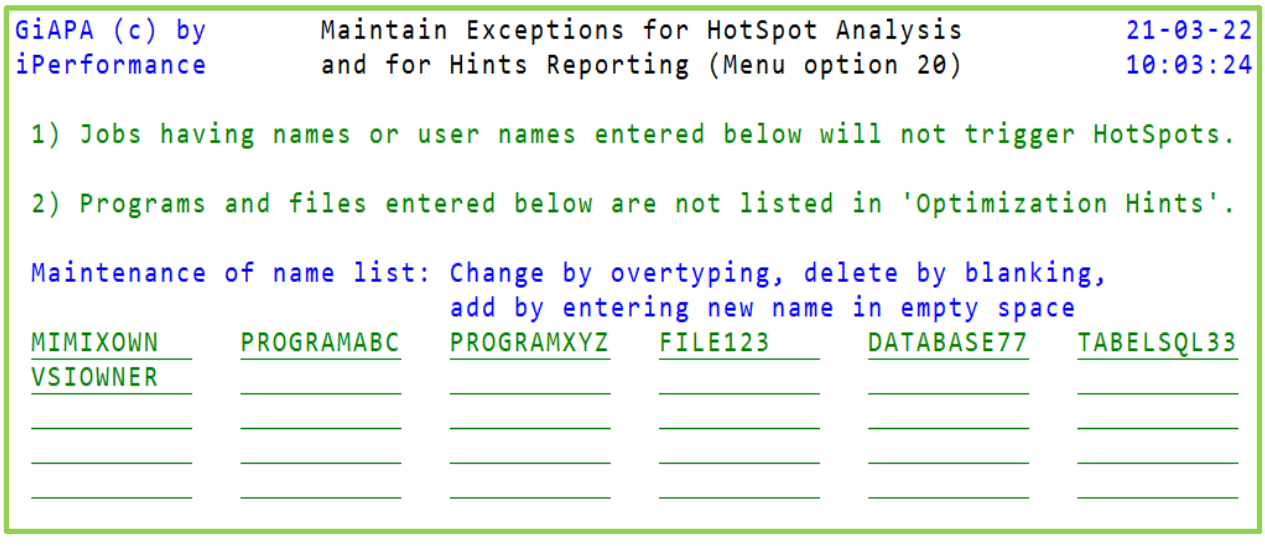


Excepting programs or files from this analysis

GiAPA without doubt also suggest some performance improvements which for good reasons cannot be implemented. One classic situation is purchased software where the supplier does not want to implement the change, but there are potentially many other cases.

There are numerous factors to be considered when specifying what GiAPA should include that some optimization suggestions probably will be incorrect or not relevant. In such a case, please report this to our technical expert at iPerformance. We are aware that the tables controlling what is reported must be adjusted over time to achieve a more consistently optimal result.

To avoid unwanted optimization hints reappearing please use **GiAPA Menu option 75** to access the following panel where you can enter names of programs and files which should not be considered for reporting.



GiAPA Menu Option 21: Collection Interval Summaries

An example of the selection screen for the Interval Summaries reports is shown below:

```
GiAPA (c) by       Select member for GiAPA interval summary          8/06/23
iPerformance                                                        14:21:08
                 Library: GIAPALIB          Select on date/time: YYMMDD hhmmss
D=Display, P=Print: D                                      From: 010101 000000
                                                           To:   991231 235959
Select one option for one or more members:  Select only if CPU % exceeds 000 %
 1=Resource usage per interval     2=Total resource usage     3=Graphics
 9=Calculate default CPU milliseconds (installation parameter for small jobs)
Opt  Member      Date       Text
 _   AUGUST07    070903     Consolidated performance data for August 2007
 _   SEP15MORN   070915     Performance data from the morning of September 15th
 _   WOCHE1808   080520     GiAPA Daten Woche 18 = 28 Apr - 04 Mai 2008
 _   VECKA3907   070927     Prestanda mätning vecka 39 - 2007
     (Additional lines have been removed for brevity)
```

Several members may be selected, resulting in e.g. a summary for an entire month.

You may also limit the number of intervals shown on the report within the member by specifying a start and/or an end date and time, and by selecting only intervals having used more than a certain CPU percentage.

Selecting option 1 combined with D for Display results in three different subfiles within the same program, allowing you to switch between the three using F6 F7 F8 – please see below. If P for print is selected, a report combining two of the displays is produced – please see report example below.

CPU Statistics per Interval

Analysis of CPU usage per interval demonstrates CPU-usage per type of workload:

```
GiAPA (c) by        Totals per Collection Interval: CPU Statistics (in pct.)   Input=APRIL05        11-06-01
iPerformance                                                                   14:34:49
Data collected at  System  Interval  NbrOf  Total ITV  Total  CFINT  Other  Inter  WEB   Multi  Other  Other
YY-MM-DD hh:mm:ss  ASP %   seconds   CPUs   CPU Sec.s  CPU%   Tasks  tasks  activ  jobs  thrd.  batch  jobs
11-04-05 19:36:15  90.0     015       2.00     19       63.3   5.9    3.4    4.7    11.7  29.7   3.4    3
11-04-05 19:36:30  90.1     015       2.00     17       56.6   3.0    3.4    13.6   4.7   30.3   2.0
11-04-05 19:36:45  90.1     015       2.00     15       50.0   3.1    3.7    1.2    3.0   36.1   2.2
11-04-05 19:37:00  90.1     015       2.00     15       50.0   3.0    3.7    2.6    3.8   36.2   1.5
11-04-05 19:37:15  90.1     015       2.00     11       36.6   5.2    2.7    1.0    9.5   .1     13.3   .1
11-04-05 19:37:30  90.1     015       2.00     2        6.6    2.1    1.7    1.6    2.5   .1     2.1    .1
11-04-05 19:37:45  90.1     015       2.00     12       40.0   1.5    .8     .5     4.6   .1     32.1   .2
11-04-05 19:38:00  90.1     015       2.00     25       83.3   1.1    .8     .2     10.4  .1     65.1   .1
11-04-05 19:38:15  90.1     015       2.00     18       60.0   .9     .5     3.8          .1     55.9   .1
11-04-05 19:38:30  90.1     015       2.00     25       83.3   4.6    .7     1.4    11.4  .1     62.7   .7

     (Additional lines have been removed for brevity)
F3=Exit    F6=CPU statistics    F7=Jobs using most CPU    F8=All resources
```

Three reports are available here. **F6**, **F7**, and **F8** allow you to switch between the report shown above and the next two, “Jobs using most CPU” and “All resources”.

On this report, **NbrOf CPUs** shows the assigned CPU capacity for the LPAR, and **Sys. ASP%** shows the disk usage percent for the system ASP.

The eight columns containing totals for job types are:

- **Total CPU%** is self-explanatory – the total CPU usage within the interval
- **CFINT** shows the use by the operating system task dispatcher(s).
- **Other tasks** are all operating system tasks except CFINT.
- **Interactive** is self-explanatory.
- **WEB jobs** show CPU used by QZDASOINIT and QZRCSRVS.
- **Multithreaded** shows the CPU used by job with more than one thread, typically JAVA.
- **Other Batch** shows all batch jobs except WEB and multithreaded jobs.
- **Other jobs** include all other jobs than the above mentioned.

Jobs using most CPU

```
GiAPA (c) by          Totals per Collection Interval: Jobs using most CPU      Input= APRIL05        11-06-01
iPerformance                                                                   14:51:20
Data collected at  Total   CPU% Job using   CPU% Jobname    CPU% Jobname    CPU% Jobname    CPU% Jobname    CPU% Jobname
YY-MM-DD hh:mm:ss  CPU%   Job1 most CPU    Job2 2nd most   Job3 3rd most   Job4 4th most   Job5 5th most   Job6 6th most
11-04-05 19:36:15  63.3    30.0 ERTCHMAYB   29.2 ERTCHMAYB  15.4 QZDASOINIT  8.0 QZDASOINIT  7.3 QPADEV003Y  2.5 GVSRXV
11-04-05 19:36:30  56.6    30.9 ERTCHMAYB   29.3 ERTCHMAYB  25.3 QPADEV003Y  9.5 QZDASOINIT  1.6 QPADEV003K  1.2 TKBKQMRPS
11-04-05 19:36:45  50.0    36.8 ERTCHMAYB   35.2 ERTCHMAYB   4.3 QZDASOINIT  1.9 QPADEV003K  1.7 QZDASOINIT  1.0 TTYKT05
11-04-05 19:37:00  50.0    36.9 ERTCHMAYB   35.2 ERTCHMAYB   7.2 QZDASOINIT  2.9 QPADEV000E  2.0 QPADEV003K

          (Additional lines have been removed for brevity)
F3=Exit    F6=CPU statistics    F7=Jobs using most CPU    F8=All resources
```

This report displays the CPU % and name for the 6 jobs that used the most CPU at the date/time shown.

Interval summary for all resources

```
GiAPA (c) by          Totals per Collection Interval: All Resources      Input= APRIL05        11-06-01
iPerformance                                                            14:54:02
YY-MM-DD  CPU%  Sync.DB   Sync.NonDB   Async.DB   AsyncNonDB   Logical DB Misc. I/O   OverFlows  Nbr.Trans Pages alloc. Communic.
HH:MM:SS  #CPU read/write read/write  read/write  read/write   read/write Perm.Write  Printlines  Tr.time  /deallocated  get/put
11-04-05  63.3     747     3.894      27.582        11             841            51     582.296
19:36:15  2.00      78     3.148       2.132       4.485           6.787          14       1       8.766
11-04-05  56.6     143     1.360       5.121                       14             22       2.697
19:36:30  2.00      20     3.718          29        228            324             2       2.326
11-04-05  50.0      14     1.058           2          1              4            20       2.931
19:36:45  2.00      17     3.409       2.306        317            392          1.959               2.939
11-04-05  50.0      18     1.040       1.145          1             26             25       4.418
19:37:00  2.00       5     3.749         420        284            322                      6.082

          (Additional lines have been removed for brevity)
F3=Exit    F6=CPU statistics    F7=Jobs using most CPU    F8=All resources
```

The column headings of this report are self-explanatory. Note that the average response time for the interval can be calculated based on the values in the column on the right by dividing the transaction time (in seconds) by the number of transactions.

Report: CPU Usage per Data Collection Interval

GiAPA (c) by iPerformance — CPU usage per data collection interval — Page 9 — 2011/11/10 11:35:42

Date of data collection: 2011-09-30 — Only intervals with CPU usage exceeding 100 % are selected

CPUs assigned to this LPAR: 1.50 — CPU seconds available per 15 seconds interval: 22.5

Itv.time hh:mm:ss	Itv sec	CPUsec used	Total CPU %	CPU% Job using Job1 most CPU	CPU% Jobname Job2 2nd most	CPU% Jobname Job3 3rd most	CPU% Jobname Job4 4th most	CPU% Jobname Job5 5th most	CPU% Jobname Job6 6th most
13:00:15	15	28	124.4	44.6 QPADEV0092	36.6 TMBYRPS	14.6 TVHFMVRKWK	5.0 THSYDV43	4.9 QZDASOINIT	4.5 THSYDV49
13:00:30	15	28	124.4	36.0 QPADEV0092	34.6 TMBYRPS	21.1 CVBRE00	5.2 QZDASOINIT	4.9 THSYDV41	4.5 THSYDV04
13:00:45	15	28	124.4	38.5 QPADEV0092	27.1 CVBRE00	13.1 TPTKLVHSKY	6.6 THSYDV41	5.7 QZDASOINIT	5.6 THSYDV49
13:01:00	15	24	106.6	36.7 QPADEV0092	13.2 TPTKLVHSKY	13.1 TPTKLVHSKY	8.1 QZDASOINIT	5.6 QPFRADJ	2.8 THSYDV03
13:01:15	15	26	115.5	42.6 QPADEV0092	13.4 TPTKLVHSKY	13.1 TPTKLVHSKY	13.1 TPTKLVHSKY	13.1 TPTKLVHSKY	6.3 THSYDV04
13:03:00	15	23	102.2	32.0 QPADEV0092	13.1 TPTKLVHSKY	8.4 QZDASOINIT	5.7 QPFRADJ	4.2 THSYDV40	3.0 THSYDV01
13:03:15	15	26	115.5	49.0 QPADEV0092	19.8 TPTKLVHSKY	13.1 TPTKLVHSKY	13.1 TPTKLVHSKY	7.9 QZDASOINIT	5.6 Y4.RPSKYK
13:16:00	15	23	102.2	48.6 TMBYRPS	26.8 QPADEV0092	9.0 QZDASOINIT	3.6 THSYDV04	2.9 EEBYBTVEL4	2.8 THSYDV06
13:22:00	15	23	102.2	38.0 CVBRE00	33.9 HKKUPMADVK	8.6 QZDASOINIT	4.6 THSYDV40	4.5 THSYDV49	3.4 RPSHCK
13:23:30	15	24	106.6	29.5 HKKUPMADVK	19.8 TPTKLVHSKY	10.6 TPTKLVHSKY	10.4 TPTKLVHSKY	9.5 QZDASOINIT	5.2 THSYDV43
13:29:30	15	23	102.2	19.8 TPTKLVHSKY	13.1 TPTKLVHSKY	13.1 TPTKLVHSKY	13.1 TPTKLVHSKY	7.5 QZDASOINIT	6.2 TPTBLY3

This report was produced with selection P=Print, output selection 1, and selection to print only details for intervals where more than 100 % CPU was used.

This was obviously running in an uncapped LPAR. The report shows the data collection intervals where the CPU usage exceeded the assigned capacity, and documents which jobs used the most CPU (= main cause of excessive CPU usage).

Resource Usage Totals based on Interval Summaries

Selecting option 2 provides an overview of all resources used for a given time period. The example shown should be self-explanatory.

If CPU usage exceeding 100 % was found in any intervals, four lines highlighted in red will appear to indicate cases where the CPU usage exceeded the assigned capacity.

The number of processors assigned to an LPAR may of course vary, for which reason the minimum and maximum numbers of CPUs found in the data are shown. The calculation of the excess usage is made for each interval, allowing the results to reflect any changes in the number CPUs assigned.

The detailed report (output selection 1) or charts (output selection 3) can subsequently be used to see the exact time for the different levels of CPU usage.


```
GiAPA (c) by            Resource Usage Summary based on totals per 15 seconds data collection interval          22-01-14
iPerformance                                 8752BD8      BIGRPS                                               17:40:38

 First interval: 2021-11-04 at 00:01:15     Last interval: 2021-11-04 at 23:59:00              Input=BANCIDATA
    Processing capacity   4.00 CPUs            Memory in megabytes     156,221           MB memory per CPU      39,055
 <======================================>  <======================================>  <======================================>
            Other Resources                             Reads                                  Writes
 <======================================>  <======================================>  <======================================>
           5,752 Intervals processed              32,081,311 Synchronous DB              33,790,430 Synchronous DB
        49:21:58 CPU time HH:MM:SS                38,054,593 Synchronous non-DB          68,098,480 Synchronous non-DB
          87,529 Transactions                    117,647,904 Asynchronous DB            136,186,148 Asynchronous DB
         4:40:49 Trans/JobqTime HH:MM:SS           2,830,537 Asynchronous non-DB         64,168,473 Asynchronous non-DB
    2,598,448,788 Pages allocated                1,908,548,714 Logical Database          42,154,561 Logical Database
    1,602,944,488 Pages deallocated                                                     242,112,251 Permanent writes  80.1 %
        5,634,034 Maximum pages used              77,718,787 Miscellaneous I/Os          91,332,565 Print lines
          527,039 Exceptions (Overflows)                  0 Communication gets                   0 Communication puts

   4.00  /   4.00 CPUs assigned to LPAR (Min / Max)                    172,024,814 Total syncronous I/Os
   51,4  /  125,0 Interval CPU percent  (Avg / Max)                    320,833,062 Total asyncronous I/Os
   59.0  /   70.4 System ASP usage      (Min / Max)                  2,028,422,062 Total logical I/Os
                                                                    2,521,279,938 Total physical + logical I/Os
            329 Intervals used more than 100 % of assigned CPU
    8,7  /   25,0 Excess CPU usage in % (Avg / Max), calculated in % of currently assigned CPU capacity for LPAR
          28:42 Total CPU used in excess of assigned LPAR capacity HH:MM:SS
         1:22:15 Elapsed time of intervals where CPU usage exceeded assigned LPAR capacity HH:MM:SS
F2=Cmd line      F3=Exit      Print
```

Graphic Report for one to five resource types

Select option 3 to reach the graphics selection panel – please refer to the detailed description of the almost identical panel described for Menu option 16, selection 4. The main difference is the input record selection possibilities, which here is date/time and/or minimum CPU % used.

```
GiAPA (c) by       Select resource type(s) for GiAPA Graphics          22-01-14
iPerformance        Library: GIAPATEST   Member: FEB26TH               17:45:02

           Date/time selection from: 220114 000000  To: 220114 235959

                              T T=Totals, A=Avg/15Sec, M=Max/15Sec, B=Both A + M
Select 1-5 resources:         2 Use 2nd Y-Axis for 0, 1 or 2 last selections?

<---Various------>  <-Physical Reads->  <-Physical writes->  <-Logical I/Os-->
4 Average CPU pct   _ DB synchronous    _ DB synchronous      _ Reads
5 Maximum CPU pct   _ DB async.         _ DB async.           _ Writes
_ CPU seconds       _ NDB synchron.     _ NDB synchron.       _ Miscell. I/Os
_ Average CPW value _ NDB async.        _ NDB async.          1 All log. I/Os
_ Print lines       _ All DB            _ All DB
_ Transactions      _ All non-DB        _ All Non-DB          <-Communication->
_ Response time     _ All synchron.     _ All synchron.       _ Comm. gets
_ Overflows                             _ Permanent           _ Comm. puts
_ Total nbr. of Intervals               _ Non-permanent       _ All communic.
_ CPUsec used exceeding       2 All phys.DB I/Os
  100% of LPAR capacity       _ All sync. I/Os
                              3 All physical I/Os

F2=Cmd Line   F3=Exit   F11=Max/Avg CPU + Phys/Log I/Os
```

The above selection (where member, date and time selections were made on the previous panel) shows the specifications appearing automatically when using F11. It will generate the popular “GoodMorning” chart giving an overview over resources used.

Command GIAPA052 can generate this chart in batch. An example of the chart can be found next to the description of command GIAPA052 close to the end of this manual.

The X-axis is calculated automatically by the program, depending on the time span of the data selected. The X-axis will be as detailed as possible without exceeding 60 different values. The scale could be 15 seconds intervals, minutes, 10 minutes, hours, days, or months. In this example data the selection of one day caused the X-axis to be hours. Had we selected data for an entire month, the X-axis scale would have been converted to days.

Generation of a graph usually completes within seconds, after which the result is displayed immediately. Should that not be the case, please see the last page of this manual.

Estimated default milliseconds for small jobs

Selection 9 is only displayed if Job Accounting is inactive. It may however always be used.

For installations not having Job Accounting active and running a very large number of very small jobs, this option may be very valuable. Please read the green text of the screen shot below:

```
GiAPA (c) by    Estimated default milliseconds for small jobs        16-05-18
iPerformance    Input member used for calculation: APR13_2016        17:07:45

Introduction: Without Job Accounting being active to supply exact CPU usage
for all jobs, GiAPA use history log messages to obtain data for small jobs
terminating so quickly that they are not reported by the Perf. Collector API.

However, QHST only reports CPU usage in seconds (minimum value one second),
and small jobs typically only use a few milliseconds. A very high number of
small jobs could therefore result in the CPU usage total for all jobs being
much higher than the total CPU time actually used on the server.

This option calculates a recommendation for the installation parameter that
defines the default average CPU milliseconds to be applied for such small jobs.

Number of small jobs for which default CPU milliseconds was used      761,992
                                  Data collection duration          23:59:00
Total CPU time used       33:00:15  Avg. small jobs per second            8.8
Total CPU for all jobs    38:20:35
Difference HH:MM:SS        5:20:20  Default milliseconds used              50
Difference percentage        16.17  Suggested new default value =          24

F2=Cmd line     F3=Exit
```

In the above example the shipped GiAPA default value of 50 milliseconds CPU usage for small jobs was used to replace the 1000 milliseconds (= one second) received from QHST history log.

The LPAR processed on one day 761.992 such small jobs. But even though the 50 milliseconds were a much more correct average CPU usage than one second would have been for the small jobs, the above statistics show a 16 % difference (= 5 hours and 20 minutes) between the actual total CPU usage of 33 hours and the total 38:20:35 obtained by calculating the sum of the CPU usage of all jobs.

GiAPA therefore suggests that the installation parameter (defined using GiAPA Menu option 78) specifying the default value to use for such small jobs is changed to 24 milliseconds. A rerun after that change resulted in the difference decreasing to 0.32 % (6 minutes).

The result of above calculation may obviously vary depending on the actual workload for different days, and may not be important for installations running only few “small jobs”.

This option is particularly interesting for installations wanting to use GiAPA to calculate CPU usage totals for certain jobs or users even though Job Accounting is not active.

GiAPA Menu Option 22: File Analysis Based on HotSpot Data

The reports generated in this section have some strength and weaknesses. They are based on HotSpot data, and do therefore not provide a full overview of how all files are used all the time. However, if the HotSpot data collection limits are set relatively low resulting in many HotSpots identified, it is reasonable to assume that the reports indeed will cover an important part of the accesses to the most used files. In this case the analyses offered here become quite valuable for spotting performance inefficiencies.

Using this option will display the following selection panel:

```
GiAPA (c) by    Select member for Global HotSpot File Analysis        7/09/23
iPerformance                                                        15:14:16

      Sort by: N N=Name      Output: D    Select files from library: __________
                 I=I/Os      P=Print,     Position list to filename: __________
                 R=Reuse     D=Display    Below mbrs. read from lib. GIAPALIB
Select one member only:
 1=Analysis per file name    2=Analysis per job name
Opt  Member      Date        Text
 _   AUGUST04    040903      Consolidated performance data for August 2004
 _   SEP15MORN   040915      Performance data from the morning of September 15th
 _   WOCHE3804   040820      GiAPA Daten Woche 38 = 13 - 17 Sep 2004
 _   VECKA3903   040927      Prestanda mätning vecka 39 - 2004
     (Additional lines have been removed for brevity)
```

Three sort criteria are available: Library and file name, Number of I/Os, or Reuse. Reuse is an estimated reuse (= repeated read, write or update by the job) of certain record numbers.

In installations having very many files this report may grow quite large. Therefore, it may be limited by specifying a file library name, implying that only files stored in that library will be included. If N is specified to obtain sorting by Name, a file name can be entered causing the report to start from that specific file name.

If selection 1 is selected for a member the output will resemble the following table:

```
GiAPA (c) by          Global HotSpot File Data Analysis Sorted by Estimated RRN Reuse  Input=V2TESTDATA        8/06/26
iPerformance                                                                          13:23:50
 File    File    Member   PF   Nbr.of   Nbr.of  Estimated  <---------- I/O Statistics ---------->
Libr.name Name    Name     LF   Jobs     Opens  Record Reuse      Total      Writes      Reads    Other I/Os
KFWGBPBD  FJBPA1L8 FJBPA1L8 LF      1       1    82,680,154   109,677,758          0  109,677,758          0
MPIEUB    MMBSUR  MMBSUR   PF      5       5    35,295,960    35,297,149          0   35,297,149          0
RBTCONLIB RBCMH11 RBCMH11  LF      1       1    11,297,214    15,062,952          0   15,062,952          0
ZIELGBPAD FDPAD1L2 FDPAD1L2 LF     1      28     2,598,688     3,458,042          0    3,458,042          0
QTEMP     ACEJCPL0 ACEJCPL0 LF     5       6     1,677,284     2,055,969     15,463    2,037,993      2,513
KFWGBPBD  WMPLD1LD WMPLD1LD LF     5      10     1,262,668   104,361,749     14,945  103,581,769    765,035
ZIELNLPAD FFJOBSKP FFJOBSKP LF     4       4     1,245,817     2,207,951          0    2,207,951          0
(Additional lines have been removed for brevity)
F2=Cmd Line  F3=Exit     If cursor in subfile: Show file details sorted by  F6=Job  F7=User  F8=Opens  F9=I/Os  F10=Reuse
F14=Create Graphics            (Note: F6-F10 displays subfile. F18-F22 displays printed report allowing use of "Find")
```

The data shown simply summarizes all uses of each file as retrieved from the Open Data Paths for the jobs when HotSpots were collected. Obviously the I/O statistics do not represent a summary of all HotSpot data for a file within each job – only the highest values per job and file are used. This may result in incorrectly high values for jobs running over several days, because the highest value for each day is used for the summary.

Estimated Record Reuse. To understand the logic behind these values, please familiarize yourself with the record reuse calculations in the “File Analysis Summary Report” described earlier in this manual. The explanation is provided in the example illustrating “Potentially Superfluous I/Os” of that specific report.

F14=Create graph data creates a chart. The records for the graph are always selected from the beginning of the subfile, and the user should position the cursor on the subfile record to specify the last selection. If the cursor is positioned outside the subfile, all records are selected – however, in either case only a maximum of 50 records are written into the graphics data file, since a very large number of records are not suited for creating charts.

The data for a file can subsequently be viewed on the following lower level, showing data for the jobs that used the file. To do so position the cursor on a file name and use **F6, F7, F8, F9 or F10** to obtain a subfile, or correspondingly **F18 – F22** to obtain the same type of output as a printed report (also being displayed onto the work station).

The difference between the F-key options is the sort criterion of the report generated. Placing the cursor on the second file MMBSUR in the above report and pressing F10 will provide the following result:

```
GiAPA (c) by          HotSpot File Data Analysis sorted by Estimated RRN Reuse     Input=WEEK3704        7/09/23
iPerformance                                                                       16:23:45
 Libr.: MPIEUB    File: MMBSUR    Mbr: MMBSUR    Text: Payroll Transaction Database              Input=WEEK3704
 <--------------- Job ----------------> Number of    Estimated  <---------- I/O Statistics ---------->
 Name      User      Number  YY-MM-DD  Opens     Record Reuse      Total      Writes      Reads    Other I/Os
 R6247ABG  PAYROLL   731584  07-07-29      1      11,782,720    11,782,720          0   11,782,720          0
 R6247ABG  PAYROLL   733570  07-07-29      1      11,770,942    11,770,942          0   11,770,942          0
 R6247ABG  PAYROLL   733492  07-07-29      1      11,739,728    11,739,728          0   11,739,728          0
 TB160RF02E PAYROLL  715999  07-07-29      1           2,402         2,402          0        2,402          0
 QPADEV0025 BTSPGM   732419  07-07-29      1             168         1,357          0        1,357          0
  F2=Cmd line    F3=Exit                                    ENTER=Go to top of subfile
```

The report clearly indicates that jobs need to be examined. The standard job performance analysis report selection (GiAPA Menu Option 15) allows specification of the job name to quickly pinpoint this potential performance flaw. Within a job, the File Statistics report or the File Analysis report will show the necessary details.

A high number of record reuse is most often a quite inefficient use of CPU time. However, it is by no means unusual – the typical case is reading job level parameter information inside the loop where the individual transaction records are read instead of before (= outside) the loop.

Note that since the information is based on HotSpots, which is comparable to random sampling, further investigation is always necessary. The results have overall proved to be reliable, but could also be incorrect in some cases.

Example: A job triggers 10 HotSpots during 600.000 reads to a file. The relative record numbers recorded for that file by the HotSpots all are within the range 35.000 to 40.000. The program therefore reports 595.000 I/Os as being potentially superfluous in the “Estimated Record Reuse” column. This could be true or very misleading. In any case it might be worthwhile checking.

If selection 2 (Analysis per Job Name) is used, the following panel will appear:

```
GiAPA (c) by   Global HotSpot File Data Analysis per Job Name Sorted by Estimated RRN Reuse   15.09.08
iPerformance                    Input=KAARETST                          21:17:01
Job       Nbr.of    Nbr.of   Estimated   <---------- I/O Statistics ---------->
Name       Jobs     Opens   Record Reuse       Total       Writes       Reads    Other I/Os
QDFTJOBD     41    1,165  2,055,579,949  2,598,119,358    8,558,208  2,477,379,222   112,181,928
ACTUAL_PL    10      511  1,905,069,330  2,068,910,002   16,042,307  2,013,344,737    39,522,958
SENDEDDIE     3       83  1,125,756,473  2,278,975,103        7,238  2,278,450,329       517,536
WAVEPICK  1,869   57,086    133,145,705    260,219,114    1,136,281    256,631,629     2,451,204
SENDEDDIE2    3       34    103,127,358    116,725,375       28,994    116,182,659       513,722
PLCHOOSE     20      775    100,519,848    321,029,994   16,145,295    233,067,765    71,816,934
BACKPACK      2    1,150     93,222,025    117,014,064    1,134,537    113,904,474     1,975,053
    (Additional lines have been removed for brevity)
  F2=Cmd line   F3=Exit   F9=Call stacks for selected jobname
```

Please observe that use of F9 with the cursor on one of the jobs displayed will show call stack list and statistics for all jobs having that job name. For jobs showing a high number in the “Estimated Record Reuse” column, the F9 option offers a convenient short-cut to see which programs and statement numbers requests the most I/Os.

GiAPA Menu Option 23: Jobs Having Priorities Modified

Some installations have problems with users changing the priority of their own jobs. This report will list all jobs that had priority changed. However, please note that if you run with dynamic priorities (defined in system value QDYNPTYSCD) the operating system will automatically downgrade the priority of jobs using excessive CPU causing disturbances in other jobs.

```
GiAPA (c) by          Jobs Having Had Run Priority Changed          8/07/24
iPerformance                                                       01:15:13
          Member: V2TESTDATA                  Library: GIAPALIB
 Jobname   User     Type Prio  Count          Jobname   User     Type Prio  Count
 TRM_PCKTOE MIMIXOWN   B  25-55     3
 UXT3_FEJ   DGXBEN     B  50-54     1
Etc., etc
```

GiAPA Menu Option 24: CPU usage per current user

QZDASOINIT, QTFT*, and QZRCSRVS are examples of names for jobs that often use excessive resources, but where it is challenging to decipher which user(s) actually were responsible. This is caused by the jobs running under a general user name (e.g. QUSER) and getting different “current user” names attached through the lifetime of the job.

```
GiAPA (c) by    Select Member for Current User CPU Statistics          19-03-18
iPerformance                                                          16:17:32
                          Select interval: YYMMDD hhmmss      YYMMDD hhmmss
Output: D D=Display P=Print            From: 000101 000000  To: 991231 235959

 1=Select when nonblank Current User is not equal to Job User       Library:
 2=Select all records having a Current User or Job User             GIAPALIB
Opt  Member      Date       Text
 _   MAR19_2019  190318     GiAPA data from March 19th, 2019   (ExpDate 19113)
 _   MAR18_2019  190318     GiAPA data from March 18th, 2019   (ExpDate 19112)
 _   MAR17_2019  190318     GiAPA data from March 17th, 2019   (ExpDate 19111)
 _   MAR16_2019  190318     GiAPA data from March 16th, 2019   (ExpDate 19110)
 _   MAR15_2019  190318     GiAPA data from March 15th, 2019   (ExpDate 19109)
 _

F2=Cmd Line     F3=Exit
(Only data expanded with "Keep detail *YES" is available for this report)
```

Also in this situation GiAPA can assist, since GiAPA retrieves the messages reporting when a current user is attached. Current user is reported as “*VARIOUS” for intervals where several users were attached during a 15 seconds collection interval,

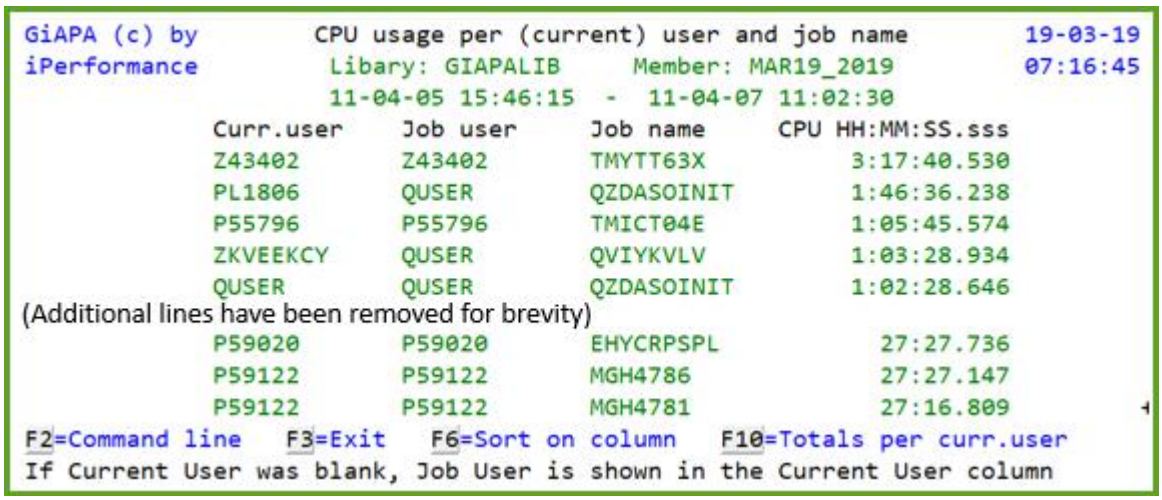
Option 24 will provide a panel allowing the user to select an expanded data member, and optionally within that data also to select from/to values for date and time. Please note that the data must have been expanded with option Keep detailrecords? *YES to use this feature.

Selection 1 only includes records where a “current user” was attached to the job.

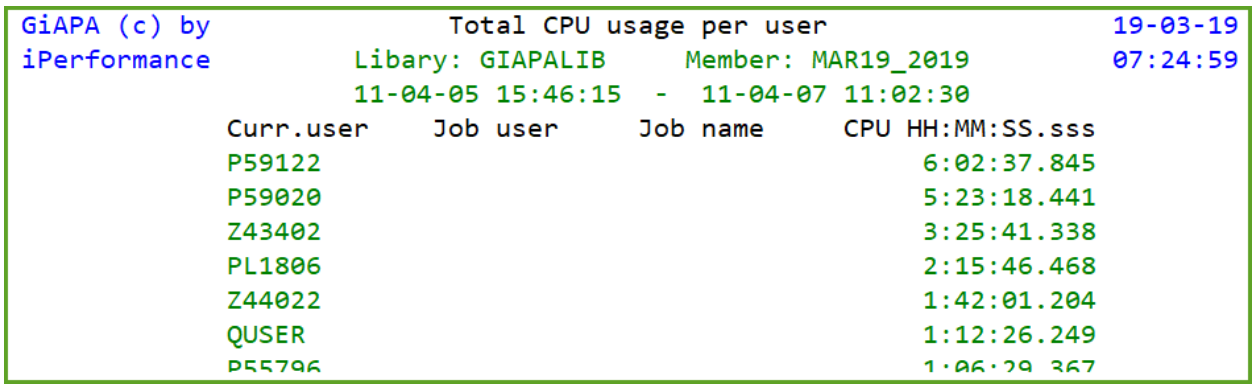
```
GiAPA (c) by          CPU usage per current user and job name          19-03-18
iPerformance          Libary: GIAPALIB     Member: MAR19_2019          16:39:08
                      19-03-19 00:01:15  -  19-03-19 23:58:30
              Curr.user    Job user     Job name     CPU HH:MM:SS.sss
              PL1806       QUSER        QZDASOINIT        1:46:36.238
              ZKVEEKCY     QUSER        QVIYKVLV          1:03:28.934
              AFEPSSDEY    QUSER        QZDASOINIT          43:40.008
              XDSDNZK      QUSER        QZDASOINIT          42:04.784
              P55573       QUSER        QZDASOINIT          17:54.518
              ZKVKYK04     QUSER        QZDASOINIT          17:31.897
(Additional lines have been removed for brevity)
              FBSYPYBM     QUSER        QZDASOINIT             22.671
              A3918        QUSER        QZDASOINIT             21.996
              ZKVEBMY      QUSER        QOVEKVLK               19.496
F2=Command line   F3=Exit   F6=Sort on column   F10=Totals per curr.user
Only records having Current User assigned were selected
```

Use **F6** with the cursor positioned on a specific column to re-sort on the selected column.

Selection 2 includes all records having a user name, i.e. operating system tasks are not selected. For jobs not having a “Current User” assigned, the normal “Job User” name is moved to the “Current User” column, enabling a total overview of the CPU usage per user name.



For both above reports, **F10** can be used to request a total per (current) user name:



GiAPA Menu Option 26: User Defined Graphics

Menu Options and Commands Used to Produce Graphics

Menu Option 26 is used to select records, define the key and data fields, and run the generation of the graphics data file used as input for a given chart.

Command GIAPA050 can be used to schedule batch generations of already existing user defined graphics, and optionally send the charts attached to Email(s). It is intended to be used following scheduled runs of command GIAPA140 (batch expansion and analysis of performance data) and optionally followed by command GIAPA141 (consolidate graphics data received from command GIAPA140 on other LPARs).

Introduction

Graphical representation is normally used to provide an overall picture, or for showing resources used by e.g. a range of jobs. To ease generation of charts, GiAPA performance data expansion and analysis creates file GIAPA144P3, containing summary records with job name and type, user name, run date and hour as control break fields. Data fields in file GIAPA144P3 include CPU usage, transactions, various types of I/Os, etc. The full file description can be seen in source file GIAPALIB/GIAPA_QDDS.

Job and/or user name may be blanked out and two otherwise blank user fields may be initialized by the user. Please refer to the comments within the source code of user exit program GIAPA_UE1 in source file GIAPALIB/GIAPAEXAMP for further details on this option. If Job name and/or user name is not required for the graphics, use of the exit program to suppress these fields reduces the number of records in file GIAPA144P3. This will in turn significantly speed up the generation of user graphics.

Specifications

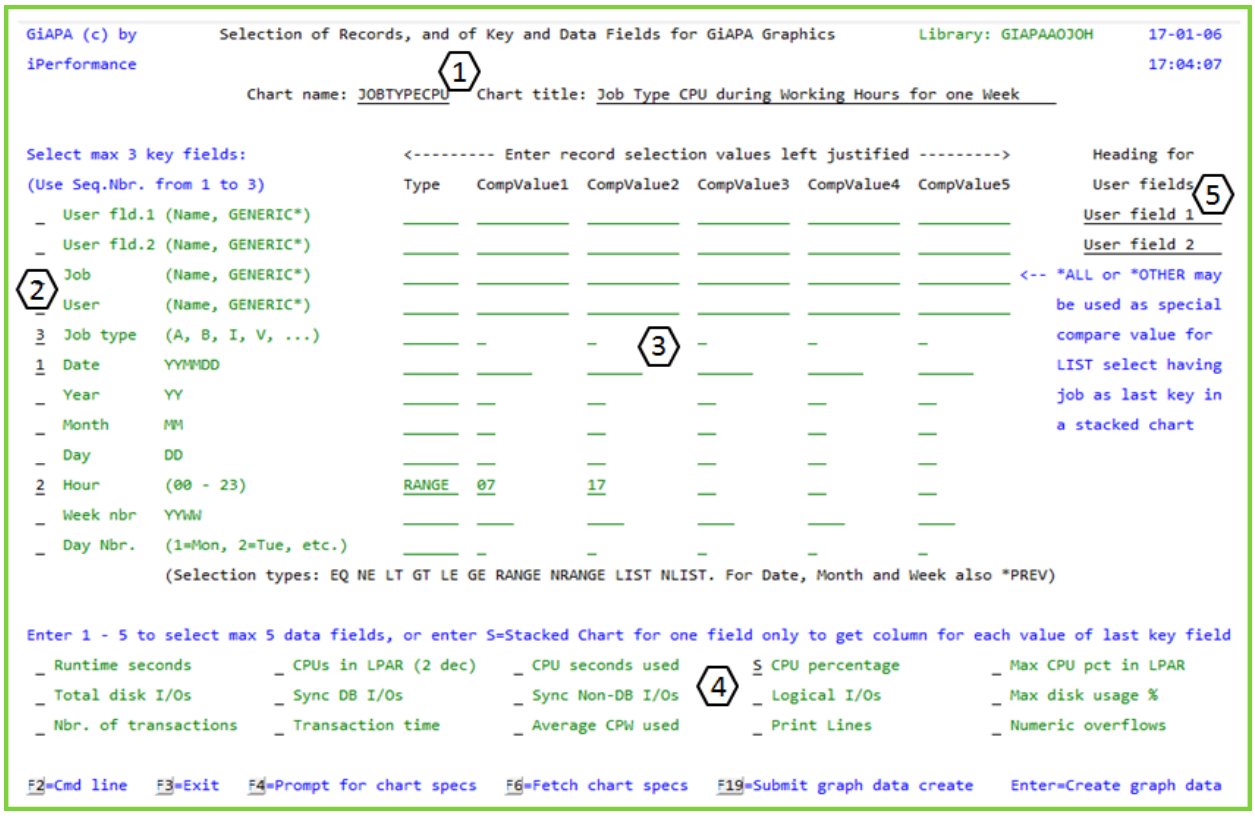
Selecting GiAPA Menu Option 26 will display the panel shown below. This allows you to select the data member(s) to be used for as input for the graphics. The lines in pink only appear if input data was transferred from other LPARS using command GIAPA140, causing the special member MULTI_LPAR to appear.

```
GiAPA (c) by                      Select Input Members for Graphics                    10/05/21
iPerformance                                                                           18:22:53
                        Delete all MULTI_LPAR records older than YYMMDD: _____
 1=Select one or more members   5=Display analysis and expansion statistics       Data library:
                  *) If selected, it must be the only member selected
Opt  Member            Created   Nbr of Recs   Member Text
 _   Week50_09         091220            215   Analysis of week 50 of 2009
 _   Dec2009           090116           1342   Expansion of all GiAPA data for December 2009
 _   JAN2010           100202           1467   Performance data from 01-01-2010 to 31-01-2010
 _   MULTI_LPAR   *)   100429             15   Summary graphics data from more LPARs

F2=Cmd line     F3=Exit
```

Please note that several members may be selected, if member name MULTI_LPAR is not included. In addition, it is the responsibility of the user not to select the same data twice for a report, exemplified by: if data for Jan. 4th, 2016 was expanded into member “MONDAY4JAN”, and this day also is included in the expansion for “WEEK01_16”, then selection of both members for one graphical report would double the Monday data. Likewise it is also the responsibility of the user that the selected member(s) really contain all the expanded data for the requested time period.

A clean-up of older records within member MULTI_LPAR can be achieved by entering a date in line 3 of this panel. To remove other members, use GIAPA Menu Option 82.

When one or more members have been selected for input, the following panel for selecting records to include, and for defining key and data fields to use for the graphics is displayed. Since quite a few specifications might be required, such definitions of graphical input may be saved under a template or chart name for later use, together with an explanatory text.



If selected data input member is MULTI_LPAR, the fields Job and User on the above panel are replaced by field names Serial Number and System Name.

The panel above is used for five different types of specifications required to define the data to be included in the input to the graphics:

1. Chart name and title
2. Between one and three key fields for the chart
3. Record selection specifications, if any
4. Between one and five data fields to be included
5. Headings for User Fields – if they were defined and requested to be used as key fields

Chart name must be specified. It is used as the name for the graphics data file created, and as a name for the template definition. The name of an existing chart may be specified, after which **F6** can be used to fetch the template data. **F4** can be used to obtain the list of existing templates:

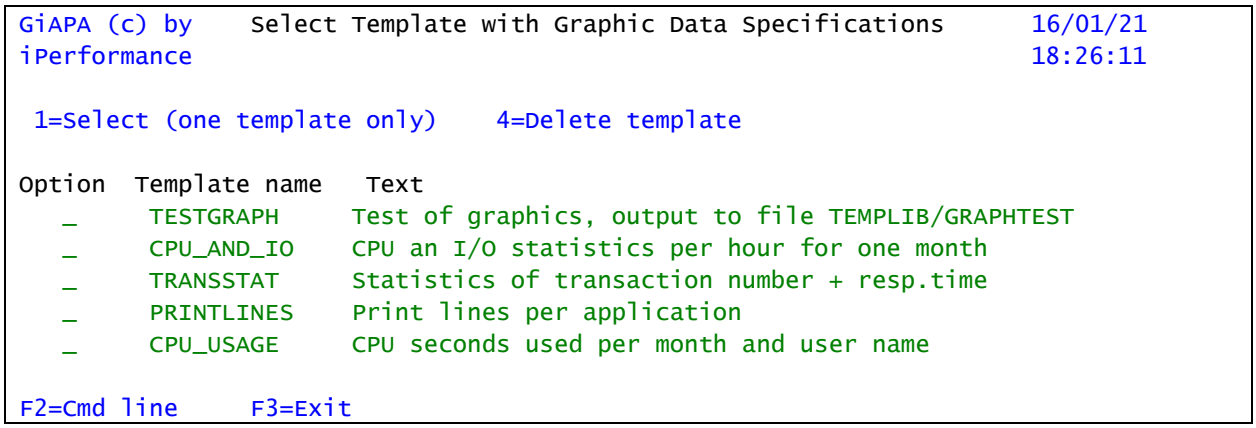


Chart title is a documentary text also used as the title for the resulting chart.

Since you may schedule daily runs of commands GIAPA140 and GIAPA050, you may want the **chart name** and/or **title** to contain the run date or week, or maybe yesterday (being the data collection date). The chart name and/or title may therefore contain variables, which at run time are substituted as follows, provided that the run was made on June 18th, 2015:

- &Y will be replaced by 15. (If you want 2015, code 20&Y.)
- &MM will be replaced by Jun.
- &M will be replaced by 06.
- &D will be replaced by 18.
- &NN will be replaced by THU (abbreviation of name of weekday).
- &W will be replaced by 24 (the week number for the week containing 2015-06-18).
- &P (P for previous date) will be replaced by 17.
- &L (L for last week) will be replaced by 23. **Note:** Cannot be used together with &P.

Please note: The dates are based on **Job Date.** Furthermore, if &P or &L is used to set an earlier date or week, the other date variables will be adjusted to agree with that older date. To exemplify this scenario: if today is January 1st, 2016 (a date actually belonging to week 53 of 2015), then a title coded as

- "Data from 20&Y-&MM-&P" will provide the result "Data from 2015-Dec-31"
- "Week &Y/&L" will provide the result "Week 15/52"

Select key fields (leftmost column of input capable fields) is used for defining 1, 2, or 3 fields that should be used as key fields when the graphics are generated. These fields also serve as control break fields for summarizing the numeric data when the graphics data file is created.

It is the responsibility of the user to ensure that the contents (i.e., number of different values) in the key fields selected do not cause an excessive number of summary records to be generated. A large number of records are not well suited as input for generation of charts.

User fld.1 and User fld2 (the first two "Select key" fields) are, as the name implies, fields that may have received a value from the optional user supplied program GIAPA_UE1 during the creation of the graphics' input file GIAPA144P3. Please refer to section "Exit Program GIAPA_UE1" in this manual under the description of Menu Option 14.

Record selection specification (in the middle of the panel) allows different types of record selections. Note that if selection specifications are made for several lines, they will be AND connected, i.e. they must all be valid to select the records. In the above example records for the generic user name VSI* are selected.

Special value ***ALL** or ***OTHER** can be used as a type **LIST** selection value for the **Job** name if Job name is defined as the last of two (or three) key fields, and S (= Stacked) is specified for the data field to be selected. Please refer to the example below.

The special selection operation code ***PREV** (previous) may only be specified for date, month, or week number. It will cause the program to select data for yesterday, last month, or the previous week, respectively, based on **Job Date**. No entry may be made in the value fields to the right of the operation code *PREV for month or week. For date, the selection value can be left blank (implying yesterday only), or a given numeric value between 1 and 31 defining how many days (the last day will always be yesterday) should be selected. Example: if today is June 18th, the specification `Date YYMMDD *PREV 10`
will select data from June 08th through 17th, both days included.

Select data fields section (the input capable fields located at the bottom) is used to choose max 5 numeric fields containing different types of resource usage.

Instead of selecting from 1 to 5 data fields, an S may be entered for one field only, causing data to be prepared for a S=Stacked Chart. Specifying S for a data field will result in the contents of the last key field selected, i.e., the key field selected with the highest number (2 or 3), to be used as “column headings” for the data field selected. This feature is very useful, but challenging to explain: please see the Stacked Chart example below.

Heading for user fields (on the right side of the panel) defines headings for the two user fields.

When the desired specifications have been entered, the graphics data file can be generated interactively by hitting Enter, or by submitting a batch job by using **F19**.

The formation of the input file for the graphics will normally only take a few seconds and depend on the number of records in the input file. The time required can most often be reduced considerably by using the user exit program GIAPA_UE1.

When the graph is generated interactively, the html result is displayed immediately. Should that not be the case, please see the last page of this manual.

Important: Generating a new graphics data file overwrites any old file with an identical name, unless it was renamed using menu option 28, but GiAPA will keep and reuse any changes made to graph type, “Show Values” and selected color palette.

Copying templates (= user defined graph specifications) to another LPAR:

File GIAPA262P1 in the GiAPA data library (defaults to GIAPALIB) contains the templates. If no templates are defined on the receiving LPAR, copy file GIAPA262P1 to the receiving LPAR.

To copy only selected template(s) please use the following procedure:

```
CPYF giapadatalib/GIAPA262P1 TOFILE(QTEMP/MYWORKFILE) CRTFILE(*YES) +
      INCREL((*IF TEMPLNAME *EQ mychart) (*OR TEMPLNAME *EQ mydiagram))
```

Transfer QTEMP/MYWORKFILE to the receiving LPAR, and add the records to GIAPA262P1:

```
CPYF QTEMP/MYWORKFILE TOFILE(giapadatalib/GIAPA262P1) +
      MBROPT(*ADD)  CRTFILE(*NO)
```

Example: Use of *ALL or *OTHER in LIST selection for Job

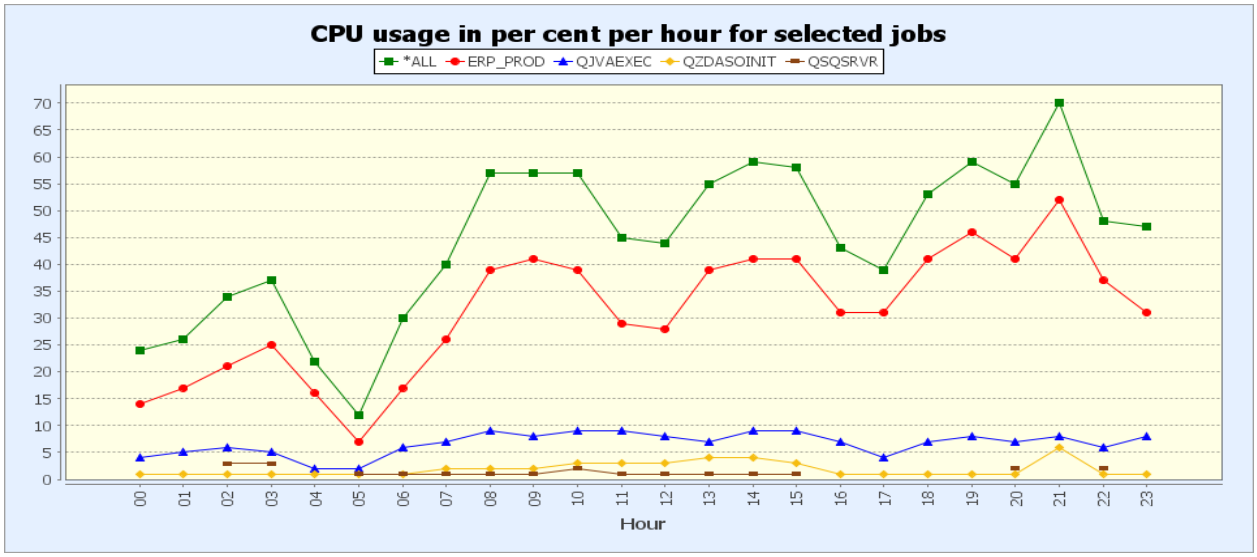
It is possible to create a diagram that depicts the use of one resource for a few major jobs compared to the total usage of that specific resource. The following examples originate from an installation where an ERP application is the main user of resources. The task was to generate a simple overview of the CPU usage of the ERP solution (running in several jobs named ERP_PROD) and the connected Data Base Server and Java jobs compared to the total load on the server.

The definition of the chart, which can be scheduled to run as an unattended batch job, is shown below together with two examples of the results. Four named jobs are selected together with either *ALL or *OTHER.

When *ALL is selected the chart type should be a “Line” diagram as shown in the first graph.

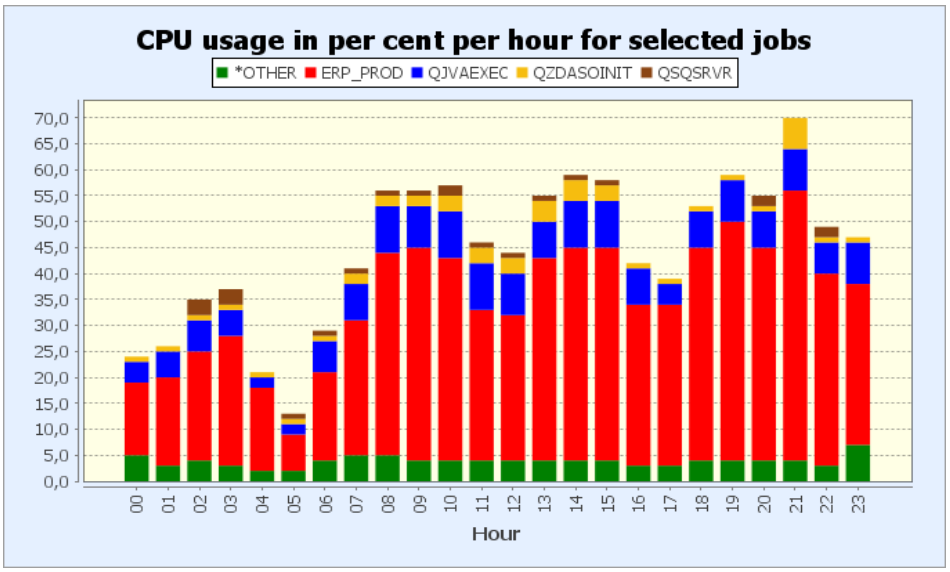
If *OTHER is selected, a stacked bar or column chart is the chart type that best demonstrates the results as shown in the histogram below.

```
GiAPA (c) by          Selection of Records, and of Key and Data Fields for GiAPA Graphics      Library: GIAPALIB      18-09-21
iPerformance                                                                                                    15:31:32
                        Chart name: CPUPERHOUR   Chart title: CPU usage in per cent per hour for selected jobs

Select max 3 key fields:                <--------- Enter record selection values left justified --------->        Heading for
(Use Seq.Nbr. from 1 to 3)              Type    CompValue1  CompValue2  CompValue3  CompValue4  CompValue5          User fields
 _  User fld.1 (Name, GENERIC*)         _____   __________  __________  __________  __________  __________          User field 1
 _  User fld.2 (Name, GENERIC*)         _____   __________  __________  __________  __________  __________          User field 2
 2  Job        (Name, GENERIC*)         LIST    ERP_PROD    QJVAEXEC    QSQSRVR     QZDASOINIT  *ALL       <-- *ALL or *OTHER may
 _  User       (Name, GENERIC*)         _____   __________  __________  __________  __________  __________          be used as special
 _  Job type   (A, B, I, V, ...)        _____   _           _           _           _           _                   compare value for
 _  Date       YYMMDD                   _____   ______      ______      ______      ______      ______              LIST select having
 _  Year       YY                       _____   __          __          __          __          __                  job as last key in
 _  Month      MM                       _____   __          __          __          __          __                  a stacked chart
 _  Day        DD                       _____   __          __          __          __          __
 1  Hour       (00 - 23)                _____   __          __          __          __          __
 _  Week nbr   YYWW                     _____   ____        ____        ____        ____        ____
 _  Day Nbr.   (1=Mon, 2=Tue, etc.)     _____   _           _           _           _           _
               (Selection types: EQ NE LT GT LE GE RANGE NRANGE LIST NLIST. For Date, Month and Week also *PREV)

Enter 1 - 5 to select max 5 data fields, or enter S=Stacked Chart for one field only to get column for each value of last key field
 _  Runtime seconds          _  CPUs in LPAR (2 dec)     _  CPU seconds used        S  CPU percentage        _  Max CPU pct in LPAR
 _  Total disk I/Os          _  Sync DB I/Os             _  Sync Non-DB I/Os        _  Logical I/Os          _  Max disk usage %
 _  Nbr. of transactions     _  Transaction time         _  Average CPW used        _  Print Lines           _  Numeric overflows

F2=Cmd line   F3=Exit   F4=Prompt for chart specs   F6=Fetch chart specs   F19=Submit graph data create   Enter=Create graph data
```



Above: the *ALL line in green illustrates the total cumulative load on the server in this case shown as average CPU percent.

Right: the CPU used by all the *OTHER jobs is shown in green. The top of each column in the stacked histogram illustrates the total CPU percent, corresponding to the green line in the graph above.



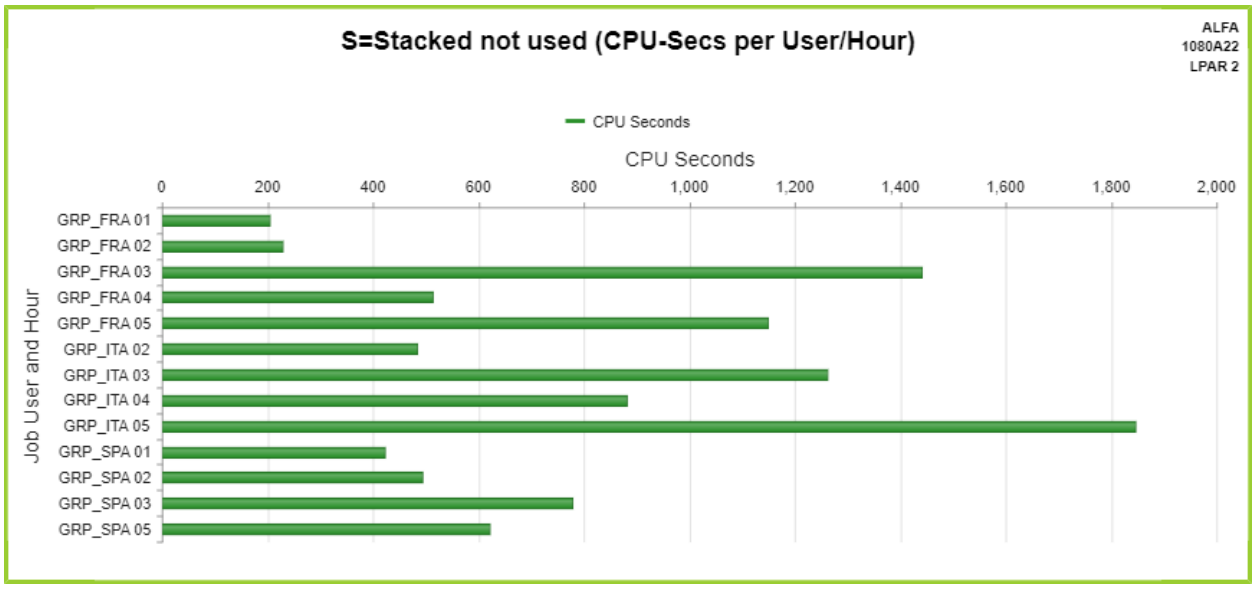
Example: Stacked Charts versus Other Chart Types

Suppose you want to analyze resource usage based on three countries during the night from 01:00 to 06:59. The users are based in France, Italy and Spain and can be selected on the generic user name GRP* (user names are GRP_FRA, GRP_ITA, and GRP_SPA). This is real data from a GiAPA site serving such a geographical area, and the three countries happened to run the same applications for roughly the same number of customers.

From GiAPA Menu Option 26 we first specify a chart not using the S=Stacked option. We select based on generic user name and hour, and we want to see the CPU seconds used. The definition panel would be the following:

```
GiAPA (c) by        Selection of Records, and of Key and Data Fields for GiAPA Graphics      Library: GIAPAAOJOH     16-01-25
iPerformance                                                                                                    14:57:00
                      Chart name: S-NOT_USED   Chart title: S=Stacked not used (CPU per user/hour)

Select max 3 key fields:              <--------- Enter record selection values left justified --------->        Heading for
(Use Seq.Nbr. from 1 to 3)            Type    CompValue1  CompValue2  CompValue3  CompValue4  CompValue5        User fields
 _  User fld.1 (Name, GENERIC*)       ______  __________  __________  __________  __________  __________        User field 1
 _  User fld.2 (Name, GENERIC*)       ______  __________  __________  __________  __________  __________        User field 2
 _  Job        (Name, GENERIC*)       ______  __________  __________  __________  __________  __________  <-- *ALL or *OTHER may
 1  User       (Name, GENERIC*)       EQ      GRP*        __________  __________  __________  __________      be used as special
 _  Job type   (A, B, I, V, ...)      ______  _           _           _           _           _               compare value for
 _  Date       YYMMDD                ______  ______      ______      ______      ______      ______          LIST select having
 _  Year       YY                    ______  __          __          __          __          __              job as last key in
 _  Month      MM                    ______  __          __          __          __          __              a stacked chart
 _  Day        DD                    ______  __          __          __          __          __
 2  Hour       (00 - 23)             RANGE   01          06          __          __          __
 _  Week nbr   YYWW                  ______  ____        ____        ____        ____        ____
 _  Day Nbr.   (1=Mon, 2=Tue, etc.)  ______  _           _           _           _           _
               (Selection types: EQ NE LT GT LE GE RANGE NRANGE LIST NLIST. For Date, Month and Week also *PREV)

Enter 1 - 5 to select max 5 data fields, or enter S=Stacked Chart for one field only to get column for each value of last key field
 _ Runtime seconds        _ CPUs in LPAR (2 dec)   1 CPU seconds used      CPU percentage          _ Max CPU pct in LPAR
 _ Total disk I/Os        _ Sync DB I/Os           _ Sync Non-DB I/Os      _ Logical I/Os          _ Max disk usage %
 _ Nbr. of transactions   _ Transaction time       _ Communic. puts+gets   _ Print Lines           _ Numeric overflows

F2=Cmd line   F3=Exit   F4=Prompt for chart specs   F6=Fetch chart specs   F19=Submit graph data create   Enter=Create graph data
```

Using the default options when creating the graph we would obtain the following result, which does not decipher well the comparison of the three countries:



Using GiAPA Menu option 28, selection 3 (Edit graph data) we can see the graph input data generated.

We have two columns containing our key fields user name and hour, and one data column containing CPU seconds.

GiAPA (c) by iPerformance — Edit Graphics Data File EXAMPLES/S_NOT_USED

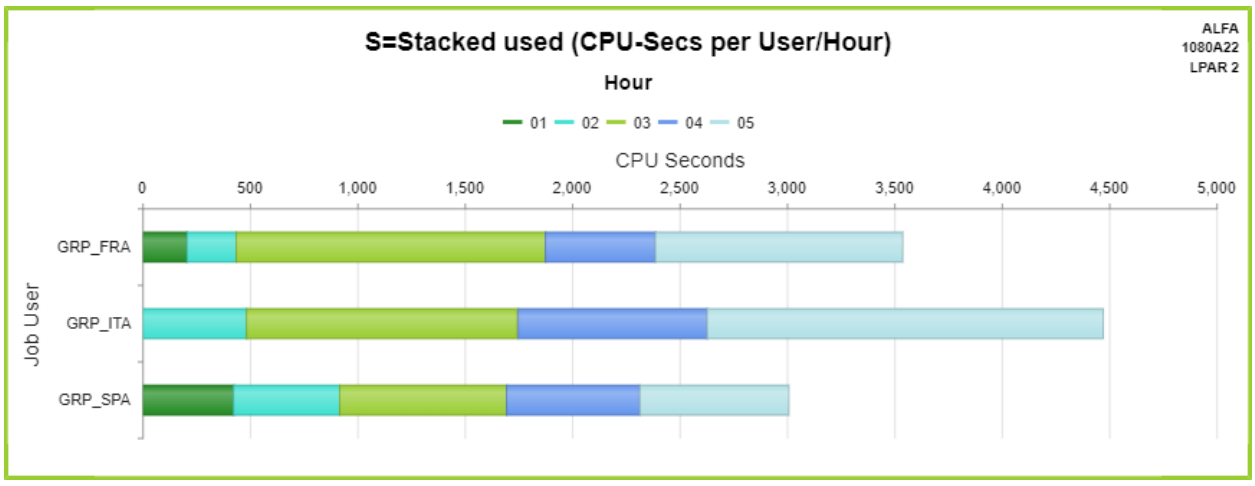
To delete a record blank out the key field(s)

RecNbr	1st Key	2nd Key	3rd Key	1st Data Field	2nd Data Field	3rd Dat
1	Job User	Hour		CPU Seconds		
2	GRP_FRA	01		000000000000203	000000000000000	0000000
3	GRP_FRA	02		000000000000228	000000000000000	0000000
4	GRP_FRA	03		000000000001440	000000000000000	0000000
5	GRP_FRA	04		000000000000514	000000000000000	0000000
6	GRP_FRA	05		000000000001148	000000000000000	0000000
7	GRP_ITA	02		000000000000484	000000000000000	0000000
8	GRP_ITA	03		000000000001262	000000000000000	0000000
9	GRP_ITA	04		000000000000881	000000000000000	0000000
10	GRP_ITA	05		000000000001846	000000000000000	0000000
11	GRP_SPA	01		000000000000421	000000000000000	0000000
12	GRP_SPA	02		000000000000493	000000000000000	0000000
13	GRP_SPA	03		000000000000778	000000000000000	0000000
14	GRP_SPA	05		000000000000622	000000000000000	0000000

If we change the “1” used to select the CPU seconds to an “S”, then the result changes to:

RecNbr	1st Key	2nd Key	3rd Key	1st Data Field	2nd Data Field	3rd Data Field	4th Data Field	5th Data Field
1	Job User			01	02	03	04	05
2	GRP_FRA			000000000000203	000000000000228	000000000001440	000000000000514	000000000001148
3	GRP_ITA			000000000000000	000000000000484	000000000001262	000000000000881	000000000001846
4	GRP_SPA			000000000000421	000000000000493	000000000000778	000000000000622	000000000000692

As you can see, the S for stacked caused the last defined key (Key 2 = Hour) not to be used as the key identifying a row (= record). Instead, the value of the second key field Hour is used as a “Column Heading” for the data. This allows generation of the following stacked chart, which in addition to a separate color for each hour also shows the total used CPU per Key1 (Job User).

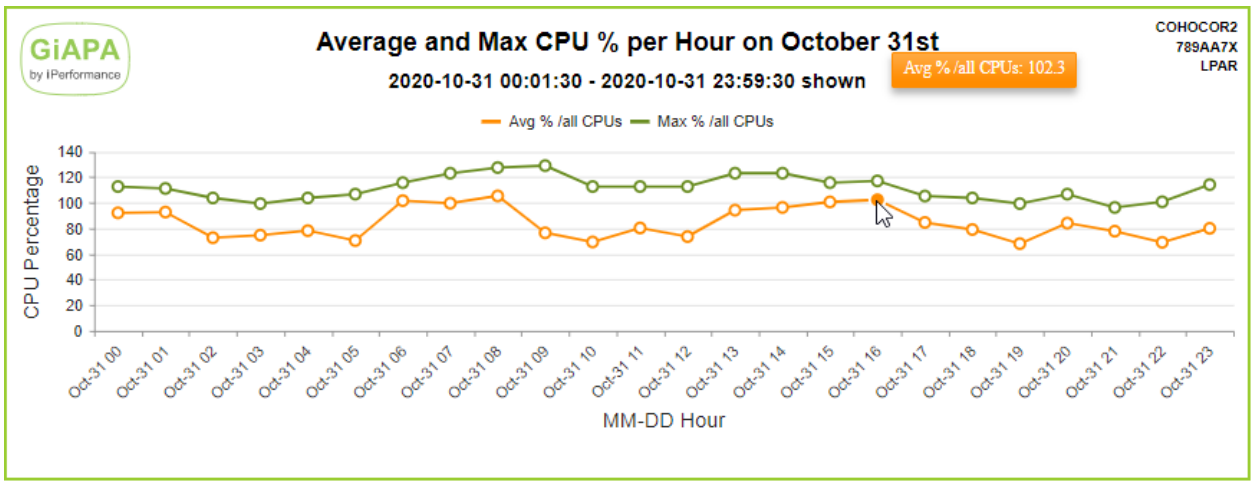


Note that placing the cursor on a value will display the data for that cell. The above graph e.g. shows that at Italy at 04 hours used 1,3K CPU seconds which translates to approximately 22 minutes.

Additional examples of GiAPA Graphics

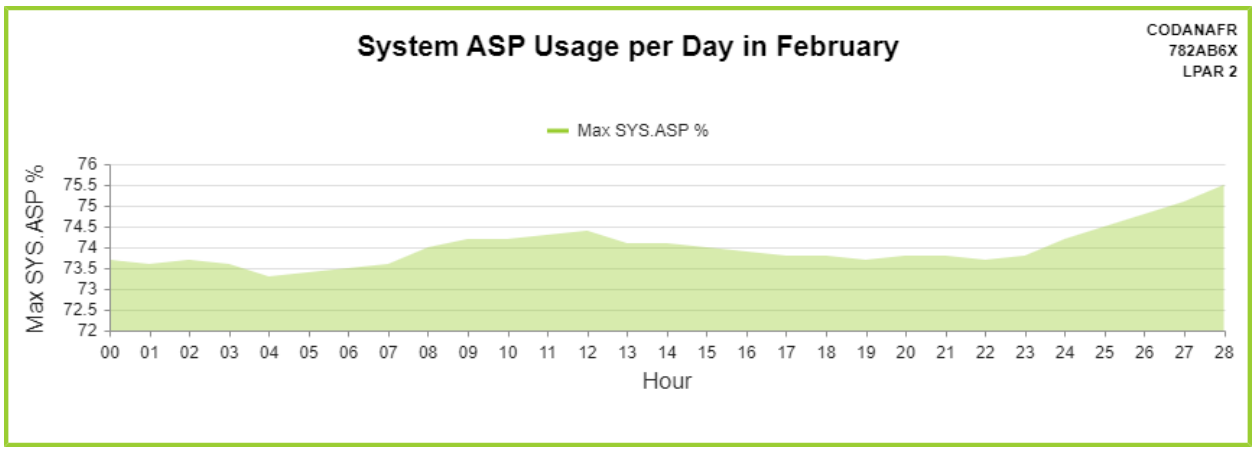
Below please find two additional “appetizers” to demonstrate additional possibilities within GiAPA graphics. Please note that both are defined from GiAPA Menu Option 26.

In the first example the question was to determine whether the LPAR ought to have more CPU resources assigned permanently. First, we selected a given date, and specified hour as key field. Second, we selected the data fields “CPU percentage” to report the average %, and “Max CPU pct in LPAR” to show the highest CPU-% found within the key defined. Please not that the “Mouse-over” function was used to show the average CPU % for October 16th.



The LPAR running uncapped “borrowed” additional resources from other LPARS. The result clearly shows that the LPAR probably should be assigned more CPU resources permanently.

For the second example we used data for all of February as input, selecting date as key and “Max disk usage %” as the only data field. Selecting an Area chart gave this result::



Disk usage was getting a serious problem around the 22nd, but a major clean-up running on the 23rd brought the situation back to normal.

An additional example showing graphics generated based on data from several LPARs is included earlier in this manual. It is located right after the sections explaining how commands GIAPA140 and GIAPA141 can be used to schedule automatic performance data expansion including transfer of data to another LPAR.

File GIAPA144P3: Input to User Defined Graphics

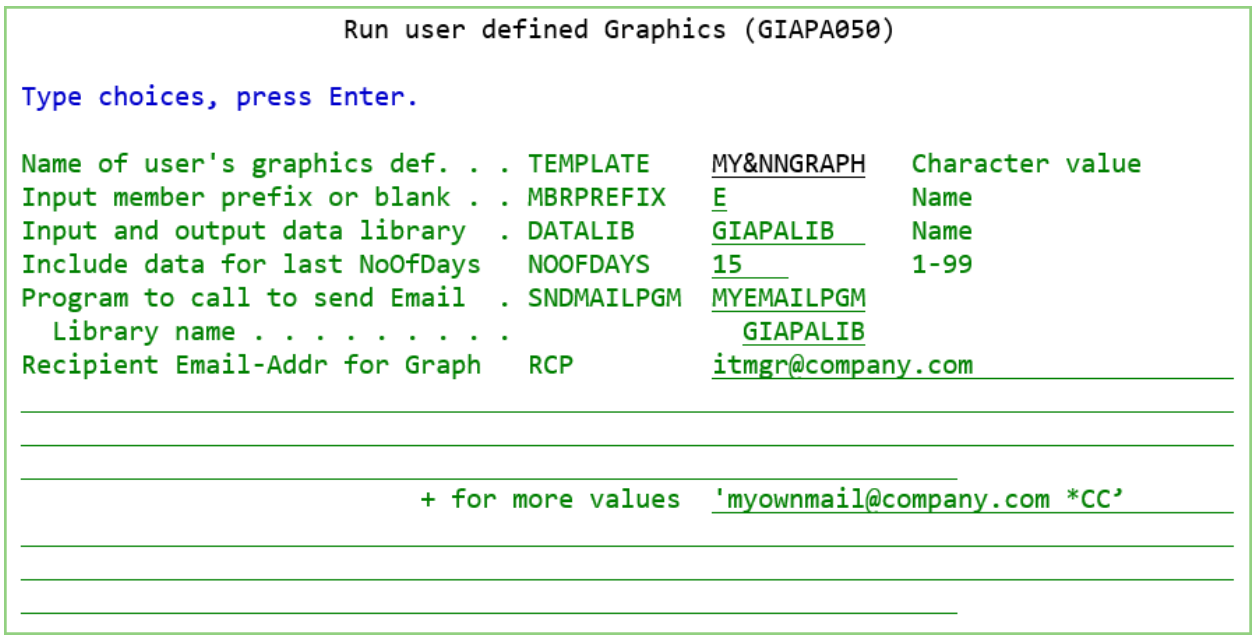
The input file used for User Defined Graphics is GIAPA144P3 containing one record per Job Name, Job User and Hour. This file may also be well suited as input for user defined queries.

For jobs like QZDASOINIT, where different “Current User” names are attached to the job, GIAPA144P3 will replace the Job User by the Current User. When such a job has served several current users during one hour, several records are generated for the job. The Job User name (typically QUSER) will be used to show the resources used in data collection intervals where no current user was attached, or where several current users were attached.

The file definition of GIAPA144P3 is available in GIAPALIB/ GIAPA_QDDS. The file contains two user fields that through a user exit program (called during data expansion and analysis) may be initialized with e.g. department codes or job accounting codes – please refer to the description of user exit program GIAPA_UE1.

Command GIAPA050 – Run User Defined Graphics

The following command allows generating user defined graphics in unattended, scheduled batch runs. The intention is to use it in connection with (i.e. after) command GIAPA140 (and optionally GIAPA141).



TEMPLATE is the name of the user’s graphics definition. Note that the definition may include variables for day, month, year and week number in the title field, so the dates on the result reflects the collection date of the data shown on the chart.

The name MY&NNGRAPH entered above would obviously select a template having that name. However, the resulting file to use for graphics input would be named MYFRIGRAPH, if today is Friday June 18th, because the variable "&NN" will in the resulting file name be replaced by the abbreviated name of the weekday. This allows the same template to be used every day even when the resulting graph should be available more than one day. New graphs having the same name as existing ones will overwrite the old data, but by using variables ad mentioned above, more graphs can be stored. For details on the variables allowed please refer to "**Chart name**" and "**Chart title**" in section "GiAPA Menu Option 26: User Defined Graphics" a few pages earlier in this manual.

MBRPREFIX may be used for a member name prefix letter for the data expansion made by command GIAPA140. If a letter is specified, the resulting member name will be XYYYYMMMDD, where X is a prefix, followed by the date for the expansion.

If MBRPREFIX is left blank, the special default member name **MULTI_LPAR** will be used automatically, and the input data used for the run will be the output file generated by command GIAPA141. This file is intended for consolidated data received from several LPARs, but could also be used within one LPAR to accumulate data from several days.

DATALIB is the name of the input and output data library.

NOOFDAYS specifies the number of days for which expanded data should be included as input for the graph e.g., if the member prefix selected is "K" and today is Thursday June 18th of 2015, a value of 2 in NOOFDAYS will include the expanded data members having the names K2015JUN17 and K2015JUN18. This would most likely contain data collected on Tuesday 16th and Wednesday 17th, if the expansions for each day only selects data from the previous day.

For runs on the MULTI_LPAR member containing data from several LPARs the NOOFDAYS parameter should not be used. The previous date, month or week may be selected by specifying the special value *PREV as selection operation code for the date, month or week number field when defining the graph template. Please see the more detailed description in the section on Menu Option 26.

SNDMAILPGM defines the name and library for a user exit program to be called if the generated chart must be sent as an Email to one or more recipients, in which case

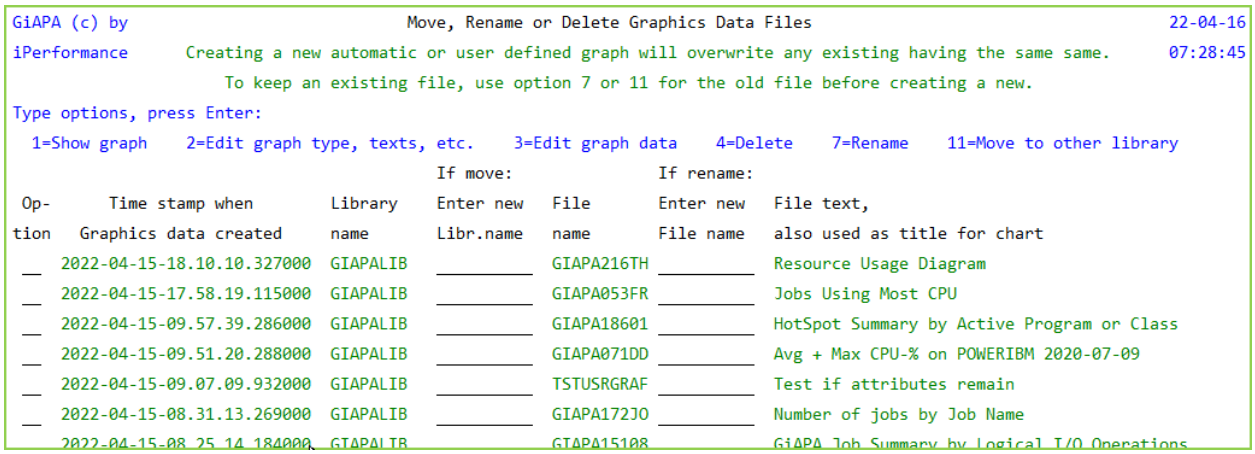
RECIPIENTS must contain the Email address(es) to which the Email(s) should be sent. A maximum of 20 addresses may be specified. Optionally each Email address may be followed by a blank and one of the entries ***PRI**, ***CC**, or ***BCC** specifying recipient type primary, carbon copy or undisclosed carbon copy. *PRI is default option.

User Exit Program Sending Charts in Emails

Please turn to **GIAPALIB/GIAPAEXAMP(EMAIL_TEST)** to see an example source code for a User Exit Email program. Comments in the source code give detailed explanation towards the functions of the program and some hints for the set-up of operating system parameters needed to send Emails from the server.

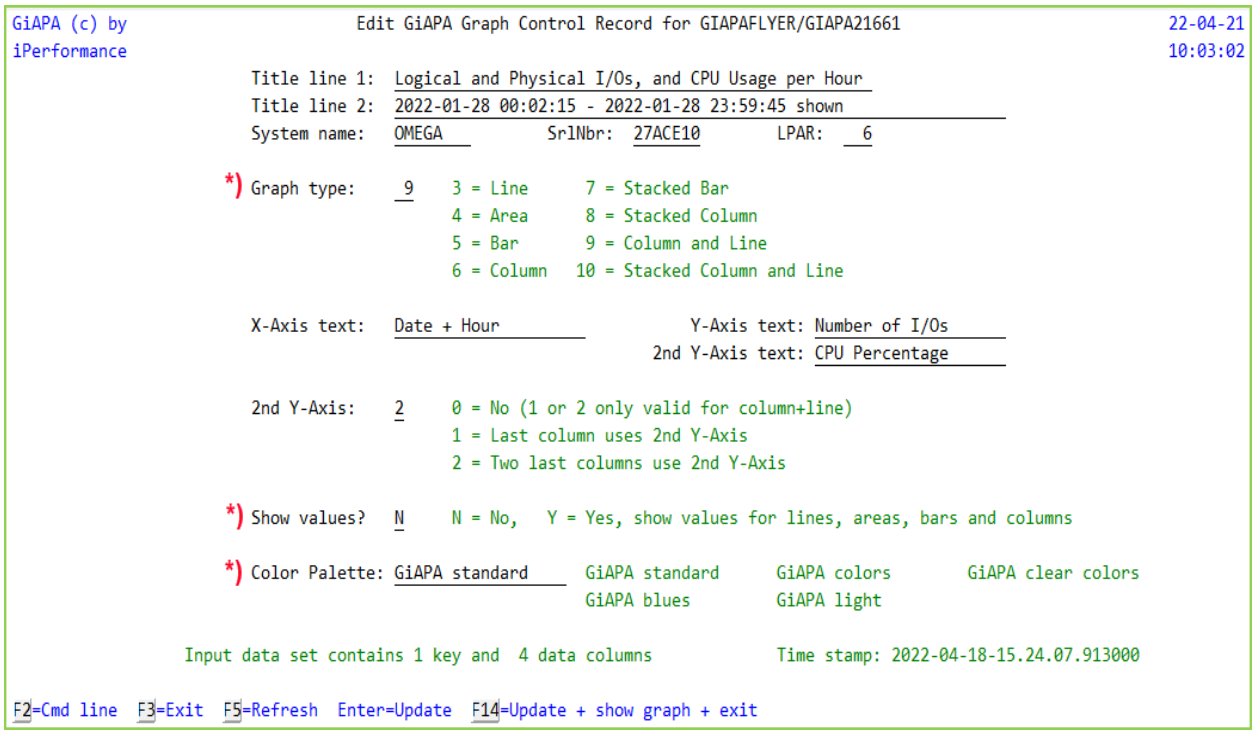
GiAPA Menu Option 28: Work with Graphics Attributes and Data

Although the automatic overwrite results in clean-up of files in most cases is convenient, you might want to keep some generated graphics data for future use. The graphics file(s) can by using **option 11** be moved to another library, or renamed using **option 7**. Names prefixed with GIAPA0 – GIAPA9 are reserved for internal GiAPA use and will therefore not be accepted.



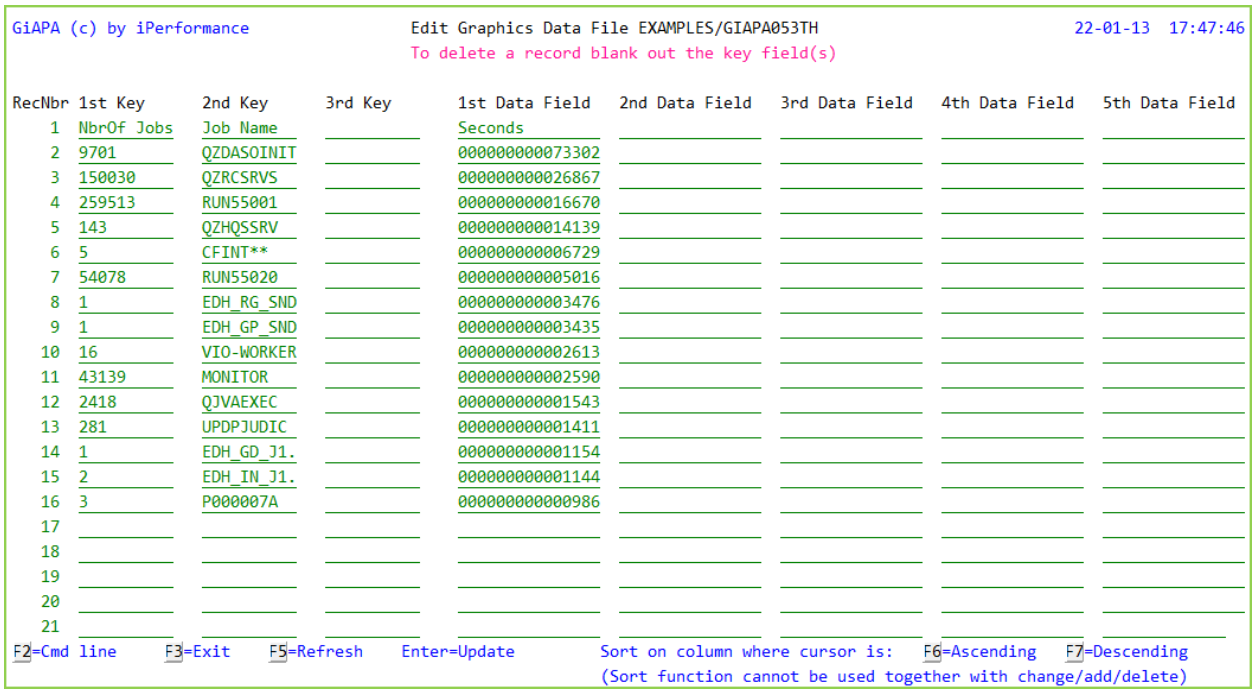
Option 1 will display the chart immediately. Should that not be the case, please see the last page of this manual.

Option 2 will result in a display like the following, opening for modifying chart type, texts, etc. The contents varies depending on the number of data columns. The “Show values?” entry may be used to request the found values to be showed at the top or end of columns, lines and bars.



***) Please observe** that generating a new graphics data file overwrites any old file having the same name, but if “Graph type”, “Show values” and/or “Color Palette” was changed for the old graph, these changes will be kept and be applied for the new graph.

Option 3 will allow you to edit the input data used for the chart:



GiAPA (c) by iPerformance — Edit Graphics Data File EXAMPLES/GIAPA053TH — 22-01-13 17:47:46

To delete a record blank out the key field(s)

RecNbr	1st Key	2nd Key	3rd Key	1st Data Field	2nd Data Field	3rd Data Field	4th Data Field	5th Data Field
1	NbrOf Jobs	Job Name		Seconds				
2	9701	QZDASOINIT		000000000073302				
3	150030	QZRCSRVS		000000000026867				
4	259513	RUN55001		000000000016670				
5	143	QZHQSSRV		000000000014139				
6	5	CFINT**		000000000006729				
7	54078	RUN55020		000000000005016				
8	1	EDH_RG_SND		000000000003476				
9	1	EDH_GP_SND		000000000003435				
10	16	VIO-WORKER		000000000002613				
11	43139	MONITOR		000000000002590				
12	2418	QJVAEXEC		000000000001543				
13	281	UPDPJUDIC		000000000001411				
14	1	EDH_GD_J1.		000000000001154				
15	2	EDH_IN_J1.		000000000001144				
16	3	P000007A		000000000000986				
17								
18								
19								
20								
21								

F2=Cmd line F3=Exit F5=Refresh Enter=Update Sort on column where cursor is: F6=Ascending F7=Descending

(Sort function cannot be used together with change/add/delete)

The example here shows data for the 16 job names that used the most CPU. By default it is sorted descending on the number of CPU seconds.

Several options are available. The 6th line showing CPU used by the task dispatcher CFINT is maybe not wanted – it can be removed by blanking out the Key fields for that line. New records may also be added.

Or maybe the chart should appear sorted by job name. If so, simply position the cursor in the job name column and use F6 or F7. Sorting into a user-defined sequence is also possible – please refer to Tutorial 10, where this is shown.

GiAPA Menu Option 31 – 33: Front End to Performance Explorer (PEX)

Introduction

IBM's Performance Explorer is an excellent tool for certain detailed analyses of performance problems, and can of course be used without GiAPA, provided that the user is familiar with the PEX commands and that Performance Tools are installed.

The GiAPA options to run PEX simply make it easier for the inexperienced user to run PEX since the necessary definitions are created and removed automatically, and the reports are generated automatically. Should Performance Tools not be installed on the machine in question (in which case the reports cannot be produced), then GiAPA automatically saves the raw collected PEX data in a save file, which can be transferred to any installation holding a Performance Tools license for analysis (e.g. to iPerformance). The PEX data collection can always run since it does not require Performance Tools.

If the LPAR running PEX does not have a GiAPA license, save file GIAPALIB/GIAPAPEXR is created. To produce the reports on an LPAR with GiAPA, simply run these commands:

1. `RSTOBJ GIAPA311P1 QTEMP *SAVF *FILE  SAVF(mylibrary/GIAPAPEXR)`
2. `CALL GIAPA319`

GiAPA provides shortcuts for running the “Statistical” type of PEX analysis, which is the most frequently used. PEX statistical run informs the user what program(s) within a job is using which resources. It is normally first used to produce a summarized report, providing an overview of the situation for a given job.

This first report does not always show sufficient details to identify within which program a problem exists. A new hierarchical data collection will then have to be run to obtain this detailed information. However, the more detailed collection often results in an enormous report that will result in much time and effort to locate the required information.

If PEX statistical analysis is run under GiAPA, GiAPA automatically adds further functionality thereby assisting in pinpointing the performance problem(s) more rapidly:

- Only one data collection is needed to create both the large detailed hierarchical report and the summarized overview report.
- Additional reports are generated to assist with the analysis of the detailed data (i.e. locating the application program causing the problem and showing the call stack for that program).
- A text field is added reporting the functions of the IBM internal programs and MI complex functions listed.

It was not the intention that this section should provide any comprehensive introduction to when PEX should be used and how the PEX reports should be interpreted. PEX is a comprehensive tool, and the IBM manuals must be consulted for a full understanding of the purpose of PEX, and for an explanation to what can be concluded from the contents of the various PEX reports.

GiAPA Menu Option 31: Start PEX Statistical Data Collection

```
                    Submit PEX STATS analysis (GIAPA310)

Type choices, press Enter.

Jobname to analyze with PEX  . . JOBID      __________    Name, generic*, *, *ALL
  User name  . . . . . . . . . .             __________    Name, generic*, *ALL
  Job number . . . . . . . . . .             ______        000001-999999, *ALL
Job queue name . . . . . . . . . JOBQ       QBATCH____    Name
  Library name . . . . . . . . .              *LIBL_____    Name, *LIBL
```

JOBID Specify the job name, user, and number for the job to be analyzed. Generic name or *ALL may be used, but the use of *ALL should be limited as much as possible to avoid the PEX data collection getting too resource consuming, and generating too many data. Detailed collection of data (Run type “hierarchical, refer to PEX manuals) are not allowed for *ALL jobs. Instead GiAPA will automatically select the best type of analysis.

JOBQ is the name of the job queue used for submitting the job starting the PEX data collection. Check that the job will start from the job queue used – it is usually not recommended to use a queue having many jobs waiting for execution.

To analyze a batch job you can hold the job immediately after it starts, use Option 31 to start PEX, and release the job again.

The PEX data collection will (when started through GiAPA) automatically terminate when the job being analyzed ends. Interactive jobs can simply sign off to achieve this aim. Alternatively GiAPA Option 32 can be used to end a GiAPA started PEX statistics run.

GiAPA will automatically produce the PEX reports when the data collection ends, if Performance Tools are installed on the machine. Otherwise GiAPA will create a save file with the collected data. This file could be sent to iPerformance for analysis.

The following four reports are produced:

1. Standard Performance Explorer report (=the original IBM PEX report) for a PEX run type *STATS *HIER. This report is very detailed and could exceed 100 pages. It consists of several parts, starting with run information detailing which job(s) were analyzed, the hardware specifications, and when the run was performed.

2. A summary by program name of the “Statistics Information” part of the above report, sorted descending by cumulative CPU usage.

3. A report sorted by program name of all user programs and all programs having used more than 1 % CPU. The record sequence number within the original PEX report for each use of the program is also provided.

4. A report like the previous, but sorted by the original PEX report record sequence number.

Explaining the use of these reports is most easily done by following an analysis of the four example reports below:

PEX Report: Statistics Information, Hierarchical

```
                                Performance Explorer Report                    5-05-17 12.52.4
                                 Statistics Information                        Page    6
Library . . : GIAPALIB
Member. . . : G311458279
Job name. . : ALL JOBS/TASKS IN SESSION
                      +----------------- Inline Stats ------------++---------- Cumulative Stats -----------+
          Times   Calls MI CPLX          CPU      DB   DB  NDB  NDB          CPU      DB   DB  NDB  NDB  Call
Name      Called    Made  Issued       (us) /  %   SIO  AIO  SIO  AIO       (us) /  %   SIO  AIO  SIO  AIO Level

NNT920         1  73992      0      525.452  0.6    0    0    2    0   20.677.822 24.9  2585   96  491    0   3
NNT921     21382 175294    330    1.438.765  1.7    0    0   22    0   18.791.504 22.6  2147   78  455    0   4
*RETDSEN       1      0      0            7  0.0    0    0    0    0            7  0.0    0    0    0    0   5
QDBGETKY   66949    270   2052    1.764.103  2.1 1034    2  131    0    1.800.887  2.2  1077    2  146    0   5
*SETCR      1031      0      0       28.943  0.0   43    0   15    0       28.943  0.0   43    0   15    0   6
*RETDSEN    1021      0      0        1.195  0.0    0    0    0    0        1.195  0.0    0    0    0    0   6
QMHSNSTA     270      0     27        6.392  0.0    0    0    0    0        6.645  0.0    0    0    0    0   6
*MATINVIF      9      0      0           10  0.0    0    0    0    0           10  0.0    0    0    0    0   7
*SNSEXCPD      9      0      0           73  0.0    0    0    0    0           73  0.0    0    0    0    0   7
*SNDPRMSG      9      0      0          169  0.0    0    0    0    0          169  0.0    0    0    0    0   7
NLLNGDIS   21383 151905    330      991.515  1.2    0    0    2    0    6.908.450  8.3  802    5  188    0   5
*MODEXCPD    330      0      0          388  0.0    0    0    0    0          388  0.0    0    0    0    0   6
QWCCVTDT   21383  21383      0       95.741  0.1    0    0    0    0      280.651  0.3    0    0    0    0   6
QWCSCVTR   21383      0      0      184.909  0.2    0    0    0    0      184.909  0.2    0    0    0    0   7
DCNVZON    21383  42766    330      442.412  0.5    0    0    0    0    2.906.180  3.5    0    0    1    0   6
*MODEXCPD    330      0      0          321  0.0    0    0    0    0          321  0.0    0    0    0    0   7
QRGXIOU    21383  21383    990      409.522  0.5    0    0    1    0      975.566  1.2    0    0    1    0   7
QWCSRTVR   21383      0    660      553.500  0.7    0    0    0    0      562.291  0.7    0    0    0    0   8
*RSLVSP      330      0      0        8.230  0.0    0    0    0    0        8.230  0.0    0    0    0    0   9
*MATPTR      330      0      0          560  0.0    0    0    0    0          560  0.0    0    0    0    0   9
*LOCK        330      0      0        2.845  0.0    0    0    0    0        2.845  0.0    0    0    0    0   8
```

Above is an example of the statistical part of IBM's PEX-report for run type *STATS *HIER. It is explained in details in the IBM Redbook "AS/400 Performance Explorer Tips and Techniques", SG24-4781. Below you will find a brief introduction.

The statistics for CPU time (in microseconds) and different types of I/Os appear in two groups: Inline statistics, and cumulative statistics. The inline statistics show the exact resources used by the program listed on each line. The cumulative columns show the total resources used by the program on that line plus all programs called directly or indirectly. If program A calls program B, which in turn calls program C, then the inline statistics for program A will show what program A used, and the cumulative statistics will show the totals for program A, B, and C.

PEX does not only show program names, it goes all the way down to "MI Complex Functions", which is very useful information when evaluating performance. The MI complex function names start with an asterisk, like *RETDSEN (Retrieve Data Set Entry) or *SETCR (Set Cursor). The PEX manual lists both the MI complex functions and the IBM programs (e.g. QDBGETKEY, Data base get by key).

Columns 2, 3, and 4 show the number of times the program was called, how many program calls the program made, and how many time MI complex functions were used.

Since this is the output from a *HIER (hierarchical) PEX run, the rightmost column contains the call level. In most applications there will be several programs calling each other, and often more than one program will read data bases files, meaning that a program like QDBGETKEY will occur many times. Following the call level numbers you can see how many reads each program completed.

Initially however an overview is useful – which is provided in the next report.

PEX program summary by cumulated CPU

```
05-05-17  12:53:51              GiAPA          PEX summary by program, sorted by cumulated CPU              Page 1

        Times   Calls MI CPLX  <---------- Inline Stats ----------> <---------- Cumulative Stats ----------> Text for IBM pgm.s
Name      called  made  issued     CPU time  CPU    DB   DB   NBD  NDB    CPU time  CPU    DB . DB   NBD  NDB
                        microsec.s pct   SIO  AIO  SIO  AIO   microsec.s  pct   SIO  AIO  SIO  AIO

NNT920       1 73992     0     525,452  2.5    0    0    2    0  20,677,822 100.0  2585  96  491   0
NNT921     21382 175294    330    1,438,765  7.0    0    0   22    0   18,791,504  90.9  2147  78  455   0
DCNVZON    65539 131078    1008    1,348,778  6.5    0    0    0    0    8,908,096  43.1    0   0    2   0
NLLNGDIS   21860 157812    331    1,036,386  5.0    0    0    3    0    7,195,259  34.8   803   6  194   0
QDBGETKY   226934  26361    7293    5,008,758 24.2  2229   12  337    0    5,795,447  28.0  2323  12  366   0 DATA BASE GET BY KEY
DRMNGTZN   65539  65539    1008     568,309  2.8    0    0    0    0    4,565,179  22.1    0   0    0   0
QMHRTVM    65539  65539       0     394,663  1.9    0    0    0    0    3,995,992  19.3    0   0    0   0 QMHRTVM HDR ILE/COBOL
NLLNGSET   21384  64186     329     440,235  2.1    0    0   17    0    3,669,429  17.8    3   0   26   0
QMHRTMSS   65539      0   5040    3,551,336 17.2    0    0    0    0    3,601,327  17.4    0   0    0   0 RETRIEVE MESSAGE
QRGXIOU    65539  65539   3024    1,256,020  6.1    0    0    2    0    2,993,133  14.5    0   0    2   0 RPG IN/OUT OF DTAARA
QWCSRTVR   65539      0   2016    1,698,825  8.2    0    0    0    0    1,725,500   8.3    0   0    0   0 RETR. DATA AREA ROUTINE
NLLNGALO     477   7228      1      101,738  .5    0    0   20    0    1,407,998   6.8  259  70   69   0
QWCCVTDT   87875  87875       0     361,356  1.8    0    0    1    0    1,071,942   5.2    0   0    1   0 CONVERT DATE FORMAT API
QDBUDR     29145      0    550     852,299  4.1  177   79   32    0     859,905   4.2  177  80   32   0 DATA BASE UPD/DLT/RLS
QMHSNSTA   28711      0   1272     729,915  3.5    0    0    2    0     741,185   3.6    0   0    2   0 SEND STATUS MESSAGE
QWCSCVTR   87875      0      0     710,581  3.4    0    0    0    0     710,581   3.4    0   0    0   0 CONV TIME SUPP ROUTINE
QRGXTIME   22336  22336       0     145,313  .7    0    0    0    0     392,484   1.9    0   0    0   0
OIS910       475   5700      1      58,362  .3    0    0    0    0     375,110   1.8   14   9    4   0
QDBPUT       476      0      1     126,864  .6   78    2    6    0     127,488   .6   81   2    7   0 DATA BASE PUT UNBLOCKED
QDBGETSQ    4733    957    384      83,832  .4    4    2    7    0     101,973   .5    4   2   12   0 DATA BASE GET UNBLOCKED
*SETCR      4044      0      0     78,576  .4   94    0   27    0      78,576   .4   94   0   27   0 SET CURSOR
*RSLVSP     2016      0      0     65,461  .3    0    0    0    0      65,461   .3    0   0    0   0 RESOLVE SYSTEM POINTER
```

In the above report all the detail lines from the PEX statistics report have been summarized by program name and sorted descending by cumulative CPU usage. Text has been added for the IBM programs and MI complex functions.

Example of PEX *STATS analysis

The following analysis will explain how the reports are used simultaneously: the name of the 3rd program name in the above report, DCNVZON, indicates that a conversion was occurring, maybe time zones. It seems odd that 43 pct. of the CPU used by the application should be used for converting between time zones. Time stamps are stored in a rather special format, and calculations using them are quite CPU intensive, but this still looked excessive.

We now wanted to explore which program performed all these time zone conversions. Therefore, we turned to the following report which shows that the 65539 calls to DCNVZON took place from six different locations. In addition, we can detect the call count for each, and only the initial three showed more than twenty thousand usages and call for a closer look. These three had record sequence numbers 141, 197, and 219 corresponding to the line numbers from the original PEX report.

Instead of turning to the PEX report, which often will be huge, we examine the fourth report. This is sorted by the sequence number of the original PEX report, but is much smaller because it only contains user programs and programs that used above 1 % cumulated CPU. By looking at the invocation level numbers we find:

- DCNVZON in line 141 had call level 6 → called from level 5 pgm NLLNGDIS in line 137.
- DCNVZON in line 197 had call level 5 → called from level 4 pgm NNT921 in line 128.
- DCNVZON in line 219 had call level 6 → called from level 5 pgm NLLNGSET in line 215.

It also becomes obvious that the same three programs, in addition to calling DCNVZON, also called QWCCVTDT, (Convert Date Format API), the same number of times, indicating that excessive resources were used on converting dates.

Non-IBM pgm.s and pgm.s using > 1% CumCPU, by Pgm.Name

```
05-05-17  12:54:06      GiAPA      Non-IBM pgm.s and pgm.s using > 1% CumCPU, by Pgm.Name                PAGE   1

  Rec.seq.  Program    Inv     Called      Calls     MI CPLX     Program CPU     Cumulative CPU   Cumulative  Cumulative
  on Splf.  name       Lvl     count       made      issued      microseconds    microseconds     Sync. I/O   Async I/O

(Additional lines have been removed for brevity)
     251  DCNVDAT     6        2         2         0           42              88          0        0
     428              4        872       874       36          11 686          26 517      1        0
     141  DCNVZON     6        21 383    42 766    330         442 412         2 906 180   1        0
     197              5        21 382    42 764    329         460 950         3 105 685   1        0
     219              6        21 384    42 768    329         409 955         2 649 467   0        0
     270              6        477       954       1           12 441          86 582      0        0
     307              7        477       954       1           11 048          75 583      0        0
     410              4        436       872       18          11 972          84 599      0        0
     406  CSELASJ     4        2 560     0         175         14 290          14 466      0        0
     150  DRMNGTZN    7        21 383    21 383    330         184 563         1 487 879   0        0
     206              6        21 382    21 382    329         202 912         1 586 177   0        0
     238              7        21 384    21 384    329         164 872         1 369 179   0        0
     289              7        477       477       1           5 575           42 852      0        0
     316              8        477       477       1           4 857           37 809      0        0
     419              5        436       436       18          5 530           41 283      0        0
     256  NLLNGALO    5        477       7 228     1           101 738         1 407 998   328      70
(Additional lines have been removed for brevity)
     166  QMHSNSTA    7        22 343    0         996         603 067         612 389     2        0
     143  QRGXIOU     7        21 383    21 383    990         409 522         975 566     1        0
     199              6        21 382    21 382    987         450 494         1 058 205   1        0
     221              7        21 384    21 384    987         356 670         870 037     0        0
     159  QRGXTIME    6        21 383    21 383    0           139 135         375 310     0        0
     139  QWCCVTDT    6        21 383    21 383    0           95 741          280 651     0        0
     160              7        21 383    21 383    0           76 264          236 175     0        0
     195              5        21 382    21 382    0           91 686          274 202     0        0
     217              6        21 384    21 384    0           86 245          246 833     1        0
     144  QWCSRTVR    8        21 383    0         660         553 500         562 291     0        0
     200              7        21 382    0         658         593 987         603 498     0        0
     232              8        21 384    0         658         502 500         510 094     0        0
                                    (Additional lines have been removed for brevity)
```

The above report is primarily used to identify where (based on line numbers) a given program name appears and when the program according to the previous report used excessive resources that it warranted further scrutiny.

The report provides the details for each usage of the program (number of times called and calls made, CPU usage, and I/Os).

Non-IBM pgm.s and pgm.s using > 1% CumCPU, by PEX line nbr.

05-05-17 12:54:07 GiAPA Non-IBM pgm.s and pgm.s using > 1% CumCPU, by PEX report line nbr. PAGE 1

Rec.seq. on Splf.	Program name	Inv Lvl	Called count	Calls made	MI CPLX issued	Program CPU microseconds	Cumulative CPU microseconds	Cumulative Sync. I/O	Cumulative Async I/O
127	NNT920	3	1	73 992	0	525 452	20 677 822	3 076	96
128	NNT921	4	21 382	175 294	330	1 438 765	18 791 504	2 602	78
130	QDBGETKY	5	66 949	270	2 052	1 764 103	1 800 887	1 223	2
137	NLLNGDIS	5	21 383	151 905	330	991 515	6 908 450	990	5
139	QWCCVTDT	6	21 383	21 383	0	95 741	280 651	0	0
141	DCNVZON	6	21 383	42 766	330	442 412	2 906 180	1	0
143	QRGXIOU	7	21 383	21 383	990	409 522	975 566	1	0
144	QWCSRTVR	8	21 383	0	660	553 500	562 291	0	0
150	DRMNGTZN	7	21 383	21 383	330	184 563	1 487 879	0	0
152	QMHRTVM	8	21 383	21 383	0	129 387	1 303 012	0	0
153	QMHRTMSS	9	21 383	0	1 650	1 157 221	1 173 624	0	0
159	QRGXTIME	6	21 383	21 383	0	139 135	375 310	0	0
160	QWCCVTDT	7	21 383	21 383	0	76 264	236 175	0	0
162	QDBGETKY	6	86 796	22 343	2 313	1 692 084	2 326 814	987	5
166	QMHSNSTA	7	22 343	0	996	603 067	612 389	2	0
195	QWCCVTDT	5	21 382	21 382	0	91 686	274 202	0	0
197	DCNVZON	5	21 382	42 764	329	460 950	3 105 685	1	0
199	QRGXIOU	6	21 382	21 382	987	450 494	1 058 205	1	0
200	QWCSRTVR	7	21 382	0	658	593 987	603 498	0	0
206	DRMNGTZN	6	21 382	21 382	329	202 912	1 586 177	0	0
208	QMHRTVM	7	21 382	21 382	0	143 705	1 382 969	0	0
209	QMHRTMSS	8	21 382	0	1 645	1 222 125	1 239 264	0	0
215	NLLNGSET	5	21 384	64 186	329	440 235	3 669 429	29	0
217	QWCCVTDT	6	21 384	21 384	0	86 245	246 833	1	0
219	DCNVZON	6	21 384	42 768	329	409 955	2 649 467	0	0
221	QRGXIOU	7	21 384	21 384	987	356 670	870 037	0	0
232	QWCSRTVR	8	21 384	0	658	502 500	510 094	0	0
238	DRMNGTZN	7	21 384	21 384	329	164 872	1 369 179	0	0
240	QMHRTVM	8	21 384	21 384	0	110 403	1 204 054	0	0
241	QMHRTMSS	9	21 384	0	1 645	1 078 426	1 093 651	0	0
247	QDBGETKY	6	21 416	16	658	328 125	332 469	11	0
251	DCNVDAT	6	2	2	0	42	88	0	0
256	NLLNGALO	5	477	7 228	1	101 738	1 407 998	328	70

The above report is used to establish the call sequence. For programs that were used more than what seemed reasonable, the call level (the 3rd column of the report) can be used to identify the level of the calling program. A program on call level 6 is called by the closest previous program on call level 5, etc.

Although the original PEX report also contains this information, this report is more user-friendly since all IBM supplied programs using less than 1 % of cumulative CPU have been removed. Therefore this report is considerably smaller.

GiAPA Menu Option 32: End PEX Statistical Data Collection

This option will simply terminate a collection of PEX statistical data, if the collection was started by GiAPA Option 31.

Note that the collection will automatically end within one minute if the job being analyzed ends.

GiAPA Menu Option 33: List Call Stack Based on PEX Data

The call stack of a job that was analyzed with a PEX statistical run can be reconstructed from the hierarchical PEX report. This option will generate a report showing how a given program

was called, provided that the necessary data are available following the use of GiAPA Menu Option 31.

Please note that this option as input data is using the last PEX report (= spooled file QPVPERPT) produced by any job with GiAPA Menu Option 31, i.e. the report must still be on the output queue. Furthermore Option 33 can only be used when a unique job-id was used when running Option 31. The reason is that if a generic job name, user name, or job number is used, the PEX report will not generate a hierarchical report showing the invocation levels.

Use of Option 33 will cause this prompt display to appear:

```
                    Print call stack for program (GIAPA330)

Type choices, press Enter.

Program to find in PEX report  . PROGRAM        QWCCVTDT__       Name
Minimum usage to be reported . . CALLCOUNT      100__            1-9999
```

PROGRAM is the name of the program for which the call invocation sequence should be shown.

CALLCOUNT can be used to exclude call sequences where a program only was called a few times.

– – – – –

An example of using Option 33: A job shows an unexpected high number of date conversions, and the programmer cannot identify the responsible program. Run a PEX statistical run followed by GiAPA Option 33 to look for calls of QWCCVTDT, the API for date format conversions. The report will show which application programs called QWCCVTDT how many times.

```
GiAPA     Call stack for QWCCVTDT  called more than  100 times in PEX *STATS *HIER analysis  2008/05/18 23:30:23  Page   1
 PEX Line #  Level   Count  Program   Pgm-1    Pgm-2    Pgm-3    Pgm-4    Pgm-5    Pgm-6    Pgm-7    Pgm-8
     139    06  301,383  QWCCVTDT  KAMNGDIS  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     160    07  301,383  QWCCVTDT  QRGXTIME  KAMNGDIS  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     195    05  301,382  QWCCVTDT  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     217    06  301,384  QWCCVTDT  KAMNGSET  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     268    06      877  QWCCVTDT  KAMNGALO  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     299    07      876  QWCCVTDT  QRGXTIME  KAMNGALO  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     305    07      877  QWCCVTDT  KAMNGDIS  KAMNGALO  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     326    08      877  QWCCVTDT  QRGXTIME  KAMNGDIS  KAMNGALO  KAS921   KAS920   KAS920CL  KBS235CL  QCMD
     408    04      836  QWCCVTDT  KAS920   KAS920CL  KBS235CL  QCMD
*** Program QWCCVTDT  was found a total of     9 times in PEX report. Total number of calls were  1,209,875
```

The report shows up to the last 9 programs in the call stack.

GiAPA Menu Option 41: Start Trace of Job

(Good for analyzing delays in jobs using very little CPU resources)

Introduction

The different GiAPA reports available from menus 10 and 20 can display resource use in numerous different ways. Included are vast possibilities for different selections and sort sequences, and merely any excessive use of resources can be pinpointed. However, sometimes a job is running very slowly even though it only uses limited resources and the overall use of resources on the computer looks reasonably low.

Such delays can be quite frustrating and sometimes very serious, and will normally be caused by frequent locks or seizes. A seize is an operating system internal lock of an object, e.g. of (parts of) an index during maintenance. Since the job being delayed by waits does not use any resources, the normal GiAPA reports will not observe anything unusual.

The “official” method to pinpoint locks is IBM’s very comprehensive performance analysis tool iDoctor. It is an excellent tool, amongst others capable of locating locks by object name and job name, but from a user point of view it has two drawbacks: (1) purchase of a license is required, and (2) with a user manual exceeding 1000 pages it takes considerable effort and knowledge to use iDoctor efficiently.

This is where a standard trace job may become very useful. The main problem with running Trace Job is that it generates an overwhelming amount of output in a minimum of time. If you run Trace Job against a batch job and print the results you probably generate a report of hundreds of pages.

The solution is to employ GiAPA: the Trace Job results can be written to a file, which subsequently is analyzed by GiAPA programs producing exception reports. These reports will typically within a few pages clearly show which function causes the delays.

How to use Start Trace of Job.

```
                  Submit trace job via SRVJOB (GIAPA410)

Type choices, press Enter.

Job name to be traced  . . . . . JOB       __________    Name, *
  User . . . . . . . . . . . . .             __________  Name
  Number . . . . . . . . . . . .             ______      000000-999999
Action when trace area full  . . TRCFULL   *STOPTRC      *WRAP, *STOPTRC
Trace job data file library  . . DATALIB > GIAPALIB__    Name
```

JOB Enter the job ID (name, user and number) of the job to be traced.

TRCFULL Only 16000 KB of trace data can be collected (operating system limit). When / if this maximum size is reached, trace will either terminate (***STOPTRC**) or the new data will overlay the oldest trace records (***WRAP**).

DATALIB is the library used to store the trace data; cannot be QTEMP.

GiAPA Menu Option 42: End Trace Job Data

Trace Job can be terminated in three different ways:

1. It will automatically terminate if the job being traced ends.

2. It will automatically terminate if *STOPTRC was used for the TRCFULL keyword (see above) and the maximum 16 MB of trace data has been collected.

3. It can be terminated by using GiAPA Menu Option 42, which will show a subfile of any jobs currently being traced, allowing selection of tracing to be ended.

GiAPA Menu Option 43: Analyze Trace Job Data

Using this option will display the following selection panel, where an option number in front of one of the members (each represent a Trace Job data collection) is used to select the required type of report.

In general it is a good idea to start with selection 1 “Elapsed+CPU time count”, since this report in one glance will inform you of any serious delays as well as of the magnitude of these delays. You need this information to determine

- if additional reports should be requested, and
- which selection criteria should be used for report number 2 (trace exceptions).

```
GiAPA (c) by          Selections for Trace Job Analysis              9/08/11
iPerformance               Data Libr.: GIAPALIB                      13:45:10

Thread: *FIRST                    Trace exception report selection:
(*FIRST, *LIST, or Thread-ID)     Include if CPU time >=         0 microseconds
                                     or elapsed time >=          0 microseconds
D=Display, P=PRINT: D             or HHMMSSssss between            and

 1=Elapsed+CPU time count    2=Trace exception report
 3=Program/module summary    4=Pgm/mod/proc. summary    5=File open/close
Opt   Member name    Run date    Time    Job name, user and number
 _    T143856344    2009-08-06 11:19:07 BRE_29314 QUSER 683533
 _    T303488359    2009-08-06 14:19:07 QZRCSRVS QUSER 053715
 _    T303488847    2009-08-06 14:27:36 QZRCSRVS QUSER 053738
 _    T303565972    2009-08-07 11:52:44 QPADEV0002 KAARE 106054
```

Thread: For a multithreaded job you may want to specify *LIST, which will run a query detailing how many trace records were found for each thread. You could then copy-and-paste the thread-id wanted from the query report shown to the Thread selection field.

Please note that the selection criteria in the upper right only apply for report selection 2 “Trace exception report”. At least one of the criteria must be used when specifying selection 2.

Do not set the selection specifications for CPU and elapsed time too low to include all data, because the resulting report would be too exhaustive being comprised of several hundred pages, and you would lose sight of the relevant information.

It is therefore important to specify reasonable selection values, which are very dependent on the model (i.e., speed or CPW) of the computer. As explained above, you should run report

selection 1 to determine which values to specify in this field; please refer to the explanation following the example output of report selection 1 below.

The selection specification for “HHMMSSsss between ..... and .....” should in most cases be left blank, at least in the first run of report selection 2. It is intended for cases where events at a certain point of time attract enough interest that you want to see all details recorded within a given interval. To avoid too much output you should only specify a very limited time interval when you use this selection, normally only one or a few milliseconds before and after the event you want to analyze in details. Faster machines may generate hundreds of print lines per millisecond.

Trace Job – Statistics for Elapsed and CPU time

This report details counts for the number of times different (groups of) intervals were found in the trace data. On any faster machine you will normally see a lot of trace records per millisecond, i.e. the highest values will be the counts for intervals showing between 0 and 5 microseconds time difference between two trace records.

The purpose of the report is to provide an overview of which time intervals were found how often, the aim being to determine which selection time limits (in microseconds) to use for report selection 2, the Trace Job Exception Report.

```
GiAPA (c) by          Trace Job Timing Statistics for Elapsed and CPU time used per Trace Record          9/08/11
iPerformance   Job QZRCSRVS  QUSER     053715 Thread 0000000000000005  20090806  Datalib GIAPALIB   Mbr T303488359  13:57:34
  Interval     Elapsed     CPU   I    Interval     Elapsed     CPU   I    Interval     Elapsed     CPU
   length      count       count  I    length      count       count  I    length      count       count
  200 - 250 ms       1         0  I   301 - 400 us       61        23  I
   60 -  80 ms       4         0  I   251 - 300 us        5        12  I
   50 -  60 ms       4         0  I   201 - 250 us       15        67  I
   40 -  50 ms      13         0  I   151 - 200 us       59        11  I
   30 -  40 ms      20         0  I   101 - 150 us       35        70  I
   25 -  30 ms       5         0  I    76 - 100 us       36        29  I
   20 -  25 ms      12         0  I    51 -  75 us       62        40  I
   15 -  20 ms       5         0  I    41 -  50 us       41         7  I
   10 -  15 ms       9         0  I    31 -  40 us       55        61  I
    8 -  10 ms       1         0  I    21 -  30 us       75        74  I
    5 -   6 ms       3         1  I    16 -  20 us      164        46  I
    4 -   5 ms       2         0  I    11 -  15 us      722        72  I
    3 -   4 ms       2         1  I     6 -  10 us    2.292       930  I
  2.5 - 3.0 ms      52         0  I     0 -   5 us  126.667   129.002  I
  2.0 - 2.5 ms       1         2  I
  1.5 - 2.0 ms       1         2  I
   1 - 1.5 ms        2         0  I
  601 - 800 us       5         0  I
  501 - 600 us       6         0  I
  401 - 500 us      18         5  I
2=Cmd.Line     F3=Exit
```

Based on the above example, 50000 microseconds (= 50 milliseconds) selection limit for elapsed time might be an optimal choice, causing only 9 (= 1 + 4 + 4) intervals to be reported.

If you want to check which programs used the most CPU, you could instead (or also) specify a CPU selection limit of 1500 microseconds, which would cause 6 intervals to be reported.

In the example below you can see these two selection limits used for the report.

Each interval reported consists of 19 lines. They show the calls and returns leading up to and immediately following the line where you find the value exceeding the selection limit(s). The line causing the selection will be line number 14.

Trace Job – Exception Report

Below is an example of the trace job exception report from a model 595. It appears that 62898 microseconds elapsed and 250 microseconds CPU time was used by what appears to be an ASCII to UCS character conversion. In comparison with the other events shown on the page it is excessive although 63 milliseconds elapsed may seem negligible. In fact, this conversion alone would take close to 17 hours, if it should be run once for each record in a data base containing 1 million records.

GiAPA only prints a few lines surrounding each event where the elapsed time or CPU time exceeds the specified selection limits. Specifying a reasonably low selection criteria based on the data from the previous report will reduce thousands of trace records to just a few pages of output.

The selection parameters used are shown on the 3rd title line of the report. The separator lines made of hyphens are used to separate records from different intervals selected.

```
GiAPA (c) by      Trace Job Exception Report - Print if Time Interval Selected, or if Time Limit(s) Exceeded      9/08/11
iPerformance   Job QZRCSRVS  QUSER     053715 Thread 0000000000000005 on 2009-08-06  Lib GIAPALIB   Mbr T303488359   16:42:07

                                Specified selection limits:    50000      1500
 Time Stamp  CallLvl Program   Module     Procedure (or Program Text)          Elapsed us     CPU us  Disk I/Os     Waits
14:19:26.737577   11 QXXRTVDA   QXXRTVDA   QXXRTVDA                                  3
14:19:26.737580   10 HAMZES8UCS HAMZES8UCS loadtbl__Fv                             8        4
14:19:26.737588   11 HAMZES8UCS HAMZES8UCS __ct__9_DecimalTXSP5SP0_FCi              9        5
14:19:26.737597   10 HAMZES8UCS HAMZES8UCS loadtbl__Fv                            27       17
14:19:26.737624   11 QDCXLATE   QDCXLATE   API: CVT DATA                          53       34
14:19:26.737677   10 HAMZES8UCS HAMZES8UCS loadtbl__Fv                             3
14:19:26.737680   11 HAMZES8UCS HAMZES8UCS __ct__9_DecimalTXSP5SP0_FCi              4        1
14:19:26.737684   10 HAMZES8UCS HAMZES8UCS loadtbl__Fv                             3        1
14:19:26.737687   11 QDCXLATE   QDCXLATE   API: CVT DATA                          44       28
14:19:26.737730   10 HAMZES8UCS HAMZES8UCS loadtbl__Fv                             2
14:19:26.737733    9 HAMZES8UCS HAMZES8UCS LaunchReport__FiPPc                     9        5
14:19:26.737742   10 HAMZES8UCS HAMZCSSCSU AsciiToUCS__FPUci                       4        1
14:19:26.737746   11 HAMZES8UCS HAMZCSSCSU cp2ucs__FiUs                            4        1
14:19:26.737750   12 HAMZES8UCS HAMZCSSCSU map_unicode_1147_to_UCS2__Fv           62750      241        3
14:19:26.800500   11 HAMZES8UCS HAMZCSSCSU cp2ucs__FiUs                            8        1
14:19:26.800508   10 HAMZES8UCS HAMZCSSCSU AsciiToUCS__FPUci                       5        1
14:19:26.800513   11 HAMZES8UCS HAMZCSSCSU cp2ucs__FiUs                            3
14:19:26.800516   10 HAMZES8UCS HAMZCSSCSU AsciiToUCS__FPUci                       2
14:19:26.800518   11 HAMZES8UCS HAMZCSSCSU cp2ucs__FiUs                            2
                        ----------------------------------------                  +
F2=Cmd.Line     F3=Exit     F11=Show/hide details
```

Use of F11 will cause a second line showing the type of I/Os to appear:

(Please note that only part of the screen is shown)

```
 Time Stamp  CallLvl Program   Module     Procedure (or Program Text)          Elapsed us     CPU us  Disk I/Os     Waits
14:19:26.737742   10 HAMZES8UCS HAMZCSSCSU AsciiToUCS__FPUci                       4        1
    0 SDBR     0 SNDR     0 SDBW     0 SNDW     0 ADBR     0 ANDR     0 ADBW     0 ANDW     0 IOPW     0 SIOW
14:19:26.737746   11 HAMZES8UCS HAMZCSSCSU cp2ucs__FiUs                            4        1
    0 SDBR     0 SNDR     0 SDBW     0 SNDW     0 ADBR     0 ANDR     0 ADBW     0 ANDW     0 IOPW     0 SIOW
14:19:26.737750   12 HAMZES8UCS HAMZCSSCSU map_unicode_1147_to_UCS2__Fv           62750      241        3
    0 SDBR     3 SNDR     0 SDBW     0 SNDW     0 ADBR     0 ANDR     0 ADBW     0 ANDW     0 IOPW     0 SIOW
14:19:26.800500   11 HAMZES8UCS HAMZCSSCSU cp2ucs__FiUs                            8        1
    0 SDBR     0 SNDR     0 SDBW     0 SNDW     0 ADBR     0 ANDR     0 ADBW     0 ANDW     0 IOPW     0 SIOW
... etc., etc.
```

Trace Job – Program/Module Summary

```
09/08/11  13:40:22          GiAPA Trace Job Analysis by Program/Module, Sorted Descending by Elapsed Time          PAGE    1
Program   Module      Times     Elapsed    CPU   I/O-pending  Synchron.  Synchron.  Synchron.   Async.     Async.
Pgm. text  or proced.   called    microSecs microSecs waits     I/O waits  data base  non-DB    data base    non-DB
               23.811  152.851.695 26.795.390     1.228      393       358       735       129       107 Reads
                                                    653      2.332      425      2.451 Writes
 ----------------
QT3REQIO  QT3REQIO       335   121.051.721   163.548       0       0       0       0       0       0 Reads
T3-SNA MODULE FOR WS & WP                                          0       0       1       1 Writes
 ----------------
QDBCRTME  QDBCRTME        55    3.745.794  1.551.087      160      171      300      87       1       17 Reads
CREATE DATA BASE MEMBER                                          190      818      148      743 Writes
 ----------------
QDBGETM   QDBGETM        246    1.426.393  1.569.754      23       0       31       13       84       0 Reads
DATA BASE MULTIPLE GET (BLOCKED)                                   0       0       4       19 Writes
 ----------------
QCLCLCPR  QCLCLCPR        44    1.224.098    500.285       0       0       0      124       0       0 Reads
RUNTIME: CALL COMMAND PROCESSING                                   0      79       0       9 Writes
```

This report details the totals for elapsed time and for use of resources for all programs found in the trace data, sorted descending on elapsed time. It is very useful to determine which program/ module(s) are responsible for the most job delays. The first two lines indicate the grand totals, and the report should overall be self-explanatory.

Trace Job – Program/Module/Procedure Summary

The following report is slightly more detailed than the previous provided that the procedure name is also part of the control break. The 2nd line will therefore contain procedure name (if available), or the explanatory text for IBM programs not going down on procedure level.

```
09/08/14  19:37:53          GiAPA Trace Job Analysis by Pgm/Mod/Proc, Sorted Descending by Elapsed Time          PAGE    1
Program   Module      Times     Elapsed    CPU   I/O-pending  Synchron.  Synchron.  Synchron.   Async.     Async.
Pgm. text  or proced.   called    microSecs microSecs waits     I/O waits  data base  non-DB    data base    non-DB
               65.236    3.227.161    95.979       1       6       2       22       0       0 Reads
                                                      2       26       4       19 Writes
 ----------------
HAMZES8UCS HAMZES8UCS     6.647     460.857     6.778       0       0       0       0       0       0 Reads
__opi__9_DecimalTXSP10SP0_CFv                                      0       0       0       0 Writes
 ----------------
HAMZES8UCS HAMZES4P2U       235     401.762    38.734       0       0       0       0       0       0 Reads
XMLGetNameFromDictionary__FP9T_XMLRootUi                           0       0       0       0 Writes
 ----------------
QDBCRTME  QDBCRTME          1     291.172     3.157       1       5       2       4       0       0 Reads
CREATE DATA BASE MEMBER                                            1      14       4      15 Writes
 ----------------
HAMZES8UCS HAMZES4P2U     9.240     263.528      18       0       0       0       0       0       0 Reads
XMLReadChar__FP10T_XML_FILEiPi                                     0       0       0       0 Writes
 ----------------
HAMZES8UCS HAMZES8UCS     3.500     217.350     3.553       0       0       0       0       0       0 Reads __ct__9_DecimalTXSP10SP0_FCi
0       0       0       0 Writes
```

Trace Job – File Open and Close

Trace Job data includes information about which files are opened and closed when and by which jobs. If other reports indicated that (frequent) opening and closing of files might be a cause of delays, the program and file names involved can be obtained by requesting this report.

Note that the program name shown is the last user program name found before the open or close record in the trace data file. Program names starting with Q (= IBM) are disregarded. This is assumed to result in the correct information in +99 % of all cases, but if an application program name happens to start with Q, such program names will not be shown on this type of report.

GiAPA (c) by Trace Job Analysis File Open and Close 9/08/11
iPerformance Job QPADEV0002 KAARE 106054 Thread 0000000000000002 Datalib GIAPALIB Mbr T303565972 11:23:19

Time Stamp	Program Library	Program Name	Program Module	Program Procedure	File Name in Program	Ac-tion	File Library	File Name	File Mbr or Dev.
2009-08-07-11.53.05.446806	GIAPALIB	GIAPA141	GIAPA141	GIAPA141	GIAPA141D1	Close	GIAPALIB	GIAPA141D1	QPADEV0002
2009-08-07-11.53.05.448271	GIAPALIB	GIAPA141	GIAPA141	GIAPA141	QAFDMBRL	Close	QTEMP	QAFDMBRL	QAFDMBRL
2009-08-07-11.53.05.539948	GIAPALIB	GIAPA14E	GIAPA14E	GIAPA14E	QAFDMBRL	Close	QTEMP	QAFDMBRL	QAFDMBRL
2009-08-07-11.53.18.312202	GIAPALIB	GIAPA14E	GIAPA14E	GIAPA14E	QAFDMBRL	Open	QTEMP	QAFDMBRL	QAFDMBRL
2009-08-07-11.53.18.439383	GIAPALIB	GIAPA14E	GIAPA14E	GIAPA14E	QAFDMBRL	Close	QTEMP	QAFDMBRL	QAFDMBRL
2009-08-07-11.53.18.466938	GIAPALIB	GIAPA14E	GIAPA14E	GIAPA14E	QFMTSRC	Open	GIAPALIB	QFMTSRC	GIAPA14E
2009-08-07-11.53.18.486817	GIAPALIB	GIAPA14E	GIAPA14E	GIAPA14E	QFMTSRC	Close	GIAPALIB	QFMTSRC	GIAPA14E
2009-08-07-11.53.18.566710	GIAPALIB	GIAPA14E	GIAPA14E	GIAPA14E	QAFDMBRL	Open	QTEMP	QAFDMBRL	QAFDMBRL
2009-08-07-11.53.18.607503	GIAPALIB	GIAPA14E	GIAPA14E	GIAPA14E	QAFDMBRL	Close	QTEMP	QAFDMBRL	QAFDMBRL

GiAPA Menu Option 51: Collect File Check Data

This option collects data about all database files in one or more libraries, and produces a number of (exception) reports, primarily listing potential inefficiencies, but also providing an overview of all the files. Using this option will prompt the following CL command:

```
                    Rtv PF and LF mbr/accpth info (GIAPA510)

 Type choices, press Enter.

 Input library to be scanned  . . LIBNAME        *SELECT     Name, *ALLUSR,
*ALL...
 Store output data in library . . DATALIB        GIAPALIB    Name
 Job queue name . . . . . . . . . JOBQ           QBATCH      Name
   Library name . . . . . . . . .                  *LIBL     Name, *LIBL, *CURLIB
```

LIBNAME specifies the library containing the database files to be checked. More than one library can be checked, either by using special values *ALLUSR, *ALL, *LIBL, or *USRLIBL, or by leaving the default value *SELECT. This will display a subfile allowing selection of one or more libraries.

DATALIB is used to specify the name of the library to store the file check data. Since the data can get somewhat voluminous and probably only is used once to generate the reports, we recommend to create a new designated library only for this purpose, and delete it as soon as the reports are produced. Alternatively, GiAPA Menu Option 85 can be used to delete file check data.

JOBQ is self-explanatory – the file check data collection is always submitted to batch since the run may take some time if the selected libraries contain many files.

This is the subfile panel appearing if *SELECT is used for keyword LIBNAME:

```
GiAPA (c) by     Select libraries for database file checking          7/09/24
iPerformance                                                        13:27:59

 1=Select library (one or more)                      Data library: GIAPALIB

Opt  Library     Text
 _   ALFAPROD    Production library for shipping order system
 _   BETATEST    Test library for financial applications
 _   GAMMASYS    Payroll application
 _   GIAPALIB    GiAPA - Global iSeries Application Perf. Analyzer
 _   M3_PROD     ERP main program library
     Etc.
```

On the above selection panel one or more libraries can be selected. Press enter for batch job GIAPA512 to produce files with data from various DSPFD commands.

GiAPA Menu Option 52: Run File Check Analysis

```
GiAPA (c) by                        Select member(s) for Data Base File Information Report(s)                    9/09/28
iPerformance                                                                                                    19:26:05
                                                          Library contaning data base info file: GIAPALIB
1=Select one or more reports
     _   File overview report.    Only include files members >=    100000 K bytes (data + index space).
     _   Files not used recently. List files not used the last     365     days, or not having any  last used  date.
     _   Files with many extents. List files having at least       5      increments.
     _   Reorganize candidates:   Files having  20  per cent or     5000 deleted records.
     _   Files with many members. List files having at least       10      members.
     _   Files having the old *MAX4GB index type.
     _   Files showing more opens than I/Os.
                                                        If F19=Submit job used: Job queue QBATCH    in libary *LIBL
1=Select one or more members to include in above selected report(s)
     _   ASN               45 records collected on 090928          _   QUSRTOOL          34 records collected on 090928
     _   AS400PROFT        55 records collected on 090928          _   SOSYDEMO           6 records collected on 090928
     _   BCHQRYLIB         8 records collected on 090928          _   UTILITIES          8 records collected on 090928
     _   GIAPAGRAPH       37 records collected on 090928
     _   GIAPASAMPL       10 records collected on 090928
     _   GIAPASCHEN       30 records collected on 090928
     _   GIAPASPEND       36 records collected on 090928
     _   GIAPAUTILI      169 records collected on 090928
     _   GIAPAVRLEA       32 records collected on 090928
     _   ICEBREAK          58 records collected on 090928
     _   LIBL            2344 records collected on 090928
     _   PFROSRC          358 records collected on 090928
F2=Cmd line     F3=Exit     F19=Submit job (Run report generation in batch)     F21=Select *ALL reports for *ALL members
```

Using Option 52 will display the above screen, which overall should be self-explanatory. The upper part of the screen is used to select the desired reports, and to enter selection criteria. The screen appears with the above shown default selection values.

The lower part of the screen is used to specify one or more data collections to include in the selected report(s). The data collection names are either library names, or the “Special Value” used with the leading asterisk removed (shown in the above sample is LIBL, which replaces *LIBL that was used to request analysis of files of all libraries in the current library list.

F21 Select *ALL may be employed to select all reports for all members.

Note that GiAPA Menu Option 85 can be used to delete collected data.

File overview report lists all data files (source files not selected) together with most of their important attributes, number of active and deleted records, access path information, and statistics for opens and I/Os. The user may change the default selection criteria for the report (100 MB in member size) as required; specify zero to list all files.

```
                                    Explanation to Access Path field:
                                      T=Type Arrival/Keyed/EVI/Shared      M=Maintenance=Immediate/Rebuild/Delayed
                                      S=Size   0 *Max4GB   1 *MAX1TB       O=Omit or Selection Yes/No
                                      A=Alternative coll.seq. Yes/No       J=Joined logical file Yes/No


 09/09/30  11:11:59                              GiAPA          General info for data files, sorted by library/file name/member                                  PAGE    1
Library    File       File Crea Member     Crea  Last Jrn Nbr. Access-Path       Size in   Max   Nbr. of Deleted  Index   Owner of     Nbr of    Logical   Physical Write+Upd.  PFM
  name       name      attr YYMM name       YYMM  used img mbrs T-M-S O-A-J       K Bytes   RecL  records records  entries shared AP   opens      reads     reads  operations reorgs
ITEMSSCHEN ITEMS143P1 *PHY 0909 B71P03     0909 090925 N     2 A- -0 N-N-          67,588    148   456901       0        0                0         0         0          0      0
ITEMSSPEND ITEMS112P1 *PHY 0903 HS03020928 0903 090824 N     4 A- -  N-N-         190,892   8160    23946       0        0                0         0         0          0      0
ITEMSSPEND ITEMS112P1 *PHY 0903 HS03030000 0903 090824 N     4 A- -  N-N-         194,444   8160    24393       0        0                0         0         0          0      0
ITEMSSPEND ITEMS143P1 *PHY 0903 SEPT2009   0903 090310 N     2 A- -0 N-N-         149,508    148   986102       0        0                0         0         0          0      0
ITEMSSPEND ITEMS143P3 *PHY 0903 SEPT2009   0903 090918 N     2 K-I-0 N-N-         229,408     97  1768749       0        0                0         0         0          0      0
ITEMSUTILI TRCJOBV4G  *PHY 0907 TRCJOBOUT  0907 090824 N     1 K-I-0 N-N-          63,252   4992    51920       0        0                0         0         0          0      0
ITEMSPROD  ITEMS143P1 *PHY 0901 PRODUCTS   0901 090127 N     2 A- -0 N-N-          30,724    148   209666       0        0                0         0         0          0      0
ITEMSPROD  ITEMS143P3 *PHY 0901 PRODUCTS   0901 090823 N     2 K-I-0 N-N-          32,780     97   236759       0        0                0         0         0          0      0
QSYS       QADBIFLD   *PHY 0506 QADBIFLD   0308 090928 Y     1 K-I-1 N-N-          36,136   4876    94664    5230        0                8      5885       130       3547      0
                                                                TOTAL                994,732
                                                                COUNT 9
*** END OF REPORT ***
```

Files not used recently list data file (members) which have:

- create date < (retrieval date minus number of days specified in selection criteria), and
- last usage date blank or < (retrieval date minus number of days specified).

It is important to notice that GiAPA will call an IBM API to check whether a logical file (that may show no usage because it was not opened by a program) has been useful to the SQL / Query Optimizer in the process of selecting an optimal access method for the physical file member(s) referenced. If GiAPA reports that a file was not used within a given number of days, then any usage by the Query Optimizer has been checked before the file is included in the report.

Files with many extents is a list of files with many increments. If a large database is created with the default values of 10.000 for initial number of records, incrementing 1.000 records at a time, and the number of increments is set to the maximum, the file may end up with thousands of extents, which probably is inefficient.

Reorganization candidates reports files with many deleted records. CL source statements required to reorganize these files are generated and stored as file member **REORGANIZE** of file GIAPA528P1. Below is an example of a generated CL:

```
     ***************** Beginning of data *********************
1.00 RGZPFM FILE(ERP_PROD/ERPCUSTOM ) MBR(ERPCUSTOM )
2.00 RGZPFM FILE(ERP_PROD/ERPSALES  ) MBR(ERPSALEJAN)
3.00 RGZPFM FILE(ERP_PROD/ERPSALES  ) MBR(ERPSALEFEB)
4.00 RGZPFM FILE(ERP_PROD/ERPSALES  ) MBR(ERPSALEMAR)
5.00 RGZPFM FILE(ERP_PROD/ERPSALES  ) MBR(ERPSALEAPR)
     ******************** End of data ***********************
```

Many members lists files having more members than the number specified in the selection criteria.

Files with old index type: The newer index type runs up to 1TB and it also seizes (= locks) a smaller part of the index during index maintenance. The new index type should therefore be used for any larger files with frequent index maintenance.

CL source statements needed to upgrade these files to the new index are generated and stored as file member **ACCPMAX1TB** of file GIAPA528P1. Example:

```
1.00  CHGLF FILE(PFROSRC/QQQ3028   ) ACCPTHSIZ(*MAX1TB)
2.00  CHGPF FILE(ERP_PROD/ERPGGPHF  ) ACCPTHSIZ(*MAX1TB)
3.00  CHGPF FILE(ERP_PROD/ERPGPKGF  ) ACCPTHSIZ(*MAX1TB)
4.00  CHGLF FILE(ERP_PROD/ERPMLSCD  ) ACCPTHSIZ(*MAX1TB)
5.00  CHGPF FILE(ERP_PROD/ERPMSCOL  ) ACCPTHSIZ(*MAX1TB)
6.00  CHGPF FILE(ERP_PROD/ERPMSSCD  ) ACCPTHSIZ(*MAX1TB)
```

More opens than I/O: This report lists all files having been opened more often than the total number of I/O operations against the file.

GiAPA Menu Option 53: List Data Base Index Generations

These reports select all HotSpot call stack records containing the job function code IDX. This causes GiAPA to wait for an unavailable call stack while the job is running below the MI-level. By the end of the wait (where maximum wait time used is less than 5 minutes) the call stack may or may not be available, depending on whether the index generation has terminated. If the index generation completes before the end of the maximum wait time, the call stack will show which program caused the index generation. If not, this information will normally be obtained by a subsequent HotSpot, unless the job ends within 15 seconds, i.e. before the next HotSpot should occur.

IBM's definition for the IDX function code, which is followed by a "value", informs us that "The value is the name of the file associated with an index (access path) rebuild operation."

Index generations will normally use enough CPU that GiAPA HotSpots are triggered every 15 seconds. However, if GiAPA detects that an earlier HotSpot already is waiting for the return of a call stack, the text "*NotAvail" is inserted immediately in the active program name field, thereby completing the process of the new HotSpot.

Since the automatic GiAPA HotSpots "only" are acquired every 15 seconds, GiAPA will only detect index generations being active when a HotSpot occurs, suggesting that smaller index generations will not be listed. However, if they happen frequently (and therefore could be a performance burden) then they will of course be recorded from time to time. This is in line with the overall GiAPA principles: limiting the collection of information to every 15 seconds consumes an absolute minimum of resources and at the same time is normally enough to detect any grave or frequently occurring performance inefficiency.

Two reports are available from the selection panel (not shown here, it is like selection panels for other reports): index generations sorted by job ID and file name, and index generations sorted by file and job. The latter, also including an estimated total index generation time for each file, is shown below. The total time for index generations running across several HotSpots is summarized and shown on one line in the "Estimated duration" column.

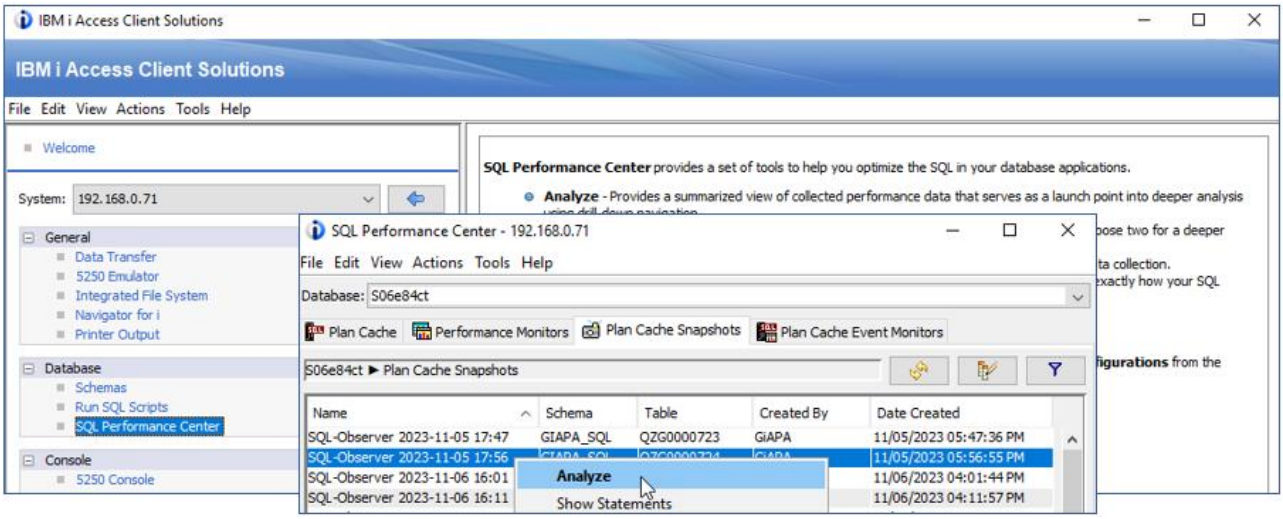
```
GiAPA (c) by            List Index Generations found by HotSpots      Input=CHECKINDEX          08-09-26
iPerformance        (Note: Duration is an estimate that could be up to 14 seconds incorrect)       17:24:32
 Physical    Job       User      Job   HotSpot requested  DataRcvd Duration Stk  Sta   Active     Non-Q    File name
file name    name      name      number YY-MM-DD hh:mm:dd  hh:mm:ss  Min:Sec Dpt  tus   program    program  duration total
JJBEX2P    RBNAFLKW   ROBOTADM   952594 08-09-04 14:00:30 14:01:30   1:00   0 RUN  *NotAvail
JJBEX2P    RBNAFLKW   ROBOTADM   958931 08-09-04 14:30:15 14:32:06   1:51   0 RUN  HITCA1C    RBT637
JJBEX2P    RBNAFLKW   ROBOTADM   965553 08-09-04 15:00:30 15:01:55   1:25   0 RUN  HITCA1C    RBT637
JJBEX2P    RBNAFLKW   ROBOTADM   972267 08-09-04 15:30:15 15:32:39   2:24   0 RUN  HITCA1C    RBT637
JJBEX2P    RBNAFLKW   ROBOTADM   993827 08-09-04 17:30:15 17:31:09     54   0 RUN  HITCA1C    RBT637
JJBEX2P    RBNAFLKW   ROBOTADM   997708 08-09-04 18:00:30 18:00:58     28  15 RUN  HITCA1C    RBT637      1:06:24
RODCLI     QPADEV030Z KORFOSJ    782111 08-09-03 14:35:15 14:35:17      2  18 RUN  HITCA1C    ROD000CL        2
SPFGBD     KORCC155   ACONROBOT  021395 08-09-04 22:07:00 22:07:07      7  13 RUN  HITCA1C    KORCC431
SPFGBD     KORCC155   ACONROBOT  841372 08-09-03 22:08:45 22:08:51      6  13 RUN  HITCA1C    KORCC431
SPFGBD     QPADEV0399 HAMZOIHH   946375 08-09-04 18:03:00 18:03:02      2  18 RUN  HITCA1C    ROD000CL
SPFGBD     RBDELEC112 ACONROBOT  119155 08-09-05 13:30:15 13:30:25     10  18 RUN  HITCA1C    KORCC431
SPFGBD     RBDELEC112 ACONROBOT  766849 08-09-03 13:30:15 13:30:21      6  18 RUN  HITCA1C    KORCC431
SPFGBD     RBDELEC112 ACONROBOT  854347 08-09-04 04:02:45  4:02:46      1  18 RUN  HITCA1C    KORCC431
SPFGBD     RBDELEC112 ACONROBOT  946111 08-09-04 13:30:15 13:30:22      7  18 RUN  HITCA1C    KORCC431        39
TAKSEND1   KDFA       SGIRFEL    022845 08-09-04 22:30:45 22:30:45      0   8 RUN  QDBCRTME   RODT02CL
TAKSEND1   KDFA       SGIRFEL    022845 08-09-04 22:32:00 22:32:00      0   8 RUN  QDBCRTME   RODT02CL        +
F3=Exit    F4=Prompt for Text
```

Please note that column "Duration" is an **estimate**. Since HotSpots only are generated every 15 seconds, the shown duration could be up to 14 seconds incorrect. An example of this situation is a HotSpot occurring at 09:32:45 that reports an index generation, which might in fact have started 09:32:31.

SQL Observer: Plan Cache dumps based on Job Watcher data

Introduction:

The intensive and growing use of SQL has made optimizing of frequently used SQL statements one of the most rewarding ways to conserve server resources. IBM's "SQL Performance Center" in the Database section of the Access Client Solution (ACS) tool is the gold standard for analysis of Access Plan snapshots from the Plan Cache. ACS is indeed the reason why analyzing the efficiency of SQL statements was not part of the initial design of GiAPA.



Collected Plan Cache data available within ACS.

The Plan Cache data which documents the access methods selected by the Query Optimizer is dynamically maintained in main storage when SQL is running. When the run environment changes, for example due to other jobs running, the Access Plan for an SQL statement may change. This results in the generation of new Plan Cache data, often causing a noticeable change of the run time.

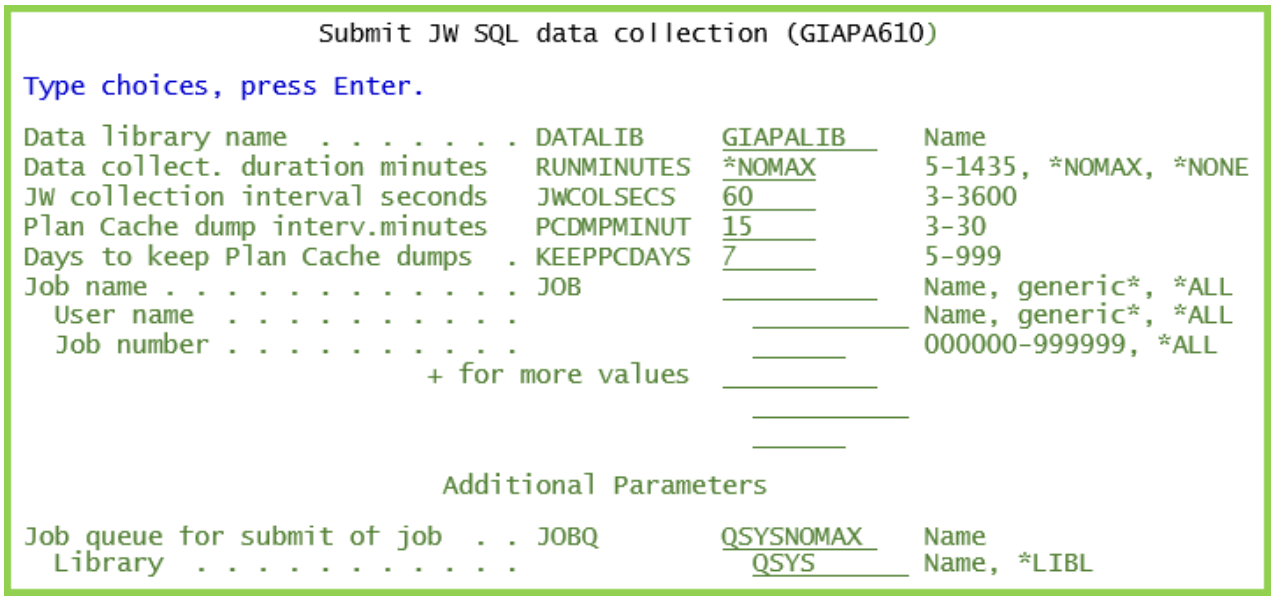
However, the Plan Cache data is not dumped automatically given it would consume excessive resources. An IBM performance expert recently suggested that a tool offering an automated and user-controlled dumping of Access Plans that might be necessary for analysis could be a popular option. This idea is implemented as GiAPA's "SQL Observer" available from the GiAPA Menu or through three CL commands.

The unique QRO code identifying an SQL activity (= one or more SQL-statements for a job and table) must be supplied when requesting a dump of Access Method information. Therefore, the first step for GiAPA's SQL Observer is to run IBM's Job Watcher, requesting the QRO code(s) for job(s) specified by the user. At the same time the user defines the frequency for returning Job Watcher data and for dumping Access Plans. An additional parameter defines the number of days the collected data is kept.

A special situation may justify collection of data every few seconds e.g. for one or a few jobs. This results in very detailed information collected for these selected case(s) without using excessive resources for the data collection overall. For the normal everyday workload, collecting data every two or five minutes may be sufficient.

One of the columns available within the collected Job Watcher data contains the Current User Name, which often is wanted in connection with analyzing heavy resource usage. GiAPA's SQL Observer also includes displaying user names per job and collection interval.

GiAPA Menu option 61 – Submit SQL Observer Collection



DATALIB defines the name of the library where the collected Plan Cache data is stored.

RUNMINUTES defines how long time the data collection should run. If *NOMAX is used for RUNMINUTES, the collection will continue until stopped by using command GIAPA630. This command can also be initiated from GiAPA Menu option 63.

Specify *NONE to only remove data older than KEEPPCDAYS – no collection will be started.

JWCOLSECS defines the interval in seconds between each collection of Job Watcher data. This obviously also affects the resources used by the collection – very frequent collections imply somewhat higher CPU usage and data volume.

PCDMPMINUT defines how often the Plan Cache dumps is scheduled. The Job Watcher data collection routine will be interrupted shortly, allowing GiAPA to dump the Plan Cache data for the QRO codes collected.

KEEPPCDAYS defines how long time the Plan Cache data fetched by GiAPA is stored. GiAPA will automatically delete expired collections.

JOB may be used for selection of max 20 (generic) job names, user names, and/or job numbers, thereby excluding all other jobs from this SQL Access Plan data collection.

Please refer to the job log in case of any errors – keyword parameters from this command are used to generate an ADDJWDFN command – IBM's rules for that command apply also here.

JOBQ has as default QSYSNOMAX defined within subsystem QSYSWRK – this is the queue normally used for, e.g. performance data collection.

An SQL activity can be based on one or more SQL statements, together defining the SQL function. GiAPA does not save more than five statements per QRO code, and displays only the first three codes which normally is sufficient to identify the case.

A prerequisite for correct interpretation of the results is understanding how the data collection works. The illustration below provides a quick overview.

A unique 4-bytes hexadecimal QRO code identifies an SQL activity (= one or more statements) that within a certain job accesses a given set of data consisting of one or more file members. The QRO codes are needed to request plan cache dumps of the access plans used.

Every JWCOLSECS (Job Watcher Collection Seconds) interval, Job Watcher saves the following details for the selected jobs

1. the name, user, and number for active jobs using SQL,
2. the SQL statement(s) "in progress" or "last completed" for each job, and
3. the unique QRO code identifying each SQL activity.

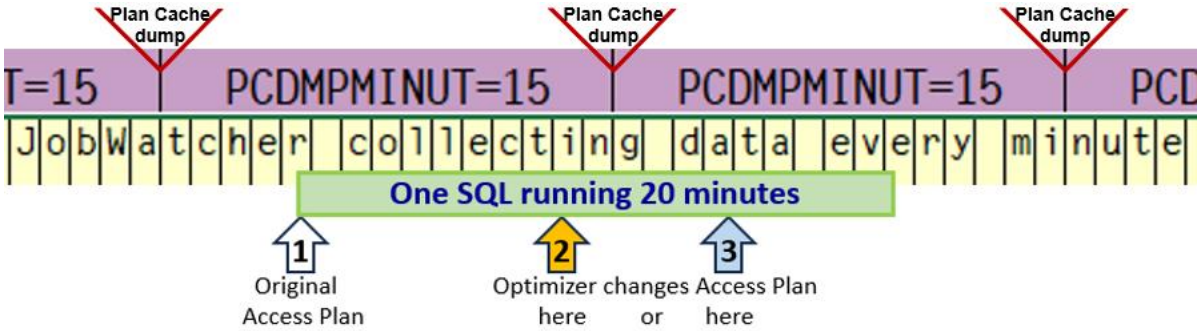
SQL Observer's default value of 60 seconds between each JW collection will normally suffice. Only jobs starting and ending within one interval are not included. To analyze SQLs within longer running jobs only, a larger interval value can save resources while still supplying sufficient data.

The JW data collection is interrupted at the end of every PCDMPMINUT (Plan Cache Dump Minutes) interval, where control is returned to GiAPA's SQL Observer to allow dumping of plan cache data for the QRO codes collected by JW during that PCDMPMINUT interval. The dumped access plans are needed as input for the IBM ACS SQL Performance Center if further analysis is needed.

An access plan has a unique plan number and consists of several 10K+ rows/records containing many columns/fields. Although plan cache dumping does not use much CPU, frequent collections for many jobs may obviously occupy much space. Each access plan is only dumped once per PCDMPMINUT interval.

The below example has JWCOLSECS set at 60 (Job Watcher data saved once a minute), and uses PCDMPMINUT=15 thereby passing data to the SQL Observer every 15 minutes.

The green square below illustrates a job running an SQL statement for 20 minutes. The narrow yellow fields represent one minute each, illustrating the JWCOLSECS=60 intervals. Job Watcher stores QRO code(s) representing active SQL(s) at the end of each 60 seconds interval. SQL Observer receives this data at the end of each PCDMPMINUT interval, and requests access plan dumping for each QRO code.



If re-optimization takes place at arrow 2 (orange), both access plans will be dumped at the end of the first PCDMPMINUT interval where the job is active, and only the new access plan is dumped at the end of the next PCDMPMINUT interval.

However, if the re-optimization takes place at arrow 3 (blue), the first plan cache dump will only include the original access plan, and both plans will be dumped at the end of the following interval.

GiAPA Menu option 62 – Display Collected Plan Cache Data

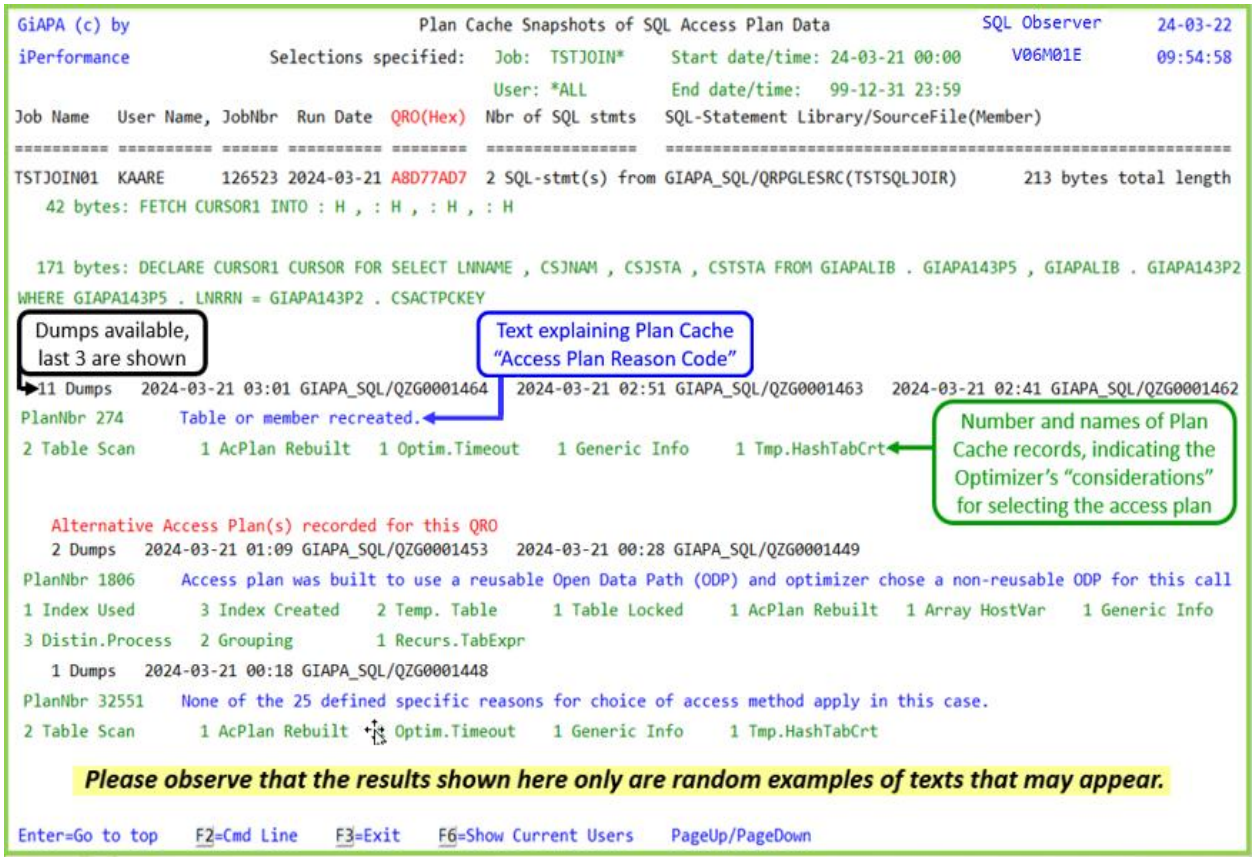
```
                    Show Collected Plan Cache Data (GIAPA620)

Type choices, press Enter.

Data library name  . . . . . . . DATALIB     GIAPALIB      Name
Only multiple Access Plan QROs   ONLYMULTIP  *NO           *Yes, *NO
Select (generic) job name  . . . JOBNAME     *ALL          Name, generic*, *ALL
Select (generic) user name . . . USERNAME    *ALL          Name, generic*, *ALL
Select date/time YYMMDDhhmmss  . STARTTIME   010101000000  Character value
Select date/time YYMMDDhhmmss  . ENDTIME     991231235959  Character value
```

DATALIB defines the library containing the dumped Plan Cache data and the GiAPA tables used to control the display of the results.

ONLYMULTIP allows selection of only re-optimized QROs (more than one Access Plan saved).

JOBNAME and **USERNAME** allow (generic) selection of the jobs to be shown.

STARTTIME and **ENDTIME** allow defining time limits for the data to be shown.



The pages are displayed in ascending sequence by Job Id and QRO code. The latest 3 access plans are shown – in most cases there is only one meaning that the plan was not changed.

Each page contains the data belonging to one QRO code (= SQL activity) within a job. One job may have accessed many different SQL statements, each resulting in a displayed page. Data from a maximum of three different Access Plans are shown.

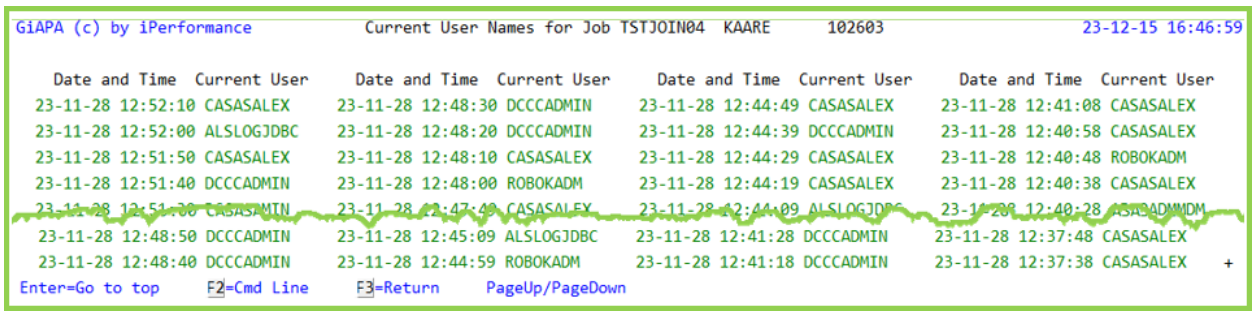
The SQL statement(s) belonging to the QRO code are shown in green. A QRO can cover several SQL statements and rarely more than three, which is the maximum displayed.

The documentation of the Plan Cache dumps collected includes the date, time, and names of the latest three files containing Plan Cache snapshots. This information, together with the QRO code, is necessary to locate the corresponding data within the IBM ACS SQL Performance Center, when a performance analysis is required.

If an Access Plan is re-optimized, data for a maximum of two additional plans are displayed which normally is sufficient. If more re-optimizations are expected, any remaining may be seen by using the STARTTIME and ENDTIME keywords to limit the time frame.

When different jobs run the same SQL statement(s) against the same set of data, the same QRO code(s) are saved repeatedly, leading to identical results shown for several jobs.

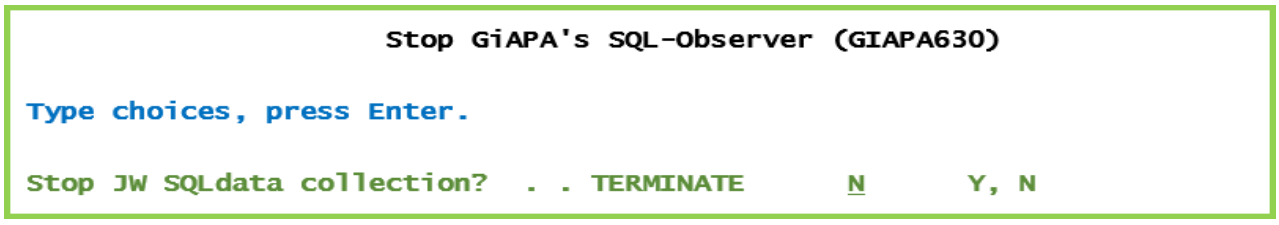
F6= Show Current User from the above panel displays the following panel with four columns of current users and the date and time where the users were attached to the job.



Additional details available in these &DATALIB files, easy to access by SQL or Query:

GIAPA612P1	Dumped Plan Cache Snapshot per job and QRO code
GIAPA612P2	SQL-statements and their assigned QRO code
GIAPA612P5	Current user names from Job Watcher QAPYJWTDE

GiAPA Menu option 63 – Stop SQL Observer collection



Use of command GIAPA630 TERMINATE(Y) will cause an active SQL Observer collection of Plan Cache data to terminate at the end of the current PCDMPMINUT interval.

GiAPA Menu Option 64: Start RUNQRY and WRKQRY Tracking

To improve Query/400 performance one of the problems in many installations is identifying who is using which queries and how often.

Using GiAPA Menu Option 64 allows you to collect exactly this type of information. The following panel, describing exactly how it is implemented, will appear:

```
GiAPA (c) by  Start tracking use of RUNQRY and WRKQRY commands      8/04/04
iPerformance                                  21:44:03

  It is a policy rule for GiAPA never to make any changes or updates to any
  programs, files, or other objects that do not originate from GiAPA.

  Tracking of use of queries is a utility implemented through the command
  validity program option for the WRKQRY and RUNQRY commands. No validation
  is made, but a record with job, user, date/time, query and file name (where
  available) is written to file GIAPA642P1, which then can be listed using
  GiAPA option 66 or a user written query to show the use of queries. In case
  of any error during the processing the program will just return.

  Since a command validation program is used to implement this option, the
  value for keyword VLDCKR for commands RUNQRY and WRKQRY must be modified
  when starting or ending the tracking.

  Pressing F23 now will display the needed command modification prompts, but
  it is the responsibility of the user to use (=accept) the CHGCMD function.
  (To run the change you must hit ENTER for both commands.)

F3=Exit
```

Before running this option you may want to use DSPCMD to verify that no command validation program is specified for commands RUNQRY and WRKQRY. The command prompt to modify the RUNQRY and WRKQRY commands has the following appearance:

```
                    Change Command (CHGCMD)

Type choices, press Enter.

Command . . . . . . . . . . . . CMD       > RUNQRY
  Library . . . . . . . . . . .             *LIBL
Validity checking program  . . . VLDCKR     > GIAPA642
  Library . . . . . . . . . . .           >   GIAPALIB

                                                                    Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display      F24=More keys
```


GiAPA Menu Option 65: End RUNQRY and WRKQRY Tracking

To end tracking of when and who used the RUNQRY and WRKQRY commands is very similar to the corresponding start command (see previous page). This is the command prompt screen:

```
                    Change Command (CHGCMD)

Type choices, press Enter.

Command . . . . . . . . . . . . CMD        > WRKQRY
  Library . . . . . . . . . . .             *LIBL
Validity checking program . . . VLDCKR      > *NONE
  Library . . . . . . . . . . .


                                          Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Note that Option 65 only stops the data collection. To delete the collected data you must use Option 87.

GiAPA Menu Option 66: List RUNQRY and WRKQRY Usage

When the use of RUNQRY and WRKQRY has been actively tracking for a while, you can use Option 66 to obtain the following report:

```
08/04/04  22:20:22    GiAPA (c) by iPerformance      List tracking of RUNQRY and WRKQRY commands      PAGE   1

Command  <--- Job using the query -->  Run    Run    Job  Query     Query     QryFile    1st file   Nbr of files
used     Name     User    Number  date   time   Type library   name      library    specified  specified
WRKQRY   QPADEV0002 PALLE     074873  050401 095254  I                                    0
RUNQRY   QPADEV0003 MICHAEL    074874  050401 095425  I  GIAPALIB   GIAPA43E                    0
RUNQRY   QPADEV0003 SVEN      074874  050401 095431  I  GIAPALIB   GIAPA43F                    0
RUNQRY   QPADEV0002 KAARE     074873  050401 115244  I  *LIBL      GIAPASRC   GIAPA091P1      1
RUNQRY   QPADEV0003 LONNY     075215  050404 222003  I  *LIBL      GIAPASRC   GIAPA091P1      1
RUNQRY   QPADEV0002 SCHIG     075208  050404 222021  I  GIAPALIB   GIAPA660                    0
 Etc., etc.
* * *  E N D  O F  R E P O R T  * * *
            Note: Extra cmd validation at SBMJOB may cause batch queries to appear twice
```

Note that CL commands are also validated when jobs are submitted. Since a command validation program is used to track the usage, SBMJOB of RUNQRY or WRKQRY will cause the command to be tracked twice, at submit time and at run time.

GiAPA Menu Option 71 – 72 – 73: Export and Import GiAPA Data

You may want to copy/send GiAPA performance data to another machine, e.g. to your software supplier when an error occurs or if you need assistance. Alternatively, you may want to consolidate results from different LPARS on a single machine.

GiAPA Menu Option 71: Export GiAPA Raw Performance Data

```
GiAPA (c) by    Save raw GiAPA Performance Data into Savefile       08/05/24
iPerformance          for Export or Offline Storing                 20:52:43

Save file: __________     Observe: The save file       Data Library: ANYLIBRARY
Savf library: __________  is cleared (or created)

 1=Select collected GiAPA performance source data members to export
Opt  Member     Size in Kb   Text
 _   PF09160005    115.928   Pfr.data from 040916 at 000500
 _   PF09170005    129.584   Pfr.data from 040917 at 000500
 _   PF09200005    121.370   Pfr.data from 040920 at 000500
 _   PF09210005     22.168   Pfr.data from 040921 at 000500
 _   PF09210840      4.662   Pfr.data from 040921 at 084003
 _   PF09210915     98.772   Pfr.data from 040921 at 091505

 F3=Exit (Cancel without exporting)     F23=Also delete member(s) after save
```

The above panel should be self-explanatory. The data library was specified on the GiAPA Menu. The parameters save file and library are mandatory, and the save file specified will be cleared (if it exists) or created. One or more members may be selected.

F23=Also delete will delete the members from file GIAPA112P1 once they have been saved into the save file specified. Should this option have been selected by a mistake, the members can still be recovered from QTEMP for the job which just used the command.

GiAPA Menu Option 72: Export Expanded GiAPA Analysis Results

This option is overall similar to Option 71 shown and described above, except that it has no F23=Delete function.

GiAPA Menu Option 73: Import GiAPA Raw Data or Results

```
                  Import GiAPA performance data (GIAPA730)

Type choices, press Enter.

Save file with GiAPA data  . . . SAVEFILE       __________      Name
  Library  . . . . . . . . . . . .                *LIBL         Name, *LIBL
Restore GiAPA data to library  . DATALIB      > ANYLIBRARY      Name
```

The same option is used to import both raw data and databases with analyzed result data.

GiAPA Menu Option 74: Maintain Loop Trap Exceptions

GiAPA's data collection includes a loop trap function that can send a message to the system operator if a job seems to be looping. This option is requested when starting a performance data collection (GiAPA Menu Option 11), and is described in details in the respective section of this manual.

Basically, the assumption is that if a job uses excessive CPU, but does not read any new data, then it is most likely looping. However, certain jobs can show such behavior despite running correctly. This may be the case for jobs that process transactions from a queue and do not need to access any files. The names of such jobs should be included on the exception list below.

GiAPA Menu Option 74 will display the following panel. A job name entered on this panel will immediately be disregarded by the loop trap function. If a job name is deleted from the list, it will normally not take effect before either that job or GiAPA data collection is restarted. Use of F23 will allow specification of job names including lower case letters.

```
GiAPA (c) by            Maintain Loop Track Exceptions              8/04/24
iPerformance                                                       00:58:22

   List of job names that never should cause a loop trap warning to QSYSOPR

 Maintenance of job name list: Change by overtyping, delete by blanking,
                               add by entering new name in empty space

 ANYJOBNAME   __________   __________   __________   __________   __________
 DATAQJOB     __________   __________   __________   __________   __________
 DOMINO       __________   __________   __________   __________   __________
 JOBNAMEXYZ   __________   __________   __________   __________   __________
 __________   __________   __________   __________   __________   __________
 __________   __________   __________   __________   __________   __________
 __________   __________   __________   __________   __________   __________
 __________   __________   __________   __________   __________   __________

Enter=Update exception list    F3=Exit without update    F23=Allow lower case
```

GiAPA Menu Option 75: Maintain HotSpot Exceptions

GiAPA's data collection fetches additional call stack and file usage information whenever a job exceeds the limits specified for CPU % within a 15 seconds' interval.

However, a run pattern tricking HotSpots may be perfectly normal for a few jobs, in which case you will want to suppress the HotSpot for these jobs. This will result in decreased resource usage during data collection / analysis and less space for storing the resulting data.

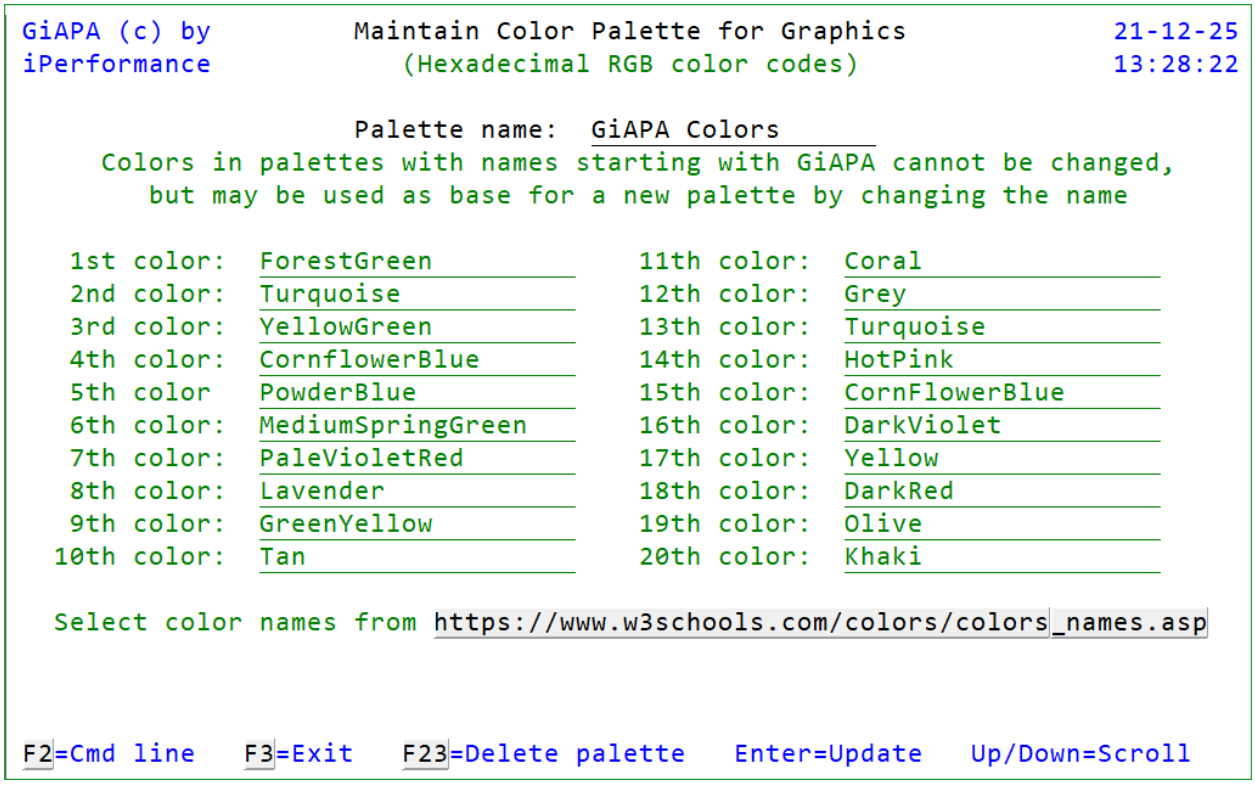
GiAPA Menu Option 75 will display a panel very similar to the panel shown just above, but in addition to job names you can also specify user names for jobs that do not require HotSpot data to be collected.

Mirroring software is a typical example of jobs that probably should be entered on the exclusion list. In most cases this is easily done by entering the user name for the mirroring jobs. To this end GiAPA is shipped with two such user names defined in this exception list: MIMIXOWN and VSIOWNER, both of which are user names of two common mirroring software products.

GiAPA Menu Option 76: Maintain Color Palette for Graphics

As described under Menu Option 28, the user may select a color palette to be used when a chart is generated. GiAPA is dispatched with a few palettes, all having names staring with GiAPA.

This Menu Option allows users to add their own palettes. User define pales names cannot have a name staring with GiAPA.



GiAPA Menu Option 78: GiAPA Installation Parameters

All the first parameters are limits that define how much of the performance data should be stored in the final files being used as input for the various GiAPA reports. The panel below demonstrates the parameters together with descriptive text to make it self-explanatory. The shipped values represent averages that may need adjustment depending on the situation.

It is a basic philosophy of GiAPA only to store information regarding the relatively few jobs that used unusually many resources, since these jobs are likely to contain most of the potential inefficiencies. At the same time they represent jobs where a boost of performance would have a significant impact that it might be worthwhile to examine.

```
GiAPA (c) by                    Installation Parameter Maintenance              8/06/26
iPerformance                                                                    15:41:09
 Shipped      Current
 PARM value   PARM value   Description of installation parameter
                          GiAPA expansion and analysis will keep a separate record for all jobs that
      3            3        - caused >= PARM HotSpot(s) to be generated, or
    120          120        - used >= PARM seconds CPU, or
      1            1        - were type I (interactive) and used >= PARM seconds CPU within any interval.
                          All other jobs are summarized into group totals (*BATCH, *INTERACT, etc.)

                          For jobs kept above, details per interval are only kept for intervals where
    300          300        - jobs with priority <= 25 used PARM millisecs. within the interval, or
    825          825        - the job used PARM milliseconds CPU within the interval, or
   2000         2000        - the job used PARM physical I/Os within the interval.

                          A job name summary record is generated if job name is not QPADEV* and
      3            3        - >= PARM jobs had the same job name, and
     60           60        - these jobs added together used >= PARM seconds CPU time.

                          A user name summary record is generated if
      2            2        - >= PARM jobs had the same user name, and
     60           60        - the jobs for the user used PARM seconds CPU time.

                          The file usage data from HotSpots are collected and reported on file level if
   1000         1000        - the file showed more than PARM I/Os.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  1000000      1000000     Minimum I/O count for including file in File Pfr. Analysis (Menu option 19)

                          ALLOCATION TRAP FEATURE: Send warning message GIA0070 (severity code 60)
999999999    999999999     to QSYSOPR if a job allocates more than PARM pages.

     20           20     GiAPA will retrieve not more than PARM call stack levels for HotSpots (Max 99)
      5            5     GiAPA will retrieve call stacks for max PARM threads at a HotSpot    (Max 10)
    100          100       and only if thread used PARM milliseconds CPU in the last interval
      2            2     Retrieve HotSpot info for threads in Wait state?  1 = *Yes,  2 = *NO

     50           50     Default average CPU milliseconds to use for very small jobs.
                          Not used if Job Accounting is active. Please refer to the explanation given in
                          the GiAPA User manual, section on Menu option 21, selection 9.

              0          CPW per processor in LPAR - used to calculate CPW usage

F3=Exit      F22=Show SrlNbr, LPAR, etc.
```

Please note: The parameter “Retrieve HotSpots for threads in wait” may have quite a significant influence on GiAPA’s HotSpot collection. Threads frequently reach wait state before the HotSpot occurs, which overall makes the call stack less interesting. A value of 1 = *Yes causing the HotSpot collection for multithreaded jobs to also collect call stacks for waiting threads may result in GiAPA using much more resources without providing additional relevant information.

If diagrams showing CPW usage are wanted, the last parameter must contain the CPW value per processor.

Allocation Trap Feature: If activated by the user through changing the default installation parameter value 999999999, warning message GIA0070 severity code 60 is sent to QSYSOPR if a job allocates more work space than the specified number of pages (one page = 4K). If a job continues to exceed the limit message GIA0070 is repeated every 20 minutes.

This option was suggested by an American GiAPA user experiencing performance problems when a job allocated excessive temporary storage – typically caused by SQL code looping while writing to a work file in QTEMP. European GiAPA sites have reported similar situations.

Sort criteria 16 'Max pages used' from the selection panel for the GiAPA Job Performance Summary report (Menu option 15) can be used to find a suitable number of pages to specify as limit for this parameter. Run the report on data for a few busy days and set the limit to e.g. ten times the highest value reported.

```
GiAPA (c) by                Job Performance Summary Sorted by Max. Number of Pages Used
iPerformance                    First / last collection interval:  19-04-08 13:19:15 / 19-0
JobName    UserName   RunTime  Typ Itvs ActualUsr    CPUtime    %  Prio      Logical      P
JobNbr HotSp  RunDate from/to Pool Thrd MaxPages   H:MM:SS.s   CPU rity       I/Os
ALLOCTRAP  KAARE      13:20:15? B     6                15.5  17.2 50      1,734,401
687193     7 19-04-08 13:21:45  02    1    1729867 Max:  2.5  16.7 50        292,227
```

If this example represents a "business as usual" situation for a server, 18.000.000 could be the limit (= the rounded result of 1,729,867 * 10), corresponding to 90 gigabytes.

F22=Show SrlNbr, LPAR, etc. results in the panel shown under menu option 98.

GiAPA Menu Option 81: Manage unexpanded prf. data members

This clean up function is employed to

- consolidate several GiAPA source data members into one member, or
- remove members containing raw and unexpanded GiAPA performance data.

```
GiAPA (c) by                  Manage unexpanded performance data members                    11/11/03
iPerformance                                                        Library: GIAPALIB       11:32:02

   1=Consolidate to one member   4=Delete source member   5=Show record statistics   8=Change expiration date
Option    Member     Perf.data Mb  Member text                      If new exp.date: Julian YYDDD date
   _     PF10171455        19.05   Pfr.data from 111017 at 145538   (ExpDate 11345)         11345
   _     PF09270001        32,70   Pfr.data from 110927 at 000101 (Consolidated)            99365
   1     PF09280001         2,53   Pfr.data from 110928 at 000127   (ExpDate 11322)         11322
   1     PF09280600         0,59   Pfr.data from 110928 at 060026                           99365
   1     PF09280710         5,25   Pfr.data from 110928 at 071030                           99365

F2=Command Line       F3=Exit       (To delete all members older than YYMMDD-date use command GIAPA819)
```

When collecting data, the performance data will be stored in one member per collection.

Option 1=Consolidate: It may be convenient to consolidate data from different collections into one member. In the example shown above, all data from the 28th and 29th of September will be consolidated into the first of these members (i.e., into the member named PF09280001). After the consolidation, the total member size will be shown in the "Pfr.data" column, and the word "Consolidated" will be added to the member text. The other members, which are added to the first member, are deleted when they have been copied.

Option 5=Show record statistics may be used to get an impression of the amount of data being collected. It can be specified for the member currently being used for data collection as a way of checking whether the data collection is active and include the correct data.

```
GiAPA (c) by      Statistics for Collected Performance Data          15-07-22
iPerformance            Input Member Name = PF07020009               16:32:59

        First/Last Interval: 2015-07-02 00:11:00 - 2015-07-02 23:58:30
                          52,583 Blocks of data read
                     36,577,669 Data records processed

Data from Performance Collector API:    HotSpot Data Collected:
          1 Header records                      1 HotSpot header records
      5,715 Intervals header records        9,040 Object name records
        691 Current user records            3,717 Extended name records
    250,863 Job identification records      377,437 HotSpot job records
  4,134,039 Job CPU statistic record      9,465,786 Open file records
 17,352,074 Resource usage records          367,779 File summary records
          1 Run end records                 347,108 Job call stack records
                                             30,329 Thread call stack records
                                          3,361,923 Call stack level records
Data from other sources:                         15 HotSpot trailer records
    598,064 History log records               2,417 SQL Activity records
    247,119 Job accounting records


F2= Command Line     Enter or F3=Exit
```

Option 4=Delete source member and **Option 8=Change expiration date** should be self-explanatory.

GiAPA Menu Option 82: Manage expanded data members

This clean up function is used to remove members containing expanded and analyzed GiAPA result data, to rename a member, or to change the expiration date or member description.

```
GiAPA (c) by                    Manage expanded performance data members                    11/11/03
iPerformance                                                    Library: GIAPALIB          13:30:52

 4=Delete member   7=Rename member   8=Change member expiration date   9=Change member text
                                                                    If rename:    If new exp.date:
Opt  Member     Size in Mb   Text                                   New mbr.name  Julian date YYDDD
 _   EXPDATE         28,3   Test of member text: Description   (ExpDate 12062)  __________       12062
 _   INTERACTIV      47,7   Analysis of interactive workload                     __________       99365


F3=Exit                            (To delete all members older than YYMMDD-date use command GIAPA829)
```

GiAPA Menu Option 83: Delete PEX Definitions and Data

```
                  Delete all PEX data in a lib. (GIAPA830)

Type choices, press Enter.

Delete PEX data in GIAPALIB? . . DLTPEXDATA     N             Y, N
```

During the use of PEX (IBM's Performance Explorer) sizeable data may have been collected in a rather large number of database files. When the results have been analyzed it is recommended to use this function to remove this data.

Please note that the default value N for NO in the second keyword parameter must be changed to Y for YES to allow this clean-up routine to perform the job.

GiAPA Menu Option 84: Delete Trace Job Data

```
GiAPA (c) by      Select Trace Job Data Members to be Deleted
8/04/03
iPerformance
13:23:54

 4=Delete                                        Data library: GIAPALIB

Opt   Member name   Date, time, name, user, and number for job traced
 _    T164631294    2005-03-16 10:54:22 CRTAS4I KAARE 072884
 _    T164631661    2005-03-16 11:01:52 CRTAS4I KAARE 072888
```

This option deletes data that were collected using Option 41.

GiAPA Menu Option 85: Delete File Check Data

Option 85 is used to clean up after having run GiAPA Menu Options 51 and 52. The panel layout is similar to that shown above for Option 84.

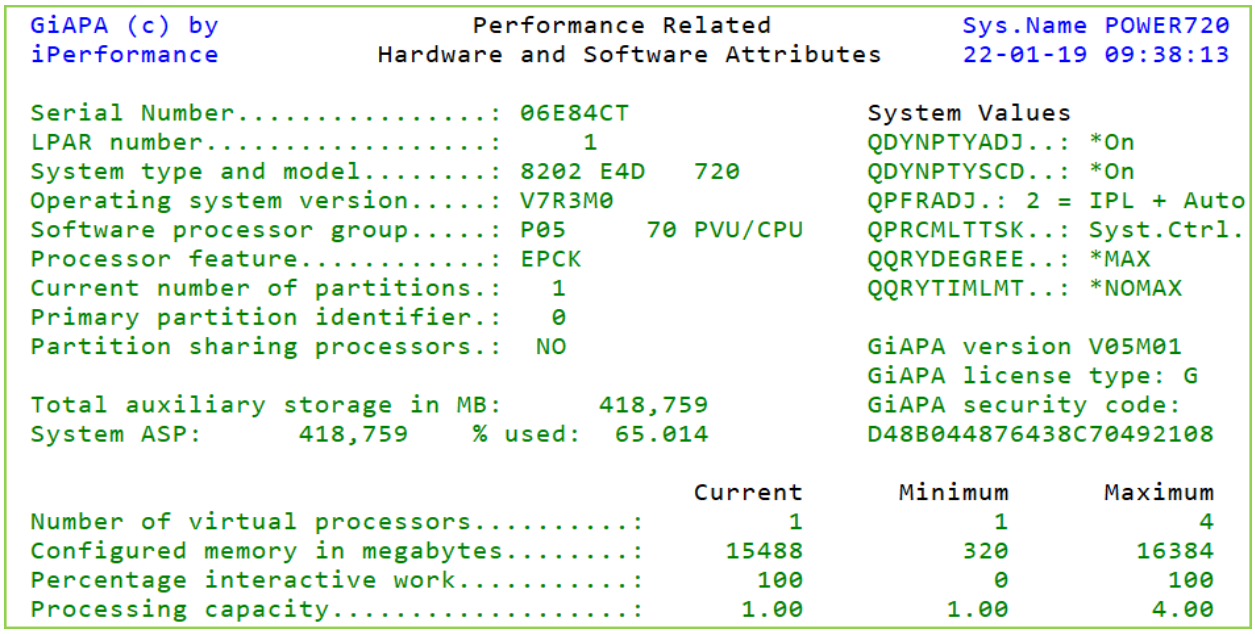
GiAPA Menu Option 87: Delete RUNQRY/WRKQRY tracking Data

This option simply clears the physical file GIAPALIB/GIAPA142P1 that is used to store the data indicating when commands RUNQRY and WRKQRY are used and by whom.

GiAPA Menu Option 89: Check if Authority OK for Data Collection

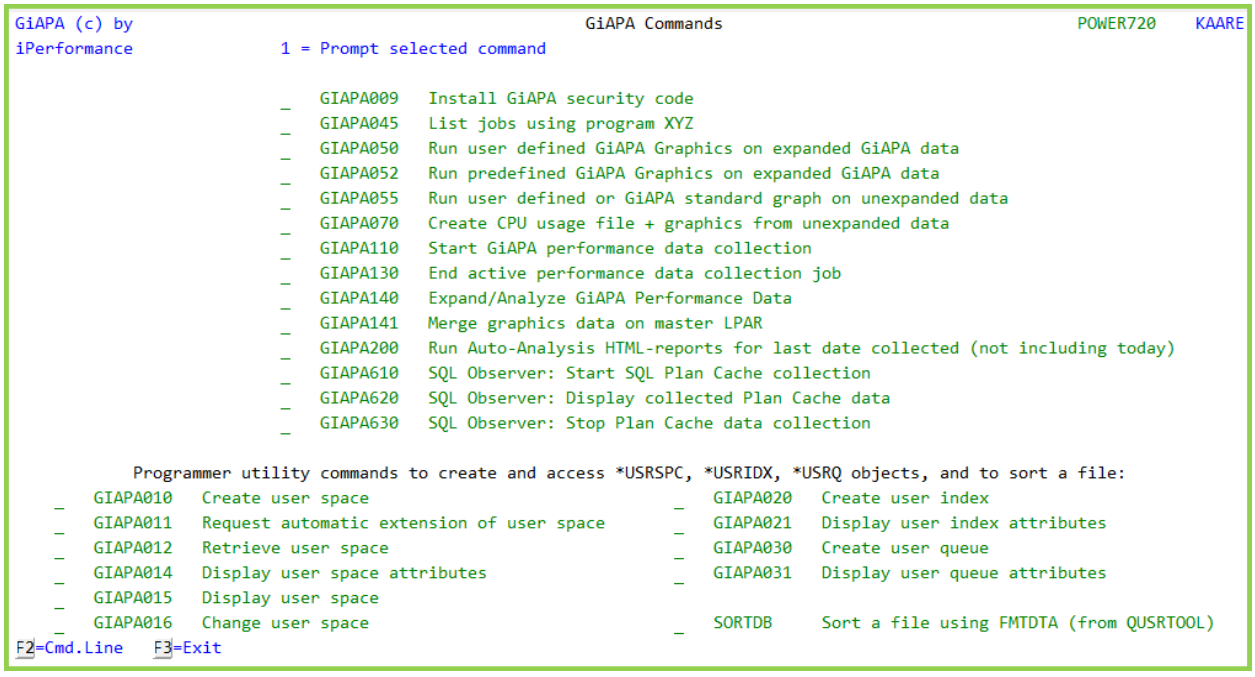
When selecting this option, messages appear to inform whether the user profile currently signed on has sufficient authority to run GiAPA data collection (Menu Option 11 or command GIAPA110).

GiAPA Menu Option 98: Display Server Attributes



GiAPA Menu Option 99: Display GiAPA Menu of GiAPA Commands

Commands are of course mainly used within a CL program or from a command line, but an overview of the most frequently used GiAPA commands is available in a separate menu, which probably does not require further explanation.



Many of the options on the GiAPA Main Menu use a command. The command name is simply a 0 (zero) added to the Menu Option number. Example: To start a GiAPA performance data collection you can use GiAPA Menu Option 11 or alternatively command GIAPA110. This obviously allows you to schedule many GiAPA functions to run in unattended batch.

Commands GiAPA040, GIAPA140, and GIAPA141 are described in the section “Expansion of Collected Performance Data in Scheduled Batch Job“

Command GIAPA045 can be used to find which jobs are using a given program, but only if the jobs involved trigger HotSpots, because only HotSpots collect program names. This implies that the command also can be used to check if the CPU usage of a program generates HotSpots.

```
                    List jobs using program XYZ (GIAPA045)

Type choices, press Enter.

Program to search for  . . . . . PROGRAM    MYPROGRAM  Name
Member with the expanded data  . MEMBERNAME TUESDAY    Name
GiAPA Data library . . . . . . . DATALIB    GIAPALIB   Name
Level of details to list . . . . DETAILS    *ALL       *JOBNAME, *JOBID, JOBIDTHR
```

The report offers four levels of details, depending on the DETAILS keyword. This is an example of the most detailed report, where DETAILS(*ALL) was requested:

```
GiAPA (c) by                List Jobs Running Program OFAW008              Page    1
iPerformance               Based on Input Member Name D20151130
Program    Library    JobName     UserName    JobNbr    Thread        Date       Time
OFAW008    APXCJOMEFO TEFH005.01  RFBEYHLBK   552716    *Initial   2011-04-04  04:09:45
OFAW008    APXCJOMEFO TEFH005.01  RFBEYHLBK   552716    *Initial   2011-04-04  04:10:00
```

Commands to generate diagrams

GIAPA050 Run user defined graphics.
Described after the section on Menu Option 26 “User defined Graphics”.

GIAPA052 Run Predefined GiAPA Graphics

GIAPA055 Run Graph on Unexpanded Data - can be used to generate both
- graphs defined by the user and predefined GiAPA Graphs.

GIA055 offers a quick way of obtaining a graph documenting resource usage in case of unexpected peaks. It runs a partial analysis directly on the raw performance data, sufficient to generate diagrams. The resulting expanded data will automatically be deleted after two days.

```
                    Run predefined GiAPA Graphics (GIAPA052)

Type choices, press Enter.

Input data library . . . . . . . DATALIB     GIAPALIB        Name
Input data member name . . . . . INPUTMBR    *LAST           Name. *LAST
Name of GiAPA Graph to create  . GRAPHNAME   TOPCPUJOBSUSERS GOODMORNING, TOPCPUJOBS
                                                               TOPCPUJOBSUSERS
Overwrite frequency  . . . . . . OVERWRITE   *WEEKLY         *DAILY, *WEEKLY, *MONTHLY
Max records limit for TOPCPU*  . MAXRECORDS  20              2-50
Program to call to send Email  . SNDMAILPGM  MYEMAILPGM      Name
  Library name . . . . . . . . .               GIAPALIB      Name, *LIBL, *CURLIB
Recipient Email-Addr for Graph   RCP          itmgr@company.com
                                                                              +
                                                                              +
                                                                              +
                           + for more values  'myownmail@company.com *CC'     +
                                                                              +
                                                                              +
                                                                              +
```

The intended use of command GIAPA052 is in batch after the daily expansion and analysis of GiAPA data, which most installations run shortly after GiAPA has restarted at midnight.

INPUTMBR: Specifying ***LAST** will automatically use the last expanded member as input.

GRAPHNAME: To date these three user-suggested charts are available:

- **GOODMORNING** generates GiAPA's standard Resource Usage Diagram, also known as the "Good Morning report". It provides an overview per hour of CPU and I/O usage.
- **TOPCPUJOBS** displays the CPU usage for the job names using the most CPU. The number of times the job name was found is shown in front of each job name.
- **TOPCPUJOBSUSERS** displays each job individually, together with the user name running the job and the CPU time used.

OVERWRITE details how long time the chart will be stored. If e.g. *WEEKLY is selected, the two last characters of the graph input data member name will contain an abbreviation of the weekday name, causing it to be overwritten when/if generating the same data one week later.

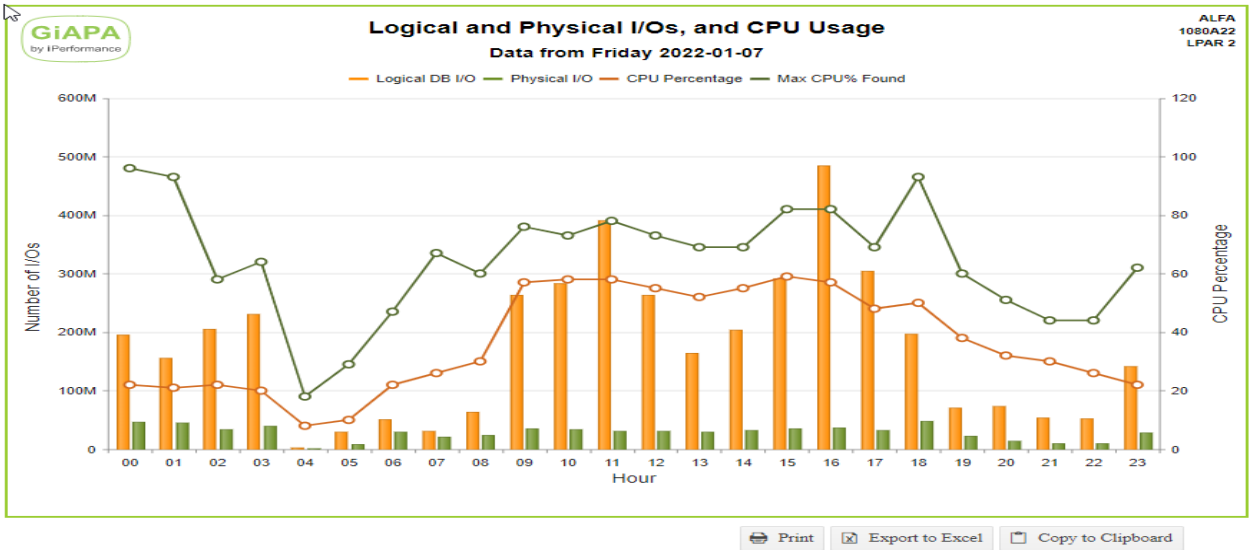
MAXRECORDS specifies the number of values to be shown; this feature is ignored for "GOODMORNING".

SNDMAILPGM defines the name and library for a user exit program to be called if the generated chart must be sent as an Email to one or more recipients, in which case

RECIPIENTS must contain the Email address(es) to which the Email(s) should be sent. A maximum of 20 addresses may be specified. Optionally each Email address may be followed by a blank and one of the entries ***PRI**, ***CC**, or ***BCC** specifying recipient type primary, carbon copy or undisclosed carbon copy. *PRI is default option.

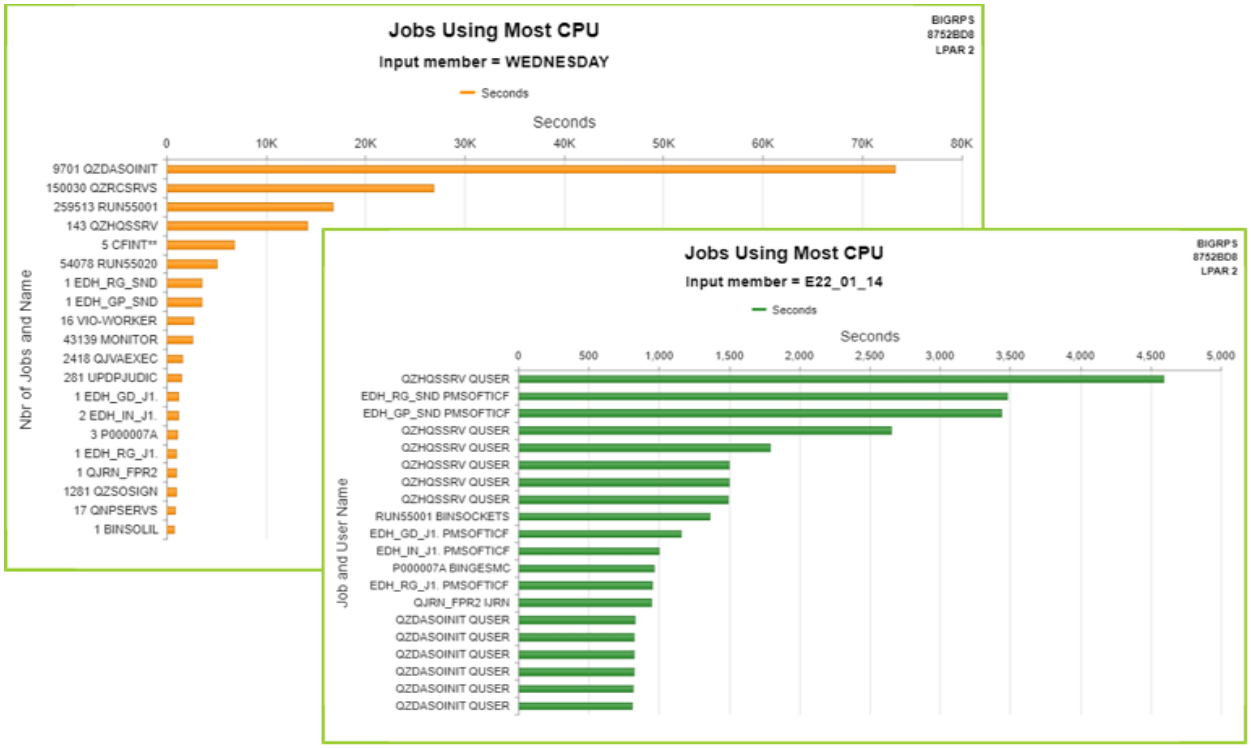
Please turn to **GIAPALIB/GIAPAEXAMP(EMAIL_TEST)** to see an example source code for a User Exit Email program. Comments in the source code give detailed explanation towards the functions of the program and some hints for the set-up of operating system parameters needed to send Emails from the server.

GiAPA's standard Resource Usage Diagram shown below provides an overview per hour of CPU and I/O usage. This report is also known as the "Good Morning report".



The following bar charts illustrate the “TOPCPUJOBS” and the TOPCPUJOBSUSERS.

The histogram on the left shows the number of jobs and job name, whereas the histogram on the right has job and user names as key fields. In both cases MAXRECORD(20) was specified.



Command to create file and diagram with CPU usage details based on unexpanded data:

GIAPA070

Command GiAPA070 is provided in response to a request from customers wanting to obtain an overview of CPU usage per day and per month without the need to run the full GiAPA data expansion and analysis every day.

```
                    Extract CPU usage per hour (GIAPA070)

Type choices, press Enter.

Data library . . . . . . . . . . DATALIB     GIAPALIB__      Name
Date (000000 = yesterday). . . . YYMMDD      000000          Character value
Create graph . . . . . . . . . . CRTGRAP     *NO___          *DAY, *MONTH, *NO
Program to call to send Email  . SNDMAILPGM  MYEMAILPGM      Name
  Library name . . . . . . . . .               GIAPALIB__    Name, *LIBL, *CURLIB
Recipient Email-Addr for Graph   RCP         itmgr@company.com
_____________________________________________________________________________
```

Command GIAPA070 is unique given that it works directly on the “raw” unexpanded performance data collected by GiAPA. Command GIAPA070 only processes data for one day at a time. The command has two functions:

- Each run creates a record in file GIAPA071P1 with detailed information per hour of the CPU usage for the day selected, and

- Optionally a line diagram showing the CPU usage per hour within a day or per day within a month can be requested.

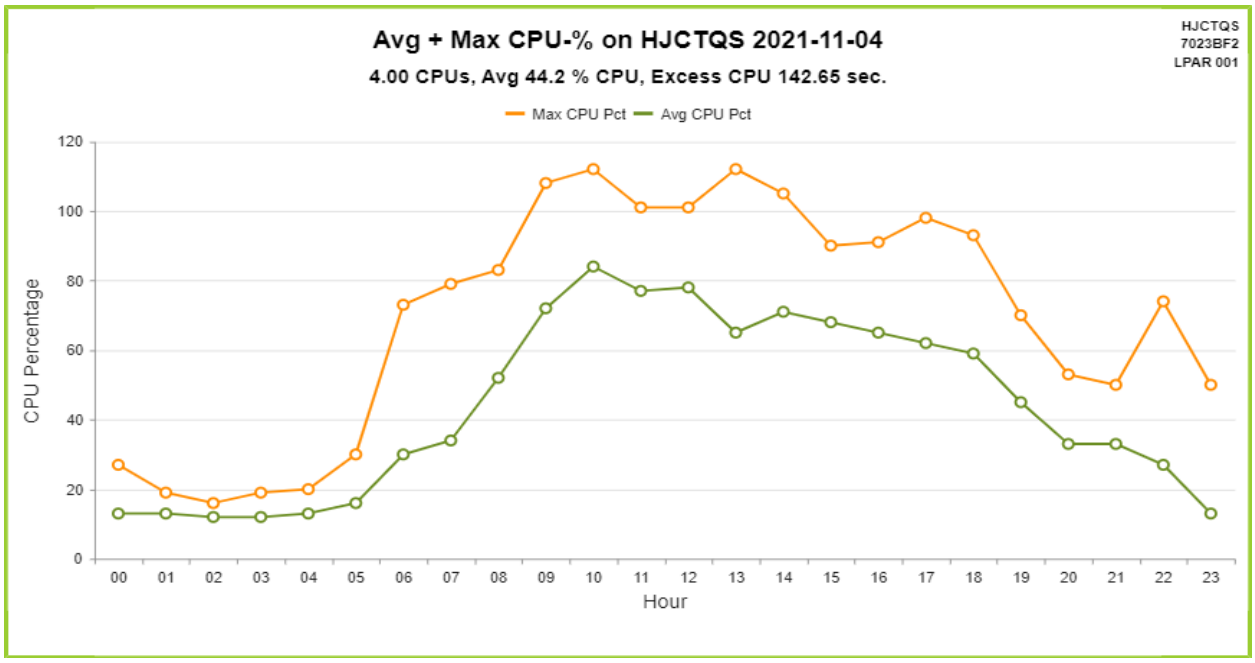
This command is intended for use in batch shortly after GiAPA has restarted at midnight.

The file definition source code of file GIAPA071P1 can be found in GIAPALIB/GIAPA_QDDS. The key field of the file is date in format YYMMDD. The file contains the following information:

• CPU Capacity (number of processors assigned to the LPAR)	These fields exist both as a total for the entire day and for each hour
• Elapsed time (seconds) where GiAPA pfr.data was collected	
• Available CPU seconds (= CPU capacity * Elapsed seconds)	
• Used CPU seconds	
• CPU percentage	
• Maximum CPU % found within a 15 seconds collection interval	
• CPU seconds used above the capacity (in uncapped LPARs)	
• Elapsed seconds of collection intervals having CPU % > 100	
• Average CPU % of intervals exceeding 100 %	Only for entire day

Please observe that to ensure that file GIAPA071P1 contain data for all jobs, job accounting should be active, so GiAPA also receives CPU usage for the very small jobs that only are active a few seconds and therefore not “seen” by the Performance Collector APIs.

If **CRTGRAPH(*DAY)** is specified for the command the following line diagram is generated:



CRTGRAPH(*MONTH) may be used when running command GIAPA070 for the last day of the month. This will cause generation of input data for a similar line diagram as shown above, but the X-axis will reflect the dates of the month instead of hour.

Error Recovery

Symptom: Job GIAPAPFCOL, program GIAPA112 terminates with halt indicator H3 on.

Cause: Error in IBM's Performance Collector APIs

The RESTART keyword in command GIAPA110 (called by GiAPA Menu Option 11) is used to restart GiAPA automatically in case the performance data collection ends due to errors internally in the IBM Performance Collector APIs. This API error will lead to message CPF0A42 or CPF0A43 being returned instead of performance data.

According to the second level text of these messages, the reason for the error can be found in the job log of either job QPMACLCT or job QPMASERV, which are active in subsystem QSYSWRK during performance data collection. In these cases, the job log of job QPMACLCT usually contains the error message CPD0A18, which has the following message text:

Cause...: Collection of job performance data by the performance collector APIs (QPMWKCOL and QPMLPFRD) has stopped due to the failure of internal function &2 (return code &3). Data collection for other resources will continue.
Recovery...: If you attempt to collect job data again and it fails, then end all applications which use the performance collector APIs. Verify that the performance collector has ended by observing that jobs QPMASERV and QPMACLCT end. Then start the applications again to collect data. The performance collector APIs use collection services to collect data. If failures continue, make sure that collection services are configured properly.

For R610 of the operating system IBM has issued PTF SI34648 – please refer to this link for more information:
http://www-01.ibm.com/support/docview.wss?uid=nas2fdd5f381999c77e28625756300421d19

Symptom: Job GIAPAPFCOL fails to properly start collection (member in file GIAPA112P1 is empty).

Cause: Performance collector APIs cannot work because QPRFCOLDTA *USRSPC and/or QPRFCOLDTA *DTAQ in library QUSRSYS is missing or damaged.

This occurs when job QPMACLCT in QSYSWRK receives the message CPD0A18 in the job log within a minute from starting data collection.

Check for the presence of the two objects QUSRSYS/QPRFCOLDTA type *DTAQ and QUSRSYS/QPFRCOLDTA type *USRSPC. If they are present, copy them to QTEMP – if a copy operation fails, the object is probably damaged and should be deleted.

If one (or both) of the objects is missing or damaged, delete both objects from QUSRSYS and call program QSYS/QYPSCOLDTA, which will recreate the two objects in QUSRSYS.

How Is GiAPA Installed and Updated?

The software can be downloaded from www.giapa.com . After downloading the software, the downloaded file must be unzipped on a PC using e.g. WinZip. The password needed to open the zipped file can be obtained from **i**Performance ApS or from a GiAPA distributor.

The following different downloads are available:

- **New installation:** The complete GiAPA software product is installed.
- **Update** to new version: Existing data is kept; data bases will be converted if needed.

The installed version number can be seen on the bottom line of the GiAPA Main Menu, e.g. V06M001F. The current number is displayed on the front page of www.giapa.com .

Authority needed: The IBM performance collector APIs QPMLPFRD and QPMWKCOL used by GiAPA are shipped with *PUBLIC authority *EXCLUDE. Therefore, installation or update of GiAPA must be made by a user profile having QSECOFR authority, and with the system value QALWOBJRST allowing a restore of programs using adopted authority. Alternatively, GiAPA data collection must run under a user profile having authority to use these APIs, as shown in the CL program source code GIAPAUSER in GIAPALIB/GIAPAEXAMP.

Remember that if FTP is used to transfer the downloaded save file to the server, you must use FTP command **bin** to run in binary mode, and the receiving save file should be created on the server before you start uploading from the PC.

Update GiAPA to a new version / modification:

Terminate GiAPA data collection before starting the update. To be on the safe side, you may want to back GIAPALIB up first, but it should be superfluous, since no files containing collected data are replaced. Update requires use of two commands: RSTOBJ and GIAPAINST.

The RSTOBJ normally only takes a few seconds. After that, command GIAPAINST will take care of any additional modifications required, also including change of data file layout (e.g. extending a field length) in which case existing data will be copied to the new format.

```
RSTOBJ  OBJ(*ALL) SAVLIB(GIAPALIB) DEV(*SAVF) +
     SAVF(savefilename) MBROPT(*ALL) ALWOBJDIF(*ALL)

GIAPALIB/GIAPAINST
```

If GiAPA data is kept in another data library than GIAPALIB, command GIAPAINST must also be used for the data library. Example: `GIAPALIB/GIAPAINST DATALIB(MYDATALIB)`

New installation: (Do not install GIAPALIB in IASP.)

Install GiAPA by restoring GIAPALIB. When the unzipped save file containing GIAPALIB has been transferred to the iSeries using e.g. FTP, run the following command:

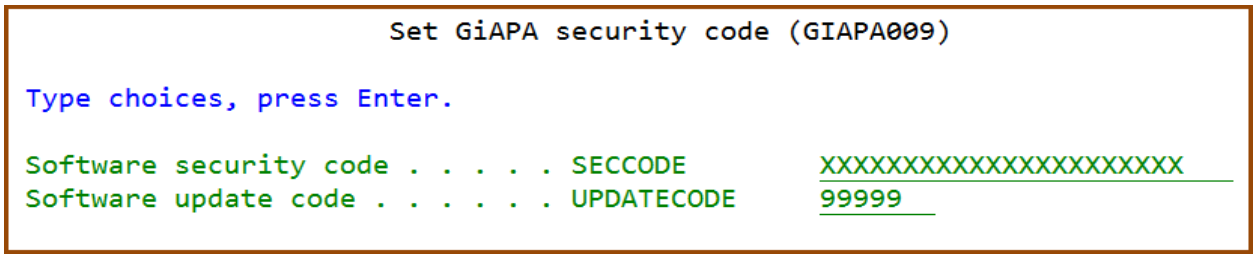
```
RSTLIB  SAVLIB(GIAPALIB) DEV(*SAVF) SAVF(savefilename)
```

Important: Modification to any automatic backup routine for GIAPALIB

If GIAPALIB is backed up using “save while active” during performance data collection, error message “Cannot allocate object” should be avoided through specifying **OMITOBJ((GIAPALIB/*ALL *USRSPC) (GIAPALIB/*ALL *USRQ) (GIAPALIB/GIAPA115* *USRIDX))**.

Command GIAPA009: Install GiAPA Software Security Code

Before GiAPA can analyze collected performance data, a valid software security code must be installed using CL-command GIAPALIB/GIAPA009. GiAPA performance data can always be collected and exported – these and a few other functions do not require a valid security code.



SECCODE: The security code must always be specified.

UPDATECODE: The update code is not always used. It will be supplied when needed. If only the security code is supplied, the update code should be left unchanged.

How can GiAPA be removed

To uninstall GiAPA from the server simply use CL-command **DLTLIB GIAPALIB**.

Displaying GiAPA results in html format

GiAPA generates html output in the /GIAPA folder of the server IFS when

a) a chart / diagram / graph is requested, or
b) automatic performance analysis (Menu option 20) is requested in html format.

The /GIAPA folder is placed directly under the root of the IFS.

When an interactive job generates GiAPA output in html format, GiAPA retrieves the IP-address of the server and runs the CL-command “Start PC command” to display the result. Example:

```
STRPCCMD PCCMD('start file://192.168.0.71/GIAPA/GIAPA15105.html')
```

The generated result address – in this example file://192.168.0.71/GIAPA/GIAPA15105.html – is also written to the server job log, from where it may be copied into a browser address line to prove if it works OK.

From PCs running Microsoft Windows

For Windows PCs used as workstations and connected to the server through a company network the IP-address alone may not be sufficient for reaching the server IFS. In such cases you will need to

1. define a SHARE to the /GIAPA folder below the IFS Root, as described in this link: https://www.ibm.com/docs/en/i/7.3?topic=i-creating-file-share
2. define a drive letter mapping directly to the /GIAPA folder in the IFS, as described here: https://support.microsoft.com/en-us/windows/map-a-network-drive-in-windows-29ce55d1-34e3-a7e2-4801-131475f9557d
3. define the selected drive letter to GiAPA using CL-command GIAPA902. If letter ‘Z’ is selected as drive letter, the CL-command would be `GIAPALIB/GIAPA902 MAPLETTER(Z)`

This would cause GIAPA to replace the IP-address and folder name with the letter specified, thus enabling the STRPCCMD to cause the result to be displayed immediately. Example:

```
STRPCCMD PCCMD('start file:///Z:/GIAPA15105.html')
```

From Apple computers (MacBook, iMAC, etc.)

The STRPCCMD command described above is unfortunately not implemented for Apple computers. Therefore, if you use an Apple computer, please follow these guidelines to get the HTML results:

1. Start IBM’s Access Client connection to the Power i Server.
2. Open “Finder”.
3. In the Finder window, press Command-K, which will open a new window
   “Create connection to Server”
4. In the input field at the top, run the following statement:
   Smb://xxx.xxx.x.xxx/GIAPA (where xxx.xxx.x.xxx is the IP address of the server)
5. When connected, a window containing the generated output is displayed.
6. Double click on the html file name (probably the latest created) to see the result.

Index

