

White Paper

Collecting DB2 Metrics in SMF

Fabio Massimo Ottaviani - EPV Technologies



1 Introduction

The DB2 Instrumentation Facility Component (IFC), provides a powerful trace facility that can be used to record DB2 data and events. Virtually everything can be measured by activating appropriate traces and classes.

As you can imagine these metrics are extremely useful for controlling and tuning DB2 subsystems and application performance. Unfortunately, the volume of data DB2 traces collect can be quite large. The overhead to produce this data can impact system performance, while the amount of data to manage and process can require a lot of additional system resources. This is the reason why many sites collect just a small part of the DB2 metrics, (or no DB2 measurements at all), in their SMF data.

In this paper, after an introduction to the DB2 measurement infrastructure, we'll focus on DB2 statistics and accounting information, showing examples of how to use SMF collected data to tune DB2 subsystems and applications.

We'll also discuss some possibilities to reduce:

- The overhead of running DB2 traces;
- The amount of SMF data produced;
- The amount of system resources needed to process SMF data.

If not explicitly stated, everything in this document refers to DB2 V9.1.

2 DB2 Traces

The DB2 instrumentation facility component (IFC) provides six trace types:

- Statistics, which collects information related to the DB2 subsystem; it should be used to check anomalies, control and tune the subsystem components' performance;
- Accounting, which collects information related to application programs; it should be used to check anomalies, control and tune performance, perform accounting of DB2 applications;

- Audit, which collects information about DB2 security controls; it should be used to ensure that data access is permitted only for authorized purposes;
- Performance, which collects a lot of detailed information about any type of DB2 events; it should be used to pinpoint the causes of performance problems only when really needed, (when you have exhausted all the other possibilities);
- Monitor, which collects information useful for performance monitoring; most of the information is also provided by other types of DB2 traces, (e.g. Statistics and Accounting); it should be used to feed performance monitors such as BMC Mainview or IBM Omegamon;
- Global, which collects information regarding entries and exits from internal DB2 modules, as well as other information about DB2 internals; it should be used only upon specific IBM request.

To allow granular control, each trace is broken down into classes. The type and amount of information produced depends on the activated classes. Generally speaking, the activation of a class causes the production of one or more Instrumentation Facility Component Identifier (IFCID) records.

The description of the START TRACE command indicates which IFCIDs are activated for the different types of trace and the classes within those trace types¹. There is no official DB2 manual providing details on what information each IFCID returns².

Trace output can be written to:

- GTF (Generalized Trace Facility); a z/OS component used to store large volumes of trace data
- SMF (System Management Facility); the standard facility used by z/OS to accumulate information and measurements.
- SRV; an exit to a user-written routine (it is used primarily by IBM support personnel);
- OPn (where n is a value from 1 to 8); a specific output buffer area normally used by monitors;
- OPX; a generic output buffer normally used by monitors; when it is used, DB2 assigns the next available OPn buffer (OP1 to OP8);
- RES; a wraparound table residing in memory.

The following table summarizes the default and possible destinations of each trace type:

Trace type	Default destination	Possible destinations
Statistics	SMF	Any except RES
Accounting	SMF	Any except RES
Audit	SMF	Any except RES
Performance	CTX	Any except RES
Monitor	OPX	Any except RES
Global	RES	Any

Figure 1

¹ See the - START TRACE chapter of the DB2 Command Reference manual.

² The best source of information about IFCIDs is the DSNWMSG member provided in the ADSNIVPD DB2 library.

3 Statistics and accounting IFCIDs mapping to SMF records

While the number of IFCIDs is very high, their mapping to SMF records is pretty simple: most of the statistics trace IFCIDs are collected in SMF type 100, accounting trace IFCIDs in SMF type 101 and all the other IFCIDs in SMF type 102.


The following table provides more detailed information about the most useful statistics and accounting IFCIDs, showing the class to activate in order to produce them, the destination SMF type/subtype and a short description of their content.

IFCID	TRACE	CLASS	SMF TYPE	SMF SUBTYPE	DESCRIPTION
001	STATISTICS	1	100	0	SYSTEM SERVICES STATISTICS
002	STATISTICS	1	100	1	DATABASE STATISTICS
202	STATISTICS	1	100	2	BUFFER POOL PARAMETERS
230	STATISTICS	5	100	3	DATA SHARING GLOBAL STATISTICS
225	STATISTICS	1	100	4	STORAGE STATISTICS
172 196 105	STATISTICS	3	102		DEADLOCK AND TIMEOUT STATISTICS DB TS MAPPING
003	ACCOUNTING	1	101	0	PLAN ACCOUNTING
003	ACCOUNTING	2	101	0	PLAN IN DB2 TIME
003	ACCOUNTING	3	101	0	PLAN WAIT TIME
239	ACCOUNTING	7	101	1	PACKAGE ACCOUNTING
239	ACCOUNTING	8	101	1	PACKAGE WAIT TIME
239	ACCOUNTING	10	101	1	PACKAGE ACCOUNTING DETAILS

Figure 2

There are some important notes to remember about the statistics trace:

- The statistics trace uses class 1 as the default.
- The statistics trace is written on an interval basis. Up to DB2 9.1 you can control the exact time that statistics traces are taken by setting the STATIME parameter. Starting from DB2 10 this parameter has no effect on the SMF 100 IFCIDs: they will be produced every minute.
- Most of the information provided in IFCID 1 and 2 are accumulated since the trace start. They have to be de-accumulated to provide meaningful values for each measured time interval.
- With DB2 versions older than 8.1, the activation of class 6 is needed to produce IFCID 225.
- Even after some recently released IBM PTF, the standard SMF dump program (IFASMFDP) doesn't support the SMF 100 subtype. So if you try, for example, to dump only SMF 100 subtype 1 by using the TYPE(100(1)) parameter you will not get any records.



There are also some important points to remember about the accounting trace:

- The accounting trace uses class 1 as the default.
- The accounting trace begins collecting information when a thread is successfully allocated to DB2. A complete record is written when the thread terminates, is reused, or when the authorization ID changes³.
- When you start additional classes, such as class 2 and 3, many additional trace points are activated. Every occurrence of these events is traced internally by DB2 trace, but these traces are not written to any external destination. The accounting facility uses class 2 and 3 traces to produce the additional information that appears in IFCID 003.
- Class 1 is always required to write IFCID 003.
- To write IFCID 239 (accounting for packages and DBRMs), accounting trace class 7 or 8 must be active. Also in this case activating more classes, such as 7, 8 and 10, causes the activation of additional trace points and the production of additional information that appears in IFCID 239.
- Class 10 was introduced in DB2 8.1; prior to this all the information about packages, (except the delays produced by class 8), was produced by activating just class 7. From DB2 8.1 onwards activating Class 7 will provide only information about the CPU consumed by packages; to get more detailed information, (such as getpages and executed SQL), you have to activate the Class 10 too.
- Accounting class 1 determines the destination of both the 003 and 239 IFCIDs.

4 Using statistics trace data

The DB2 statistics trace produces many metrics which can be extremely useful to control and tune DB2 subsystems. While discussing each one of these metrics is far beyond the scope of this paper, in this chapter we'll discuss some of the most interesting areas covered by DB2 statistics trace; they are:

- Critical events counts;
- System address spaces overhead;
- DBM1 virtual/real storage usage;
- Lock activity;
- Buffer pool activity.

All the metrics discussed in this chapter, unless otherwise stated, are collected in SMF 100 (IFCID 001, 002, 202, 225 and 230).

All the words in **bold** are SMF 100 record fields.

4.1 Critical events counts

There are a number of counters which should always show zero values in a "perfect" DB2 subsystem. They are reported in the following table.

³ While this is the general rule, a certain number of exceptions exists such as, for example, the use of the TXIDS0=NO CICS parameter which eliminates the sign-on process when only the transaction code changes.

Logging	<p>QJSTBPAC, number of waits for an output log buffer to be paged in. When DB2 wanted to write a log record and log buffer was not available because it was paged out, DB2 and application must wait for the log buffer to be paged in. A nonzero value could indicate that the output log buffer size is too large or there is insufficient real storage to back up it.</p> <p>QJSTWTB, number of waits because of an unavailable output log buffer. When DB2 wanted to write a log record and log buffer was not available, DB2 and the application must wait for an available log buffer. A nonzero value could indicate that the active log device is not able to sustain the active log buffer activity.</p>
Threads	<p>QDSTQDBT, number of times that a data base access thread (DBAT) was queued due to having reached the maximum for concurrent remote threads.</p> <p>Q3STCTHW, number of times that a create thread request (not including database access threads) was queued due to having reached the maximum for concurrent threads.</p> <p>Pools</p>
Pools	<p>QISEFAIL, number of failures because the EDM pool was full and the LRU algorithm was not able to free space for new CTs and PTs. It means that the pool is too small. In this situation an SQL error code is sent to the application program.</p> <p>QISESFAL, number of failures because the EDM above the 2 GB "bar" pool was full and the LRU algorithm was not able to free space for new CTs and PTs. It means that the pool is too small. In this situation an SQL error code is sent to the application program.</p> <p>QISEDFAI, number of failures because the DBD pool was full and the LRU algorithm was not able to free space for a new DBD. It means that the pool is too small. In this situation an SQL error code is sent to the application program.</p> <p>QISECFAL, number of failures because the dynamic statement pool was full and the LRU algorithm was not able to free space for a new SQL PREPARE. It means that the pool is too small. In this situation an SQL error code is sent to the application program.</p> <p>QISEKFAL, number of failures because the EDM SKELETON pool was full and the LRU algorithm was not able to free space for new SKCTs and SKPTs. It means that the pool is too small. In this situation an SQL error code is sent to the application program.</p> <p>QXNSMIAP, number of times DB2 detected no storage was available to hold a list of RIDs during a given RID list process for list prefetch, including single index list prefetch, access via multiple indexes and hybrid joins. This is an undesirable situation as the portion of the query which would use the index falls back to table scan processing. It normally means that the RID pool is too small.</p>

Buffer Pools	<p>QBSTDMC, number of times the Data Manager Threshold (DMTH) was reached. DMTH is set at 95%. It controls the percentage of unavailable pages, (in-use or updated), in the Buffer Pool. This threshold is checked before a page is read or updated. If the threshold is exceeded, DB2 accesses the page in the buffer pool once for each row that is retrieved or updated in that page. Reaching this threshold has a significant effect on processor usage and application performance, so it should be absolutely avoided. It is a “fixed” threshold which means it cannot be changed by the user. Possible solutions are: increasing the size of the buffer pool or reducing the DWQT and VDWQT variable thresholds.</p> <p>QBSTSPD, number of times the Sequential Prefetch Threshold (SPTH) was reached. SPTH is set at 90%. It controls the percentage of unavailable pages, (in-use or updated), in the Buffer Pool. This threshold is checked:</p> <ul style="list-style-type: none"> • Before scheduling a prefetch operation; if the threshold has been exceeded, the prefetch is not scheduled. • During buffer allocation for an already-scheduled prefetch operation; if the threshold has been exceeded, the prefetch is cancelled. <p>Reaching this threshold has a significant effect on processor usage and application performance for all activities using sequential prefetch, as in the case of large table scans, so it should be absolutely avoided. It is a “fixed” threshold which means it cannot be changed by the user. Possible solutions are: increasing the size of the buffer pool or reducing the DWQT and VDWQT variable thresholds.</p> <p>QBSTREE, number of times that a sequential prefetch was not performed because the maximum number of concurrent sequential prefetches was reached. In that case normal reads are done. The maximum number of concurrent sequential prefetches cannot be changed by the user. It could be an indication of contention in the I/O subsystem that is slowing down prefetch operations and maintaining “engines” busy more than needed. Possible solutions are: tuning the I/O subsystem to improve performance or tuning the applications to reduce sequential prefetch requests.</p>
Group Buffer Pools	<p>QBGLWLF, number of times that a coupling facility write request could not complete due to a lack of coupling facility data entries in the Group Buffer Pool structure. It could be an indication that the castout process is not quick enough or simply that the number of data entries is too small.</p>

Figure 3

Figure 4 shows an example of many log buffer wait events at a site. All applications performing inserts and updates were slowed down when the log buffers were exhausted.

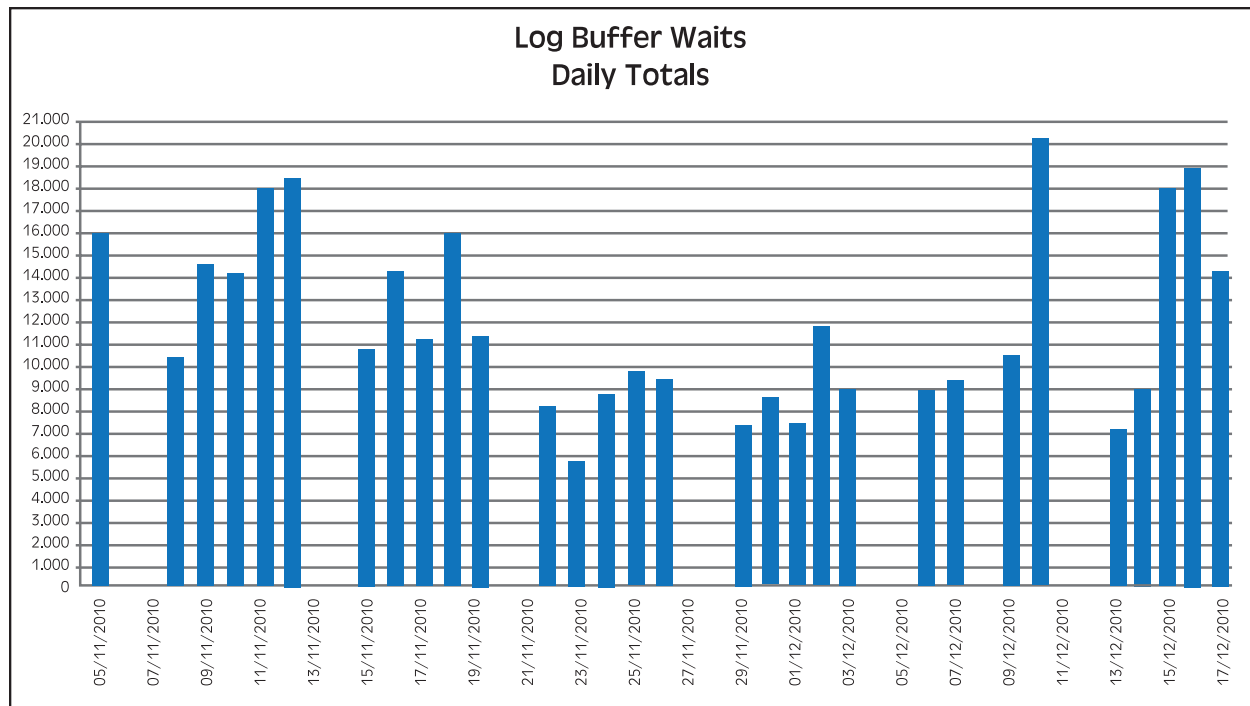


Figure 4

Checking the DSNZPARM settings, we discovered that the number of log buffers, (OUTBUFF parameter), was 4,000 which is the current default and it's absolutely not adequate for that DB2 workload.

We increased OUTBUFF to 100,000 (the maximum allowed is 400,000) and the problem was solved.

4.2 System address spaces overhead

Each DB2 subsystem always includes three system address spaces:

- Master (MSTR), providing overall control functions such as logging and backout;
- Database Manager (DBM1), providing database related functions such as buffer pools and EDM pool management;
- Internal Resource Lock Manager (IRLM), providing locking support.

In DB2 statistics a section for each DB2 system address space is provided. By checking the **QWSAPROC** field you can identify each DB2 system address space while **QWSAEJST** and **QWSASRBT** provide TCB and SRB consumptions.

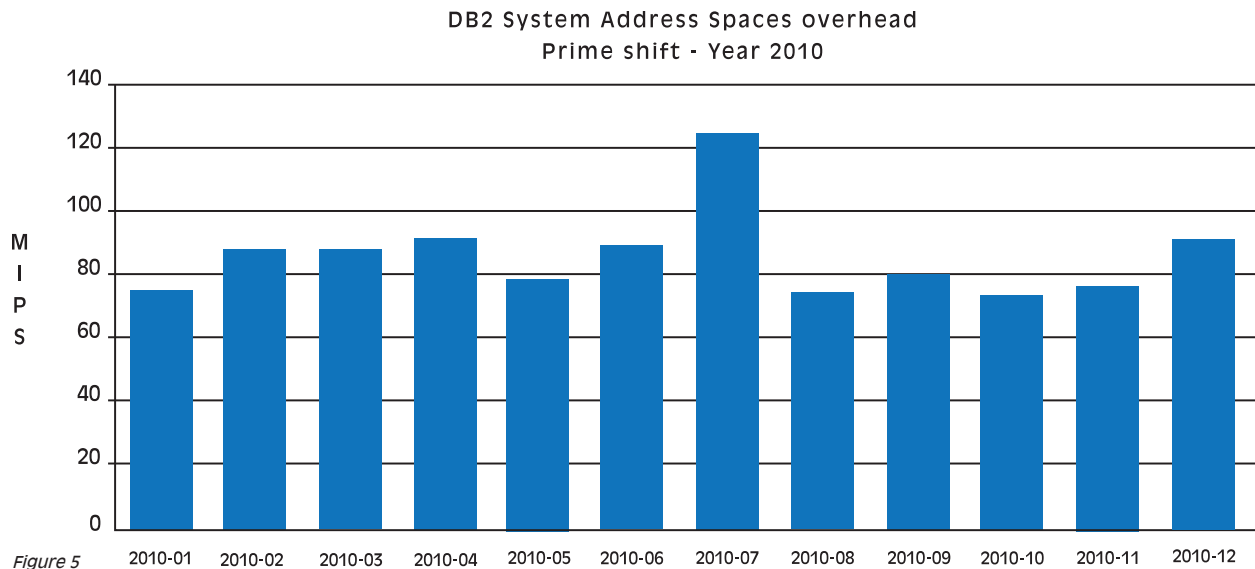


Figure 5

Figure 5 shows the monthly trend of the CPU overhead of DB2 system address spaces in the prime shift.

The peak in July was caused by the migration to z/OS 1.11, which introduced a performance problem when accessing PDSE with many members. At this site some DB2 applications unload many thousands of LOBs to PDSE every day, so DB2 overhead increased by about 50% after the migration. The problem was solved by applying the OA34214 correction.

Many other metrics are available to control the CPU overhead of DB2 system address spaces. More details can be found in the "Analyzing DB2 overhead" white paper.

4.3 DBM1 virtual/real storage usage

For many years the DBM1 available virtual storage below the 2 GB "bar" has been a "hot" issue⁴. A specific ICID (IFCID 225), is therefore written by the statistics trace, providing a lot of detailed information about virtual storage usage⁵.

The most relevant information to track is:

- Available virtual storage (**QW0225AV**);
- Used virtual storage; it is calculated as: virtual storage below 2 GB, (**QW0225RG**), minus available virtual storage;
- Virtual storage cushion

The "cushion" is the amount of virtual storage which has to be available to allow DB2 to run without starting the system contraction process to free storage segments.

⁴ Starting with DB2 10, the attention will move to real storage.

⁵ IFCID 225 is currently written in SMF 100; in versions older than 8.1 it is written in SMF 102.

The “cushion” is composed of:

- Storage needed for “must complete” operations⁶ (QW0225CR);
- Storage needed for datasets open/close (QW0225MV);
- Storage needed to warn before starting contraction (QW0225SO).

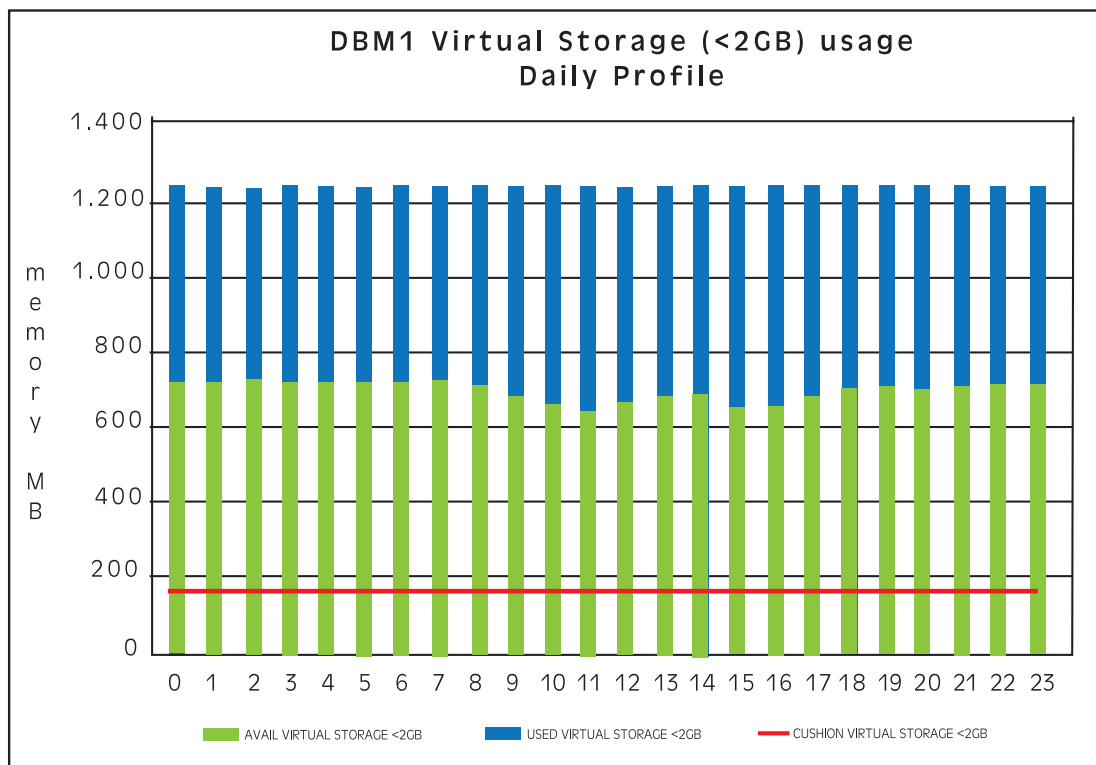


Figure 6

The red line in Figure 6 represents the “cushion” limit. If the green bar (available storage) falls below that line, DB2 will start the storage contraction process.

Most of DB2 DBM1 address space virtual storage is also resident in real storage, so DBM1 is normally one of the major real storage exploiters.

The most relevant information to track DBM1 real storage usage is:

- Used real storage (QW0225RL);
- Used auxiliary storage (QW0225AX).

While virtual storage can be analyzed on a single DB2 subsystem basis, real storage should also be tracked taking into account all the DB2 subsystems running in the same system. The following graph shows the monthly profile of DBM1 real storage usage, in the prime shift. Six DB2 subsystems run in the same system. Their global real storage footprint is about 9 GB.

⁶ Such as aborts and commits.

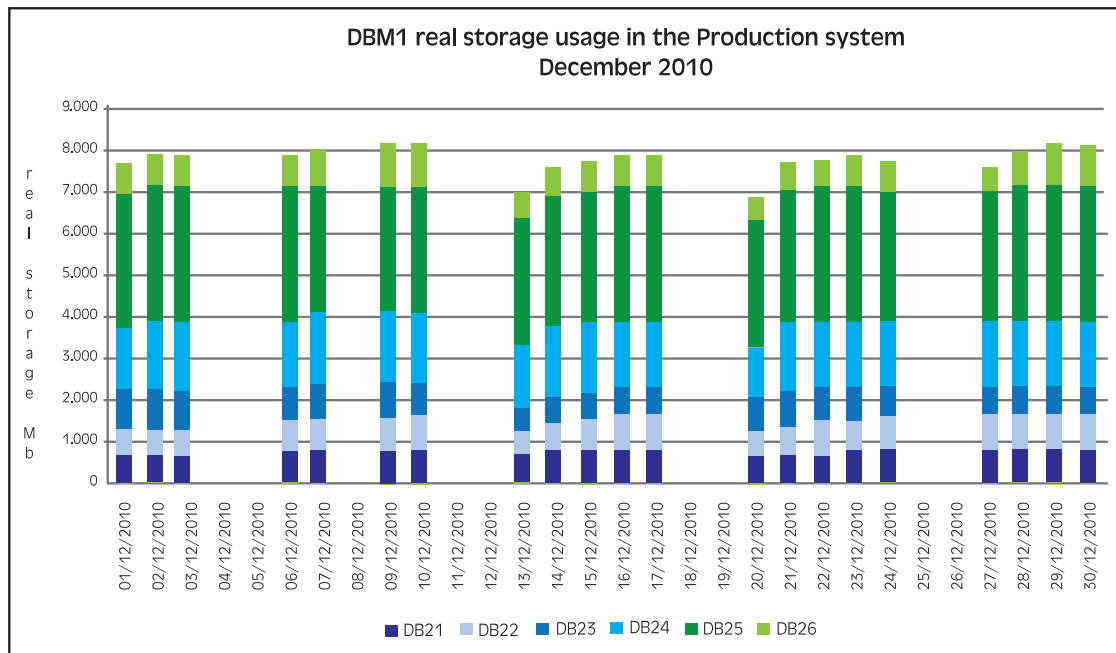


Figure 7

4.4 Lock activity

This is a very important area in DB2 Performance analysis. The most interesting metrics are:

- Number of lock requests (QTXALOCK);
- Number of lock suspensions (QTXASLOC);
- Number of lock escalations exclusive (QTXALEX);
- Number of lock escalations shared (QTXALES);
- Number of deadlocks (QTXADEA);
- Number of timeouts (QTXATIM).

Additional metrics specific for data sharing activity (such as P/L Locks propagated to XES, Lock and Unlock requests for P-Lock, etc.) are also provided.

Figure 8 shows the daily profile of timeouts, deadlocks and lock escalations. A peak of 27 timeouts in an hour occurred between 15 and 16 (15 in the graph).

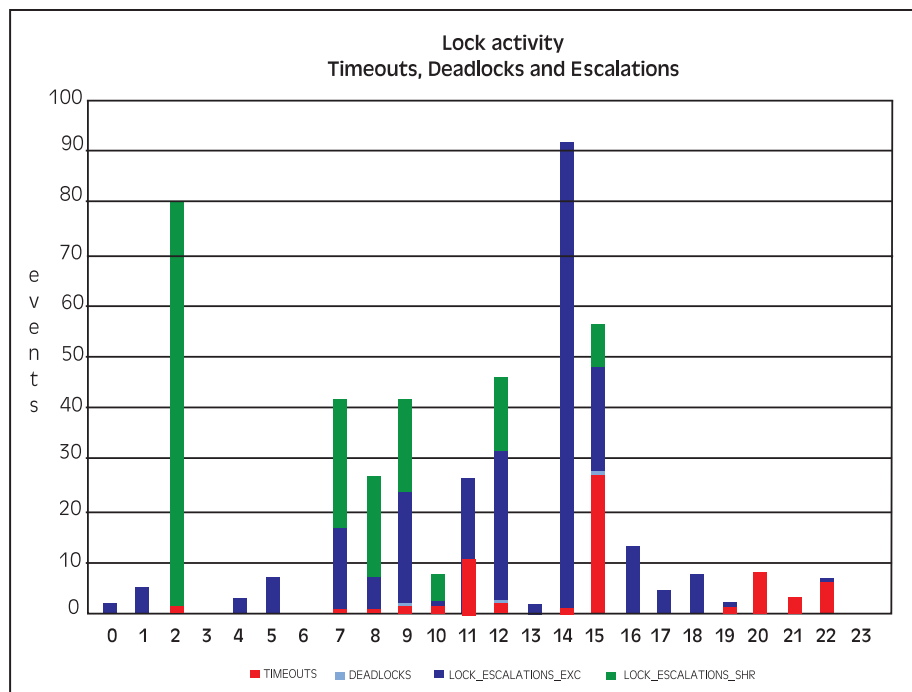


Figure 8

The relevant questions are:

1. Which unit of work was timed out ?
2. Which resource was it waiting for ?
3. Who was blocking that resource ?

To answer these questions you need more detailed information. You can get it by starting statistic trace Class 3. It will produce many other IFCIDs which will be recorded in SMF 102. The most relevant for locking analysis are:

- IFCID 172⁷ and 196, they will provide information about the units of work involved in deadlocks and timeouts, showing the resource⁸ they were contending, the attributes of their requests, etc.
- IFCID 337, it will provide information about package/instruction starting the escalation process, the involved resources, the final lock state, etc.

We will see in the next chapter that by using SMF 101 it's possible to answer at least question 1.

⁷ IFCID 172 records are written every time that DB2 takes action to resolve a deadlock. IFCID 002 contains all the deadlocks including the ones broken without intervention by DB2 (e.g. when a process times out)

⁸ Resources are identified by using DBID (database id) and OBID (object id; table space or index space); you have to translate them to meaningful names by using the IFCID 105 content. IFCID 105 is produced by statistics trace Class 1.

4.5 Buffer pool activity

DB2 buffer pools are a key resource for ensuring good applications performance. Their goal is to maintain the most accessed pages in memory to avoid synchronous I/O operation. This is increasingly important because of the difference in speed between processors and disks, which widens with each new generation of processor.

4.5.1 Buffer pool read efficiency

A read I/O can be avoided if the requested page is still in the buffer pools. The most important metrics to measure buffer pools' read efficiency are:

- Total getpages (QBSTGET);
- Pages read synchronously (QBSTRIO);
- Pages read asynchronously using sequential prefetch (QBSTSP);
- Pages read asynchronously using list prefetch (QBSTLPP);
- Pages read asynchronously using dynamic prefetch (QBSTDPP);
- Asynchronous read I/Os using sequential prefetch (QBSTSIO);
- Asynchronous read I/Os using list prefetch (QBSTLIO);
- Asynchronous read I/Os using dynamic prefetch (QBSTDIO).

Using them you can easily calculate:

- Buffer pool hit ratio, it is calculated as: $(a - (b + c + d + e)) / a$;
- Read rate, it is calculated as: $(b + f + g + h) / (\text{number of seconds in the measurement interval})$.

Buffer pool hit ratio is one of the most commonly used indexes to evaluate buffer pool performance. It is useful but it can be misleading at times.

A 95% hit looks very good, but what if that 5% miss translates to 10,000 read I/Os per second ?

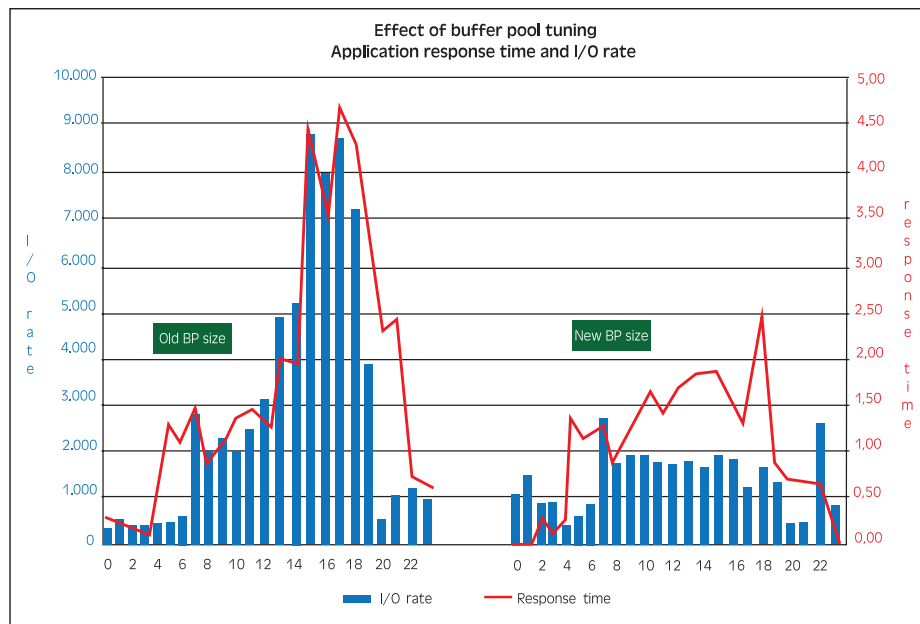


Figure 9



This is the reason why the read rate is even more important. I/O operations are very bad for performance and they also increase the CPU consumptions of both applications (synchronous I/Os) and DBM1 address space (asynchronous I/Os).

The graph in Figure 9 shows the effect of buffer pool tuning in a real life situation. A very important web application suffered with performance problems in the peak days. By looking at the SMF 100 data, the cause was identified in an excessive read I/O rate concentrated on a specific buffer pool. We decided to double the buffer pool size. The result was that application response time returned within the established SLO and we saved more than 6,000 I/O per second in the peak hours⁹.

It's worth noting that, before the change, the buffer pool hit ratio was about 93%.

4.5.2 Buffer pool write efficiency

A write I/O can be avoided if the requested page can be updated when still in the buffer pools. Write I/Os can also be reduced by bunching more modified pages together and writing them in just one operation.

The most important metrics to measure buffer pools' write efficiency are:

- i) Buffer page updates (**QBSTSWS**);
- j) Pages written (**QBSTPWS**);
- k) Synchronous write I/Os (**QBSTIMW**);
- l) Asynchronous write I/Os using list prefetch (**QBSTWIO**).

Using them you can easily calculate:

- Page updates per page written; it is calculated as: i / j ;
- Pages written per write I/O; it is calculated as: $j / (k + l)$.

5 Using accounting trace data

The DB2 accounting trace produces hundreds of valuable metrics that are contained within SMF 101 records. Many of these metrics provides information which are absolutely essential to control and tune application performance.

The most important categories of information provided are:

- Plan name, authorization id, connection id, correlation id, subsystem id;
- Plan executed SQL, getpages, synchronous I/O, asynchronous I/O, BP and GBP activity;
- Plan deadlocks, timeouts, lock escalations, lock suspensions;
- Plan elapsed times, CPU times, wait times;
- Package name, package collection;
- Package elapsed, CPU times, wait times;
- Package getpages and executed SQL.

⁹ We estimated a saving of more than 100 MIPS thanks to the I/O rate reduction.

All of the metrics discussed in this chapter, unless otherwise stated, are collected in SMF 101 (IFCID 003 and 239). All the words in **bold** are SMF 101 record fields.

5.6 Class 1 metrics

Class 1 is the default accounting trace class.

It provides many of the information types needed to start plan analysis and tuning, such as plan identification, lock activity, SQL activity, and many others which are collected in IFCID 003 and then in SMF 101 subtype 0.

Just to give an example, you can use this information to identify all the plans' "victims" of deadlock (**QTXADEA**) and timeout (**QTXATIM**) events.

However the most important information provided are elapsed and CPU time.

Class 1 elapsed time has to be calculated as the difference between Ending Store Clock Time (**QWACESC**) and Beginning Store Clock Time (**QWACBSC**)¹⁰.

Class 1 CPU time has to be calculated as the sum of the CPU seconds used by:

- Main task; calculated as the difference between Ending CPU Time (**QWACEJST**) and Beginning CPU Time (**QWACBJST**)¹¹;
- Stored procedures (**QWACSPCP**);
- UDF¹² (**QWACUDCP**).

All these metrics include the time spent in DB2 and outside DB2 (usually called "in application"). This is the major limitation of class 1 elapsed and CPU time.

Another limitation of class 1, is that it only provides the total elapsed and CPU time from the first SQL call thru the thread termination. So this information, as depicted in Figure 10, is generally less complete than what you can get from other sources (such as SMF 110 for CICS transactions, IMS log records for IMS transactions, SMF 30 for batch jobs, etc.), because it doesn't cover all the UOW (Unit Of Work) life.

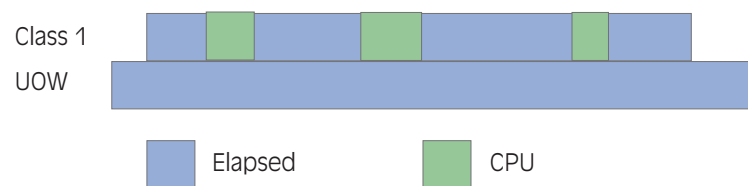


Figure 10

Additional class 1 metrics provide elapsed time spent running stored procedures (**QWACSPEA**) and UDF (**QWACUDEA**). These times are already included in the Class 1 elapsed time. The **QWACCLS1_ZIIP** field provides information about the amount of the CPU time that is consumed while running on a zIIP.

¹⁰ QWACESC contains the accumulated elapsed time when a roll-up record is written with accumulated accounting data.

¹¹ QWACEJST contains the accumulated CPU time when a roll-up record is written with the accumulated accounting data.

¹² User Defined Function.

5.7 Class 2 metrics

When you start accounting class 2, many additional trace points are activated to measure plan entry/exit from DB2 events. Gathered information are not written to any external destination but the accounting facility uses it to compute the additional total statistics that appear in SMF 101 (IFCID 003).

Class 2 elapsed time has to be calculated as the sum of the elapsed seconds spent in:

- Main task (QWACASC);
- Stored procedures (QWACSPEB);
- UDF (QWACUDEB);
- Native stored procedures (QWACSPNF_ELAP);
- Triggers under an enclave (QWACTREE);
- Triggers not under an enclave (QWACTRET).

Class 2 elapsed time is the time spent in DB2. The time spent “in application” can be easily calculated by subtracting Class 2 from Class 1 elapsed time. It is reported in Figure 11 using a white bar. This is very important information to know because it will drive your tuning efforts in the right direction.

A similar approach has to be used to calculate Class 2 CPU time; you have to sum the CPU seconds used in:

- Main task (QWACAJST);
- Stored procedures (QWACSPTT);
- UDF (QWACUDTT);
- Native stored procedures (QWACSPNF_CP);
- Triggers under an enclave (QWACTRTE);
- Triggers not under an enclave (QWACTRTT).

Class 2 CPU time is the time used in DB2. The CPU time used “in application” can be easily calculated by subtracting Class 2 from Class 1 CPU time¹³. As for the elapsed time, this is very important information to know because it will drive your tuning efforts in the right direction.

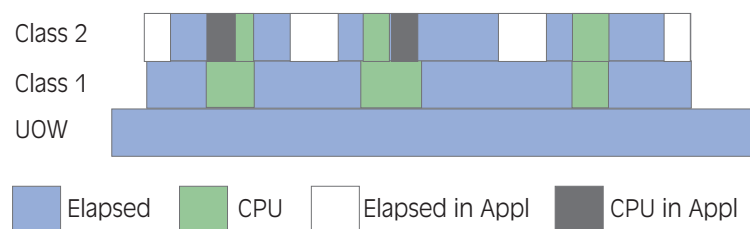


Figure 11

¹³ Before comparing Class 1 and Class 2 CPU time you need to add the CPU consumptions measured by QWACSPNF_CP, QWACTRTE and QWACTRTT to Class 1.

A real life example is provided in the following figure showing a comparison between three different UOW types: a batch job, a DB2 utility and an IMS long running transaction (BMP).

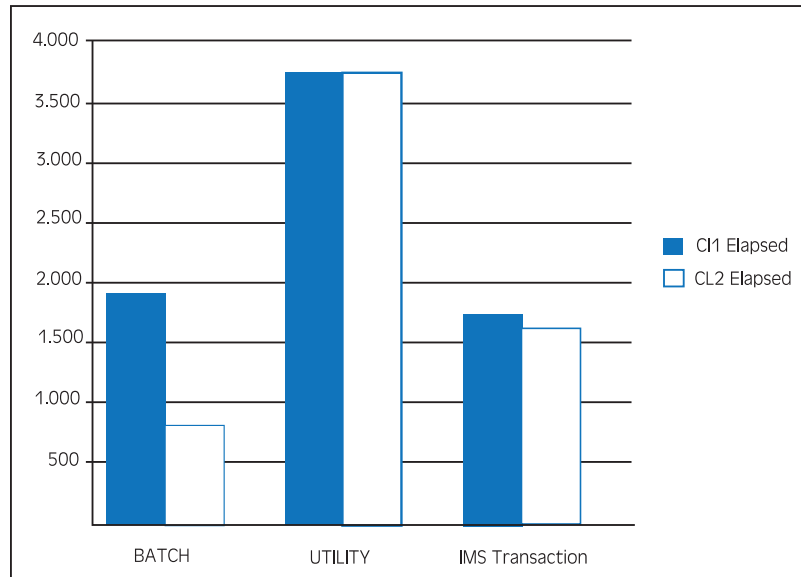


Figure 12

Most of the batch elapsed time has been spent in application, while all the DB2 utility elapsed has been spent in DB2. The IMS transaction has spent most of the time in DB2 and only a small part of it in application.

A similar behavior is shown in Figure 13 for the CPU time use by the batch job and the DB2 utility, while the IMS transaction CPU time is about the same in application and in DB2.

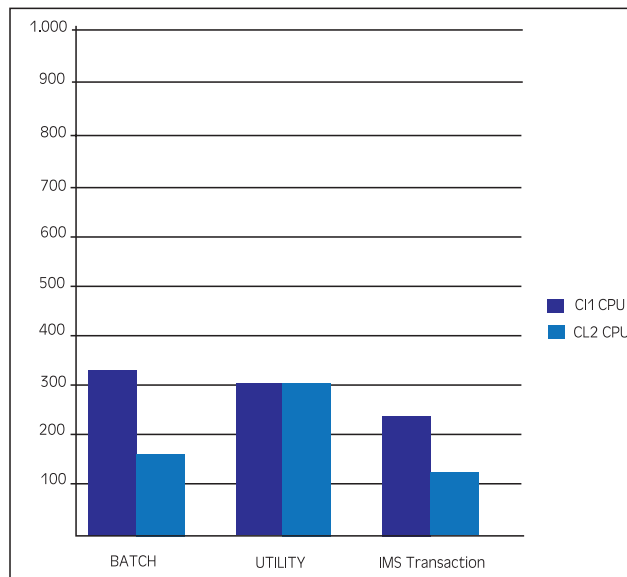


Figure 13

5.8 Class 3 metrics

There can be a big difference between Class 2 elapsed and CPU time. So a very important question to answer is: Where does a plan spend time when not using CPU? To get this answer you need to start accounting class 3. It will provide additional detailed information about a lot of possible wait events and times in DB2 which will be added to SMF 101 (IFCID 003).

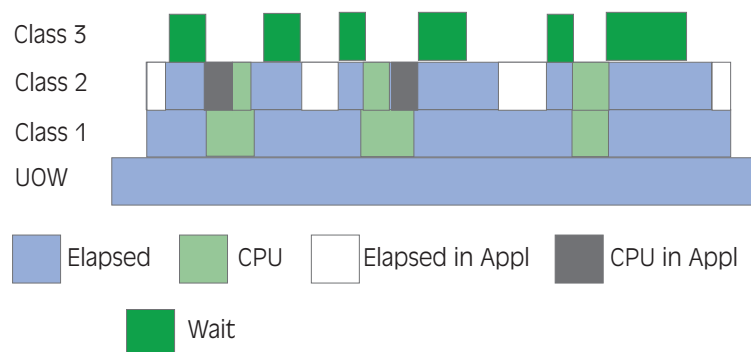


Figure 14

About thirty different plan wait times are tracked¹⁴; the most common of them are:

- Total lock/latch wait time (QWACAWTL); it doesn't include locks and latches due to group-level lock contentions in a data sharing environment;
- Total synchronous I/O wait time (QWACAWTI); while it includes both synchronous read and write I/O, it is normally dominated by read operations;
- Total other read I/O wait time (QWACAWTR); read I/O performed under another thread; it includes sequential prefetch, list prefetch, sequential detection;
- Total other write I/O wait time (QWACAWTW); write I/O performed under another thread;
- Total log write I/O wait time (QWACAWLG).

Figure 15 shows the relevant wait times for the three UOW in our example. Wait time for synchronous and other read I/Os explain all the wait time for the DB2 utility and the IMS transaction. Main wait reasons for the batch job are synchronous I/Os and lock/latch.

¹⁴ Corresponding variables showing the number of wait events are also available.

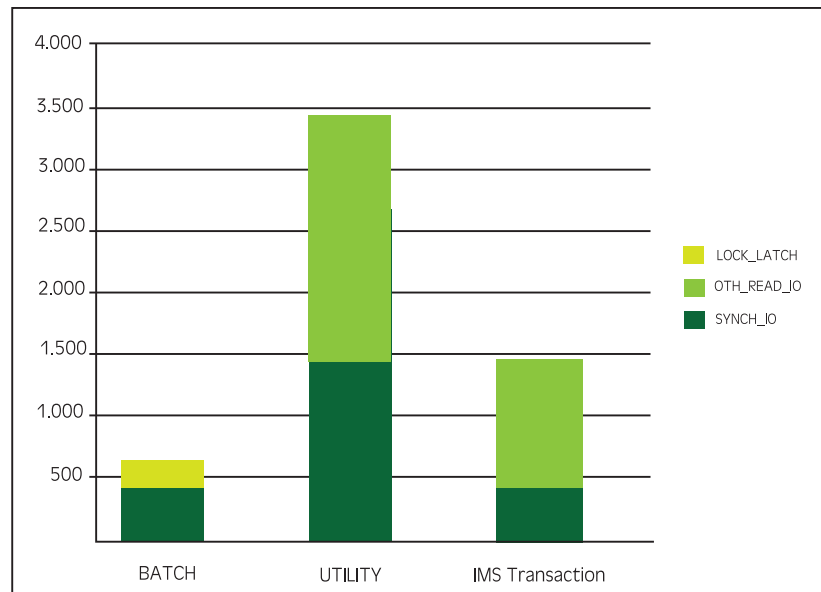


Figure 15

If you subtract the total Class 3 wait time and the Class 2 CPU time from Class 2 elapsed time you get what is normally called a "not accounted" DB2 wait. It is that part of the Class 2 elapsed time when DB2 is waiting for external reasons such as CPU, paging, etc.

You can expect not accounted DB2 wait to be negligible for important and short UOW such as CICS or IMS transactions while you can normally accept bigger values for batch jobs.

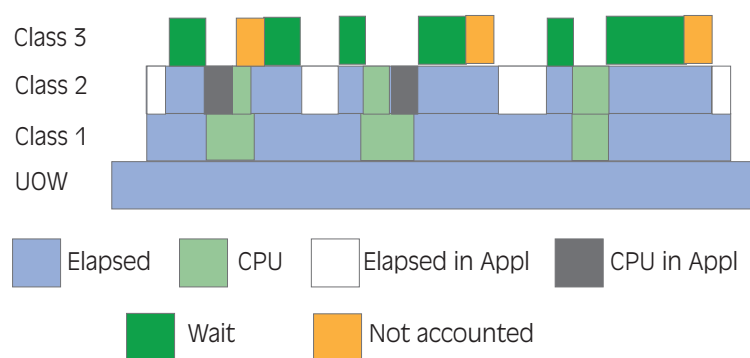


Figure 16

5.9 Class 7, 8 and 10 metrics

Accounting trace 7, 8 and 10 produce package (or DBRM¹⁵), information which are collected in IFCID 239 and then in SMF 101 subtype 1.

If a package is executed multiple times, all executions of the package are grouped into a single block of information.

Class 7 is similar to class 2, in that it shows in DB2 elapsed and CPU times, while class 8 is similar to Class 3, in that it shows wait times.

Class 10 is only available starting from DB2 V8. In V8, IBM decided to provide a lot of new information about lock activity, buffer pool activity and SQL statistics in IFCID 239. They were initially produced using the class 7 accounting trace but this lead in some situations to unacceptable CPU overhead so IBM decided to provide this information through a new accounting trace class: class 10.

In the following figures we show the result of different combinations of accounting class 7, 8 and 10.

PLAN NAME	PACKAGE NAME	PACKAGE COLLECTION	SWITCH	DB2 CPU	DB2 ELAPSED	DB2 WAIT	GETPAGES (K)	SQL DML
P5060PL1	S5060CSK	T5105	6	70.6	80.0	8.5	2,680	12
P5060PL1	S5060165	T5164	14	77.3	319.1	198.3	1,826	2,820.427
P5060PL1	S5060537	T5135	6	77.2	81.0	3.6	716	98,768
P5060PL1	S507008	T5075	1	31.0	35.2	4.1	1,153	27
P5060PL1	S5060165	T5124	2	11.0	51.2	38.3	261	395,783

Figure 17

Data reported in Figure 17 come from a DB2 subsystem where all the classes are active. You can see which packages have been executed by the P506PL1 plan, how much CPU they used, and their elapsed time (class 7).

The total wait time, (and all the related details not shown here), is provided by class 8, while the total number of executed getpages and SQL (and all the related details not shown here), is provided by class 10.

PLAN NAME	PACKAGE NAME	PACKAGE COLLECTION	SWITCH	DB2 CPU	DB2 ELAPSED	DB2 WAIT	GETPAGES (K)	SQL DML
S6144835	C6144A	S6125860	480	33.2	261.9	220.0	.	.


Figure 18

Data reported in Figure 18 come from a DB2 subsystem where only classes 7 and 8 are active. You miss information about getpages and SQL.

PLAN NAME	PACKAGE NAME	PACKAGE COLLECTION	SWITCH	DB2 CPU	DB2 ELAPSED	DB2 WAIT	GETPAGES (K)	SQL DML
P4965SAM	S4965MEE	C4965SAM	1	18.5	32.8	0	79	1,208,350
P4965SAM	S4965CAM	C4965SAM	1	10.8	14.5	0	65	630,270
P4965SAM	S4965FO5	C4965SAM	3	27.3	564.8	0	929	117,492
P4965SAM	S4965PRD	C4965SAM	78	17.2	38.8	0	2,183	154
P4965SAM	S4965DY5	C4965SAM	76	21.3	37.9	0	3,220	1,051

Figure 19

¹⁵ BIND for a DBRM is always to a package in DB2 V10. Every DBRM will be bound to a package, and plans will need one or more packages for execution.



Data reported in Figure 19 come from a DB2 subsystem where only classes 7 and 10 are active. You miss information about DB2 wait.

6 Trace overhead

Every performance analyst understands the value of collecting DB2 metrics; you can answer virtually any performance question if the right trace was running when an issue occurred. Unfortunately it's not always possible to run all the traces all the time; the main reason being the overhead.

The overhead of running a DB2 trace depends essentially on three factors:

- The trace type¹⁶,
- The activated classes,
- The workload type.

In this chapter we will discuss three types of overhead:

- The CPU overhead to produce the DB2 IFCIDs and write them in SMF;
- The disk space overhead to collect DB2 SMF data;
- The CPU overhead to process DB2 SMF with Capacity Management applications.

The CPU overhead to produce statistics trace IFCIDs is negligible. This is one of the reasons, (the other is the usefulness of statistics trace data), why, in DB2 V10, IBM forces the writing of the most important IFCIDs to 1 minute.

The amount of SMF data produced by the statistics trace is also negligible, as is the related SMF processing overhead.

Every site can safely activate the statistic traces. At least classes 1 and 3 should always be activated while class 5 is only needed in data sharing environments.

In the following pages we will therefore discuss only the accounting trace overhead.

6.10 CPU overhead to produce accounting trace IFCIDs

The likely cost of the accounting traces is documented in the DB2 manuals. Unfortunately which manual and chapter to check depends on the DB2 version.

In any case the indications provided are neither clear nor complete. Only in DB2 V9 has an estimate of the possible overhead of each specific accounting trace class been provided; in other DB2 versions only class 2 estimates are provided, (which, by the way, are very different from the ones provided in V8 and V10).

From the table below, the following conclusions can be drawn:

- Class 2 is the most expensive trace to run.
- Class 3 adds minimal overhead to that.

¹⁶ Only statistics and accounting traces will be discussed in this document.

DB2 version	Manual	Chapter	Description
7/8	Administration Guide	26/25	Typically, an online transaction incurs an additional 2.5% when running with accounting class 2/ A typical batch query application, which accesses DB2 more often, incurs about 10% overhead when running with accounting class 2.
9	Performance Monitoring and Tuning Guide	21	Class 1: Typically, less than 5% transaction overhead, except for fetch-intensive applications that do not use multi-row fetch. Class 2: CPU overhead can be 20% for fetch-intensive applications. Class 3: Less than 1% CPU overhead. Class 7 and 8: Less than 5% CPU overhead.
10	Managing Performance	38	Typically, an online transaction incurs an additional 2.5% when running with accounting class 2. A typical batch query application, which accesses DB2 more often, incurs about 10% overhead when running with accounting class 2.

Figure 20

In other parts of the manuals you can find the following sentence:

"If accounting trace classes 2 and 3 are activated, activating accounting trace classes 7 and 8 incurs minimal additional performance cost".

Accounting class 10, (package detail trace), has been separated from class 7 because of the CPU overhead, so you can assume that using this class can be expensive.

The only way to reduce the CPU overhead, is to not run some trace classes or to not run them all of the time. Unfortunately, classes 1, 2 and 3 are essential for application performance tuning and, as we saw, class 2 is the one introducing most of the overhead.

Based on our experience the biggest impact of adding classes 7 and 8 is not the CPU overhead but the amount of data produced. However, for one reason or another, many customers only activate these classes at certain times of the day.


It's important to note that accounting and monitoring classes 2, 3, 7, 8 and 10 produce the same IFCIDs. So if, for example, you are already running monitor classes 2 and 3, activating accounting classes 2 and 3 will add negligible overhead.

6.11 Collecting SMF

When SMF records are created, they are written to buffers inside the SMF address space. When a buffer fills up, or the MAXDORM¹⁷ time is reached, the buffer is written to a Control Interval in one of a set of SMF VSAM datasets (SYS1.MANx).

When a SYS1.MANx fills up, SMF automatically switches to the next empty SYS1.MANx dataset. At this point a user process is normally started to extract all the data from the full SYS1.MANx dataset, save them to a sequential file, and clear that dataset, making it available for use again.

¹⁷ See SMFPRMxx parameters.



If SMF records are being created faster than they can be moved from the SMF address space to the SMF VSAM datasets, one of the following things will happen (depending on the setting of the NOBUFFS parameter in the SMFPRMxx member):

1. A message will be issued to inform you that all SMF buffers are full, and subsequent SMF records will be discarded until space becomes available in the buffers;
2. The system enters a restartable wait state.

The same may happen if the user process is not able to dump and clear SYS1.MAN datasets quickly enough.

The amount of SMF data produced by the accounting trace can be very large. There are normally a large volume of SMF 101 records and they provide hundreds of different metrics¹⁸. In some sites activating the accounting trace could stress the use of both the SMF buffers and SYS1.MANx datasets.

The first thing to do, in these situations, is to try to tune the SMF process by setting:

- An appropriate number of SMF buffers to handle peaks (BUFSIZMAX parameter in SMFPRMxx);
- Appropriate values for AVGLRECL, CISIZE and BUFSPACE when allocating the SYS1.MANx VSAM datasets;
- Appropriate record blocking (27,998) of SMF sequential files.

If this is not enough, you may decide to move to SMF logstream mode using the System Logger and its Coupling Facility structures to greatly improve the SMF collection process performance¹⁹.

In addition or as an alternative, you may also try to reduce the number and/or the size of SMF 101 records.

Possible actions to reduce the number of SMF 101 records are:

- Deactivating all the package related classes (7, 8 and 10); this will eliminate SMF 101 subtype 1 records;
- Using ACCUMACC to accumulate more DDF or RRSF threads executions in one SMF record²⁰.

Possible actions to reduce the size of SMF 101 records are:

- Deactivating some plan or package related classes (e.g. collect only 1, 2, 7);
- Using SMF compression²¹.

In the following figure a comparison is provided between the size of SMF records produced daily using classes 1, 2 and 3, (plan), and classes 7, 8 and 10, (package).

¹⁸ SMF 101 subtype 0 (plan) are normally less than SMF subtype 1 (package) records but they provide more metrics.

¹⁹ See: "SMF Logstream Mode - Optimizing the New Paradigm", IBM Redbooks, SG24-7919-00

²⁰ ACCUMACC has been greatly improved in DB2 10

²¹ SMF compression could reduce SMF size by 90%. It is only available in DB2 10. It's important to remember that using compression will introduce some CPU overhead both when writing and processing SMF.

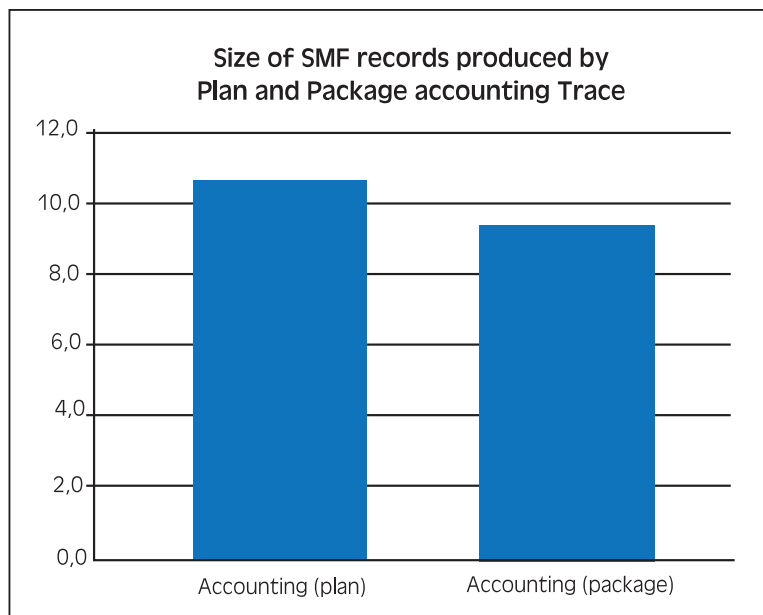


Figure 22

At this site, activating all the accounting trace classes for packages almost doubled the size of SMF 101 records.

6.12 Processing SMF

Every company has products or home-made applications in place to analyze SMF data. SMF processing is normally done in non critical hours (often during the night).

However it may happen that, for many reasons, SMF processing invades the hours when business critical applications run. This may have some undesirable consequences such as:

- An increase in the overall CPU utilization, which will degrade system performance;
- An increase in the 4-hour rolling average, which could raise the monthly software bill;
- An elongation of the SMF processing due to contention with other, more important, workloads and the consequent delay of the availability of capacity management information.


SMF processing can be moved to an external platform, (e.g. Intel based).

This will provide many advantages in terms of cost and it will offer comparable or better performance.

In the following table a comparison between the SMF processing of a 3 GB file on two different platforms, using the same tools, is provided.

	Clock (Ghz)	CPU	Memory (GB)	Region (MB)	Disks	Concurrent workloads	Elapsed (sec)
IBM 2097-722	4.4	12	64	128	many	many	1.195
PC Intel based	2.67	4	4	n/a	1	none	1.026

Figure 23



It's worth noting that on the mainframe there were many concurrent workloads, while the PC was dedicated. On the other hand, the mainframe used different disks for input, output and working files, while only one physical disk was available on the PC.

The bottom line is that SMF processing performance on a PC²² can be comparable, or better, than on the mainframe, because you can easily add disks to a server but it is unlikely you can dedicate a mainframe system to SMF processing.

A final note is about CPU usage: Processing a 3 GB file used about 450 CPU seconds in about 20 minutes. This means that, in these 20 minutes, SMF processing used about 38% of one processor which accounts for about 240 MIPS.

7 Conclusions

Collecting DB2 metrics in SMF is a challenge at many sites, but it can't be avoided because the information provided is essential for tuning DB2 subsystems and applications.

To tune DB2 subsystems, you need to activate the appropriate statistic trace classes. They produce negligible overhead, both in terms of CPU used and space needed in memory and on disks, so they should be activated at every site using a 1 minute interval.

To tune DB2 applications you need to activate the appropriate accounting trace classes. This can produce consistent CPU overhead, stress the SMF architecture, use a lot of disk space and greatly increase the CPU used to process SMF for Capacity Management activities. However, any of these issues can be addressed using the right parameters, software levels and technologies but you need to be aware that there is a cost you will have to pay.

But as an old CMG motto says: "If you can't measure IT, you can't manage IT".

SEGUS Inc is the North American distributor for EPV products

For more information regarding EPV for z/OS, please visit www.segus.com or call (800) 327-9650

²² In reality, companies running SMF processing outside of the mainframe use a server.