SEGUS

White Paper

Analyzing DB2 Overhead

Fabio Massimo Ottaviani - EPV Technologies

1 Introduction

In almost any mainframe environment, DB2 is the main repository where all of the most important company information is stored. This information is normally accessed by many applications via Batch, TSO, CICS, IMS , DDF, WebSphere, etc.

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.

A fourth address space is normally also present: The Distributed Data Facility (DIST) address space (aka DDF), which provides support for remote requests, (coming from outside the system where DB2 resides).

The z/OS standard accounting mechanism, based on cross memory services, attributes CPU usage to the requesting address space. Only a small part of the CPU used to serve requests arriving in DB2 is normally charged to MSTR, DBM1 and IRLM address spaces. These parts can be considered as wholly DB2 overhead.

The situation is different, however, for the DIST address space, which serves remote requests coming from outside the system, and therefore CPU usage cannot be charged to any requesting address space. In this case, DIST CPU usage includes both application activity and DB2 overhead.

Other address spaces can be used to run DB2 stored procedures[1]. Almost all the CPU used to run stored procedures is charged to the requesting address space, which can eventually be the DIST (if the stored procedure is called by a remote request). The CPU charged to the address spaces running DB2 stored procedures can be considered as wholly DB2 overhead, but it is normally negligible, so it will not be discussed further in this paper.

In the following chapters we'll discuss:

- Which metrics can be used to evaluate the DB2 system address spaces overhead;
- How to separate application productive work from the DIST address space overhead;
- What can be done to understand if the DB2 overhead is excessive and where it comes from.

1 Native stored procedures run in DBM1.

2 Evaluating DB2 system address spaces' overhead

DB2 overhead for MSTR, DBM1 and IRLM can easily be evaluated using SMF 30,100 or 72 records.

a) Using SMF 30 interval records (subtype 2 and 3); select the records belonging to MSTR, DBM1 and IRLM address spaces and sum the CPU time provided in the SMF301CS, SMF301CU, SMF30HPT, SMF30RCT, SMF30CPS, SMF30CPT fields;

b) Using SMF 100; you have to use the QWSAEJST and QWSASRBT fields. A section for each DB2 system address space is provided, so to get MSTR, DBM1 and IRLM overhead you have to sum the values corresponding to all of them[2]. Remember that these counters have been accumulated since DB2 was last started. So a de-accumulation step is required to get the numbers relative to the analyzed period of time.

c) Using SMF72 records requires a preliminary assignment of DB2 system address spaces to a specific WLM service or report class. Then you have to:

1. Select the records belonging to these classes;
2. Normalize zAAP and zIIP service units, provided in the R723CIFA and R723CSUP fields, to standard CPUs speed multiplying respectively by the R723NFFI and R723NFFS coefficient (they normally have the same value), and dividing by 256;
3. Subtract normalized zAAP and zIIP service units from the service units provided in the R723CCPU field;
4. Convert TCB and SRB service units values in R723CCPU and R723CSRB to CPU seconds, multiplying by the system service units per second value, and dividing by the TCB and SRB coefficients (provided in the R723MCPU and R723MSRB fields);
5. Sum the values obtained in the previous step to the values provided in the R723CIIT, R723CHST and R723CRCT fields.

We applied all the methods described above and calculated the DB2 System Address Spaces overhead using the following variables:

- AS1000VH, based on SMF 100 records;
- AS0720VH, based on SMF 72 records'
- AS300VH, based on SMF 30 interval records.

The graph in Figure 1 (see next page), allows us to compare the results we obtained. Only the "prime shift" hours, between 8am and 5pm are presented.

2 The QWSAPROC field allows you to identify each DB2 system address space.

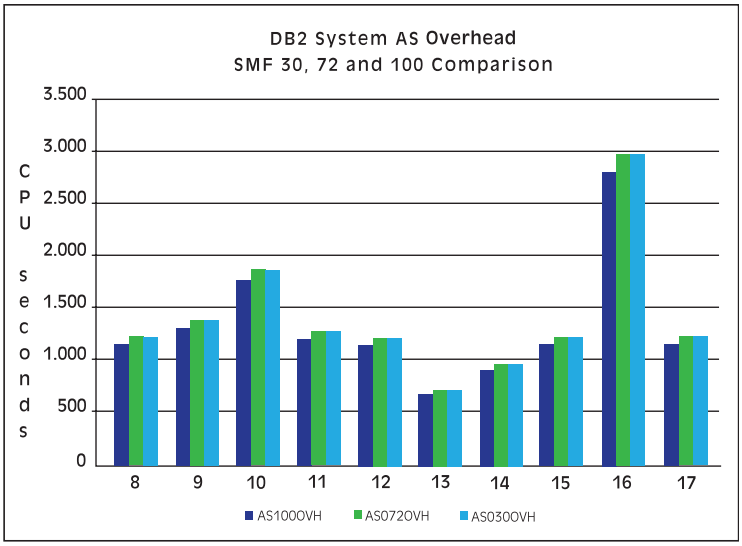

Figure 1

AS0300VH (based on SMF 30, shown here in light blue) and AS0720VH (based on SMF 72, green) values are almost identical.
AS1000VH (based on SMF 100, dark blue) values are always a little lower than them.

3 Evaluating the DIST address spaces overhead

To evaluate the DIST address space overhead a different approach is needed. This address space serves requests coming from outside the system, so there is no address space to charge CPU back using standard cross memory services. Therefore DIST CPU usage includes both overhead and application productive work.

Again you can use SMF 30, 100 or 72 records to estimate DIST overhead, but the required steps are not exactly the same as before.

a) Using SMF 30; select the records belonging to the DIST address space, then:

   1. Estimate the TCB part of the overhead by subtracting SMF30DET and SMF30ENC values[3] from SMF30CPT;
   2. Sum the obtained value to the CPU time provided in the SMF30ICS, SMF30ICU, SMF30HPT, SMF30IIP, SMF30RCT, SMF30CPS fields.

b) Using SMF 100; use the QWSAEJST and QWSASRBT fields. In other words, the section of the SMF 100 record that represents the DIST (DDF) address space (QWSAP-ROC = DIST). Remember that these counters are accumulated since DB2 was last started. So a de-accumulation step to get the numbers relative to the analyzed period of time is required:

   1. Estimate the SRB part of the overhead subtracting QWSAPSRB from QWSASRBT;
   2. Add the value obtained to the TCB time provided in the QWSAEJST field.

3 Application productive work runs on an enclave.

c) Using SMF 72 records requires a preliminary assignment of DIST system address space and DDF work to a specific WLM service or report class[4]:

1. Select the records belonging to the DIST class;
2. Normalize zAAP and zIIP service units, provided in the R723CIFA and R723CSUP fields, to standard CPUs speed multiplying respectively by the R723NFFI and R723NFFS coefficient (they normally have the same value), and dividing by 256
3. Subtract normalized zAAP and zIIP service units from the service units provided in the R723CCPU field;
4. Convert TCB and SRB service units values in R723CCPU and R723CSRB to CPU seconds, multiplying by the system service units per second value and dividing by the TCB and SRB coefficients (provided in the R723MCPU and R723MSRB fields);
5. Add the values obtained in the previous step to the values provided in the R723CIIT, R723CHST and R723CRCT fields.

In this case too, we applied all of the methods described above and calculated the DB2 DIST Address Spaces overhead using the following variables:

- DS1000VH, based on SMF 100 records;
- DS0720VH, based on SMF 72 records;
- DS0300VH, based on SMF 30 interval records.

The graph in Figure 2 allows us to compare the results we obtained. Only the "prime shift" hours between 8am and 5pm, are presented.

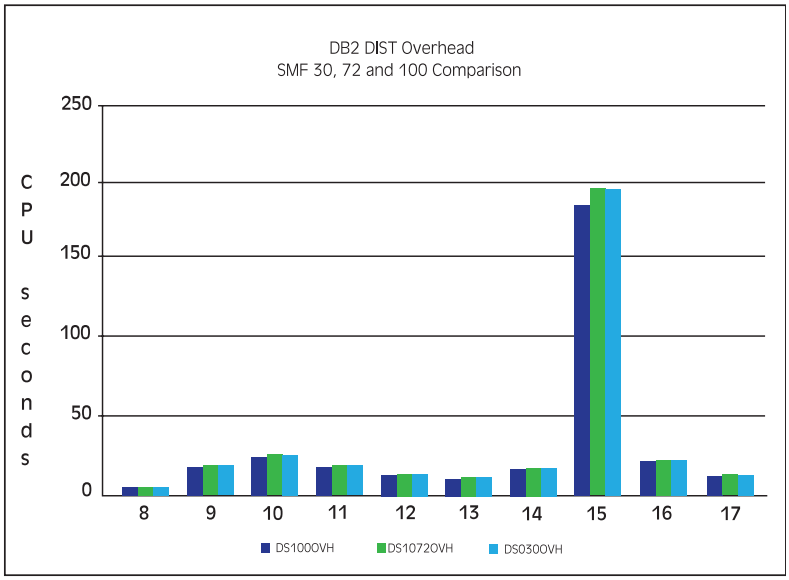

Figure 2

DIST overhead is much lower than in figure 1. However it's important to remember that this is only a part of the overhead introduced by DDF which, as every DB2 workload, requires service to MSTR, DBM1 and IRLM.

4 **A DDF servi**ce class always has to be defined to avoid DDF requests run in the default SYSOTHER service class as Discretionary work.

DS0300VH (based on SMF 30) and DS0720VH, (based on SMF 72), values are again almost identical. DS1000VH (based on SMF 100) values are normally lower than them. It's important to note, that both DS0300VH and DS1000VH are estimated values, while DS0720VH is measured.

The graph in Figure 3 doesn't refer to the DB2 overhead. It is a comparison of DDF workload consumption obtained after subtracting the DB2 overhead from the DIST address space (based on SMF 30 and SMF 100) CPU usage[5].

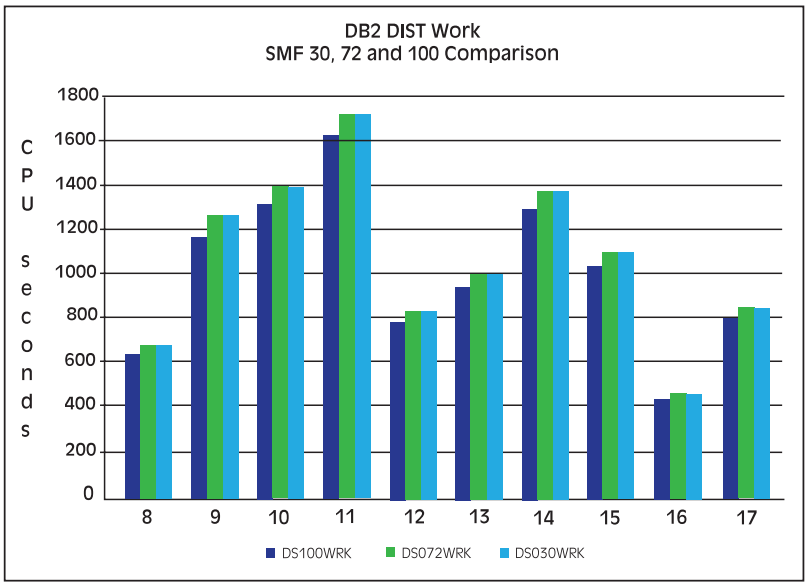

Figure 3

4 Is DB2 overhead excessive?

Everyone agrees that DB2 overhead (and overhead in general), should be as low as possible. Exactly how low depends on workload characteristics and DB2 infrastructure, (e.g. data sharing), but also on system and applications tuning.

How can you evaluate the DB2 overhead percentage? The process is conceptually very simple, you just have to divide the total DB2 overhead (including all DB2 system address spaces, DIST and stored procedures address spaces) by the sum of:

- total CPU used by DB2 applications (productive work);
- total DB2 overhead.

Unfortunately, getting the CPU used by DB2 applications is not that easy.
The same address space may, or may not, be used by DB2 in different executions and there is no way to tell from looking at SMF 30 records. The same is true for service and report classes found in the SMF 72 records.

5 DS072WRK is based on separate SMF records for service/report class dedicated to application productive work.

However, the CPU used by DB2 applications can be obtained very easily by using SMF 101. You just have to sum the CPU usage[6] recorded in all the SMF 101 records produced by a DB2 subsystem.

There are a couple of issues to consider when using SMF 101, these are:

a) Producing and managing SMF 101 records is a very long and resource-intensive task;
b) They are written at thread termination (or commit), so it's not easy to synchronize them with SMF 30, 72 and 100 records, which are used to estimate the DB2 overhead. These are written at interval expiration.

At most sites, SMF 101 records are already collected and used. If you have to start SMF 101 records collection, you can reduce their size by activating only the required accounting trace classes (1 and 2) and reduce their number by exploiting the ACCUMAC parameter[7].

The second issue is even more critical. To get the correct DB2 overhead percentage, you should calculate it by considering all the DB2 subsystem "life", from the start to the end, which normally spans several weeks or months. Depending on workload characteristics, when data is aggregated at day or, especially, hour level, the results can be erratic.

A good compromise can be obtained by using a rolling average; this gives a daily profile of the DB2 overhead percentage and mitigates against that erratic behavior.

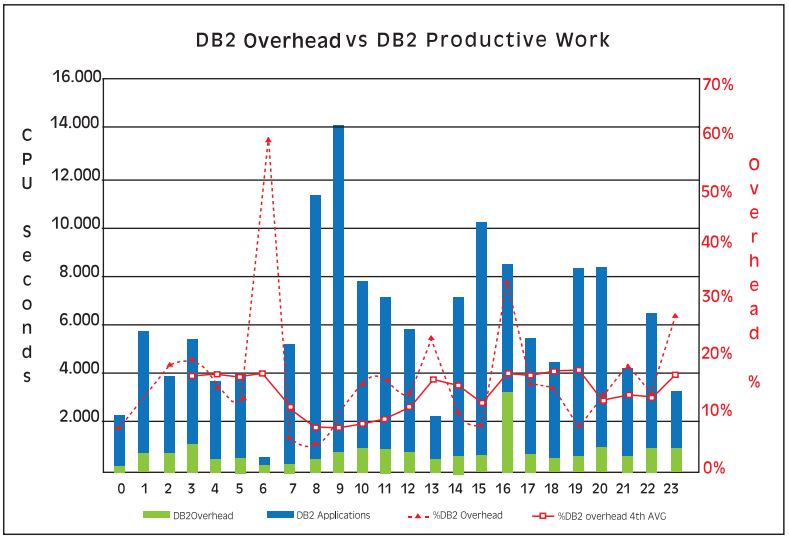

Figure 4

The dotted line in Figure 4 represents DB2 overhead percentage by hour. You can see the erratic behavior due to very light load, (at hours 6, 13 and 23), and synchronization issues at 16.
The solid line represents DB2 overhead, calculated using a 4 hour rolling average. Values are in the 9%-19% range. DB2 overhead calculated at the daily level is about 14%. E.g: (9% + 19%) / 2 = 14%

Once you have estimated the DB2 overhead percentage you have to decide if it is excessive or not.

6 Class 2 (in DB2) CPU has to be used; it is provided in the QWACAJST field.
7 See DB2 Installation Guide.

You can use rules of thumb such as:

- Less than 10%, in the peak hours, for non data-sharing DB2 subsystems;
- Less than 15%, in the peak hours, for data-sharing DB2 subsystems.

Normally, a better approach is to track DB2 overhead for a sufficient length of time when DB2 has just been tuned, and to then set an appropriate baseline for the DB2 overhead percentage.

5 Where does the DB2 overhead come from?

If the DB2 overhead appears to be too high, you should try to understand what the reason is for the excess overhead by looking at the DB2 address spaces' CPU usage.

There are some general rules to keep in mind:

- DBM1 is normally the major contributor to DB2 overhead, followed by MSTR;
- IRLM and DIST (or DDF) overhead is normally very low;
- most of DB2 address spaces' activities are performed in SRB mode.

If any of these rules are violated you should start a deeper investigation.

All the metrics presented in the following figures are provided by the DB2 Statistics trace and collected in SMF 100 records.

The graph in Figure 5 shows the situation at one of our customers' sites.

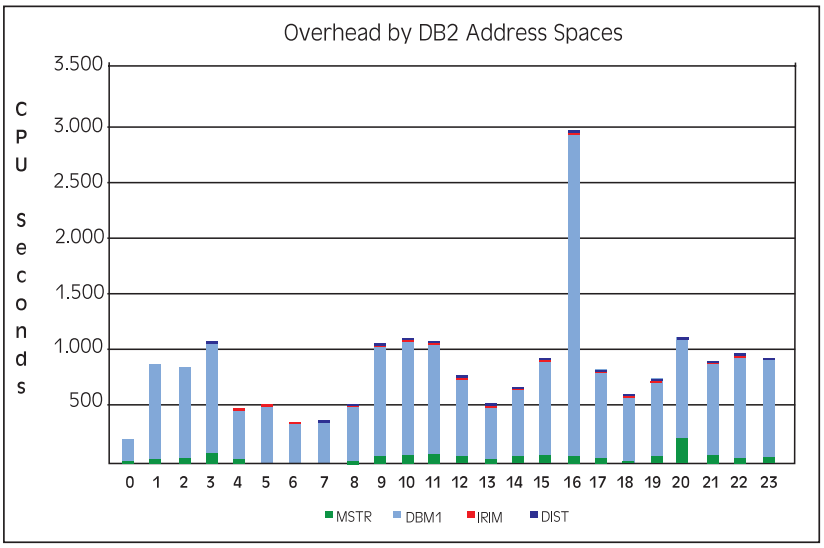

Figure 5

IRLM and DIST overhead are negligible. DBM1 accounts for most CPU usage while MSTR takes the rest. MSTR shows a peak at hour 20 while DBM1 has a very high peak at hour 16.

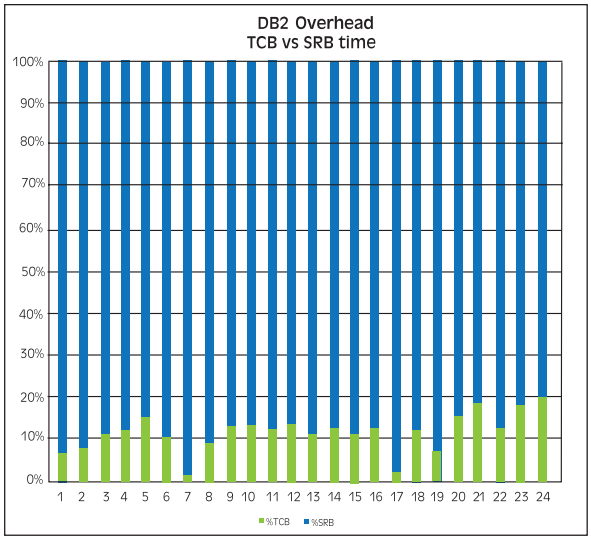

Figure 6

The graph in Figure 6 above shows that SRB time (shown here in blue), accounts for about 85% of DB2 overhead in the peak hours, and it's never less than 80%.

5.1 Analyzing MSTR overhead

Looking at the following graph in Figure 7, you can appreciate the correlation between the amount of log control intervals written (QJSTCIWR from SMF 100), and the total amount of CPU used. The peak in MSTR overhead is mainly due to the evening peak in log write activity.

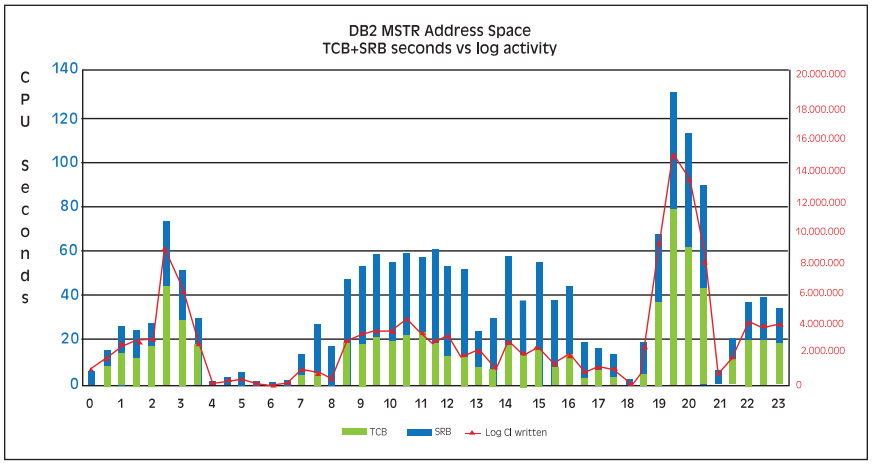

Figure 7

The most important activities of MSTR are: logging, check pointing, BSDS (Boot Strap Data Set) access and back-out. They all run in SRB mode.

Logging, check pointing and BSDS access normally doesn't use a lot of CPU, while backout can be a very CPU-intensive task, depending on the number of updates to be discarded. Another important task performed by the MSTR address space is Archiving. Archiving is performed in TCB mode, so when archiving takes place TCB time equals and also exceeds MSTR SRB time.

Looking at the graphs in Figures 7 and 8 you can note that TCB time is comparable to SRB time at many times, and during the evening peak it's the major contributor to MSTR overhead. The red line in Figure 8 shows the strong correlation between the number of archive logs (QJSTALW from SMF 100), produced and the TCB time used by MSTR.

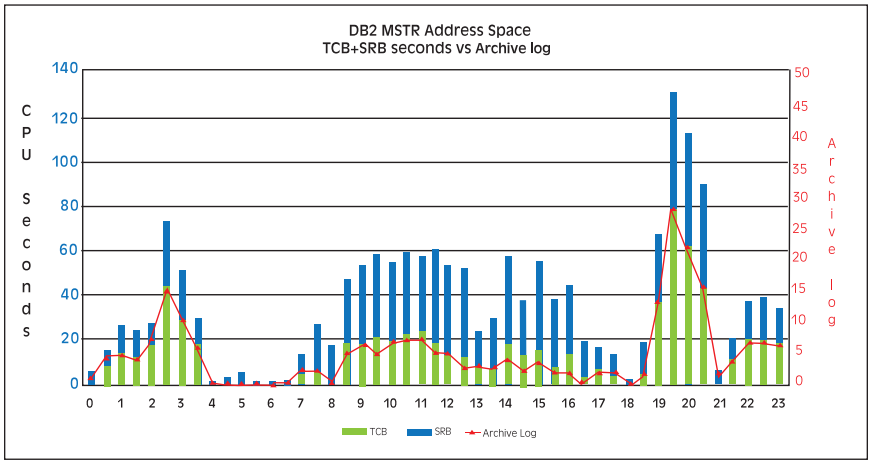

Figure 8

5.2 Analyzing DBM1 overhead

The DBM1 address space should show the largest DB2 overhead component, and most of it should be SRB.

DBM1's most important activities are: asynchronous reads (list, dynamic and sequential prefetch), deferred writes, virtual storage management (pools) and data set open/close[8].
All of these activities are performed in SRB except for open/close, which runs in TCB mode.

8 Castout, GBP asynchronous writes, P-lock negotiation and other data sharing related activities can also increase DBM1 overhead.

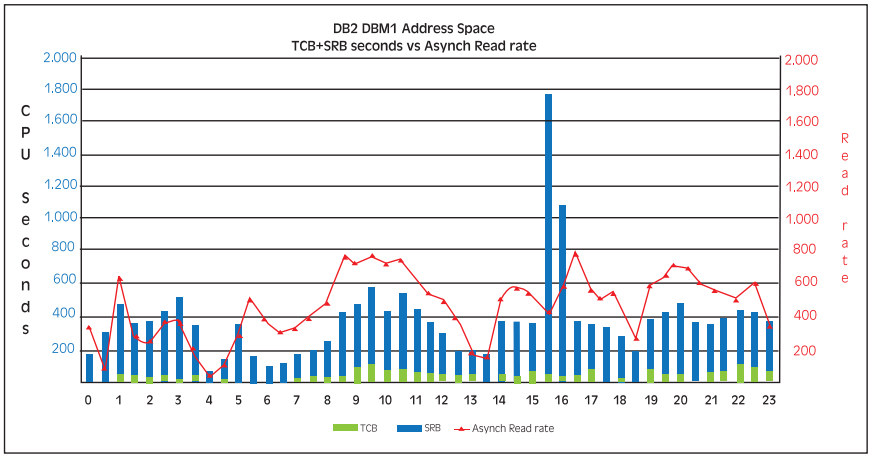

Figure 9

The graph in Figure 9 shows low TCB time values; this reflects the open/close rate, (QBSTDSO from SMF 100), not reported in the graph, which is well below the commonly accepted rule of thumb (1 per second).

It also shows a good correlation between asynchronous read rate (the sum of QBSTLIO, QBSTDIO and QBSTPIO from SMF 100), and DBM1 overhead in most intervals. However, asynchronous read rate doesn't explain the afternoon peak.

Looking at the EPV for DB2[9] Critical Events view the reason for that peak can quickly and easily be discovered. Sequential prefetch has been disabled thousands of times between hours 15 and 16 on Buffer Pool 7. So many pages have been loaded individually instead of using optimized prefetch I/Os, thus increasing DBM1 CPU consumptions.

DB2X CRITICAL EVENTS BY HOUR																								
EVENTS	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
LOG BUFFER WAITS	546	23	603	78	1	8	0	213	45	181	317	543	1076	463	335	156	253	60	54	2,050	3,954	5	99	16
BP1-SEQ PREFETCH DISABLED NO READ ENGINE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	61	10	0	0	0	0	0	0	0
BP2-SEQ PREFETCH DISABLED NO READ ENGINE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	0	0	0	0	0	0	0
BP7-SEQ PREFETCH DISABLED NO READ ENGINE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6,180	896	0	0	0	0	0	0	0
BP87-SEQ PREFETCH DISABLED NO READ ENGINE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0	0

Figure 10

Finally, it's interesting to note that a lot of log buffer waits have been measured between the hours of 19 and 20, (see row 1), as a consequence of the peak in log CI written, as discussed in the previous chapter.

9 EPV™ for DB2 is a product developed by EPV Technologies SrL and distributed throughout North America by SEGUS Inc.

6 Conclusions

DB2 has to perform a lot of activities in order to provide the services needed by applications, so a certain amount of overhead has to be accepted.

A systematic measurement of this overhead must be performed in order to be aware of its amount compared to the DB2 productive work.

It's also important to track it historically in order to be able to quickly identify any anomalies, especially when changing the DB2 release, parameters, or when new applications enter the system.

Finally, by splitting the overhead by DB2 address space and processing mode (TCB or SRB), very useful indications can be obtained regarding the possible reasons for the overhead, allowing you to focus your tuning efforts on specific DB2 activities.

This paper shows how to use a simple set of SMF metrics to do that.

SEGUS Inc is the North American distributor for EPV products

For more information regarding EPV for DB2, please visit www.segus.com or call (800) 327-9650