SEGUS

White Paper

CPU and zIIP usage of the DB2 system address spaces

Fabio Massimo Ottaviani - EPV Technologies

1 Introduction

Each DB2 subsystem always includes three system address spaces (AS):

- 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.

The z/OS standard accounting mechanism, based on cross memory services, attributes CPU usage to the requesting address space. Only a part of the CPU used to serve requests arriving for DB2 is charged to MSTR, DBM1 and IRLM address spaces.
This part, which can be considered as wholly DB2 overhead, is normally a small percentage of the DB2 application CPU but it can be pretty high in absolute terms.

For many years the focus of DB2 overhead analysis has been on DBM1 that was, among the DB2 system address spaces, the major CPU consumer.
DB2 evolution during recent years significantly changed this picture by allowing the off-loading of a big portion of DBM1 and, from V11, also part of MSTR activity to zIIP.

However, at the same time, new functions have been provided in the MSTR address space which greatly increased its CPU usage. Sometimes this may cause real issues which need to be addressed.

In this paper we'll discuss:

- the amount of work which has been off-loaded, or could be off-loaded, to zIIP;
- the impact on CPU usage of new functions available in the MSTR address space;
- the impact of insufficient zIIP capacity on DB2 CPU usage and performance.

2 System address spaces CPU usage in DB2 V9

In the table in Figure 1[1], the most important functions contributing to CPU and zIIP usage of each DB2 system address space is provided. Underlined functions only apply to data sharing environments.

System AS	TCB mode	SRB mode
DBM1	Opening and closing of dataset	Deferred writes
	DBM1 full system contraction	Prefetch reads
	Preformat	Parallel child tasks
	Extend	Castouts
		Asynchronous GBP writes
		P-lock negotiation
		Notify Exit
		Page set close or pseudo-close to convert to non-GBP dependent
		GBP checkpoints
	Archiving	Physical log writes
	BSDS processing	Thread deallocation
MSTR		Update commit (including unlocking of page P-locks)
		Backouts
		Checkpoints
	Error checking	Local IRLM latch contention
	Management	IRLM and XES global contention
IRLM		Asynchronous XES contention
		P-lock negotiation
		Deadlock detection

Figure 1

Up to DB2 V9, no off-loading to zIIP was possible for any of the above functions.
The DBM1 address space was by far the major CPU user. MSTR and IRLM usage was normally much lower.

An example of a typical environment is shown in Figure 2 on the next page.

[1] DB2 11 for z/OS Managing Performance — SC19-4060-07

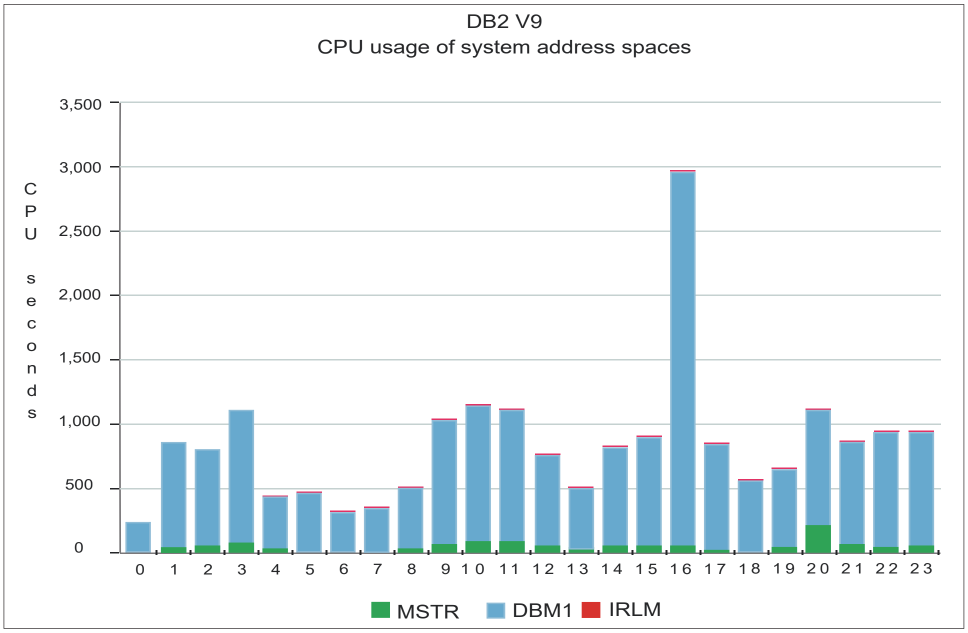

Figure 2

The most important factor to consider, when looking at DBM1 CPU, was the split between TCB (Task Control Block) and SRB (Service Request Block) time. The guideline was very simple: most of the activity of DBM1 had to be in SRB mode. Every different situation had to be investigated.

An example of what you had to expect is presented in Figure 3.

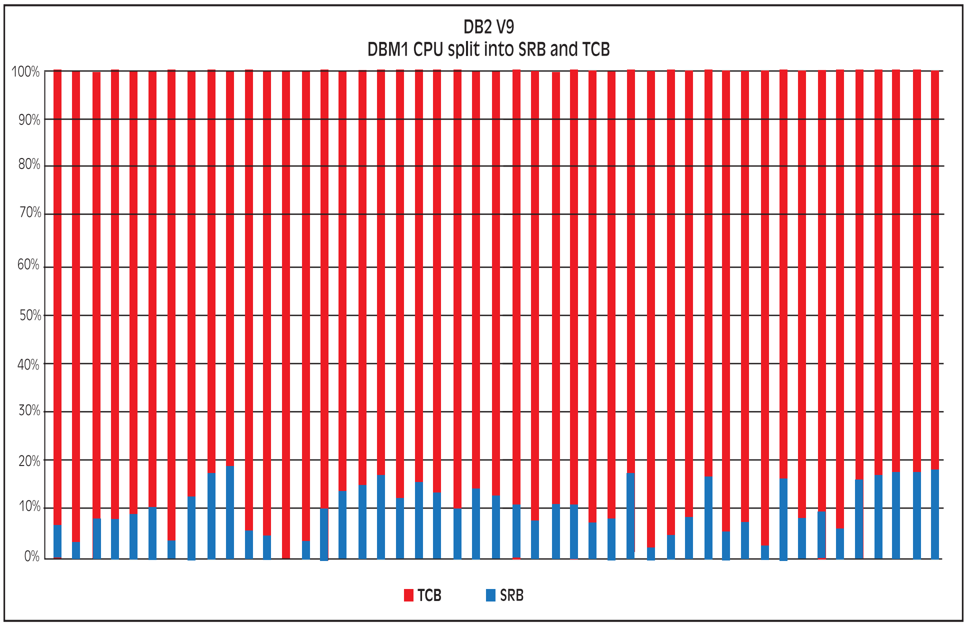

Figure 3

3 System address spaces CPU usage in DB2 V10 and V11

Starting with DB2 V10, IBM off-loaded some DBM1 functions to zIIP—highlighted in blue in Figure 4. The most important of them is the prefetch activity[2].

With DB2 V11, other functions have been off-loaded; they are highlighted in green in Figure 4. As you can see, almost all the DBM1 activities in SRB mode have been off-loaded together with the MSTR log related activities.

System AS	TCB mode	SRB mode
DBM1	Opening and closing of dataset	Deferred writes
	DBM1 full system contraction	Prefetch reads
	Preformat	Parallel child tasks
	Extend	Castouts
		Asynchronous GBP writes
		P-lock negotiation
		Notify Exit
		Page set close or pseudo-close to convert to non-GBP dependent
		GBP checkpoints
	Archiving	Physical log writes
	BSDS processing	Thread deallocation
MSTR		Update commit (including unlocking of page P-locks)
		Backouts
		Checkpoints
	Error checking	Local IRLM latch contention
	Management	IRLM and XES global contention
IRLM		Asynchronous XES contention
		P-lock negotiation
		Deadlock detection

Figure 4

The effect of this off-loading completely changed the picture. A lot of DBM1 activity now ran on zIIP instead of CPU.

A real life example, based on one of our customer's data, showing the amount of CPU and zIIP capacity used by DBM1 is provided in Figure 5 on the next page.

[2] It's important to remember that the CPU used for synchronous rad activity needed to move pages from disk to buffer pools is charged to applications not to DBM1.

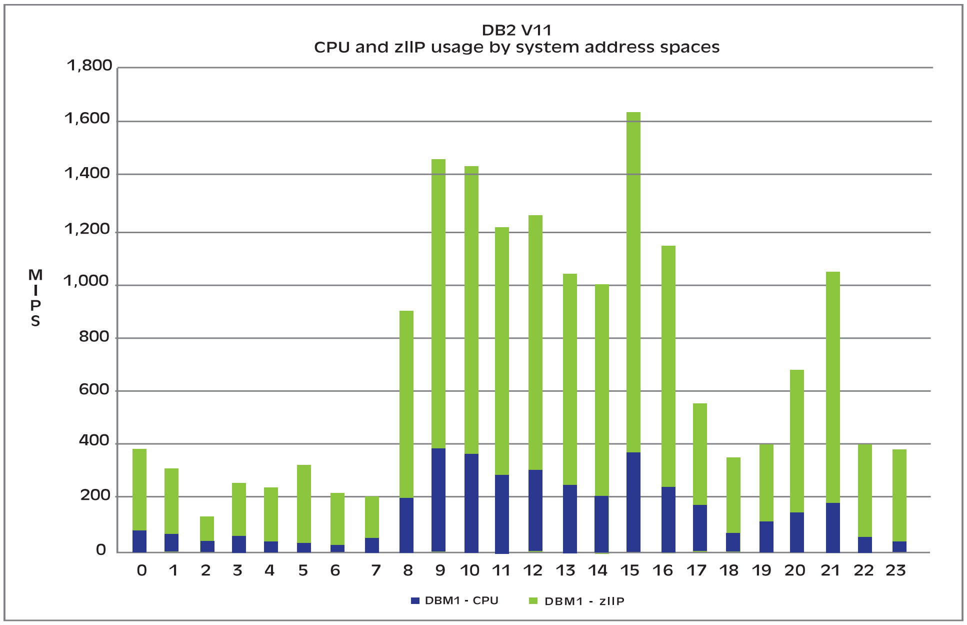

Figure 5

However, by looking at the full picture, which includes all the DB2 system address spaces, something unexpected can be found in Figure 6:

- MSTR zIIP off-load seems to be very low;
- MSTR CPU usage has become much higher than we were used to seeing in V9.

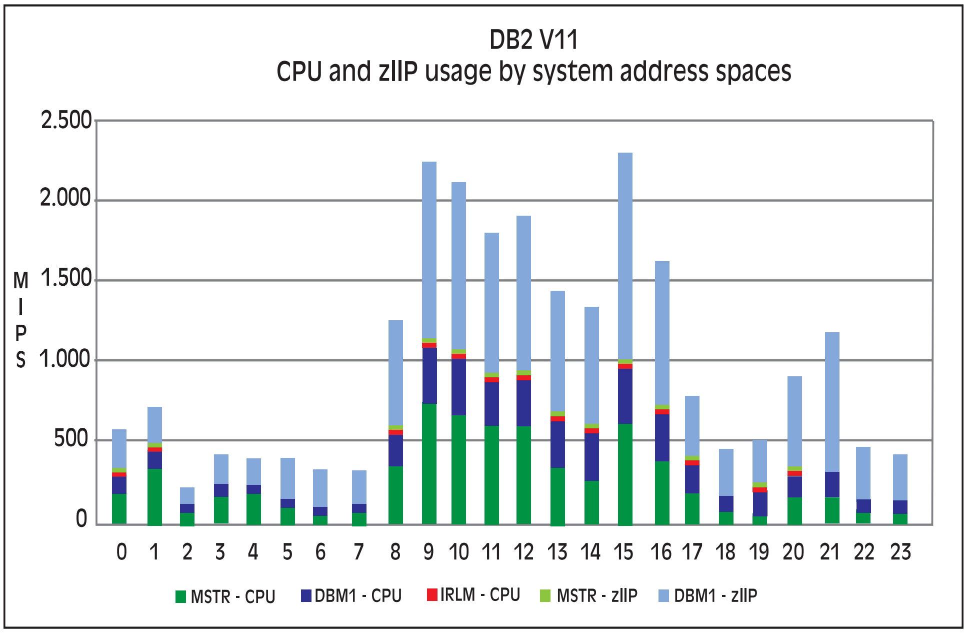

Figure 6

In Figure 7 on the next page, we compare the log write rate to the zIIP MIPS used by the DB2 MSTR address space. We can estimate that between hours 18 and 24, zIIP MIPS every 1,000 write per second are used.

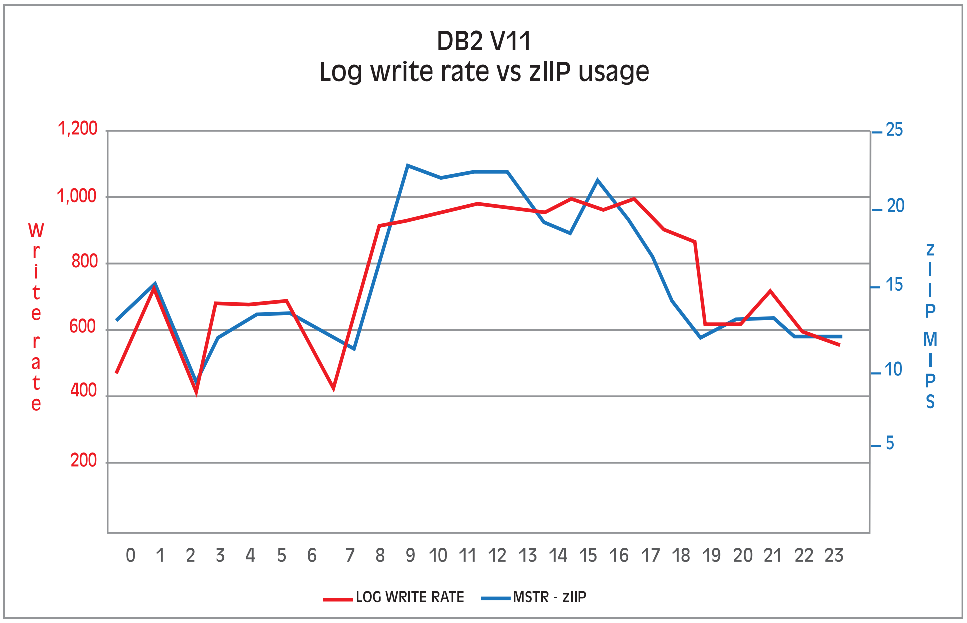

Figure 7

So, unless your DB2 subsystem has very high logging activity, you mustn't expect big savings in this area.

Now the question is, why MSTR CPU usage has become so high?

4 Performance impact of zIIP over utilization

Unfortunately, configurations where the number of zIIPs is insufficient to host all the zIIP eligible workload, are still quite common.
When the system parameter IIPHONORPRIORITY is set to “YES” (default)[3], the most evident consequence of zIIP being too busy, is the overflow of zIIP eligible work to standard CPUs—with a possible increase of hardware and software costs.

However, this is not the only consequence.
The overflow is driven by the ”zIIP needs help” mechanism which is not immediately activated as soon as a zIIP eligible piece of work is queued due to all zIIPs being busy, but it has to wait the time specified in the ZIIPAWMT OPT parameter.
By default, when HIPERDISPATCH=YES, a piece of work has to wait for up to 3.2 milliseconds before it may receive help from standard CPUs.

The bottom line is that the unavailability of zIIPs not only increases CPU utilization but also introduces a performance penalty for zIIP eligible workloads.

The table below, in Figure 10, shows the Performance Index (PI) of a WLM service class hosting the DBM1 address space.
The goal—80% of execution velocity—is always missed, as indicated by PI values higher than 1. A PI of 2.1 at 8 in the morning means that the DBM1 velocity was lower than 40%.

As you can see here, the major delay reason is zIIP delay.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
PI	4.0	1.4	9.2	1.8	1.5	2.6	11.7	1.6	2.1	1.5	1.4	1.7	1.4	1.6	1.3	1.3	1.2	1.3	1.2	1.8	13.3	1.6	1.3	3.1
CPU DELAY	2.6	0.4	10.3	1.4	0.9	1.8	11.4	2.7	3.9	3.4	2.9	3.8	3.9	4.4	3.1	3.0	2.9	4.2	4.1	5.7	39.8	2.4	1.9	1.3
DISK DELAY	0.1	0.1	0.1	0.2	0.3	0.2	0.0	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.1	0.2	0.3	0.2	0.9	0.1	0.1	0.1	0.2	0.1
Ziip DELAY	20.4	3.7	37.8	4.7	4.5	9.2	38.6	5.4	13.0	9.1	12.6	16.6	8.6	12.6	6.9	9.1	7.5	7.0	7.9	14.2	28.0	5.5	2.7	14.1

Figure 10

This customer is currently running DB2 V10.
When they upgrade to V11, if the zIIP capacity remains so inadequate, that performance penalty will be exacerbated because, as discussed in previous chapters, many more DBM1 activities will become zIIP eligible.

Insufficient zIIP capacity in DB2 V11 will also delay MSTR log activity—with direct and severe consequences on DB2 application performance.

[3] Setting IIPHONORPRIORITY to NO may degrade system and application performance so most sites use the default value.

5 Evolution of MSTR functions

Many functions have been extended and added to the MSTR AS in the recent DB2 versions.

The first new important function, introduced with DB2 V9, is the System Monitor task; this task continuously checks the health of the system at one-minute intervals, trying to address the following major issues:

- CPU stalls in DB2 resulting in latch contention;
- critical usage of DBM1 virtual storage below the 2GB bar.

In the first case, the DB2 System Monitor tries to clear the latch contention by temporarily increasing the latch holder priority by invoking WLM services.
In the second case, the DSNV510I warning message is sent when thresholds[4] are reached together with DSNV512I messages listing the agents that consume the largest amount of DBM1 storage below the bar.

With DB2 V10 and V11, virtual storage constraints have practically been eliminated by exploiting 64-bit virtual storage and allowing DB2 to use much more real storage to improve application performance.

So IBM introduced other new functions in order to help customers monitor the real storage used by DB2 subsystems. IFCID225 was enhanced, with many new fields providing information about real and auxiliary storage used, to back DBM1 and DIST private areas, shared and common 64-bit objects.

With DB2 V10, the amount of virtual storage that needs to be monitored is an order of magnitude higher than before, so an increase of MSTR CPU usage can be considered normal.

However, by tracking MSTR CPU usage when migrating to DB2 V10, some customers discovered an unacceptable increase. They reported it to IBM who identified the bugs in the code and fixed them[5].

In Figure 8, you will find a comparison of MSTR CPU usage between V9 and V10 in a “normal” situation (with all of the necessary corrections applied).

[4] Thresholds set at 88, 92, 96, and 98% of the available storage.

[5] Most relevant corrections: DB2 APAR (PM49816) and z/OS APAR (OA37821).

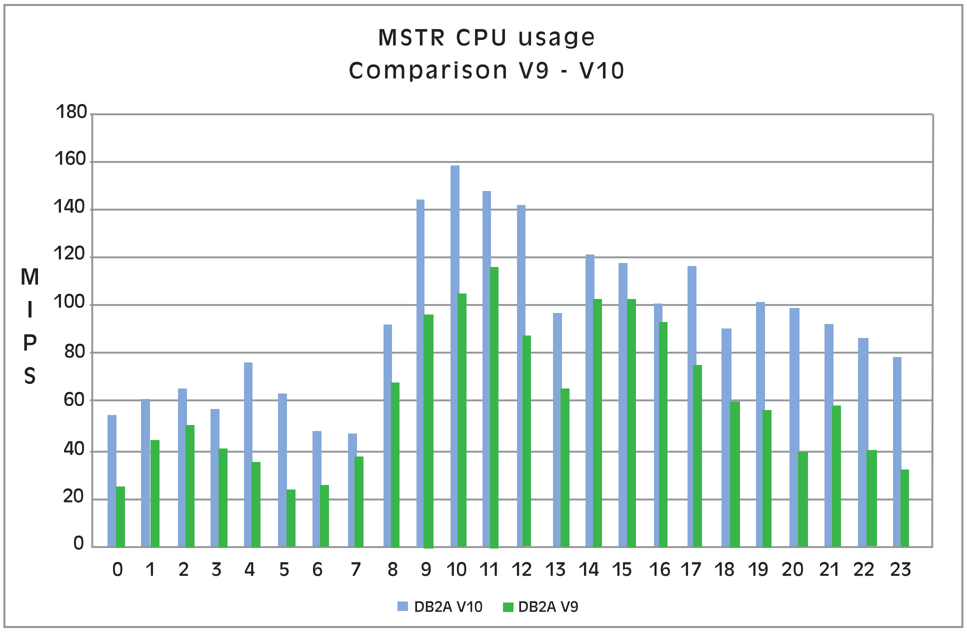

Figure 8

As you can see, the increase in the amount of CPU used by MSTR is pretty evident.

Perhaps the most important new function performed by MSTR in DB2 V10 and V11, is the interaction with the z/OS RSM (Real Storage Manager) to control and manage the DB2 real storage.

Two new system configuration parameters are provided:

- REALSTORAGE_MAX; this parameter sets the maximum of real and auxiliary storage (in GB) a DB2 subsystem can use; DB2 will terminate if this threshold is reached; new message DSNS003I will be written by MSTR when DB2 approaches the specified threshold;
- REALSTORAGE_MANAGEMENT; this parameter will tell DB2 how to manage thread storage pages that are backed in real storage but not used anymore.

DB2 thread storage is now allocated in a memory object in 64-bit shared storage. To give back real storage frames backing virtual pages inside a memory object, the IARV64 DISCARDDATA service has to be used together with the optional KEEPREAL parameter.
When DB2 uses DISCARDDATA with KEEPREAL(YES), the storage is only "virtually freed"; RSM flags the page as freed or unused, but the storage is still in real storage with the data and charged against DB2.
When DB2 uses KEEPREAL(NO), RSM frees and reclaims the page immediately.

Depending on the above parameter settings and on the system condition, MSTR will enter "contraction mode"[6] in order to free all unused storage that is associated with threads which have done a certain number of commits or have ended and issued appropriate DISCARDDATA requests.

IBM released several corrections in this area, trying to find the best compromise between real storage usage and CPU overhead.

[6] CONTSTOR has to be set to YES (default).

This is the situation after APAR PM99575[7]:

- if REALSTORAGE_MAX boundary is approaching, or z/OS has notified DB2 (through ENF 55 signal) that there is a critical auxiliary shortage; MSTR issues DISCARDDATA requests with KEEPREAL=NO;
- if REALSTORAGE_MANAGEMENT is set to OFF, MSTR will not issue DISCARDDATA requests; more real storage is used;
- if REALSTORAGE_MANAGEMENT is set to AUTO (default) with no paging in the system, MSTR issues DISCARDDATA requests with KEEPREAL=YES to free storage at thread deallocation or after 120 commits; more CPU is used and charged to MSTR;
- if REALSTORAGE_MANAGEMENT is set to AUTO (default) with paging, or REALSTORAGE_MANAGEMENT is set to ON, MSTR issues DISCARDDATA requests with KEEPREAL=YES to free storage at thread deal location or after 30 commits; stack storage is also discarded; even more CPU is used and charged to MSTR.

If you have XCF CRITICALPAGING enabled, you also need to apply z/OS RSM APAR OA44913 or provide Flash Express to the LPAR to obtain the above described behavior.

Unfortunately, even after applying all fixes, some of our customers still complain about excessive real storage usage when REALSTORAGE_MANAGEMENT is set to OFF.

Running with the AUTO default should be the best option but one of our customers, running DB2 V11, recently, had a very bad surprise.

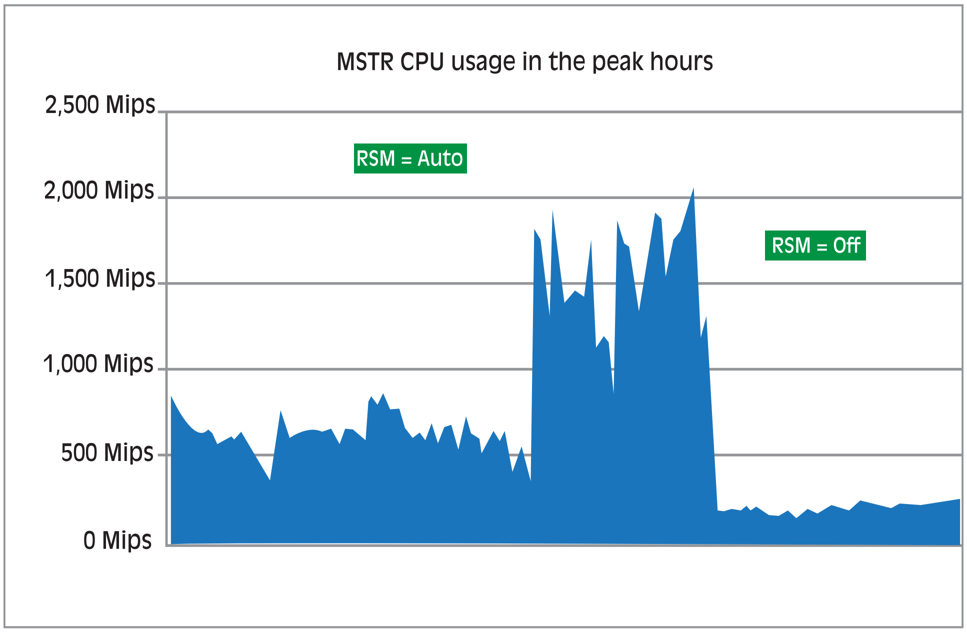

Figure 9

As you can see in Figure 9, MSTR CPU usage, that was already pretty high above 500 MIPS, at a certain point increased up to 4 times.

IBM opened a PMR and, as a workaround, suggested to set REALSTORAGE_MANAGEMENT to OFF. The MSTR CPU usage fell and is now stable, at around 200 MIPS.

[7] This APAR is superseded by PTF UI17080 for DB2 10 and PTF UI17081 for DB2 11.

6 Conclusions

DB2 evolution in the last years allowed more and more DBM1 activities to become zIIP eligible—with positive effects on hardware and software costs. With DB2 V11 MSTR logging activities are also eligible to zIIP.

The availability of enough zIIP capacity is not only a requirement to obtain these benefits, but also an essential condition to avoid a performance degradation of DB2 applications.

On the other hand, the exploitation of 64 bit virtual storage and the availability of much bigger quantities of real storage have increased the functions to be performed by the MSTR address space and consequently its CPU usage.

All the above aspects have to be fully understood in order to be able to control and tune DB2 subsystems and applications.

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