Practical Capacity Planning for Memory

Fabio Massimo Ottaviani – EPV Technologies

July 2010

1 Introduction

The cost of mainframe memory has been substantially reduced during recent years; however its current cost is still in the order of some thousands of Euro per GB.

On the other hand the number of GBs supported by the newest z/OS releases has greatly increased making it possible to almost eliminate paging activity and to fully exploit the many available DIM (Data In Memory) techniques.

While it's true that z/OS can sustain high paging activity, paging and especially page faults are very disruptive to online applications performance.
If a CICS region is getting 10 page faults per second this means that 10 transactions per second are stopped (or a transaction is stopped 10 times) and have to wait for the required pages to be loaded from page datasets. Furthermore you have to consider that CPU cycles are needed to perform paging. This is one of the sources of uncaptured CPU time.

Maintaining data in memory has two additional advantages: it improves performance by avoiding the I/O operations delay and it eliminates the CPU consumption still needed to perform I/O operations.
It’s also important to remember that I/O operations can only be performed by standard CPUs and not by specialty processors (zAAP and zIIP) so the I/O reduction has positive effects on the software bill too.

All these factors mean that memory capacity planning is still a very important activity to perform.

The good news is that from a conceptual point of view the process is always the same:

- evaluate current capacity;
- estimate the utilization baseline;
- estimate the growth;
- forecast future resource needs.

The bad news is that dealing with memory, because of its nature, is more complicated than with other resources.

In this paper we’ll discuss a practical methodology to do that.

2 Evaluating current capacity

The total memory available to a z/OS LPAR can be easily calculated by using the values provided in the following fields of the SMF 71 record type:

- SMF71TFC, contains the number of page frames defined in central storage;
- SMF71FIN, contains number of central storage frames in nucleus;

and applying this simple formula.

$$\text{Total Memory GB} = \frac{4.096 * (\text{SMF71TFC} + \text{SMF71FIN})}{1.024 * 1.024 * 1024}$$

Unfortunately this is not enough. There can be a lot of memory on your mainframe you can't measure through SMF such as memory assigned to:

a) LPARs that are not active (e.g. Disaster Recovery);
b) LPARs hosting Integrated Coupling Facility (ICF)[1];
c) LPARs hosting z/VM[2];
d) LPARs hosting zLinux[3].

At some customer sites we have even found memory not assigned to any LPAR.

Having a complete picture of all memory assignment normally requires some manual activity to be done looking at contracts and the Hardware Management Console (HMC).
This is important when performing Capacity Planning because sometimes instead of buying more you can optimize the use of the existing memory.

The following table shows LPAR memory assignment at a customer site.
All data have been collected automatically (Y in the AUTO column) except for the DRZOSA and DRZOSB disaster recovery LPARs.

LPARNAME	MEMORY GB	TYPE	AUTO
ZOSD	28	z/OS	Y
ZOSC	18	z/OS	Y
ZOSF	14	z/OS	Y
CF01	2	ICF	Y
CF02	4	ICF	Y
DRZOSA	16	DR	N
DRZOSB	14	DR	N

Figure 1

We built this table during a Performance Health Check study at a customer site while searching some memory to give to one of the z/OS LPARs which was suffering strong paging activity.
We discovered that the DRZOSB LPAR was not needed anymore and therefore 14 GB were effectively not used at all.

[1] An estimate can be done by looking at the R744GCSD and R744GTSD fields provided in the SMF 74 record type.
[2] Information about the memory assigned to z/VM LPAR can be obtained by collecting Monitor records.
[3] Information about the memory assigned to zLinux LPAR can be obtained by running the FREE command.

3 Estimating the utilization baseline

Estimating the utilization baseline is an essential step of the capacity planning process; it answers a very complex question: how much is a resource used ?

It may look simple but there are a lot of other questions hidden inside it such as:

- What reference interval to use (30 minutes, 1 hour, a longer interval) ?
- Which hours of the day to consider (just a couple of peak hours, 8-18, 24 hours) ?
- Which days to include (just some specific days, working days, all days) ?
- What is a representative period of the month/year to base it on ?

Once you have answered all these questions you need to use an algorithm (such as top values, average, percentiles, etc.) to get the baseline value.

The process to estimate memory baseline is conceptually the same but there is an important difference to keep in mind: memory is a passive resource so you will not find a ready metric showing memory utilization as for CPU and devices.

The approach we suggest to estimate the utilization baseline uses different methods depending on the type of system you are analyzing.

The first step is classifying z/OS systems as:

- Constrained;
- Abundant;
- Balanced.

3.1 Constrained systems

To understand if a system is memory constrained the best metric to use is the page fault rate (PFR). Starting from the SMF71PIN field provided in the SMF 71 record type, which contains the number of non-VIO page-ins from auxiliary to central storage in each RMF interval, PFR can be easily calculated dividing the SMF71PIN by the number of seconds in the interval.

To decide if the PFR is too high you need a threshold value. In EPV for z/OS we start from a 50 PFR default threshold but we know that each system is different and the right threshold to use really depends on the workload running in the system and on the agreed service levels[4].

So if the value is higher than the threshold you have also to check which address spaces are suffering and if service levels are degrading.
We have seen many situations where DB2 buffer pools or Java heap memory had been defined too large to be hosted by the available real memory thus causing an increase in PFR without any impact on application performance.

[4] So the best thing to do is to look at the system's historical behavior and set the PFR threshold at the maximum PFR the system can sustain without missing service levels.

If the system is really memory constrained in order to set the baseline you have to increase system memory by adding the memory needed to relieve the current constraint and eliminate or greatly reduce PFR.

How can we estimate this “memory demand” ?

We suggest a graphical method consisting of drawing a curve which shows the relationship between system PFR and real memory. To draw the curve we need at least 4 points and, unless you can do real measurements by increasing/decreasing the system real memory, some assumptions have to be made in order to plot them.

The following example will help to understand the method.

SYSP uses 20 GB memory. In the peak hours PFR values are around 250 (above the PFR threshold set to 100) and performance of mission critical applications are degrading. About 4 million slots are also used on page datasets[5].

Putting PFR on the x axis and memory GB on the y axis, the first point of the curve is (250, 20). A second point can be derived making the hypothesis of eliminating PFR by increasing the system real memory by the amount of GB used on page datasets. So the second point should be (0, 36).

A second, more arbitrary, assumption has to be made that there is a quadratic relationship between real memory reduction and PFR increase. This means that if we reduce system memory to 10 GB (20/2) the expected PFR will be 1.000 (250*4).
So the third point should be (1.000, 10) and the fourth point (4.000, 5).

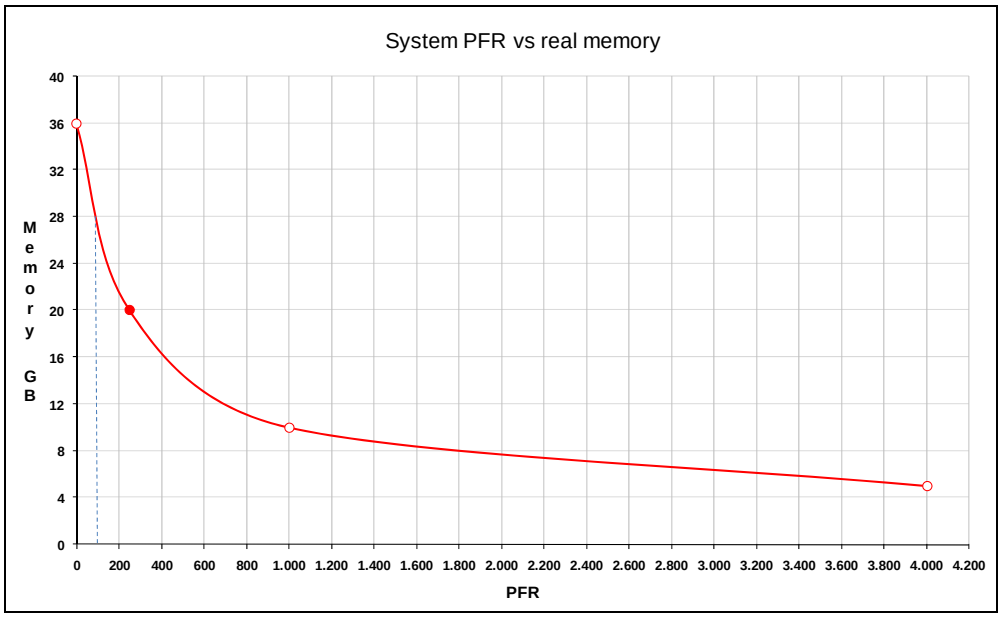

Figure 2

The initial part of the curve (the most interesting part) shows that to reduce PFR to 100 additional 8 GB memory should be enough.
So the memory baseline for the system in the previous example should be set to 28 GB.

[55] From the SMF75AVU field provided in the SMF 75 record type.

3.2 Abundant systems

To understand if the memory assigned to any system is abundant is very important because the excess memory can be reassigned to a memory constrained system.
The first symptom of abundant memory is a zero PFR in most of the critical hours.

If you are in this situation to estimate the memory utilization baseline you can use the following method:

1. estimate the memory used by each service class periods starting from the R723CPRS field provided in the SMF 72 record type which contains the total page residency time in 1024 micro seconds units;
2. sum the memory used across all service class periods;
3. add the memory used to host logical swapped address spaces (batch, TSO, etc);
4. add nucleus and common areas size;
5. divide the above calculated value by 0.95 to get some margin.

Step 1) can be performed starting from the R723CPRS field provided in the SMF 72 record type, which contains the total page residency time in 1024 micro seconds units for each service class period, and applying the following formula:

$$\text{Service Class Period Memory GB} = \frac{\text{R723CPRS} * 1.024 * 4.096}{1.000.000 * \text{RMF interval duration}}$$

To perform step 3) you have to subtract the value calculated in 2) from the values of the SMF71AVS field, provided in the SMF 71 record type, which contains the total number of private frames in use.

Nucleus and common areas size can be roughly estimated in about half GB[6].

The following table shows an example of system memory utilization in the peak hours during some meaningful days calculated using the method described above.
By choosing the 95 percentile the memory utilization baseline can be set to 13,5 GB[7].

This system has 20 GB memory and it shows zero PFR all the time so it can be classified as abundant.

[6] You can calculate a more precise value by summing the SMF71AVP, SMF71ASR, SMF71ALP, SMF71FIN fields provided in the SMF 71 record type.
[7] Percentile has been evaluated on the last 60 days; not all the days are shown in the table.

MEMORY UTILIZATION (MB)					
DATE	8	9	10	11	12
17/06/2010	13.498	13.652	13.738	13.779	13.819
16/06/2010	13.203	13.346	13.424	13.447	13.474
15/06/2010	12.305	12.508	12.601	12.649	12.682
14/06/2010	11.046	11.549	11.657	11.752	11.871
11/06/2010	13.466	13.596	13.702	13.756	13.773
10/06/2010	13.335	13.439	13.536	13.631	13.650
09/06/2010	13.351	13.434	13.527	13.612	13.659
08/06/2010	13.269	13.362	13.413	13.471	13.526
07/06/2010	12.464	12.663	12.768	12.821	12.886
04/06/2010	13.257	13.413	13.481	13.530	13.608
03/06/2010	13.384	13.429	13.483	13.555	13.623
01/06/2010	12.856	12.987	13.050	13.090	13.114
31/05/2010	11.939	12.094	12.178	12.258	12.351
28/05/2010	13.259	13.426	13.539	13.620	13.565
27/05/2010	13.137	13.298	13.409	13.489	13.566
26/05/2010	12.834	12.985	13.056	13.121	13.178
25/05/2010	12.593	12.697	12.784	12.919	12.921
24/05/2010	12.071	12.257	12.357	12.432	12.527
21/05/2010	12.933	13.061	13.163	13.233	13.272

Figure 3

An alternative method can be based on the total amount of available frames provided in the SMF71AFC field (SMF 71). Starting from that, you can very easily calculate the amount of memory used but you'll not have the possibility to analyze memory usage in detail and eventually correct anomalies.

3.3 Balanced systems

If a system has not been classified as constrained or abundant, it can be classified as balanced, which means it has approximately the required amount of memory.
So current system memory is the baseline.

4 Estimate memory needs

4.4 Workload characterization

This method is based on a workload characterization aimed to evaluate the amount of memory used by each workload component. The best data source are the service and report class information[8] provided in SMF 72.

	MB Used				
RPRTCLAS	9	10	11	12	AVG
RBATCH	48,1	46	47,9	46,9	47,2
RDB2C	1006,1	1013,7	1014,8	1019,9	1.013,6
RDB2CDDF	0	0	0	0	-
RDB2D	1736,3	1763,2	1787	1836	1.780,6
RDB2DDDF	0	0	0	0	-
RIMSC	300,7	300,8	300,8	301	300,8
RIMSCMSG	699	707,3	718,7	719,9	711,2
RIMSD	334,9	334,7	335	336	335,2
RIMSDEF	0	0	0	0	-
RIMSDMSG	795,9	803,6	811,8	820,9	808,1
RMONITOR	334,6	337,8	341,3	345,5	339,8
RMQC	388	388,5	388,9	389,5	388,7
RMQD	253,8	254	254,5	254,5	254,2
ROPENMVS	209,3	210,1	213,7	220,9	213,5
RSTC	400,8	401,3	402,5	402,3	401,7
RSYSSTC	1142,2	1161,8	1189,4	1195,4	1.172,2
RSYSTEM	732,1	731,7	733,4	740,6	734,5
RWAS	998,2	1003,6	1010,6	1010,8	1.005,8
RWASCTRL	595	607,3	613,9	614,4	607,7
RWASSERV	1980,7	1992,1	2084,7	2099,5	2.039,3
RWEB	37,9	37,9	37,9	38	37,9
RTSO	0,3	2,8	3,7	0,4	1,8

Figure 4

Figure 4 shows an example of the amount of memory used by report class. Only the peak hours are presented. The AVG column provides the average memory usage calculated over the whole peak period.

The first thing to note is that memory usage is not charged to report (and service) classes assigned to independent enclaves (e.g. RDB2CDDF, RDB2DDDF) or IMS (and CICS) transactions (e.g. RIMSDEF).
RSYSTEM and RSYSSTC accumulate the memory usage from system address spaces[9].

It's important to remember that the memory usage provided in SMF 72 records refers only to the address spaces resident in memory. So to characterize the workload we need to know, for each report class, the average number of resident address spaces (also called MPL).

It can be easily calculated by dividing R723CTRR (total in storage residency time) by SMF72INT (duration of the RMF measurement interval). Both metrics are provided in the SMF 72 records.
Adding the average number of IN address spaces in the peak period (AVG IN AS) we get the following table.

[8] Appropriate WLM definitions are needed in order to get a good characterization.
[9] Operative system level is z/OS 1.9 in this example.

	MB Used					
RPRTCLAS	9	10	11	12	AVG	AVG IN AS
RBATCH	48,1	46	47,9	46,9	47,2	10,9
RDB2C	1006,1	1013,7	1014,8	1019,9	1.013,6	4,0
RDB2CDDF	0	0	0	0	-	
RDB2D	1736,3	1763,2	1787	1836	1.780,6	4,0
RDB2DDDF	0	0	0	0	-	
RIMSC	300,7	300,8	300,8	301	300,8	3,0
RIMSCMSG	699	707,3	718,7	719,9	711,2	187,0
RIMSD	334,9	334,7	335	336	335,2	3,0
RIMSDEF	0	0	0	0	-	
RIMSDMSG	795,9	803,6	811,8	820,9	808,1	123,0
RMONITOR	334,6	337,8	341,3	345,5	339,8	7,0
RMQC	388	388,5	388,9	389,5	388,7	3,0
RMQD	253,8	254	254,5	254,5	254,2	3,0
ROPENMVS	209,3	210,1	213,7	220,9	213,5	4,8
RSTC	400,8	401,3	402,5	402,3	401,7	32,4
RSYSSTC	1142,2	1161,8	1189,4	1195,4	1.172,2	32,0
RSYSTEM	732,1	731,7	733,4	740,6	734,5	23,0
RWAS	998,2	1003,6	1010,6	1010,8	1.005,8	0,5
RWASCTRL	595	607,3	613,9	614,4	607,7	3,0
RWASSERV	1980,7	1992,1	2084,7	2099,5	2.039,3	3,0
RWEB	37,9	37,9	37,9	38	37,9	4,0
RTSO	0,3	2,8	3,7	0,4	1,8	0,2

Figure 5

The one piece of information we still need to complete our characterization: the number of address spaces by report class that are logically swapped. Normally CICS, IMS, DB2 and IMS components are not swapped so you can assume that logical swap concentrates on RBATCH (batch jobs), ROMVS (USS forked address spaces), RSTC (generic started tasks) and RTSO (TSO users).
To estimate the number of logically swapped address spaces we also need some information included in the SMF 70 records.

	Address spaces				
States	9	10	11	12	AVG
IN	450,9	450,8	451,8	451,8	451,3
IN*READY	1,7	1,6	1,8	1,8	1,7
OUT*WAIT	0	0	0	0	
OUT*READY	0	0	0	0	
LOGICAL*OUT*WAIT	59,2	63	70,9	71,5	66,2
LOGICAL*OUT*READY	0	0	0	0	
ASCH*ACTIVE	0	0	0	0	
BATCH*ACTIVE	336,6	336,3	337	336,9	336,7
OMVS*ACTIVE	21	21,6	28,3	29,3	25,1
STC*ACTIVE	150,1	150,8	152,3	153,8	151,8
TSO*ACTIVE	2,4	5,1	5	3,4	4,0

Figure 6

Figure 6 shows that, on average we have about 66 logically swapped address spaces.

To apportion this number to report classes we can use the following algorithm:

- The RBATCH logical swapped AS can be calculated starting from the BATCH*ACTIVE AVG value (in Figure 6) and subtracting the sum of the AVG IN AS values in Figure 5 corresponding to RIMSCMSG and RIMSDMSG report classes[10]; then you have also to

[10] IMS regions runs as batch jobs at this site.

subtract 4 AS out of the 6 in RIMSC and RIMSD[11] and the RBATCH AVG IN AS value (in Figure 5):

$$336,7 – 187 – 123 – 4 – 10,9 = 11,8$$

- ROMVS logical swapped can be calculated starting from the OMVS*ACTIVE AVG value (in Figure 6) and subtracting the AVG IN AS value in Figure 5 corresponding to the ROPENMVS report class:

$$25,1 – 4,8 = 20,3$$

- RSTC logical swapped AS can be calculated starting from the STC*ACTIVE AVG value (in Figure 6) and subtracting the sum of the AVG IN AS values in Figure 5 corresponding to all the report classes running STC (RDB2C, RDB2D, RMONITOR, RMQC, RMQD, RSYSSTC, RSYSTEM, RWAS, RWASCTRL, RWASSERV, RWEB); then you have also to subtract 2 AS out of the 6 in RIMSC and RIMSD[12] and the RSTC AVG IN AS value (in Figure 5):

$$151,8 – 89,5 – 32,4 = 30,9$$

- RTSO logical swapped can be calculated starting from the TSO*ACTIVE AVG value (in Figure 6) and subtracting the AVG IN AS value in Figure 5 corresponding to the RTSO report class:

$$4 – 0,2 = 3,8$$

The complete characterization table is shown in the following figure.

	MB Used									
RPRTCLAS	9	10	11	12	AVG	AVG IN AS	AVG LS AS	% IN	MB/IN AS	MB Total
RBATCH	48,1	46	47,9	46,9	47,2	10,9	11,8	48%	4,3	
RDB2C	1006,1	1013,7	1014,8	1019,9	1.013,6	4,0				1.013,6
RDB2CDDF	0	0	0	0	-					
RDB2D	1736,3	1763,2	1787	1836	1.780,6	4,0				1.780,6
RDB2DDDF	0	0	0	0	-					
RIMSC	300,7	300,8	300,8	301	300,8	3,0				300,8
RIMSCMSG	699	707,3	718,7	719,9	711,2	187,0			3,8	
RIMSD	334,9	334,7	335	336	335,2	3,0				335,2
RIMSDEF	0	0	0	0	-					
RIMSDMSG	795,9	803,6	811,8	820,9	808,1	123,0			6,6	
RMONITOR	334,6	337,8	341,3	345,5	339,8	7,0			48,5	
RMQC	388	388,5	388,9	389,5	388,7	3,0				388,7
RMQD	253,8	254	254,5	254,5	254,2	3,0				254,2
ROPENMVS	209,3	210,1	213,7	220,9	213,5	4,8	20,3	19%	44,5	
RSTC	400,8	401,3	402,5	402,3	401,7	32,4	30,9	51%	12,4	
RSYSSTC	1142,2	1161,8	1189,4	1195,4	1.172,2	32,0				1.172,2
RSYSTEM	732,1	731,7	733,4	740,6	734,5	23,0				734,5
RWAS	998,2	1003,6	1010,6	1010,8	1.005,8	0,5				1.005,8
RWASCTRL	595	607,3	613,9	614,4	607,7	3,0			202,6	
RWASSERV	1980,7	1992,1	2084,7	2099,5	2.039,3	3,0			679,8	
RWEB	37,9	37,9	37,9	38	37,9	4,0				37,9
RTSO	0,3	2,8	3,7	0,4	1,8	0,2	3,8	5%	9,0	

Figure 7

The last two columns show the average number of MB of memory needed for each resident address space and the total MB of memory needed (where it doesn't make much sense calculating that average) by each report class.

Starting from this table you can note run that:

- the memory needed by the system (RSYSSTEM + RSYSSTC) is about 2 GB;
- DB2C (RDB2C) is using about 1 GB while DB2D (RDB2D) is using about 1.8 GB;

[11] IMD DBRC run as STC.
[12] IMD DBRC run as STC.

- each Websphere Application Server (WAS) control region (RWASCTRL) is using about 0,2 GB while each WAS servant region (RWASSERV) is using about 0,7 GB; there are 3 WAS (1 control and 1 servant regions each), on average each one of them is using about 0,9 GB;
- each IMS subsystem (RIMSC and RIMSD) is using about 0,3 GB;
- a message region in IMSC (RIMSCMSG) is using 3,8 MB while a message region in IMSD (RIMSDMSG) is using 6,6 MB;
- each batch job (RBATCH) is using a 4,3 MB while each TSO user (RTSO) is using 9 MB;
- etc.

Using the values provided in Figure 7 you can also estimate the amount of memory needed for logical swapping[13].

4.5 Additional memory needed for workload growth

Estimating the memory needed to support workload growth can be relatively easy when the growth can be translated in additional address spaces.
So if you will add 100 TSO active users in the peak hours, based on the values in Figure 7, we can assume that:

- 5 TSO users will be resident in memory (%IN is 5%);
- 95 TSO users will be logically swapped out;
- About 45 MB memory will be needed for the IN users (5*9 MB);
- About 428 MB memory will be needed for the Logical Swapped OUT users (95*9*,50) assuming that their working set has been trimmed at 50%.

You can follow a similar approach if your growth estimate leads to the need of additional regions (CICS, IMS and Websphere) or complete subsystems.
Based on our characterization if you need to add 10 message regions to IMSD you will need about 66 MB (10*6,6 MB) while a new WAS will require about 1 GB.

In some situations you can also introduce some calibration factor if you know that, for example, the new WAS will host more memory demanding applications.

However sometimes the workload growth can’t be translated to additional address spaces.
It may happen when more transactions are performed by the same CICS, IMS or WAS regions.

Another typical case is DB2. If you increase TSO users, IMS regions or WAS subsystems you will reasonably expect an increase of the DB2 work but how can you understand if and how much additional memory will be needed to support it ?

Unfortunately there is not a general and easy answer in this case.

You have to do an estimate based on your experience, knowledge of the environment and on additional information and tools which are workload specific.

[13] Keep in mind that logical swapped AS are subject to trimming so the MB/IN AS values have to be reduced.

5 Estimate memory needs (an alternative method)

An alternative method to estimate memory needs can be used; it is based on the following assumptions:

1. System resources (CPU, memory, I/O, etc.) have to be balanced in order to allow the system to perform optimally;
2. Ratios[14] between resources utilization change slowly over time depending on software and hardware evolution;
3. In the short period (12-18 month) these ratios can be considered as constant if the workload in the system doesn't dramatically change.

The idea is to track the ratio between memory and CPU utilization over time and use it to estimate memory needs.

We call this ratio MMC; it says how many MB of memory are you using for each used MIPS (including CPU, AAP and IIP); this ratio can be calculated for each system and for specific time periods or shifts.

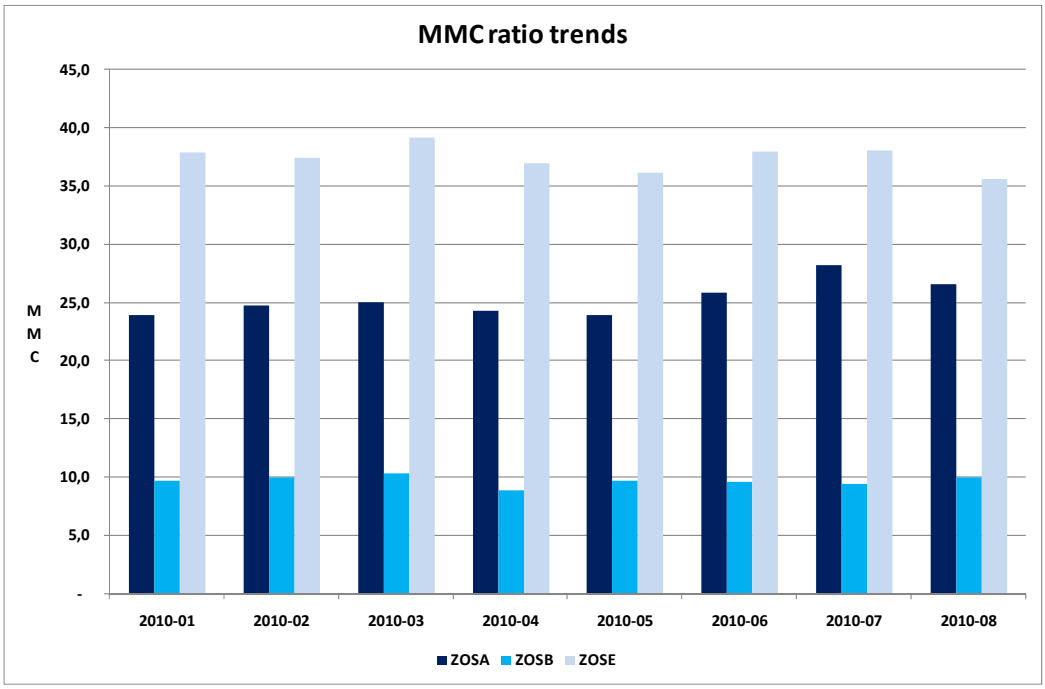

Figure 8

In Figure 8 the MMC ratios of three different kind of system are presented.

The first thing to note is that MMC values are very stable, for each specific system, over time[15].

[14] EPV for z/OS automatically tracks the utilization ratio between different resources over time. Each of these ratios forms an EPV index. EPV indexes are self adaptive, user specific, rules of thumb and can be a powerful tool for performance analysts if carefully investigated and interpreted.

[15] You should expect MMC to grow in the medium-long period because memory usage tends to grow faster than CPU usage.

The second note is that these systems look very different:

- ZOSA is a small but important online system running IMS and Websphere applications accessing DB2. It doesn't use a lot of CPU (500-700 MIPS) but its applications require a lot of memory.
- ZOSB is a much bigger system (using 10 times more CPU than ZOSA) running any type of workload but more TSO and batch oriented.
- ZOSE is a sand-box system; its CPU utilization is very low compared to the memory needed to run system and subsystems.

This is one of the issues of this method: MMC tends to present inflated values, affected by the minimum memory usage needed to run the z/OS operating system itself.
For this reason this method should not to be used to size sand-box systems such as ZOSE.

However it also has effects on the other systems; you can correct for it by excluding the memory used by the z/OS System (SYSTEM and SYSSTC service classes).
Applying this correction the ZOSA MMC drops from about 25 to 21 while the ZOSB MMC drops from about 10 to 9.

Now to estimate the memory needed in our capacity planning we only need to know the expected CPU growth for each system and multiply this value by the correspondent MMC.
Suppose we expect 150 MIPS growth for ZOSA and 800 MIPS growth for ZOSB we can use the following simple formulas:

```
ZOSA MB = 150 MIPS * 21 MB/MIPS = 3.150 MB
ZOSB MB = 800 MIPS *  9 MB/MIPS = 7.250 MB
```

Some important rules to remember when using this method are:

- Don't use MMC values calculated when memory is constrained; look at another period of time;
- Remember that MMC is very workload sensitive so it can dramatically change if you introduce a completely different workload in the system;
- MMC can be heavily impacted by anomalies in CPU usage (loops) or memory usage (memory leaks).

6 Growth due to technological evolution

Technological evolution is an additional key factor when estimating memory growth.
Every new system and subsystem release improves and extends the available Data In Memory techniques.
The best example is DB2: each new version gives you the possibility to use bigger and bigger buffer pools. Starting from DB2 8.1 you have also the possibility to fix buffer pools in memory in order to improve application performance.

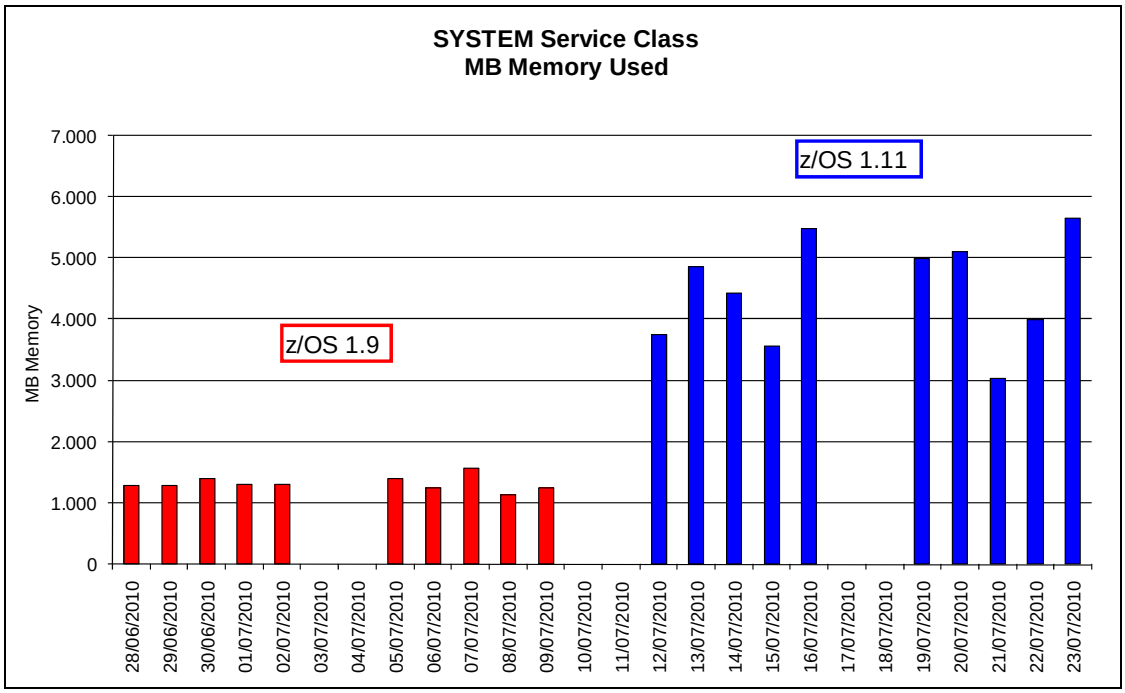

Figure 8

The graph in Figure 8 shows what happened at a customer site when migrating from z/OS 1.9 to z/OS 1.11.
The red bars show the memory used by the System service class the two weeks before the migration; the blue bars show what happened the two weeks after.
The memory increase (about 3 GB) is due to the SMSPDSE address space. The interesting thing is that they didn't change any SMS parameters.

Sometimes evolution reduces the memory needs. This happened with the Hardware Storage Area (HSA) which is mainly designed to host the definitions stored in the IOCDS.
Starting from the zSeries machines the HSA size has greatly increased at every new machine generation reaching a size of 2 GB and more with the z9 hardware. This was a big problem, especially for small customers, so IBM decided to include 16 GB memory[16] (apart from the memory bought from customers) to host the HSA in z10 and z196 machines.

As you can see the memory reduction is only apparent, but it has to be considered this way from the Capacity Planning perspective.

7 Conclusions

To forecast memory needs you have to:

- Evaluate the baseline;
- Apply the growth due to workload evolution;
- Apply the growth due to technological evolution.

In this paper we described a methodology to do that by using mainly standard SMF measurements.

[16] 8 GB on z10 BC machines.