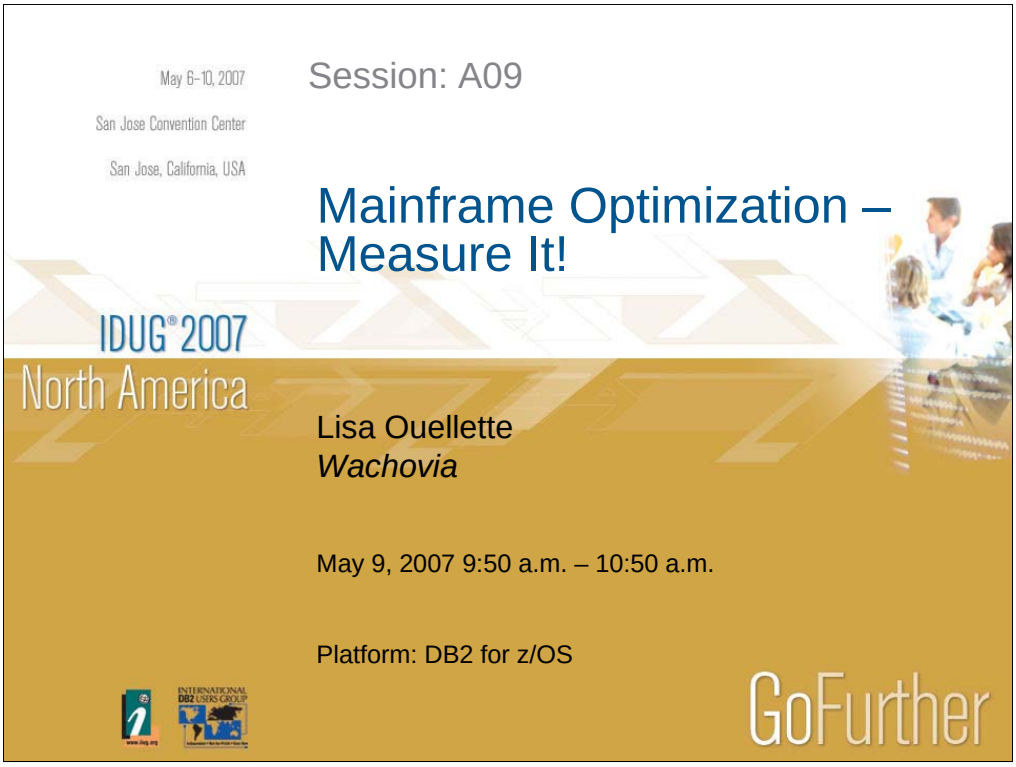


Reducing the capacity requirements of your applications can delay upgrades and save on hardware and software licensing charges. How do you measure success? How can you avoid some of the common mistakes made in benchmark measurements? DB2 accounting and statistics trace records might not tell the whole story. This presentation will take a look at various data sources for measuring the performance of an application. A detail walk through examples will be enlightening.

Agenda

- Introduction – why measure?
- Data sources
- Example #1 – Websphere and z/OS DB2
- Example #2 – stored procedures
- The list of mistakes to avoid when measuring

Goal is to provide:

•Inspiration to measure and become more aware of performance.

•Some guidance around what data is available beyond DB2 traces.

•A couple of examples of real measurement data from Wachovia.

•A list of mistakes to avoid (I have made).

Why Measure?

- Understand effects of changes in order to accurately predict the effects of changes in the future.
- Focus optimization efforts on 'biggest bang for the buck'.
- Instill a 'cost of performance' mentality across the IT organization which affects product, architecture and contract decisions.
- Celebrate and reward optimization efforts.
- Make sure the business understands the cost of implementing the business requirements.

If you measure the performance impact of adding an index to a table enough times, eventually you get a feel for the impact to insert / update / delete / load / reorganization. It becomes easier to make a 'gut' call about the worthiness of adding an index.

In addition, over time you develop techniques and history for estimating benefits and can prioritize efforts.

Measurements in 'pilot' or trial can help estimate cost of running software in production.

Celebration

- Contributors get something to put in their performance review:

Appl ID	Program/ Plan/ Job - Where - Impl Date	Change Description	Annualized Net Billing Avoidance	Percent Savings	Annualized Net HDW+SOFT Cost	Prime Time %	Zone 1 Mips Daily
SK	P8SK0010 PJX1 11/10/06	Improve account number wild carding in SQL call	$65,878	99.91%	$24,076	86.2%	9.67

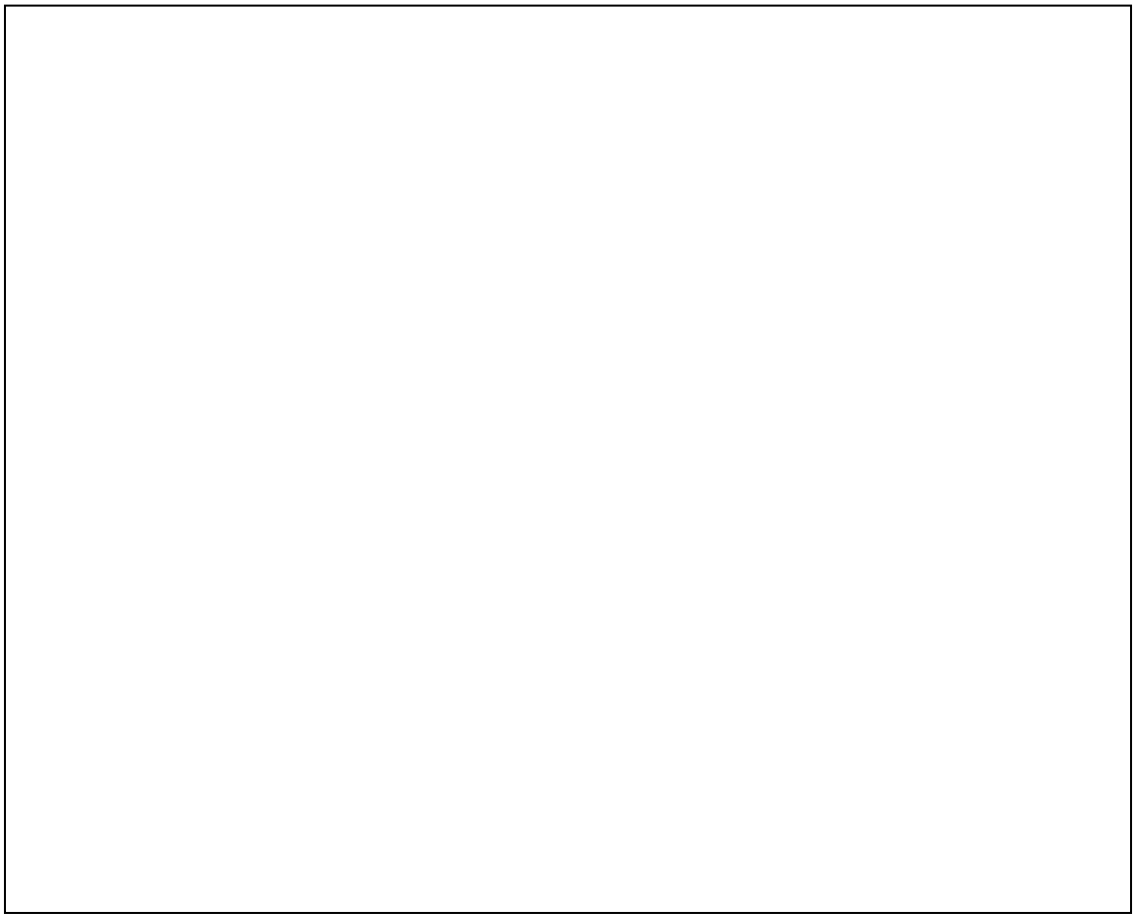
- Hardware and software cost avoidance is based on actual cost per mip.

Zone 1 is our terminology for business day 9AM-5PM. The CPU usage during these times is what drives our capacity requirements. So it is the CPU saved during these times that actually saves on hardware and software licensing costs long term. We do not bill any different for prime time and non-prime time CPU consumption.

Business Requirement?

- IMS screen required user to only enter first 8 digits of account number – wild card the end.
- Conversion to .Net / Stored procedure – User can enter any 8 digits of account number – resulting in scan of partition for all accounts in a state.
- All accounts moved to one 'bogus' state.
- Now stored procedure does table space scan (22 million rows):

  ACCT_NUM LIKE %keyeddata%%%%



Business Requirement?

	Average 158 CPU Seconds	Bill per transaction
IMS Screen	175	$0.14
Stored Proc - by state	3710	$2.87
Stored Proc - one state	13665	$10.55
Stored Proc - no wildcard	1.79	$0.001

Only about 200 inquiries per day. So encouragement to correct the problem after the one state implementation was "Would you key full account numbers if it saved you over $2000 a day?"

So, what do you count as savings? In this example, we counted the difference between the 'Stored procedure – by state' implementation which had been running for over a year to the 'Stored procedure – no wildcard' implementation to calculate savings. We call events like the one state implementation, "Critical Incidents". We still report on the incidents in $ terms to help people understand the impact, but do not count the savings since we never planned capacity for these events.

Normalized (158) CPU time

- Common industry z/OS capacity planning practice is to 'normalize' CPU time.
- Bill consistently
- Track and plan for ‘true’ application and workload growth over time.
- Determine if a set of applications will ‘fit’ on a new model processor.

Wachovia normalizes to an old 158 model CPU because it is conveniently about a 1 mip machine. We often have a mix of CPU models on the floor within a single sysplex as we progress towards newer technology. For example, today LPAR A is running on a 2084 and LPAR B is running on a 2094. A 2094 engine is about 35% faster than a 2084 engine. So a transaction that runs on LPAR B uses about 35% less time on the CPU than a transaction running on LPAR A. We don’t want to bill differently for the same transaction based on a workload managed lpar selection.

Let's say the 'Loan' application has grown at 12% per year for the last 10 years in terms of CPU consumption. Unless we hear different from the 'Loan' department, we plan on 12% growth for next year. If we didn't normalize the CPU we wouldn't have any way to really track their growth since when we installed new technology it would appear as if the application had 'shrunk' - less time spent on the CPU.

‘Normalizing’ isn't perfect - it works from an overall perspective in terms of capacity planning but an individual transaction or job might not normalize perfectly. In this presentation, some of the numbers have been ‘normalized’ and will be identified as ‘158’ time. Others are not.

Data Sources

- RMF Workload activity data (WLM service class and report class – SMF type 72)
- Address space data (SMF type 30)
- DB2 accounting trace (SMF type 101)
- CICS (SMF type 110)
- Websphere (SMF type 120)

SMF (System Management Facilities) is a mechanism the z/OS system uses to collect and record system and job-related information. Program products, such as DB2, CICS, and Websphere, can also use SMF to record information.

RMF (Resource Measurement Facility) can write records to SMF for later reporting. RMF records data about z/OS system performance.

In general, program products write one or two types of data to SMF:

•Interval data that contains information about what has happened in the last interval. A product can define their own interval or use the default. For example, DB2 writes statistics SMF data for an interval.

•Activity data that contains information about one activity. For example, DB2 writes accounting trace data to SMF for each thread.

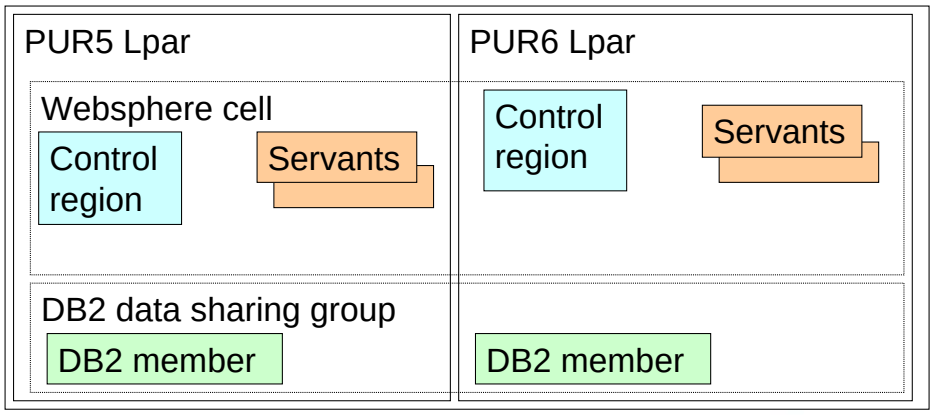
Typical RMF intervals are 5-15 minutes. Typical SMF intervals are 30 minutes.

There are a myriad of ways to specify and override turning on and off SMF recording and interval lengths. Ask the owner of the program product at your company.

Data Sources - Reporting

- MICS – Computer Associates' product used by Wachovia to read SMF data and create a SAS performance database.
- MXG – Merrill consultants' product that reads SMF data and creates a SAS performance database.
- RMF reports.
- DB2 performance expert or a plethora of others – see exhibitors.

Example #1 Websphere and DB2



10

Type of questions to be answered:

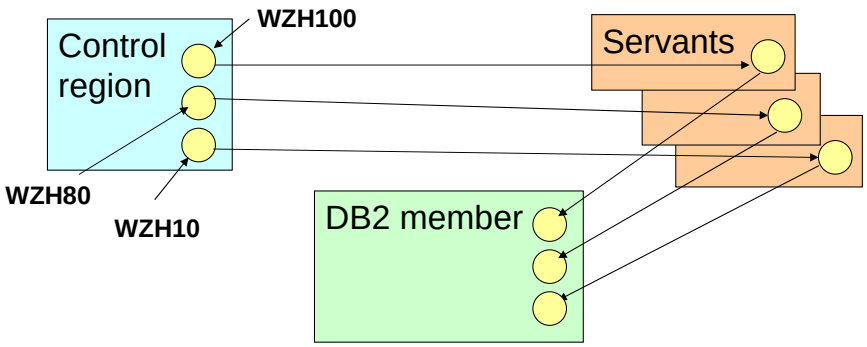
How much capacity will this new application require?

How much will run on a zAAP- zseries application assist processor (special engine that only runs java)?

And others, for example, how much does it 'cost' to receive and send encrypted messages – SSL encryption / decryption.

Application normally requests CICS and IMS transactions on other lpars / sysplexes and uses the local DB2 only for some low volume services. However, when an IMS or CICS does not respond, the local DB2 is used for 'stand-in'. How does this change the CPU capacity requirements?

Example #1 Enclaves



Customer type web service are classified to report class WZH100.

Account type web services are classified to report class WZH80.

Miscellaneous other web services are classified to report class WZH10.

Web service 'transaction' received from TCP/IP into Websphere control region. Transaction is classified by Websphere and then workload manager, enclave is created and queued to a servant region for processing. The web service does DB2 SQL calls (local).

An enclave is an entity that encapsulates the execution units (TCBs and SRBs) that execute programs on behalf of the same work request.

You can also think of an enclave as a unit of work without address space boundaries.

These are 'independent' enclaves – they have no association with a particular address space from a priority and account point of view.

Data Sources – SMF Type 72

- Interval data collected and written by RMF for Workload Manager service class and report classes (subtype 3).
- Contains the resources consumed in the previous RMF interval.
- Identifying work of interest and putting it in its own report class can make measurement easier.

Service class defines how workload manager should manage the work. Report class can subdivide a service class for reporting purposes. A report class can be defined to multiple service classes but Wachovia doesn't do this because it is confuses us.

Befriend your Workload Manager. Explain what you are trying to measure and accomplish.

Resources include CPU time, I/O, storage.

Capture ratio – how much of the CPU time consumed is actually captured and recorded.

RMF Workload Activity Report

REPORT BY: POLICY=URPPLEX REPORT CLASS=WZH100

TRANSACTIONS		TRANS.-TIME	HHH.MM.SS.TTT	--DASD I/O--	
AVG	0.95	ACTUAL	115	SSCHRT	0.1
MPL	0.95	EXECUTION	114	RESP	1.1
ENDED	714866	QUEUED	0	CONN	0.3
END/S	8.27	R/S AFFINITY	0	DISC	0.6
#SWAPS	0	INELIGIBLE	0	Q+PEND	0.2
EXCTD	0	CONVERSION	0	IOSQ	0.0
AVG ENC	0.95	STD DEV	330		
REM ENC	0.00				
MS ENC	0.00				

---SERVICE----		--SERVICE TIMES--		PAGE-IN RATES		----STORAGE----	
IOC	0	TCB	6753.7	SINGLE	0.0	AVG	0.00
CPU	140154K	SRB	0.0	BLOCK	0.0	TOTAL	0.00
MSO	0	RCT	0.0	SHARED	0.0	CENTRAL	0.00
SRB	0	IIT	0.0	HSP	0.0	EXPAND	0.00
TOT	140154K	HST	0.0	HSP MISS	0.0		
/SEC	1622	IFA	5729.1	EXP SNGL	0.0	SHARED	0.00
		APPL% CP	1.2	EXP BLK	0.0		
ABSRPTN	1707	APPL% IFACP	0.1	EXP SHR	0.0		
TRX SERV	1707	APPL% IFA	6.6				

Example of report for 24 hour period for one of the Websphere transaction report classes – this one is for report class WZH100.

DINTV(2400)

SYSRPTS(WLMGL(RCLASS,SYSNAM(PUR5)))

This is typical of a transaction report class workload activity report.

We'll look at two sections – transaction volume and service times.

Websphere transaction report classes for this application are WZH100, WZH80 and WZH10.

RMF Workload Activity Report

WZH100 Report Class

TRANSACTIONS	
AVG	0.95
MPL	0.95
ENDED	714866
END/S	8.27
#SWAPS	0
EXCTD	0
AVG ENC	0.95

--SERVICE TIMES--	
TCB	6753.7
SRB	0.0
RCT	0.0
IIT	0.0
HST	0.0
IFA	5729.1
APPL% CP	1.2
APPL% IFACP	0.1
APPL% IFA	6.6

14

AVG = Average Active Transactions (sum of all transaction active time divided by the interval time ….) For address space type report classes this is the number of address spaces.

MPL = Average Resident Transactions (in storage – not paged out)

ENDED = # of transactions that ended during the interval.

END/S = Ended per second

AVG ENC = Average number of independent enclaves (active).

Service times are in seconds. TCB is total TCB. IFA is seconds on zAAP processor (included in TCB).

The % fields you can think of as

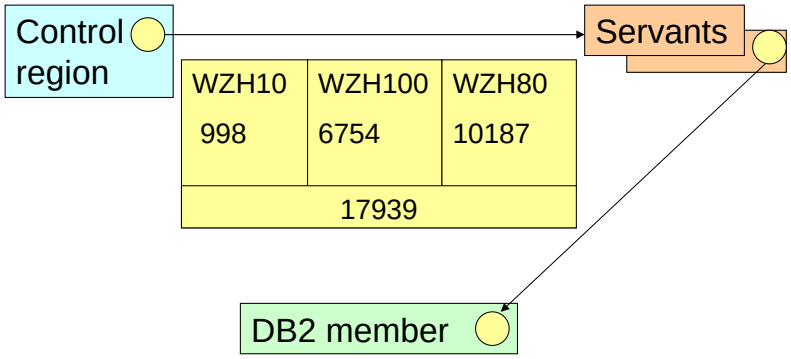
% of one engine that my application is using on:

CP = general processor

IFACP = could have run on zAAP (IFA) but didn't

IFA - zAAP

Example #1 RMF enclave CPU



15

Type 72 – RMF Workload Activity Report TCB times for the three report class enclave transactions. The previous RMF Workload Activity report example was for the WZH100 report class.

Includes CPU time spent in the control region, servant region and DB2 by the enclave.

RMF Workload Activity Report

REPORT BY: POLICY=URPPLEX REPORT CLASS=WAS_SVNT

TRANSACTIONS		TRANS.-TIME HHH.MM.SS.TTT		--DASD I/O--	
AVG	8.00	ACTUAL	0	SSCHRT	1.0
MPL	8.00	EXECUTION	0	RESP	3.3
ENDED	0	QUEUED	0	CONN	0.2
END/S	0.00	R/S AFFINITY	0	DISC	2.8
#SWAPS	0	INELIGIBLE	0	Q+PEND	0.2
EXCTD	0	CONVERSION	0	IOSQ	0.0
AVG ENC	0.00	STD DEV	0		
REM ENC	0.00				
MS ENC	0.00				

---SERVICE----		--SERVICE TIMES--		PAGE-IN RATES		----STORAGE----	
IOC	31919K	TCB	3504.2	SINGLE	0.0	AVG	114303
CPU	72721K	SRB	118.3	BLOCK	0.0	TOTAL	914422
MSO	0	RCT	0.0	SHARED	0.0	CENTRAL	914422
SRB	2454K	IIT	0.0	HSP	0.0	EXPAND	0.00
TOT	107094K	HST	0.0	HSP MISS	0.0		
/SEC	1240	IFA	2405.8	EXP SNGL	0.0	SHARED	1961.00
		APPL% CP	1.4	EXP BLK	0.0		
ABSRPTN	155	APPL% IFACP	0.6	EXP SHR	0.0		
TRX SERV	155	APPL% IFA	2.8				

--SERVICE TIMES--	
TCB	3504.2
SRB	118.3
Total	3622.5

16

GoFurther

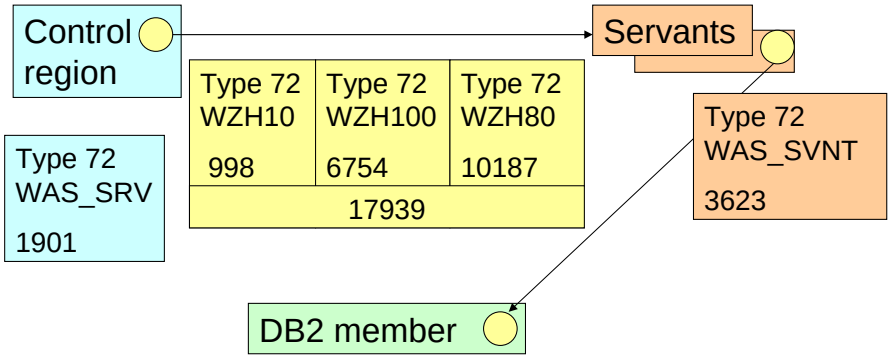
Control Region report classes are WAS_SRV.

Servant region report classes are WAS_SVNT.

This example is for the servant regions.

So on PUR5 we had 8 Websphere servant address spaces active on average for the 24 hour period. Notice no enclaves, no response time section but you do get paging and storage (memory) usage information.

Example #1 'Overhead' CPU



Type 72 – Workload Activity

Control Region (WAS_SRV) – Communications, transaction classification.

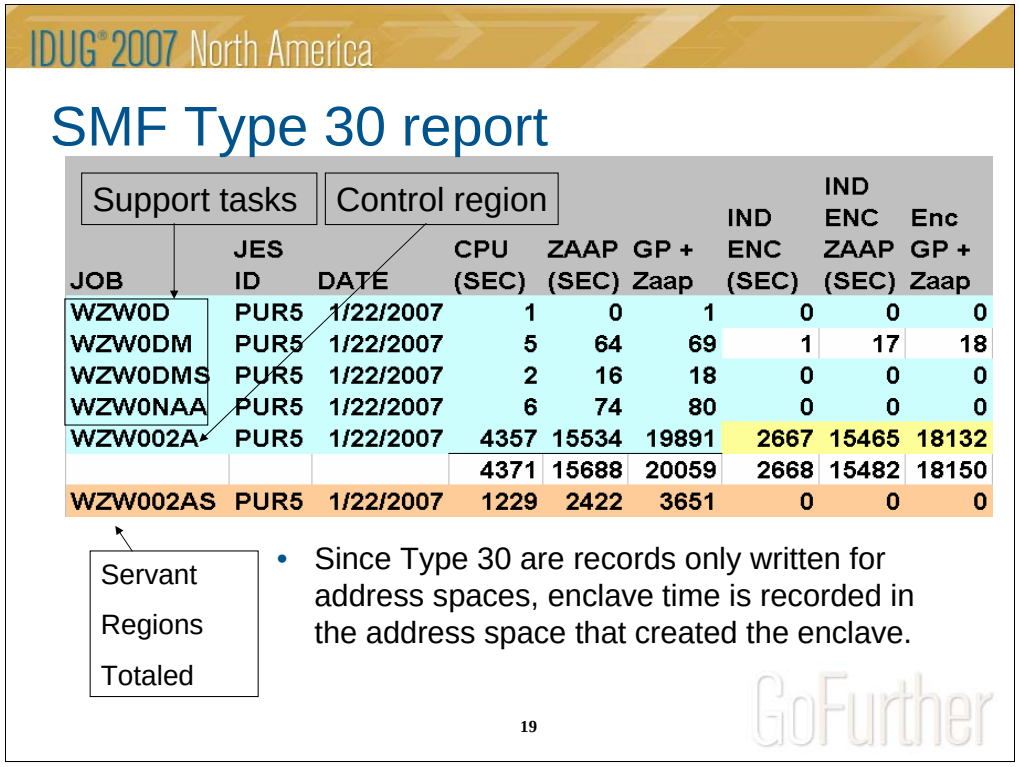
Servant region (WAS_SVNT) – Work selection, java garbage collection, spawned user threads

Websphere transactions (WZH100, WZH80 and WZH10) – Web service application code – includes DB2 time.

So 'overhead' is (1901+3623)/(1901+17939+3623) ~ 24%

Data Sources – SMF Type 30

- Records written by z/OS at every SMF interval and job step termination for every address space.
- A good alternative if you can't get your own report class.
- Also contains information about consumption of CPU, I/O and storage.



This sample is data extracted from our MICS performance database that is summarized to a day and put in a spreadsheet.

'CPU (SEC)' = SMF30CPT (tcb time) + SMF30CPS (Srb time) = Amount of time on general purpose processors

'ZAAP (SEC)' = SMF30_TIME_ON_IFA = Amount of time spent on zAAP processors

'IND ENC ZAAP (SEC)' = SMF30_ENCLAVE_TIME_ON_IFA = Amount of independent enclave time spent on zAAP processor

'IND ENC (SEC)' = SMF30ENC = Amount of independent enclave time spent on general purpose processors

Enclave time is already included in total CPU time fields, so for example, of a TOTAL 4357 general purpose processor seconds on WZW002A, 2667 of them were consumed in independent enclaves.

Blue tasks (top 5 lines) are the ones in WAS_SRV report class. Only WZW002AS (bottom line) tasks are in the WAS_SVNT report class.

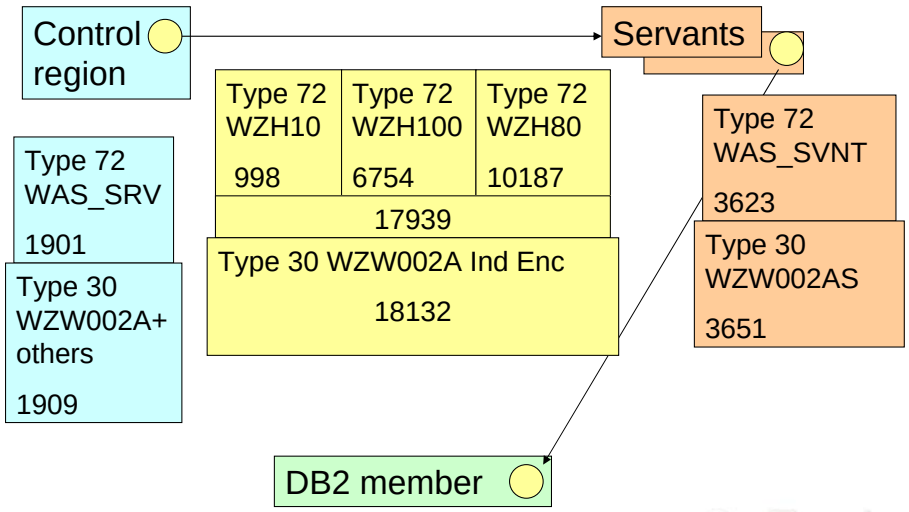
So non-enclave time for WAS_SRV address spaces is:

GP(General purpose processor): 4371-2668=1703

zAAP: 15688-15482=206

GP+zAAP: 1703+206=1909

Example #1 Type 30 CPU



Reality check / double check. Do my SMF Type 30 records add up to what I expect for the RMF Workload Activity (Type 72) report classes they fall into – or have I missed something?

Does the enclave time reported in Type 30 come close to the RMF Workload Activity report classes for those enclaves?

What is my overhead?

From a RMF Type 72 record perspective it is 5524 CPU seconds.

From an SMF Type 30 perspective it is 1909+3651=5560 CPU seconds.

From a percentage view:

5524/(5524+17939)=24%

5560/(5560+18132)=23%

So for example, this overhead would be reduced slightly if we were not doing encryption / decryption.

Data Sources – SMF Type 120

- z/OS Websphere application server SMF records.
- 'Activity' records too voluminous (subtypes 1,5 and 7).
- Subtype 3 contains information about communications (number and size) as well as heap sizes.
- Subtype 8 contains information about Web containers – servlets.
- Subtype 6 contains information about J2EE containers – beans.

SMF Type 120 Report

DATE	END TIME	SERVER	SERVLT NAME	NBR REQS	Avg CPU seconds	Min CPU seconds	Max CPU seconds
22-Jan-07	16:47	WZW002	RelationshipInquiry	3738	0.008855	0.003897	0.429658
22-Jan-07	16:52	WZW002	RelationshipInquiry	3757	0.008895	0.003852	0.218446
22-Jan-07	16:57	WZW002	RelationshipInquiry	3833	0.008557	0.003829	0.192392

- Multiply for each interval the number of requests by the average CPU time to get total CPU time for the interval.
- THEN add the totals to get total CPU time for multiple intervals and servlets (ie. Beware of averaging averages).

From MXG and put into spreadsheet

This is from subtype 8 ‘Web Container Interval’ record which breaks measurements out by servlet.

Server = SM120WIC

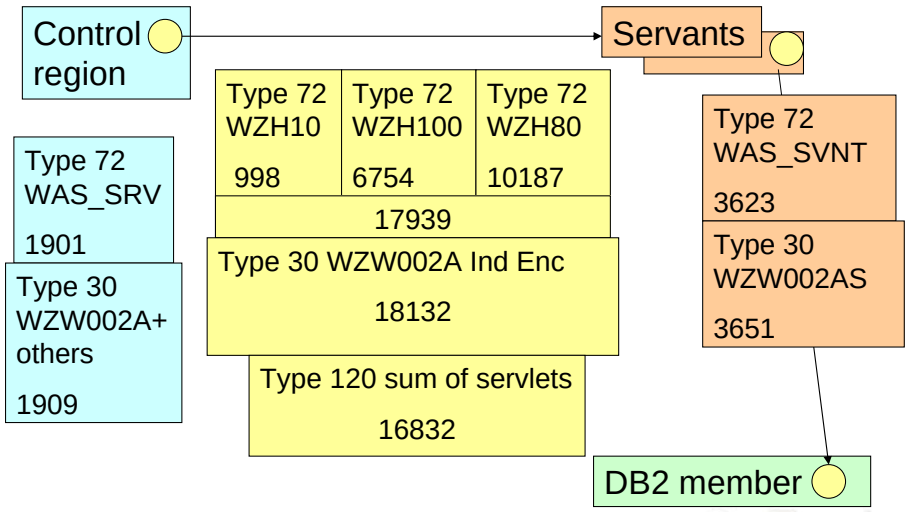
Application = SMF120WIQ

Number of Requests = SMF120WIX

Average, Minimum, Maximum CPU time = SM120WJ4, SM120WJ5, SM120WJ6

“The CPU expense of the garbage collection might be skewing the SM120WJ6 (servlet MAX cpu time) indirectly. A servlet in an interval that is making a method call while GC is occurring could in turn have longer CPU time, especially if is cleaning up instances of classes that are loaded and/or used by the web container. To reiterate, JVM garbage collection is not explicitly computed for SMF 120 records but could possibly skew SM120WJ6.”

Example #1 Type 120 CPU



23

Type 120 is ‘capturing’ less CPU than RMF workload activity (Type 72) or Type 30.

16832/18132 ~ 93% of Type 30.

But in the future you could estimate Type 30 data if you know this ratio.

Data Sources – SMF Type 101

- DB2 Accounting Record written via DB2 accounting trace.
- Contains many metrics about thread consumption – CPU, I/O and buffers.
- CPU time is broken out by:
  - 'Class 1' – application create thread to termination.
  - 'Class 2' – CPU used by DB2 on your behalf.

'Never ending' threads

Affect of 'inactive' thread Zparm

'Rollup' accounting

Plan versus package accounting

Use of commit count instead of thread count for averages might be appropriate

SMF Type 101 report

PRIMAUTH: SAVY04P PLANNAME: ?RRSAF

AVERAGE	APPL(CL.1)	DB2 (CL.2)
ELAPSED TIME	8:02.12947	1.254581
NONNESTED	8:02.12947	1.254581
STORED PROC	0.000000	0.000000
UDF	0.000000	0.000000
TRIGGER	0.000000	0.000000
CP CPU TIME	0.433943	0.365214
AGENT	0.433943	0.365214
NONNESTED	0.433943	0.365214

HIGHLIGHTS

#OCCURRENCES :	407
#ALLIEDS :	407
#ALLIEDS DISTRIB:	0
#DBATS :	0
#DBATS DISTRIB. :	0
#NO PROGRAM DATA:	0
#NORMAL TERMINAT:	407
#ABNORMAL TERMIN:	0
#CP/X PARALLEL. :	0
#IO PARALLELISM :	0
#INCREMENT. BIND:	0
#COMMITS :	119493

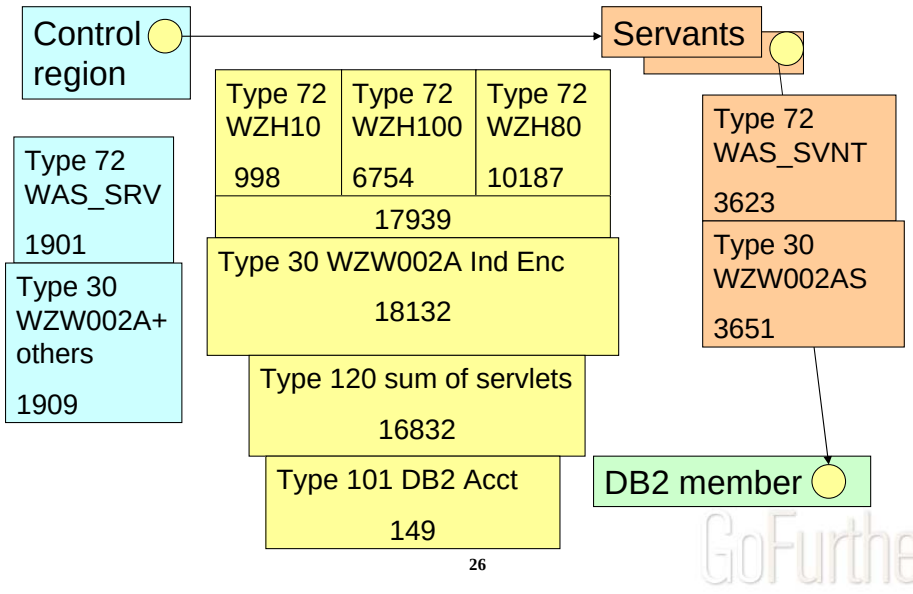
25

GoFurther

Sample DB2 Performance Export report from DB2 accounting trace records.

Averages are per ‘occurrence’ so to get total DB2 CPU time you can multiply average Class 2 CPU time (.365214) by 407 = 149 seconds.

Example #1 Type 101 CPU



26

Warning – NOT TO SCALE!

The 149 is included in type 72 17939 seconds, the type 30 18132 CPU seconds, and the type 120 16832 CPU seconds.

Also, remember only threads that have ENDED in the time interval are counted.

What is interesting is that in stand-in the DB2 Acct CPU time goes up. But the others remain about the same. My explanation is that the formatting of the CICS/IMS transactions and the parsing of the results is as much CPU time as the DB2 work that replaces it.

Data Sources – SMF Type 110

- Produced by CICS Monitoring facility (subtype 1)
- Performance class data (class 3) provided information about the resource usage of each transaction.
- CPU time INCLUDES DB2 time.
- Data can be match to DB2 accounting records – correlation ID contains transaction code.
- SMF Type 72 INCLUDES Type 110 CPU time.

The CORRELATION fields shows the 12-byte thread correlation ID which is made up as eeeetttttnnnn where eeee is either COMD, POOL or ENTR indicating whether it is a command, pool or DB2ENTRY thread; tttt is the transid, and nnnn is a unique number.

Example:
CORRELATION-ID=POOL**KHVA**0002
CONNECTION-ID=CICSP101

SMF Type 110 + 101 Report

```
------------DATE=31JAN ... DB2ID=DP52 -----------------

                 TOT   AVG  AVG   AVG   DB2   AVG    AVG
TRAN DB2         TRN   RSP  CIC   SIO   TRD   CL1    CL2
 ID  PLAN        CNT   TIM  CPU   CNT   CNT   TCB    TCB

KHJ3 CHJ130P     362   3.4  2.212  0.0   344   2.330 2.067
HC01 HCTICINQ 69600    0.1  0.015  4.0 69600   0.010 0.010
```


'TRAN ID' = CMF TRAN:Field ID 1 matched to bytes 5-8 of DB2 correlation id (QWHCCV)

'DB2 PLAN' = DB2 plan from DB2 accounting (QWHCPLAN)

'TOT TRN CNT' – Total # of CMF subtype 1 records for this transaction code

'AVG RSP TIM' – Total of (STOP:Field ID 6 - START:Field ID 5) / 'TOT TRN CNT'

CPU Time from CMF data = USRCPUT(Task Processor Time): Field ID 8 + RLSCPUT(SRB CPU time for RLS): Field ID 175

'AVG CIC CPU' = CPU Time from CMF data / 'TOT TRN CNT'

'AVG SIO CNT' = FCAMCT(File access method requests):Field ID 70 / 'TOT TRN CNT'

'DB2 TRD CNT' = DB2 thread count – count of accounting records

'AVG CL1 TCB' = Average Class 1 CPU time = End TCB Timer:QWACEJST – Beginning TCB Timer:QWACBJST

'AVG CL2 TCB' = Average Class 2 CPU time = DB2 CPU time:QWACAJST

Example #2 Stored procedures

- In general, where is CPU time accounted?
- Consider z/OS to z/OS distributed DB2 calls – stored procedure overhead versus distributed 'network call' overhead.

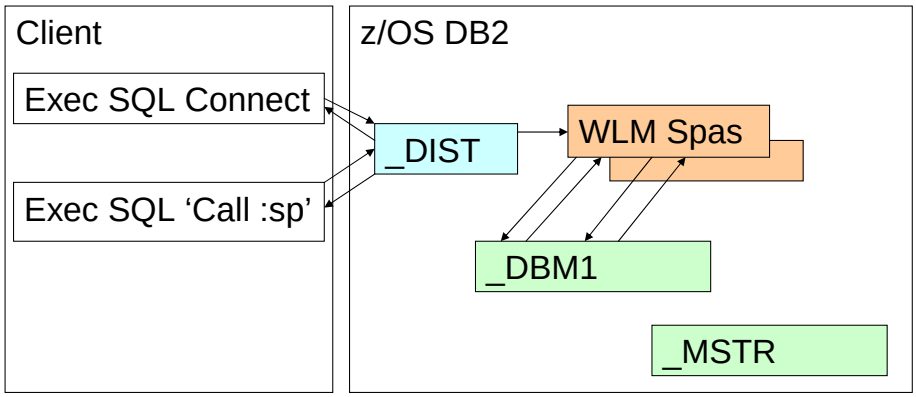
A stored procedure is a compiled program that can contain SQL statements.

A client application program uses a stored procedure by using the SQL Call <stored procedure> statement.

Stored procedures are used to reduce network traffic by encapsulating multiple SQL statements which can be executed with only one send and receive operation from a client.

Wachovia does a lot of z/OS to z/OS distributed DB2 work between three sysplexes that all run at the same physical location.

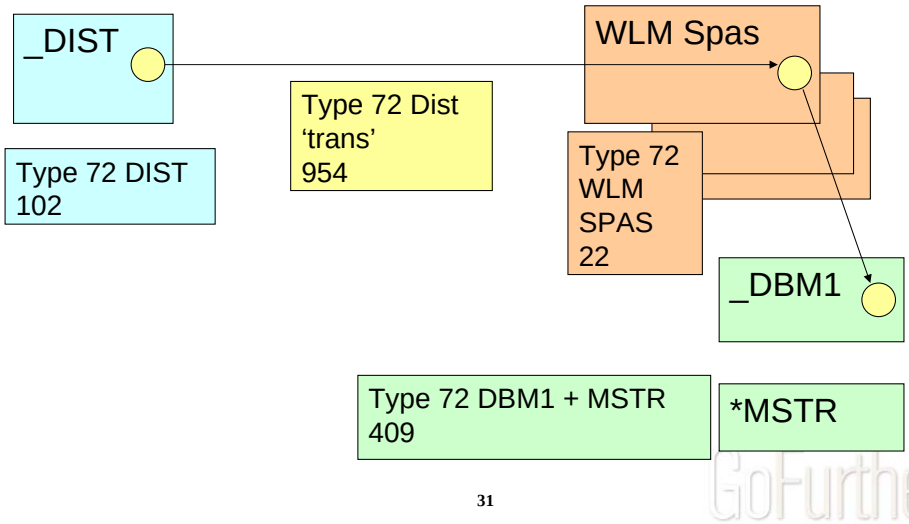
Example #2 Stored procedures



30

- SQL CONNECT – DB2 creates a thread.
- SQL CALL storedproc…… (:inparm1, :inparm2, :outparm1, :outparm2)
- DB2 looks in the catalog to find the procedure and get the name of the load module.
- DB2 tells Workload Manager (WLM) to queue the request to be executed in a stored procedure address space.
- Stored procedure address space loads module and executes it. Stored procedure 'fills in' output parameters and completes.
- DB2 copies output parameters to the client application parameter area and passes back to client.
- Each address space can execute more than one stored procedure at a time based on a parameter (NUMTCB).
- Workload Manager (WLM) will start as many stored procedure address spaces as necessary to process the work.
- Fixed overhead in scheduling of stored procedure in another address space. For a stored procedure that executes very quickly (does few, well-performing SQL statements) the fixed overhead can be a high percentage of the total processing.
- With a high-volume of executions the overhead can be significant.

Example #2 Type 72 (RMF) CPU



158 CPU Hours (ie. Normalized numbers in this example)

RMF Workload Activity (Type 72):

- _DIST(DDF work) – creates enclave
- _DBM1(buffers and I/O services) and _MSTR(system services) address spaces
- Stored Procedure address space(WLM SPAS)
- Distributed transactions (many service / report classes) – includes DB2 time

In our case several of the 'transaction' report classes are used in multiple DB2 systems so you can't cross-foot Type 72 data to Type 30 data.

Example #2 Type 30 report

Type 30 Data

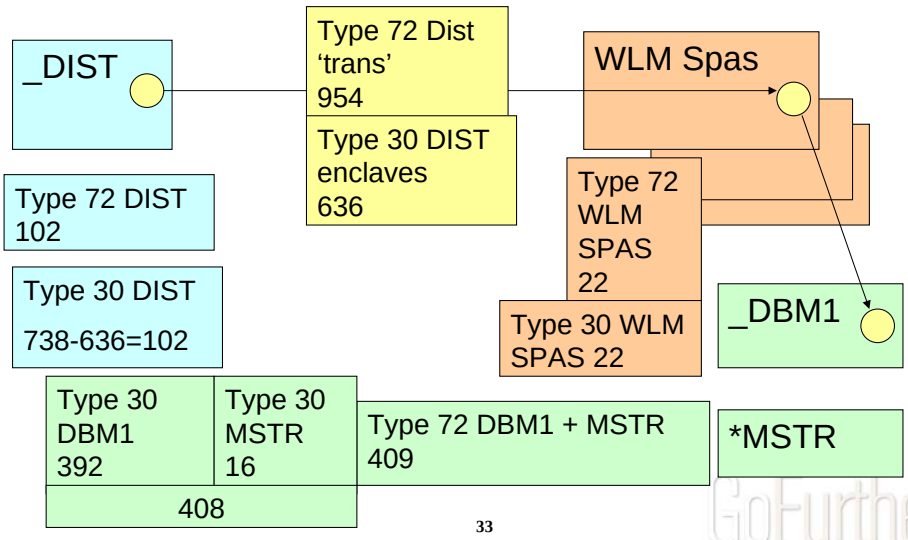
JOB	JES ID	DATE	TCB (158 HRS)	SRB (158 HRS)	CPU (158 HRS)	ENC (158 HRS)
DP71MSTR	PJX1	2/1/2007	6	10	16	0
DP71DBM1	PJX1	2/1/2007	5	387	392	0
					408	
DB2WLMP1	PJX1	2/1/2007	19	3	22	0
DP71DIST	PJX1	2/1/2007	717	21	738	636

32

Type 30:

- DIST address space – includes DB2 time
- DBM1 address space – mostly asynchronous I/O
- MSTR address space – control functions
- DB2WLMP1 tasks - Workload Managed Stored Procedure Address Spaces – environment initialization

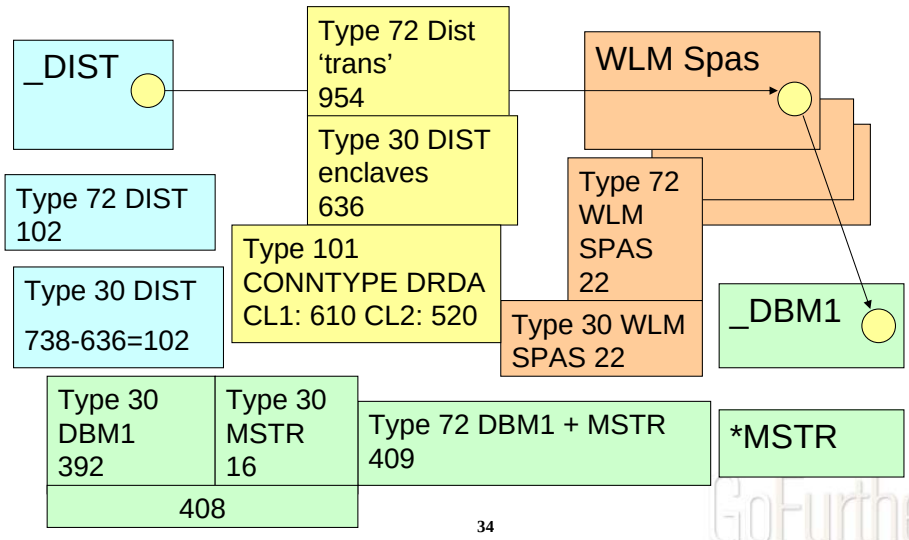
Example #2 Type 30 CPU



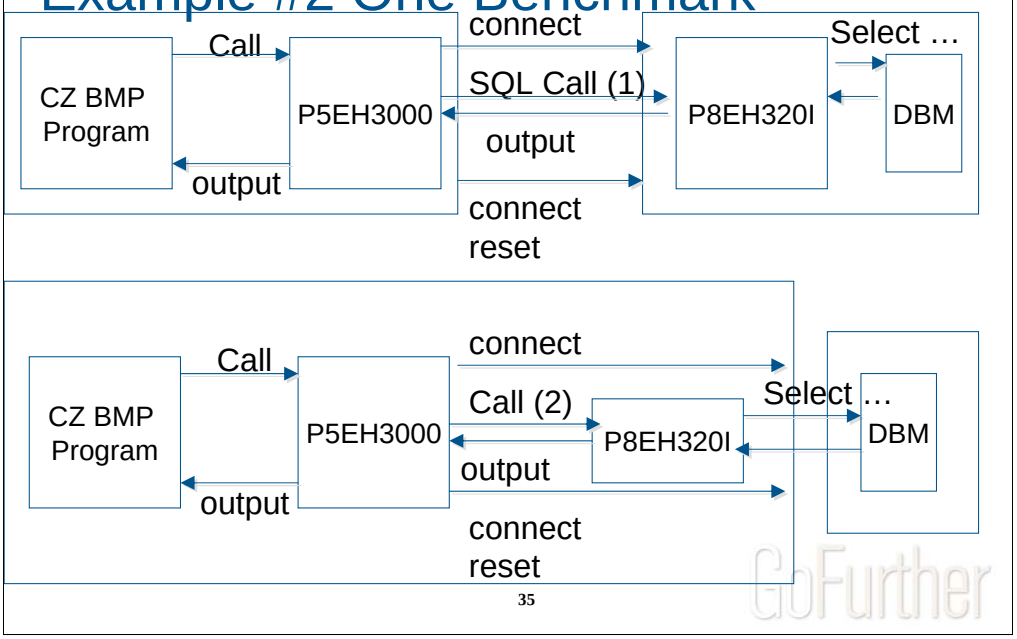
33

Note that the Type 72 'trans' classes contain several classes that can execute on another DB2 system …. This is why it is 'more' than the Distributed Address Space Type 30 records.

Example #2 Type 101 CPU



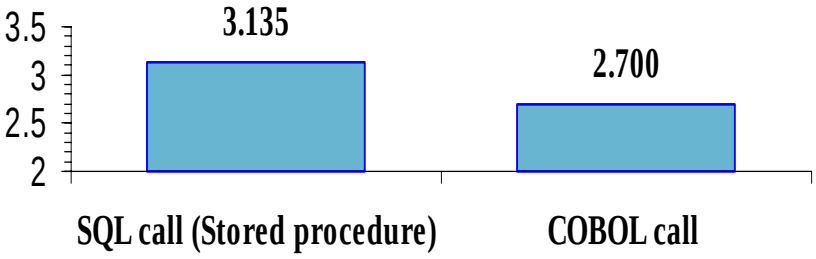
Example #2 One Benchmark



Question to be answered ….

Is the stored procedure overhead more or less than distributed overhead for z/OS to z/OS distributed environment?

Example #2 Distributed select



- Stored procedure did on the average 9 singleton select calls on behalf of the caller.
- Results were not similar for cursor processing (next slide).

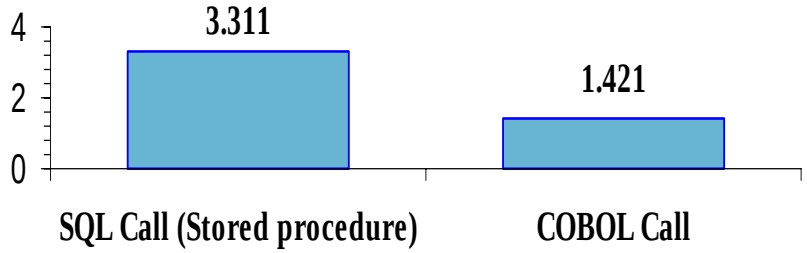
158 CPU seconds

Got solo use of test DB2 system.

Used type 30 records.

Included Job, Stored Procedure address space (already started), DIST (requester and server)

Example #2 Distributed cursor

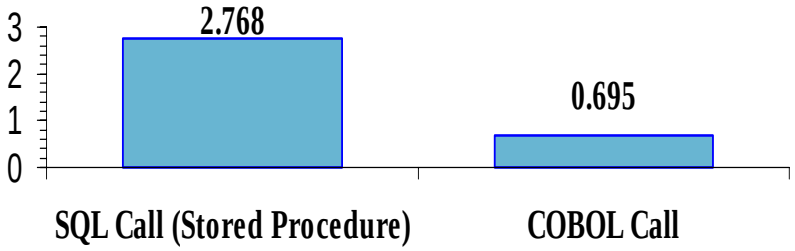


- Stored procedure did one open, 1-10 fetches, and one close of a cursor.
- Shows trips through distributed address space were fewer for one cursor than 9 singleton selects (block fetch).

158 CPU seconds

Jury is out on best technique. BUT exercise also provided data that was used to justify changing our program implementation process to implement programs both as stored procedures and subroutines (where possible) so that local calls were just subroutine calls instead of stored procedure calls.

Example #2 Local select

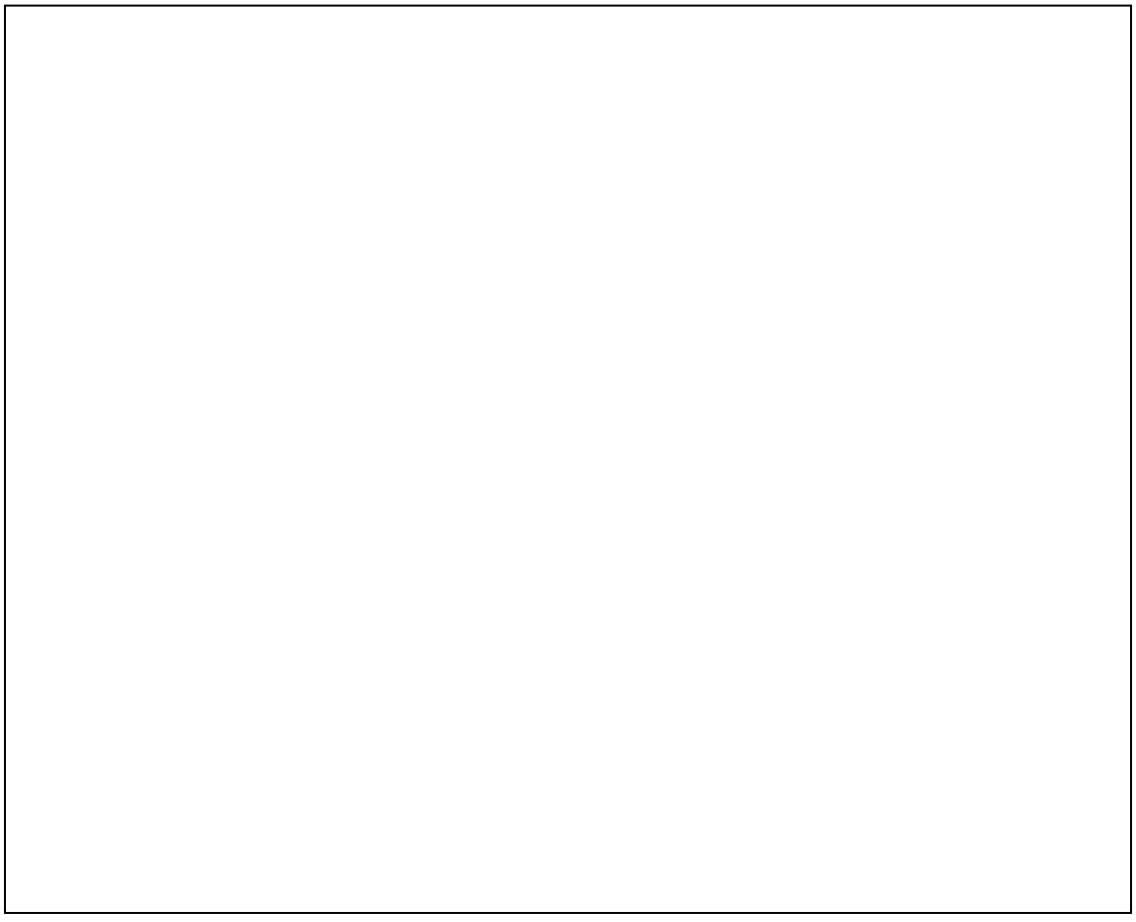


- Stored procedure did on the average 9 singleton select calls on behalf of the caller.
- Similar results for one open, 1-10 fetches, and one close of a cursor.

158 CPU seconds

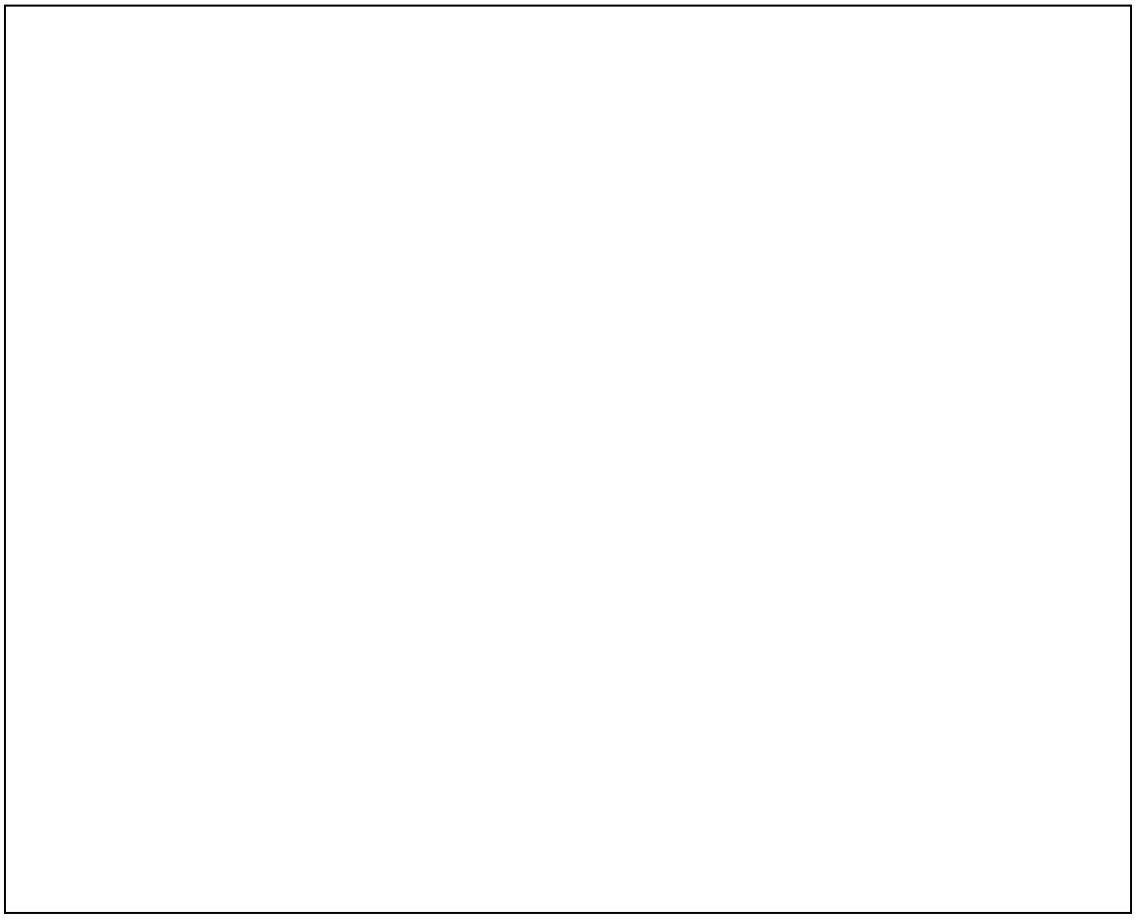
Mistake to avoid #1

- Don’t identify all jobs / transactions / address spaces involved and get measurements.
- BAD example:
  - Application changed from using batch ‘data mover’ job to move data off-host to using MQ series in order to save CPU.
  - Measured batch job CPU but forgot to measure MQ series address space CPU.



Mistake to avoid #2

- Run the test you want to be the better performer first.
- Once you run the test, your data is in the disk cache, buffer pools, etc.
- Immediate re-run will be much faster. It takes CPU to do I/O.
- Best to run a 'prime the system' job and then run tests back to back so none pay I/O penalty (unless of course you are trying to change I/O).

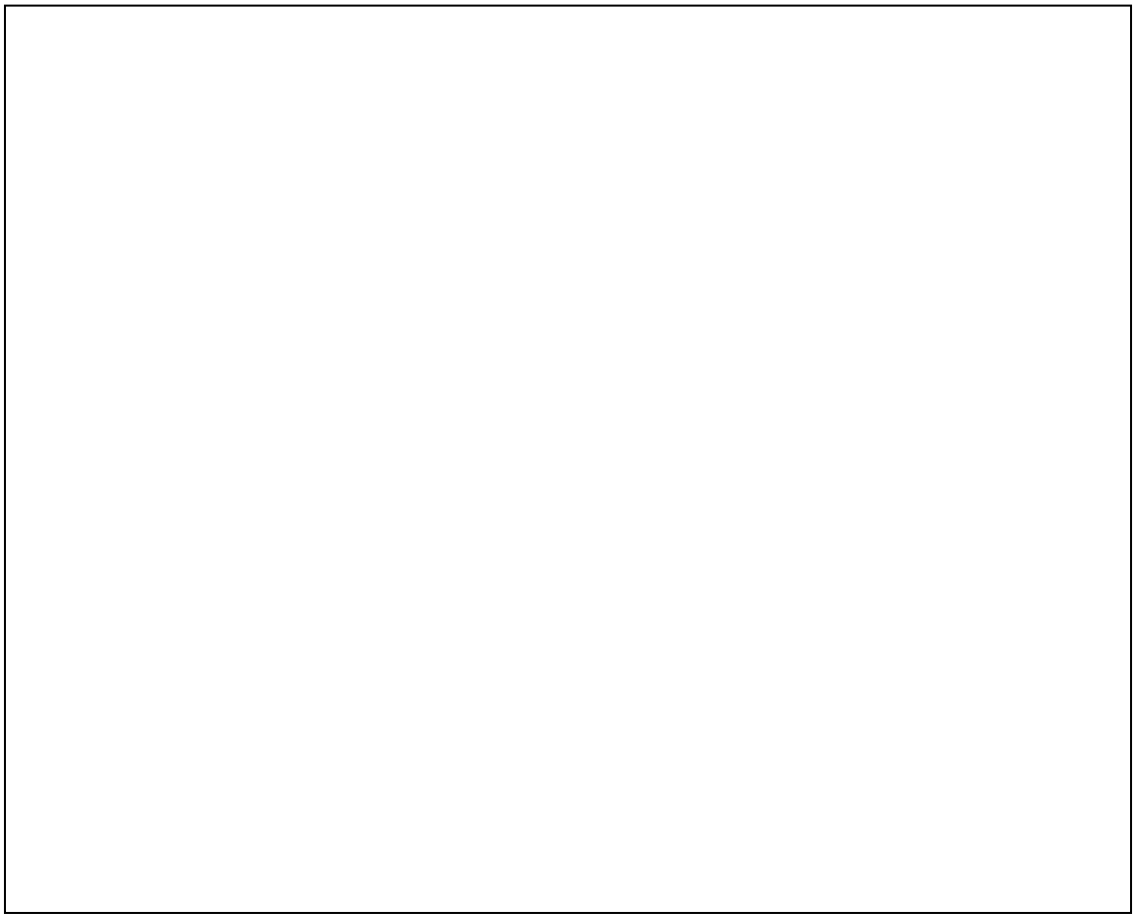


Mistake to avoid #3

- Run the tests / benchmarks on different CPUs … or different DB2 groups / members.
- We often have a mix of CPU models even in our development environment.
- With many programs, not only the model of CPU will make a difference but the size of the lpar in terms of available engines and memory.
- Watch out for the test last week was prior to an upgrade of some kind! The test this week was after an upgrade.

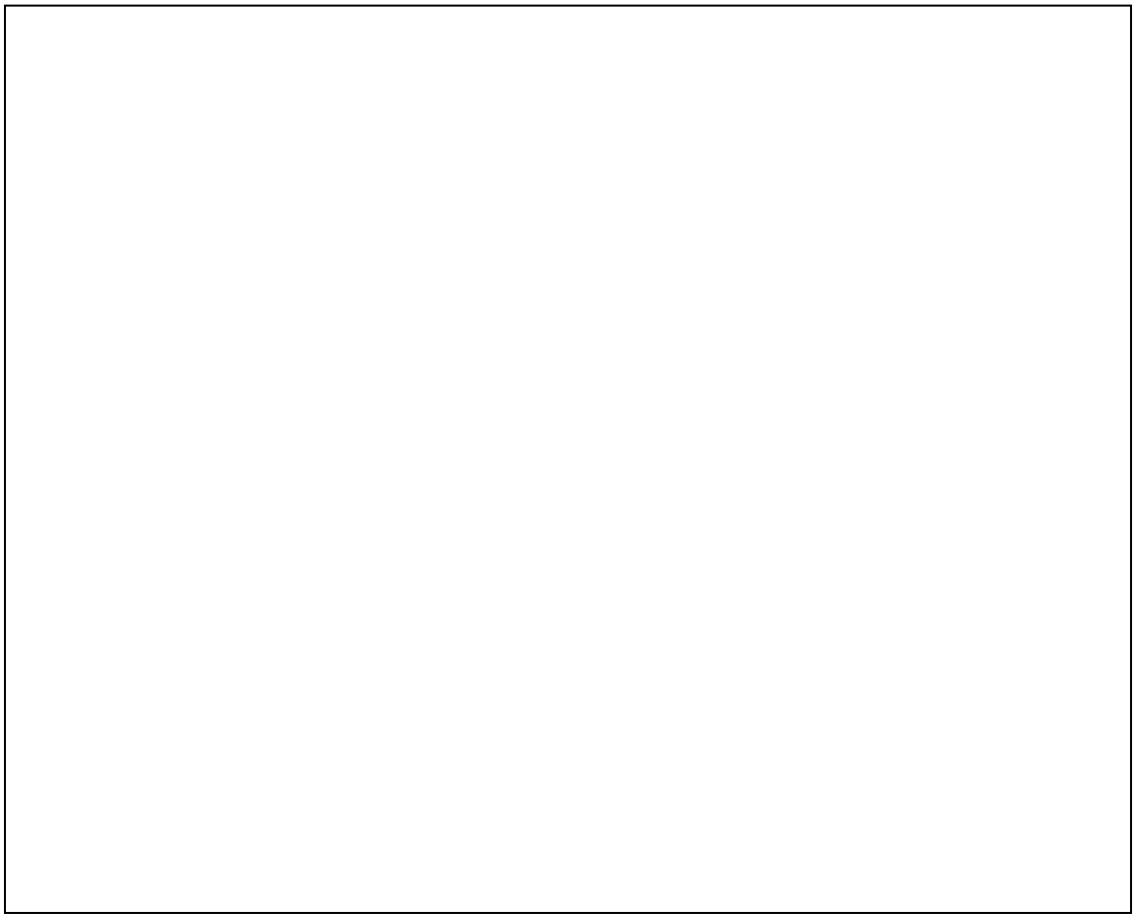
Mistake to avoid #4

- Forget that DB2 accounting records are not cut until end of thread.
- System person is looking at Type 72 Interval data and sees lots of CPU consumption
- You are looking at Type 101 data and see nothing.
- Check a monitor! Your thread is still running.



Mistake to avoid #5

- Be so focused on performance, you forget to check if the test actually worked!
  - Application reports error to web page but it is not a true HTML error.
  - Mercury Load Runner (or what ever testing tool you are using) reports SUCCESS or you just don't validate results.
  - Errors can perform very well. ☺



References

- z/OS MVS System Management Facilities (SA22-7630)
- http://www.watsonwalker.com/SMFreference.pdf
- Glenn Anderson: WebSphere App Server for z/OS Ver 6 Performance Monitoring Tools – WTE 2006
- Ned A. Diehl: DB2 CPU and Response Metrics – CMG2005
- Peter A. Enrico: WLM - The Lives and Times of Transactions on z/OS – CMG2003

Session: A09

Mainframe Optimization – Measure It!

Lisa Ouellette

Lisa.Ouellette@wachovia.com

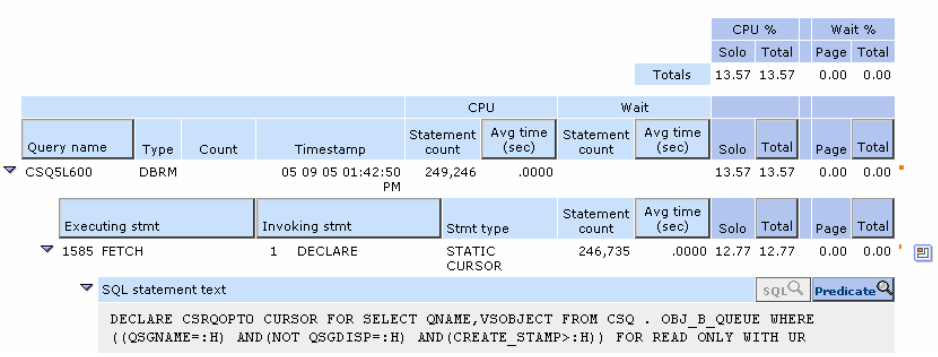
GoFurther

Appendix: Example #3 MQ Series

LOCATION	DATE	PROGRAM	AVG 158 TCB SECS	NBR EXECUTIONS	CONN ID	TOTAL 158 TCB HOURS
DP31	18-Aug	CSQ5L600	0.18	193815	RRSAF	9.8
DP32	18-Aug	CSQ5L600	0.19	2952545	RRSAF	158.7
DP41	18-Aug	CSQ5L600	0.20	465118	RRSAF	26.3
DP42	18-Aug	CSQ5L600	0.20	2438769	RRSAF	135.3
DP51	18-Aug	CSQ5L600	0.23	174489	RRSAF	11.3
DP52	18-Aug	CSQ5L600	0.22	655918	RRSAF	39.4
DPW1	18-Aug	CSQ5L600	0.27	17270	RRSAF	1.3
DPW2	18-Aug	CSQ5L600	0.34	17265	RRSAF	1.6

With the implementation of Shared MQ Queues we started noticing this DBRM/Package in our ‘Top’ Reporting.

Appendix: Example #3 MQ Series



			CPU %		Wait %	
			Solo	Total	Page	Total
Totals			13.57	13.57	0.00	0.00

				CPU		Wait					
Query name	Type	Count	Timestamp	Statement count	Avg time (sec)	Statement count	Avg time (sec)	Solo	Total	Page	Total
CSQ5L600	DBRM		05 09 05 01:42:50 PM	249,246	.0000			13.57	13.57	0.00	0.00

Executing stmt	Invoking stmt	Stmt type	Statement count	Avg time (sec)	Solo	Total	Page	Total
1585 FETCH	1 DECLARE	STATIC CURSOR	246,735	.0000	12.77	12.77	0.00	0.00

SQL statement text — SQL — Predicate

```
DECLARE CSRQOPTO CURSOR FOR SELECT QNAME,VSOBJECT FROM CSQ . OBJ_B_QUEUE WHERE
((QSGNAME=:H) AND(NOT QSGDISP=:H) AND(CREATE_STAMP>:H)) FOR READ ONLY WITH UR
```

47

GoFurther

Using Strobe found an SQL statement repeated over and over.

Appendix: Example #3 MQ Series

My favorite research tool – Google:

In setting up shared queues that reside in the coupling facility, you must BIND several DB2 plans including CSQ5R220 and CSQ5L220 (CSQ5L530 in MQ 5.3, CSQL531 in MQ 5.3.1). **CSQ5L220 executes once every 5 seconds for every QMGR within the QSG as it checks for newly defined shared queues**. If no new ones are found, then DB2 returns SQLCODE +100 (row not found). "Row not found" really means that an object matching a specified criteria was not found so no list was returned.

But the numbers didn't add up for most of our DB2 systems.

Appendix: Example #3 MQ Series

So look at MQ Series SMF Type 30:

JOB	JES ID	DATE	RES TIME HH:MM	CPU (158 HRS)	PCT CPU UTL
CSQAMSTR	PJX2	8/18/2006	24:00:00	1313	14%
CSQAMSTR	PJX2	8/19/2006	24:00:00	1032	11%
CSQAMSTR	PJX2	8/20/2006	24:00:00	945	10%
CSQFMSTR	PUR2	8/18/2006	24:00:00	1142	11%
CSQFMSTR	PUR2	8/19/2006	24:00:00	1041	10%
CSQFMSTR	PUR2	8/20/2006	24:00:00	978	10%

CSQFMSTR = DB2 DP32

CSQAMSTR = DB2 DP42

Ask IBM.

Qpasa (Monitoring tool for MQ Series) is possible culprit. Qpasa vendor helps us figure out how we shot ourselves in the foot and how to fix it:

-Turn off monitoring for queues that no longer exist on z/OS (About 200 of them)

-Reduce query interval from 30 seconds to 60 seconds.

-Turn off queue status monitoring (since we weren't using it)

NOTE! Qpasa started task was showing very little CPU time in SMF type 30 records. It was using the MQ series API to do /Display queue commands. The CPU time for all these showed up in the *MSTR tasks.

Appendix: Example #3 MQ Series

LOCATION	DATE	PROGRAM	AVG 158 TCB SECS	NBR EXECUTIONS	TOTAL 158 TCB HOURS
DP32	18-Aug	CSQ5L600	0.19	2,952,545	158.7
DP32	19-Aug	CSQ5L600	0.19	2,983,268	156.8
DP32	20-Aug	CSQ5L600	0.18	3,008,083	153.3
DP32	18-Sep	CSQ5L600	0.20	246,185	13.6
DP41	18-Aug	CSQ5L600	0.20	465,118	26.3
DP41	19-Aug	CSQ5L600	0.20	465,012	25.9
DP41	20-Aug	CSQ5L600	0.19	465,638	24.8
DP41	18-Sep	CSQ5L600	0.19	209,446	11.1
DP42	18-Aug	CSQ5L600	0.20	2,438,769	135.3
DP42	19-Aug	CSQ5L600	0.19	2,480,456	128.2
DP42	20-Aug	CSQ5L600	0.18	2,490,138	125.4
DP42	18-Sep	CSQ5L600	0.20	294,189	16.1
DP51	18-Aug	CSQ5L600	0.23	174,489	11.3
DP51	19-Aug	CSQ5L600	0.23	174,698	11.2
DP51	20-Aug	CSQ5L600	0.23	174,713	11.0
DP51	18-Sep	CSQ5L600	0.24	113,716	7.6
DP52	18-Aug	CSQ5L600	0.22	655,918	39.4
DP52	19-Aug	CSQ5L600	0.21	657,491	38.0
DP52	20-Aug	CSQ5L600	0.20	658,342	37.2
DP52	18-Sep	CSQ5L600	0.24	329,489	21.5

Significant reduction on all systems.

Appendix: Example #3 MQ Series

JOB	JES ID	DATE	RES TIME HH:MM	CPU (158 HRS)	PCT CPU UTL
CSQAMSTR	PJX2	8/18/2006	24:00:00	1313	14%
CSQAMSTR	PJX2	8/19/2006	24:00:00	1032	11%
CSQAMSTR	PJX2	8/20/2006	24:00:00	945	10%
CSQAMSTR	PJX2	9/18/2006	24:00:00	522	6%
CSQFMSTR	PUR2	8/18/2006	24:00:00	1142	11%
CSQFMSTR	PUR2	8/19/2006	24:00:00	1041	10%
CSQFMSTR	PUR2	8/20/2006	24:00:00	978	10%
CSQFMSTR	PUR2	9/18/2006	24:00:00	241	2%

SMF Type 30 data ‘after’