

8081 So you want to manage your Z-series MIPS? Then detect & control application workload variance!

Prepared by
John Van Wagenen
Caterpillar Inc, Peoria IL USA
van_wagenen_john_s@cat.com

for the
34th Annual International Conference of the Computer Measurement Group, Inc.
December 7-12, 2008
Las Vegas, Nevada USA

Workload analysis on the mainframe can be complicated if your business runs operations 7/24 with hundreds of applications. At our enterprise, we have a success story where the Z-series MIPS capacity has been held flat for more than two years. Each month every application is measured for current activity and compared to a baseline. Those applications that exceed a variance threshold are detected as “out of standard” or OOS for short. The OOS applications receive scrutiny to determine a named cause of the variance. This paper provides basic instruction and lessons learned.

INTRODUCTION

Has your CPU ever spiked one day and you couldn't begin to guess why? Perhaps your business environment is simple enough, that you just yelled out into the office and asked who made a bad change. Maybe your change control process and SOX documentation is so good that you simply scan an easily accessible web-log to identify the culprit. More than likely, your company is much like the enterprise where I work. It is large, integrated, complicated and filled with daily (even hourly) events that can swing CPU utilization from normal to danger with no early warning of the change.

PROBLEM STATEMENT

Our enterprise utilizes an IBM System z9 with 13 CP engines and 3 zIIP engines. This mainframe processes traditional BATCH/CICS/IMS/TSO workloads. Due to the advent of distributed DB2 access (DDF), this mainframe is also the single largest data-server in the company. Between 2000 and 2004, capacity for the enterprise was increasing at a rate greater than business growth (see figure 1a). The performance of our business applications was adequate, but we ran with variable workloads that were generally unpredictable. Our executive management questioned whether the capacity growth was realistically connected to business growth. We were challenged to control mainframe engine growth below the forecast of 100% MIPS growth during 100% sales growth as shown in figure 1a. This forecast was made in January 2005 for the next 36 months.

HYPOTHESIS

Could we provide an effective engine capacity management process that would:

- i) establish an expected baseline for monthly application CPU utilization?
- ii) detect “out of standard” applications by comparing monthly usage against the baseline?
- iii) scrutinize those applications with significant variance for cause?

We required a process that would not only find the applications that were OOS, but could determine a “named cause” for the variance. We wanted this process to be sensitive to monthly application change. We wanted a method to communicate the “named cause” to an application owner with the intent to assign a disposition for the

detected variance. Finally we wanted the process to be effective in controlling aggregate CPU utilization, yet be responsive to business growth. Creating a process that detected problems with no resolution was not an option.

Figure 1a

Mainframe projected trends through 2007

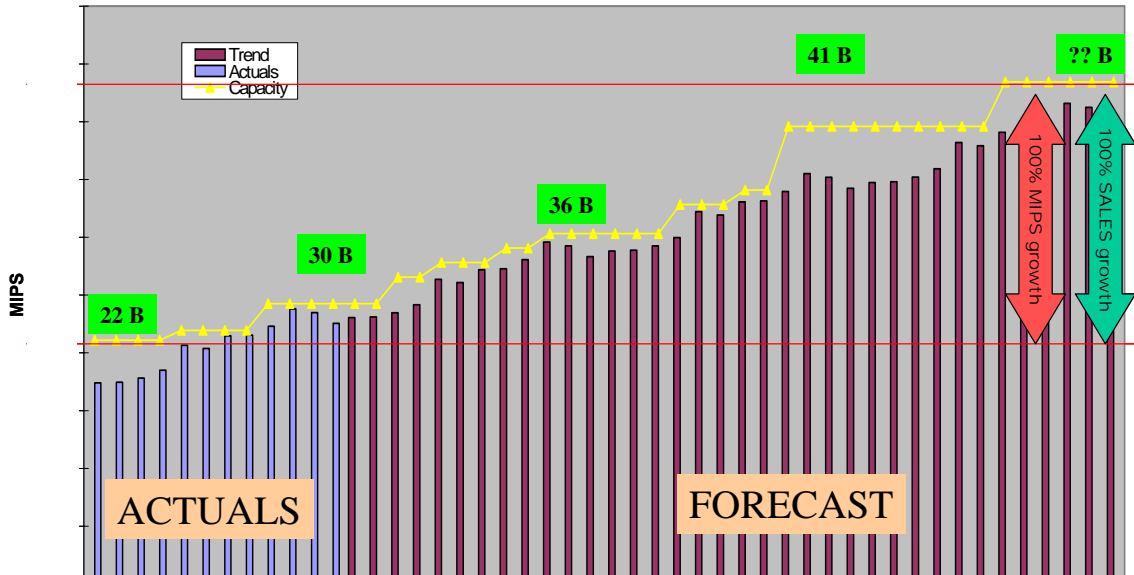


Figure 1a illustrates a forecast for MIPS growth over 36 months when projected sales growth would equal or exceed 100%. Management challenged the expense of 100% MIPS growth.

Enterprise gross sales are plotted against the capacity line for mainframe growth.

DATA

The data for this process is SMF data available on most Z-series machines. To arrive at utilization metrics by application, we use SMF type 30 records for BATCH and SMF 101 records for DDF. We have the resources to pull large amounts of detail data and summarize at various levels. This summarization is necessary, as real-time logging would create files too large to manipulate in EXCEL spreadsheets. We keep various summary data for different durations. Our lowest level of summary is to capture the application activity for each LPAR, by hour within a day. We capture both CPU seconds and volume (such as job count or transaction count). If the workload that we track is delineated by an identifier (such as jobname or transaction name), we create a record for each identifier. Our goal is to capture data at a level so that any monthly variance can be attributed to a named item, which will uniquely identify the workload. We keep the hourly/daily data for 180 days. We take each month's daily data and summarize into a few records per application. This allows us to easily calculate items like daily average or monthly utilization. The monthly summarization is the basis for both the application baseline, and the recent month's workload benchmark. We keep the monthly summary data for 24 months.

ANALYSIS

At our enterprise, we have summarized the workloads by technology for many years. For example, we know the average utilization, as well as the peak demand for workloads like TSO, IMS, CICS, BATCH, etc. Using this data, we discovered that 90% of recent growth in workloads was attributed to either BATCH or DDF. Reducing the scope of the technologies that must be scrutinized is a valuable first step. There is no need to investigate workloads that do not significantly contribute to the requirements for total capacity. We found that problems and tuning opportunities are quite often defined by the technology. Recognizing this allowed us to become "super-experts" in the few technology areas that we scrutinize closely.

Within in the areas of BATCH and DDF, we were fortunate that the applications used a consistent naming convention that could easily be tracked. For BATCH, this was the jobname. For DDF it was the primary authid.

The naming convention for jobnames and authids uses a format of ffssxxxx, where ff is the facility, ss is the application, and xxxx is a process identifier. Most applications are an aggregate collection of programs, jobs, transactions and processes. When these objects execute and use CPU seconds, we call the aggregate total of CPU time across a month, the application workload.

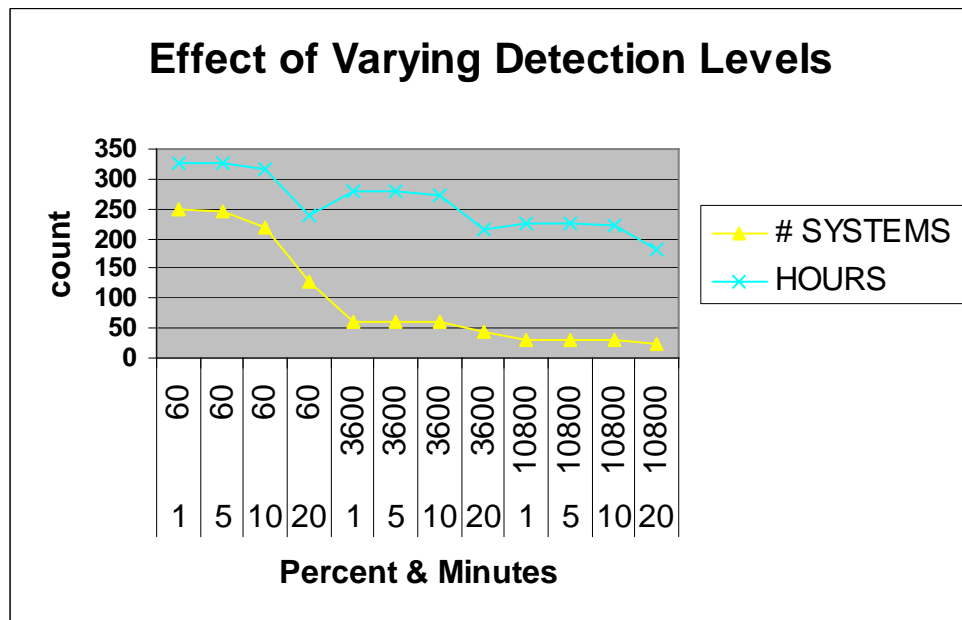
After collecting data every day for all applications that run BATCH or DDF, we summarize and store the data as a single monthly number per application in terms of CPU utilization. That metric is the total amount of CPU seconds that any set of jobs or transactions used, as identified by the application code in the name. We had accumulated this data for several years prior to establishing the OOS process. By using the accumulated monthly metrics, we could calculate the expected baseline for CPU utilization by application. This is just an average. We defined the baseline as the average of the past 13 months (not counting the current month). So considering the current month of comparison data, we are always working with the past 14 months of data. Taking an average over this span of time will smooth most cyclic workloads and extraordinary anomalies. The total of this baseline for all enterprise applications represents the expected enterprise workload each month. This is the number we are ultimately trying to control, which is accomplished by limiting variable workload and forecasting growth.

One key element to this process is to identify every application workload in the enterprise and perform this comparison every month. We found that once an application establishes its CPU utilization baseline, significant variance is quite unusual. Our process tracks over 700 BATCH and DDF applications each month, and 95% of these workloads do not vary significantly from the expected baseline.

Many of the parameters in this process would be best defined at each enterprise. The workload conditions will also vary based on business characteristics. At our enterprise, we decided that a 20% positive variance from the established baseline would be worthy of closer scrutiny. So for example, if the baseline of an application was 100 CPU hours per month, and the current month activity was measured at 121 hours, we would investigate. But if another application with a 200 CPU hour baseline had increased to 238 hours, we would not. Since most applications are not balanced in size, we also require that the variance amount is greater than an absolute amount of CPU time, which in our case is three hours. An application that doubled from two hours to four hours would not be investigated since the variance is too small.

Using current data, we modeled several scenarios of detection based on a percentage of variance and an absolute CPU second limit. We plotted the amount of variance hours, and the total number of applications detected against each scenario. The goal was to find the detection scenario that yielded the greatest amount of variance hours, represented by the least number of applications. The following chart (figure 2a) illustrates the results of this modeling. We chose the optimal detection threshold at 20% and 10,800 CPU seconds.

Figure 2a



Considering the original hypothesis, we needed three metrics in order to detect variance and investigate the cause. The first is a baseline utilization number, the second is the current application workload, and the last is the detection threshold. Once we had an ability to capture and process this data on a monthly basis, we needed a

process to compare the current application workload (we call it the benchmark) to the established 13 month average (we call this the baseline). At our enterprise, an application is “out of standard” when **{(benchmark – baseline)/baseline > detection threshold}**.

OOS PROCESS

The “out of standard” (OOS) process used at our enterprise is an in-house creation of data collection, metric analysis, and management presentation techniques that have been refined over several years. Any organization could emulate this process, but each computer center has its own evolution and maturity. Do not assume that your business could simply “plug & play” the process steps that are outlined. The concepts are common to most mainframe computer workloads, but characteristics will vary depending on the business and computer center. Some corporations would see this process as overkill. Others might think it is not sufficient to diagnose specific performance/capacity issues.

The timing of the OOS process has been established to capture application metrics on calendar day one, compare benchmark to baseline, looking for variance by the third accounting day, and then communicate the named cause of the variance by the fifth accounting day in the month. This timing is established so that the enterprise carries the effect of application variance the least possible amount of time.

The OOS process can be thought of as being driven around the two scheduled meetings including the preparation and assignments that occur as a result. The steps are to prepare for our “day 3” meeting, hold the meeting, and make assignments from that meeting. Then we do the same for our “day 5” meeting.

The process began with daily data collection. Each day, a BATCH and DDF dataset are created representing the hourly workload running on our mainframe LPARS. An example of each daily file is shown in figure 3a & 3b. This is the lowest level of detail that is stored historically. Sometimes these are called the “daily/hourly” files.

Figure 3a

LPAR	DATE	SHIFT	JOBNAME	FAC	SYS	FACSYS	CPU hours	Wait seconds	JOBCOUNT
BROWSE	TST.Z1DW.BATCH.DAILY.G1274V00						Line 00046637	Col 001 080	
Command	====>						Scroll	====>	PAGE
A2S2	2008-05-27	PRIME	H0EV1830	H0	EV	H0EV	0.000161	11	2
A1S1	2008-05-27	PERIOD3	H0EV1805	H0	EV	H0EV	0.000003	0	1
A1S1	2008-05-27	PERIOD3	H0EV1850	H0	EV	H0EV	0.000028	3	1
A1S1	2008-05-27	PRIME	H0EV1805	H0	EV	H0EV	0.000006	0	2
A1S1	2008-05-27	PRIME	H0EV1850	H0	EV	H0EV	0.000039	2	2
A1S1	2008-05-27	PRIME	H0EV1881	H0	EV	H0EV	0.000033	1	1
BGSG	2008-05-27	PERIOD2	H1FA2010	H1	FA	H1FA	0.000108	8	1
BGSG	2008-05-27	PERIOD2	H1FA2020	H1	FA	H1FA	0.000003	0	1
BGSG	2008-05-27	PERIOD2	H1FA4110	H1	FA	H1FA	0.000014	0	1
BGSG	2008-05-27	PERIOD2	H1FA4230	H1	FA	H1FA	0.000017	1	1
BGSG	2008-05-27	PERIOD2	H1CE0096	H1	CE	H1CE	0.000875	46	2
BGSG	2008-05-27	PERIOD2	H1CE0807	H1	CE	H1CE	0.000692	65	1

Figure 3a & 3b illustrate the detail data available for both BATCH and DB2. This data is very useful because the hourly workload is sub-divided by each jobname and/or authid name that contributes. This means at an hourly level, we can determine frequencies and the unit-of-work for most components of most applications. In a tuning problem, we can pinpoint the hour and day when a change created a negative impact.

Figure 3b

```
BROWSE      MST.Z1DW.PR.DDB2HR.G1590V00                Line 00000000 Col 001 080
Command ==>>>                                         Scroll ==>> PAGE
***** Top of Data *****
2008-05-27  00.00.00  Z1P1    Z1DD4BIC  AutoExe.exe        1256      948
2008-05-27  00.00.00  Z1P3    Z1KDCSPI  db2jcd             590       218
2008-05-27  00.00.00  Z1P5    Z1HRPSAP  PSAESRV.exe       537     14799
2008-05-27  00.00.00  Z1P5    Z1HREPMP  phantom           398       214
2008-05-27  00.00.00  Z1P5    Z1HRPSAP  PSAPPSRV          393    126849
2008-05-27  00.00.00  Z1P3    Z1QY6SSS  db2jcc_appli      255        6
2008-05-27  00.00.00  Z1P5    Z1HRPSAP  PIPINRUN.exe     127       426
2008-05-27  00.00.00  Z1P5    Z1HRPORT  PSAPPSRV           95    79367
2008-05-27  00.00.00  Z1P1    Z1DHCSS1  db2jcc_appli      85     4988
2008-05-27  00.00.00  Z1P6    N4GK1WEB  db2jcc_appli      68    77905
2008-05-27  00.00.00  Z1P1    D7GDHISW  db2jcc_appli      65     1106
2008-05-27  00.00.00  Z1P3    Z1JIPIS1  db2jcc_appli      61     2253
2008-05-27  00.00.00  Z1P9    Z1RPSIMS  db2jcc_appli      56    19820
```

DATE	TIME	SUBSYS	AUTHID	CORRID	CPU seconds	SQL count
------	------	--------	--------	--------	-------------	-----------

The daily files are stacked into monthly files (all detail retained). This monthly stack is then summarized so that each tracking identifier (jobname or authid) has one record per LPAR for the month. In this fashion, we can now assign a single value to the aggregate CPU utilization. Using our enterprise naming conventions allows us to summarize at the application level, producing a monthly file with each application code summarized by LPAR. An example of this data for DDF is shown in figure 4a. For DDF, SUBSYSTEM is analogous to LPAR.

Application data is available on the first calendar day of the month. The aggregate application files for BATCH and DDF are created on this day. The source of this data is the daily/hourly application data shown in figures 3a and 3b. A simple utility language like SAS was used to summarize the records.

Figure 4a

DATE	SUBSYS	SYSTEM	CPUSEC	SQLCOUNT	CPUCP	CPUZIP
1-Apr-08	Z1P1	DW	5212	483287	2478	2734
1-Apr-08	Z1U2	DW	2692	33806	1239	1453
1-Apr-08	Z1P2	DX	130	67969	77	53
1-Apr-08	Z1P3	EA	0	4	0	0
1-Apr-08	Z1P3	EK	664	77451	328	336
1-Apr-08	Z1P6	EK	215	9367	99	117
1-Apr-08	Z1P9	EK	68	59345	33	35
1-Apr-08	Z1P1	EY	7	5195	3	4
1-Apr-08	Z1P3	FC	182	5	82	100
1-Apr-08	Z1P3	FL	728	17753	339	389
1-Apr-08	Z1U2	FL	718	5045	351	367
1-Apr-08	Z1P3	FP	20683	979459	10123	10560

Figure 4a illustrates the summarized application data with one line per application per Subsystem.

The monthly summary file data is imported into an EXCEL spreadsheet where a pivot table will be updated. A pivot table is a spreadsheet tool that allows for the arrangement of data to show monthly CPU utilization in comparison to the calculated baseline. This tool helped to accomplish the second step in our original hypothesis. By entering a set of spreadsheet formulas, and the detection thresholds, the pivot table can easily identify the applications that are OOS. This list of application codes is investigated for the cause of the variance.

After this data is loaded into the spreadsheet as raw data for the current month, a set of calculation cells determines if any of the applications are "out of standard". At our enterprise, we use a threshold of 20% variance above baseline; where the absolute variance amount is more than 10,800 CPU seconds (3 hours). The calculation cells and the pivot table are built so that each month, all applications will be compared against the current threshold. An example of the BATCH pivot table is shown in figure 5a. Notice the threshold variables and how we tabulate the number of applications that are OOS and the total variance hours for the entire enterprise.

Figure 5a

Sum of normalized	DATE							
SYS	Mar-07	Apr-07	May-07	Jun-07	Jul-07	Aug-07	Sep-07	Oct-07
AA	2398	2116	2211	2130	2102	2310	1938	2369
AB	7441	6692	6986	7772	7746	8338	7385	8580
AC	13320	10205	10694	9605	9297	10985	9014	10364
AD	26286	28086	30883	27026	26284	30116	26345	35304
AE	9854	9735	9036	8907	8981	9779	8885	9402
AF	419	327	307	302	336	349	316	365
AG	74	69	69	69	66	72	72	69
AH	32784	23935	25239	22903	22384	24202	21282	26893
AI	1863	1931	2061	2144	2259	2489	2079	2551

% tolerance	20%	Total systems "out of control"	11
min cpu seconds	10,800	Total hours "out of control"	77

Figure 5a is an example of the pivot table for BATCH data used in the OOS process.

This BATCH spreadsheet, including a variety of pivot tables and charts, represents the single control point for all mainframe application monitoring at our enterprise. The data is considered to be trustworthy and represents application workload that can be investigated down to daily/hourly detail for up to two years of history.

Sometime before the third accounting day of the month, the OOS analysis will be performed to identify the top 10 BATCH and DDF applications in terms of CPU utilization variance. It is important to remember that these are the applications that have changed the most in the last 30 days. They will not necessarily be the largest applications, nor the most critical to the enterprise. However, they are contributing directly to CPU utilization above an expected baseline amount. This group of applications is why more capacity will be needed if the utilization is not challenged. We use the OOS process to try to maintain a consistent enterprise baseline, and to ensure that growth is linked to approved business initiatives and costs.

All of these activities constitute the preparation for the "day 3" meeting, and involved a large amount of automated data collection and summarization. In addition, the OOS team leader updates the pivot tables and associated application utilization charts. At this point in our process evolution these activities require about 4 hours.

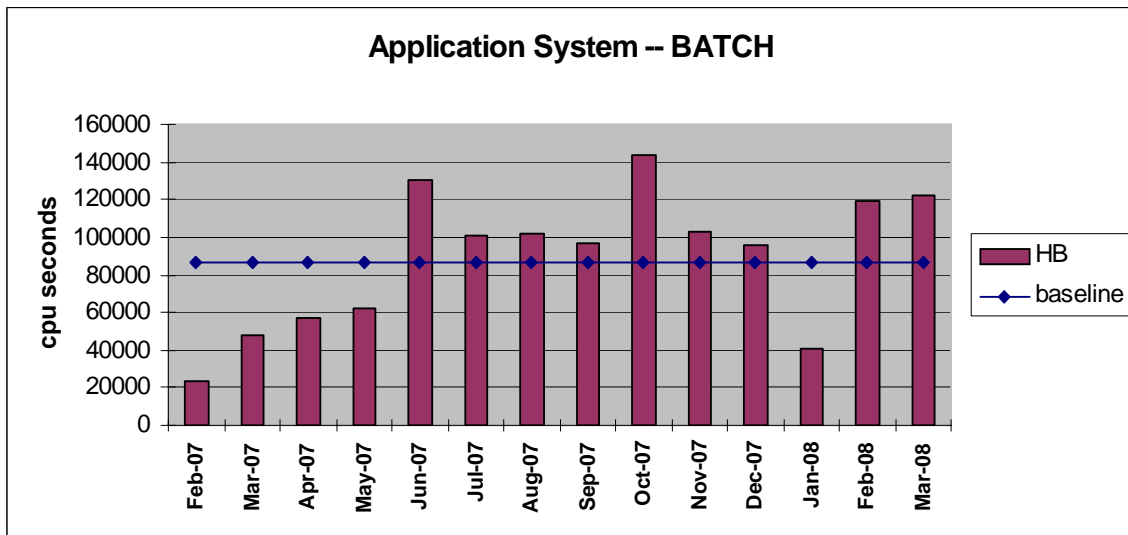
We have assigned a team of 10 analysts to execute the OOS process each month. A leader facilitates the team and the OOS process, using two meetings and several formal assignments. The first meeting occurs on the third workday and lasts 90 minutes. During this meeting, the actions taken on the previous month's application list are reviewed for final disposition. In addition, the current month's top 10 DB2 & BATCH applications are assigned to the team members for further scrutiny. In this fashion, the top 20 causes of the application variance for the past two months are reviewed. Discussion is limited to factual statements about workload. The team's purpose is to quickly arrive at an accurate cause and disposition of recent application variance.

Upon receiving an OOS application to investigate, the team member will be provided an initial chart that illustrates the baseline and current variance. This is shown in figure 6a. The team member's assignment is to find the "named cause" of the variance and to communicate this to the appropriate application support group. This is

accomplished via an assortment of comparative methods to determine “what happened this month that didn’t before”. We have simplified the process so a quick comparison of the monthly detail files will almost always yield the net difference that caused the variance.

Figure 6a

Example of baseline/variance chart for a BATCH application.



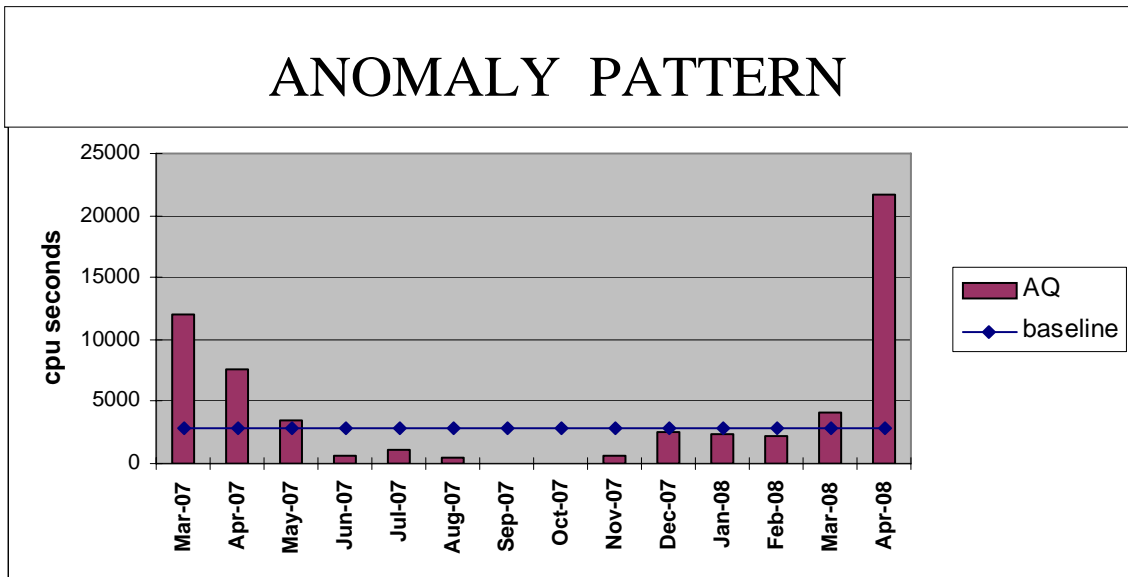
After the assignment is made on day 3, but before the fifth workday, each team member should contact the responsible application support group and describe the variance in the application. By providing the named item(s) that changed, most application groups can offer commentary about the variance.

At this point, we reflected that the ultimate measure of success would be if we contained MIPS growth as our management had desired. We had developed a responsive measurement technique that could determine when and why we used more capacity. We now needed a way to effectively reduce increases to application workload. Our enterprise goal was not to reduce the aggregate baseline, but to control growth. We discovered that success occurred when the application could return to the baseline utilization as soon as possible. Secondary success involved the identification of application workload growth that had been forecasted and approved.

Since we identified 20 application opportunities each month, a substantial amount of resource could be applied simply to create an explanation, with no actual effect to the future application workload. We discovered that three dispositions could describe what our actions were and what the outcome will be to the enterprise. We use one of three descriptive terms: **anomaly, tuning, or growth**.

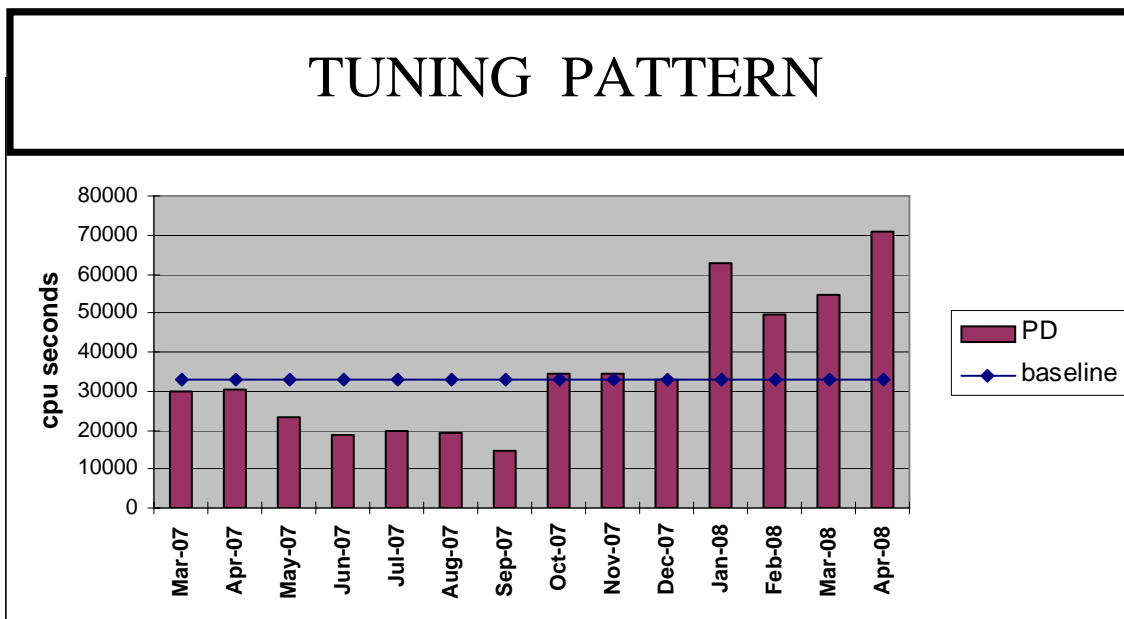
An **anomaly** is our description of workload that varied for the most recent month, but is neither expected growth nor an ongoing problem. Think of this as a rerun, or a bad test. The workload caused a spike, but will most likely not repeat and the application will return to its previous baseline with no further action. We include seasonal demand and cyclic processing as part of this description. See figure 7a for an example of an application that demonstrates an anomaly workload pattern.

Figure 7a



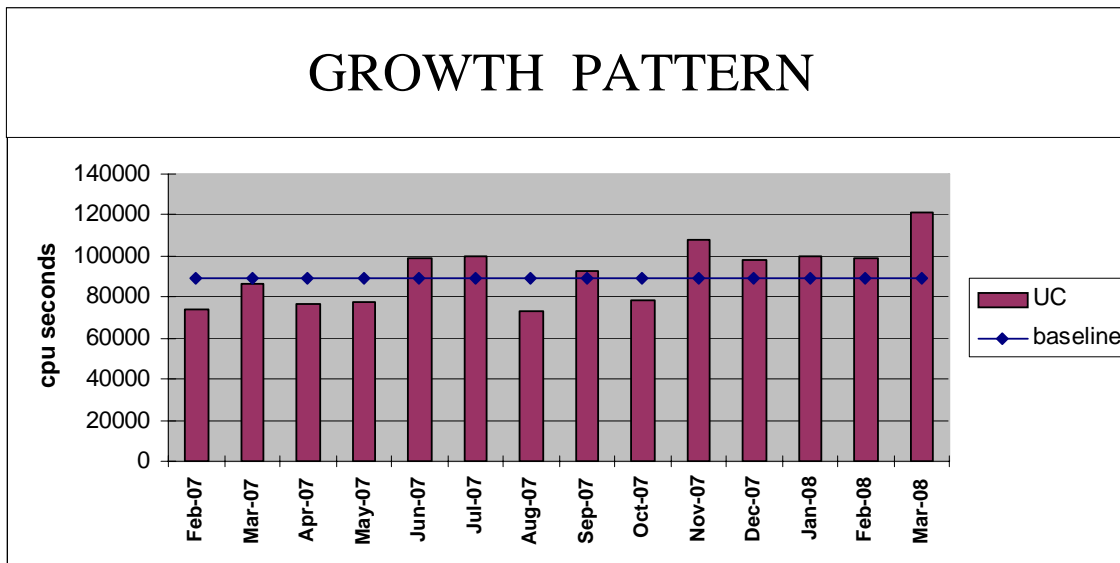
A *tuning* opportunity means we have discovered an increased workload that is not efficient and is impacting the enterprise. In many cases this occurs as a result of untested application change, yet we have discovered that subtle revisions in business processes can also cause wasteful or unnecessary CPU utilization. The workload increase is basically unplanned resource utilization. See figure 8a as an example of a TUNING workload pattern.

Figure 8a



The third disposition is called *growth* and is described as increased application workload that will continue to occur each month. Growth is represented by the aggregate total of all application variance and will be the ultimate driver of capacity increases. There are many types of growth. For the purpose of this discussion, we will assume all growth has value to the enterprise. All workloads that are accepted as growth should be scrutinized to confirm that they are efficient, appropriately sized, and scheduled for optimal load balancing. See figure 9a as an example of a GROWTH workload pattern.

Figure 9a



When monthly variance is appropriately assigned to these dispositions, we experienced the best ROI. In the case of an anomaly or tuning, the workload will ultimately return to the baseline as measured before the OOS detection. When the workload variance is due to growth that was expected by the business, capacity growth can be planned. This methodology of analyzing and attacking application variance creates the smallest rate of growth in MIPS for the enterprise. That was our goal as desired by management.

Basically we ignore the variance that doesn't matter, fix the variance that is broken, and accept the rest as growth.

How did we improve at diagnosing the variance disposition? The answers were in the daily/hourly detail files that we collect and store for up to two years. By comparing the detail of an application workload between a month that represented the baseline and the current month, patterns became apparent and consistent in our analysis. The anomaly pattern is usually indicated by detail data that exists one month, but has not existed in previous months. Or it can be verified as a seasonal or cyclic workload based on historical data. The tuning pattern is identified subtly in the detail data by changes in the frequency-of-execution or the unit-of-work. Most tuning opportunities will follow a known application change. The growth pattern is identified by detail data that exists one month, but has not existed in previous months. Initially this will look like an anomaly, but application support staff or previous forecasts should confirm that the workload is ongoing.

After establishing that the ultimate disposition of the variance determined our team actions and ROI for our effort, we documented standard ways to communicate about application variance and our expectations for the OOS applications detected. In the case of an anomaly, we are going to accept the workload variances ultimately, so we need to perform the least amount of application scrutiny. In many cases, no application contact is made at all. When a tuning opportunity has been identified, our most effective action is to notify the application support group and identify the named cause of the variance. This action path requires the largest amount of enterprise resource to accomplish unknown gain in either frequency and/or unit-of-work improvement. If a tuning opportunity is not successful, then the disposition of the application variance is effectively growth. When the variance is a growth workload, the utilization is being accepted and will add into the enterprise baseline. It should meet minimal expectations. For a new workload to be efficient it should demonstrate that all unit-of-work is within enterprise tolerance. To be appropriately sized means that a reasonable comparison to an existing workload can be made. In other words, adding one more department to a process with 10 already should cause a 10% increase, not double. Lastly, new workload should be scheduled in order to minimize the impact on enterprise peak utilization. Considering this fact, some growth does not matter if the utilization occurs during non-peak times.

By day 5, each team member should have determined the disposition of the application variance they were assigned to investigate. If possible, we try to get application confirmation. On the fifth workday, we have a two-hour meeting to discuss the current month OOS applications: the top 10 in both BATCH and DDF. Twenty total assignments are discussed each month.

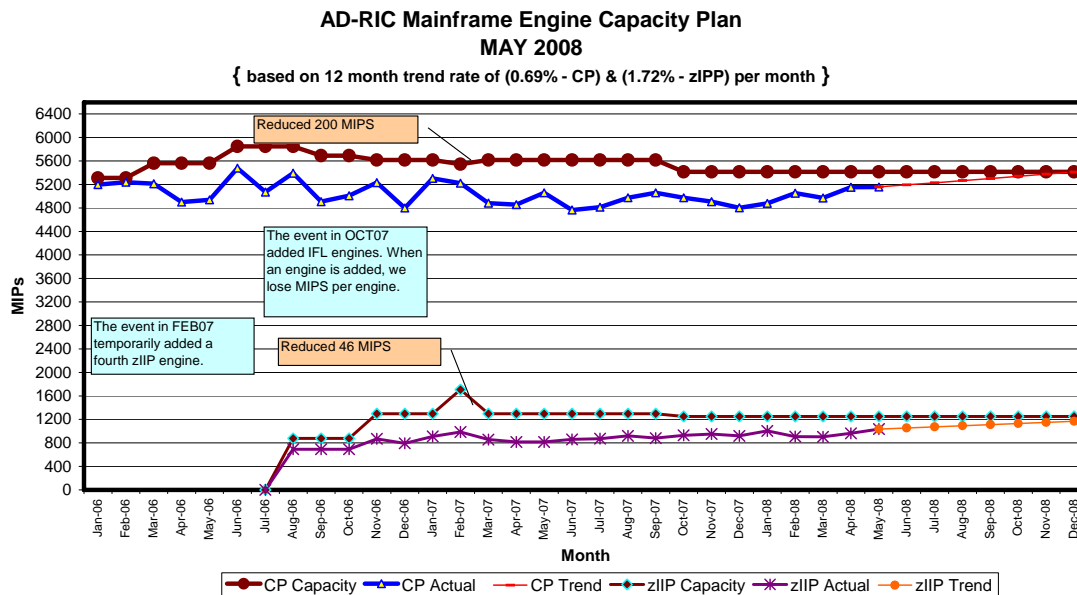
This ends the team responsibilities for the OOS process. Application support will follow-up with tuning changes if required, or will help to monitor growth and anomalies. Upon request, the OOS team will offer tuning consultation or metric analysis for a specific application. The enterprise expects all tuning opportunities to be pursued and completed within 60 days of the OOS notification. Enhancements for new application workload have more aggressive targets for improvement and are expected to occur in 30 days after the OOS notification.

Since the determination of the disposition (for the application variance) is based on the analysis and interpretation of metrics, we have at times guessed wrong. One of the purposes of the OOS process is to continuously review all of the enterprise applications. In this fashion, as time passes and more workload data is available to analyze, we refine our diagnosis and make appropriate assignments. Our experience is that workload variance which we miss or misinterpret is normally caught within the next 30-90 days as the monthly OOS process is executed.

Finally we arrive at the most desired management metric. That number is the amount of application growth that we expect to carry forward every month. It drives the requirement for acquiring future engines. This growth metric is plotted against the enterprise capacity pad during peak utilization. In this fashion we can adequately forecast the engine additions by month for the next year. We define capacity pad to be the difference between the total enabled capacity and the average utilization during PRIME. At our computer center, the peak utilization occurs during a 45 hour window of time known as PRIME. We can then anticipate when the next engine bump will occur due to growth and if we may be able to delay by applying tuning efforts. For example, if the growth rate was 15 hours per month during PRIME, we would need an extra engine every three months. While this is rather unsophisticated as a projection, we found it accurate enough for planning engine acquisition.

VALIDATION

Figure 10a



*Figure 10a shows the recent stabilization of the CPU utilization each month
In 2004, the rate of growth was 100% every 36 months. We have controlled that rate of growth. .*

In terms of cost control for mainframe engine growth, the chart in figure 10a shows the recent history of capacity planning at our enterprise. The MIPS level for our mainframe CP engines has stabilized during the past 36 months at 12% above the actuals in DEC2004. We have not added a mainframe CP engine since March 2006 and do not anticipate capacity growth in the remainder of 2007. See note in the summary about zIIP engine growth since 2006. The total aggregate growth through the end of 2008 is projected at 37% compared to 125% sales growth during the same timeframe (JAN2004 – DEC2008). It appears we satisfied the objective to control enterprise cost for mainframe CP engines in relation to sales growth.

While the long-term cost control is the ultimate ROI for our management, we needed a more immediate measure of success for the OOS process. To understand whether we were having an effect, we needed to monitor all applications contacted to see if they returned to the baseline, or showed signs of improvement.

The next chart is a report card for the OOS process and the net effect to enterprise application baseline. See figure 11a for an example of the report card through April 2008. This chart shows for each application that is detected to be OOS, what the established baseline is, and how the application performs against it currently.

We interpret this with two calculations. First divide the 383,044 by the total hours in a month (720). This yields the least enterprise impact and would imply for this example, that we need 532 extra CPU seconds per hour above the current baseline workload. The second calculation is to divide by the PRIME hours of the month (180). That comes from nine hours per day multiplied by 20 workdays. This yields the worst enterprise impact and would imply for this example that we would need about 2128 extra CPU seconds per hour above the expected baseline workload. 3600 CPU seconds would be an extra engine. So in this case, we were experiencing extra workload between 9 minutes and 36 minutes each hour above our baseline.

The real answer is somewhere between these two values. At this point in time, we had enterprise pad measured at 1.0 engines per hour. In other words, we had average peak utilization of 15.0 engines out of 16. Since the total application variance is between 9 and 36 minutes, then we do not need more capacity and the enterprise performance is probably tolerable (if not good). Managing the amount of enterprise pad in comparison to the monthly variance of applications allows us to enable the least amount of total capacity.

Figure 11a

	JAN	FEB	MAR	APR	April Actual	difference
US	306346	US 323900			323900	360131 36231
AQ	11386	AQ 14999	AQ 18070	AQ 21826	58652	36826
TJ	55093	TJ 57806		57806	59997	2191
BL	33398	BL 38247	BL 42995	BL 47580	67702	20122
SZ	35421	SZ 37876	SZ 40970	SZ 44298	61664	17366
OE	22436	OE 26180	OE 29535	OE 33003	47559	14556
HR	96411			96411	124599	28188
HB	73029		HB 80065	HB 86383	122739	36356
UB	8263			8263	6068	(2195)
CW	9010	CW 10595	CW 12166	12166	25525	13359
		NV 136700		NV 150532	197981	47449
		BN 8242		8242	19303	11061
		MM 21609		21609	30679	9070
			DD 69603	69603	75733	6130
			PO 12154	PO 13023	27485	14462
			NP 1968	1968	11348	9380
			YS 3388	3388	8680	5292
				UC 89468	120955	31487
				MX 127888	158365	30477
				ST 48920	64156	15236

383044

The OOS process is not labor intensive since most of the work is done by batch jobs that collect and summarize the detail data about the applications. For the 10-person team, the monthly staff burden is about 50 hours of analysis and communication time. When this team successfully identifies a tuning opportunity, we track the before and after benchmarks. The annualized savings is projected based on historical volumes. Our enterprise places a chargeback value on an hour of CPU at \$335. In the year 2006, we documented tuning savings that exceeded \$1,000,000 or about 3000 CPU hours. Considering the OOS team worked less than 720 total hours, this ROI is very impressive to our management. In the year 2007, the documented savings exceeded \$700,000. During 2008, the enterprise is reacting to substantial business growth, which influences the resources for tuning.

SUMMARY

After executing the OOS process for over 36 months, we feel our initial hypothesis has been proven. The enterprise cost control for mainframe CP engines has been accomplished. In no 12-month period of time before, had we ever avoided a CP engine upgrade. Now we have avoided this cost for 27 months. The growth rate of utilizing this critical resource has flattened to below the business growth rate making our management happier.

During the timeframe between July 2006 and November 2006, the enterprise added approximately 18% more MIPS by utilizing zIIP engines. The OOS process established the DB2 baseline and allowed us to size this growth accurately. We measure the zIIP engine capacity as a separate resource from the aggregate CP engine total capacity. This is primarily due to the distinct cost and license characteristics of the specialty engines. The zIIP capacity is included in the aggregate comparison of 37% MIPS growth to 125% sales growth.

Indirectly we gained several additional benefits from this process. Application performance overall has improved. Most of our software must meet performance standards now. There are many improved toolsets for the detection, analysis, and monitoring of our business applications. **Paying attention to this process has paid off!**

The ongoing success of this process revealed yet another unforeseen benefit in 2008. As the process controlled the baseline workload for longer periods of time, we were able to wait until the next generation of mainframe architecture became available. In our case, our next capacity increase will occur on a Z10 machine. Advantages beyond the new technical features will yield more computing power for less total enterprise cost. This outcome has also pleased our management, who in turn provide support for the process by emphasizing how important application tuning efforts are to the enterprise.

Like many diagnosticians who dedicate themselves to a process, their skills and speed increase impressively during the first few years. The ability to detect a problem, apply diagnosis for cause, provide metric-based solutions, and apply them cost-effectively in the enterprise is the framework of this process. The tenure of the team members provides applications the ability to attain performance levels that meet or exceed the corporate standards. This allows our business resource (the mainframe computer) to exceed published service level agreements. **Enabling our business to be more successful and more profitable.**