BACK TO THE
TABLE OF CONTENTS

Global and Application Levels Exception Detection System, Based on MASF Technique

Igor Trubin, Ph.D.
Capital One Services, Inc.

The paper describes one site's experience of using Multivariate Adaptive Statistical Filtering (MASF) to produce web-based exception reports against SAS/ITSV performance databases for a large, multi-platform environment. In addition to global exceptions, the system can capture application level exceptions by using standard workload characterization. The history of exceptions, kept in a separate database, is used to analyze seasonal stresses, considering it as a natural test to discover the weakest subsystem.

1. Introduction

The Statistical Process Control concept became popular for detecting statistically significant exceptions of the computer system's behavior. The approach comes from the mechanical engineering discipline [1] and was successfully adjusted to computer systems by developing the Multivariate Adaptive Statistical Filtering (MASF) technique [2].

The Exception Detection System (EDS) is used for automatically scanning through large volumes of performance data and identifying measurements of global metrics that differ significantly from their expected values. Extending on the MASF method, this author suggested using some new derived metric such as "amount of exceptions per day" and keeping the history of exceptions in a separate database to produce advanced capacity planning analyses. This paper presents an interesting case of interpreting the historical data.

In order to increase the accuracy of exception detection, an application level was added to the system and is described in this paper also.

2. Review of the existing tools (SAS and BMC)

The SAS system has the SAS/QC (Quality Control) interactive tool, which is a good example of a classical implementation of Statistical Process Control concept, and can be applied on an ad-hoc basis to detect statistical exceptions against SAS data sets with the computer performance data. However, there is no way to use any special filtering policy to differentiate any subgroup such as weekdays. Indeed, a Monday, for example, can have a different pattern from other weekdays, and only the MASF technique can take that into consideration. But for some summary group such as weekly or monthly averages, it might be not as important, and this tool can detect the statistical exceptions. Figure 1 shows an example of using SAS/QC to find the weeks that had the statistically unusual usage of a server's CPU.

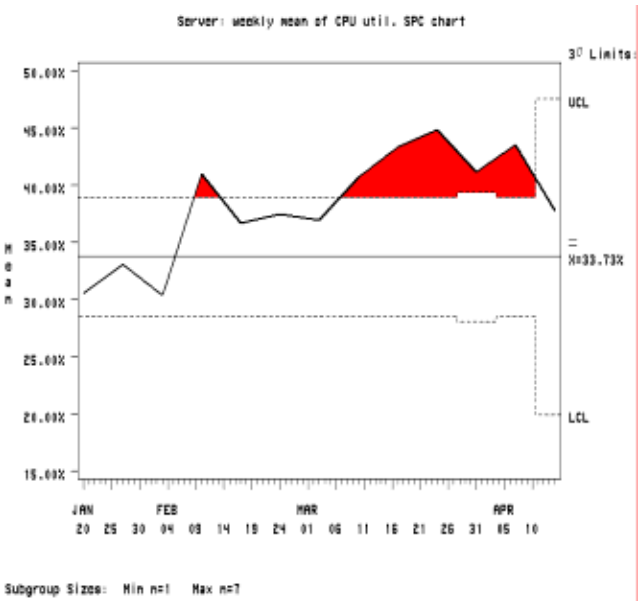

Figure 1 - SAS/QC SPC chart example shows the significant exceptions of a server's CPU usage

The Patrol Perform and Predict tool from BMC software has several procedures for various types of exception detection. It can capture the exceptions using either constant thresholds or flexible ones calculated based on the MASF procedure. This implementation was discussed in the 2001 CMG conference [4]. The system can automatically generate and publish SPC charts and even keeps

some exceptions and MASF policies in the database. BMC Patrol does not have a built-in capability of setting up alarms by dynamic or statistical exceptions. In addition, the database keeps only simple non-statistical exceptions such as constant thresholds and numbers of exceptions. Figure 2 shows a typical MASF chart generated by the Patrol Perform system.

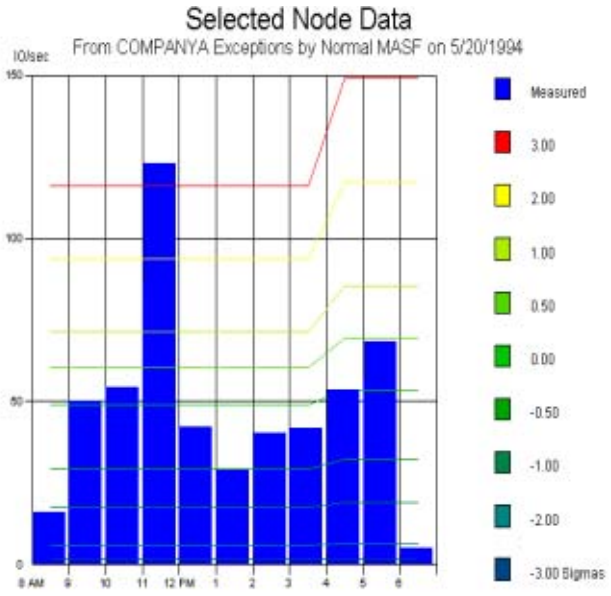

Figure 2 - BMC MASF chart example shows the significant exception of DISK I/O between 11:00 AM and 12:00 PM

Both tools can only state and illustrate the fact of the statistical exception occurrence. However, they cannot estimate the magnitude of exceptions during a particular interval (for the day, week, or month) and cannot keep their history. The Exception Detection System, which was developed by this author, is also based on MASF and also can publish SPC charts. It additionally has a mechanism to produce exception estimation; generate the statistical exception alerts; and keep the history of statistical exceptions in the separate database.

3. Exception Detection System Structure

The Exception Detection System is the subsystem that uses inputs from the SAS/ITSV Performance Database (PDB). The structure is presented in Figure 3 and consists of the following main parts:

- exception detectors for the most important metrics such as CPU, memory and disk utilization, memory page rate, and CPU run queue;
- Exception Detection System database with history of exceptions;
- statistical process control daily profile chart generator;
- exception server name list generator;
- Leader/Outsider servers detector and detector of runaway processes; and
- Leaders/Outsiders bar charts generator.

Within MASF, the exception detector (SAS program) scans the six-month history of each server every day for hourly performance data. The full "7 days X 24 hours" adaptive filtering policy is applied to calculate the average, upper, and lower limits (three standard deviations) of a particular metric for each weekday for the past six months. See Figure 4, where Monday is the example.

To generate a list of servers that experienced a significant exception for the last day, the following special rules are applied in addition to statistical filters:

- Ignore a slight increase of workloads for underutilized servers that used less than 20% CPU, Memory, or Disk capacity.
- Ignore dates when the server had a runaway process when calculating mean and standard deviations. A runaway process might be, for instance, a parasite infinite loop capturing free resources and hiding the normal picture of server utilization.
- Ignore insignificant isolated spikes (in most cases it must be at least two hourly exceptions to create an alarm).
- For metrics of percentage of utilization, deviations that exceed 100% are converted to the value 100%.
- Deviations that go below zero for all metrics are converted to the value of 0.

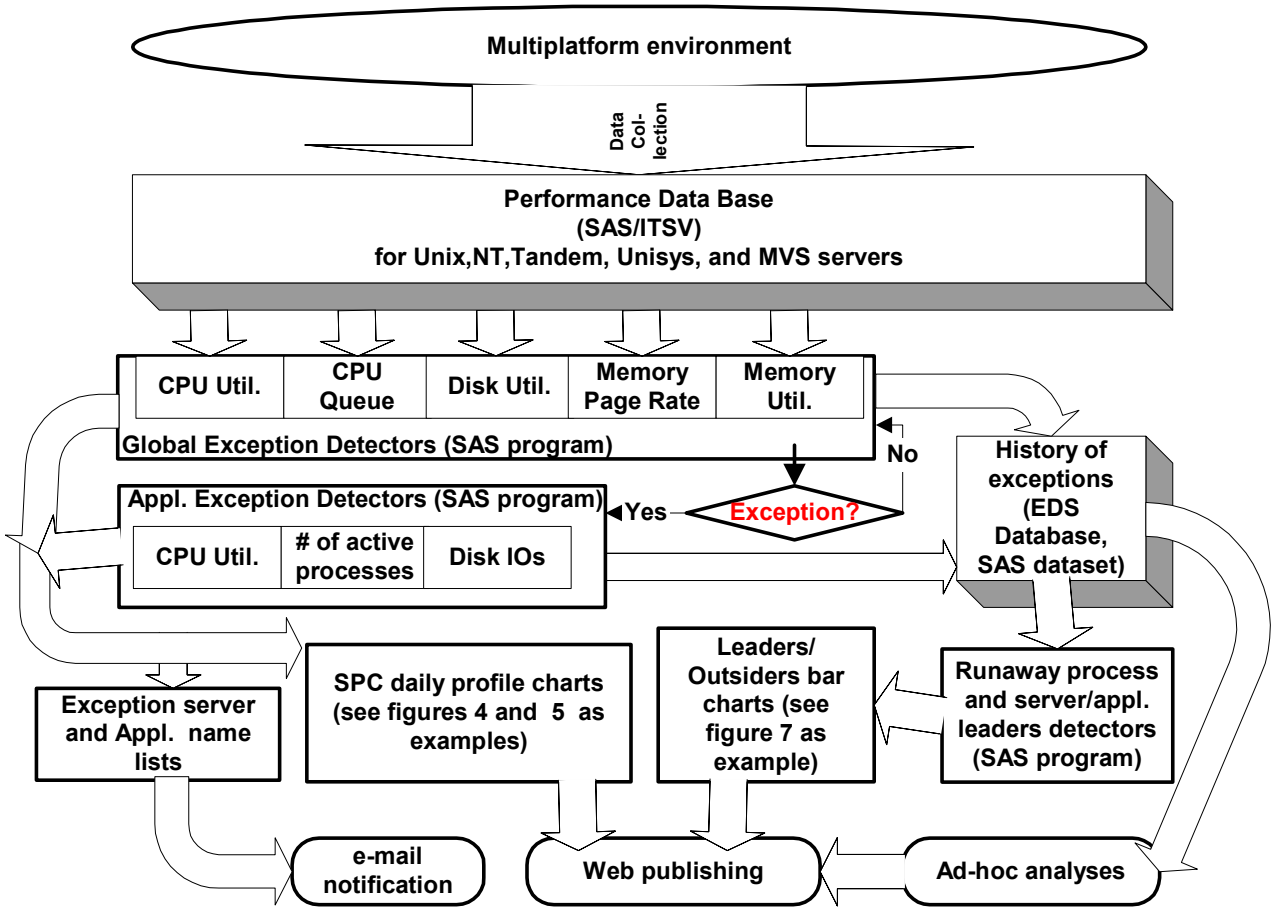

Figure 3 – Exception Detection System Structure

The Exception Detection System outputs lists of server names categorized by the type of exception. These lists are compiled in Statistical Process Control charts for each metric as shown in Figure 4 for CPU utilization.

When a Global Metric exception is detected, the system starts a similar procedure of scanning application level data to detect any particular workloads that also had exceptions and were responsible for the global exception.

Currently the system supports only three application level metrics: CPU utilization, number of active processes (which is related to global CPU queue), and Disk IO. The output of the procedure is the list of application names categorized by the type of exceptions.

Both lists (global and application exceptions) are automatically forwarded by email to individual capacity planning analysts and performance engineers.

The Leaders/Outsiders bar charts and runaway process server lists are also emailed every day to appropriate analysts and managers.

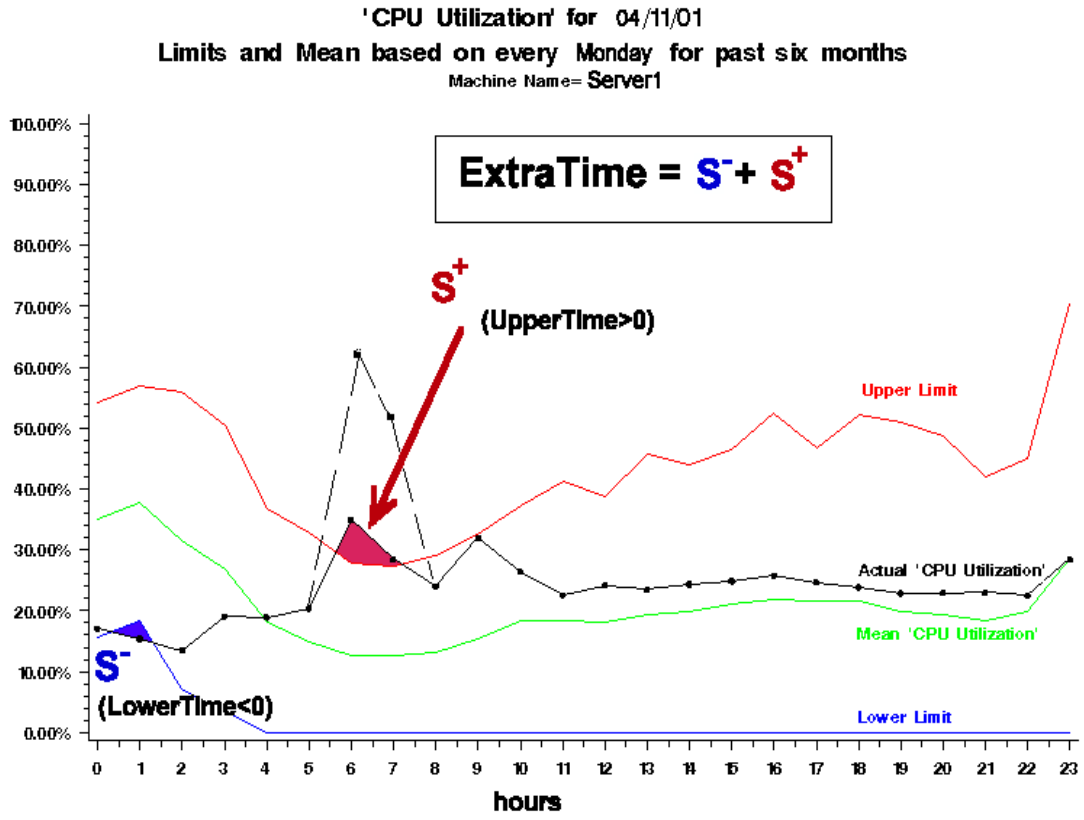

Figure 4 – Statistical process control chart
and example of ExtraVolume metric

4. Notification and Web Publishing

The notification email text has three sections:

- Exception list of servers that had exceptions for a particular metric. Each metric has its own list. In front of each server name, there is a sublist of application names that had exceptions as well for immediate identification of the critical workload.
- Null data list with servers that did not have any performance data. This is an indication of a data delivery problem.
- Insufficient data list. The server might be on this list if the number of observations is less than a certain quantity (empirical rule is "< 6"). In this case, the system cannot make a statistically adequate detection.

The lists may be sorted by platforms, business areas, persons (planners, engineers, managers), and so on.

In the current environment, there are more than 1000 servers, each server having up to 20 workloads. Responsibility for these servers is divided among five planners. These planners receive daily emails noting the exception from the previous day (from 3 to 12 server names a day), which provides them a good start for analyzing performance issues and possible forthcoming capacity issues.

To get detailed information of the server's behavior for the previous day, the system publishes an SPC chart on the INTRANET Web site for each exception, as shown in Figure 4. The Upper and

Lower Limits are calculated as 3 standard deviations from average. A quick analysis of the chart allows the analyst to identify immediately the part of the day where the limits were exceeded.

The system does not automatically generate SPC charts for application exceptions but can produce them by request. An example of the chart is shown in Figure 5.

The example shows that EDS captured a CPU utilization exception at about 9:00 PM on server A, and it found that the application "Workload B" had an exception as well. By special request, the second SPC chart was built to prove that fact. Based on this chart, the reason of the exception can be recognized easily. It was an unusual shift of the Workload B execution.

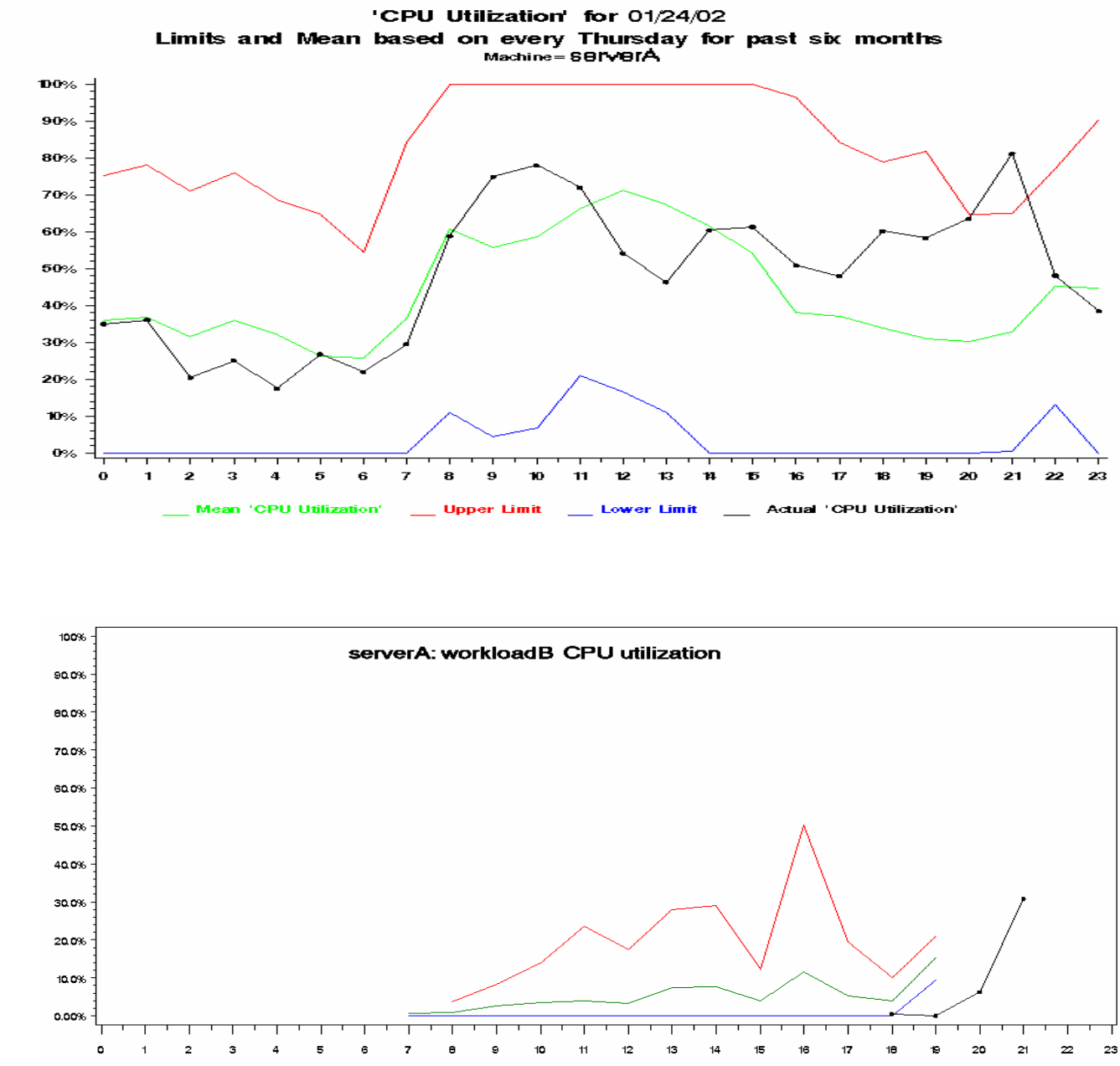

Figure 5 – Global and application CPU utilization SPC charts.

5. Exception Database

The history of exceptions can be used for analysis to show longer trends of metrics outside the third standard deviation, which can indicate system resource problems, server load growth (or reduction), or seasonal deviation. The EDS database was developed to store this data; it is actually a log file in which the exception detector records each occurrence.

The EDS database is a SAS dataset with the following structure:

NAME - server name;
DAYMEAN - metric daily average;
FREQ - number of weekdays in the server's data history (must be >6);
NUP - number of upper limit exceptions;
NLOW - number of lower limit exceptions;
DATE - exception appearance date;
PLATFORM - server configuration;
METRIC - performance metric name.

The two numeric fields NUP and NLOW are useful when trying to determine the duration of unusual stress on a server; servers that had the most extreme exceptions for a certain period; and servers that required more (or fewer) resources overall during that day.

The example of the table in Figure 6 shows the records sorted by METRIC. In addition to statistical exceptions, it shows that some servers had problems with the performance data collection. For example, server20 did not have data at all (EDS puts that name in "Null data" list), and server11 had too few observations (EDS puts it in "insufficient data" list).

To keep application level exceptions, the additional table is used with similar structures. Instead of PLATFORM, the table has the WORKLOAD field to keep the name of the application that had an exception. Both tables can be linked by NAME and DATE to have a normalized relational structure.

NAME	DAYMEAN	FREQ	NUP	NLOW	DATE	PLATFORM	METRIC	S+	S-	Extra Volume = (S+)+(S-)	Unit to measure ExtraVolume data
server19	1.70	25	4	0	10/17	unix SUN	CPUqueue	4.5		4.5	length
server2	3.64	12	0	20	10/17	unix HP	CPUqueue		5.7	5.7	length
server3	17.46	24	5	0	10/17	unix HP	CPUqueue	12.1		12.1	length
server11	342.77	15	3	0	10/17	unix HP	DiskIO	2945		2945.0	# of I/O
server12	3650.58	15	9	0	10/17	unix IBM	DiskIO	38455		38455.0	# of I/O
server20	.	8			12/1	unix IBM	CPUutil			0.0	CPU sec
server13	0.73	25	2	0	11/21	unix IBM	CPUutil	274		274.0	CPU sec
server2	0.32	24	4	0	11/21	unix HP	CPUutil	5973		5973.0	CPU sec
server15	0.55	25	2	9	11/20	MVS	CPUutil	34	3600.8	3634.8	CPU sec
server16	0.99	23	5	0	11/24	unix SUN	MEMutil	239		239.0	Kb
server17	0.61	20	0	8	11/24	unisys	MEMutil		998	998.0	Kb
server18	0.53	21	0	6	11/24	tandem	MEMutil		490	490.0	Kb
server11	0.66	3			12/1	unix HP	MEMutil			0.0	Kb

Figure 6 – EDS database example

6. "Extra Volume" Metric

If the Exception Detection System only kept information about the number of times historical trends were exceeded, then the exception trend accuracy would be very low. For example, in the SPC chart in Figure 4, server1 had

NUP = 2 (for 6:00 AM and 7:00 AM) and
NLOW = 1 (for 1:00 AM).

But if the real value of CPU utilization for 7:00 AM was not 30%, but 90%, the record in the EDS database would be the same although the exception would be more significant.

To increase the accuracy of this approach, a derived system performance metric should be added in the EDS database. Rather than simply counting the number of exceptions, it is necessary to calculate the square between the limit curve and the actual data curve (see Figure 4). In the case of exceeding the upper historical limit, the area would be positive (call it UpperVolume, or S+ in Figures 4 and 6); it would be negative if the lower historical limit were exceeded (call it LowerVolume, or S- in Figures 4 and 6). The best metric to record would be the sum of those values:

ExtraVolume=UpperVolume+LowerVolume

This metric is an integrative characteristic of the exceptions that happens for the day, and it has a simple physical meaning that depends on the source (parent) metric. For example, if the parent metric is CPU utilization, ExtraVolume will be the daily CPU time (ExtraTime in Figure 4) that the server has taken more than a standard deviation. If the parent metric is CPU run queue, ExtraVolume will be the daily extra queue length versus the usual queue, and so on.

As with LowerVolume and UpperVolume, the ExtraVolume metric might be less or more than zero.

If the server showed a positive value for the last day, it means more capacity was used on the server than in the past. In the same way, if the server showed a negative ExtraVolume metric, less capacity was used than usual. Those metrics can be summarized by day, week, or month, which will provide a quantitative estimation of server behavior for a certain period.

Based on this method, the system automatically produces this calculation for the last day and records that in the EDS database using S+ and S- fields as shown in Figure 6. The database also has the calculated ExtraVolume metric. This data is used for generating Leaders/Outsiders charts for the last day, last week, and last month, and for publishing the bar charts as shown in Figure 7.

Figure 7 shows the top 10 servers that had the maximum CPU time used beyond a standard deviation for the last week (CPU utilization ExtraVolume >0).

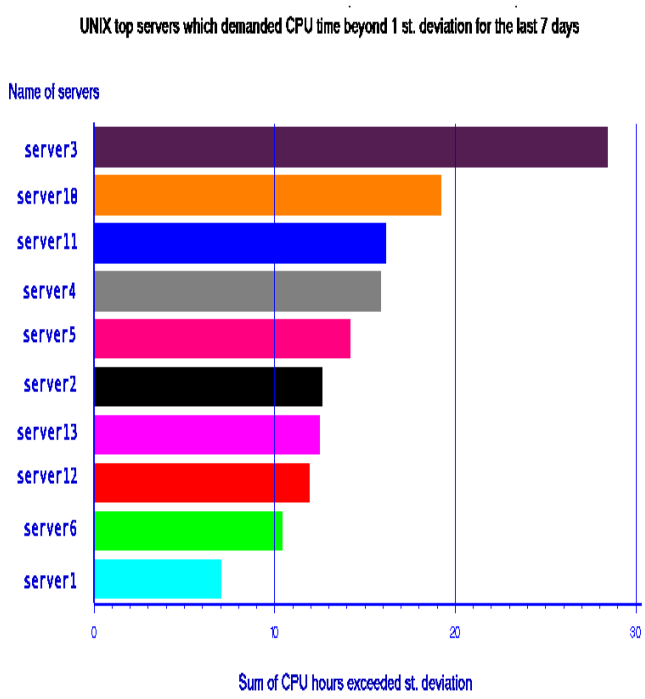

Figure 7 – Leaders bar charts

Similar charts can be generated to show the opposite end of the server list and to demonstrate the top 10 servers with CPU time below a standard deviation (ExtraVolume<0).

The Application table in the EDS database also has the ExtraVolume type data fields from which similar charts can be generated. But it makes sense to do only the top 5 exceptional workloads for each particular exceptional server.

7. Ad-hoc analyses against EDS database

One example of an interesting analysis against the EDS database was already discussed in the last CMG conference. In the previous paper about EDS [3], there was an example on how to compare the statistical exceptions of the servers with different "horsepower" configurations. CPU utilization ExtraVolume was suggested for recalculating to abstract transaction rate using a benchmark such as TPM or SPEC.

Another analysis was done to estimate the impact on a big SUN server during the Christmas holiday shopping season. Figure 8 shows the result of this analysis. The seasonal increase of the server usage can be considered as a natural stress test. The EDS database can help to take advantage of this "free of charge" test. Usually during the stress test the various subsystems, such as disk, memory, or CPU, may react differently. If the system is well tuned for supported business, all metrics should present statistical exceptions almost synchronously. Otherwise the metrics that are least frequently presented in the exception database show which subsystems are underutilized, and the frequently listed metrics show the overutilized subsystems.

Such capacity imbalance can be seen in the example in Figure 8. The server did not have CPU utilization or run queue metrics exceptions at all. This means that some metrics were within statistical thresholds while others were not.

As shown in Figure 8, Disk and Memory metrics did have positive exceptions during the stress period. Just after the end of the holiday season, they had negative exceptions. The last fact reflects the process of getting back to normal resource usage.

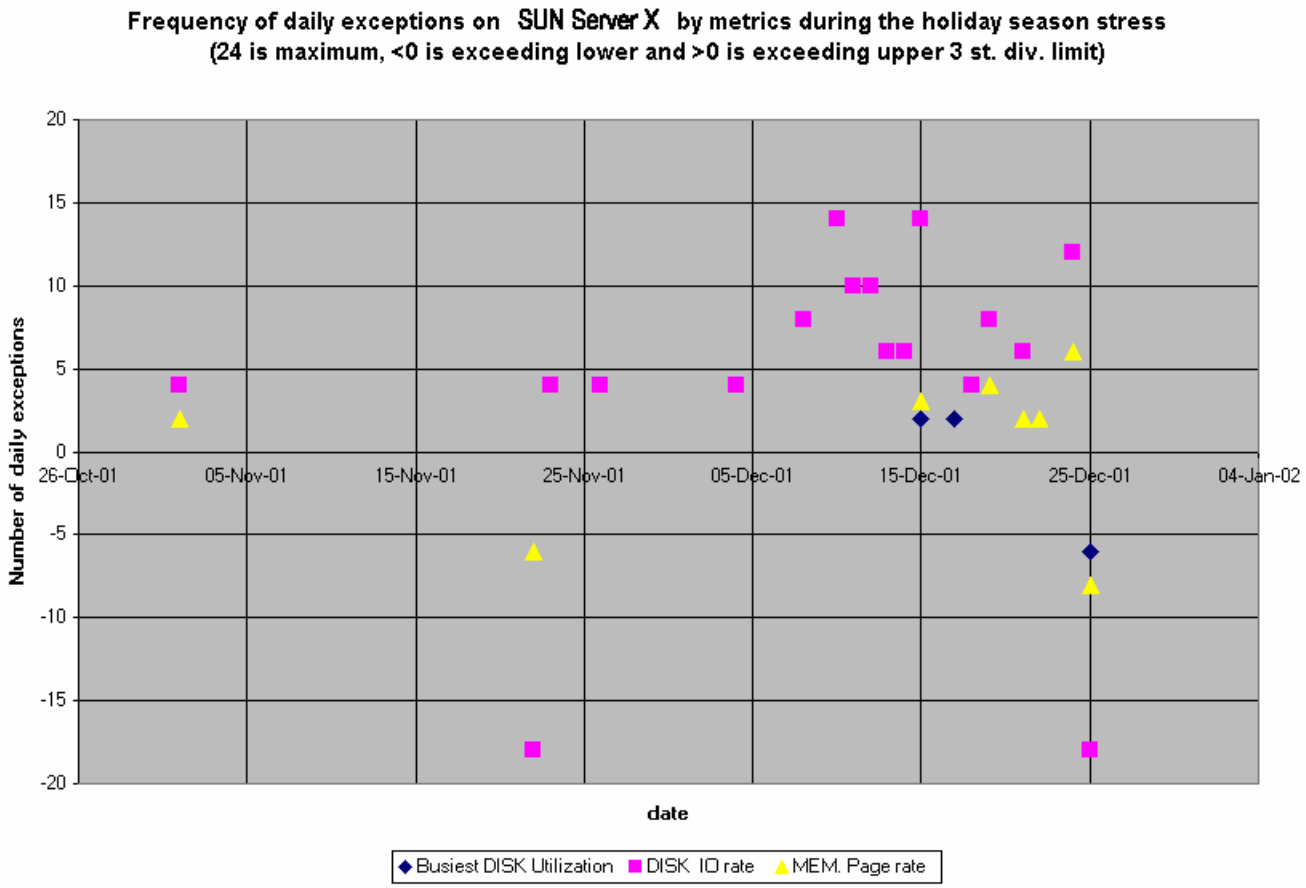

Figure 8 – Exception history analysis

This analysis has the following conclusions:

A. The usage of the server's resources is not balanced.
B. CPU subsystem has excess capacity.
C. Disk subsystem mostly experienced the impact. It is a possible performance and/or capacity bottleneck.
D. Memory page rate had a few exceptions, which probably correlate to Disk I/O activity, and is not a concern.

A similar analysis can be done against application level exception data. A hidden and very interesting (and potentially dangerous for servers) pattern of an application's behavior can be discovered. But that is the subject of another discussion . . ..

8. Summary

The Exception Detection System was developed as a combination of the classical MASF technique and some new ideas such as an EDS database to keep a history of exceptions. The system uses some new integrative metrics such as ExtraVolume to better analyze and plan unusual server resource consumption. In addition to global exceptions the system can capture application level statistical exceptions to determine which particular workload caused the global one.

The Capacity Planning group at the author's company has been using this system for about two years. The system adequately supports the rapid growth of the company, and it doesn't require buying new analysis software (when using existing SAS tools). The efficiency of this system has helped reduce the reaction time to exceptions and the amount of time needed to prepare exception reports. In addition, the ad-hoc analyses against the exception database have helped to discover the system performance and capacity bottlenecks based on the data of some seasonal workload deviations.

9. References

[1] Krajewski / Ritzman: “Operation Management”, 1990, Addison-Wesley Publishing Company, Inc.

[2] Jeffrey Buzen and Annie Shum: "MASF - Multivariate Adaptive Statistical Filtering", Proceedings of the Computer Measurement Group, 1995, pp. 1-10.

[3] Kevin McLaughlin and Igor Trubin: "Exception Detection System, Based on the Statistical Process Control Concept", Proceedings of the Computer Measurement Group, 2001

[4] Yefim Somin: "Hoarding or Herding: How To Sleep Well on Your Performance Data And Wake up Only for True Alarms ", proceedings of the Computer Measurement Group, 2001