

# Capturing Workload Pathology by Statistical Exception Detection System

Igor Trubin, PhD, Vice Chair SCMG

The paper describes one site's experience of using Multivariate Adaptive Statistical Filtering (MASF) to recognize automatically some common computer system defects such as run-away processes on one or multiple CPUs and memory leaks. A home made SEDS (Statistical Exception Detection System) that captures any global and application level statistical exceptions was modified to recognize, report and alert about those defects.

## 1. Introduction

In 2000 the Statistical Exception Detection System (SEDS) was developed and implemented to support IT Capacity Management process in a large U.S. financial services company. Exploiting the MASF method, SEDS is used for automatic scanning through large volumes of performance data and identifying global metrics measurements that differ significantly from their expected values. The first public introduction of the system was made in CMG 2001 conference [1]. The current Structure of the system is shown on Figure 1.

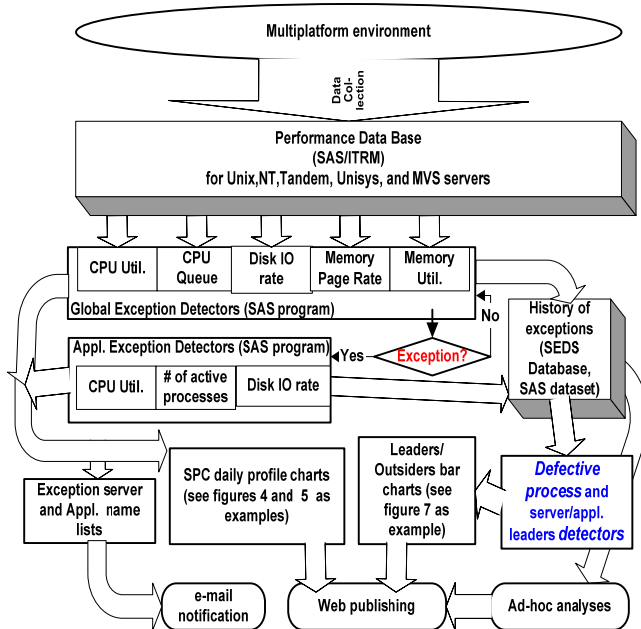


Figure 1 – SEDS structure

When the system went to production the typical exception looked like Figure 2.

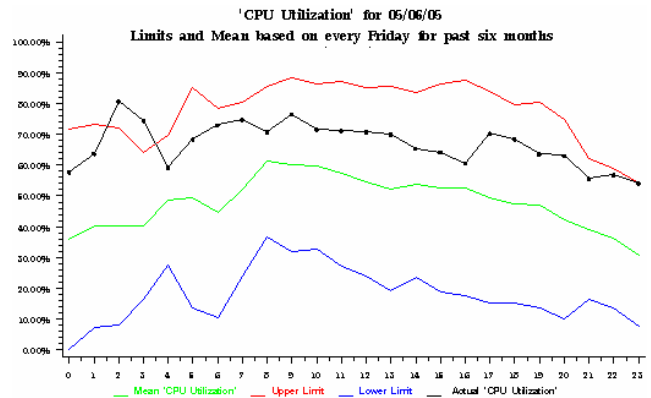


Figure 2 – Typical Exception

From these types of control charts the following facts can be read. Normally, the server (HP-UX N-class 4-way box) is heavily used but more or less stable. All day yesterday it had more than the average load plus three hours of statistically unusual CPU utilization that is represented by exceeding the red upper limit curve (Mean + 3σ). Using this approach, SEDS filters out the real performance issues from thousands of servers and allows the relatively small group of capacity planners to provide proactive performance/capacity management of the large computer farm.

But soon SEDS also captured a different type of issue shown in Figure 3. The performance analysts quickly recognized known system defects such as memory leaks and run-away processes (infinite program loop). In terms of statistical analysis, those exceptions are definitely outliers and SEDS was

modified to exclude them from historical data to avoid an exaggerated upper control limit calculation for future detections.

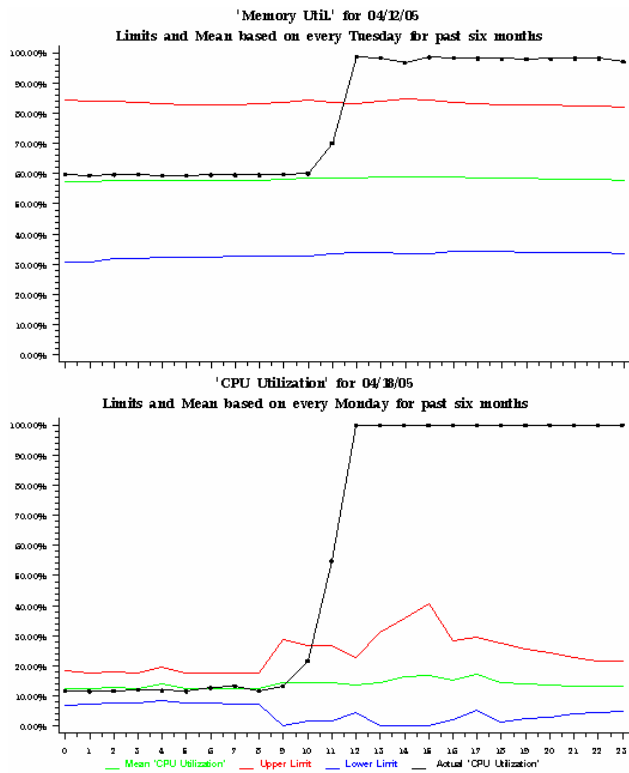


Figure 3 – Non-Typical Exceptions for CPU and Memory Utilization

After a while this sudden SEDS “side effect” becomes a very effective way to detect some common computer system pathologies that is the subject of this paper.

## 2. Workload Pathology Overview

Looking at the classical performance charts any experienced system performance analyst can recognize a memory leak pattern or run-away process situation (Figure 4).

Sometimes it's not so obvious and requires a deeper analysis to recognize a pathology. For instance, a global metric might not clearly show it, but the workload view can help to show the run-away process on a particular application as seen in Figure 5.

No doubt these workload defects cause some problems on the server because of capturing excess resources and capacity. Even if the system performance with a parasite process is still OK, the real resource usage is hidden and the capacity usage picture becomes inaccurate and unpredictable.

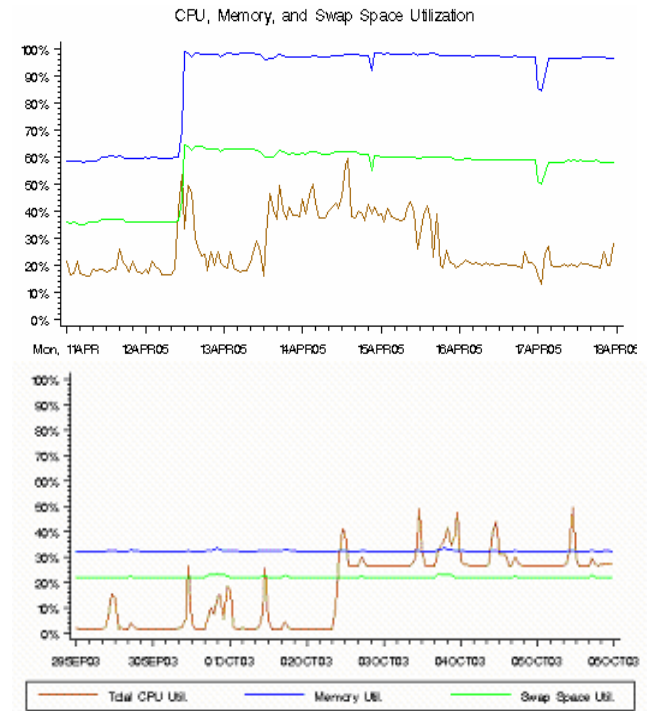


Figure 4 – Typical Pathology Patterns: “Fast” Memory Leak and CPU Run-Away Process on a 4-Way Server

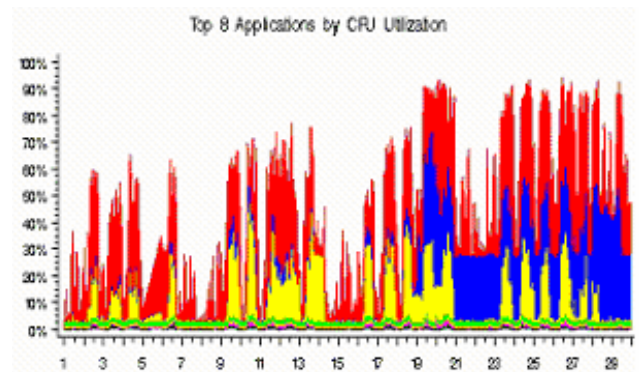


Figure 5 – Monthly Workload Chart Shows Run-Away Application (Marked by Blue Color)

When the automatic future trend chart generator is used [2], the history data with pathologies causes inaccurate prediction, as shown in Figure 6.

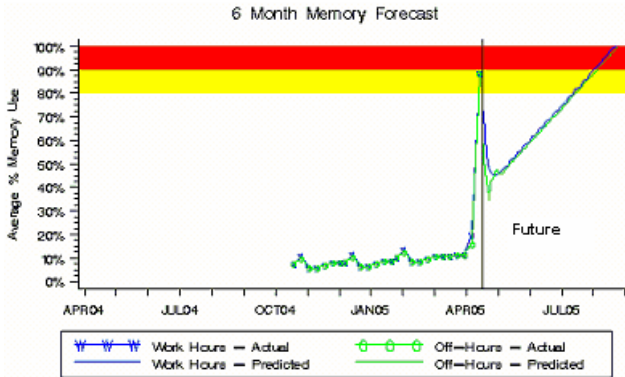


Figure 6 – Memory Leak Issue Spoiled the Forecast

The problem of system pathology recognition has been discussed numerous times, including some CMG presentations.

For example, there is a good definition of a memory leak in one CMG 2001 paper [3]: *“Memory leaks are when the application acquires memory to create objects and then ‘forgets’ to release the memory for reuse. As a result, the application memory usage continues to grow as the application executes”.*

Another 2004 CMG paper [4] has listed a set of rules of thumb to develop automated analysis of computer subsystem “hot-spots”, including memory leaks and run-away processes that considers as some of memory and process “hot-spots”.

One more 2004 CMG paper [5] underlines the necessity of adding to IT service management tools an *“intelligent pattern recognition to look for the pathology of processes (such as memory leaks or program loops)”*.

The best research in this field has been recently done and published for the 2003 CMG conference by Ron Kaminski [6]. An excellent computer system defects classification was presented in this study as well the detailed capturing algorithms.

The advantage of using the statistical process control technique to recognize this pathology is simplicity. SEDS can do it without any complex

analytical algorithms since any pathology is treated as statistically unusual behavior and easily can be captured on the more or less stable production environment.

Also SEDS has a special database to keep all exceptions as shown on Figure 1. This database has a table to keep separately all exceptions about possible pathology cases.

### 3. Capturing Workload Pathologies by SEDS

In the first SEDS implementation, there were extremely simple criteria to add an exception to that table:

*Several hours in a row the CPU or/and Memory utilization was more than some very high threshold (from 99.5% for UNIX and 95% for Wintel)*

If the exception had been added to that table, SEDS’s e-mail notification in addition to the list of servers with general exceptions will create a separate server list with possible run-away or/and memory leak patterns. SEDS produces this type of smart alarm every morning and sends to capacity planners or performance analysts.

On the SEDS web report as shown in figure 7, the server with a possible pathology will be colored red. The line with the server name has links to control charts as well as classical performance charts that allow a user to make additional analysis and determine an appropriate action plan to address this issue. Other exceptions are marked by other colors as explained on the Figure’s legend.

SEDS creates the data report based on any data source available in SAS/ITRM. In this case the pathology can be detected from BMC BGS type of data, Concord eHealth data or HP Vantage Point data (MeasureWare –MW).

If the application (workload) level data is available, in case an exception is detected, SEDS generates a control chart for any workload definitions, which helps identify immediately what application or process is responsible for the problem. The example is in Figure 8.

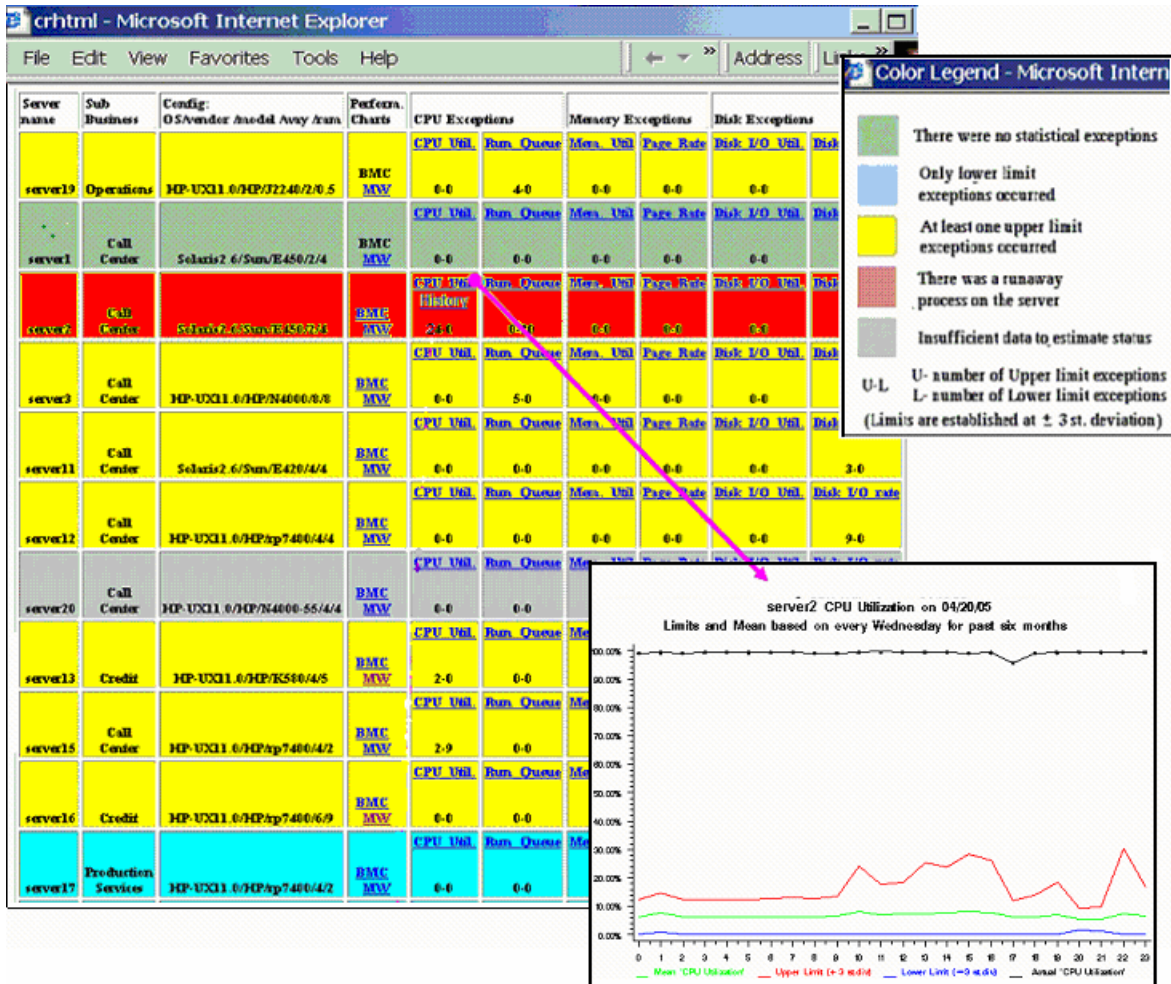


Figure 7 – SEDS Web Report

#### 4. What If a Run-Away Process Did not Consume the Entire Machine?

Indeed, another typical situation on an SMP type of server is when some process takes not all CPUs but only some of them as shown in Figures 4, 5 and 8.

Any of these situations will be captured as statistical exceptions for sure, but to filter out that pathology the following non-statistical algorithm was added to SEDS.

Let's assume:

- U<sub>h</sub>** - each yesterday's hour CPU util.
- U<sub>ah</sub>** - the same hour six-month average of CPU util.
- N<sub>cpu</sub>** - number of CPUs

The server most likely had a runaway process at least on one CPU, if for each hour yesterday (24 hours in a row) the following condition is true:

$$U_h > U_{ah} + 100\% / N_{cpu}$$

For example, on a 4-CPU system, this will highlight a process consuming an additional  $100\%/4 = 25\%$  of system capacity.

The server that matches this criterion will be added to the pathology table of the SEDS database. For better visual recognition of this type of an issue, SEDS publishes the list of servers with run-away processes as the bar chart as shown in Figure 9, where X axis is the daily CPU utilization average. If all of the CPU resource was used, the bar value will be close to 100%. If not, it's going to be significantly less than 100%. For instance, for an underutilized server it might be slightly bigger than 50% for a 2-way server presented on Figure 8.

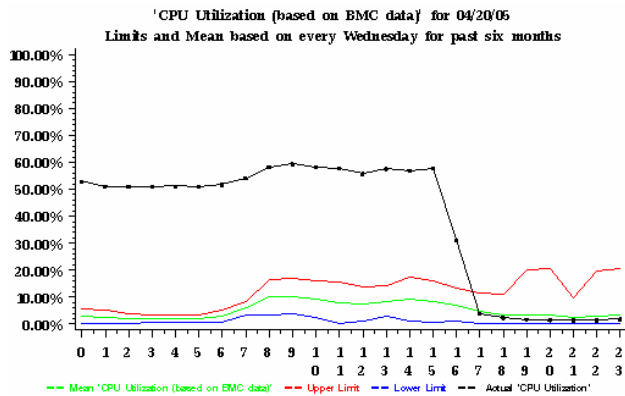


Figure 8 – Run-Away Workload on One CPU of Two

Figure 8 also shows confirmation data that the run-away process was successfully killed around 16:30.

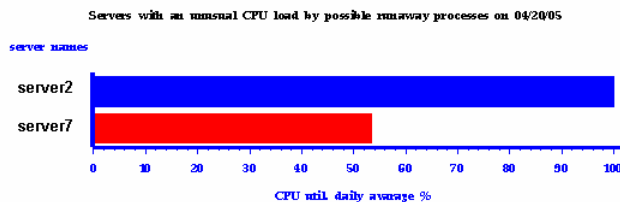


Figure 9 – Run-Away Server List

### 5. Recognition of Other Complex Defects

The current version of the SEDS still cannot filter out some other complex defects such as a slow (long term) increases (“ramps”). This typical situation is presented on the Figure 10 where memory utilization has slow growing component representing some process with a memory leak. SEDS captured this server with a memory utilization exception as shown in the second charts on Figure 10.

This case has a distinct pattern, which is a very small deviation from the statistical upper limit and could be probably formulized and added to the SEDS filter to put these cases to the Pathology table.

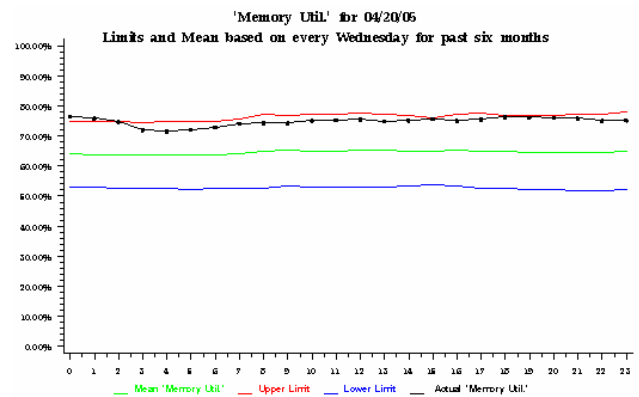
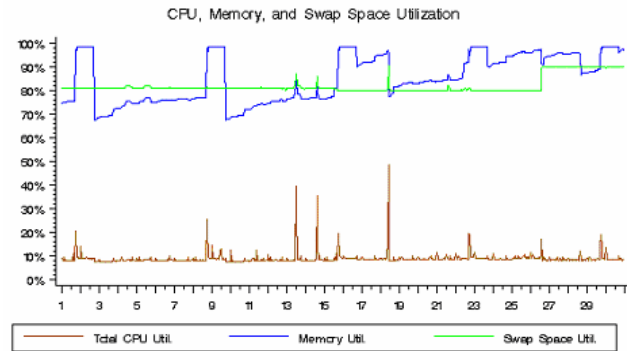


Figure 10 – Complex Memory Leak Example; Classical Performance Chart and Control Chart from SEDS

### 6. SEDS Exception Database Usage

The history of exceptions can be used for other analysis to show longer trends of metrics outside the three standard deviations range. These can indicate system resource problems, server load growth (or reduction), seasonal deviation [7] or possible pathologic situation. The SEDS database was developed to store such data; it is actually a log file in which the exception detector records each occurrence.

Even if the exception was not put into the special pathology table, the history of them can uncover the hidden pathology. Looking at the chart in Figure 11,



## 7. Summary

The workload pathology can be discovered by statistical tools when that is normally very unusual system behavior. Then that should be excluded or masked from healthy historical data as outliers to avoid statistically incorrect conclusions.

In most cases an additional non-statistical filter can be added to the MASF-type detector to recognize a typical defect such as run-away process (infinite program loop) or memory leak. Even if this filter does not recognize a defect, the exception should be the subject of additional analysis and the hidden defect can be discovered. The history of previous exceptions can help to identify that.

This approach was implemented to the SAS based Statistical Exception Detection System, and it has been successfully used for 5 years to automatically report about possible workload pathologies on a large computer farm.

- [1] Igor Trubin, Kevin McLaughlin; "[Exception Detection System, Based on the Statistical Process Control Concept](#)", Proceedings of the Computer Measurement Group, 2001
- [2] Igor Trubin, Linwood Merritt; "[2003 - Disk Subsystem Capacity Management Based on Business Drivers I/O Performance Metrics and MASF](#)", Proceedings of the Computer Measurement Group, 2003
- [3] Jack Woolley; "[How to Validate Application Quality, Performance and Scalability](#) ", Proceedings of the Computer Measurement Group, 2001
- [4] Rogers, Russell; "[Ubiquitous Data Collection in a Large Distributed Environment](#) ", Proceedings of the Computer Measurement Group, 2004
- [5] Adam Grummitt, "[Corporate Performance Management as a Pragmatic Process in an ITIL World](#) ", Proceedings of the Computer Measurement Group, 2004
- [6] Ron Kaminski, "[Automating Process and Workload Pathology Detection](#)", Proceedings of the Computer Measurement Group, 2003
- [7] Igor Trubin, "[Global and Application Levels Exception Detection System, Based on MASF Technique](#)", Proceedings of the Computer Measurement Group, 2002

## 8. References