Exception Based Modeling and Forecasting

Igor Trubin, Ph.D., SunTrust Bank

How often does the need arise for modeling and forecasting? Should it be done manually by ad-hoc, by project requests or automatically? What tools and techniques are best for that? When is a trending forecast enough and when is a correlation with business drivers required? The answers to these questions are presented in this paper. The capacity management system should automatically provide a small list of resources that need to be modeled or forecasted; a simple spreadsheet tool can be used for that. This method is already implemented on the author's environment with thousands of servers.

1. Introduction

One of the essential jobs of a Capacity Planner is to produce forecasts based on current and future situation models. This type of job is a bit dangerous. Just like a weather or stock market forecasting job, if the prediction is correct nobody usually gives you credit for it, but in case a prediction is wrong, everybody notices and blames you!

My first and quite unpleasant experience with this type of activity was during one of my dead-end career paths when I was hired by a stock brokerage company to develop a forecasting system. I passionately tried to predict the unpredictable market, and finally decided to give up because I realized that the management did not really need accuracy, but just nice pictures and graphs to persuade the customers to buy something.

In spite of all those unpleasant side effects of forecasting, I finally found this job to be fascinating once I produced my very first precise forecast shown on Figure 1. That was a rather simple model, based on common benchmarks (TPM) to recalculate UNIX box CPU usage for a few possible upgrade scenarios.

After one of my recommendations was accepted, I collected performance data on the newly upgraded configurations and - bingo - the server worked as I had predicted! That was a moment of truth for me, but of course I did not get any special appreciation from the customer.

(I have previously experienced similar excitement during my teaching career when I would see some sparks of understanding in the eyes of my students. I hope all the readers of this paper have those sparks as well.)

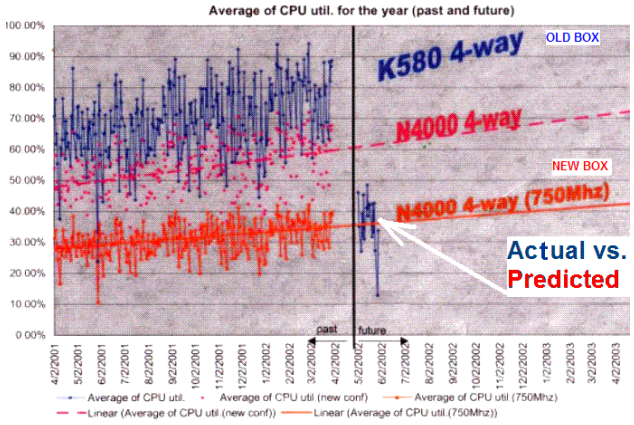

Figure 1 – Old and Good Prediction Example
(Scanned Image from Personal Archive)

Since that first good forecast, our Capacity Planning team kept a special focus on producing a different type of forecasting/trending analysis, automating the process as much as possible. The result was automatically updated trend forecast charts for every server for several metrics and most of the charts looked nice and beautiful, but in most cases were useless as you can see on Figure 2.

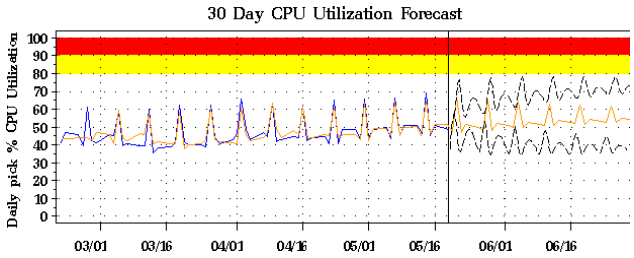

Figure 2 - Beautiful Trend-Forecast Chart

Why? Because in a good environment most of the production boxes are not trending much and you could only enjoy this type of healthy future predictions. As a result, we had to scan all those thousands of charts manually (eyeballing) to select

only those that had real and possibly dangerous trends.

Some of these experiences are discussed previously in CMG papers [1] and [2] where "*… authors used the SAS language to automate the production of resource projections from business driver inputs* (see Figure 3*). If projections are stored in a database, automatic "actual vs. projected" comparisons can also be displayed*…"

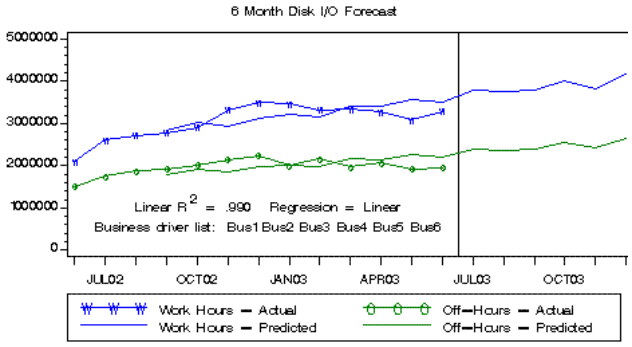

Figure 3 – Business Driver Based Forecast

However, some challenges were reported [2]: "…*the authors' experience is that this approach appears to be more accurate for a single resource such as CPU. When applied to multiple file systems' disk I/O, results may vary greatly*…". And there were really great variations even for CPU resources! All in all, it appeared that the maintenance and usage of the "bulk" forecasting with or without business driver correlations becomes a nightmare.

Living with all those challenges, I dreamed of automating the process of selecting the really dangerous up or down trends of resource consumption and finally have discovered a way of doing so.

2. Basic Forecasting Rules

First of all let's review some basic and obvious rules of how to avoid common mistakes in forecasting models:

- **A.** Historical data should have the **right summarization,** including data for correlation (e.g. business drivers).
- **B.** **Do not mix shifts**: forecasts should be done separately for working days and/or hours or off-shifts.
- **C.** The result depends on the **statistical model chosen**.
- **D.** The starting historical time point should take into account the **significant events** such as hardware upgrades; virtual machines, databases and application migrations; LPAR reconfigurations and so on**.**
- **E.** "Bad" data points should be excluded from historical samples as **"outliers**".

RULE A: "Right Summarization". Data collection must provide very granular data (10 sec -15 min at least). However, for analysis data should be summarized by hour, day, week or month. It is not a good idea to try and produce trend forecast analysis against the raw and very granular data even using good "time series" algorithms. On the other hand, if you have only hourly or daily snap-shots type of data the forecast could be very misleading even after summarization.

If correlation analysis is needed, all data (independent and dependent variables) should be normalized by the same interval, and usually the less granular the summarized data is – the better the correlation that can be found. Below in paragraph five of this paper you can see an example of correlation analysis of CPU usage vs. Web hits that uses 5 min. interval data summarized by days with a very good result (see Figures 16, 17 and 18).

RULE B: "Do not mix shifts". This is a pretty obvious rule. The following real data example on Figure 4 shows the difference between a trend forecast that includes weekends and one that excludes weekends. Note that a "no-weekends" chart forecast reaches the yellow zone sooner!

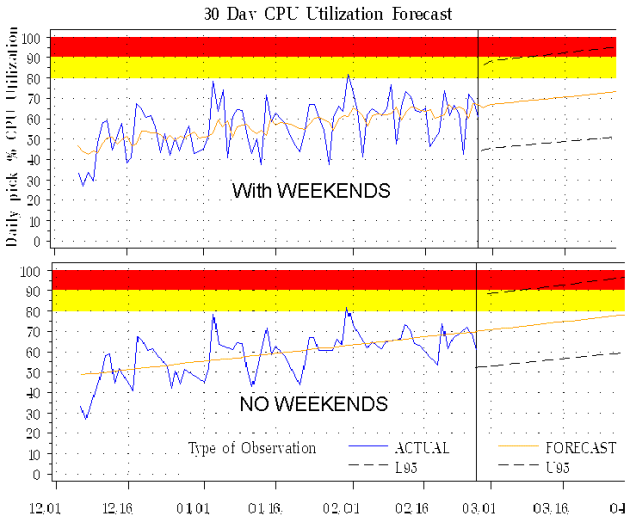

Figure 4 – Trend forecast with and without weekends

RULE C: "Statistical model choice". Let's leave the detailed discussion of this subject to real statisticians/mathematicians and here let's just formulate the basic rule: start with linear trend. Play with other statistical algorithms if they are available and use them only if absolutely necessary.

The chart in Figure 5 shows the same data with different future trends because three different algorithms were used. That figure shows that the 1st method tries to reflect weekly oscillation (seasonality) while others try to keep up along with the most recent trends; the last one is the most aggressive.

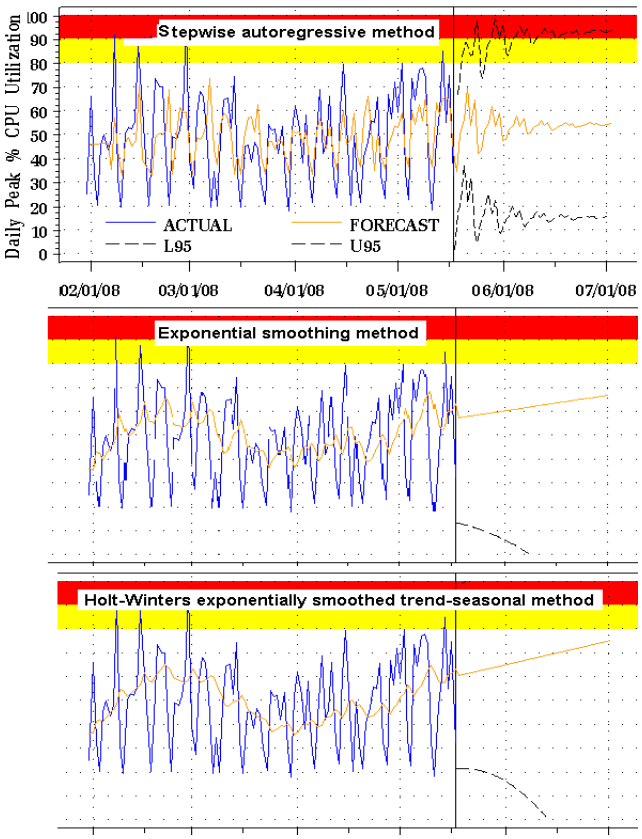

Figure 5 – Different Statistical Forecasting Algorithms Results Comparison

Using SAS language it is easy to play with these models, through changing the "**method**" and "**trend**" parameters on the following proc statement:

```
proc forecast
     data=<actual dataset>
     interval=day lead=30
     method=<STEPAR|EXPO|WINTERS>
     trend=<1|2|3>
     out=<actual&predicted_dataset>;
run;
```

(The **stepar** and **trend=**2 are default values and they mean "*stepwise autoregressive*" with the *"linear trend model")*

RULE D: "Significant Events". The standard forecast procedure (time series algorithm based) might work well where the history is consistently reflecting some natural growth. However, often due to upgrades, workload shifts or consolidations, the historical data consists of phases with different patterns. The forecasting method should be adjusted to take into consideration only the latest phase with a consistent pattern.

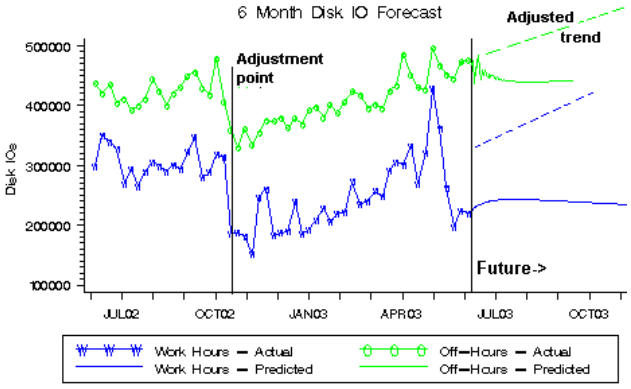

Figure 6 –Trend Forecasts Based on Whole and Partial Data History

For instance, if the history shown in Figure 6 began in October instead of July, the future trend would be more realistic as shown by the dashed lines on the "future" side of the chart.

RULE E: "Outliers". Unfortunately, a server can occasionally experience some pathological workload events such as the run-away processes capturing all spare CPU resources or memory leak situations where the application consumes more memory than it can actually use.

No doubt these workload defects cause some problems for the server. Even if the system's performance with a parasite process is still acceptable, the real resource usage is hidden and the capacity planning for resource usage is unpredictable. When the automatic future trend chart generator is used, the historical data with pathologies causes inaccurate prediction, as shown on Figure 7. To improve the quality of trend-forecast analyses such types of events along with some unexpected outages should be removed from the history as outliers.

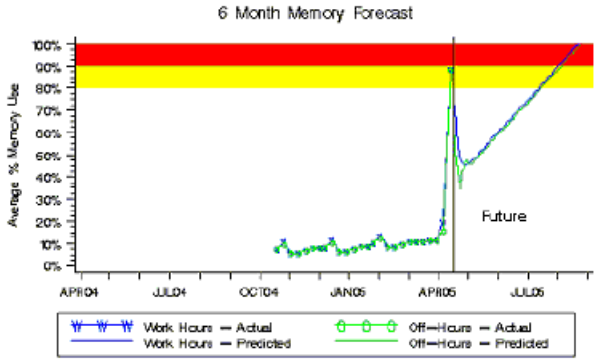

Figure 7 – Memory Leak Issue Spoiled the Forecast

3. Forecasting vs. Exception Detecting

About 7 years ago, the first manager of my first Capacity Planning team faced a tough dilemma: he needed to assign the following tasks, one to me and one to another mathematically well educated analyst:

- Forecasting System development, and
- Exception Detection System development.

Naturally, I dreamed of working on the first task, but unfortunately (or fortunately?) my manager assigned the second task to me because of my mechanical engineering background in which the use of Statistic Process Control methodology is widespread.

My colleague (he is the co-author of my pervious CMG paper [8]) did a great job developing a forecasting system to produce trend-forecast and business correlated forecast charts based on performance data from hundreds of servers and databases. His scripts are still used today to generate high quality graphs. The charts shown in Figures 3, 6 and 7 are based on code he wrote. I learned from him how to produce similar charts and I greatly appreciated his help.

He also implemented some of the basic forecasting rules described above including **RULE D "Significant Events".** He developed scripts to forecast when the future trend of the "database used space" metric intersects with the "database current allocation" metric. His script does this task three times for short, medium and long history sample data to show the best and the worst scenarios. As a result, only a few problematic databases and table spaces show up on his report which is automatically colored yellow or red based on the future date threshold intersections.

It is a known approach to mark a resource as potentially running out of capacity by using future trend intersection with some obvious threshold. However, this approach does not work when the threshold is unknown. Below we will discuss another method that does not have this weakness.

While my colleague was developing the forecasting system, I worked on the Exception Detection System using as a basis some of the CMG publications starting with the first MASF paper [3]. The basic idea I implemented was the following:

Take some representative historical reference data; set it as a baseline and then compare it with the most recent actual data. If the actual data exceeds some statistical thresholds, (e.g. Upper (UCL) and Lower (LCL) Control Limits are mean plus/minus 3 standard deviations), generate an exception (alert via e-mail) and build a control chart like shown on Figure 8 to publish on the web:

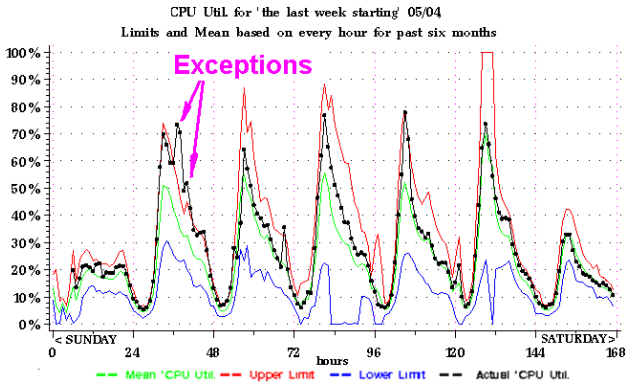

Figure 8 – Exception Detection Control Chart

This weekly control chart is a powerful tool. It shows at a glance not only the last week's worth of hourly data, but also the daily and weekly profiles of the metric's behavior plus a historical view. It actually shows you if the data are trending up or down and when exactly the trend occurs. Gradually, I realized that I had developed a trending analyzer in addition to an Exception Detector!

While improving this approach, I also realized that some of the basic forecasting rules already apply to the Exception Detection System:

- **RULE A: "Summarization"** says that all metrics and subsystems under the Exception Detector should be consistently summarized and the best summarization level is a 6-8 month history of hourly data.

That, for instance, allows you to see where system performance and business driver metrics correlate simply by analyzing control charts.

- **RULE B**: "**Do Not Mix Shifts**" is easily demonstrated by the weekly/hourly control chart because it visualizes the separation of work or peak time and off time.

- **RULE C: "Statistical Model Choice"** means playing with different statistical limits (e.g. 1 st. dev. vs. 3 or more st. dev.) to tune the system and reduce the rate of false positives.

- **RULE D: "Significant Events"** is another important tuning parameter of the system. RULE D is used to determine the depth of the reference set. Even with a constant for the reference set (e.g. 6 months), the Exception Detector has the ability to adjust itself statistically to some events because the historical period follows (moving forward) the actual data and every event will occasionally be older than the oldest day in the reference set.

- **RULE E: "Outliers"** are easily found statistically by the Exception Detector as all workload pathologies are statistically unusual. By adding some (non-statistical) filters to the system, the most severe of these pathologies could and should be excluded from the history to keep the reference set free from outliers. [4]

Finally, to increase the accuracy of the Exception Detector and to reduce the number of the false positive situations (false alerting), a new meta-metric was added - "**ExtraValue"** (**EV**) of exception. Geometrically speaking, it is the area between the actual data curve (black line on a control chart) and the statistical limit curves (red and yellow lines on the control chart) (Figure 9). This metric is basically a magnitude of exception and is always equal to zero, if actual data fluctuates within statistically healthy boundaries. But if EV stays >0 or <0 for a while that means there is an unusual growth (trending up) or drop (trending down), respectively. (This metric was first introduced in a 2001 CMG paper [7].) See **APPENDIX** for a mathematical method for calculating EV.

Having this metric recorded to some exception database allows for in-depth analysis of the system behavior and some examples of this analysis were published in another CMG paper [5].

But the most efficient use of this metric is to filter the top most exceptional resources in terms of unusual usage.

Publishing this top list in some way (e.g. bar charts shown on Figure 9 or 16) along with links to control charts significantly reduces the number of servers that require the focus of Capacity Planning or Performance Management analysts.

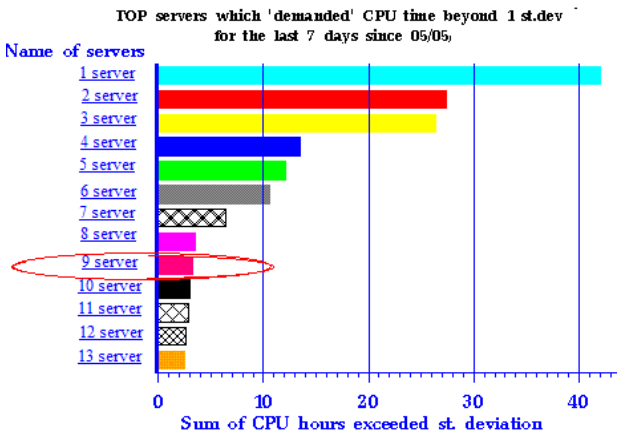

Figure 9 – Top list of servers with unusual CPU usage

4. Exception Based Forecasting

The Exception Detector provides a list of resources with highly unusual consumption. In so doing, the Exception Detector provides a targeted list and helps to apply all of the forecasting rules described below.

Here is the suggested method for implementing that approach:

- The data for the Exception Detector and forecasting system should be the same.
- The trend-forecast charts should be generated only for resources listed in the Exception Detector outputs.
- The data for trending analysis should be freed up from outliers based on Exception Detector pathology filters (e.g., free from run-away and memory leak days or hours).
- The starting time point(s) in the historical data for trending analysis can be found based on exception database data with

"***ExtraValue***" or **EV** metric records, as the most recent negative value of this metric indicates time when the data actually started trending up.

To illustrate how this works, let's look at a couple of case studies with actual data. One day, server number 9 has hit the exception list as shown on Figure 9. Clicking on the control chart link, which the web report should have on the same page, brings up the control chart (shown on Figure 8). That indeed shows some signs of exceptional server behavior:

- Some hourly exceptions occurred on Monday.
- During the entire previous week the actual CPU utilization was slightly higher than average (green mean curve).
- On Friday the upper limit (red curve) reached the 100% threshold for a few hours, which indicates that in the past the actual data might be at 100% level on other Fridays; and Friday average curve is higher than on the other days.

Based on this information, it is a good idea to look at the historical trend. But which metric statistic is better suited for that: daily average or average of peak hour? Let's look at Figure 10 where both statistics are presented:

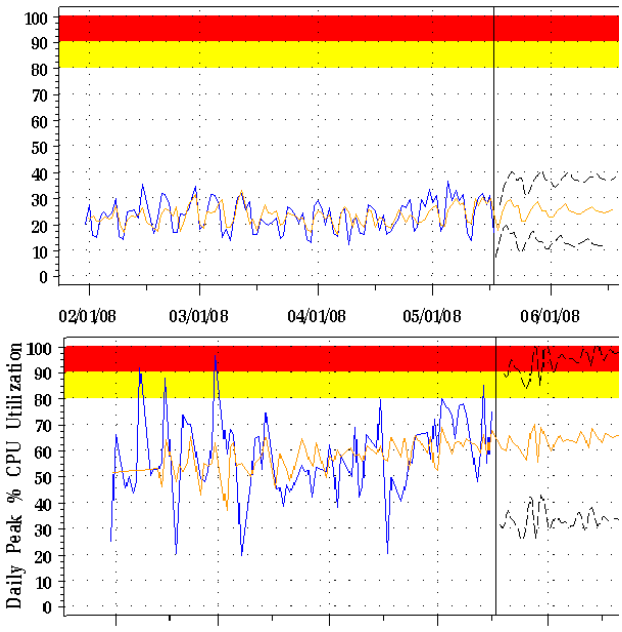

Figure 10 – Trend Forecast Chart of Daily Average vs. Daily Peak Hour Average CPU Utilization

Which presentation is better? It depends. The common recommendation is to use the daily peak for OLTP or web application servers and daily average for back-up and other batch oriented application servers.

But even looking at the Daily Peak trend forecast, the future looks good. Why? Because **RULE D** is not applied and the entire history was used for forecasting. But the history is obviously more complicated, and it's a good idea to analyze only the last part of the history. It can be seen clearly just by eyeballing historical trend chart. Could that decision be done automatically? Certainly, if one looks at the history of "***ExtraValue***" or ***EV*** metric on Figure 12.

Note that the most recent negative value of **ExtraCPUtime** metric (which is the ***EV*** meta-metric derived from CPU utilization metric) points exactly to the point of time when CPU utilization started to grow. Basically, to find a good starting point for analyzing the history one needs to find the roots in the following equation:

$$EV(t) = 0$$

where ***EV*** for this example is "**ExtraCPUtime**" (unusual CPU time used) function of time ***t***. (in days)

If EV metrics are recorded daily in some database, this equation could be easily solved by developing a simple program using one of the standard algorithms. The solution for the real data example shown above is ***t =~ 04/22.***

The final trend-forecast chart is shown on Figure 11. It predicts that after 06/16 the server might be running out of CPU capacity:

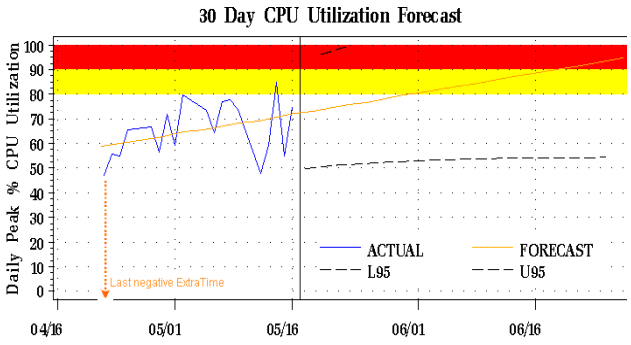

Figure 11 – Corrected Trend Forecast

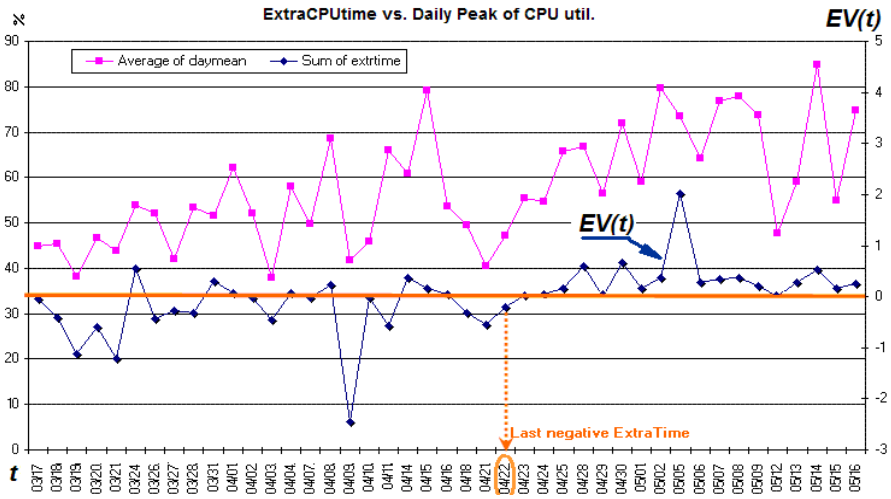

Figure 12 – History of "ExtraValue" Metric (ExtraCPUtime) vs. Daily Average of CPU Utilization

Another example is presented in Figures 13-15. It is interesting that the **EV** metric somewhat mimics the original metric but makes trending more obvious.

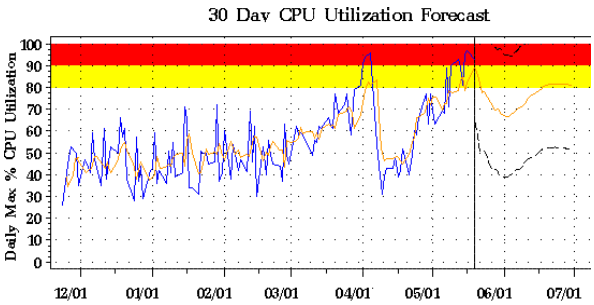

Figure 13 – The Whole History Based Forecast

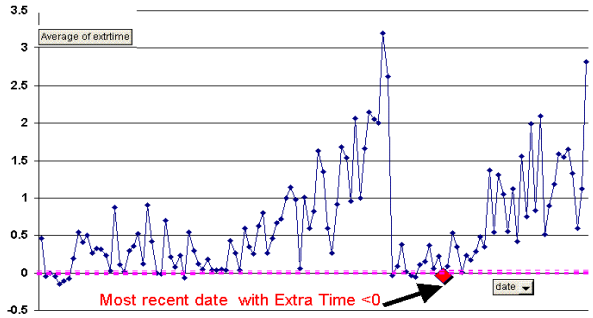

Figure 14 - ExtraTime data analysis

Some oscillations are seen around the most resent negative **EV** value, but that might be tuned out as those cases are using 1 st. dev. threshold which is too sensitive. And of course by the term "recent" one should assume at least a few weeks or more to have enough data for meaningful trend analysis.

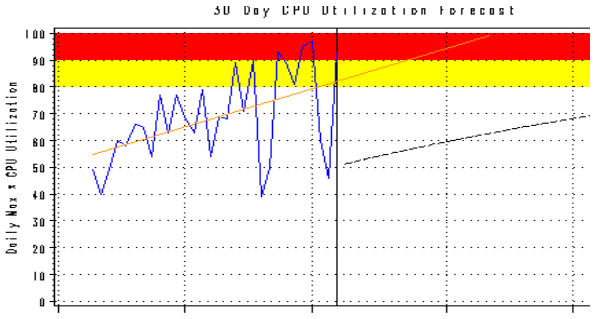

Figure 15 – The Short History Based Trend-forecast

But what about the opposite side of the exceptions list that reports resources with unusually low usage? Yes, the Exceptions Detector publishes that as well and it makes perfect sense to build control and trend-forecast charts for each resource from that list. One example of such a web report is shown on Figure 16 for the VM host from which some virtual machines were recently moved to another host. The Exception Detector captures that event perfectly and gives an analyst the possibility to see how much resource was released.

One of the unique parts of this method is the following. If a metric does not have an obvious threshold (e.g. I/O, paging or Web hits rates) the approach works anyway and the trend-forecast will be built only for resources (e.g. disk, memory or Web application) that recently started dangerously trending up. Additional modeling may be needed to estimate what drives the increase and how to avoid potential problems.

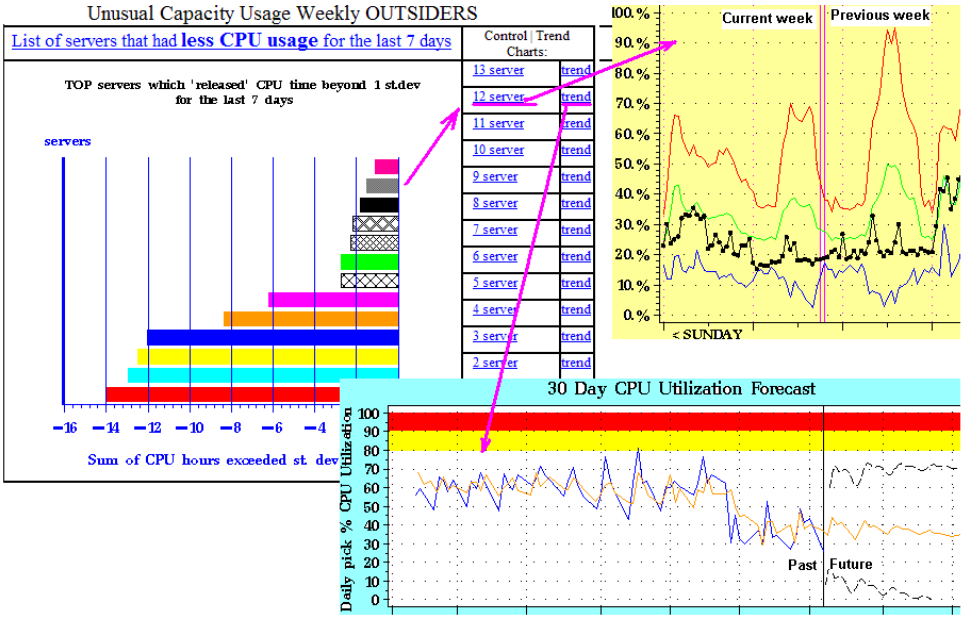

Figure 16 – Web Report about Top Servers that Released CPU Resource

5. Exception Based Modeling and Forecasting

How often do we need to perform modeling? What tools are good for that? Obviously, if there is a project to upgrade hardware or to consolidate resources or consumers (applications, servers, VMs or databases), a good capacity management process assumes that the capacity planner is involved as a project resource and should model various what-if scenarios. The capacity planner is fortunate if he has good data and good modeling tools (for instance, a queuing theory based analytical tool).

A good capacity management service should be able to initiate this type of project. The Exception Detection System and the forecasting based on that system can be very helpful in this process. In most cases, if one has good data and some statistical and capacity management experience, the modeling can be done effectively with a spreadsheet.

Even control charts can be built using just a spreadsheet, as demonstrated in one of the past CMG sessions [6]. Trend analysis (forecasting) can be also done by a spreadsheet. (One of these types of techniques such as ***Forecast(…)*** formula usage was also demonstrated at CMG – [1]). Let's look at some real data in a case study to see how that works. Here is the most recent data for a previous example mentioned in the introduction; see Figure 1.

One day, the Exception Detector notified me that some web application had an unusual number of web hits producing the hits rate control chart similar to the one on Figure 8, but without obvious threshold. The trend forecast for CPU usage was automatically generated for the server hosting this application as shown on Figure 10. Finally, both CPU and Web hits data were downloaded to a Spreadsheet for modeling. Combining those metrics in one scattered chart (Figure 17) shows excellent correlation with R^2=0.96.

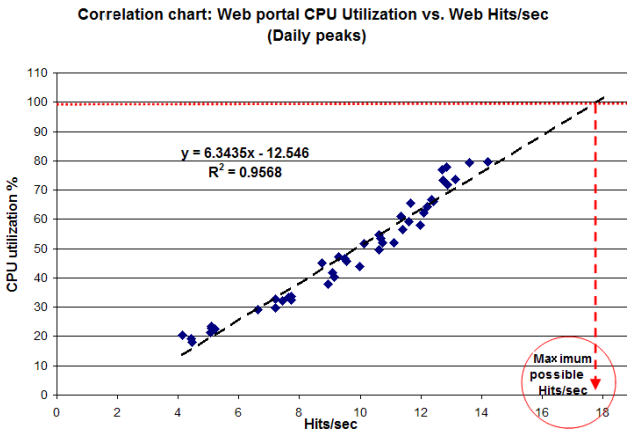

Figure 17 – Correlation between CPU utilization and Web hits rate

This simple correlation model shows us that the maximum number of hits per second that this server can handle is about 18. This is a meaningful result and if the application's support team anticipates a higher hit rate in the near future based on specifications, stress test results, customer behavior, forecast and/or business projections, this server will need more processing capacity to meet the requirements.

The model also shows (Figure 18) that if the pattern of this application usage remains the same, the server will be at capacity in about two months.

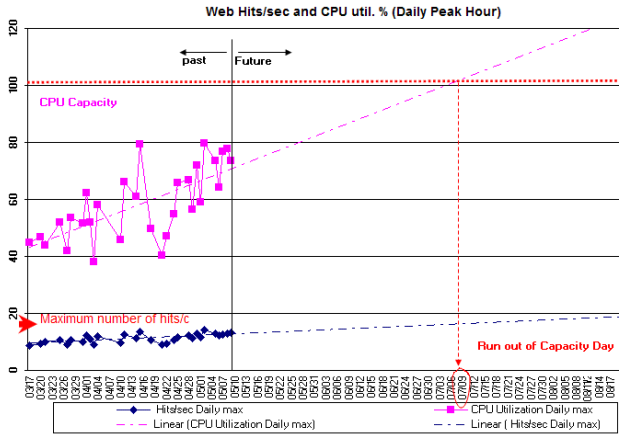

Figure 18 – Current Capacity Usage Projection

But what if the number of anticipated hits is higher? How much additional capacity would be needed? This model can help someone to play out this type of scenario. For instance, Figure 19 shows that adding 25% more processing power to that server gives it the ability to handle 20 hits per sec and that capacity will be reached in a period twice as long in about 4 months.

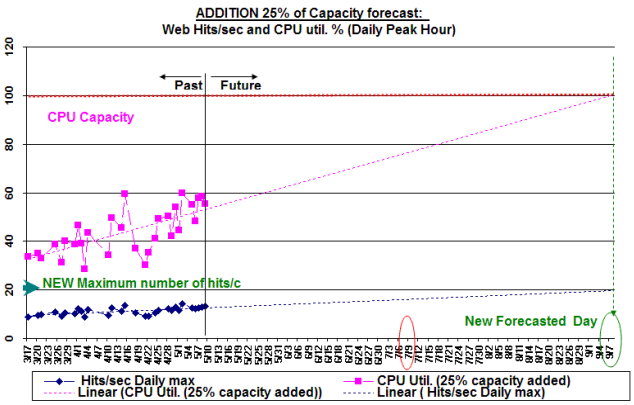

Figure 19 – Proposed Capacity Usage Projection

This model also allows for a more complex analysis. To apply the method explained in the previous paragraph of this paper – calculate the historical starting point based on the most recent negative **EV**, and look at the most recent trend which is apparently the worst case scenario as shown on Figure 20:

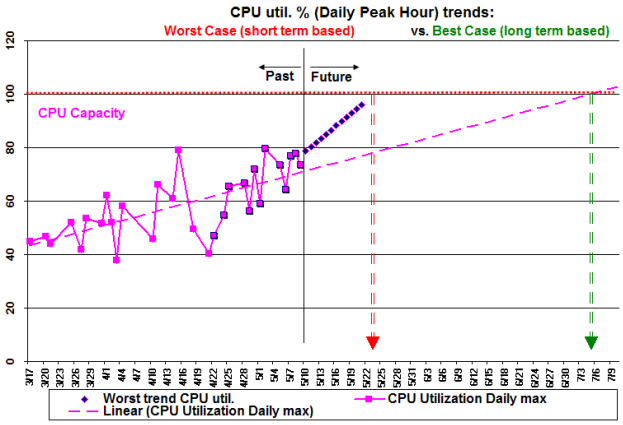

Figure 20 – Worst Case Scenario

What spreadsheet features were used for this modeling exercise? All are pretty standard and simple:

- **Figure 17:** "*XY (scattered)*" standard chart type plus "*Add trendline*" wizard.
- **Figure 18, 19:** Just "*Add trendline*" wizard with the coefficient =0.75 to reflect the proposed capacity increase.
- **Figure 20:** In addition to "Add trendline" wizard the future data were populated by just dragging down the selected range of historical data as shown on Figure 21.

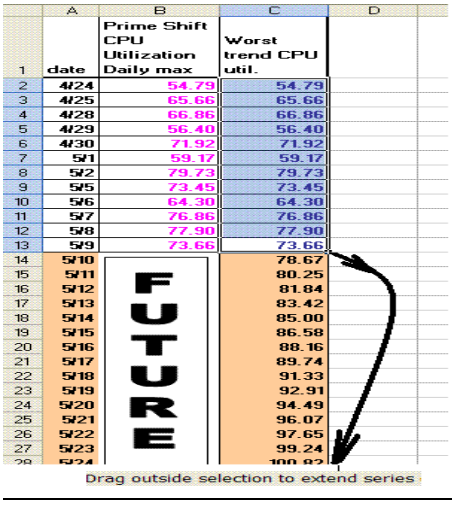

Figure 21 – Future Data Population Technique

6. Summary

Capacity management in a large IT environment should perform forecasting and modeling only when it is really needed. This saves a lot of man-hours and computer resources.

Exception Detection techniques along with an Exception Database could be used to automate the decision making process with regard to what needs to be modeled/forecasted and when.

MASF Control Charts have the ability to uncover some trends showing actual data deviations from an historical baseline. The most recent negative **EV** (**ExtraValue** of exceptions meta-metric first introduced in CMG'01) is an indicator of the moment of time when it is good to start the trending analysis of an historical sample.

A common way of raising a future capacity concern by calculating future trend intersection with some constant threshold does not work for metrics without obvious thresholds. The Statistical Exception Detection approach helps to produce the trending analysis necessary for those cases.

Workload pathologies (e.g. run-aways or memory leaks) should be excluded from an historical sample in order to improve the forecasting. The Exception Detector provides data (dates and hours) for that.

Application data (e.g. web-hits) vs. Server performance data (e.g. CPU utilization) correlation analysis gives a priceless opportunity to add some meaning to forecasting/modeling studies and that analysis can be done using standard spreadsheet tools.

7. References

[1] Merritt, Linwood, ***"Seeing the Forest AND the Trees: Capacity Planning for a Large Number of Servers"***, Proceedings of the United Kingdom Computer Measurement Group, 2003

[2] Merritt, Linwood and Trubin, Igor, Ph. D., ***"Disk Subsystem Capacity Management, Based on Business Drivers, I/O Performance Metrics and MASF"***, CMG2003 Proceedings.

[3] Jeffrey Buzen and Annie Shum, ***"MASF -- Multivariate Adaptive Statistical Filtering"***, CMG1995 Proceedings, pp. 1-10.

[4] Igor Trubin, ***"Capturing Workload Pathology by Statistical Exception Detection System"***, CMG2005 Proceedings.

[5] Igor Trubin, ***"Global and Application Levels Exception Detection System, Based on MASF Technique"***, CMG2002 Proceedings.

[6] Igor Trubin, "**System Management by Exception Part 6**", CMG2006 Proceedings.

[7] Kevin McLaughlin and Igor Trubin, "***Exception Detection System, Based on the Statistical Process Control Concept***", CMG2001 Proceedings.

[8] Igor Trubin and Ray White: "**System Management by Exception, Part Final**", CMG2007 Proceedings.

8. APENDIX:

ExtraVolume meta-metric calculation

ExtraVolume (let's call it **EV)** is basically a magnitude of exception that occurred at a particular time with some metric [7].

Let's look at a 2D model first. The flat and linear model of some performance metric behavior is shown on Figure 22, where ***U*** is the metric, ***t*** is time, ***UCL*** is Upper Control Limit and ***LCL*** is Lower Control Limit.

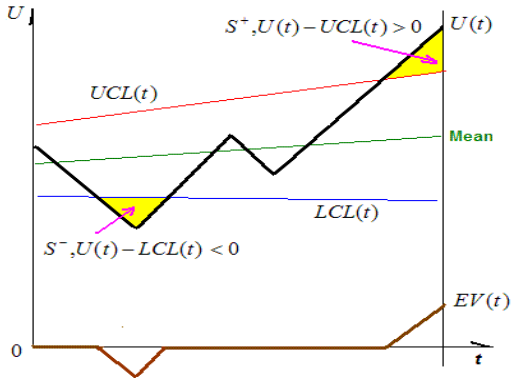

Figure 22 – 2D Model

For this model the formula for EV calculations is

$$EV(t) = \begin{cases} S^+, U(t) - UCL(t) > 0 \\ S^-, U(t) - LCL(t) < 0 \\ 0, UCL(t) \le U(t) \le LCL(t) \end{cases}$$

where S^+ = U(t)-UCL(t) and S^- =U(t)-LCL(t)

Three-dimensional model is more realistic: ***h*** – hours of a day (or week) or days of a week, ***t*** - days (or weeks) and ***U*** – performance metric. By the way, the MASF control chart such as the one shown on Figure 8 is the 2D cut (projection) on a particular week. This 3D model was introduced in last year CMG paper [8] and one example of that 3D view is shown here (Figure 23). For the full 3D model case EV(t) calculations are a bit more complex:

$$EV(t) = S^+ + S^-$$

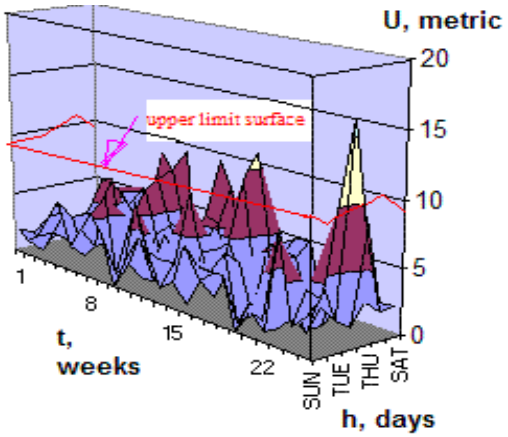

Figure 23 – 3D Model Example:
(Built by Spreadsheet Graph Wizard)

Where

$$S^+ = \begin{cases} \int (U(h,t) - UCL(h,t))dh, U - UCL > 0 \\ 0, U - UCL \le 0 \end{cases}$$

$$S^- = \begin{cases} \int (U(h,t) - LCL(h,t))dh, U - LCL < 0 \\ 0, U - LCL \ge 0 \end{cases}$$

In a general case S^+ and S^- as shown on Figure 24 have the following geometrical meaning: it is the area between the actual data curve (U) and the statistical limit curves (UCL and LCL). They should be calculated only on intervals where the actual metric is outside of the UCL - LCL band. If the metric U is within the band, then both S^+ and S^- as well as EV are equal to zero.

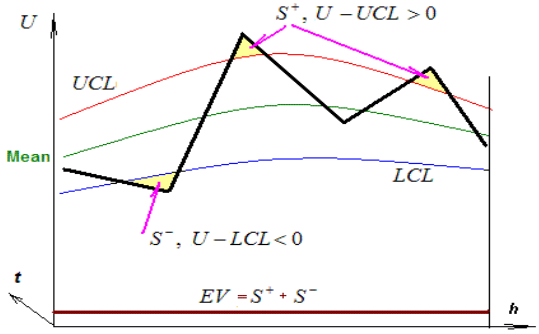

Figure 24 – EV Geometrical Meaning