

Method General Approach Performance Profiling (Method-GAPP)

Msc. G.H. Hendriksen

gerwin.hendriksen@gmail.com

<http://method-gapp.com>

<http://blog.gerwinhendriksen.com/>

March 2011, the Netherlands

v. 1.0

Contents

Contents.....	2
Preface	3
Introduction	4
What is “Method-GAPP”	5
“Method-R”	6
Response time and coherency.....	7
“Method-GAPP” Data Collection	8
“Method-GAPP” Data Synchronization.....	9
“Method-GAPP” Data Modeling	10
“Method-GAPP” Data Mining - Explain.....	15
“Method-GAPP” Data Mining – Model.....	19
“Method-GAPP” Data Mining – Predict.....	24
“Method-GAPP” Data Interpretation.....	25
Conclusions	33
References	34

Preface

After having worked for more than 13 years with Oracle products and specialized in performance, I developed through time a new performance method I baptised “Method-GAPP”. “GAPP” is an abbreviation of “**G**eneral **A**pproach **P**erformance **P**rofiling”. The method makes smart use of underlying queuing models and data mining to find bottlenecks in complex architectures, for specific end-user processes within an enterprise. Another refer can be the “gap” of not having enough tracing data that can be jumped. In base the method uses data which is most of the time already in place, and the method itself is technology independent.

At the Hotsos Symposium 2008 in Dallas (march 2008), I presented “Method-GAPP” the first time. At that time I had already worked over 6 years on the idea of making it possible to do predictions on end-user performance, based on background metrics. The last couple of years the method has been further enhanced and a lot of data modelling was added to the method. These latest steps made predictions on technical architectural changes much better.

When reading this white paper I hope you get a good impression on how the method works. Some of the procedures in the white paper are programmed by me in a package, e.g. the M-curve calculations and the procedure to find the correct M-curve fits. This information can always been obtained on the “Method-GAPP” website, or you can ask me directly.

Although there are in the market some methods and tools, which might have some similarities with “Method-GAPP”, I can ensure you that the method was completely developed by my own effort, and has not been build on other methods / tools. After all I hope you will enjoy the white paper and be impressed by its potential.

Special thanks I like to give to Cary Millsap and Dr. Neil Gunther for their inspiration and support.

Regards,

Gerwin Hendriksen

March 2011

Introduction

The complexity of IT-infrastructures has increased over the years. The number of applications inside an enterprise has increased and this has led to more consolidations, more shared components, more virtualization, etc. Also the complexity of applications has introduced more tiers, like a web sever, an application server, an authentication server, a database server, a SAN, etc. It gets harder and harder to be able to find performance bottlenecks for end-user processes. In the old days we could trace a client process and see what the session was doing on a database server and in this way follow where wait time (queuing time) has went in the underlying infrastructure. Nowadays most applications work via a web based client what makes tracing of the process much harder, although of course not impossible.

If we are in the end able to trace such a client process (the “chain”), we still are confronted with external influences we can’t explain easy. For example a storage or network which is used by several applications (shared) within the enterprise. If we trace we can find out that sometimes the I/O is faster and sometimes the I/O is slower, but we are not able to say where this influence is coming from. It would be nice if there was a way to get more insight into this. In this case “Method-GAPP” can be of great help, because it is able to find also influences of applications and components outside your “chain”.

Sometimes we are in an enterprise and questions are raised if we should do a certain investment in hardware, to solve a certain performance problem. In these cases we have to be careful and make sure we really know what the performance bottleneck actually is for the end-user process, to prevent doing a disinvestment. For these questions the method can give you indicating prediction models of future scenario’s before the actual investment has happened.

In most cases we are not allowed to trace actual in the application, and / or not allowed to put hooks in the application to checkout where our response time was spent within the architecture. This is most of the time the case in applications from third party vendors. Also in these cases the method can be of great help because it can use simply retrieved data to do its predictions and see what is from influence and in which way. Data which can be used:

- Operating system data from all architecture tiers (e.g. sar, perfmon, nmon files)
- Database data (e.g. AWR, ASH, ADDM, stats pack)
- SAN metadata
- Web server metadata
- Response time data from the business processes (e.g. Mercury Loadrunner data, Moniforce data, etc.)
- Instances metadata from BPEL Process manager (e.g. SOA Suite, etc.)
- Business performance indicator data (e.g. BAM tools, etc.)
- Etc.

What is “Method-GAPP”

“Method-GAPP” is an abbreviation of Method “**General Approach Performance Profiling**”. The method makes smart use of underlying queuing models and data mining to find bottlenecks in complex architectures, for specific end-user processes within an enterprise. The method can be used to performance profile business (end-user) processes on high level, but depending on the components you put in, even on detailed level. The method uses a data mining technique to determine which components are the most involved in the end user response time of a business process. Further more are models created, which makes it possible to predict future scenarios. The components should be related to the different components of an architecture “chain” or also contain components from outside the “chain”. In the “chain”, for that specific end-user process, are the directly involved infrastructural components, e.g. the web server, the application server, the database server, the SAN, etc. “Method-GAPP” consist of five important steps, the five “D’s”:

- Data Collection
- Data Synchronization
- Data Modelling
- Data Mining
- Data Interpretation

The mentioned steps will further be described in the white paper. To understand better how “Method-GAPP” works, we will start how “Method-R” from Cary Millsap is describing influences on response time. This way of looking at the response time is an important view point for “Method-GAPP”.

“Method-R”

“Method-R” (*Millsap 2003*) describes how end-user processes are influenced by the “chain” of tiers in a technical infrastructure. In figure 1 is shown how different end-user processes response times (R) are influenced by different tiers of a technical infrastructure. Important to realize is the fact that the response time (R) of each individual tier is determined by the wait / queuing time (Q) and the service time (S), so $R=Q+S$. Further that the end-user process response time (R) is basically the sum of all the individual tier response times, so: $R=R_{AS}+R_{NET}+R_{DB}+R_{SAN}+R_{STOR}$. For example the purple process on the application tier, shows some queuing time (Q) and service time (S) only on the application server, but for the red process all tiers are used. So tunings effort on the database server will not have any effect on the purple process, but will have influence on the red process, etc.

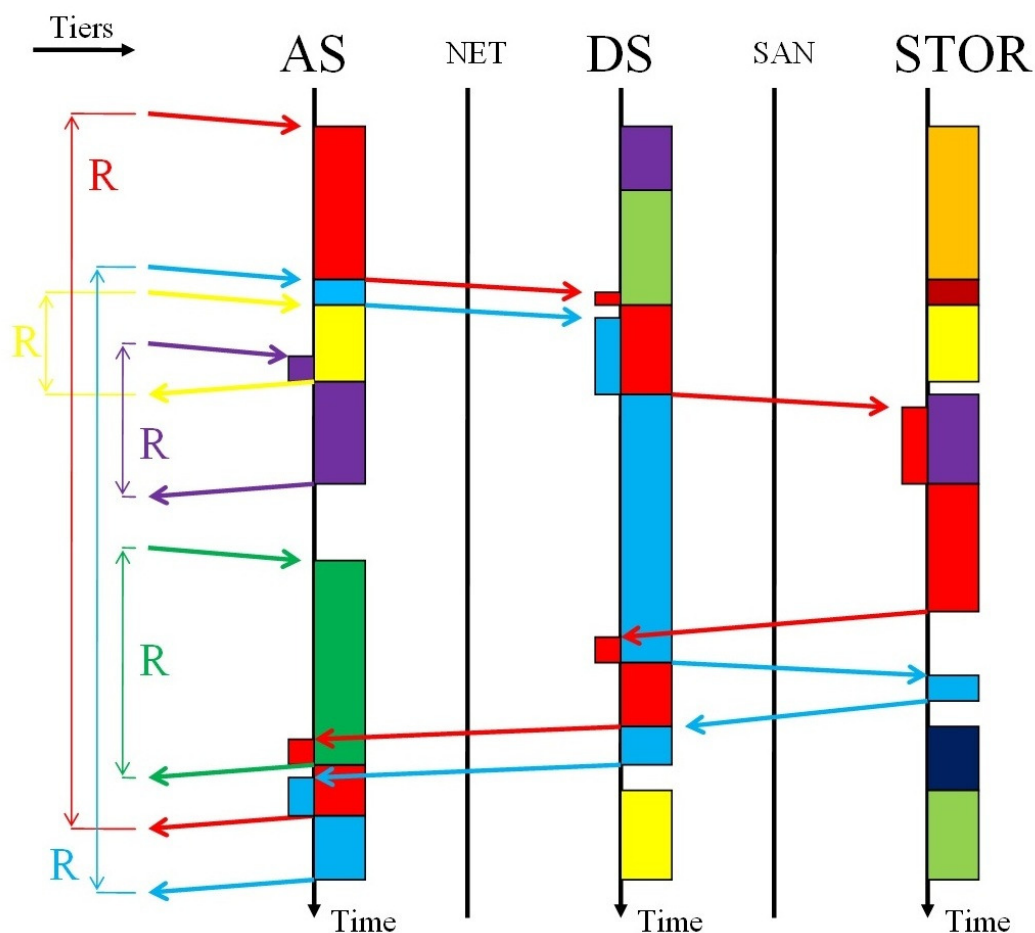


Figure 1: Sequence diagram to make clear how Cary Millsap's Method-R (*Millsap 2003*) is describing how different processes (different colours), are using the architecture in different ways. In the picture are five tiers mentioned: Application Server (AS), Network (NET), Database Server (DS), Storage Area Network (SAN) and Storage (STOR). For the network (NET) and the storage area network (SAN) is only a line drawn, to make the diagram simpler. The blocks on the left side of the line is queuing time (Q) and on the right side of the line service time (S), for each tier.

Response time and coherency

To go a little further for understanding what “Method-GAPP” is doing we have to realize that the response time (R) is not only influenced by the queuing time (Q), but also by the stretching service time (S), due to coherency. Coherency is the effect that although you are serviced, you’re not doing real work, because you’re waiting for coordination with another process. The effect is that service time (S) gets stretched, e.g. trashing of the CPU cache, trashing of SAN cache, etc. With queuing time (Q) you actually are in the queue waiting to be serviced. See Figure 2 how the actual impact of the response time (R) variance is build up.

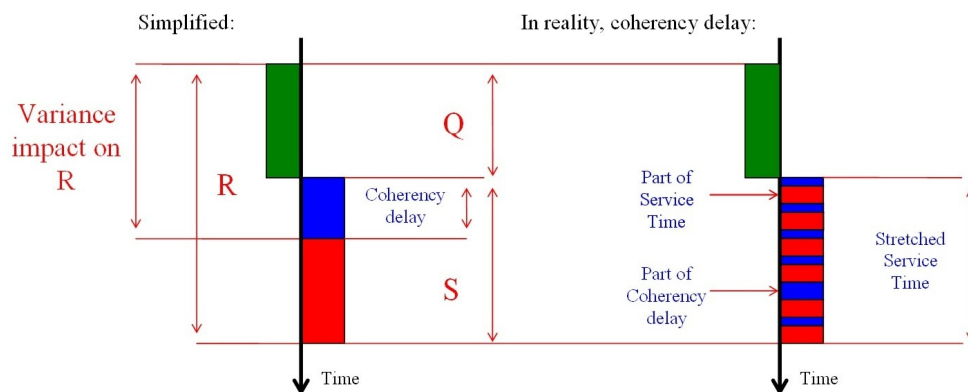


Figure 2: Diagram to make clear how the variance of response time (R) of a specific part of the infrastructure (e.g. CPU resource), is influenced not only by queuing time (Q), but also the stretching of service time due to coherency. In real the coherency is not just one block in the service time, but small repeating delays.

So basically we can say that for each end-user process the difference between fast and slow response time (R) execution can be described as a diagram like shown in figure 3:

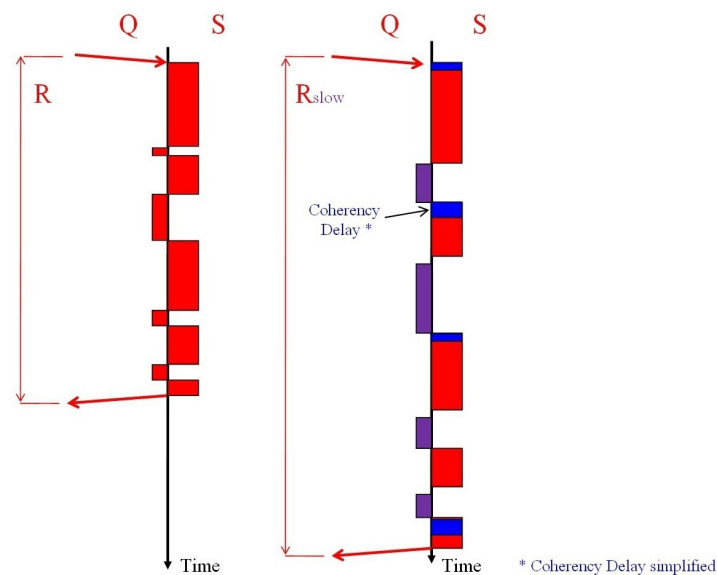


Figure 3: Diagram to make clear how changes of queuing time (Q) and coherency can make total response (R) time longer (R_{slow}).

“Method-GAPP” Data Collection

Based on the knowledge we discussed we can start thinking of what data should be collected for an analyses. If we are collecting data we need data of each tier in the technical architecture of the “chain”, but also data from tiers not part of the “chain” can be used as part of the analyses to checkout if there is influence or not on the response time (R). In Figure 4 is shown how the response time graph (red) can possibly be described by several metrics (blue), it is only not clear which metric or combination of metrics is describing the response time graph the best.

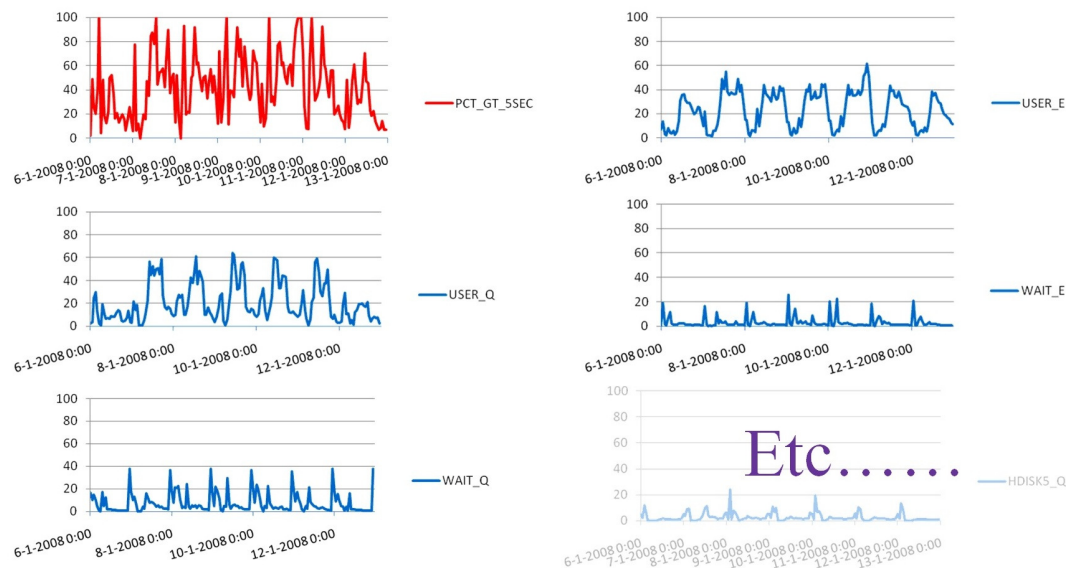


Figure 4: Diagram to show how the response time (red colour) should be possible to be explained by metrics of the technical infrastructure.

Primary components and secondary components data

When start to collect data we have to understand that there are basically two kinds of data within the method, the primary component data and the secondary component data. The primary component data is from the CPU's, I/O (also network) and Memory (the system recourses), the secondary components data is basically the rest of possible data to be collected. Within the primary components data we can find utilizations data, queue length data, memory swap usage data and physical CPU data (in case of virtualization), other data from the system recourses are basically also secondary components. Primary component data is the most important to collect, this makes modeling of the response time possible. Examples of secondary component data is: SQL statement AWR information, number of latch waits, number of java threads, etc.

In principle the method can work with any data source but a very important rule is that all data should be able to be related by timestamps, so time synchronization is from vital importance. To be able to do this, in some cases averaging over different timestamps has to be done (aggregate data). This has as a bad side effect that variance in the data will be reduced. Although this is the case, the relation between the metrics and the response time will not be reduced in such a way that relations between metrics will be changed.

All the collected data should be stored in one database. For this white paper there has been worked with the Oracle database, but in base also another database could be used. The advantage of using the Oracle database is the fact that Data Mining (ODM) is part of the Oracle database. Although this is the case there might be alternatives in the market to do the actual data mining (*e.g. Mahout (Hadoop)*).

Values Examination

Before doing anything it is important to check if the data is valid and in which format the data is collected. Sometimes we can find strange values like negative utilizations or utilizations far above 100%. In these cases it is important that strange values should be deleted. Utilizations over 100% can sometimes be present when the utilization is from a CPU resource within a virtual machine. The metric should in this case be treated as a secondary component. If the values are sometimes just a little over 100%, you can change them to be 100%. Of course utilizations are sometimes in percentages or in ratios (0-1).

Next to the above it is handy to checkout if data is in multiples available. This can be encountered when looking at I/O devices, different devices have exact the same metrics reported. In the analyses just one should be chosen. The Data Mining step to determine the importance of factors (components) will also help to indentify this data.

“Method-GAPP” Data Synchronization

When all the data is inside the database we try to aggregate the data on such a level that we can synchronize them based on the timestamp. It is important to check the time differences between the servers you collected the data from in the first place. Based on this you can decide to aggregate the data on a higher level, so e.g. from 10 minute interval to 30 minute interval. The end result should be one big table (or view) containing one column to identify the case (time, id), one column with the response time (R) to be analyzed (or multiples when used for several application business processes), and for each collected component one column.

Workload profiles

When the data synchronization is done it is good to make sure it is easy to recognize what kind of workload was on the architecture of the direct involved application (chain). For example a system is totally differently used during the night than during the day when the most OLTP is done. This influence will also be discussed in the data modeling steps. One of the most important things of a drastically changing workload profile is the fact that I/O is differently cached. The problem is that e.g. instead of doing 100 I/O's for the end-user process we have to do now 1000 I/O's for the same end-user process. This is for modeling a nightmare, but we can correct our models on this by adding a metric (secondary component) about the SGA size, the number of sort processes, etc. Also by just having longer periods of data will make the model better. A start of a system is due to this fact also hard to be modeled.

“Method-GAPP” Data Modeling

As many of you will realize is the fact that for example CPU utilization and I/O utilization will not have a linear correlation with response time. The correlation curve behind is dependant on the number of server / threads of the resource. In the past Erlang made a formula, the Erlang-C formula, to calculate the probability that all the servers (e.g. CPU's) for the resource are busy ([Gunther 2005](#)). Together with the formula for response time: $R=Q+S$ we are able to calculate the response time (R), for given number of server / threads, see figure 5 with the formula's.

$$C(m, \rho) = \frac{\frac{(m\rho)^m}{m!}}{(1-\rho)\sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!}}$$
$$R = \frac{C(m, \rho)S}{m(1-\rho)} + S$$

Figure 5: The first formula (1) is to calculate the probability (C) that all the servers (e.g. CPU's) for the resource are busy, with as input: number of servers / threads (m) of the resource (e.g. number of CPU's) and utilization of the resource (ρ). The formula was created by Erlang and is called the Erlang-C formula. The second formula (2) is to calculate the resulting response time (R) when the probability (C), the number of servers / threads (m) and service time (S) is filled in.

When we use the formulas to calculate the normalized response time we can find the graph shown in figure 6. The line for 1 server is called the M1 curve, for 2 servers the M2, etc.

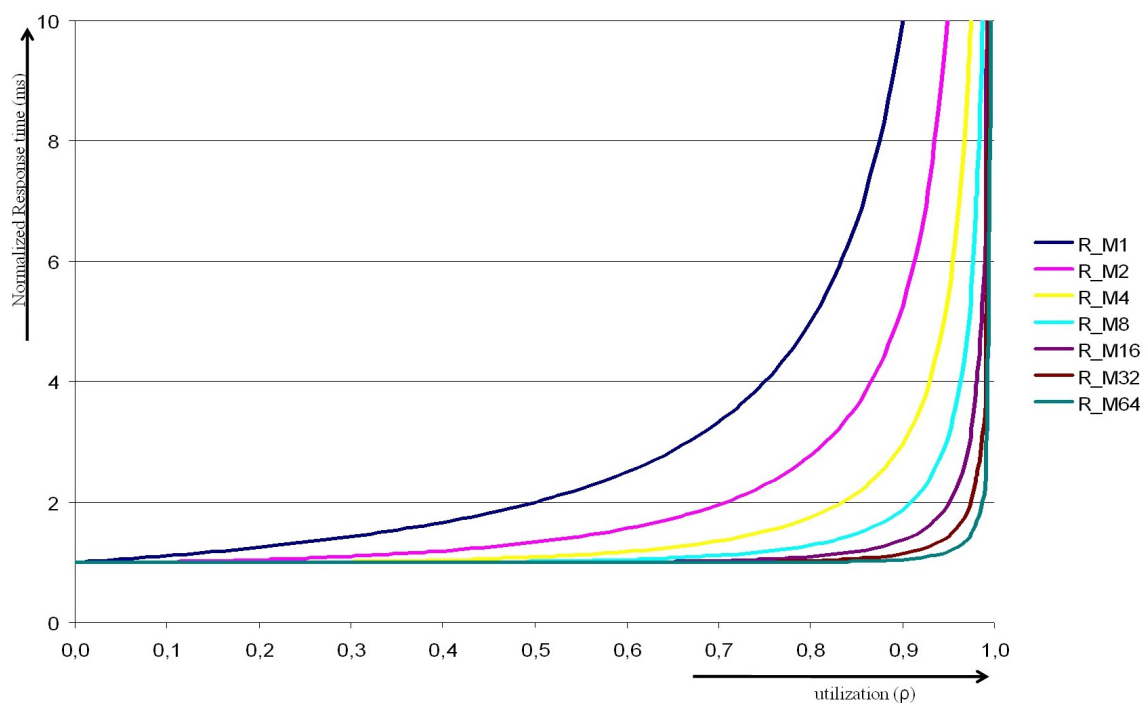


Figure 6: The graph shows how with different number of servers / threads, the normalized (brought back to 1 ms) response time (R_Mx) would be predicted based on Erlangs-C formula, with given utilization (ρ).

Starting a simple test with Swingbench

To illustrate the method some better a simple test was done using a laptop and a virtual machine running Linux. To generate some load and have an end-user process, the load generator “Swingbench” (Order Entry), from Dominic Giles (Oracle) was used in character base. See figure 7 the GUI with the same input values. On Linux sar data was collected.

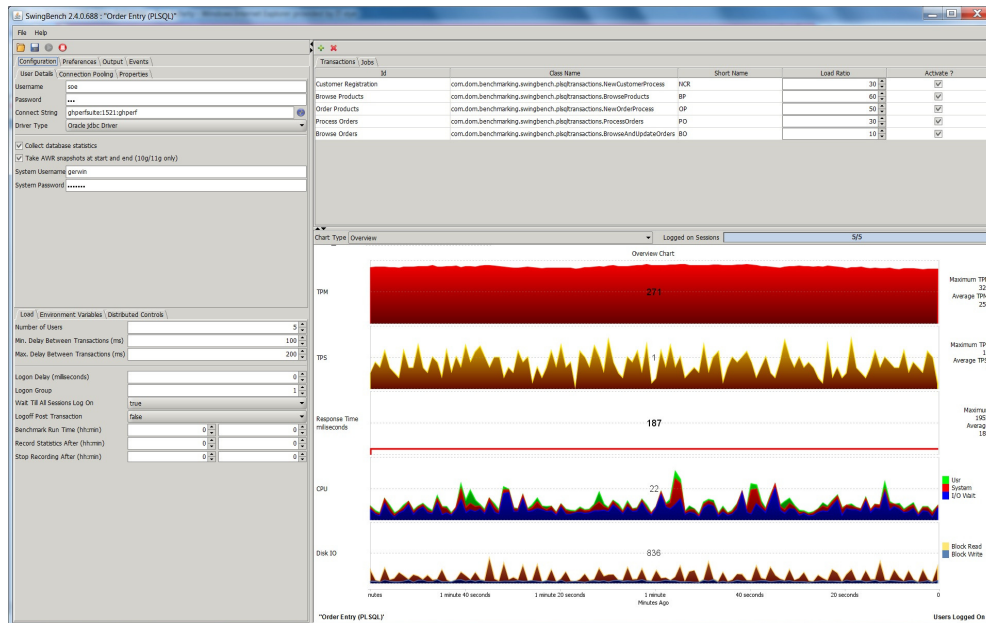


Figure 7: The picture shows the GUI from Swingbench, while running the load test.

The test was done with different amounts of users and in this test was determined where the system looked to be overloaded. This point was around 15 users; the graph of the test is shown in figure 8.

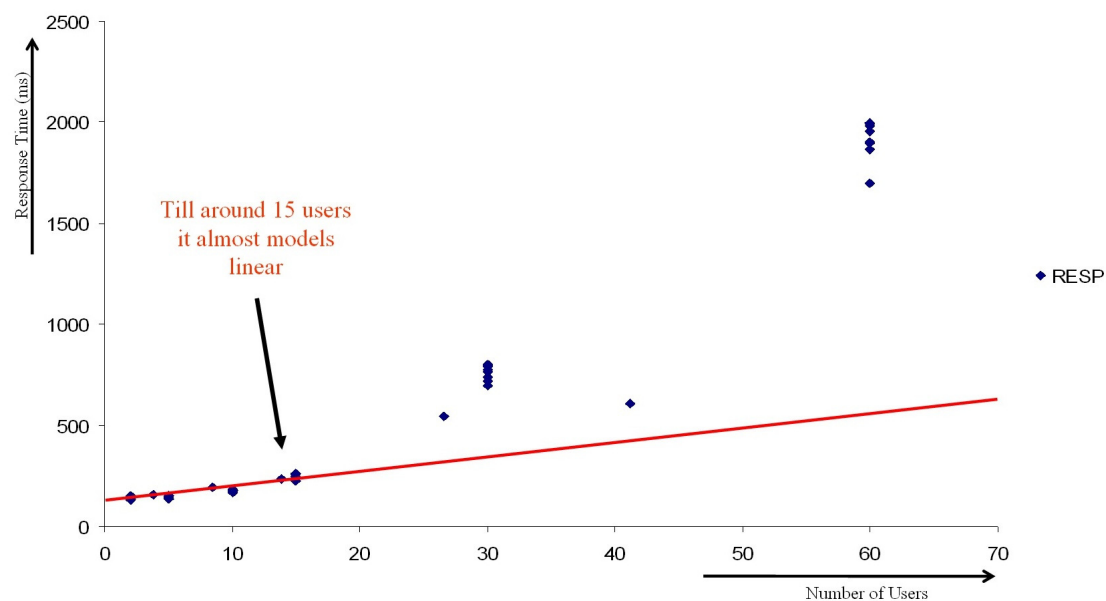


Figure 8: Graph of the measured response time (RESP) versus the number of user added in the test. While adding extra users to the test the response time suddenly started to increase rapidly above 15 added users.

The graph shows clearly how the response time at a certain point is diverting from an almost linear curve. After the above was found a new test was done now knowing that after around 15 users the response time would start to go up fast. The number of users in this new test is shown in figure 9.

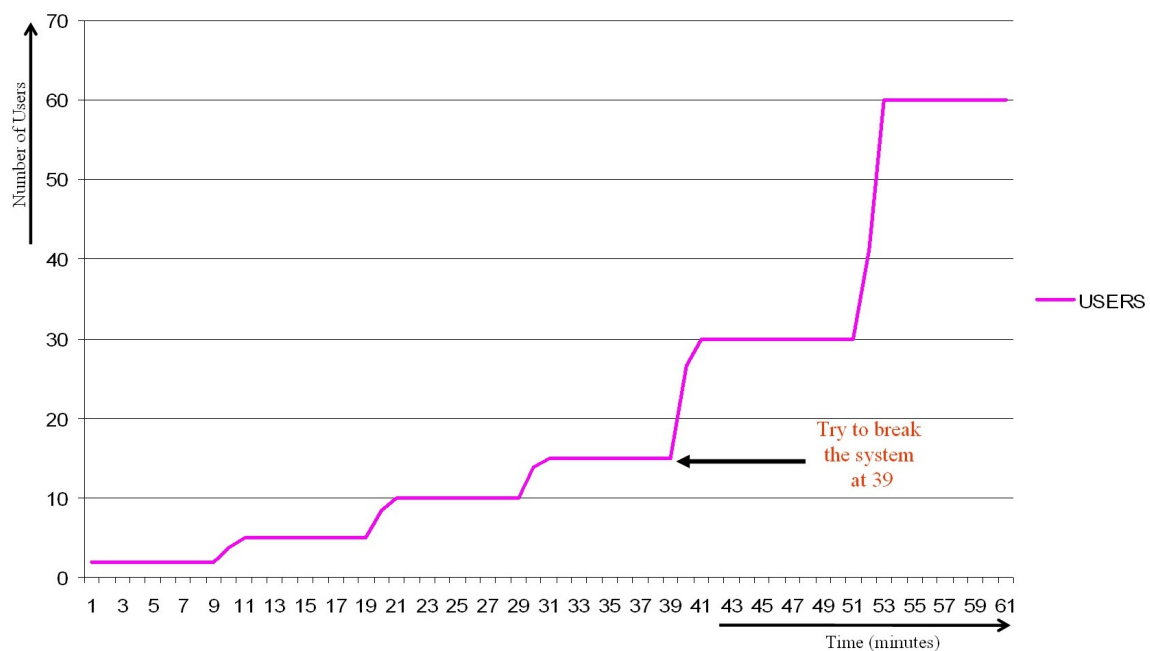


Figure 9: Graph of the measured response time (RESP) versus the time elapsed during the test. From time 39 and onwards we have tried to break the system by overloading it with too many users.

Although I already knew the system would get overloaded the response time through time was measured with increasing amount of users. This behaviour is illustrated in the graph of figure 10.

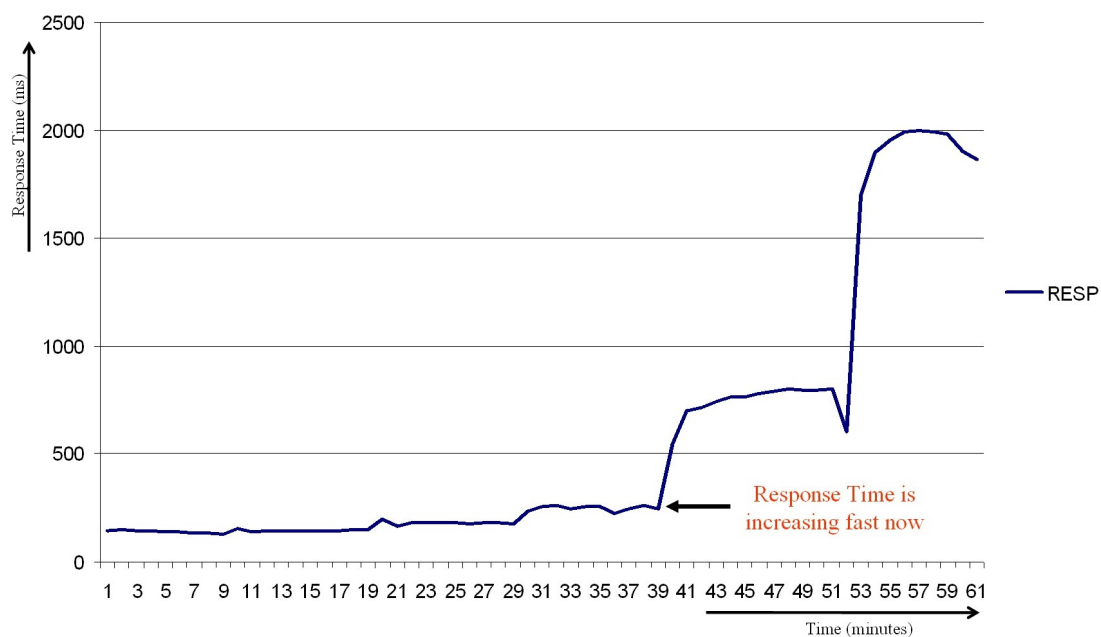


Figure 10: Graph of the measured response time (RESP) versus the time elapsed during the test. From time 39 and onwards we have tried to break the system by overloading it with too many users.

Looking at what is going on at the background we can see for the utilization of the I/O (Device 8-0 in the test), the graph shown in figure 11.

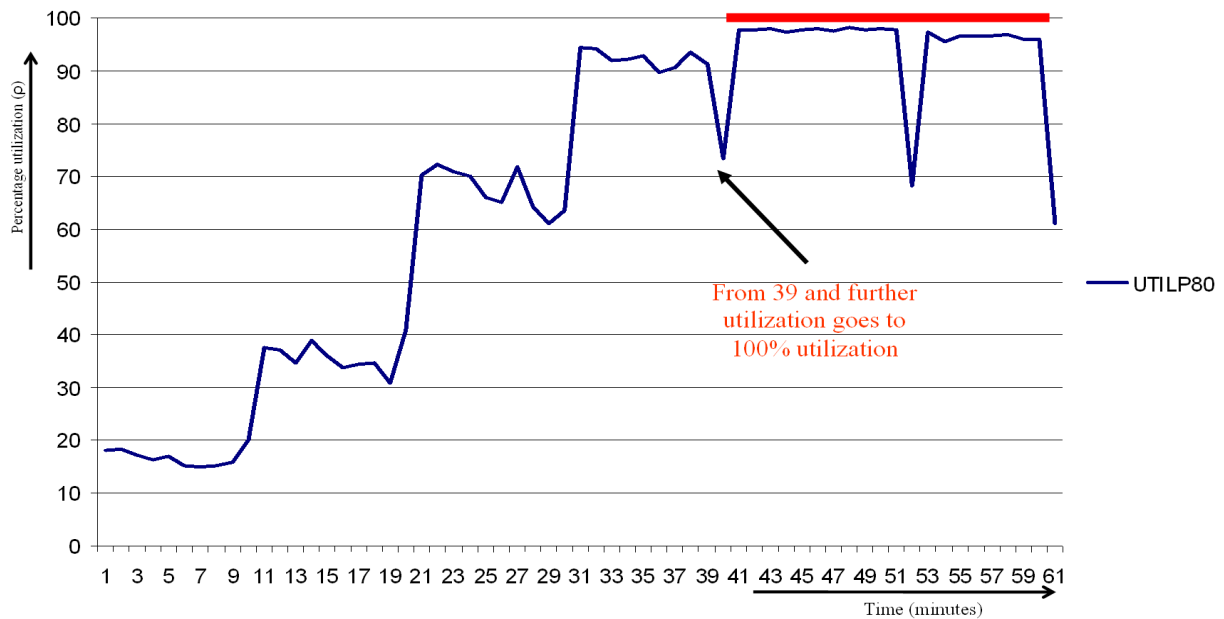


Figure 11: Graph of the I/O utilization of device 8-0 (AVGQUSZ80) versus the time elapsed during the test. The horizontal red line shows how the utilization is starting to hit the 100% utilization.

The graph shows clearly how we start to hit the 100% utilization and due to that probably start to queue on the I/O device. You can see how the run queue of the I/O device is getting higher in the graph of figure 12.

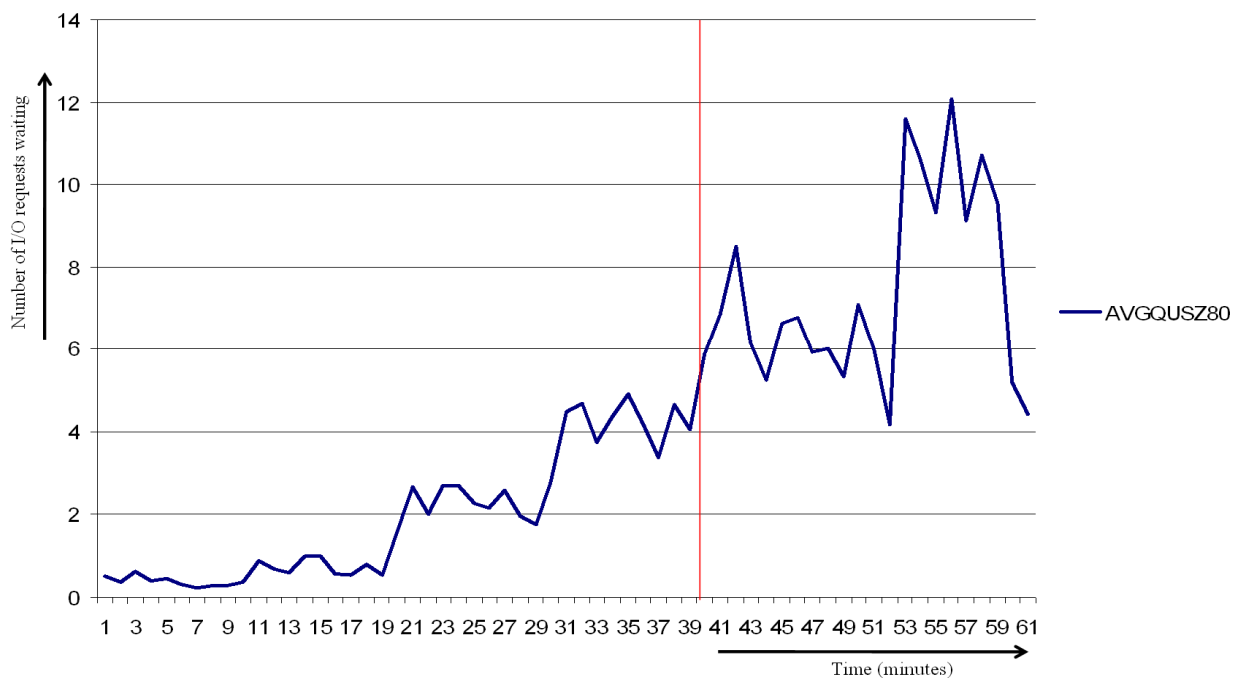


Figure 12: Graph of the average queue length for device 8-0 (AVGQUSZ80) versus the time elapsed during the test. The vertical red line shows the time in the test we tried to break the system by overloading it with too many users.

If we look at the CPU utilization during the test we can find the graph shown in figure 13.

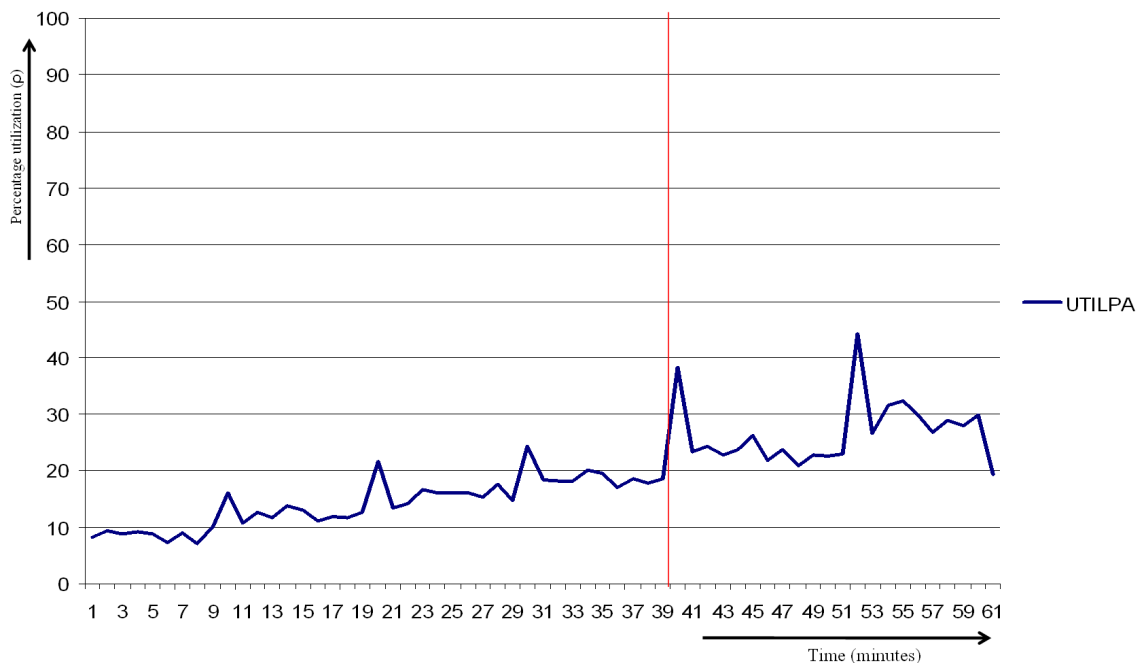


Figure 13: Graph of the CPU utilization (UTILPA) versus the time elapsed during the test. The vertical red line shows the time in the test we tried to break the system by overloading it with too many users.

The graphs show clearly that the system is fully I/O bound and that the I/O is probably mostly responsible for the measured response time. When plotting the response time data against the I/O device utilization with the M1 and M2 modelled curves we can find the graph in figure 14.

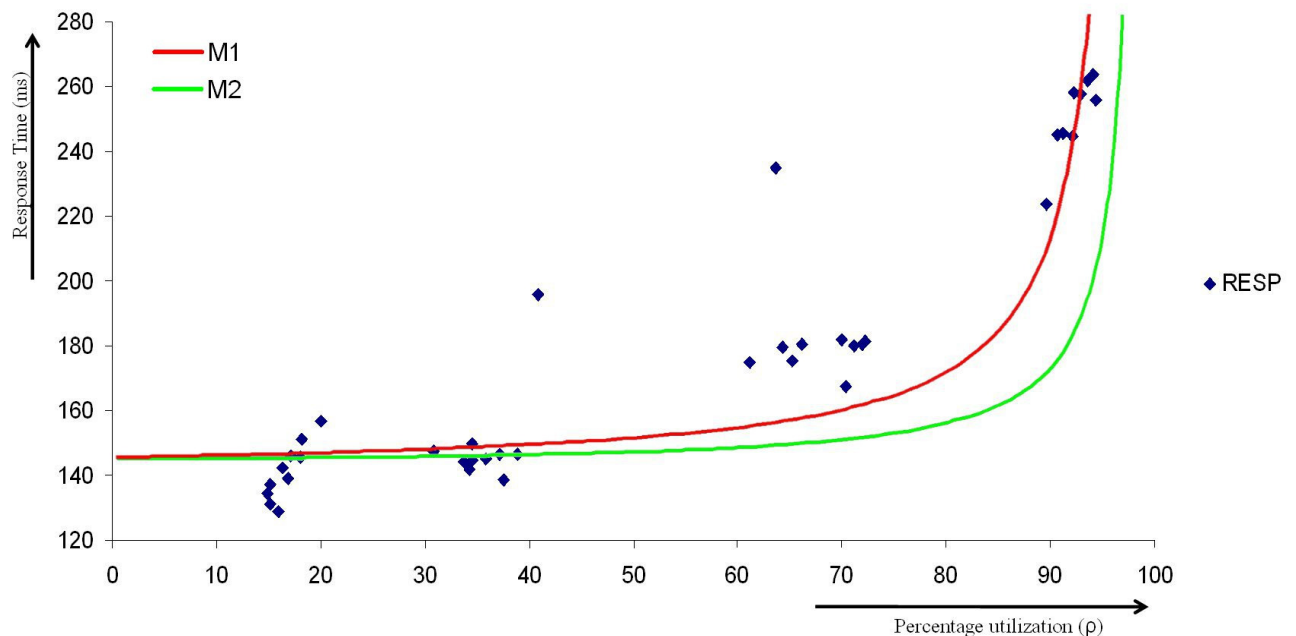


Figure 14: Graph of the measured response time (RESP) versus the utilization of the I/O device (device 8-0 in the test). The M1 and M2 curves show how the data is fitting to the graphs. Important to remark is the fact that response time is also influenced by the CPU impact, coherency and other unknown components (e.g. the Swingbench client) what makes the data not fit complete.

In the graph it is clear that the influence of the I/O is probably described as something like an M1 curve. If we try the same fit with the data of the overloaded situation we can find the curve in figure 15.

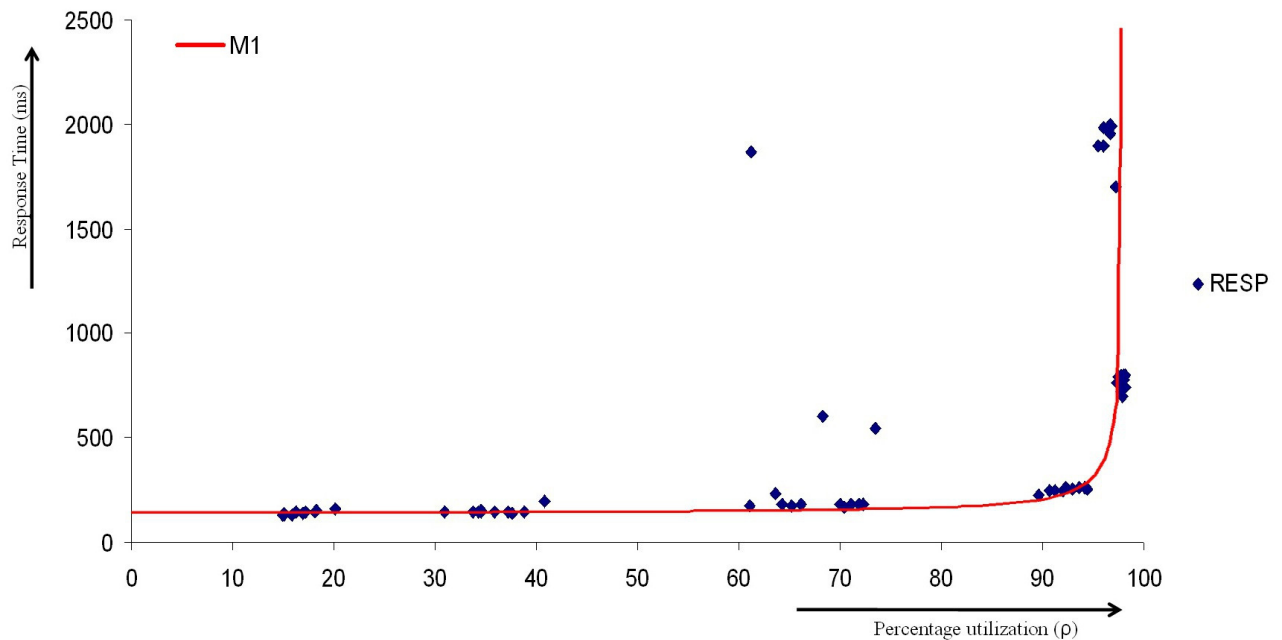


Figure 15: Graph of the measured response time (RESP) versus the utilization of the I/O device (device 8-0 in the test). The M1 curve drawn in the graph shows how the data is fitting the M1 curve.

The curve shows clearly that the influence of the I/O device is far from linear and that the M1 curve is highly likely to be correct.

“Method-GAPP” Data Mining - Explain

In real big systems it is not possible to checkout every component and how it is influencing the response time of the end-user process. For this we can use data mining, to make the computer do the monks job in principle. One of the Data Mining’s functions around is the attribute (predictors) importance explaining, which is in base a factorial analysis. Within Oracle Data Mining there is a procedure “EXPLAIN” within the package “DBMS_PREDICTIVE_ANALYTICS” (*Oracle 2010*) to do this. For this can also be found alternatives, but in this white paper only the Oracle Data Mining’s procedures were used.

So going back to our small test we used the “EXPLAIN” procedure to determine which components were actually the most important in predicting the measured end-user response time. The graph of this factorial analysis is shown in figure 16.

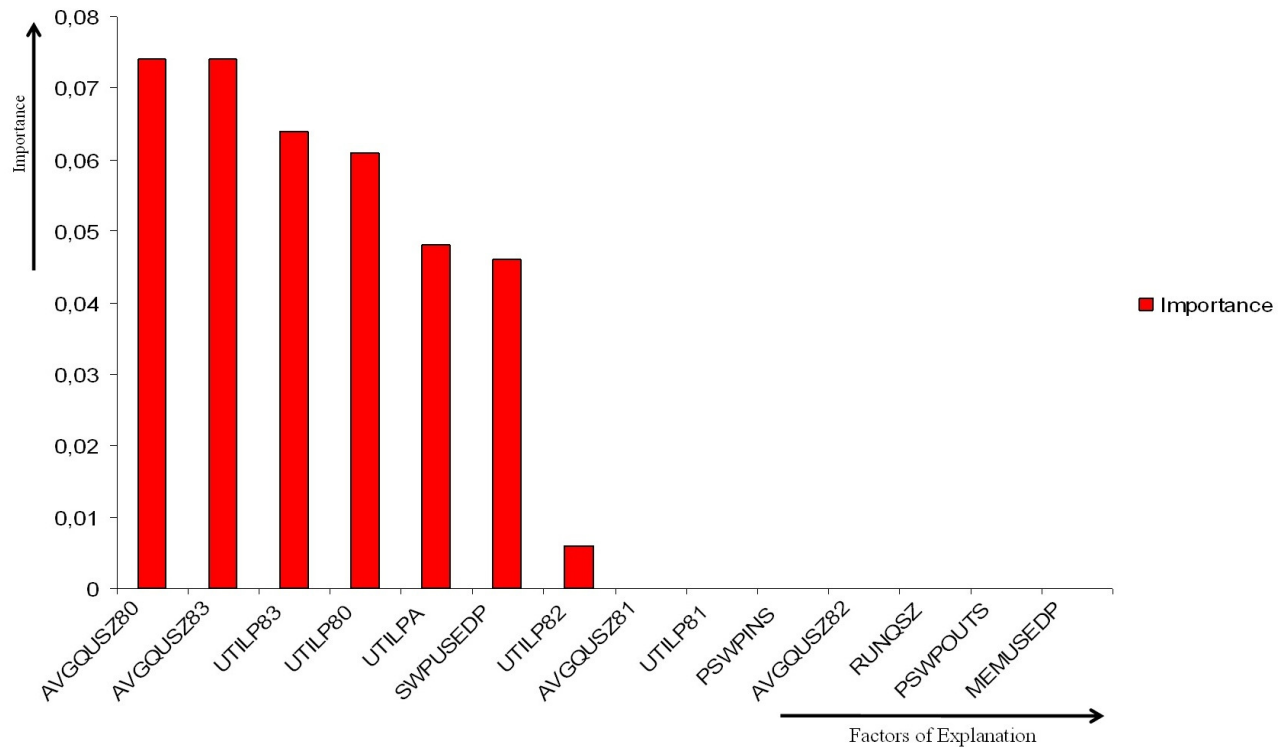


Figure 16: Graph shows the factorial analyses (GAPP primary components) of the measured (full 61 minutes) response time in the test. In the graph in order: average queue length of device 8-0 (AVGQUSZ80), average queue length of device 8-3 (AVGQUSZ83), utilization of device 8-3 (UTILP83), utilization of device 8-0 (UTILP80), utilization CPU's (UTILPA), swap used percentage (SWPUSEDP), utilization of device 8-2 (UTILP82), average queue length of device 8-1 (AVGQUSZ81), utilization of device 8-1 (UTILP81), page file swapins (PSWPINS), average queue length of device 8-2 (AVGQUSZ82), run queue of CPU (RUNQSZ), page file swapouts (PSWPOUTS) and memory used percentage (MEMUSEDP).

The graph shows clearly that the run queue of device 8-0 and device 8-3 are the most important predictors, after that the utilization of device 8-3 and device 8-0. A small investigation shows that Device 8-0 and 8-3 are in real the same device, so only device 8-0 is further used as the I/O device.

Basically the factorial analyses can determine which component is the most responsible for the variance in the business (end-user) process response time. In this way we can find out in very complex systems where the response time is the most influenced. To make in bigger systems with a lot of components the diagrams a little easier to interpret we can give the component with the highest importance the index number 100, and all other component on ratio a lower index number.

If we checkout now the diagrams from the beginning of the whitepaper what the factorial analyses actually is doing we can look at the diagram in figure 17.

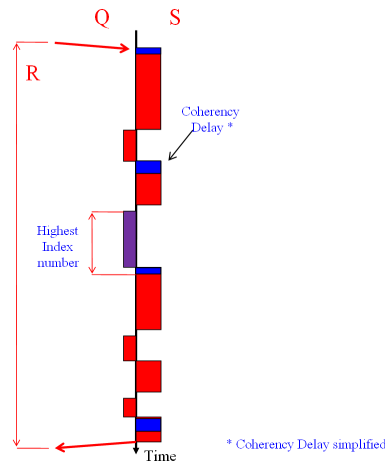


Figure 17: Diagram how the factorial analyses graphs are mapped to the sequence diagram. The biggest variance of queuing time (Q) and coherency together for a specific part of the architecture will have the highest bar in an importance graph (index number 100% or the highest importance).

The diagram makes clear that basically the factorial analyses find the performance bottlenecks in the system. If also components are involved outside the “chain” we can find out what component, maybe not even part of the “chain” have influence on our investigated end-user process. Now we know the bottlenecks we can start tuning these parts.

Another nice thing about doing the factorial analyses with secondary components is the fact that we also can find out which SQL is involved in our investigated end-user process. This is illustrated in the graph in figure 18. The big advantage is that we can find out easily which SQL is involved and which of this SQL is relatively easy to be tuned, to have our end-user response time better.

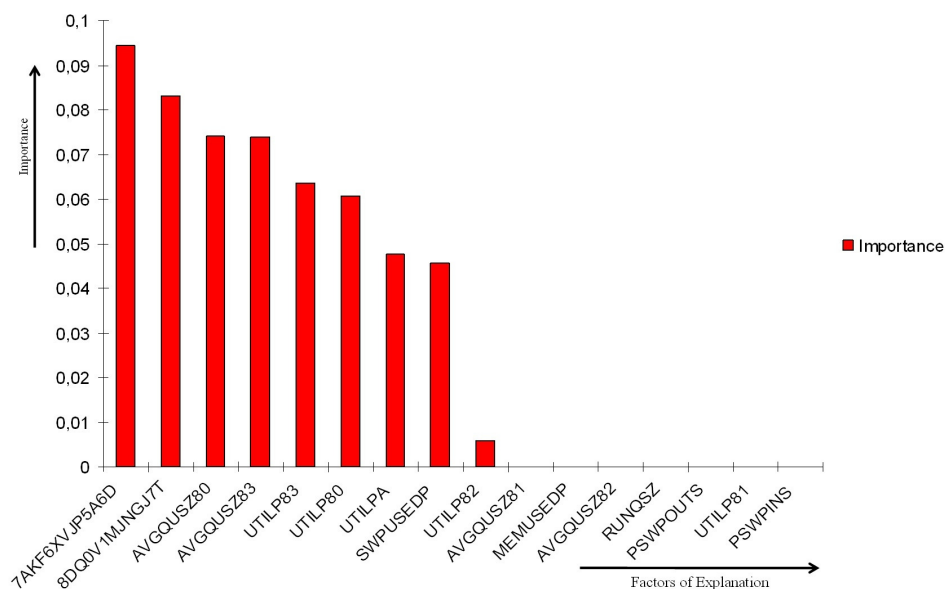


Figure 18: Graph shows the factorial analyses (GAPP primary and secondary components) of the measured (full 61 minutes) response time in the test. In the graph in order: response time of sql statement “7akf6xvjp5a6d” (7AKF6XVJP5A6D), response time of sql statement “8dq0v1mjngj7t” (8DQ0V1MJNGJ7T), average queue length of device 8-0 (AVGQUSZ80), average queue length of device 8-3 (AVGQUSZ83), utilization of device 8-3 (UTILP83), utilization of device 8-0 (UTILP80), utilization CPU’s (UTILPA), swap used percentage (SWPUSEDP), utilization of device 8-2 (UTILP82), average queue length of device 8-1 (AVGQUSZ81), memory used percentage (MEMUSEDP), average queue length of device 8-2 (AVGQUSZ82), run queue of CPU (RUNQSZ), page file swapouts (PSWPOUTS), utilization of device 8-1 (UTILP81) and page file swapins (PSWPINS).

The “seconds per executions”, response time from the SQL statements was retrieved from AWR data. The graph shows clearly that response time from SQL with SQL id “7akf6xvjp5a6d” is highly involved in the measured end-user process.

If we only do a factorial analysis on the data till the point the system gets overloaded we can find the graph in figure 19.

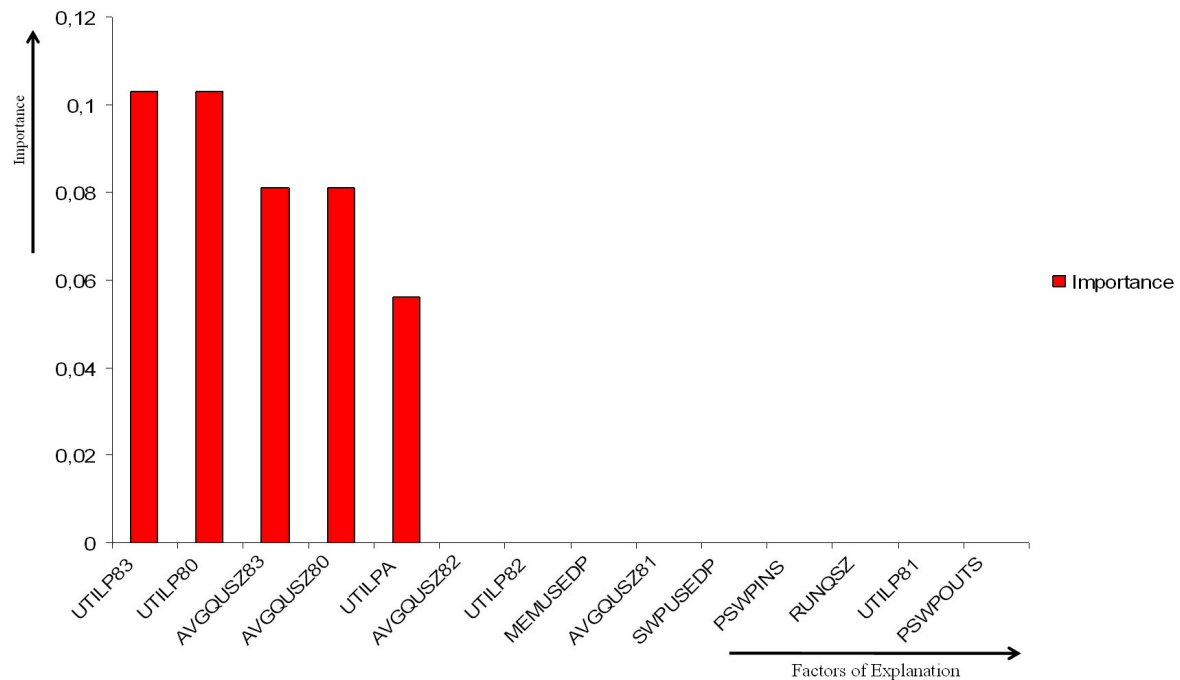


Figure 19: Graph shows the factorial analyses (GAPP primary components) of the measured (first 39 minutes) response time in the test. In the graph in order: utilization of device 8-3 (UTILP83), utilization of device 8-0 (UTILP80), average queue length of device 8-3 (AVGQUSZ83), average queue length of device 8-0 (AVGQUSZ80), utilization CPU's (UTILPA), average queue length of device 8-2 (AVGQUSZ82), utilization of device 8-2 (UTILP82), memory used percentage (MEMUSEDP), average queue length of device 8-1 (AVGQUSZ81), swap used percentage (SWPUSEDP), page file swapins (PSWPINS), run queue of CPU (RUNQSZ), utilization of device 8-1 (UTILP81) and page file swapouts (PSWPOUTS).

In the graph it is clear that now the utilizations are more important as the I/O queues, and that the CPU utilization has also some influence on the end-user response time.

When choosing the factors for explanation (components) it is important to choose factors which are as less as possible correlated. This means normal to have just one primary component per resource. In this case the utilization of device 8-0 and the CPU utilization were chosen.

One of the problems with data mining in general is the fact that predictors should be independent. A lot of data in the world has some correlation between each other and for this problem Ridge Linear regression can help. By playing with the Ridge variable sometimes better linear models can be found.

“Method-GAPP” Data Mining – Model

After knowing what the most important components are we can actually start making a model of the data to predict the measured response time. Within Oracle Data Mining we can use function “Regression” with algorithm “Linear Regression (GLM)”, these functions are also available in other data mining’s tools. We use this function on a view with only the selected components. With a small trick we can in this way also determine which M-curve is fitting the best with the data. Inside Oracle we can use the Oracle Data Miner, the latest Oracle SQL Developer or from SQL the package “DBMS_DATA_MINING” (Oracle 2010).

To determine the best M-curve fitting we can use an iterative process. We start with a view containing all the selected columns as the raw data retrieved from the system, and we calculate a model with the involved columns, after the calculation we store the “root squared error” from the model, and do the calculation again with a normalized response time of the most important factor with the M1 curve, with M2 curve, etc. After some calculation we than can find what the smallest “root squared error” is and so determine which M-curve was fitting the best. We repeat the procedure with the second most important component with the already determined most important component M-curve in place. After all components have been done the procedure is repeated again, starting with the most important component. In figure 20 is illustrated what the M-curve fitting tries to accomplish.

$$R_{tot} = c_1 R_{1,n=8} + c_2 R_{2,n=4} + c_3 R_{3,n=1} + etc.$$

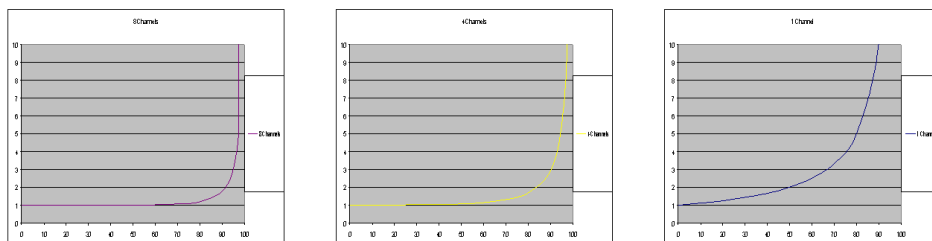


Figure 20: The picture shows an example of what the M-curve fitting tries to accomplish. In the picture the “c” is the coefficient of the calculated model and the R_1 , R_2 , etc. is the normalized response time of the “n” amount of servers, for that specific component.

After we know the best M-curve fit for each component we can calculate the model with the components transformed with the correct M-curve. See the picture in figure 21 for a calculated model, after the M-curves were been determined for the I/O and CPU.

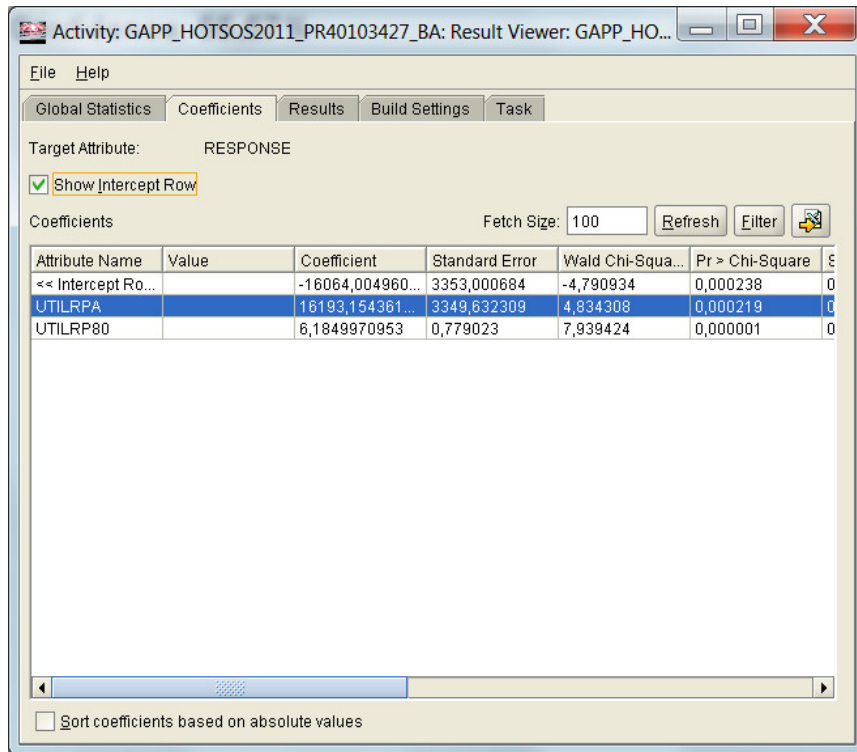


Figure 21: Picture shows the result of the data mining done in Oracle Data Miner on the selected primary components.

The discussed procedure delivers us via the “coefficients” (see figure 21) of the model for the predictors and the “intercept row” of the model, the actual linear formula’s as shown in figure 22.

$$R_{RESP} \approx 16193.15 \times M_{CPU\ n=4} + 6.19 \times M_{IO\ n=1} + -16064.00$$

$$R_{RESP} \approx ((16193.15 \times M_{CPU\ n=4}) - 16193.15) + 6.19 \times M_{IO\ n=1} + (-16064.00 + 16193.15)$$

$$R_{RESP} \approx ((16193.15 \times M_{CPU\ n=4}) - 16193.15) + 6.19 \times M_{IO\ n=1} + 129.15$$

$$R_{RESP} \approx R_{UTILRAU} + R_{UTILR80} + R_{REST}$$

Figure 22: Formula extracted of the linear regression model used for the data mining. In this formula the modelled CPU utilization (Mcpu with 4 servers, normalized with Erlang-C) and the modelled I/O utilization (Mio with 1 server, normalized with Erlang-C) was used to calculate the predicted response time (Rresp). At the bottom we can say that the response time from the CPU component (Rutilrau), the response time of the IO component (Rutilr80) and the always present rest response time (Rrest) is giving us the predicted overall response time (Rresp).

The formula makes clear that on base only 6,19 ms I/O service time (S_{io}) is used for the request / transaction what is running at minimum in 135,34 ms (129,15+6,19), in a way this is the service time of the request / transaction (S_{total}). Important to realize is the fact that big change in I/O caching can make the modelling unreliable (see Workload profiles). Another remark regarding the workload profiles is the fact that on occasion when long running processes are running on the system (e.g. night batch), a resource might not be modelled e.g. via a M4 curve, but e.g. with a M1 curve. This can give big issues with the found formula. So it might be an option to split the data in different workload profiles (e.g. day and night).

Now we know the formula behind the model we can checkout the used normalized model for the I/O channel (device 8-0 in the test). This graph is shown in figure 23.

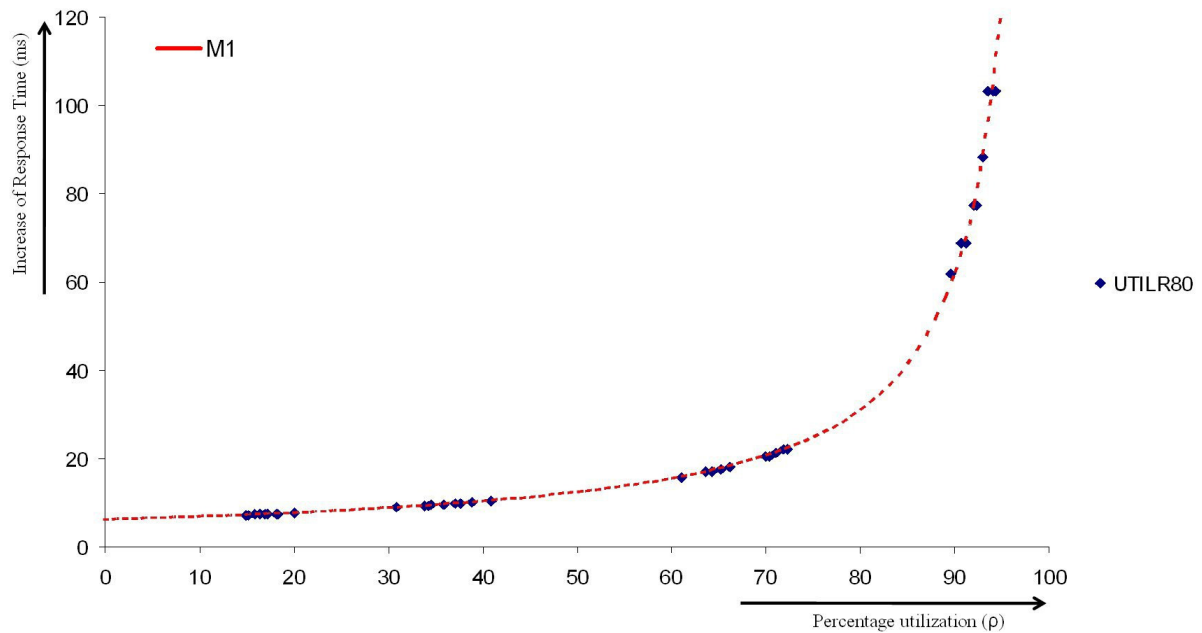


Figure 23: Graph of the predicted curve for the I/O impact (UTILR80) versus the utilization of the I/O device 8-0 in the test. The predicted curve is based on a model based on CPU and I/O.

The involved normalized response time for the CPU would be look like the graph in figure 24.

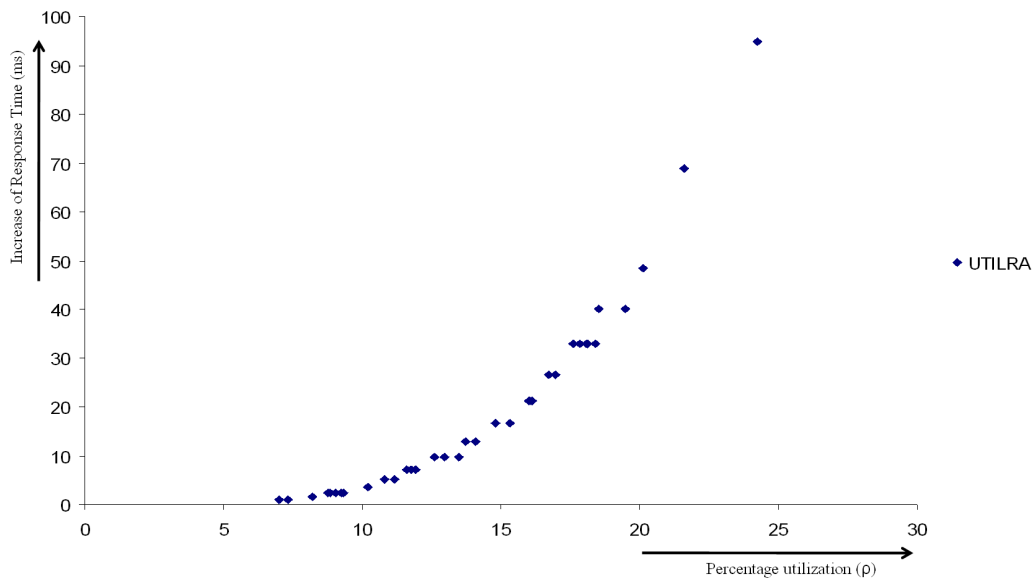


Figure 24: Graph of the predicted curve for the CPU impact (UTILRA) versus the utilization of the CPU. The predicted curve is based on a model based on CPU and I/O.

The graph and the formula show that the CPU utilization has modelled a little odd, but this can be due to the following facts: The server is on Vmware (virtualized), the available data is for the CPU utilization for just a too small range, too little data in the model, some unknown client component which was not put in the model, etc.

In the end we should check how the found formula is actual describing the reality. A bar graph representation can be found in figure 25.

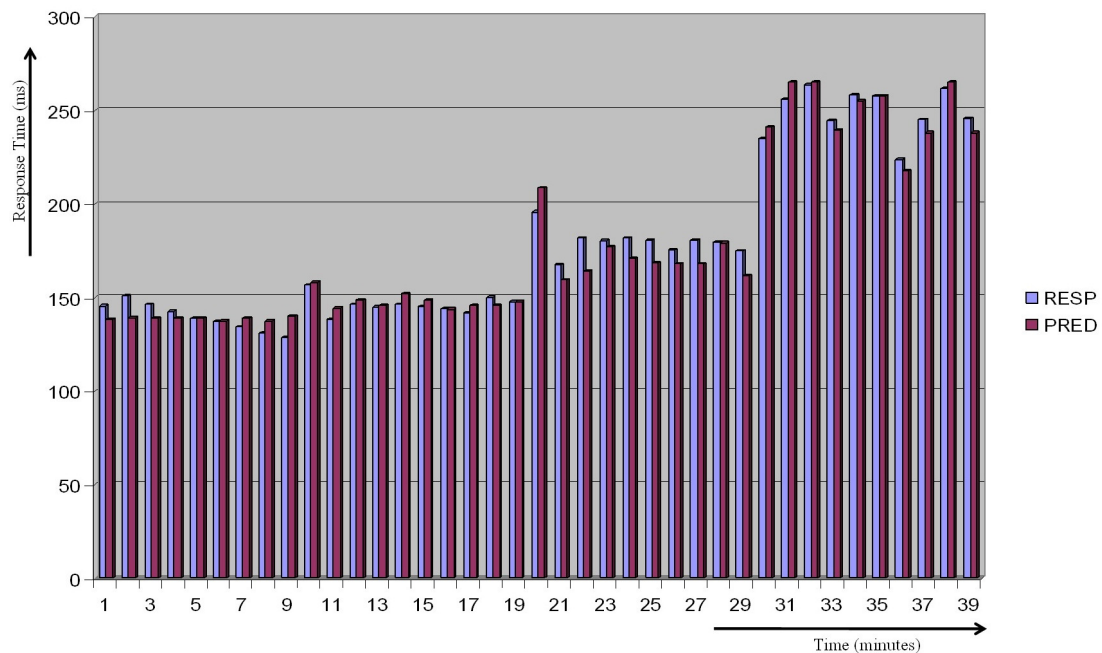


Figure 25: Graph of the measured response time (RESP) and the predicted response time (PRED) by the model based on CPU and I/O versus the time elapsed during the test.

We are now also able to create a graph with of each component the influence on the total predicted response time. This is shown in the graph in figure 26.

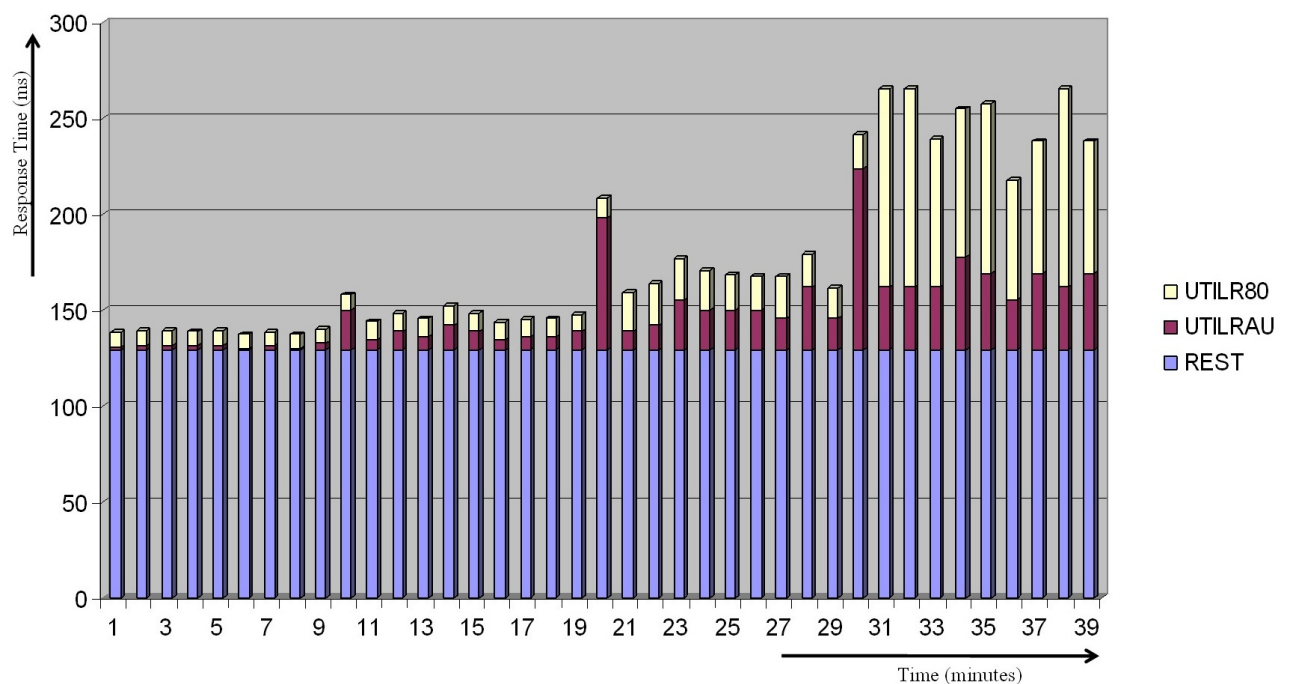


Figure 26: Graph of the predicted response time by the model based on CPU and I/O versus the time elapsed during the test. In the bars the CPU component (UTILRAU), the I/O component of device 8-0 in the test (UTILR80) and the base response time (REST) is shown.

The graph shows clearly what the influence is from each individual component within the response time. Remarkable is the bar at time 30 minutes, the influence of the CPU is here very high but the impact of the I/O is little.

To have a good understanding of the fact what component selection can have as impact, a model was created without the CPU, so only based on the I/O. The formula for this model is shown in figure 27.

$$R_{RESP} \approx 7.52 \times M_{IO_{n=1}} + 145.41$$

$$R_{RESP} \approx R_{UTILR80} + R_{REST}$$

Figure 27: Formula extracted of the linear regression model used for the data mining. In this formula the modelled I/O utilization (Mio with 1 server, normalized with Erlang-C) was used to calculate the predicted response time (Rresp). At the bottom we can say that the response time of the IO component (Rutilr80) and the always present rest response time (Rrest) is giving us the predicted overall response time (Rresp).

After having this formula we can again create a factorial analyses bar chart with the influences of the used components. This graph is shown in figure 28.

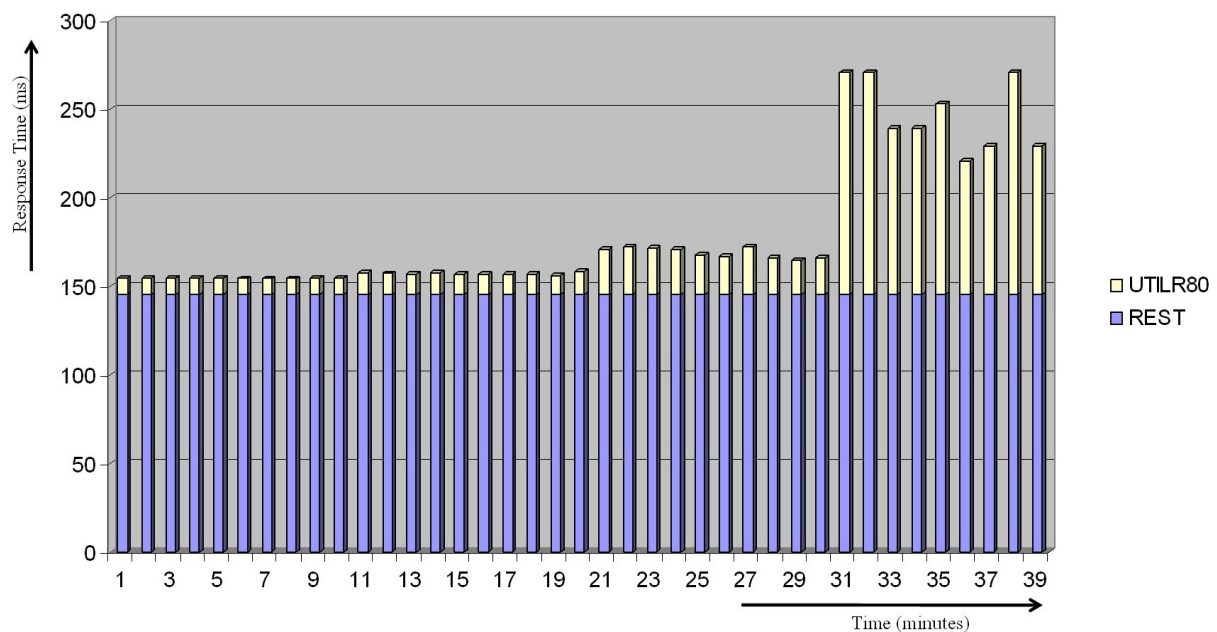


Figure 28: Graph of the predicted response time by the model only based on I/O versus the time elapsed during the test. In the bars the I/O component of device 8-0 in the test (UTILR80) and the base response time (REST) is shown.

The figure shows again a nice bar chart, explaining where the response time was used. But if we look now at the data at minute 30 we can't see the actual high response time (see figure 26). Now that we have the two created models we can compare them visual versus the actual measured response times. This is shown in the graph in figure 29.

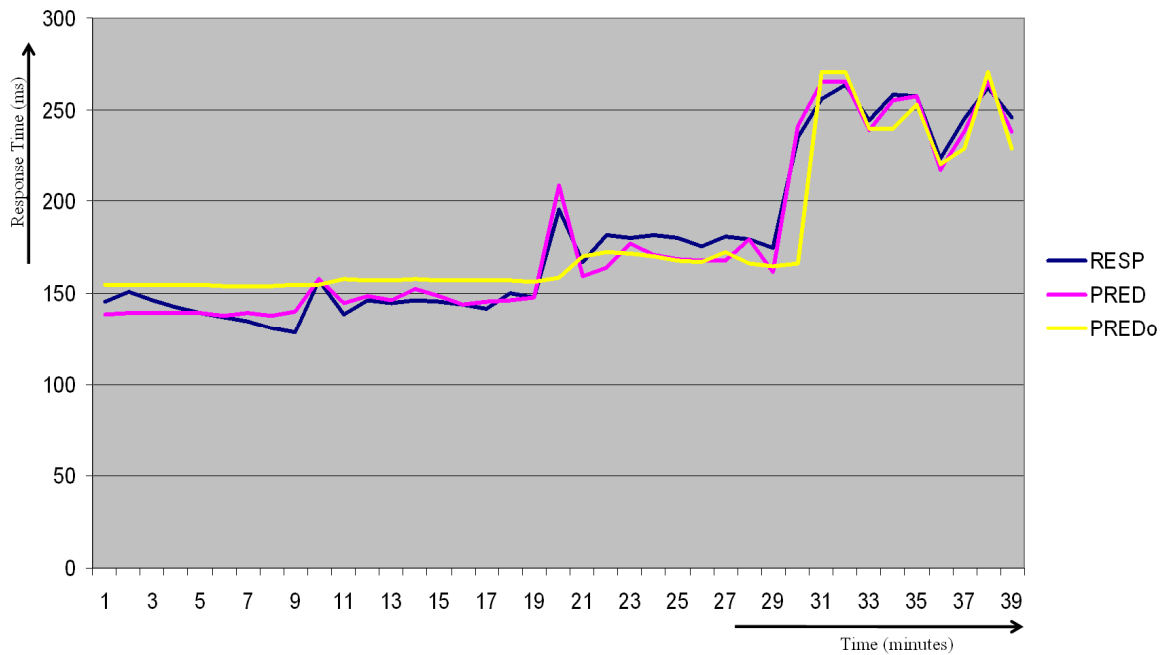


Figure 29: Graph of the measured response time (RESP), the prediction of the model with CPU and I/O (PRED) and the prediction of the model only based on I/O (PRED_o) versus the time elapsed during the test.

The graph shows clearly that the created model including the CPU has a better predicting value than the one created without the CPU.

“Method-GAPP” Data Mining – Predict

Now we have been able to make models of measured response times, we will go a step further by using the data mining models to predict what will happen if things are changing in the infrastructure. Figure 30 shows the prediction of the response time in case the number of I/O channels was changed from 1 to 2 (if it is possible) and the utilization of the I/O stays the same with the same amount of transactions. In real we will see that a positive I/O configuration change, will allow that the same amount of users will do more transactions and due to this fact increase also the CPU utilization, so the graph is a good indication of what will happen in real.

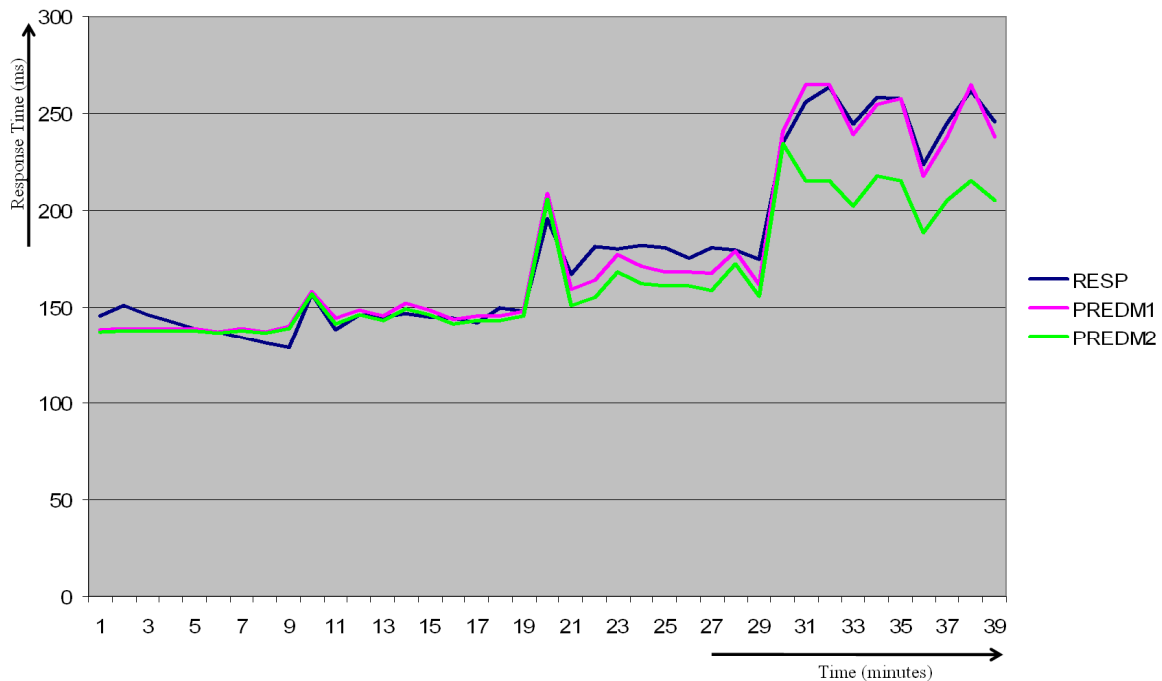


Figure 30: Graph of the measured response time (RESP), the prediction of the model with an M1 modelling of the I/O (PREDM1) and the prediction of the model with an M2 modelling of the I/O (PREDM2) versus the time elapsed during the test.

The graph shows well how the response time would be much lower in the area's the utilization gets higher. Of course normally double the amount of channels will also reduce the utilization of the device. So the effect will be bigger than shown in the graph. Based on the graph we can decide if the change would be worth the investment.

In the coming paragraphs we will look deeper into real customer cases where "Method-GAPP" was used very successful. Because of the fact that the method was not as far as it is now, the analyses was done using a function inside Oracle Data Mining package "DBMS_PREDICTIVE_ANALYTICS", called "PREDICT" (Oracle 2010). Still this function might be very handy if the data is very complex and hard to be modelled with "linear regression". So it is still a very valuable alternative of the "linear regression" model. The big advantage of having a linear model over a model created with the "PREDICT" procedure is the fact that basically we can see what the service time (S) and the queuing time (Q) part is within a components impact on the response time (R).

"Method-GAPP" Data Interpretation

When starting to predict response times and influences of technical changes, the "Method-GAPP" analyses are very helpful but still it is important to use your performance and subject matter expertise to make a correct interpretation of the analyses. The interpretation of the analyses is a vital step in the whole method.

For example as the secondary component “network packets” is selected in a factorial analyses it might be very well possible that it is of high importance, but in real it doesn’t mean that we have a problem with the network. It only says that when we are using the system more we sent more network packets through the system. This kind of interpretation problems can occur on all kind of levels. To ensure the interpretation is correct we should base most conclusions on the primary component factorial analyses. When using the secondary components we should always be more careful.

As a rule of thumb when doing a factorial analyses of the primary components we can say a higher importance of the queue length than of the utilization is a strong indication that the involved component is a performance bottleneck. This was also the indication when we overloaded our system in the small test with “Swingbench” we did. Still we need to be careful because for some systems 100% utilization is not really a problem (e.g. Netapp I/O), while on other systems 100% utilization means that the resources are starting to be exhausted.

Customer Case: Banking Application

In figure 31 a schematic representation is shown of the customer case. The problem in this environment was the fact that there was no test system and tracing was not allowed. Basically we were trying to find a performance bottleneck with both hands tied on our back.

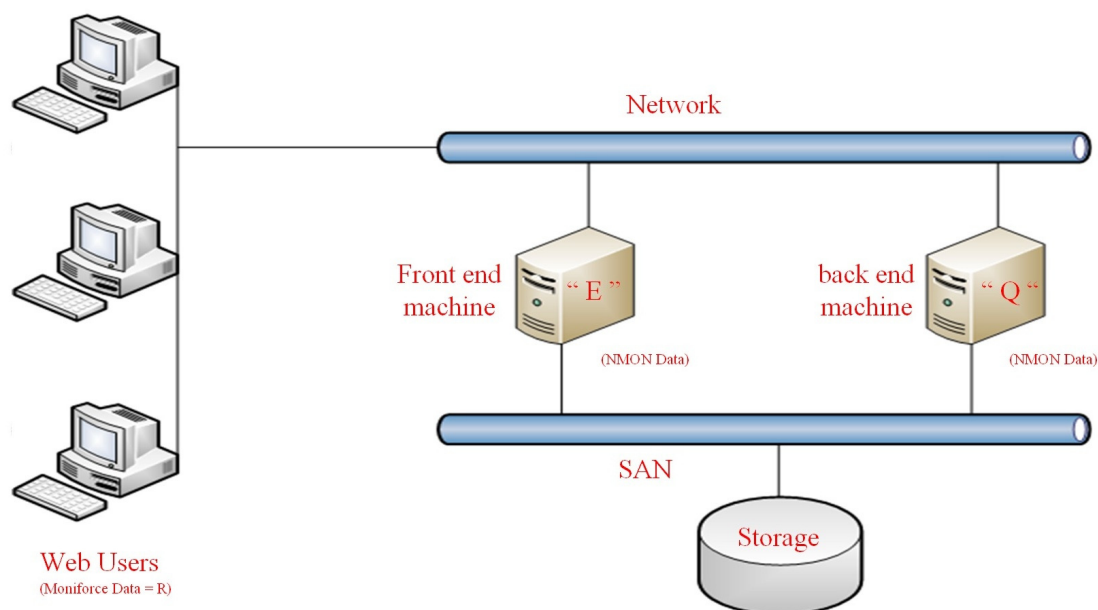


Figure 31: Figure shows how schematic the customer technical architecture looks like. The application is a banking application, which allows users at home to check their bank account, etc. The architecture has two databases, the database on machine “E”, which is the database always giving an answer, and a database on machine “Q” which refreshes “E”. The business case here is that refreshing takes more than 5 seconds what will mean that the end user gets old bank account data.

For this environment a factorial analyses was done and the result is shown in figure 32.

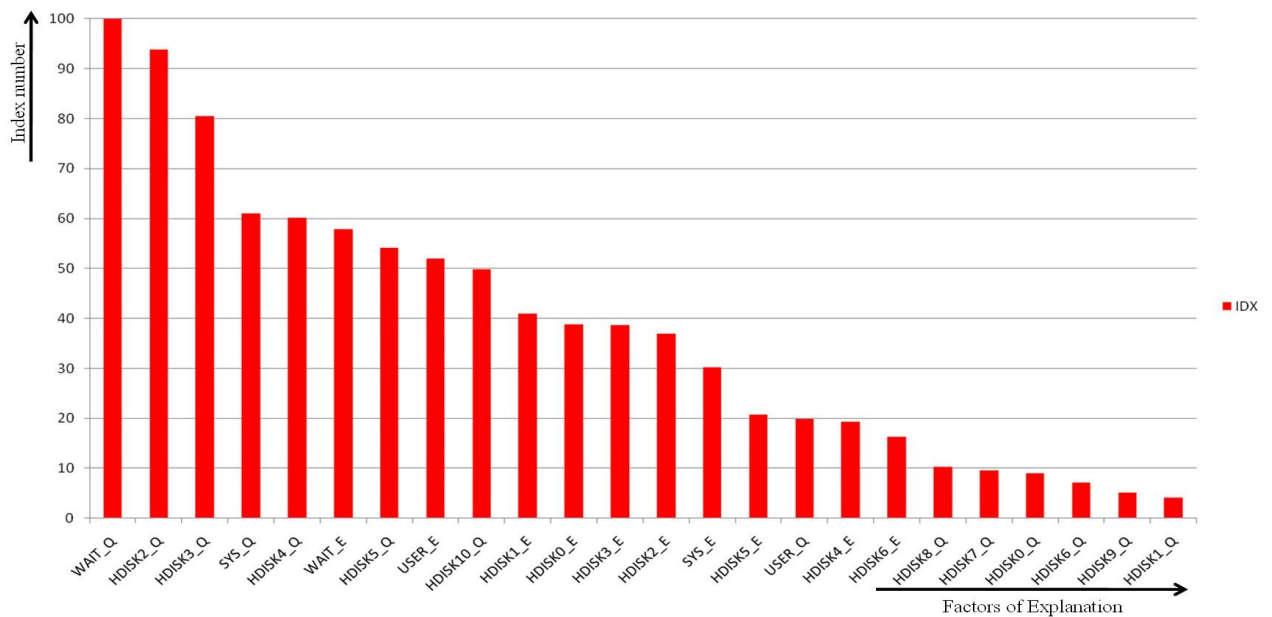


Figure 32: Graph shows the factorial analyses of the measured percentage failures of response times higher than 5 seconds. The model was based on months August, September and October 2007. In the graph in order, the first five: CPU I/O wait percentage of machine Q (WAIT_Q), utilization of device 2 at machine Q (HDISK2_Q), utilization of device 3 at machine Q (HDISK3_Q), CPU system mode percentage of machine Q (SYS_Q) and utilization of device 4 at machine Q (HDISK4_Q).

In the graph it is clear that the first three components have the most influence on the response time. The three metrics can be used to create a model to describe the response time. In this case the earlier mentioned Oracle Data Mining “PREDICT” procedure was used to create the model. The predicted response time graph of the created model is shown in figure 33.

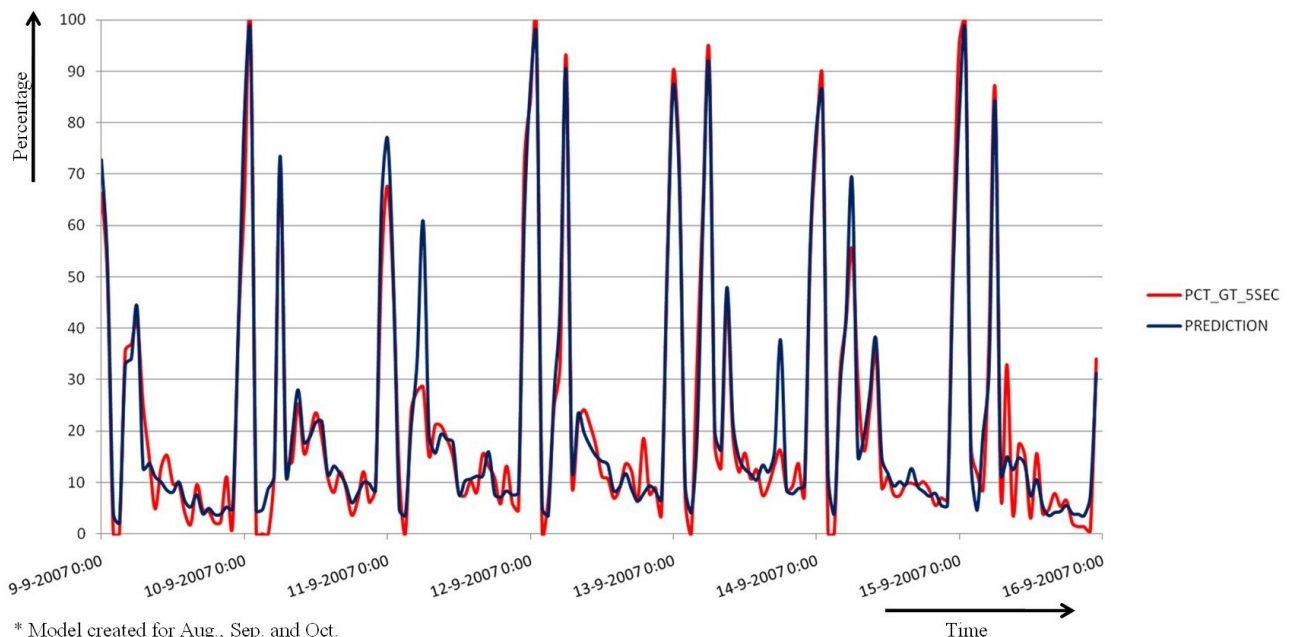


Figure 33: Graph of the measured percentage failures of response times higher than 5 seconds versus the predicted percentages over one week in September 2007. The model was based not only on the week presented, but on months August, September and October 2007. Modelling was done with the dbms_predictive_analytics.predict routine.

In the picture it is clear that the prediction is very accurate describing the real measured response time. By having the model it becomes possible to make a copy of known cases and modify its input metrics values as being new unknown cases to predict the new response time graph. In figure 34, this prediction is plotted against the old predicted response time. In this case the I/O metrics of the copied cases were reduced with 25% and the response time for these new cases was predicted.

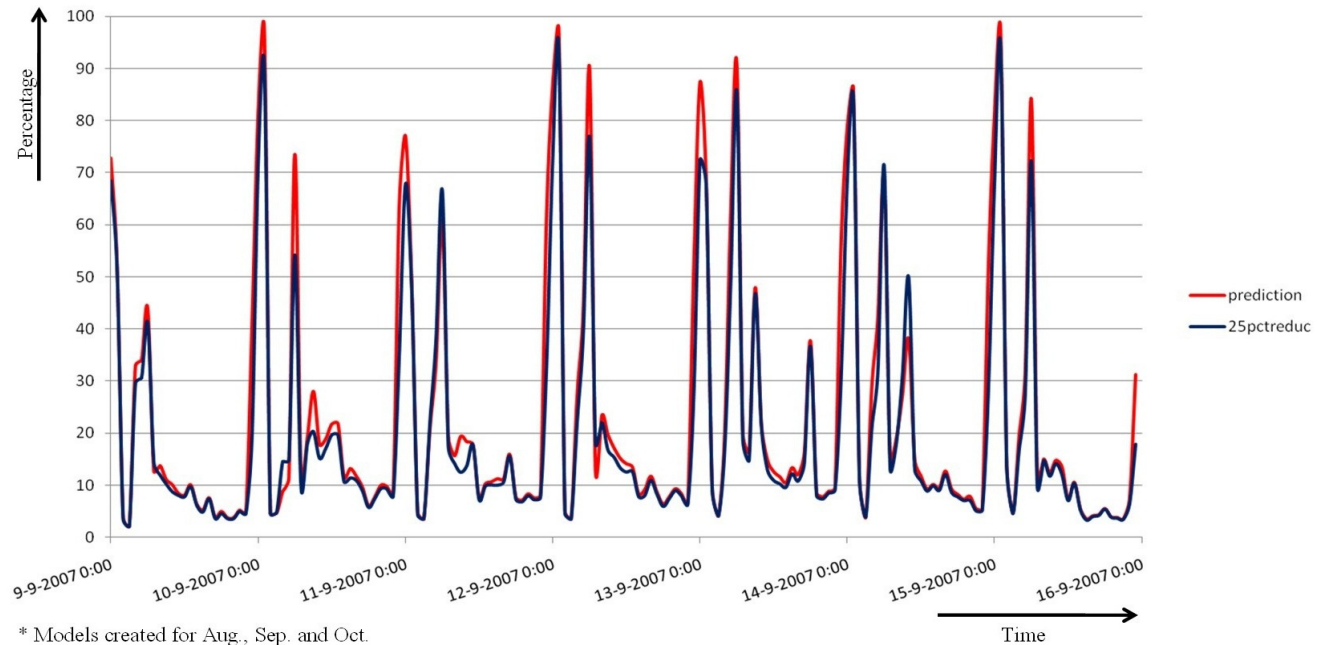


Figure 34: Graph of the predicted percentage failures of response times higher than 5 seconds versus the prediction when I/O was reduced with 25% over one week in September 2007. The model was based not only on the week presented, but on months August, September and October 2007. Modelling was done with the `dbms_predictive_analytics.predict` routine.

The graph shows clearly how the new response time would look like, after reducing the I/O with 25%. Based on the outcome we can decide if the hardware investment would be worth the money. So we have an instrument now in the form of “Method-GAPP” to do predictions of the new situation before actual doing the investment.

After doing a deeper investigation of the involved hard disks in the factorial analyses the problem could be pinpointed to a wrongly setup in the backup of the system. A simple change of the involved backup procedure made the performance problems be acceptable low.

Customer Case: Time and Labor Application

In this customer case a complex environment was analyzed, the system consist of one big physical machine with on top of it 15 virtual machines (LPAR's at IBM). The case here is a very good example on how not directly involved components can have a big influence on the measured response time. In figure 35 the environment is schematically represented. In the representation the directly involved LPARS are written in red (in the application “chain”).

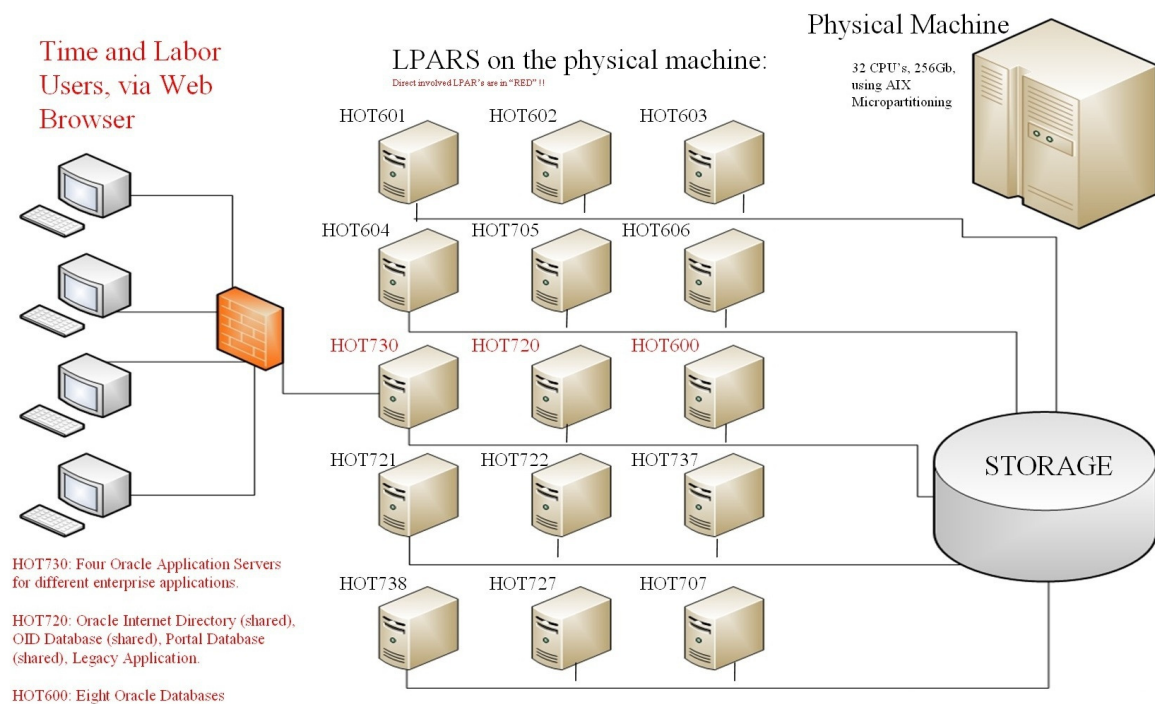


Figure 35: Figure shows how schematic the customer technical architecture looks like. The application is a Time and Labor application. The architecture consists of 15 LPAR's (virtual machines) on one big physical AIX machine. The LPAR's "HOT730", "HOT720" and "HOT600" are in the "chain" of the application. The other 12 LPARS are on the same physical machine, but have no functional impact in the application process (chain) analyzed.

The response time of the involved end-user process is shown in the graph in figure 36.

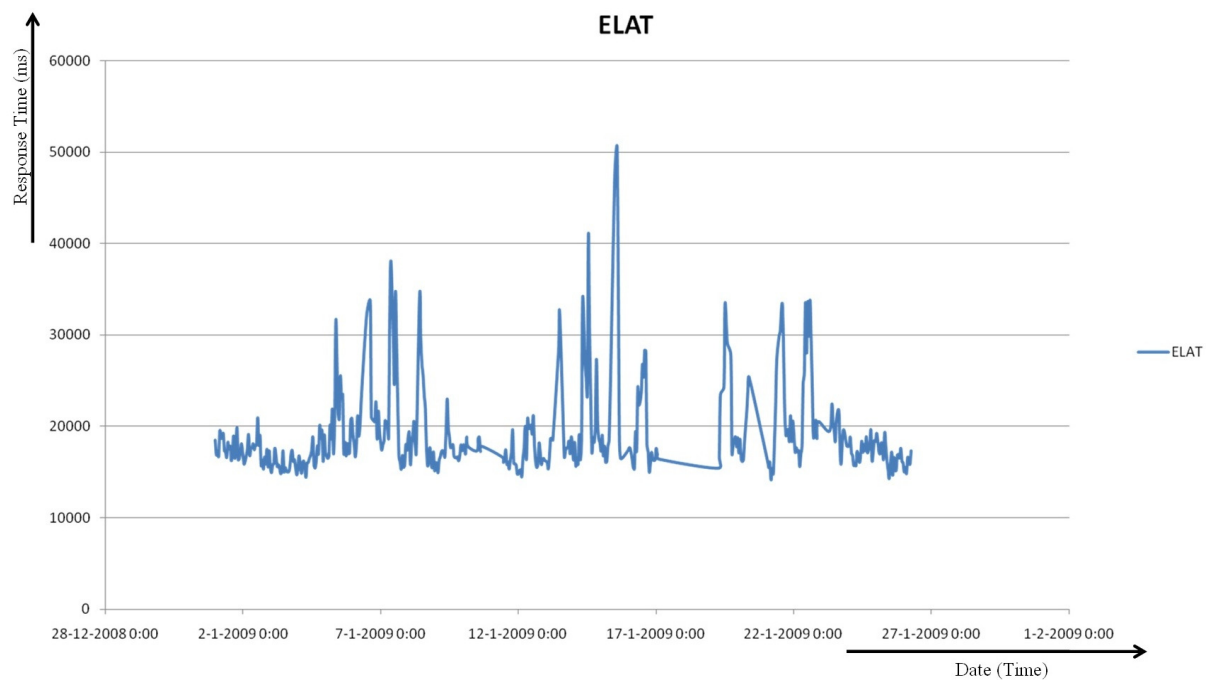


Figure 36: Graph of the measured response time (ELAT) versus the elapsed time (month January 2009).

The graph shows that there is a lot of variance in the measured response time. To get insight in the components which are of high influence on the variance we do a factorial analysis on components of the LPAR's direct in the "chain" of the application. The result of this factorial analysis is shown in figure 37.

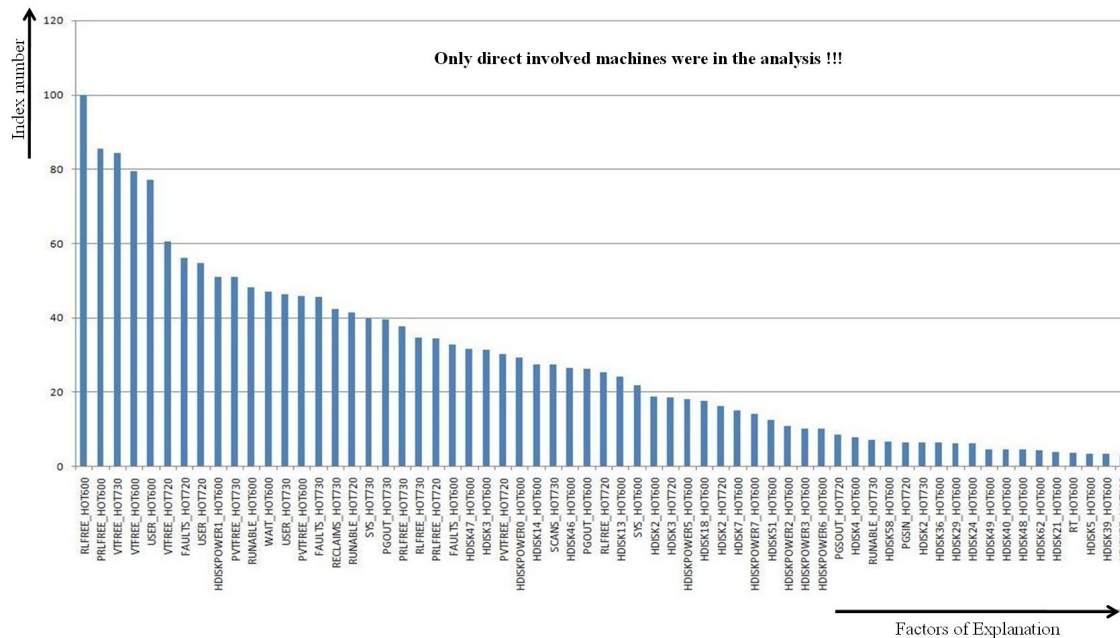


Figure 37: Graph shows the factorial analyses of the measured response times. In the graph the first columns have to do with memory on different LPAR's. In this predict run only the direct involved LPAR's were in the analysis.

The factorial analyses shows high impact of the memory of different LPAR's in the "chain". Based on the factorial analyses we can create a model with the procedure "PREDICT". The result of this model is shown in figure 38.

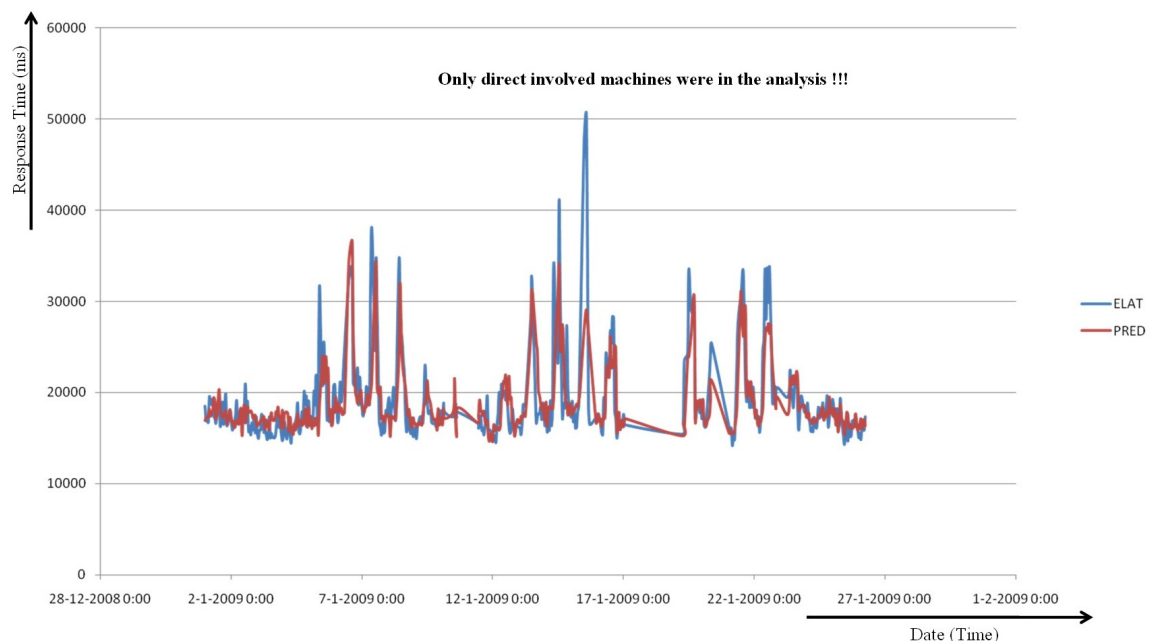


Figure 38: Graph of the measured response times (ELAT) versus the predicted response time (PRED) over one month in January 2009. Modelling was done with the dbms_predictive_analytics.predict routine, over only the direct involved LPAR's.

The graph shows that a lot of the measured response time is predicted, but for some reason some high peaks are not predicted at all. At this time we probably don't have enough data in the model to model the measured response time well.

After we draw this conclusion also data from the other LPAR's on the same physical machines was collected and put also in the database. So the collected data also contained now the not directly involved LPAR's. The result of the factorial analyses is shown in figure 39.

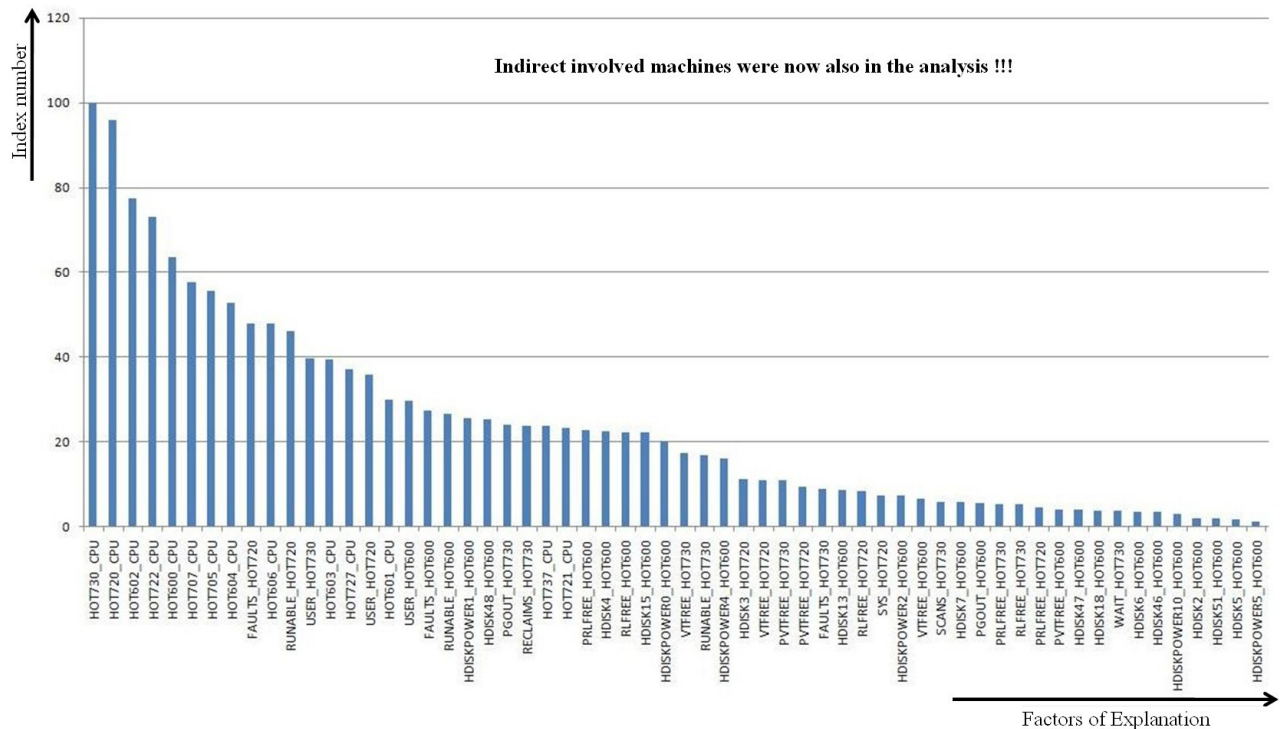


Figure 39: Graph shows the factorial analyses of the measured response times. In the graph the first columns have to do with the physical CPU's at different LPAR's. In this predict run also the indirect involved LPAR's were in the analysis.

The factorial analyses show now a total different view on the situation. The first bars in the factorial analyses have all to do with the physical CPU amount used of not only the directly involved LPAR's, but also of the LPAR's which are not directly involved.

Based on the factorial analyses a new prediction graph is created and shown in figure 40.

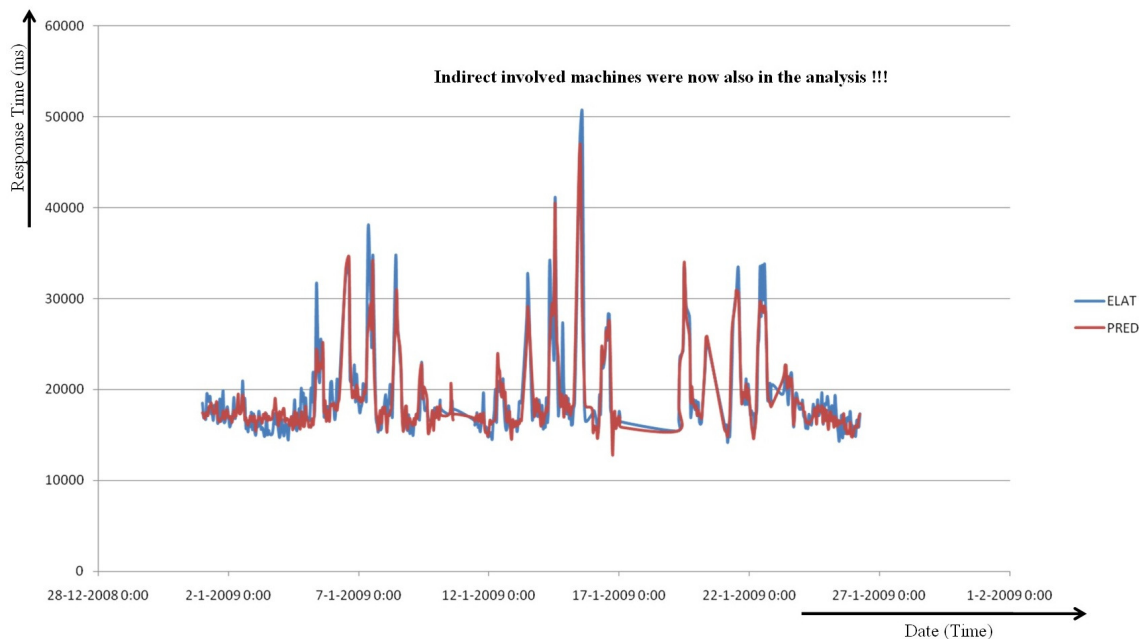


Figure 40: Graph of the measured response times (ELAT) versus the predicted response time (PRED) over one month in January 2009. Modelling was done with the `dbms_predictive_analytics.predict` routine, over also the indirect involved LPAR's.

The graph shows that the new created model is very well predicting the measured response time. When we start to interpret this data we come to the conclusion that there is a very strong indication that competition between the virtual machines, trying to get the CPU resources out of the free pool of “physical CPU” is the bottleneck. The virtual machine with the highest importance on the “physical CPU” is probably having a competition with the needed CPU's out of the free pool of “physical CPU”.

Further investigation showed that the not directly involved LPAR's were running applications which caused a competition on getting the free “physical CPU” out of the pool. Simply changing the priority and some changes in the run schedule of these applications solved the problem.

Conclusions

After working with “Method-GAPP” for several years the method has proven to be a very powerful method to find out where in a technical architecture performance bottlenecks can be found, what will happen in future scenario’s and what components outside the “chain” are influencing the end-user response time. The big advantage of the method is the fact that it doesn’t rely on a specific platform, it can basically be used in any IT environment, so it can be said that the method is a real **General Approach of Performance Profiling**.

When we start at first with the method it looks very complicated but after getting some experience it is easy to use, especially in complex environments. A very good way to position the method is in an enterprise performance management situation. When in this position, data is collected enterprise wide and all collected data is put in one big performance metric data warehouse. Together with response time data of several important business processes over the entire enterprise, we can use the method at its full capacity. It is able to find relations between applications which probably were not known and in this way the method can also be used to see how processes should be scheduled enterprise wide. Also strange performance degradations can be explained and tunings effort can be done on the performance bottlenecks found.

The method is also very successful in situations where limited data is available. In principle “Method-GAPP” can already do a good analysis only on the available data. The only thing what is most of the time not standard available, but necessary, is the response time data of the end-user process (business process) to be investigated. Although also this data is in some companies collected to see if the performance / availability SLA’s are met. The used data mining can be used of Oracle Data Mining (ODM) but also from another data mining provider (e.g. Mahout (Hadoop)).

The data modelling inside the method makes it possible to find out what the amount of service time (S) and wait time (Q) is for specific part of the architecture. So when looking at the small test we did we could determine that the amount of service time for the I/O was about 6,19ms (S_{IO}) per request. This can give an indication if we need to tune e.g. SQL to bring down the service time (S_{IO}). Having the formulas via the linear regression data mining can be of great help to see exact the impact of each individual component.

As we have seen in this whitepaper, the predictions of the models created via the method can help us to checkout if future investments would be worth the investment. This makes the method to be a powerful instrument of decision making for hardware / software investments or configuration changes.

For the future of the method it will be nice if the used models could be enhanced (currently M-curves, based on Erlang-C and linear curves). Probably we can use models also having coherency inside, this will probably enhance the reliability of the models. Also the ridge regression should be checked out further to make the models also better. Further more the choice of components might be further enhanced.

References

Optimizing Oracle Performance – Cary *Millsap 2003*

Analyzing Computer System Performance with Perl::PDQ - Dr. Neil J. *Gunther 2005*

1 page 81, formula 2.66

2 page 81, formula 2.64

Oracle® Database PL/SQL Packages and Types Reference 11G – *Oracle 2010*