Resources and Capacities
Michael Kok
(3rd in a series)

Hardware resources, capacities and Moore's law

So far we have seen two of the main inputs of an application performance model: its Performance-DNA and the usage volume. In my appreciation of modeling it is of crucial importance to be able to analyze the performance of both the application as whole and each of its transaction types individually. This is called Transaction Aware Performance Modeling or TAPM. So Performance-DNA preferably should be made visible for each transaction of an application.

In the previous article we had a look at resources utilizations caused by a certain usage volume while silently assuming capacities. This means that the model described the resource behavior of the application specifically for the environment where the measurements were taken. This environment has certain capacities. But it is only a test environment, which is not very interesting by its own. It is the application performance on the target production environment that is interesting. Therefore in this article we are going to have a look at what happens when the application runs in an environment where the characteristics of the hardware resources change. For these hardware resources we can change their capacities within the model, or what is called "scaling of the hardware". We have also the options of horizontal or vertical scaling. We have not yet seen the time behavior of an application so let's have a look at it before we look at scaling.

Time behavior under the load of the usage volume

When the usage volume is increased the resources start to be loaded, which shows by their %utilizations. When these %utilizations increase, for the underlying resources can introduce wait times. The higher the %utilization the higher the waiting time. They increase exponentially, not linearly with loads. When the %utilization approaches 100% the waiting times can become excessive. These waiting times are calculated by the performance model and are added to the Performance-DNA. Let's have a look at this using the example in article 1. An application with only one transaction type was observed on a very simple infrastructure.

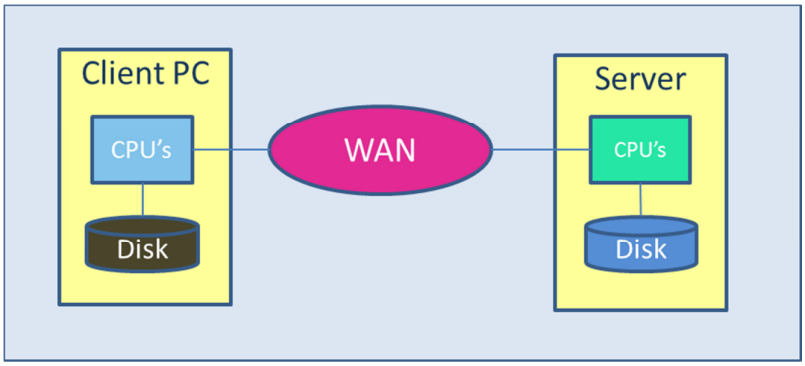

Figure 3.1 Infrastructure

This application has the following Performance-DNA:

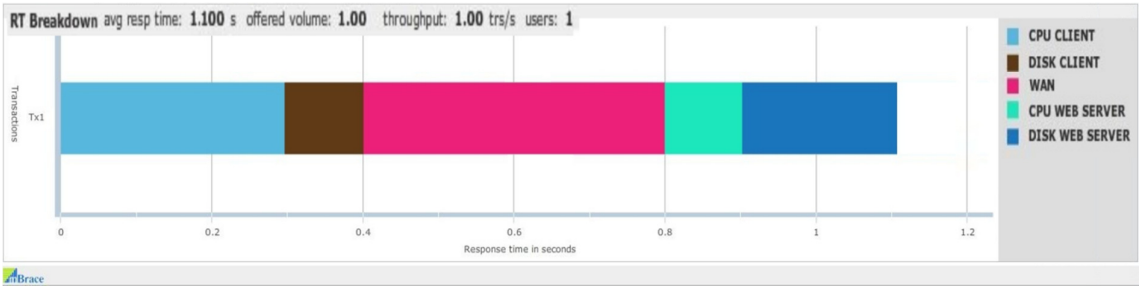

Figure 3.2 Performance DNA

Next the load on the application is increased, 100 users generate a transaction volume of 9.49 transactions / second. The hardware resources get loaded at a certain utilization percentage and queuing occurs for the resources. In this example only the CPUs of the server have a clear utilization percentage of 75%. At this utilization a waiting time for the CPUs of the Server is caused that visibly increases the response time. The one disk used by the application is utilized for only 2%, which does not generate a waiting that can be sensed. The next figure shows the utilization percentages of CPUs and disk. In this example the utilization on the WAN is negligible, therefore it is left out of the figure.

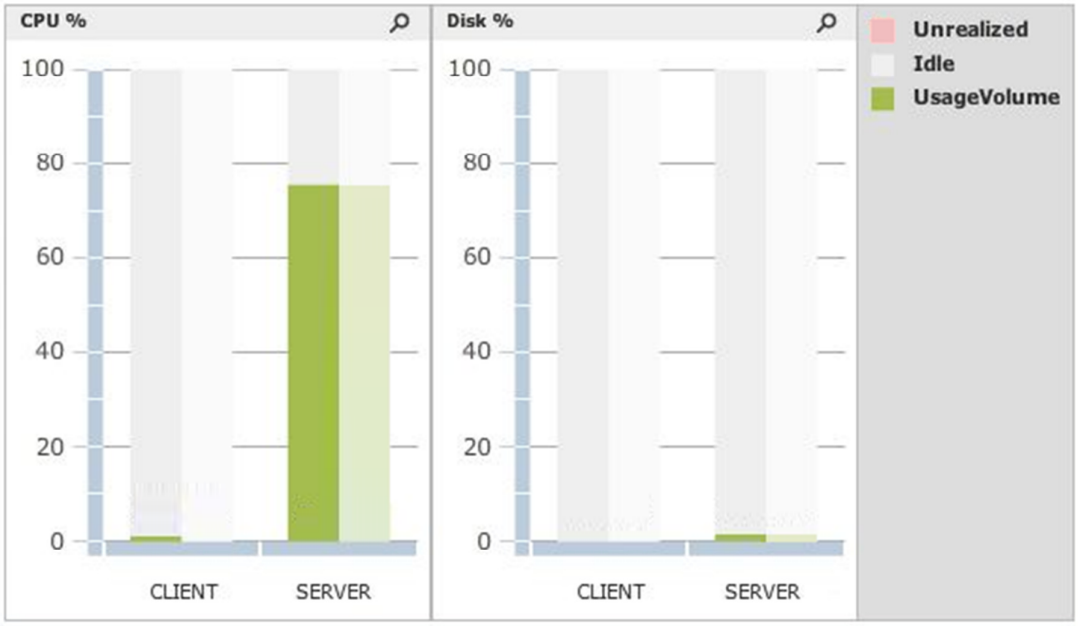

Figure3.3 Resource behavior

The next figure shows the time behavior of the transaction type of the application at the increased load. This is called a predictive projection or projection of the response time of the application.

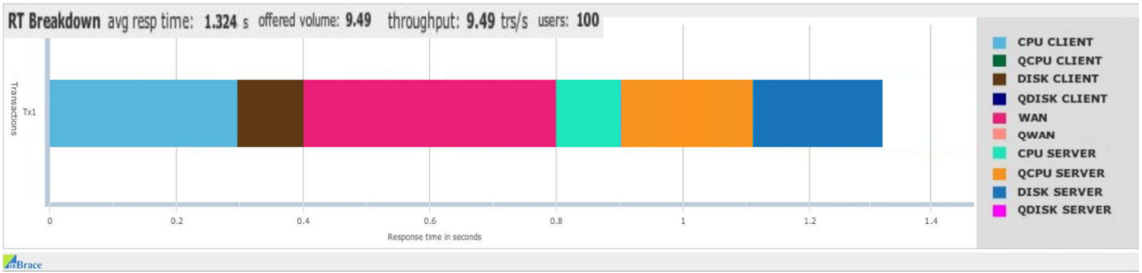

Figure 3.4 Time behavior

The figure reveals a number of features:

Four key metrics are displayed at the top: **average response time**, **offered volume**, **throughput** and number of concurrent **users**. When the throughput is lower than the offered volume we have one or more bottlenecks. The legend changed in the figure of the projection. In Figure 3.5 the legends of the above charts are compared.

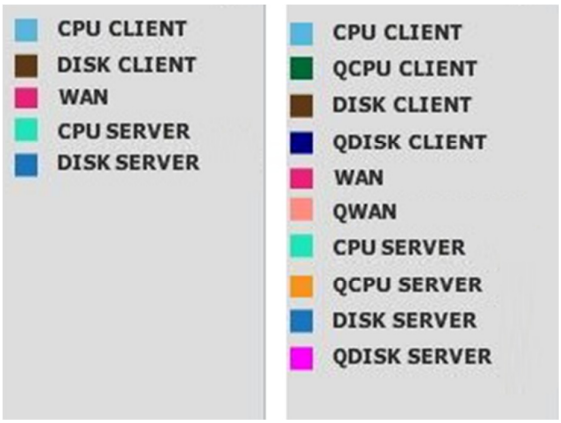

Figure3.5

The items in the legend of the Performance-DNA refer to the hardware resources. In the chart of the response time breakdown the colors show the time spent at the corresponding resource. In the other legend of the projection new items are added that have the letter "Q" in front of the resource name and that introduce new colors. **QCPU Client** represents the time spent while waiting for the CPUs of the Client PC. **QDISK SERVER** represents the time spent while waiting for the disks of the Web Server etc. With these new items and their colors we can show the impact of waiting for resources in the response time.

There is only waiting time spent for the CPU´s of the Server. Consequently only the color orange of **QCPU SERVER** shows in the response time breakdown. No waiting times occurred for other resources, obviously because no waiting times occur for the other resources because there utilization percentages are still low at this load. So no other new colors appear in the figure.

The lengths of the other colors have remained unchanged, meaning that the times the transaction spends at these resources remained the same. This means that the Performance-DNA does not change. In other words, the hardware resources are load independent.

The response time increased from 1.1 to 1.3 seconds, the difference this time exclusively being caused by the orange part of **QCPU SERVER** only.

It is important to recognize that when the load increases, the response time increases only by the waiting times introduced while the original Performance-DNA remains the same. There is one exception that I will describe further in this article.

As a next step in this example we further increase the usage volume by increasing the number of users from 100 to 300. Now the utilizations look as follows:

Now the CPUs of Server have a utilization of 100% and the disks of 6%. Clearly the CPUs of Server are a bottleneck. Bottlenecks can be easily recognized. In the next chart each resource has two adjacent vertical bars. The left bar represents the utilization percentage realized on the resource at the particular usage volume. This part shows values between 0% and 100% and will never exceed 100%. The right part (called the shadow bar) shows to what value the load targets the utilization percentage. In this particular example of a bottleneck you see that the shadow bar shows a value of 224%. The part exceeding 100% is colored red and is called Unrealized Utilization. A percentage exceeding 100% is of course physically impossible, but models allow us to calculate this theoretical value. This is very handy because it allows us to determine immediately how the hardware resource should be scaled, even more; it enables us to perform auto scaling of the resources in the model.

Figure 3.6

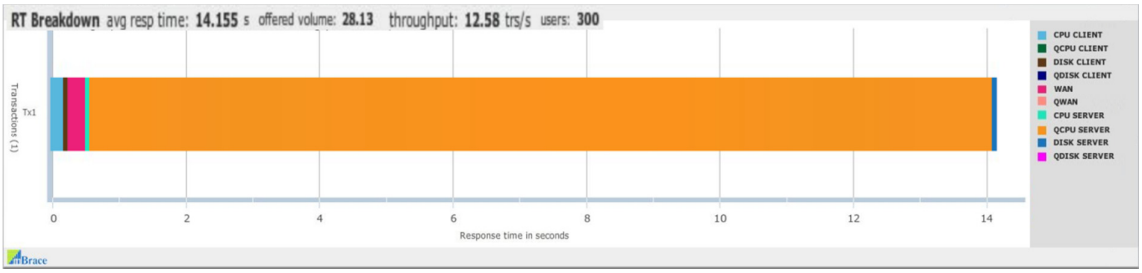

Figure 3.7

The above figure shows the time behavior of the application with an excessive response time of over 14 seconds. Clearly more than 13 seconds is caused by waiting for the CPUs of Server. So the two charts in combination show at a glance what the cause of the performance problem is, lack of CPU capacity in Server. The key metrics further show that of the offered volume of 28.13 transactions per second only a throughput of 12.58 transactions per second is realized by the 300 users.

The next figure shows an interesting example of the resource behavior of an application on a three-tier infrastructure. The application has a performance problem due to insufficient CPU-capacity. Where is / are the bottleneck(s)? We can immediately see three bottlenecks on the CPUs of all servers. The CPUs of Web Server are loaded to 100%, which is a clear sign of overload. The shadow bar (the right part of it's vertical bar) shows a target utilization of 675%, so the capacity must be increased with a factor 12 to reduce the utilization below 60%. The CPUs of Database Server are utilized at 40%. It is obvious that full usage of the Database Servers CPUs is blocked by the Web Servers bottleneck. As soon as the Web Server CPU capacity is sufficiently increased, the Database Servers CPU utilization will increase to 100% and they become the next bottleneck blocking in its turn the CPU usages on both Application Server and Web Server. The shadow bar of the Database Server CPUs shows a target utilization of 260%, which indicates that we need to increase the CPU-capacity with a factor 5 to reduce the CPU utilization below 60%. The Application Server CPUs have a utilization of 15% and a target utilization of 100%. After increasing the Web Server capacity the CPU utilization of Application Server increases to 37%. After increasing the CPU capacity of the Database Server the CPU utilization of the Application Server raises up to 100% as was predicted by the shadow bar. Originally the %utilization reported for Application Server was 15% while the shadow bar indicated 100%. This means that the CPUs of Application Server are a

bottleneck and when all other bottlenecks are removed its %utilization will become 100%. So right from the start we know that the capacity of the Application Server's CPUs must be increased with 50% in order to bring the % utilization down from 100% to 60%.

Once the picture with the target values of utilization is understood we can change all capacities at once.

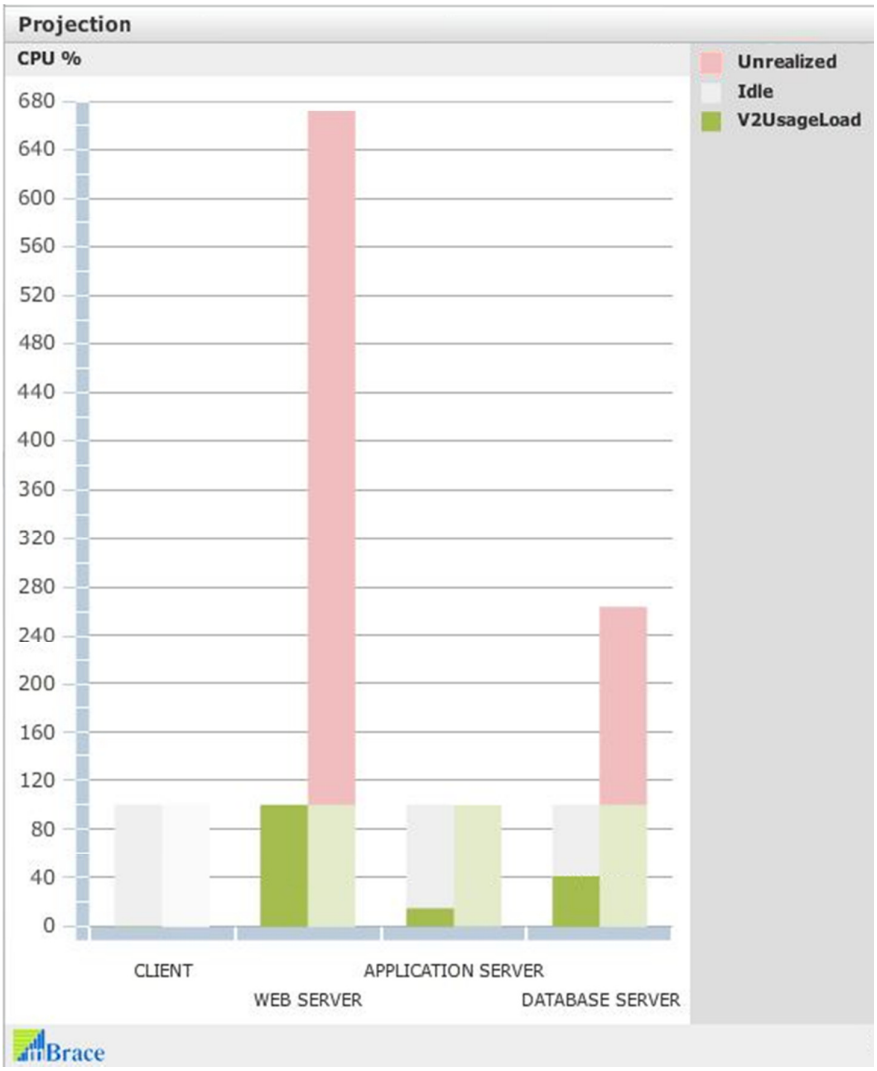

Figure 3.8.

This example demonstrates that when observing the realized utilizations only one resource showed to be a bottleneck with its utilization of 100%. For the same it might have been possible that the prominent bottleneck was a software resource. As a result none of the bottlenecks would show 100% utilization so that all hardware bottlenecks are obscured as is the case in practice many times.

It takes calculation of the target values to get some clue about what the situation is. Without them it is not possible upfront to exactly identify the bottlenecks and how they can be solved.

The following figure shows the result of the above scenario.
The CPU capacities are now chosen in such a way that all tiers have about the same utilizations. This is called a "balanced configuration".

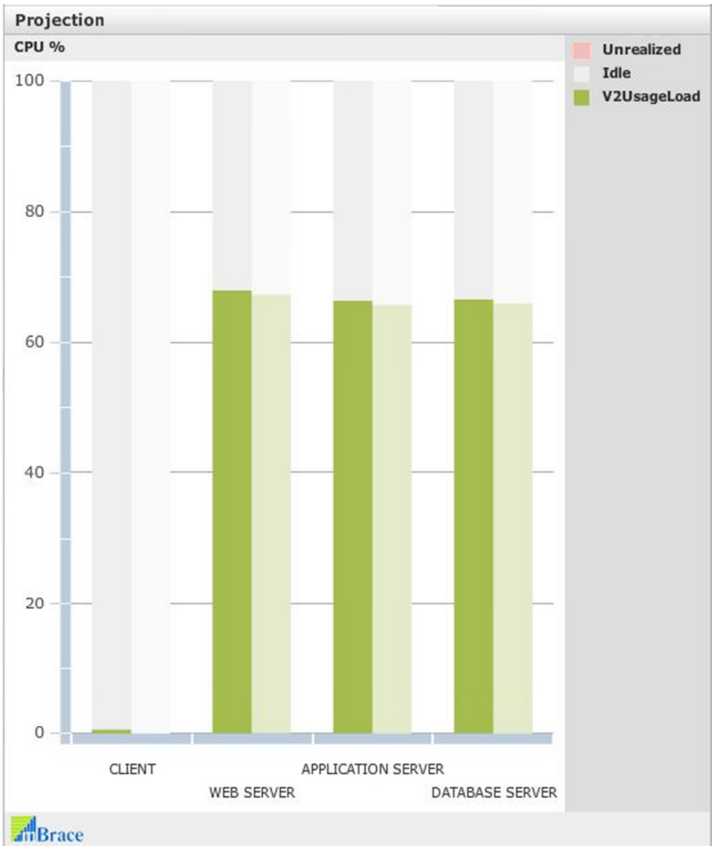

Figure 3.9.

Comparison with conventional load and stress testing

This scenario usually is applied in an alternative load and stress testing procedure. The example of a capacity optimizing scenario reveals two other interesting capabilities of performance modeling. In the conventional load and stress testing procedure you increase the volume until you run into the first bottleneck. Then you first have to increase the capacity of the bottleneck and rerun the test until you stumble into the next bottleneck. Increase its capacity . . . etc. With modeling you only need to take the measurements in one test cycle, feed the Performance-DNA into the model and conduct the scenario. And no destructive stress testing, just determine the max throughput in the model.
The other feature? When you do a conventional load test, you need a test environment with production like capacities. For the modeling alternative procedure you can use a small test environment and test the application by scaling from the test environment to the production environment.

By the way, the waiting times are calculated using queuing theory. Since this series of articles focusses on how to use models and what can be achieved with them, rather than how they are constructed. So queuing theory is not a subject but I may describe some aspects of queuing theory in a qualitative manner. To readers who are interested in this subject there are many books on the subject. I like the books of Dr. D. Menascé et al. *Performance by design* and *Capacity planning for web services*.

Specification of hardware resources and scaling

We have to specify hardware resources and their capacities as input for the model. Next an example of two analysis scenario's is presented: 1) using horizontal scaling and 2) vertical scaling. Let's first have a look to how

to specify hardware capacities in a model.

In the simple example of a client and server, we deal with the following resources: CPUs, Disk storage and a WAN. See the next table. First we set up the existing resource capacities in the model (the baseline) and then alter them to study the effects with help of the model. The baseline capacities relate to the state of hardware in Figure 3.7. The outcome of this scenario is a predictive projection.

Horizontal scaling

Capacity specs	**Baseline**			**Projection**		
	Client	Server	WAN	Client	Server	WAN
CPU						
Number of CPU's used	2	**2**		2	**8**	
Relative speed	1	1		1	1	
Disks						
Number	2	**1**		2	**2**	
Relative speed	1	1		1	1	
WAN						
Bandwidth in Megabits per second			100			100

Table 3.1. Resource specification for horizontal scaling scenario

The changes in hardware capacities are marked yellow in the table above. E.g. In the baseline Server has 2 CPUs and in the projection we study the effect of increasing that number to 8. Also the number of disks is increased from 1 to 2. The capacities of the WAN remain unchanged. In fact for the Projection we provide 4 times as much CPU capacity, by increasing the number of CPUs. The disk capacity is doubled by providing twice as many disks. This is called horizontal scaling.

Vertical scaling

The next table shows a scenario of vertical scaling. Departing from the state of capacity overload shown in figures 3.6 and 3.7 the CPUs of Server are replaced by ones that are four times as fast. In addition the disks are replaced by ones that are twice as fast. This means that we increase the capacities by the same factors as we did by horizontal scaling, however this time not by increasing the numbers but the speed of the resources. Let's have a look at the outcomes and compare horizontal and vertical scaling.

Capacity specs	**Baseline**			**Projection**		
	Client	Server	WAN	Client	Server	WAN
CPU						
Number of CPU's used	2	2		2	2	
Relative speed	1	**1**		1	**4**	
Disks						
Number	2	1		2	1	
Relative speed	1	**1**		1	**2**	
WAN						
Bandwidth in Megabits per second			100			100

Table 3.2. Resource specification for vertical scaling scenario

Figures 3.6 and 3.7 show the baseline performance with the CPU bottleneck. The next figure shows the impact of both horizontal and vertical scaling. The effect on resource behavior is the same for both horizontal and vertical scaling. Figure 3.6 showed a CPU utilization of 100%, with a target of 220%. The next figure makes clear that increasing the CPU capacity by a factor 4 decreases the utilization by the same factor down to 55%. Since the number of disks increased by a factor 2 the utilization of the disks decreases by the same factor 2 from 6% to 3%.

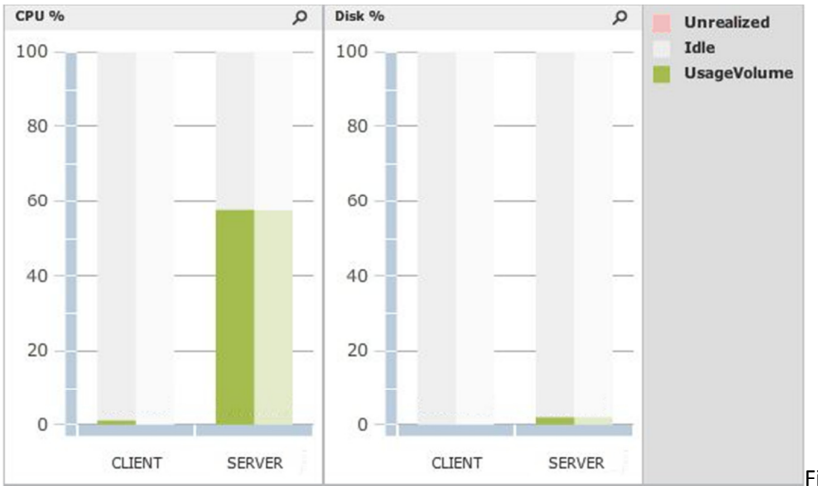

Figure 3.8.

In the next figure the time behavior is compared for horizontal and vertical scaling.

With horizontal scaling the response time is reduced to 1.25 seconds, offered volume and throughput are equal again which confirms that we got rid of the bottleneck. We did not change the number of users so that is still 300. The parts of the Performance-DNA have not changed, only the queuing for the CPUs was reduced due to the lower utilization of the CPUs.

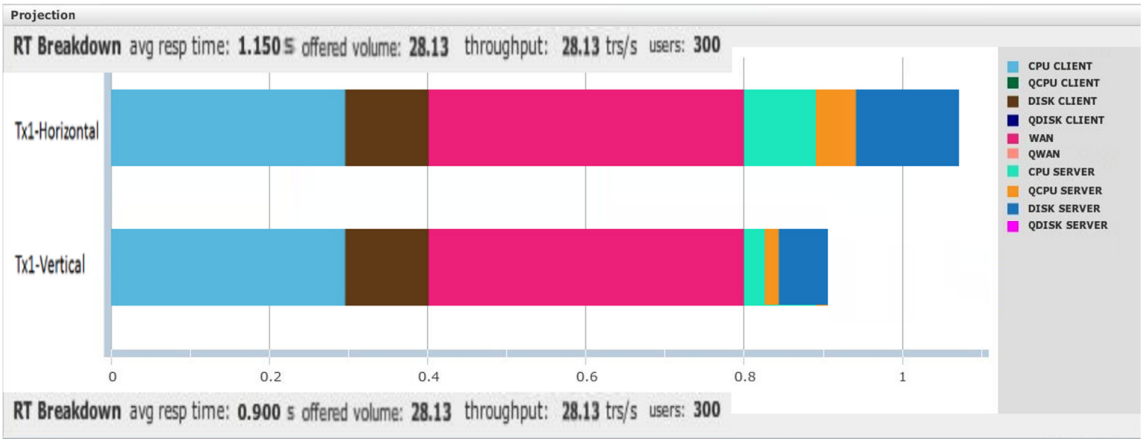

Figure 3.9.

The response time breakdown shown for vertical scaling in Figure 3.9 entails interesting differences with the outcome of the horizontal scaling scenario. The response time is now 0.90 seconds. The parts of **CPU CLIENT**, **DISK CLIENT** and **WAN** remain unchanged. However **CPU SERVER**, **QCPU SERVER** and **DISK SERVER** have changed significantly. By comparison we can see that the time for **CPU SERVER** is reduced to a quarter of the original value due to the increased speed (factor 4) of the CPUs. Similarly the time for **DISK Server** is cut in half due to the double speed of the disks. In fact the Performance DNA was altered due to the vertical scaling of the resources. This is the exceptional example where the Performance-DNA changes. The queuing time depends both on the CPU utilization and the time spent on the CPUs and has shortened drastically.

Comparing horizontal and vertical scaling we may conclude:

- With horizontal scaling the performance improves only when we have a bottleneck because the utilization of the bottleneck-resource is reduced, which leads to shorter waiting times.
- With vertical scaling application performance improves always because the Performance-DNA improves. In addition the performance improves in case of a bottleneck since the waiting time decreases.

Vertical scaling seems to be more advantageous, but one has to consider the costs too. Increased CPU-speed is a rare phenomenon these days. I remember a case in 2006 where we tested an application on an old IBM RS6000 machine and projected that to a model from the IBM P570 series. The latter had CPUs that where 4 times faster. Today CPU speed does not increase as spectacular as it used to. Instead the market offers increasing numbers of cores. In order to convert that into speed, our industry is challenged now to apply internal parallel processing in applications. It looks like Moore's law works in a different way. Performance models should be capable of handling parallel computing.

In modeling we use the relative speed of resources, which is the ratio between the baseline and the target speeds. For most CPUs you can find their speed at SpecMark, which is an independent institution for benchmarking CPUs. The speed of single CPUs is expressed in the CINT2006 benchmark.

Hardware specification for modeling input.

As usual the focus was on CPUs and as usual we realize of course there is more than that. The other resources are dealt with by the model in a similar way as the by changing their specifications.

Here is a more complete overview of the data you have to collect for a resource specification:

	Client	Cluster1	Cluster2	Cluster n	WAN	LAN 1	LAN 2	LAN n
Cluster								
Number of servers in cluster								
Server								
Number of CPU's per server								
CPU								
Number per server								
Total number used								
Speed								
Storage								
Cache								
Hit rate								
Hit time								
Disks								
Number								
Miss time								
WAN								
Latency								
In bound bandwidth								
Outbound bandwidth								
LAN 1								
Latency								
Bandwidth								

Virtual servers

Virtual servers affect performance in two ways:

1. Virtualizing imposes extra overhead that materializes in extending the CPU-components of the Performance-DNA. You have to find out how much overhead for all resources. For CPU commonly 10% to 15% has to be added. (Any reader who has different figures?)
2. Virtualizing allows for providing CPU capacity by fractions of CPUs. This results in a form of vertical scaling. If the application is assigned 0.4 of a CPU this should be modeled as the original CPU at 0.4 relative speed.

More complete examples

So far the examples were based on a simple case of an application with only one transaction on a one-tier infrastructure chain. Modeling systems completely, entails of course more than CPUs alone. Commonly we also have storage systems with cache and disks, networks in- and outbound and software resources.
To give you an idea of what more complete systems look like next two examples. Even these examples are limited in the number of transaction types. Commonly an application performance modeling study covers 20 to 100 transactions types.

The next example shows 8 transaction types on an infrastructure with 3 servers and 4 networks, WAN and four LANs. Network C is the WAN. It is too complicated to make all graphic details legibly. At the same time it is important to understand that it is the patterns that count in the first place. From the explanations so far you should be capable now to understand what they express.

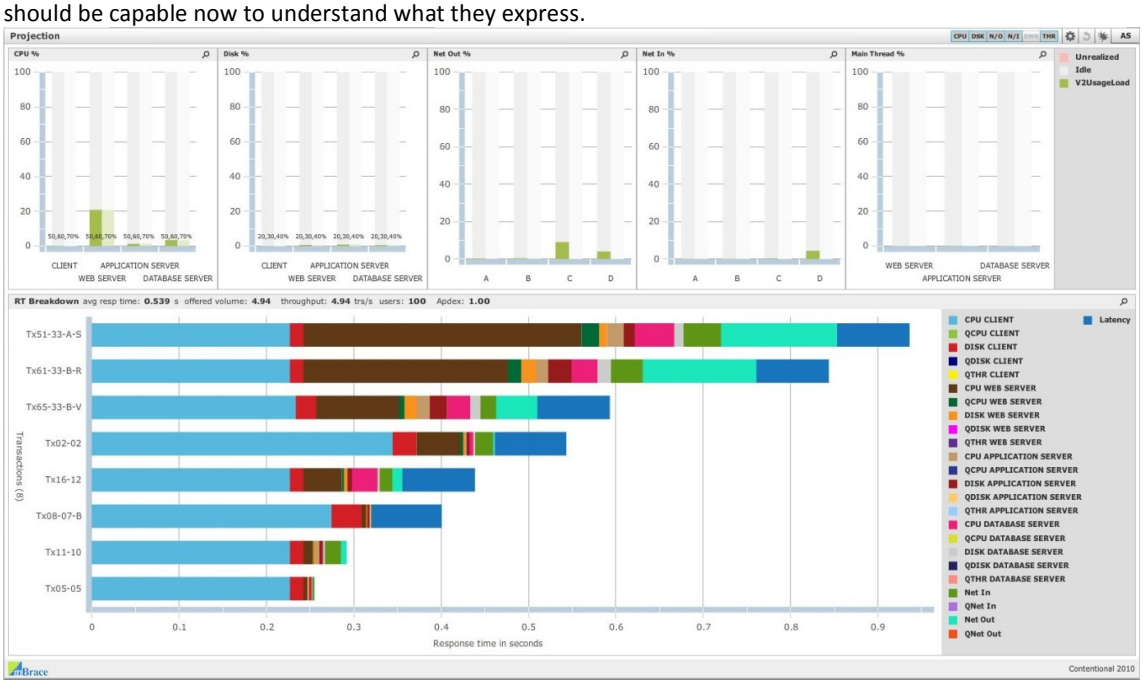

Figure 3.10.

The above example shows no bottlenecks.

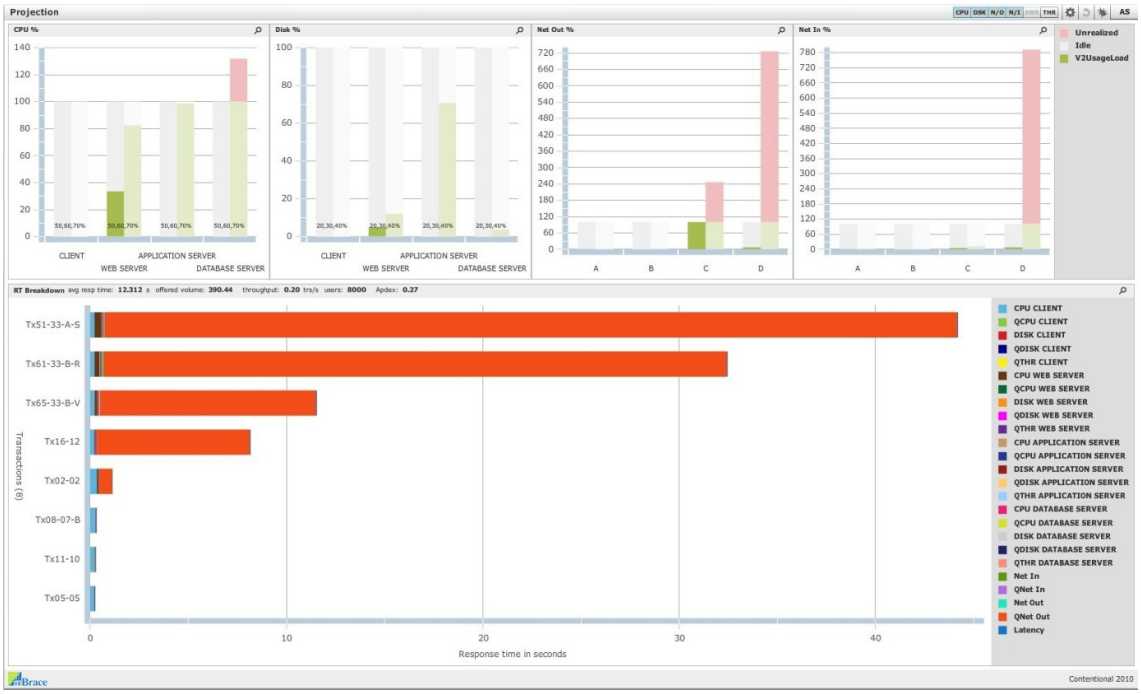

Figure 3.11.

The above figure shows a predictive projection of the same system for time and resource behavior at 8,000 concurrent users. The time behavior displays one dominant bottleneck, which happens to be the outbound part of network C . At the resource behavior 5 bottleneck are visible.

Summary

In this article we saw how the charts of the Performance-DNA transform into predictive projections when the usage volume of the application is pumped up. The Performance-DNA remains the same but waiting times due to contention for the resources is added, extending the response times of the transaction types. The metric that seems to drive this is the utilization percentage of the resource.
With help of a performance model we can study the way application performance is affected by horizontal and vertical scaling of hardware resources.
For this we

- Specify the characteristics of the hardware
- Conduct (analysis) scenario's with the model. In these scenario's we can study the effects of changing the usage volume and scaling of the hardware resources.

The Performance-DNA of an application is invariant when the load changes since the hardware resources are load-independent resources. There is one exception, vertical scaling, that affects the Performance-DNA.
Most important: we predict the application's performance.

In the three articles so far the basic concepts of modeling, most of it about the inputs and outputs, have been described. This is basic stuff, in the following articles now we can look at some real interesting things.

Any questions, debate, criticism? Email: michael.kok@mbrace.it.