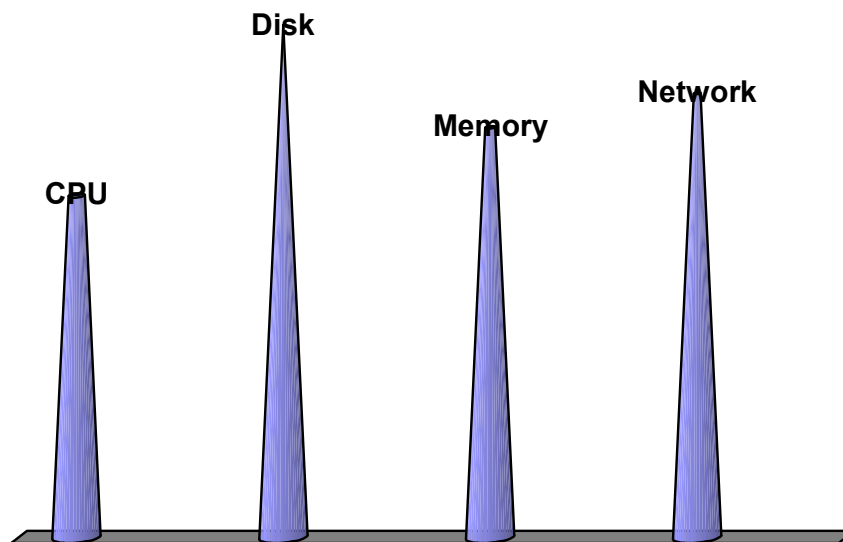


Scalability/Performance testing of server software



This article is intended to discuss the concepts of performance and scalability testing with respect to four resources CPU, disk, memory and network. The four resources are related to each other and we need to completely understand their relationship to implement the strategy for scalability and performance testing.

Like the children in the home the four resources in the computer require equal attention, as they want to grow together to enhance the scalability and performance purpose. Many software engineers do not understand the purpose of this growth very well. All these resources are interdependent and will have issues if other resource is modified. For example, if the memory requirements in the system are addressed, the CPU will become more intensive. This in turn results in multiple cycles of upgrade as the requirements of the customers keep increasing. All these resources tend to grow, when a new release of the product happens. These aspects show only upward growth as software becomes more and more complex. Multiple upgrade cycles were possible till last few years. Due to several factors like recession in economy, cost consciousness and software being utilized in plenty of other equipment such as Mobile phones, and PDAs, there is a requirement to change the development and testing strategy. This is needed to justify the growth of these four resources. The software is not only for computers but also for equipment; hence up-grading the resources are not easy anymore.

Let us discuss the following issues and steps to derive a good testing strategy based on some assumptions:

- Choosing the right product configuration
- Choosing the right resources
- Utilizing the benchmarking and scalability results

Choosing the Right Product Configuration

The confusion that often arises between the customers and software vendors is in deciding the right product configuration. This happens since the developers and the testers of server software assume only one typical customer scenario. They exhibit a reluctant attitude for appreciating the fact that each customer scenario is different and requires a specified customization at the design level of the product.

Often, the customers are lost when they have been told to increase the number of CPUs, memory, and the network bandwidth for better performance and scalability. Also, measurable guidelines to specify the level of performance/scalability improvement/degradation when resources are adjusted, is not provided. This results in multiple upgrade cycles.

Choosing the Right Resources

It is important to understand and acknowledge customer's view about the resources the product intensively uses and the resources that are critical for the product usage. It is easy to tell the customer that product A is CPU intensive, product B requires more memory and product C requires better bandwidth. Some of the products that run on a particular server could be from multiple vendors. But it is very difficult for the customer to increase all the resources in the server as all the products are expected to run in the same server at the same time. Many times the engineers are not aware of this fact and they look at only one product to give the solution, which often fails to meet the expectations of the customer. The concept of system is missing in development and testing of many companies. The product development teams do not get to decide what components constitute a system; it is rather decided by the customer and may involve multiple products from multiple suppliers.

Utilizing Benchmarking and Scalability Results

The benchmarking and scalability results achieved in the test lab are not repeatable for the customer in their setup. This is because of the fact that the product is tested in a simulated environment or by keeping all parameters such as memory, CPU, disk and network in "safe condition". These parameters are generally available to customers in the marketing docs.

The perspectives and requirements for each of the customer are completely different and it is very difficult to set the right strategy. Let us have a look at some basic assumptions before we proceed further.

- The CPU can be utilized, as long as the CPU can be freed when a high priority job comes in.
- The "memory available" can be used by the multiple threads of the software, as long as the threads exit after performing the task within reasonable amount of interval, and the memory is relinquished when another job requires memory.
- The cost of adding CPU or memory is not that expensive as it was before, to get better performance as long as we know what the increased % of performance and scalability is for each added resource
- Network packets can be generated by the product, as long as the network bandwidth is available and the cost is also less. There is a difference in this assumption that most of the packets generated are for LAN and not for WAN. In the case of WAN or routes involving multiple hops, the packets need to be reduced.
- More disk space or the complete I/O bandwidth can be used for the product as long as they are available. While disk costs are getting cheaper, IO bandwidth is not. This may limit the amount of "disk footprint" for a product.
- The customer gets the maximum Return on Investment (ROI) only if the resources such as CPU, Disk, memory and network are optimally used. So there is intelligence needed in the software to understand the server configuration their usage and auto-tune the parameters accordingly (e.g. Number of threads) to get optimum returns
- Graceful degradation in performance or scalability can be expected when resources in the machine are also utilized for different activities in the server
- Predictable variations in performance or scalability is acceptable for different configurations of the same product
- Variation in levels of performance and scalability is acceptable, when some parameters are tuned, as long as we know the impact of adjusting each of those tunable parameters.
- The product can behave completely different in low-end and high-end servers as long as they support ROI. This in fact motivates the customers to upgrade their resources.

From the above assumptions, you may get an opinion that our perspectives may be completely different from what customers really expect. The above statements can't be directly told to the customer in verbatim. However the above assumptions will guide us to derive answers to some of the problems that traditionally exist in software but never answered for the customers. These assumptions help us to create a basic scenario for performance and scalability.

Performance testing can be conducted in different conditions. The analysis of four resources (CPU, Memory, Disk & Network) gives us the following test scenarios;

- a. Record the normal (Without fine tuning the parameters) and the best performance data (after fine tuning the parameters) on a typical recommended server configuration (as documented in the product user guide as minimum requirement). Note down the utilization of CPU, disk space, disk I/O usage, memory usage, network usage (called baseline utilization)
- b. Record the normal and best performance data on a typical customer configuration (obtained from sales/marketing data - also called popular configuration). Note down the utilization of CPU, disk usage, disk I/O, memory and network usages (called normal utilization)
- c. Record the normal and best performance data on the specific customer scenarios and note down the resource usage (special configuration). There could be multiple special configurations that are possible depending up on how many customer scenarios we would like to simulate in the lab.

Tuning the configurable parameters of the product is a trial and error method and getting the best performance data may require multiple iterations of testing. Trial and error method doesn't mean you do the testing without any baseline data. Design and architecture of the product can definitely give indicative performance and scalability, which becomes the baseline as well as improvement area. Most of the time, the limitations in performance and scalability are introduced by design and architecture. Tuning as a procedure doesn't mean that we need to tune only product parameters. The tuning includes OS, services and product parameters. Trial and error method for understanding the benefit or the impact of tuning the parameters is normally a pre-system or pre-integration test activity. This should not be performed during test cycles. Also one need to understand that tuning a parameter for performance may affect another type of testing such as, reliability. Such points need to be kept in mind and setups that are created for performance/scalability testing need to be reused for other types of testing to understand the complete impact.

While we talked about setting up configuration, it may not be always possible to get a specific customer scenario. However the experience of testing various configurations would give some ideas about the expected performance levels, tuning details etc. Such points along with assumptions need to be documented so that customers can derive specific performance levels from the product.

Validating the performance results is another gray area of this discussion. There needs to be some basic assumptions that are to be made to validate this data. The actual expectation may vary from product to product. Some of the basic assumptions can be,

A11. The performance should increase 50% when I double the number of CPUs from minimum requirement (The performance here could be response time or throughput) and 40% thereafter, till x number of CPUs are added (normally you get breakeven at 16 or 32 CPUs). This aspect has to be tested with as assumption that the product is CPU intensive.

A12. The performance should increase 40% when I double my memory from minimum requirement and 30% from thereafter

A13. The performance should increase by at least 30% when I double the number of NIC cards or increase network bandwidth in the system

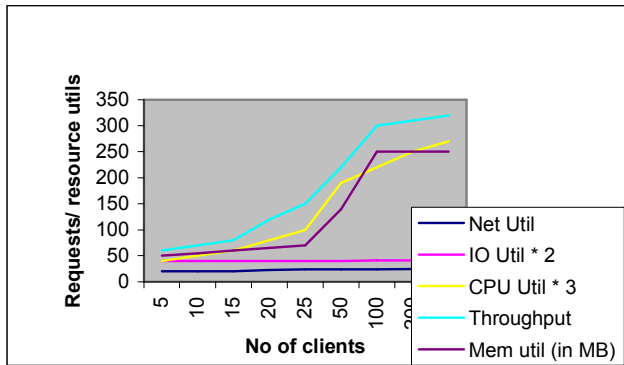
A14. The performance should increase at least by 50%, when I double I/O bandwidth

The above assumptions try to validate the four resources discussed, what and how they need to grow and the benefits because of the growth. These assumptions also help us to develop scalability requirements for the product. In the points said above, we have see situations where there is a demand for increasing the resources. Obviously, you can't see improvement in performance after upgrading the resources such as RAM, CPU unless utilization of those resources is quite high already. Unnecessary upgrades need to be avoided.

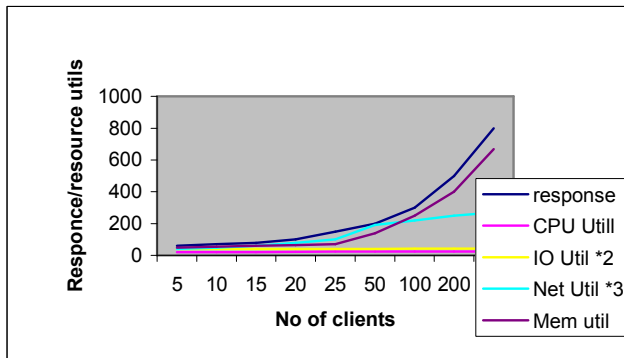
Metrics and Measurements for performance and scalability testing

Now it is time to propose a set of methodologies and define metrics and measurements based on the above discussion and assumptions. The metrics and measurements defined under have a strong relationship with the assumptions stated above, and these metrics need to be changed when one or more of the assumptions are changed.

Throughput Chart



Response Chart



In the above charts, both performance and resource utilization are plotted. The utilization values are multiplied and normalized to make them visible in the charts. Coming to analysis, in the throughput chart the throughput of the software is proportional to CPU and memory utilization. So, it is easy to conclude that increasing these resources can improve the performance. The order of the upgrade, for these resources can be memory and then CPU. Mem util line becomes flat after number of clients increased to 100 and this behavior can be supported by "malloc()" calls failing in the software or creation of further process threads failing.

In the response chart, the response time is proportional to the memory utilization. It doesn't mean memory needs to be upgraded, but memory to be kept in the watch list for a possible upgrade in future. This also means the no of clients or the load need to be increased to find out saturation limit. In this chart "network utilization" is also growing proportionally. If the software is meant for LAN environment this could be an acceptable behavior but not in the case of WAN as the data in chart shows it is reaching close to 90% network utilization.

As explained before, the resources (mem,cpu,IO,net) are interrelated to each other and experiences may not be same always. The order of upgrade (one at a time) is important to analyze the impact on performance. For example, lack of memory can prevent threads being created or destroyed, thereby causing CPU to schedule the jobs repeatedly. This will result in increased CPU utilization. This may create a "thrashing" scenario (memory pages copied between secondary memory and primary memory repeatedly) by which IO utilization may increase. In these situations it is advisable to increase only memory and repeat the tests to find out whether other resource upgrades are necessary. Further, all such scenarios and finding out solutions for those scenarios may require complete understanding of the OS, system, the products running in the machine and proper analysis of charts from testing.

In many situations, the product performance gets limited because of the design and architecture, and increasing the resources may not yield good results. In these cases one needs to analyze the product under test, before looking at upgrades. In this case, you will see product performance not improving irrespective of resource utilization (CPU, memory etc.) being kept low.

Sometimes a spike in resource utilization may be observed. It can be considered normal, as there may be background processes or a high priority request or the product itself is designed to work this way. In practice, these spikes are removed from charts using a technique called trend or noise reduction. It is very important to

analyze whether such behaviors happen due to the product under testing or other products/services running in the system. These variations can make the product functionality or the system to fail. Such noises need to be analyzed before being removed from the chart. Such problems are very difficult to reproduce at later point of time, if reported by customers.

By just looking at one chart we can't conclude the optimum resource usage of the product. The complete impact is known only when the product is tested with different test scenarios discussed below. The resources required by the product or system, have a close relationship with some of the tunable parameters of product or OS. When these parameters are tuned, they not only impact the performance of the system but also the resource utilization. The tuning of the parameters are also done to improve the resource utilization to the optimum value and also to limit the resources that are used by the product. The tunable parameters in the system, is an active factor responsible for the mountain's growth.

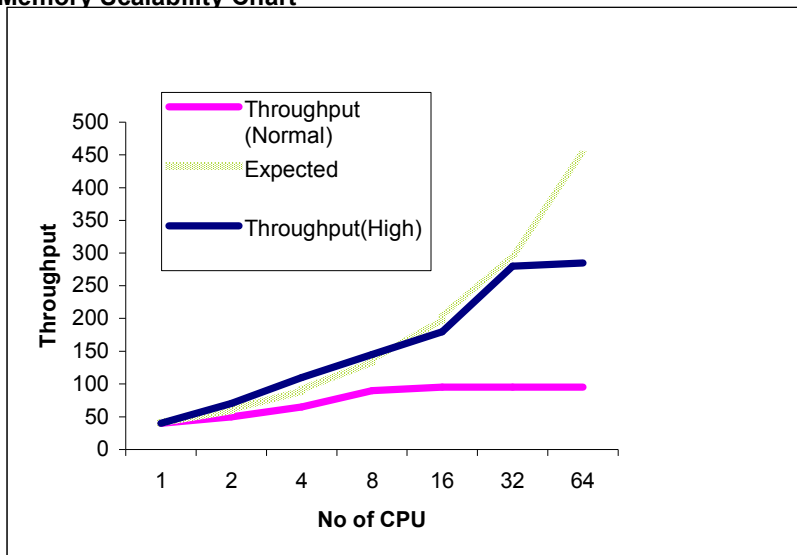
So the throughput and response time charts need to be prepared for the following test scenarios:

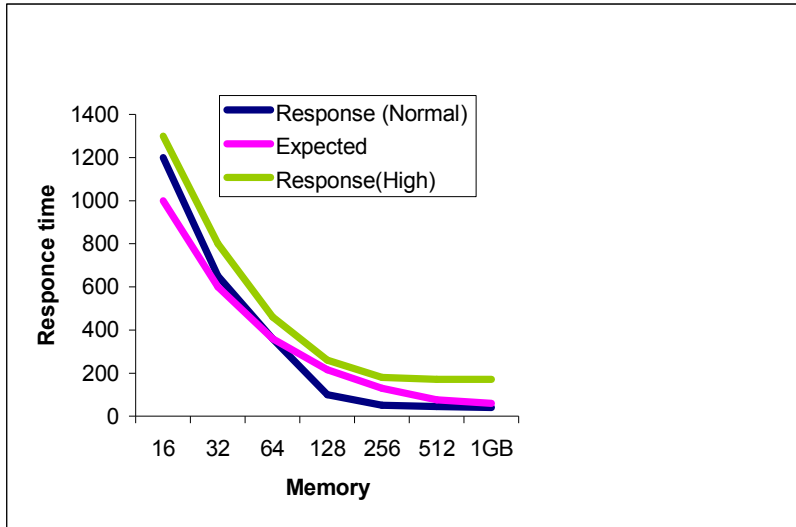
1. For minimum configuration as stated in user manual - Without tuning the product and OS parameters
2. For minimum required configuration as stated in user manual - After tuning the parameters for getting best performance data. Finding out the best set of values for tunable parameters could be multiple rounds of testing.
3. For typical configuration for majority of the customer base - Without tuning the product and OS parameters
4. For typical configuration for majority of the customer base - After tuning the product and OS parameters for best performance data
5. For special setups of the customers (with or without tuning parameters, as required by the customers)

The objective of repeating the test for these configurations is to reproduce the lab results in the customer place to solve the problem 3 stated above. This exercise, also help us to suggest the best set of tunable parameters and their values for the customer configuration without wasting much time at later stages.

Return on investment (ROI) is very important aspect that has to be considered before any upgrade. The above discussion on metrics will definitely give an idea what resources need upgrade but the question that remains in the customer mind is "what will I get when I upgrade?" If the product is scalable, for every mountain that grows, there will be proportional improvement in product performance. ROI calculation is not that easy unless we come out with some basic expectations in the form of assumptions and validate them during test cycles. The assumptions stated above, A11-A14 helps us in validating the expectations. The assumptions stated are only indicative numbers and needs to be worked out based on contexts and products. There are two load conditions that are talked about. Normal load condition is the load that is derived based on current needs of the customer and high load conditions are derived based on future needs.

CPU Scalability Chart
Memory Scalability Chart





In the CPU scalability chart it is easy to conclude that for normal load conditions there is a case exists for up to 16 CPU upgrade (Short term returns). To support high load condition in future, there is a business case to go up to 32 CPUs in the system (Medium and long term returns). In the memory scalability, 128MB of memory is sufficient for normal load condition and upgrade scope exist for high load condition only up to 256MB. As discussed earlier, tuning of parameters and making design level changes in product can improve the scalability.

This analysis also helps us to decide when the upgrade of resources needs to be planned. If the product is meeting the expectations with normal load conditions when upgrading the resources, that means that the ROI exists in short term. When resources are scaled up, if the product is not giving better performance with normal load condition but meets the expectation for high load condition, then it means Return on investment exist for medium and long term and in such cases upgrade of resources can be avoided for immediate term and planned for the future.

The "throughput" and "response time" charts have to be repeated for each scalable resource (CPU, memory, network and IO). This will help us to validate all the assumptions stated above (A11-A14) with respect to expectations that are set.

Summary

The charts used in previous sections were plotted from simulated data and real life data from testing could be much different. However it is important to collect these charts and do the analysis. The analysis can bring out some interesting findings on the design of the product and on the ROI, which are otherwise not detected and normally received as complaints from the customer. Hence the story of mountains not only gives us many problems, analysis & findings, but also gives us a great opportunity to improve the quality and coverage of performance and scalability testing for the products and systems we deal with everyday.