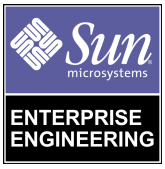
Scenario Planning - Part 1

By Adrian Cockcroft - Enterprise Engineering

Sun BluePrints™ OnLine - February 2000



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-4633-10
Revision 01, February 2000

Scenario Planning - Part 1

Abstract

You can use techniques to help predict latent demand during overload periods, which help define the minimum upgrade needed for saturated components. Using management inputs and modelling alternative scenarios, you can predict the effects of workload mix changes, marketing driven load changes, performance tuning, and hardware upgrades. In this part 1 you are shown how to simplify your model down to a single bottleneck. In part 2 a simple planning methodology based on a spreadsheet will be used to break down the problem and experiment with alternative future scenarios.

These articles will form part of a BluePrint book on High Growth Rate Capacity Planning that will be published later this year.

Introduction

Start by trending based on recent history, then add forward prediction using business inputs to predict what the future workload should look like.

Trending techniques use a mixture of step functions that track changes in your configuration and a technique sometimes known as Multivariate Adaptive Statistical Filtering (MASF). This process extracts cyclic variations from your data so that you can see the underlying trends. In this technique, analytical models are used to predict the behavior of complex workloads that consist of several classes of user and transaction types.

You should not attempt to build a detailed model of every part of your system. Doing so is too complex, too hard to calibrate, and would never be finished in time to be useful. Model the primary workloads; make sure you include the ones that make money for your business, like order entry. These need to be maximized.

A Recipe for Successful Scenario Planning

Successful planning must be based on firm foundations. It is quite common for planning to be attempted and abandoned as a technique, because the effort put in was misdirected and the return on investment (i.e. useful planning results obtained) was too low. This recipe provides a step by step guide to the process and gives examples of the kind of information that needs to be recorded in each step.

1. **Sketch a Simplified System Architecture**

   You probably have an overall system diagram showing every machine, every network, and all kinds of other details. You need to reduce the complexity by looking for the primary flows through the system. These are usually the flows that are either subject to service level agreements, or make money for the business. The initial sketch should be based on the physical systems in place, as that is the simplest place to start. Where multiple systems are employed to do the same job, show them as a simple replicated set even if they are not exactly the same. Be ruthless about paring down the complexity to a minimum to start with. It is better to go too far in this direction and add systems back in later than to start off with too complex an architecture and get bogged down trying to model it.

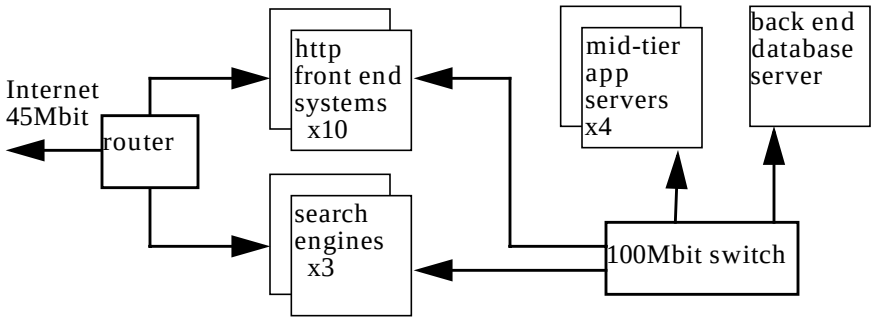


FIGURE 1 Example Physical Architecture Sketch

The physical sketch usually needs to be supplemented by a dataflow sketch that shows the way that the main classes of users and applications are connected together.

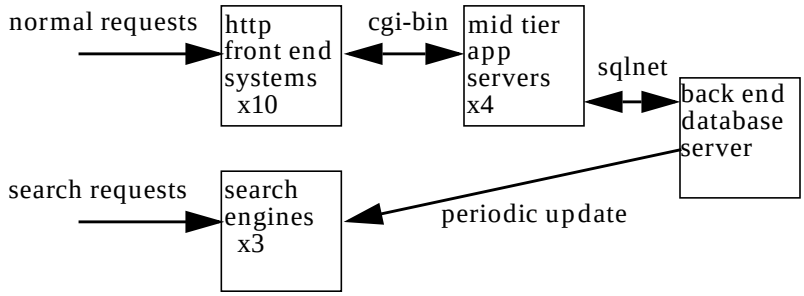


FIGURE 2 Example Dataflow Architecture Sketch

These sketches provide a framework that information can be attached to, and later steps in this recipe will refer to the systems and flows.

2. Determine Primary Bottlenecks

The model should concentrate on the primary bottlenecks. It may be sufficient to look at just the CPU load on a central database server, or you may also need to watch network capacity or a wider range of systems depending upon your situation. The bottlenecks will change over time so they must be listed explicitly each time a scenario is modelled.

For each of the systems and network interfaces that are identified in the physical sketch, an average utilization during the peak usage period needs to be recorded. The utilization of each network interface needs to be recorded as a percentage based on Mbits/second. For shared network components the combined utilization of the network backbone or switch needs to be measured. For each system, the overall CPU

utilization and the utilization of the busiest disk on that system is needed. The highest utilizations you find are the components most likely to be the primary bottlenecks.

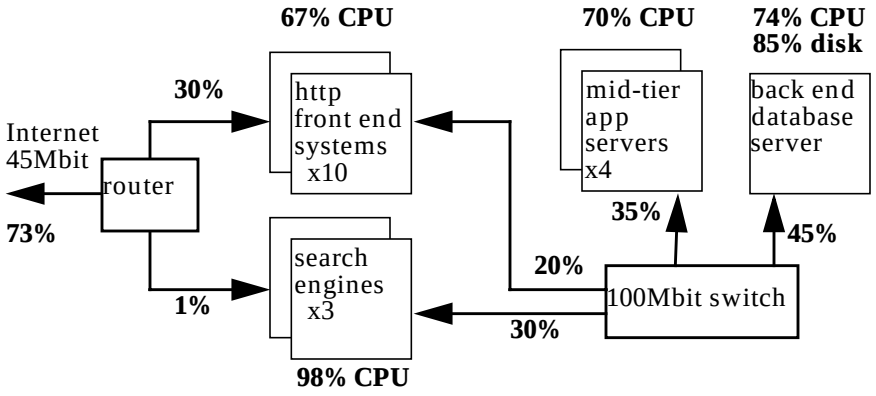


FIGURE 3 Example Physical Architecture Sketch with Utilizations

The example shown in FIGURE 3 indicates a system where the primary bottlenecks appear to be the CPU and disk on the back end database, and the CPU on the search engines. Disk utilizations were found to be very low on the other systems, which operate mainly from memory caches.

3. Measure Service Levels and Workload Intensities

If you have a service level agreement, or some estimates for response times and throughput, then the data flow architecture sketch should be annotated with both the latencies and the frequency of transactions wherever this can be determined or estimated.

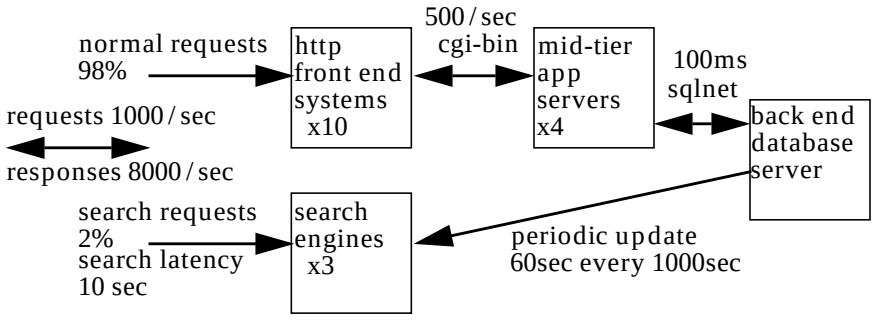


FIGURE 4 Dataflow Architecture with Latencies and Intensities

4. **Determine Operational Modes**

   Performance modelling assumes that the workload is essentially constant. Therefore, if the system has periodic changes in behavior, it is best to pick out the main modes and measure and model them separately. One of the main modes occur when backups are taking place. In many cases, backups must take place online during quiet periods because, when supporting Internet based users around the world, there is no convenient time for offline backups. Modelling backup behavior is useful as fast growing environments will find that the amount of data to backup increases rapidly. As a result, the backup duration increases and often overspills from the quiet period, impacting performance. The disk subsystem will probably be at its busiest during the backup.

   In the example shown in the figures, there is a periodic search update, and the load is different while the update is occurring. It will be useful to model behavior in the normal state, and also during an update. It seems possible that the back end server's network interface will be a bottleneck during the update. The utilization levels will be different in each mode so they will need to be recorded separately.

5. **Choose One Bottleneck**

   It is quite simple to model a single function on a single system, and extremely complicated to model the entire network of systems. However, the performance of a network of systems is limited by its bottleneck, so you get useful results very simply and quickly by focussing on one core bottleneck and assuming that bottlenecks elsewhere in the system can be removed fairly easily.

   One way of looking at this issue is that the biggest and most expensive system that you have, should be run at the highest utilization so that you are not wasting expensive resources by leaving them idle. This system then tends to be the bottleneck, and is the best candidate for modelling. You should find it more cost effective to upgrade surrounding systems so that they are no longer bottlenecks.

   In the example I have been using, the back end database server is a single system (or perhaps a highly available pair of systems) and it is common for this system to be both the most expensive resource, and also the primary bottleneck. It is much easier to replicate smaller systems to take care of front end services than to decompose a single database instance into multiple systems.

6. **Choose Service and Utilization Indicators**

   The initial model is going to be a crude and over-simplified basis for planning future scenarios. The hardest thing to learn for successful modelling is to let go of the details. You can build a very useful model with incomplete and inaccurate data as long as you are smart in what you include and what you leave out. For the back end server, a mixture of transactions are taken from the application servers. However, as long as that mixture remains fairly constant, you can average all of them together into a generic transaction rate. If the back end server load level fluctuates a lot due to several competing applications, you need to perform a workload breakdown to obtain just the CPU and disk utilization that is involved in servicing the incoming transactions. If this is the primary load on the system, then you can probably get

away with using the overall CPU utilization. The assumption you need to make is that there is a direct linear relationship between the total number of transactions processed and the CPU utilization. The average CPU time per transaction will change when the system configuration or application changes, but it should stay constant for long enough to be a useful basis for a simple growth model.

7. **Pick a large Timescale**

   For scenario planning we are interested in what will happen over the coming days months and years, not the next few seconds - this is why we can ignore so much fine detail. Also to quote the mainframe capacity planning guru Pat Artis:

   > *To predict the future, you must know the past.*

   You need to collect historical data that spans a time period similar to the period you are trying to predict. So a week's collected data will give you a good prediction of next week. To predict the next year, you will need collected data for the whole of the previous year.

8. **Work Through The Planning Process**

   The rest of the scenario modelling and planning process will be illustrated using a simple parameterized spreadsheet in next months article.

9. **Compare The Prediction With Reality**

   Once you have predicted the transaction rate and system utilization for the coming months, you need to measure and compare with reality. This way you learn which estimates in the model were optimistic or pessimistic and you learn how to build better models.

10. **Recalculate The Predictions Regularly**

    After each time period you predict (e.g. monthly) you have a new baseline of real data and the model needs to be recalculated.

Summary

The art of modelling computer system performance is to simplify the model without losing its usefulness.

Next month, a simple spreadsheet based scenario planning model is explained. It should be very quick and easy to implement, and if nothing else, help you think more clearly about the assumptions you are making over the coming months and years.

Author's Bio: Adrian Cockcroft

The author of Sun Performance And Tuning, Adrian is an accomplished performance specialist for Sun Microsystems and recognized worldwide as an expert on the subject.