Modeling Is Doable and Useful:

It's not as scary or difficult as you might believe

By: Bob Wescott

Summary

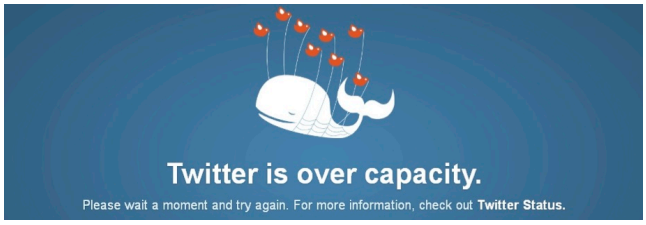
Most people think computer performance modeling is best left to an elite super-smart few. That is nonsense. There are two basic types of modeling that regular humans can use in performance work. There are also a few basic modeling truths that underlie all models. Both are covered here.

This is an excerpt (with modifications) from: ***The Every Computer Performance Book*** a short, practical, and occasionally funny book I wrote on doing computer performance work.

Two Kinds of Models

There are as many kinds of models as there are PhD candidates to dream them up. For most people, they feature complex math and nearly impenetrable terminology. I'll bet many of them are wonderfully useful. I encourage you, if you so desire, to explore them in other books. In this book we'll look at just two kinds of models: Capacity and Simulation.

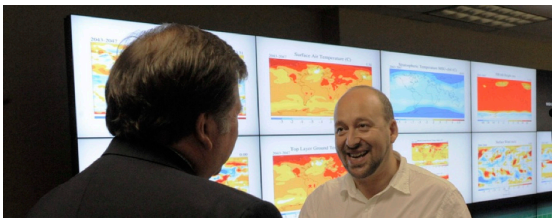
Capacity Models



Capacity models are just regular old capacity planning with a bit of a twist. Instead of "find a utilization and scale it," now there is more work to do. In a capacity model you are redirecting the flow of work, adding new work, adjusting the mix of current transactions, or incorporating new hardware. There are more things to count and account for.

To do a capacity model you have to understand what you've got, and then figure out how to adjust the numbers to compensate for the changes you are modeling. It requires performance monitoring, capacity planning skills, simple math and an eye for detail. Capacity models can do a lot, but they can't predict the response time under load. If you need to know that, then you need a simulation model.

Simulation Models



Simulation models are a funny combination of an accounting program, a random number generator, and a time machine. They simulate work arriving randomly at a pace you select and then simulate the flow of work though a simulated computing world by accounting for costs and delays at every step. They can run faster, or slower, than real time. They can skip ahead through time when they've got nothing to simulate at the moment. They give you throughput, utilization and response

time information for any computing world that you can dream up. The only problem is that they sound scary.

I used to believe that simulation modeling could only be done by super-smart NASA engineers and was only reasonable to do in situations where things had to work the first time or people would be killed and millions of dollars worth of hardware would be destroyed. I used to believe that simulation modeling was incredibly expensive, hard, and time consuming. I was wrong.

I've found modeling to be a useful, and important tool in my toolbox. I taught modeling concepts and a PC-based simulation modeling tool to rooms full of regular people who worked for regular companies, doing the normal work of maintaining and improving commercial systems. From that I learned modeling is doable. The stories I heard from my students about the models their companies relied on taught me that modeling is useful.

Now lets looks at some surprising truths about modeling.

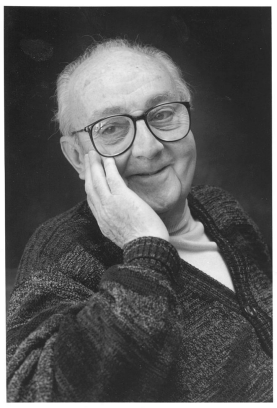
Modeling Is Necessary

There are two kinds of important performance problems you can't solve without modeling. You can't do performance monitoring, capacity planning, or load testing on un-built systems, as there is nothing to test. You can't use simple capacity planning or load testing to predict future performance on systems that are about to undergo radical transformations.

In both cases there is a bit of a chicken-or-egg problem as the company wants to know the cost of the hardware for the un-built or radically transformed computing world before it is built, but until you build/transform it, you don't have all the data you need to make those projections. This is solvable.

All Models Are Wrong...

George Box once artfully said: "All Models are wrong, some models are useful." So, please take a moment and get over the fact that your model won't generate a perfect result.



Nobody models to a high degree of accuracy because to get that you have to build wildly complex models that model every little thing. You have to put so much time into the model that the business is out of business before the model sees its first run. The 80:20 rule of chapter three applies here. A simple model can give you a ballpark answer. That is often more than good enough to green light a project or size a hardware request.

...Some Models Are Useful

Imagine an inaccurate model where you are guessing at many of the input parameters and unsure about the transaction mix or peak demand. You run the model and, even with the most optimistic assumptions, it forecasts that you'll have to buy somewhere between two and five new servers. If the budget is closed for the rest of the year, then this useful model just saved you a lot of time on that sure-to-fail idea. The thing that makes a model useful is your confidence that it is accurate enough to answer your question.

"If you have to model, build the least accurate model that will do the job."
– Bob's Ninth Rule of Performance Work

Modeling Can Be Done At Any Time

All models can be built at different stages of a project's lifecycle: design, testing, and production. At each stage there is data available to build a model.

In the design stage, models are built on educated guesses, the results of quick tests, business plans, and other less than concrete data. They answer big scale questions like "How many servers will we need?" and are not all that precise.

In the testing stage, models can be built with better data as the design is fairly fixed and there is some running software to meter. Here you can ask small scale questions such as "Can both these processes run on the same server?" as well as big scale questions like "Will this configuration handle the peak?"

In the production stage the entire application and computing world can be metered and tested against. Here, with enough work, you can build a model to answer almost any question.

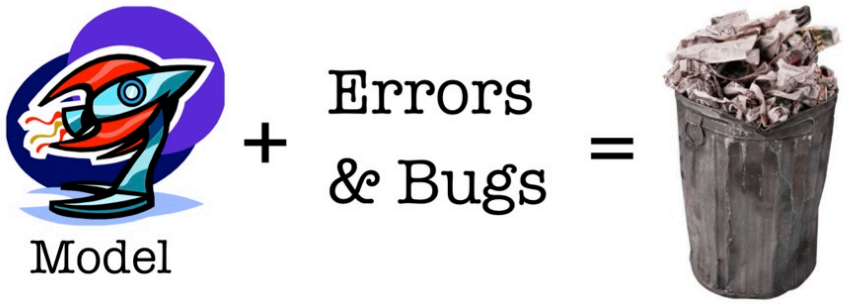
Models Can Be Built To Different "Resolutions"

At some point in every model you treat some part of your computing world like a black box. Data goes in, data comes out, and you don't care about the exact inner workings. Depending on the question you want answered, the model can treat any part of your transaction path as that mysterious black box. It could be as small as a single communications link or as large as the entire datacenter.

The higher the resolution of the model, the more costly and time consuming it is to build. Do not confuse high resolution with high accuracy, as they are not the same. A low-resolution model can give you a spot-on accurate answer. For example, if you are modeling your datacenter's Internet connection, you don't care what happens inside the datacenter, you care about how the bandwidth requirements will change as the transaction mix changes.

Models Need Verification and Validation

A model can be a complex thing. There are plenty of opportunities for your errors, and any bugs in the software, to render your models output useless.



How do you learn to trust the model? There are two key things you must do: verification and validation.

Verification asks: "Did we build the model right?"

Here is where you check:

- Does the model run without errors or warnings?
- Does running a small number of transactions burn exactly the expected modeled resources in the expected modeled places? Pump exactly two transactions though your model and hand check every resource count and utilization number you get to see it is exactly double the resource utilization you specified in your resource consumption data.
- Is the mix of transaction types correct for our workload? Run the model at a low transaction rate, so there are no appreciable queuing delays or resource constraints. Then hand check to see if the model is reporting the expected ratio of X, to Y, to Z transactions.
- Does the output feel right, make sense, and leave you feeling comfortable? Run the model and check to see if all the resources that you are modeling are showing simulated utilization. Does the average response time for transactions go up as you push more simulated work through the model?

All of this detailed work allows you to be more confident that you built the model right, it is working as expected, and you understand the results the model is generating.

Validation asks: "Did we build the right model?"

- If we are modeling an existing system, do the modeled results match the metered system? There is nothing like having the model match the live system to inspire confidence in the model. Remember, it will never match exactly. All it has to be is close enough to be a good predictor of future performance.
- Does the model's output make sense to key experts? Humans have an amazing ability to notice when things look wrong. The more people who look at your results and say: "That can't be right," the more you should double-check your model, your assumptions, and your input data.
- Can this model answer the business question that started this whole effort?

To Boldly go...

Capacity models are flexible and somewhat easier to do than simulation models. However, they assume a steady state environment and cannot typically deal with algorithms that dynamically tune performance as the load changes. Like any tool, they have limits.

Simulation models can do it all and can give projected response times for complex transaction paths. Regular people can build them, in a reasonable amount of time, with the appropriate tools.

Choose the modeling technique based on the question you are trying to answer.