# The Four Numbers of Capacity Planning:

Capacity planning for any given resource boils down to finding four numbers and doing a bit of multiplication.

By: Bob Wescott

## Summary

Capacity planning for any computing resource is essentially multiplying three numbers together and then comparing that number with the max utilization for a give resource. Here we will examine those four numbers and how you find them

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.

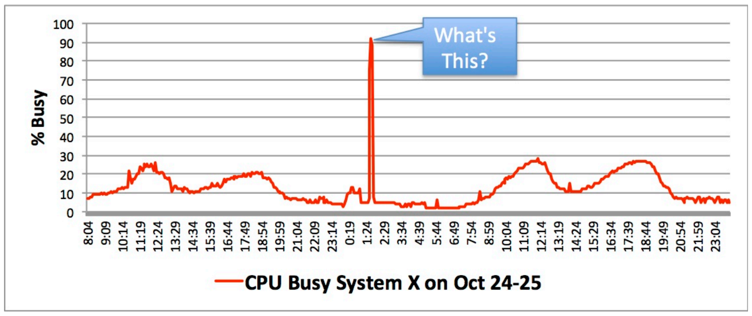
## The Four Numbers of Capacity Planning

Capacity planning for any computing resource is essentially multiplying three numbers together (Utilization * Scaling factor * Safety Margin) and then comparing the result with your maximum utilization for the given resource. If the calculated utilization is greater than the maximum utilization, then you've found a future bottleneck. The number of times you will repeat this calculation and comparison depends on the size and complexity of your computing environment. Do this for everything that gives you a utilization, like CPUs, disks, etc. There is a different procedure for handling the things with limits, like free disk space, application limits, etc., that we'll explain in the next section.

## Utilization

Step one of capacity planning is to find a time that you want to base your capacity plan on. A time when the users are sending your computing world a moderate, stable load and the users are happy with your overall response time and throughput.

Your daily peak load is often a good place to start. Typically that time is decided by watching the system behavior over several days, or weeks, and then selecting some reasonably busy time. Usually, you start by looking at a key system's CPU consumption as it is the one resource that all transactions consume. Let's look at some data and pick a number.



In the above graph you see the CPU busy data from System X sampled every 5 minutes for most of Oct 24th and 25th.

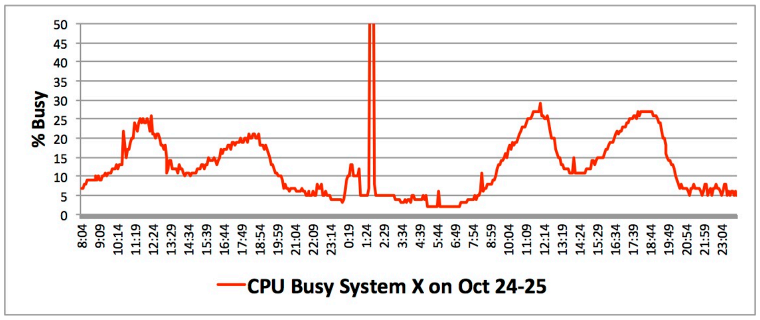
All transactions pass through this computer. The first thing that jumps out at you is the sharp 90%+ busy peak in the middle of the night. A couple of good questions to ask are:

- Is that normal processing or did something go haywire?
- If normal, how often does this happen?
- Does this ever happen during the daily peak user load?
- Are we going to tolerate the response time increases this late-night job creates for about 30 minutes, or do we need to capacity plan for this too?

If this is a background job that runs in the middle of the night when few care about response time, then you can ignore it. If the boss cares about response time 24x7, then you have to plan for this, too.

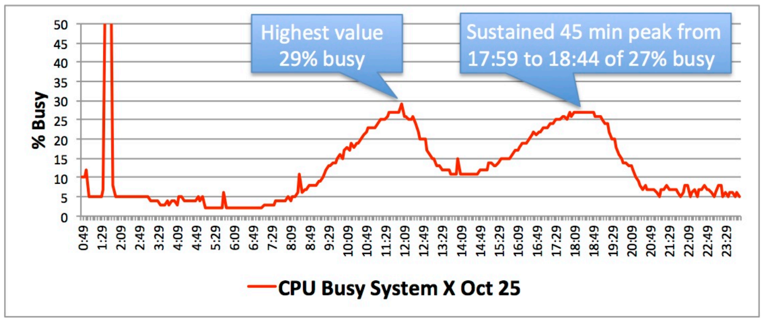
If this peak is the result of some event that suddenly and rapidly dumps work into the system (e.g. a computer coming back online after a communications disruption) then you need to plan for this. You might still go ahead and build your plan with the normal transaction load, but you'll need to mention the possibility of this spike happening at the worst possible moment in your written capacity plan.

For the moment, let's ignore the early morning peak, and change the scale on the Y-axis so we can see the data we really care about more clearly.



Now that we've trimmed the Y-axis to a max of 50% busy, the next question is which day's peak to use. The daily peaks on the 25th are a bit higher and somewhat more consistent than the daily peaks on the 24th, so let's focus on those peaks and redraw the chart to only show that one day.

Below we can see a sustained 45 minute long peak of 27% busy and a maximum recorded value of 29% busy.



The period where we see the sustained peak, holding steady through multiple samples, is a good place to start. It’s good to have multiple adjacent samples in agreement because it gives you confidence that the overall system was at a steady state.

The highest value recorded that day was 29% busy, but you’ll notice it is not a sustained peak. Someone may look at that chart and say you have to capacity plan based on the busiest moment. Here it makes little difference as 27% and 29% are not that far apart. However, sometimes there are bigger differences, and sometimes the person insisting on using the highest values is your boss. If your boss is adamant, then go with it. Why fight over a small difference that you can easily correct for in the scaling factor, the safety margin or max utilization values? Pick your battles.

So now you’ve got two things: a time range for a steady state peak and a value for CPU busy (27%) on this system.

Before you declare victory and go out for a long lunch, look at the metering data from the other systems and devices in your computing world and see if they are showing a peak at about this time with about the same shape. It is entirely possible that the other systems in your computing world did not see this sustained 20 minute peak because it was caused by a little program someone ran unbeknownst to you on this system.

If you want to build a capacity plan based on this observed peak, it should also show up throughout your computing world during the same time and with the same magnitude.

***“The meters should make sense to you at all times, not just when it is convenient.”***
**–** Bob’s Sixth Rule of Performance Work

## The Scaling Factor

The scaling factor is a number that represents how much busier the future is anticipated to be when compared with the time you sampled. Sometimes it is based in fact (e.g. you just bought a competitor and you know how much business they will bring), and sometimes it is a guess pulled right out of thin air. To get the scaling factor:

2X

- Collect performance data from your computing world
- Pick a time when the load is moderate and level
- Show your graphs to the key players
- Have them give you the scaling factor

The scaling factor they pick will typically be an SRN (Suspiciously Round Number) and, in my experience, it is never exactly right, but it is often close. Humans are an amazing species. If you are tasked with making this guess, I recommend using the Delphi Technique discussed in chapter three as that can be quite helpful in getting an unbiased group opinion.

Capacity planning assumes that the resource demands of most applications scale linearly because, within normal boundaries, they do. To be precise, 99% of the computer programs I've seen increase their resource consumption in direct proportion to their throughput. If you push twice the work though an application, it will consume close to twice the computing resources such as: CPU cycles, disk IO's, packets sent/received. The exceptions to this rule are:

- When the system or application is starting up. At startup, files have to be opened, programs paged in, caches filled, and initializations performed. Unless you are studying ways to recover faster after a restart/crash, ignore the meters during this time.
- When the application is hopelessly bottlenecked. When overwhelmed algorithms designed to manage about ten things in the queue suddenly have a billion things in queue, they don't work well.
- When errors are happening. They cause retries and retransmissions, poorly tested and inefficient error handling routines to run, processes to crash and restart, and general suffering.

When capacity planning, none of the above apply. You don't capacity plan a system reboot or an application restart, you don't plan to have the peak load experience a bottleneck, and you can't plan for all possible errors. However, you can create a capacity plan or model that takes into account the load being shifted to the remaining systems when a system fails. We'll set aside those unfortunate possibilities until the end of this chapter and the chapter on modeling.

Convert any scaling factor they give you into a multiple of one. So, for example, "Plan for a 30% increase" becomes 1.3, "Twice the load you metered" becomes 2.0.

## The Safety Margin

Every company has a level of corporate courage and a certain aversion to pain. These attributes are shaped by their people, their culture, how much money they have to spend, and by their recent disasters. It is the rare company that likes to hang by a fingernail on the edge of disaster. Only a fool will insist upon planning for a peak load where every resource is at its maximum utilization as there is no margin for error. Most people feel better, and make better decisions, when they have a margin of safety. Besides, capacity planning is not a perfect science:

- The workload mix and intensity are always changing and are somewhat difficult to precisely predict. They are often influenced by such unpredictable forces as the weather, the economy, and your competitors.
- Strange things happen in big companies, and sometimes the demands of other parts of the business on shared resources can change without warning and not in your favor.
- Software upgrades, network changes, equipment swap-outs, and configuration adjustments that happen between the plan and the actual peak can alter performance.
- Capacity planning looks only at the question of "enough." It can give you no hint about the response time changes you will see at the projected peak load. The closer you are to the limit for a given resource, the uglier the response times consequences will be if too much work shows up.

Do not let the person that gave you the scaling factor tell you that it includes the safety margin. You need to keep these two values separate. The scaling factor is your estimate of the future load; the safety margin is how sure you are about that estimate.

The safety margin inflates your projected utilizations by the percentage that you choose. Most companies I've worked with have chosen a safety margin value between 10% and 50%, with the most common values in the 20% range. When using this in capacity planning, convert it into a multiple of one. A 20% safety margin becomes 1.2.

## Max Utilization

For resources that provide a service there is a utilization beyond which the delays caused by queuing effects become too painful. As a general rule: the slower the resource is, the lower this pain threshold should be. Why? It's all about wait time and service time.

The busier the resource is the more likely you'll have to wait in a queue to be serviced and the longer that queue will be. Each thing ahead of you in the queue will have its full measure of service time before you get to run.

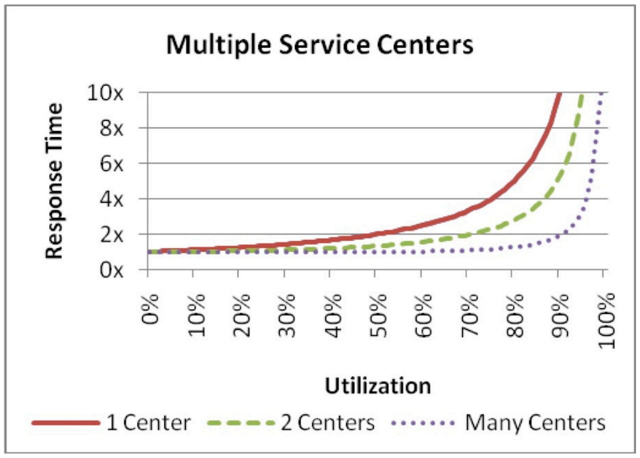
### How Busy Is Too Busy?

As we learned in chapter 3, any device that is 50% busy will have an average response time of twice the service time as there is an average of one job to be processed before you get your turn to use the device. The slower the device is, relative to the other things in your computing world, the more painful this math becomes. Think about it this way... Which would you rather see doubled: the cost of your next haircut or the cost of your next mortgage payment?

In the early 21st century spinning magnetic disks are the slowest part of any computer system by several orders of magnitude and thus, the rule of thumb was to keep the utilization of a disk below 50%. If we switch to solid state disks, their vastly lower service time, which translates into a lower wait time and thus a lower response time for a similarly busy device, would justify picking a higher max utilization number.

When picking the max utilization for a device, there is always the temptation to shove money into the discussion with comments like: Those disks were very expensive, and now you are telling me I can only use X% of their capacity. Your reply should point out, in a gentle way, that the cost of running some device at 100% busy is remarkably bad response times for the customers – see queuing theory.

Device exclusivity plays a big part in the number you pick for max utilization as well. When a process needs CPU, any one of the multiple CPU's in the system will do. If you can go to many places to get serviced then the odds are good that one of them will be free and that will hold down response time as the utilization climbs toward 100%.



On the other hand, if the data you need is on one device (e.g. reading a specific record from disk), then you can only go to that device, and the response time curve turns ugly at a much lower average utilization.

To pick a max utilization number for a given device, start by doing your homework; read the manuals, search the web, and then talk to the vendor. If you are laughing at my suggestion to start by reading rather than calling, remember that when you call, you will end up talking to either: people who don't know, so they bluff and bluster, or people who do know. If you start as an informed person, you can quickly identify and disregard the people who don't know. If you luck out and get to talk to someone who does know, they are more likely to give you a rich and complete answer because it is clear that you've done your homework.

You can also experimentally select a max utilization value of some resource through testing by adding load to the resource until you see the response time start to grow unacceptably. The number you get should be between 50% (for really slow resources) and 80-90% for the speediest resources where the incoming work also has many service centers to choose from.

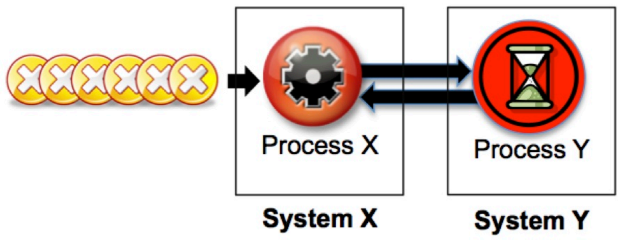
It is not that hard to write a program that keeps some device busy, but busy is not the same thing as backed-up with work. The real response time pain of a busy device comes from the line of transactions waiting to run before you do.

If your busy program is just one process that submits a transaction and then waits for it to complete, you won't create a long line of waiting transactions. Your response time will equal your service time and you won't feel the pain of a long queue of waiting transactions.

## How Busy Is Too Busy For A Process

A process executes code until it has to wait for something like a reply from another process, a lock, or an IO to complete. CPU consumption for a process does not tell the whole story of how busy it is. Here are some guidelines to help you figure that out:

- With rare exception, a process that is 100% CPU busy (burns one second of CPU per second) can't do anything more for you, and has most likely hit a bug, and is doing nothing useful for you.
- A common performance analysis mistake is to assume that a process consuming only a small amount of CPU couldn't possibly be the bottleneck. Processes wait for things, and when they wait, they consume no CPU.
- If the software has remained unchanged since the last peak load, you can take the per process CPU utilization data from that peak and feel pretty sure that process can consume at least that much CPU at the next peak.
- Some applications have a dynamically tunable number of processes doing a given task. Often they have the same name (e.g. FE01, FE02, FE03...) with a number appended to it. If the load is spread evenly, you only need to study one example of each group. You can create an artificial peak load for those processes by reducing their number and letting the incoming workload overwhelm them for a minute, or two, while you gather some metering data. Then start additional processes to return things to normal. Note: This is not a perfect test, but it is better than nothing.
- If you can see the queue of incoming requests for a given process and that queue is never empty, it's clear that the process is working about as fast as it can, regardless of how little CPU it consumes. However, that process may not be the root cause of the bottleneck. In the example below, for every transaction Process X works on, it has to ask the somewhat slower Process Y for a reply before proceeding. Process Y is on a different machine, and that is why you can see Process X bottleneck and backup even though there are plenty of resources on System X.



- Use your common sense. If you are planning for a peak that is ten times the load you are currently measuring, and the process in question is already consuming 0.2 seconds of CPU/second, then clearly this process will be using 10 * 0.2 = 2 seconds of CPU/second, which is impossible.
- Some processes are involved in the main transaction path, and some come into play less often. Focus your efforts on the main transaction path processes. How do you

identify them? The processes that consume most of the resources and whose consumption rises and falls as the load does are the processes you want to study. They are typically a small subset of all the processes running.
- If none of the above suggestions work for you, then you either have do some load testing or make a good faith estimate. When estimating, include others in the process and use the Delphi Technique as described in chapter 3.

When you have a max utilization (expressed as a number from zero to one) for all your resources then you are ready to do some capacity planning.

## Doing The Math of Capacity Planning

Now you are ready to scale up to the projected peak load. The formula is straight forward:

Utilization * Scaling Factor * Safety Margin = Projected Peak

Imagine you have a resource that is 40% busy (utilization is 0.4), and you need to plan for a peak load that is 50% larger than what you are seeing now (scaling factor is 1.5), and you are reasonably sure of your projected peak load within about 10% (safety margin is 1.1).

Doing the math you get 0.4 * 1.5 * 1.1 = 0.66 and now you know this resource will be 66% busy at the projected peak load. You've determined the max utilization for this resource is 75%, and so you feel reasonably sure that this resource will not be a bottleneck at your projected peak. Now do that calculation for all your other resources.