

"How To" Guides for Computers

"How To" Guides for doing things with computers

How To Guides for Databases

How To Guides for Operating Systems

"How To" Guides for Computer Systems

Horizontal scalability (more processors) vs. vertical scalability (faster processors)

Scalability

There are a number of advantages of vertical scalability (faster processors) over horizontal scalability (more processors):

For a lot of software, license costs and maintenance fees depend on the number of processors where the software is running, and, in some cases, the size of the processors. In the cases where the size of the processors is a factor, the cost for a 2x processor is seldom twice that of a 1x processor. Where the cost does not depend on the processor size, it's a no-brainer; the cost increases when adding more processors, but not for faster ones.

Upgrading to faster processors provides as much improvement in throughput (see calculations below) as adding enough additional processors to reach the same total MIPs. Even if the original processors must be removed and the one-time purchase price of the new ones is more than adding more processors of the same speed, the reduction in ongoing software costs can easily offset that expense.

Faster processors can improve response time and reduce transaction queue wait times, while additional processors can only do so if transactions can be split across processors so that they can run in parallel. As a result, scaling vertically is more likely to provide noticeable improvements, such as making web pages load faster.

Throughput Calculations

My first college computer course was long enough ago that some of the exercises included the calculations for throughput and disk seek times. With RAID and caching nowadays, disk seek times aren't very relevant for end consumers, but the throughput calculations taught us something about CPUs that was quite counterintuitive, that is:

All else being equal, the fewer processors you have, the better. Put another way, vertical scalability is better than horizontal scalability.

Example

Consider this example. Say you are sizing a system for an average load of three transactions-per-second and a maximum of five transactions-per-second at 80% utilization and that typical transactions take around 500 megacycles. You have a choice between two 1 GHz processors at \$100 each or one 2 GHz processor at \$200 (\$100 per GHz in both cases). If the transactions arrive at regular intervals, the response time for the two 1 GHz processors will be .5 second, and the response time for the 2 GHz processor will be .25 second. Now consider the case where, during a given two-second interval, the six transactions arrive at two-tenths of a second intervals. What are the response times? If needed, draw three two-second

long timelines, one above another, and start the transactions at the .1, .3, .5, .7, .9 and 1.1 second marks.

With two 1 GHz processors, the response time of the first two 500 megacycle transactions, which are running on separate processors, will be .5 second. The next four transactions will have a .1 second wait time and .5 second processing time, for a total of .6 second response. Therefore, the average response time will be .57 second.

With one 2 GHz processor, the six transactions will have wait times of 0, .05, .1, .15, .2 and .25 and reponse times of .25, .3, .35, .4, .45 and .5 respectively for an average response time of .37 second. So even under a varying load, the single-processor configuration is able to maintain much faster response times.

Throughput vs. Actual Bottleneck

So the moral is that, scaling horizontally by adding more processors to your system configuration or more computers to your network will only improve throughput. If the bottleneck in the system isn't throughput, you are likely to see no improvement at all.

Response Time / Page Load Time

If instead you need to improve transaction response time or web page delivery times, you need to scale the system vertically, by upgrading the system with larger, faster processors.

There's just one caveat. In systems with fairly rigid process priorities, there may be an advantage to having two processors in a single box, so that a second processor is available if one processor is tied up with a high priority task. Having a second processor available may be useful if it becomes necessary to terminate a runaway high priority task and someone needs to get through the lower priority login process because no high priority users are already logged in. So a good rule of thumb is:

Buy a single, dual-processor system with the fastest processors available, or at least fast enough to handle the expected load. Upgrade the speed of the processors as needed and available. Consider adding additional processors or servers only if you are absolutely sure that your bottleneck is with throughput, not I/O or some other resource, and either no faster processors are currently available or else there is no advantage to improving response time or speed of web page delivery.

Index