Jon's Performance Musings: Performance Is Dead?

Jon E. Schmidt
Transaction Design, Inc.
San Rafael, CA, 94901, USA
1.415.256.8369
inform@banbottlenecks.com

Jon is the founder of Transaction Design, Inc. (TDI), a consulting firm located in the San Francisco Area which specializes in performance and capacity studies with clients worldwide. He is the creator of the Ban Bottlenecks® service and has an extensive background in the implementation, testing, and tuning of high-availability systems.

A recent article declared that we have "pretty much left behind the primary issue ... performance. Our systems have become blazingly fast..." That's really good to know. All of us concerned with performance can either retire, take the week off, or spend all our time surfing. I'm sure our vendors would be happy to stop doing R&D to make faster chips, disks, and interconnects. Whew! Nirvana has been reached.

Or Maybe Not

Or maybe not. Of course, if you have infinitely deep pockets, you can always over-buy. Money will solve any performance problem, if you know where to apply said cash. But you know, times aren't as flush as they once were. IT budgets everywhere are constrained. Dollars must be applied judiciously to solve or prevent IT problems.

The Basic IT Laws For Performance Analysts

Everyone knows Moore's Law. There are other laws and corollaries which provide a more realistic view of what we performance types have to deal with. Here's what quick search of Wikipedia[1] comes up with:

- The Great Moore's Law Compensator: TGMLC, generally referred to as bloat, is the principle that successive generations of computer software acquire enough bloat to offset the performance gains predicted by Moore's Law.
- Hofstadter's Law: It always takes longer than you expect, even when you take into account Hofstadter's Law.
- Parkinson's Law: Work expands so as to fill the time available for its completion.
- Parkinson's Law Corollary: Data expands to fill the space available for storage.
- Parkinson's Second Law: Expenditures rise to meet income.
- Wirth's Law: Software is getting slower more rapidly than hardware becomes faster.
- Gate's Law (or Page's Law): The speed of commercial software generally slows by fifty percent every 18 months thereby negating all the benefits of Moore's Law.
- Thatcher's Law: The unexpected happens. You had better prepare for it.
- The Jevons Paradox: Technological progress that increases the efficiency with which a resource is used tends to increase (rather than decrease) the rate of consumption of that resource.
- Amara's Law: We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run.
- Finagle's Law of Dynamic Negatives: If something can go wrong, it will go wrong, and at the worst possible moment.

Laws, Not Jokes

The rules above are amusing, I agree. However, the facts are there to confirm the validity of these rules. We all know them: Software developers now write applications using languages that are not compiled. Entire application systems are written using scripted languages. Java and scripted languages execute much more slowly than compiled languages. [2]Databases are used instead of flat files.

The industry has made a choice that hardware is inexpensive, and that software development is very expensive. So we see applications being built that require dozens of frames, hundreds of processor cores, terabytes of main memory, and multiple terabytes of disk to implement.

On the other hand, it probably wouldn't be possible to create some modern applications without the powerful tools found in the "Web 2.0" environment.

Virtual Rant

The loss of control implicit in the new software paradigm drives me up the wall. In the old days, we performance types would obsess over a new operating system release, testing it to see if it added to the inherent overhead. We would have serious discussions with the application developers if their latest update used more resource per transaction. We'd compare clock speeds and channel bandwidths.

Now we despair over the loss of control and seeming indifference to overhead. For example, I certainly understand the need for virtualization. However, virtualization always comes at a price. A hypervisor will always add to the overhead. And when I hear of an environment where a system running Windows is the host OS, running a hypervisor, running daughter OS's I just have to shake my head.

[1] http://en.wikipedia.org/wiki/List_of_eponymous_laws
[2] http://verify.stanford.edu/uli/java_cpp.html

When my application's performance is totally dependent on the implementation of a scripted language by a vendor or an open-source community and I can't see under the covers to understand locks and resource consumption I worry. I worry about "gotchas" like Java's "stop-the-world" garbage collection and how it will affect my 100 TPS transaction stream.

When the disks for my mission-critical OLTP system are sharing the communications fabric, processors, cache, and perhaps drives of a SAN with major batch application systems, I worry.

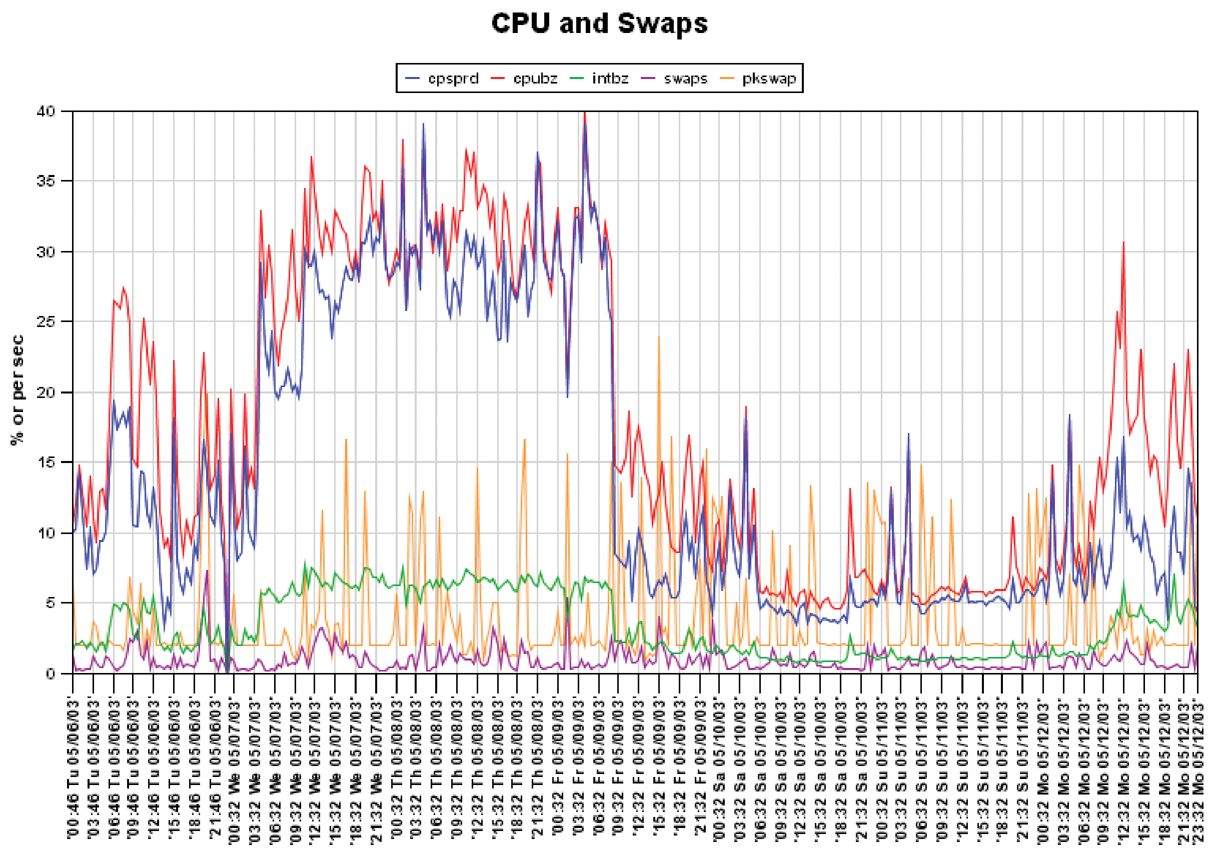
Holistic Approach

In the new world of software and performance analysis we must take a new approach. We performance types have to become experts at "black box" analysis. We have to look at the entire environment we are given, and take a step back from the nanosecond-style analysis that we are used to doing. There are simply too many factors outside our control. Our job becomes one where we must measure a complex environment, try to correlate usage to the business demand, and project what's going to happen when that demand increases. We also have to understand how outside-the-box competing systems and environments affect our critical system.

And, we have to understand where there are constrains on scalability.

Give Me The NonStop

Now that I've ranted enough, let's come back to the best environment for highly-critical, high-volume, high-value transaction processing: A dedicated NonStop server. Maintaining a focused, non-virtualized, controllable, manageable, scalable processing environment with optimized languages and database access is still the best solution.


CPU and Swaps
cpsprd
cpubz
intbz
swaps
pkswap
% or per sec
'00:46 Tu 05/06/03'
'23:32 Mo 05/12/03'