

Linux & Scaling: The Essentials

Contributed by Administrator
Monday, 08 March 2010

Linux & Scaling: The Essentials

by Mark Rais, senior editor for reallylinux.com and author of Linux for the Rest of Us 2nd Ed. You may also find this article beneficial: So You Want to Learn MySQL by Jon Stephens. I vividly remember quietly listening to one of the many operation status calls that day, when someone interrupted the conversation and notified us that a person from the Congressional office was being patched into the call.

The question posed to the OPS personnel by this gentleman went something like this: "You're one of the few sites still delivering the report. How do we get a machine like yours?"

To which came a few muffled chuckles, followed by a mildly terse retort from an engineer who obviously hadn't slept for over 36 hours, "Sir, it's not a machine..."

There was more to the conversation but it clarifies my point. Scaling was and still is a major issue for all growing organizations. And scaling failures often result from a serious lack of experience with alternative systems.

"Scaling is still a major issue for all growing organizations."

A number of web sites I've been involved with that started just fine using Microsoft servers ended up becoming either bloated messes, or simply choked on ever increasing loads.

I've led several major re-architectures in my life. It takes a lot of time and money to scrap and revamp, especially if scaling is a core issue.

The key is to setup infrastructure with systems that scale without major effort or costly re-architectures.

A few years ago a major deal was made between AOL and CBS regarding the "Big Brother Show." At that time, a couple of my engineers and a handful of some of the best operations and network personnel you could wish for setup infrastructure that would handle what we anticipated to be something around the range of 250,000 hits per minute or around 4,000 hits per second. We were dead wrong.

Actually the non-technical people estimated we might see roughly 50,000 hits per second multiplied or more, but few of us expected this as reality. Regardless, up popped the website url on the show and a good chunk of those 10 million viewers of the premier show flipped on their computers and came running down the Internet pipe straight for our systems.

You want to know what it looks like when you see a spike like that? My lead engineer is pulling his hair out, the database engineer keeps yelling "the requests aren't getting through to the database," and the operations personnel are scrambling to try to cut the pipe before every router shuts down under the volume. Oh, and every executive near a phone is screaming at someone under them to fix the problem!

I recall getting numbers later indicating something around 800,000 hits per second. Regardless what exactly happened that night, some try to forget it all together, it was a bit much and everyone knew it.

So, as obvious as this may sound now, reflected through hind sight, the engineers converted all of the dynamic content into static html and rdist'ed content to every available web server. These outstanding and quick thinking engineers got the site backup within a couple hours, now running as static html on over 220 servers at a continuous 105%. Ironically, by then the heaviest volume of users had already melted away. At this time someone came up with an unwritten rule:

Web servers under peak need to deliver a sustainable 1,000 hits per second and that routing to the infrastructure has to have latency times less than 1/10th of a second. Let me assure you that at the time this kind of infrastructure cost A LOT OF MONEY!

However, today, it is not uncommon for a home grown Linux and Apache website to support these specifications for under a couple thousand dollars. IT IS FAR CHEAPER TODAY TO REACH THIS GOAL.

Linux and Apache, I propose, have changed the face of scaling and Internet serving forever.

"No kidding." you might add. You've already figured this out.

"Linux & Apache, I propose, have changed the face of scaling and Internet serving forever."

However, there are many organizations that simply do not understand the principal and continue to try scaling their Microsoft based web serving infrastructure.

The result is inevitably to pass the point of diminishing returns but become too attached to change. Or, the result is to simply throw more and more resources and money at the issue. In one recent example, I helped a non-profit organization setup an intranet server to support roughly 150 people. This is utterly small fry. I used a basic server, Linux, and Apache along with a few other OpenSource free software. The result was a stable, reasonable performance intranet site for less than \$700. At the same time, their technology leadership, who were versed exclusively with Microsoft products, purchased other hardware, Microsoft server licenses, and a dynamic intranet software for a second system at the cost of roughly \$6,500, not including the consulting fees. And even with this the server was slow. Scaling is just not the same when trying to address web server infrastructure between Linux and Microsoft OS worlds.

Not even Microsoft misses this point. They have been using UNIX based systems for their key infrastructure all the way back to the late 1990s. It often took four times as many of their own servers to match the same throughput so they switched. Today there is no doubt that Microsoft employs some Apache web servers for their MSN infrastructure, and I recall the days when hotmail was choking on volume and moved entirely OFF NT SYSTEMS to a more viable UNIX derivative (FreeBSD).

There are of course reasons behind all of this. Likely it has some to do with the bowels of software components which make up IIS, Microsoft's core web server, as well as some overarching aspects of the OS itself. However, I really prefer to focus more on why UNIX like systems have been effectively driving high volume web traffic for a long time.

"Apache is a formidable and professional web server that incorporates a decade of experience."

Apache isn't just a nifty name that conjures up images of the wild west and horses. It's a formidable and professional web server that incorporates a decade of experience. But underlying the Apache's core capabilities, and frankly the simplistic efficiency of the HTTPd, are the principles found in all UNIX variants for effectively handling threaded applications, memory management, and network protocols. That's also why Apache on Microsoft might very well have resulted in the same limited scaling solution you see today with IIS.

What makes this so apparent is that using Microsoft products often adds complexity. You take an ADS environment, tie it to the NT infrastructure and a SAN that's incorporating Exchange server and then drop in an IIS server or two or three and you are going to have the darndest time just trying to figure out how you do basic things like architect failovers into the infrastructure. Not to mention what you will need to do to try to address future scaling. One approach hailed by some is to use the wonderful power of the DNS round-robin to add redundant servers... except the Microsoft DNS services require you to manually intervene when one of those systems goes down. It all seems so complex and so manually intensive in my opinion.

Now take a different approach. Take a simple Linux & Apache model within existing organizational infrastructure. Take this model and add OpenExchange, add SAMBA services to accommodate the existing Microsoft ADS logins, add a SAN using RAID0 on large disks for the static content backups and RAID5 striping for the volatile data needs (on smaller disks mind you, to avoid death by positional latency) and you may actually have something beautiful.

Perhaps a piece of art in some people's eyes. In my own view you have a very robust, wide breath, great depth, highly scalable infrastructure without the complexity. Costs less too. But forget cost for now. Consider this. I need to now take this same infrastructure and support multiple web properties delivering dynamic content to users that often appear in spikes. Scaling this thing becomes a matter of properly placing the servers and a few switches for failover response. Sounds easy? It is reasonably so.

Best of all, what I need to scale an Apache server running hot (over 105% peak) is to simply add some more inexpensive hardware. No software complexity, no integration clutter, and no licensing fees.

I hope this brief encapsulation provides you with a reasonable perspective why Linux and Apache are increasingly vital infrastructure components to any growing organization. Scaling matters, and in the Linux world, scaling is integrated by design.