Is it time to get rid of the Linux OS model in the cloud?

Thursday, January 19, 2012 at 9:15AM

Todd Hoff in ChangeEverything, Linux, Paper

You program in a dynamic language, that runs on a JVM, that runs on a OS designed 40 years ago for a completely different purpose, that runs on virtualized hardware. Does this make sense? We've talked about this idea before in Machine VM + Cloud API - Rewriting The Cloud From Scratch, where the vision is to treat *cloud virtual hardware as a compiler target, and converting high-level language source code directly into kernels that run on it.*

As new technologies evolve the friction created by our old tool chains and architecture models becomes ever more obvious. Take, for example, what a team at UCSD is releasing: a phase-change memory prototype - *a solid state storage device that provides performance thousands of times faster than a conventional hard drive and up to seven times faster than current state-of-the-art solid-state drives (SSDs).* However, PCM has access latencies several times slower than DRAM.

This technology has obvious mind blowing implications, but an interesting not so obvious implication is what it says about our current standard datacenter stack. Gary Athens has written an excellent article, Revamping storage performance, spelling it all out in more detail:

Computer scientists at UCSD argue that new technologies such as PCM will hardly be worth developing for storage systems unless the hidden bottlenecks and faulty optimizations inherent in storage systems are eliminated.

Moneta, bypasses a number of functions in the operating system (OS) that typically slow the flow of data to and from storage. These functions were developed years ago to organize data on disk and manage input and output (I/O). The overhead introduced by them was so overshadowed by the inherent latency in a rotating disk that they seemed not to matter much. But with new technologies such as PCM, which are expected to approach dynamic random-access memory (DRAM) in speed, the delays stand in the way of the technologies' reaching their full potential. Linux, for example, takes 20,000 instructions to perform a simple I/O request.

By redesigning the Linux I/O stack and by optimizing the hardware/software interface, researchers were able to reduce storage latency by 60% and increase bandwidth as much as 18 times.

The I/O scheduler in Linux performs various functions, such as assuring fair access to resources. Moneta bypasses the scheduler entirely, reducing overhead. Further gains come from removing all locks from the low-level driver, which block parallelism, by substituting more efficient mechanisms that do not.

> *Moneta performs I/O benchmarks 9.5 times faster than a RAID array of conventional disks, 2.8 times faster than a RAID array of flash-based solid-state drives (SSDs), and 2.2 times faster than fusion-io's high-end, flash-based SSD.*

The next step in the evolution is reduce latency by removing the standard IO calls completely and:

> *Address non-volatile storage directly from my application, just like DRAM. That's the broader vision—a future in which the memory system and the storage system are integrated into one.*
>
> *A great deal of the complexity in database management systems lies in the buffer management and query optimization to minimize I/O, and much of that might be eliminated.*

But there's a still problem in the latency induced by the whole datacenter stack (paraphrased):

> *This change in storage performance is going to force us to look at all the different aspects of computer system design: low levels of the OS, through the application layers, and on up to the data center and network architectures. The idea is to attack all these layers at once.*

In Revisiting Network I/O APIs: The netmap Framework, written by Luigi Rizzo, the theme of a mismatch between our tools and technology continues:

> *Today 10-gigabit interfaces are used more and more in datacenters and servers. On these links, packets flow as fast as one every 67.2 nanoseconds, yet modern operating systems can take 10-20 times longer just to move one packet between the wire and the application. We can do much better, not with more powerful hardware but by revising architectural decisions made long ago regarding the design of device drivers and network stacks.*
>
> *In current mainstream operating systems (Windows, Linux, BSD and its derivatives), the architecture of the networking code and device drivers is heavily influenced by design decisions made almost 30 years ago. At the time, memory was a scarce resource; links operated at low (by today's standards) speeds; parallel processing was an advanced research topic; and the ability to work at line rate in all possible conditions was compromised by hardware limitations in the NIC (network interface controller) even before the software was involved.*

There's a whole "get rid of the layers" meme here based on the idea that we are still using monolithic operating systems from a completely different age of assumptions. Operating systems aren't multi-user anymore, they aren't even generalized containers for running mixed workloads, they are specialized components in an overall distributed architecture running on VMs. And all that overhead is paid for by the hour to a cloud provider, by greater application latencies and by the means required to overcome them (caching, etc).

Scalability is often associated with specialization. We create something

specialized in order to achieve the performance and scale that we can't get from standard tools. Perhaps it's time to see the cloud not as a hybrid of the past, but something that should be specialized, that it's something different by nature. We are already seeing networking transform away from former canonical hardware driven models to embrace radical new ideas such as virtual networking.

You mission, should you choose to accept it, is to rethink everything. Do we need a device driver layer? Do we need processes? Do we need virtual memory? Do we need a different security model? Do we need a kernel? Do we need libraries? Do we need installable packages? Do we need buffer management? Do we need file descriptors? We are stuck in the past. We can hear the creakiness of the edifice we've built layer by creaky layer all around us. How will we build applications in the future and what kind of stack will help us get their faster?

Related Articles

- The Non-Volatile Systems Laboratory at UCSD
- Moneta: A High-performance Storage ArrayArchitecture for Next-generation, Non-volatileMemories
- Phase Change Memory-Based "Moneta" System Points to the Future of Computer Storage
- Revamping storage performance by Gary Athens (requires ACM membership)
- Machine VM + Cloud API - Rewriting The Cloud From Scratch
- A Processor for Apps - a chip design that is specialized for Android mobile devices could be 11 times as energy efficient as a conventional mobile processor.
- Exokernel - The idea behind exokernels is to force as few abstractions as possible on developers, enabling them to make as many decisions as possible about hardware abstractions.

- Turning Down the LAMP: Software Specialisation for the Cloud
- On Reddit
- Mirage - an exokernel for constructing secure, high-performance network applications across a variety of cloud computing and mobile platforms.
