Every Silver Lining Has a Cloud

**CACHE IS KING.
AND IF YOUR
CACHE IS CUT,
YOU'RE GOING
TO FEEL IT.**

who is
KV?

click for video

Dear KV,

My team and I have spent the past eight weeks debugging an application performance problem in a system that we moved to a cloud provider. Now, after a few drinks to celebrate, we thought we would tell you the story and see if you have any words of wisdom.

In 2016, our management decided that—to save money—we would move all our services from self-hosted servers in two racks in our small in-office data center into the cloud so that we could take advantage of the elastic pricing available from most cloud providers. Our system uses fairly generic, off-the-shelf, open-source components, including Postgres and memcached, to provide the back-end storage to our web service.

Over the past two years we built up a good deal of expertise in tuning the system for performance, so we thought we were in a good place to understand what we needed when we moved the service to the cloud. What we found was quite the opposite.

Our first problem was very inconsistent response times to queries. The long tail of long queries of our database began to grow the moment we moved our systems into the cloud service, but each time we went to look for a root cause, the problem would disappear. The tools we would normally use to diagnose the issues we found on bare metal also gave far more varied results than expected.

The savings in cloud computing comes at the expense of a loss of control over your systems.

In the end, some of the systems could not be allocated elastically but had to be statically allocated, so the service would behave in a consistent manner. The savings that management expected were never realized. Perhaps the only bright side is that we no longer have to maintain our own deployment tools, because deployment is handled by the cloud provider.

As we sip our drinks, we wonder, is this really a common problem, or could we have done something to have made this transition less painful?

Rained on our Parade

Dear Rained,

Clearly, your management has never heard the phrase, “You get what you pay for.” Or perhaps they heard it and didn’t realize it applied to them. The savings in cloud computing come at the expense of a loss of control over your systems, which is summed up best in the popular nerd sticker that says, “The Cloud Is Just Other People’s Computers.”

All the tools you built during those last two years work only because they have direct knowledge of the system components down to the metal, or at least as close to the metal as possible. Once you move a system into the cloud, your application is sharing resources with other, competing systems, and if you’re taking advantage of elastic pricing, then your machines may not even be running until the cloud provider deems them necessary. Request latency is dictated by the immediate availability of resources to answer the incoming request. These resources include CPU cycles, data in memory, data in CPU caches, and data

on storage. In a traditional server, all these resources are controlled by your operating system at the behest of the programs running on top of the operating system; but in a cloud, there is another layer, the virtual machine, which adds another turtle to the stack, and even when it's turtles all the way down, that extra turtle is going to be the source of resource variation. This is one reason you saw inconsistent results after you moved your system to the cloud.

Let's think only about the use of CPU caches for a moment. Modern CPUs gain quite a bit of their overall performance from having large, efficiently managed L1, L2, and sometimes L3 caches. The CPU caches are shared among all programs, but in the case of a virtualized system with several tenants, the amount of cache available to any one program—such as your database or memcached server—decreases linearly with the addition of each tenant. If you had a beefy server in your original colo, you were definitely gaining a performance boost from the large caches in those CPUs. The very same server running in a cloud provider is going to give your programs drastically less cache space with which to work.

With less cache, fewer things are kept in fast memory, meaning that your programs now need to go to regular RAM, which is often much slower than cache. Those accesses to memory are now competing with other tenants that are also squeezed for cache. Therefore, although the real server on which the instances are running might be much larger than your original hardware—perhaps holding nearly a terabyte of RAM—each tenant receives far worse performance in a virtual

instance of the same memory size than it would if it had a real server with the same amount of memory.

Let's imagine this with actual numbers. If your team owned a modern dual-processor server with 128 gigabytes of RAM, each processor would have 16 megabytes (not gigabytes) of L2 cache. If that server is running an operating system, a database, and memcached, then those three programs share that 16 megabytes. Taking the same server and increasing the memory to 512 gigabytes, and then having four tenants, means that the available cache space has now shrunk to one-fourth of what it was—each tenant now receives only four megabytes of L2 cache and has to compete with three other tenants for all the same resources it had before. In modern computing, cache is king, and if your cache is cut, you're going to feel it, as you did when trying to fix your performance problems.

Most cloud providers offer systems that are nonelastic, as well as elastic, but having a server always available in a cloud service is more expensive than hosting one at a traditional colocation facility. Why is that? It's because the economies of scale for cloud providers work only if everyone is playing the game and allowing the cloud provider to dictate how resources are consumed.

Some providers now have something called Metal-as-a-Service, which I really think ought to mean that an '80s metal band shows up at your office, plays a gig, smashes the furniture, and urinates on the carpet, but alas, it's just the cloud providers' way of finally admitting that cloud computing isn't really the right answer for all applications. For systems that require deterministic performance guarantees to work well, you really have

to think very hard about whether or not a cloud-based system is the right answer, because providing deterministic

guarantees requires quite a bit of control over the variables in the environment. Cloud systems are not about giving you control; they're about the owner of the systems having the control.

KV

Related articles

➡ Cloud Calipers

Kode Vicious

Naming the next generation and remembering that the cloud is just other people's computers

<https://queue.acm.org/detail.cfm?id=2993454>

➡ 20 Obstacles to Scalability

Sean Hull

Watch out for these pitfalls that can prevent web application scaling.

<https://queue.acm.org/detail.cfm?id=2512489>

➡ A Guided Tour through

Data-center Networking

Dennis Abts, Bob Felderman

A good user experience depends on predictable performance within the data-center network.

<https://queue.acm.org/detail.cfm?id=2208919>

Kode Vicious, known to mere mortals as George V. Neville-Neil, works on networking and operating-system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of

ACM, the Usenix Association, and IEEE. Neville-Neil is the co-author with Marshall Kirk McKusick and Robert N. M. Watson of The Design and Implementation of the FreeBSD Operating System (second edition). He is an avid bicyclist and traveler who currently lives in New York City.

Copyright © 2018 held by owner/author. Publication rights licensed to ACM.