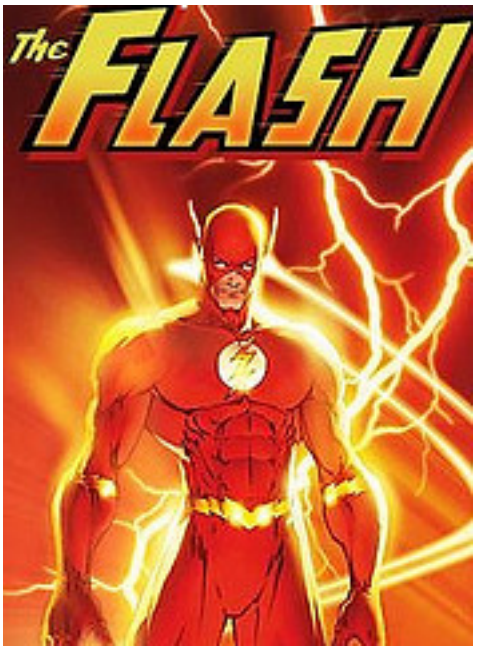
Vertical Scaling Ascendant - How are SSDs Changing Architectures?

Wednesday, July 25, 2012 at 9:17AM

Todd Hoff



With Amazon announcing new High I/O 2TB SSD instances the age of SSD has almost arrived. I say almost because the $27K a year price tag for the hi1.4xlarge on demand instance type is outside the budget of many. Yet even at the full on demand rate the price per IOP for the high IO instance is attractive: 27 cents ($27K/100K IOPS) per vs $1.25 for disk. With the obvious benefits of giant SSD machines combined with 10 Gbps networking, it’s interesting to consider: what architecture decisions might you make differently in the future?

More Headroom for Vertical Scaling Simplifies Everything

The beauty of higher hardware performance is it shifts effort away from the programmer which allows developers to focus on the business of business, minimizing trickeration. This has always been the allure of vertical scaling and is well realized by SSDs through a combination of high throughput, low latencies, and just as important, high densities.

We have a few early examples showing the performance punch of the new High IO instance:

- Mike Krieger of Instagram reports:
  - Very cool--striping the SSD drives together in the new EC2 instance type yields ~0.5 ms avg latency even under pretty heavy load.
  - Graph of the day: disk latency for EC2 SSD drive (in orange) vs EBS (in green), while the SSD does about 4000x the IOPS http://d.pr/i/3QZ2
- Colin Howe from Conversocial wrote about their experiences transferring MongoDB to a hi1.4xlarge and found:
  - Average response time is 43% faster (392ms to 274ms).
  - One random IO heavy view data is now 74% faster (877ms to 504ms).
  - CPU usage went from around iowait 90% to 3%.
- Netflix has already published a very thorough post on the wonderfulness of SSD for both performance and taming the long latency tail:
  - They see 100K IOPS or 1GByte/sec on a untuned system.
  - The hi1.4xlarge configuration is about half the system cost for the same throughput.
  - The mean read request latency was reduced from 10ms to 2.2ms.
  - The 99th percentile request latency was reduced from 65ms to 10ms.

But not all is bliss in the garden. Jeff Darcy wrote in Testing [GlusterFS] on Amazon's New SSD Instances that they've found real world performance unimpressive and even worse in some cases, concluding:

> *They're certainly a welcome improvement for this worst-case kind of workload, but I've seen their ilk before so the only thing that's really new to me is the*

high price tag.

Practical Implications of the High IO Instances

- **Turbocharging**. When your performance is going to heck and you need help now, the High IO instance is a little light in the dark. Maciej Dobrzanski from dba square ran some benchmarks and concluded: This can actually make EC2 usable even for large and busy MySQL databases, which up until now often wasn't a viable option.
- **Delay horizontal architecture/sharding**. That step to a horizontal architecture is a big one. If you are worried about all your data fitting into RAM, 2 TBs is a lot of headroom. Do you still really need a disk? Stuff it all on SSDs.
- **Less time spent optimizing**. SSDs are often used as a magic performance sauce. Use an SSD and performance instantly improves. That, along with the ability to use a vertical architecture means programmers can do much of less of the finicky optimization type programming that wastes so much time.
- **Remove caching layers**. Netflix was able to remove a memcached layer on 48 instances and replace it with 15 I/O instances with no intervening cache, while maintaining a lower overall latency. Using RAM to offload IO from the database was no longer necessary with the SSD backed Cassandra nodes and there was still plenty of CPU headroom remaining on the nodes.
- **Large memory algorithms**. Mbreeze gives a good example why this is helpful:
    - I do genome mapping where our indexes won't entirely fit in memory. It would be very handy to be able to spin up a few of these instances, load the indexes from an EBS volume onto the local SSDs, then run for a couple of hours or so. This is a

very I/O intensive job that we need to run about once a week, but then the rest of the time could be idle. SSDs would make our jobs run significantly faster. So much so that we've toyed with the idea of adding SSDs to our in-house cluster, but couldn't quite justify the costs. This might actually shift the cost savings to get our lab to migrate to EC2 as opposed to our in-house or university cluster.

- **Peak usage handling**. From pdeshen:
  - We use ec2 within our telecom infrastructure which manages peaks of several thousands of calls with full duplex call recording on (think IO here) We are using the elasticity of ec2 to scale on demand. This kind of instance is a nice addition for us as far as our auto scaling is concerned.
- **Skip EBS entirely**. Fizzer makes an interesting point about EBS:
  - This is a game changer for big sites on EC2. The key word here is local: 2 TB of local SSD-backed storage. In this video, Foursquare says the biggest problem they're facing with EC2 is consistency in I/O performance. They say that the instance storage simply isn't fast enough for them, and while EBS is fast enough when RAIDed, it isn't consistent since it isn't local (EBS is traffic goes over the network). Foursquare says in that video they're planning to migrate off of EC2, in part due to I/O performance. I'll be interested to hear whether or not this instance type changes their minds.
- **Changes to database selection criteria**. Given the garbage collection overhead of SSDs, databases that use log structured update strategies may be preferred over databases that use in-place update strategies. Cassandra, for example, uses LSM Trees to stream data to disk, which also happens to reduce write amplification. Now you just need to have enough CPU to drive all that IO. Though Vladimir Rodionov asserts: Cassandra is not able to utilize the random read IO capacity of 100K of 4K blocks per

second using 8 CPU cores of new Amazon High I/O instance.

- **Consolidation**. A number of people have mentioned using the high-capacity servers to reduce their server count. Netflix calculated they could shrunk their server count by replacing 48 m2.4xlarge and 36 m2.xlarge with 15 hi1.4xlarge nodes.
- **Shift to server side processing**. The high IO instances are so fast it's a waste to get data, bring it to a different machine, perform a computation, and then write it back again. Use all those CPUs by using coprocessors/plugins to shift computations to the server instead of out to an intermediate cluster.
- **Design to be IO bound on reads**. Not a new idea, but SSDs make architecting towards reads even more attractive as there is no penalty for random reads. One idea is to precalculate on writes so getting results are simply read operations.

Fewer Machines Means a Lower Total Overall Cost

- **Half the cost**. Netflix found that the hi1.4xlarge configuration is about half the system cost for the same throughput, even though the individual instance costs are higher.
- **Reserved price tuning**. Using reserved instances drops prices considerably. A one year lease reservation for a "heavy utilization" instance comes out to $7,280 per year + $0.621 per hour for a total of $12,719. Reserving for three years brings down the cost to $656 a month, though some argue locking in for this amount of time will rob you of subsequent price discounts as new hardware comes along.

High Availability is Still Necessary

- **SSDs are persistent but they are not a magic reliability sauce**. A

resilience architecture is still needed to handle failure and recovery. Replication and write behind to EBS/S3 still need to be in your toolbox. SSDs can be corrupted just like disk. If a hard failure occurs, what state is the SSD in when the node comes back up? Keep in mind your connection to S3 is 1Gb/s so restoring from S3 will take a considerable amount of time, which makes replication a necessity for interactive workloads.

- Adrian Cockcroft **(of Netflix) brings up an advantage of SSDs for HA**: faster net and disk greatly reduces repair time and impact so we can load up the instances with far more data.
- **Traditionally SSD performance degrades over time**. Wonder if it makes sense to fail over to different machines periodically to "reformat" SSD drives?

It's Not All Peaches and Cream

- **What's a programmer to do?** There's a general theme that our experience with programming SSDs is neither deep or wide, so there isn't a lot of best practices information available. Do you want to use append only algorithms? Is writing in place prefered? Is sequential IO fast? How long can you run an active system on SSDs? Are B-trees still a go to data storage structure? There's still lots of debate about these kind of questions.
    - For example, the folks at Acnu have found working with SSDs is a very different experience: Our simple tests show that the Disk Access Model (DAM) and its derivatives are not very good predictors of the performance of write sequences on SSDs. We found a scheme to efficiently use SSDs for the Acunu storage core only once we freed ourselves of the ideas that the DAM planted in our heads.
- **SSDs from different vendors differ quite a bit**. This means parameters and algorithms will need experimentation to decide what

works best. And what works best on Amazon may not work best on another machine. This adds another interesting twist to cloud portability and vendor lock-in.
- **Faster IO pushes bottlenecks elsewhere in the system**. Low level problems in the TCP stack or IRQ handling will be uncovered. Higher up the stack is also a problem. For example, The Voldemort team found Java and SSDs were a troubling mix: On SSDs, we believe that the garbage collection overhead will have a significant performance impact for Java based systems, since IO is no longer the bottleneck.
- **Need more lower-end instance types over more regions**. These things come in time.
- It would be great to have direct access to flash without having to pretend all the world is a disk.

Related Articles

- New High I/O EC2 Instance Type - hi1.4xlarge - 2 TB of SSD-Backed Storage (On Hacker News)
- Netflix: Benchmarking High Performance I/O with SSD for Cassandra on AWS (On Hacker News)
- Expanding The Cloud – High Performance I/O Instances for Amazon EC2 by Werner Vogels
- Log-structured file systems: There's one in every SSD
- I/O Performance (no longer) Sucks in the Cloud by James Hamilton
- Testing on Amazon’s New SSD Instances by Jeff Darcy
- Solid-state revolution: in-depth on how SSDs really work by Lee Hutchinson
- Quora: Do append-only DB log structures benefit from Flash memory and SSDs?
- SSDs and Distributed Data Systems by Jay Kreps

- Stratified B-trees and versioning dictionaries
- Why theory fails for SSDs by Irit Katriel
- Log file systems and SSDs – made for each other? by Andy Twigg
- Flashing Databases: Expectations and LImitations
- Quora: Are algorithms optimized for regular hard drives *not* optimized for solid state drives?
- Velobit Blog on SSD, Application Performance and Storage
- Fusion-io shoves OS aside, lets apps drill straight into flash
- What every developer should know about database scalability by Jonathan Ellis
- Voldemort on Solid State Drives by Vinoth Chandar
