NVRam for Fun and Profit

Author: Shaun Thomas
Date: September 15th, 2011
Venue: Postgres Open 2011

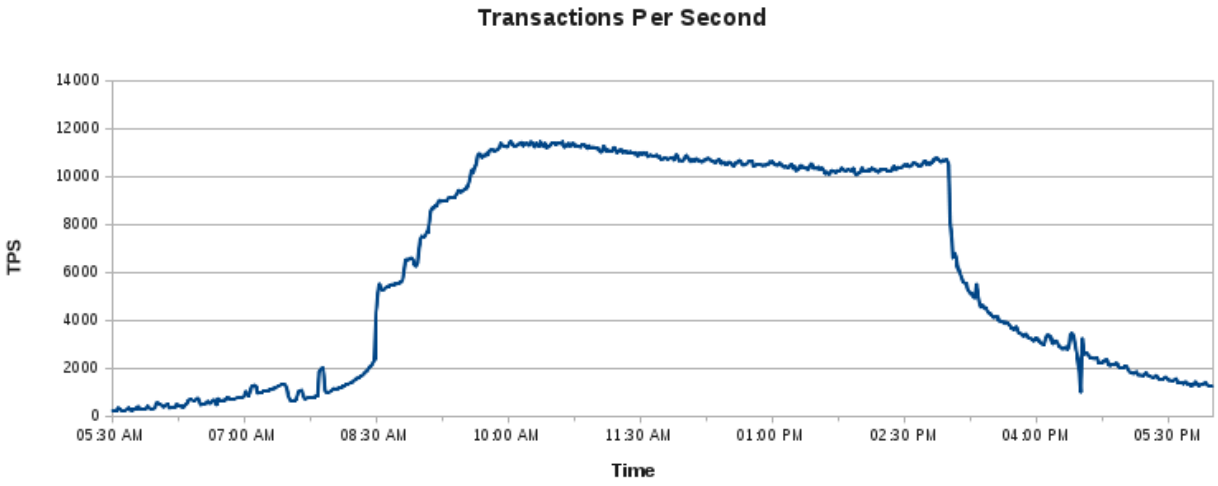
Your Presenter

Shaun M. Thomas
Senior Database Administrator
Peak6 OptionsHouse

Be very, very afraid.

Welcome to the Jungle

"This is your stock market.
This is your stock market... on drugs."



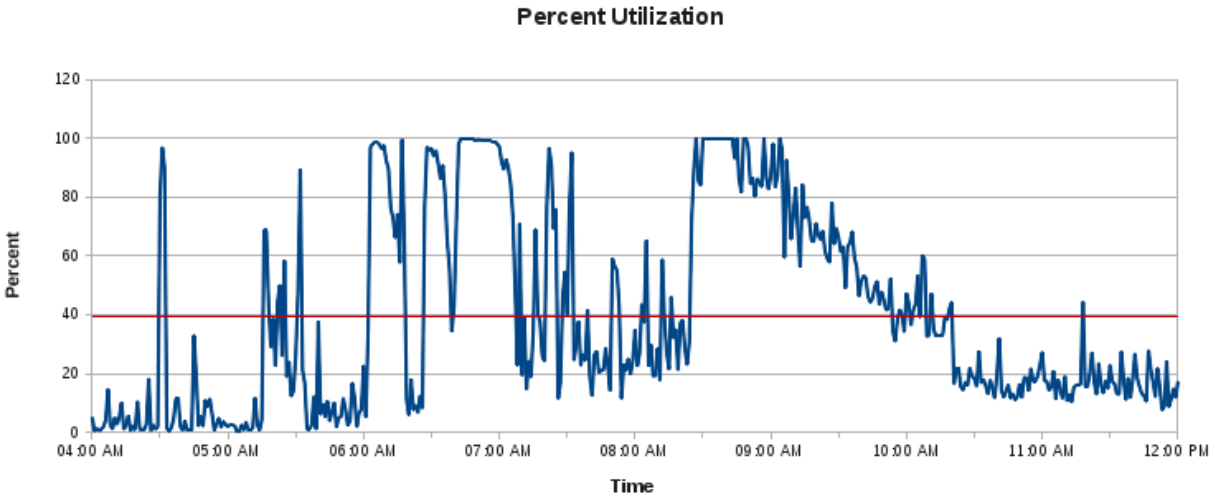
This is what the stock market basically looks like to the database: a whole lot of nothing until the market opens, and then it's exponential growth until the trading frenzy stabilizes. That graph is what's left after several levels of client-side application caching and load balancing. There are a few things to remember about trading in general:

- It's not open to interpretation.
- The market is absolutely unforgiving.
- Clients can literally lose millions if our platform dies.
- The database can *never* go down.
- The database can *never* go down.

The OptionsHouse application is spread across a massive parallel cluster of webservers, connection poolers, request queues, and cache servers. Even still, over ten thousand transactions per second manage to reach the database. From our pretty little graph, I'm sure you can imagine what would happen if the database became unavailable for any reason between 8:30am and 3:00pm.

Natural Consequences

"I'm surprised that MSA hasn't burst into flames!"



The reality of this situation was actually manageable, but it was immediately obvious to anyone that any increase in customer load would make the database ignite like dry kindling. Mostly at issue was our RAID makeup; too few disks and too few IOPS for the random IO needs. The system was also a little RAM starved, but we'll get to that.

But Who's Counting?

1. 1140 TPS from a 6-disk RAID1+0
2. 11,500 TPS at peak from the app
3. Client count doubles roughly every year

This is what happens when a high-TPS service meets a low-TPS device. Why did we only have a 6-disk RAID-10? Forget about that for a second. After rebalancing the drives in that MSA and formatting with XFS, we bumped that figure up to 1440 TPS (or 1660 TPS with moderate short-stroking). That's a massive improvement, but doesn't even dent the needs of our rapidly growing client-base.

This is a common problem in the OLTP arena. How do you balance high transactional needs with relatively slow disk bandwidth?

Trouble in Paradise

Things that can break caches:

- VACUUM
- Maintenance
- Errant queries
- Database dumps

And this is where we get to the heart of the matter: cache poisoning. Have a giant table that doesn't fit into memory, but enough is loaded that TPS is high? OS Cache is relatively primitive, and a few cache reads are enough to wipe out hours of random seeks triggered by user queries through the day. A badly timed autovacuum will ruin your day. Even timed vacuums and other maintenance will effectively reset the query cache to some useless state for the next high TPS period. It's so fragile even, that one ad-hoc query against the wrong table will flush necessary data segments. Backups have the same problem as vacuum, but at least that can be shipped over to an offline server thanks to hot standby and replication.

This perpetuates a continuous cycle of slow response during high TPS times of day after maintenance is complete. More memory, even enough to cache the whole database, is a great thing to have, but it must be bootstrapped!

By the Bootstraps

Things that can fix caches:

- Brute force: `dd` (relfilenode)
- More elegant: pgfincore
- MacGyver would be proud!
- Doesn't solve constant turnover
- May be iffy under crash-recovery

The last point is the most crucial, here. Using `dd` is a bit hacky, but `pg_class` *does* tell us which files belong to which tables and indexes, and we can use that to preload memory at any time. Throw it in cron, and you have a poor-man's bootstrap.

And pgfincore? What's better than taking a snapshot of the filesystem cache when the database is actually running smoothly, and reverting to that? Not much, actually. And it is probably the closest thing available to always having the database in an ideal state.

But then Turnover crashes the party, and he's not a very nice guy. For extremely active databases, there may *never* be an ideal state to preload or snapshot, and disk performance isn't very forgiving. There are also scenarios where a database must be made available immediately, such as hardware failover. In that case, the time required to bootstrap might not be available.

So what's the alternative?

A Way Out

Faster storage...

- SAN
  - EMC
  - RAMSAN
  - Whiptail
- SSDs

        - Intel, OCZ, etc.
    - PCIe
        - RAMSAN
        - FusionIO
        - Virident

SAN-tastic

Though awesome, SANs have a few drawbacks.

- Expensive
- Big. Huge. Immense. Gigantic.

What about SSD / NVRAM SANs?

- Still expensive
- Usually combine NVRAM chips with RAID

Drives are a necessary thing in a SAN, and we're glad SANs have all of that nifty de-duplication, snapshotting, copying, exporting, internal integrity checking, and other great features. But you pay for each and every one of those things, and if you don't need that amount of storage, a LUN with 80 drives in a RAID-10 (to get the IOPS you want) will be mostly wasted. Back to our usage example, assuming an empty cache and drives that can supply 200 IOPS each, we'd need 60 drives in a RAID-10 just for 1/1 with the existing transaction load. A transaction load estimated to double within two years. That's a *big* SAN.

SSD-based SANS generally solve the problem of size, but the expense is still there. Whiptail retails for almost $200k, and RAMSAN devices go from $50k on up depending on capacity. This is still much cheaper than a traditional SAN, however. What these vendors don't like to mention though, is that there are a few ways to turn NVRAM into large sequential blocks, and one of them includes using some RAID level, usually 0 or 5. RAMSAN says it uses RAID-5 on its MLC-based devices. One thing about NVRAM we all know, is that they hate to write. RAID-5 is a write multiplier. Thus these devices sacrifice longevity for redundancy. It's a valid tradeoff, but one many would miss as a consideration.

Don't get us wrong, many RAID levels would produce ridiculous IOPS thanks to all those parallel reads!

SSD Revolution

SSDs are great!

- Orders of magnitude more IOPS
- Can be used as a passthrough cache
- May have built-in deduplication

But...

- Most lack capacitor-backed cache
- Many rely on TRIM-support
- Others have overly aggressive GC

SSDs really are great, aren't they? They really have revolutionized latency and IOPS on both consumer and enterprise-class equipment. They cost a bit more, but they're orders of magnitude faster, use less

power, and have no moving parts prone to failure. With ZFS in particular, both read caching and the write log can be redirected to fast SSDs for major performance boosts. With internal deduplication, lifetime of NVRAM cells is extended as less writes are necessary to store the same data.

But there's a dark-side here. Due to the current technology used for NVRAM chips, writes are much slower than reads, and degrade the cell as a bonus. In addition, the more data on the drive, the slower writes become. Some drives circumvent this by having a large area of reserve space to use as a scratch or replacement area, but write amplification is still an issue. Thus many drives have a write cache that helps hide the slow writes from the host operating-system. Unfortunately a power-outage can render this cache moot, and many drives don't have capacitor or battery-backing to ensure it survives such an outage.

But hidden dangers lurk within the methods used to clean up old data blocks. Many drives rely on TRIM, which not only needs OS support, but controller support in the case of RAIDS. But TRIM is relatively new, and few RAID controllers have support for it. Some enterprise-class SSDs have tried fixing this by adding firmware-level garbage collection that doesn't rely on TRIM, which runs during low utilization. In order to preserve performance, early iterations of these garbage collectors are actually quite aggressive, and may drastically shorten the life of the drive itself.

Besides the above, even expensive enterprise-class drives have a history of crippling firmware bugs which render the drives effectively unusable. They're a relatively new and unproven technology with a lot of drawbacks, relegating them to the afore-mentioned intermediate cache layer, or primarily read-heavy environments. But the tech has drastically advanced in just the last year, and we may be seeing many of these problems disappear with subsequent generations.

Get PCIe'd

Same benefits as SSDs, and...

- Direct communication over the PCIe bus
- May offer passthrough drivers
- Available in "raw" configurations
- Generally have capacitor-backed caches

Though,

- They're somewhat expensive
- Must be bought in pairs for failover configurations

Now we're getting somewhere. While they still have many of the write limitations of SSDs since they use the same chips, the configuration and hardware is vastly different. While some vendors such as OCZ and RAMSAN still use RAID to link chips and make the devices bootable and increase throughput, makers such as Fusion IO and Virident provide drivers instead. Fusion IO in particular doesn't even combine the separate cells in their larger cards, allowing customers to decide how, or even if, RAID should be involved.

This kind of "raw" availability can be a double-edged sword, because it means the devices are simply not bootable, and are completely dependent on vendor-supplied drivers. This also means increases in host CPU directly affect the performance of the PCIe card. At the same time, garbage collection isn't tied to TRIM, and can be altered through driver settings. With the larger reserve area (a 640GB card may actually have 900GB of NVRAM on-board) performance is less likely to degrade as the card fills. And all those writes are backed by capacitors.

But all this comes at a cost. 900GB of even Intel's top of the line SSD would cost about $1500, while it's not uncommon for the same amount of space to cost five to ten times that much on a PCIe card. And don't forget, they must be purchased in pairs for dual failover configurations since an internal device can't really be multipathed. Though vendors are starting to offer hot-swap PCIe expansion chassis, so even this

caveat won't be true for much longer.

Buy the Numbers

PGBench tests: scale of 1000, caches dropped before each test, peak recorded from 10 to 60 concurrent clients running 1000 transactions.

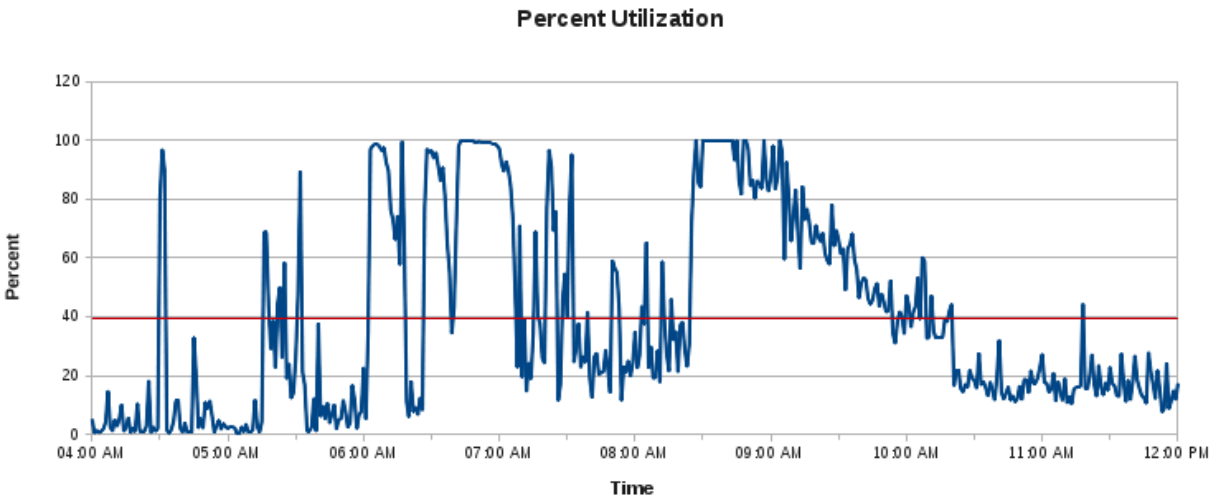
- 8-disk RAID1+0: 1,370 / 1,440 TPS
- ioDrive 640: 5,760 / 17,300 TPS
- R/W performance of a 30+ drive RAID1+0
- 100-drive RAID1+0 to match read performance

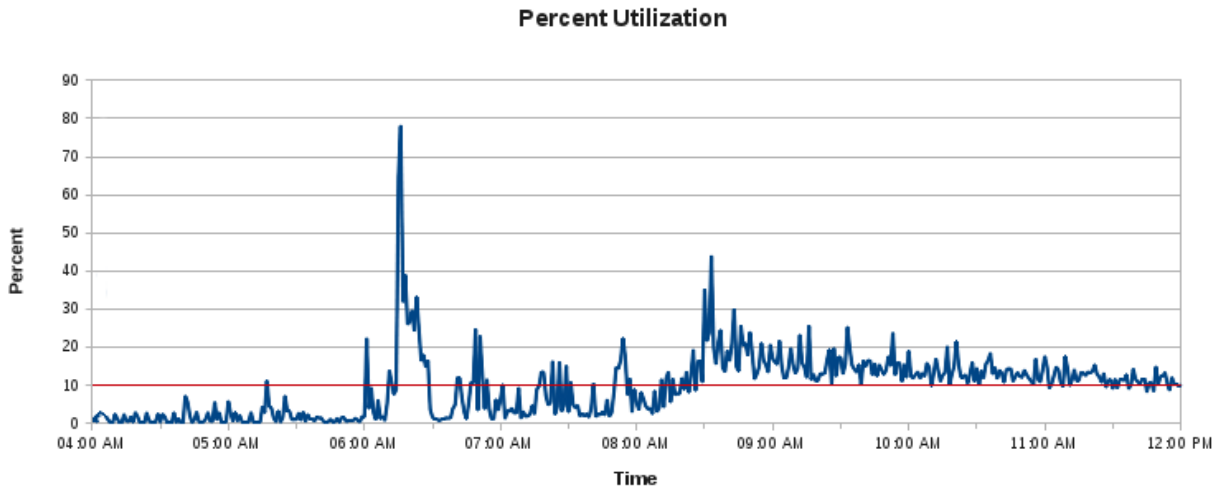
Remember that first slide that showed our financial application using nearly 12,000 transactions per second? An ioDrive can serve that directly, with no OS caching needed at all. This is the same as instant startup with no cache warming or disk IO saturation even during our heaviest client load. PGBench tests don't translate exactly, but our tests also allowed absolutely no OS-level caching for multiple reads. In practice, the ioDrive filled the cache so quickly, our practical TPS was much higher than these tests.

It's true that a tiered SAN or hybrid storage device could likely provide much better readings than a mere 8-disk RAID. Really, all we needed to do was reduce the multiple between the TPS our application needed, and what the disks could provide. A single DAS extension would have accomplished that, or even possibly switching over to Fuse + ZFS and an NVRAM based ZIL. But at the time, we wanted a drop-in solution that would scale for at least the next 18 months while we migrated to a more scalable cluster. And indeed, we still have the ability to use the FusionIO cards in the future as a ZIL in a much larger ZFS system, or just use FusionIO's intermediate cache driver.

These cards quite literally saved our butts. But the point isn't the performance of an ioDrive, but of any enterprise class NVRAM-based device. A single PCI card can directly replace high-throughput storage for instantaneous access relatively safely. If a data set is too large to fit, an intermediate cache layer or a separate tablespace can set it aside as tiered storage. In practice, the particular card we chose provided about half the TPS of direct memory reads on our test server. That's pretty hard to beat.

In Conclusion





Those graphs are for the same time period, but there are two important differences:

1. All drive saturation is gone.
2. The second graph was driven by 20% more customers.

The problem with the first chart is that it hit 100% so frequently for long periods of time. There was really no way to know what kind of disk bandwidth it really needed due to the limited throttle. The second graph, combined with our measured numbers for the ioDrive, explain why. A couple batch jobs still drive utilization into the 80's for a short period because none of that data is cached. There is a small spike at market open, but otherwise the median average is 10% instead of 40%; it would seem our mixed usage tests with pgbench were fairly accurate.

More Information

- pgfincore
- ZFS Best Practices
- FusionIO
- RAMSAN
- Virident
- Dell PowerEdge PCIe Chassis
- OptionsHouse

Questions