Accelerating Hadoop at Twitter with NVMe SSDs: A Hybrid Approach

Matthew Singer, Sr. Staff Hardware Engineer

Dave Beckett, Staff Site Reliability Engineer

Varun Sampat, Staff Hardware Engineer

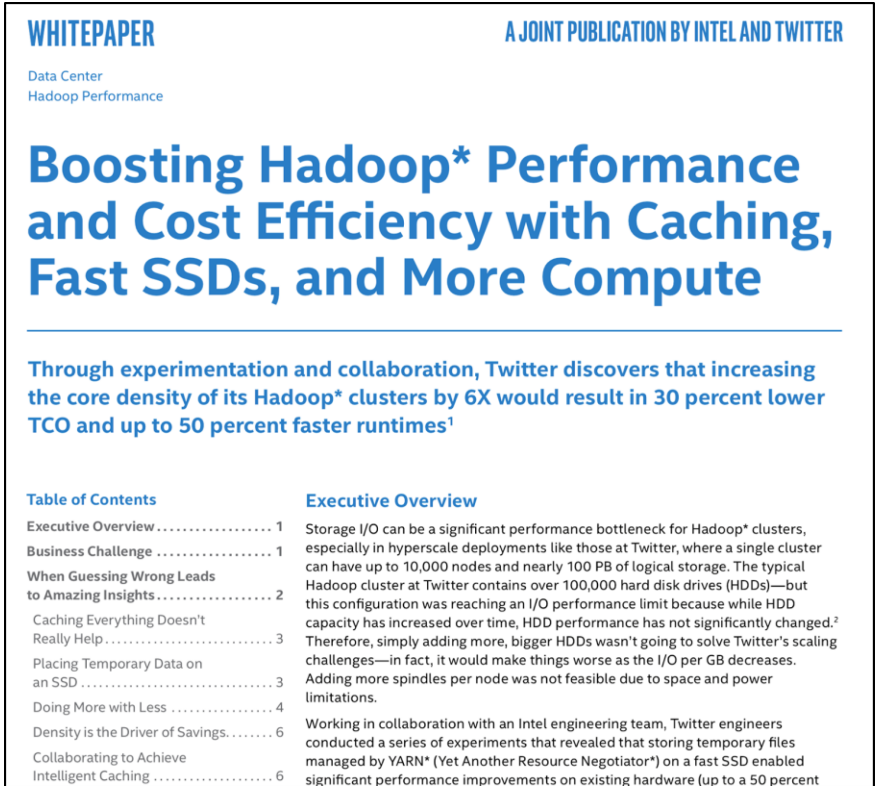
Mark Schonbach, Sr. Site Reliability Engineer

Paper Introduction

https://intel.ly/2Soo7J

How do you scale a service like Hadoop, which is heavily reliant on HDDs? HDDs are getting slower per unit of storage.

We experimented with NVMe SSD caching solutions that helped us break our reliance on spindle count, and found that we could add flash to these systems while reducing costs.



WHITEPAPER

A JOINT PUBLICATION BY INTEL AND TWITTER

Data Center
Hadoop Performance

Boosting Hadoop* Performance and Cost Efficiency with Caching, Fast SSDs, and More Compute

Through experimentation and collaboration, Twitter discovers that increasing the core density of its Hadoop* clusters by 6X would result in 30 percent lower TCO and up to 50 percent faster runtimes[1]

Table of Contents

Executive Overview

Storage I/O can be a significant performance bottleneck for Hadoop* clusters, especially in hyperscale deployments like those at Twitter, where a single cluster can have up to 10,000 nodes and nearly 100 PB of logical storage. The typical Hadoop cluster at Twitter contains over 100,000 hard disk drives (HDDs)—but this configuration was reaching an I/O performance limit because while HDD capacity has increased over time, HDD performance has not significantly changed.[2] Therefore, simply adding more, bigger HDDs wasn't going to solve Twitter's scaling challenges—in fact, it would make things worse as the I/O per GB decreases. Adding more spindles per node was not feasible due to space and power limitations.

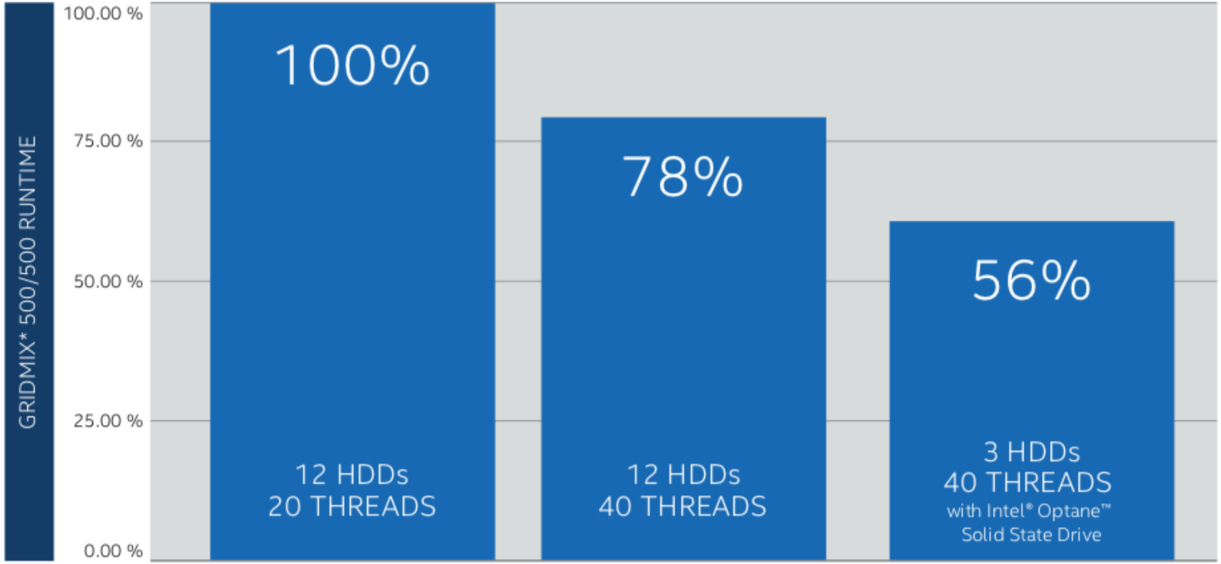
Working in collaboration with an Intel engineering team, Twitter engineers conducted a series of experiments that revealed that storing temporary files managed by YARN* (Yet Another Resource Negotiator*) on a fast SSD enabled significant performance improvements on existing hardware (up to a 50 percent

What We Found

We could categorize the data into two buckets:

1. HDFS Reads and Writes
   - Big sequential access, 1 Per HDD, Uncachable
     - Input data is generally generated once and not consumed for some time.
2. YARN Reads and Writes
   - Random size, random access, 1 per container
     - Short lived data is usually produced then consumed.
     - ***Great candidate for storage in flash memory.***

Conclusions of the Paper



We can approach linear compute scaling, while reducing the HDD count, if we move the YARN data to an NVMe SSD.

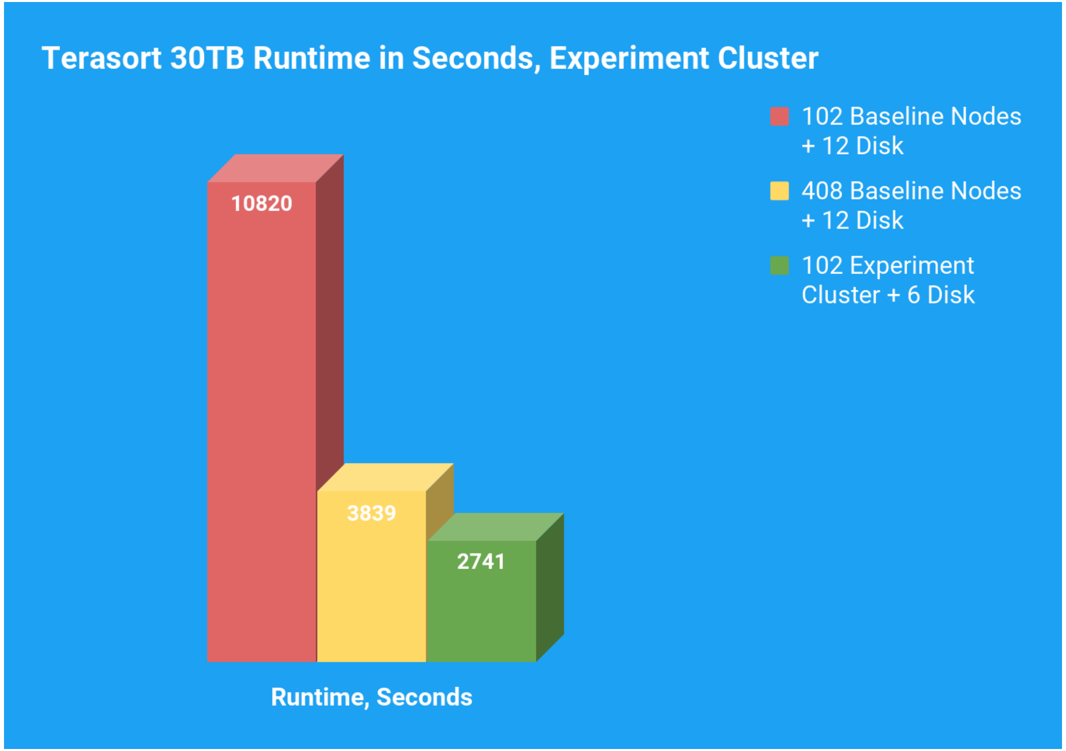
Primer on Hadoop Benchmarks

- Terasort
  - **Synthetic** benchmark that creates large numbers of rows of random data and then sorts that data.
  - Regular compute pattern. Heavier on I/O and network.
- Gridmix
  - Capture **production workload** traces
  - Replay traces with synthetic input to simulate production
  - The “gold standard” we use for testing Hadoop.

Non-Linear Scaling

Tested scaling **non-cached** cluster with 4X nodes. Runtime change: 10820/3839 = 2.81X was disappointing
Below Linear scaling

Tested cluster **with caching** from the white paper, with the same node count. Runtime change: 10820/2741 = 3.95X
Linear Scaling



Results are based on the systems described in the paper.

Test Cluster Comparison

Based on total compute power, we expect that each new server can do 4X the work with many fewer HDDs per Thread in the cluster.

	Baseline	New Config
Servers	~140	~140
Threads/Server	8	48
Core Speed	3.5GHz	2.5Ghz
Compute Power	1X	4X
HDD/Server	12	8
Total Threads	1,120	6720
Total Spindles	1,680	1,120
HDDs per Thread	1.5	0.17
YARN	HDD	NVMe SSD

How We Selected the SSD

We used eBPF to monitor disk writes into production clusters and existing observability data to monitor the size of the data.

From this we can estimate size and endurance needs.

Metric (Over 1 Month)	Production Cluster	6X Estimate
Avg YARN Writes (per node)	~1.5 TBW (Per Day)	~16 PBW (5 Years)
Max Yarn Data observed (p100)	~1.75 TB	~10 TB
p99.9 Yarn Data Size	~400 GB	~2.4TB

From this, we can select a drive that will meet our endurance needs, and have enough space to handle the p99.9 case for sizing. For this project, we selected the Intel P4610 6.4TB NVMe SSD.

Hadoop YARN Data Storage

Most Hadoop clusters store their YARN (jobs) temporary data on the same HDDs as HDFS data because they cannot make an a priori estimation of max YARN space.

- Jobs that run out of YARN space die, are rescheduled, and just die over and over.

So we don't want to simply dedicate the SSD to YARN.

- Only a small number of jobs will exceed the size of a SSD that meets specs for factors such as endurance, speed, and price. But we still need a solution for those jobs...

Our primary objective is to optimize spending on the flash.

Hybrid Caching Approach

So rather than having separate flash and HDD we want to use HDDs to store **both** HDFS data **and** spill of YARN temporary data.

We used an Intel supported OpenCAS Linux build to enable this. CAS uses a **directory classifier** to selectively cache the YARN writes (and some metadata) so that HDFS activity doesn’t pollute the cache.

Old Design vs. New Design

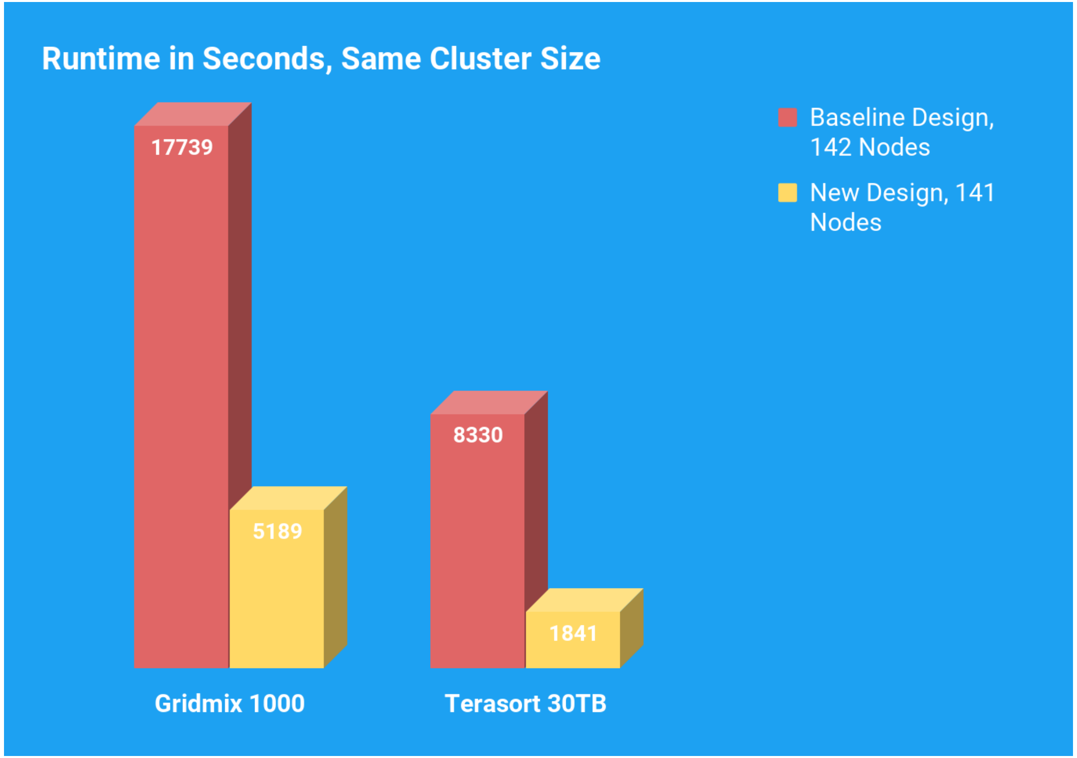
Gridmix 1000 Runtime:

17739/5189 = 3.42X

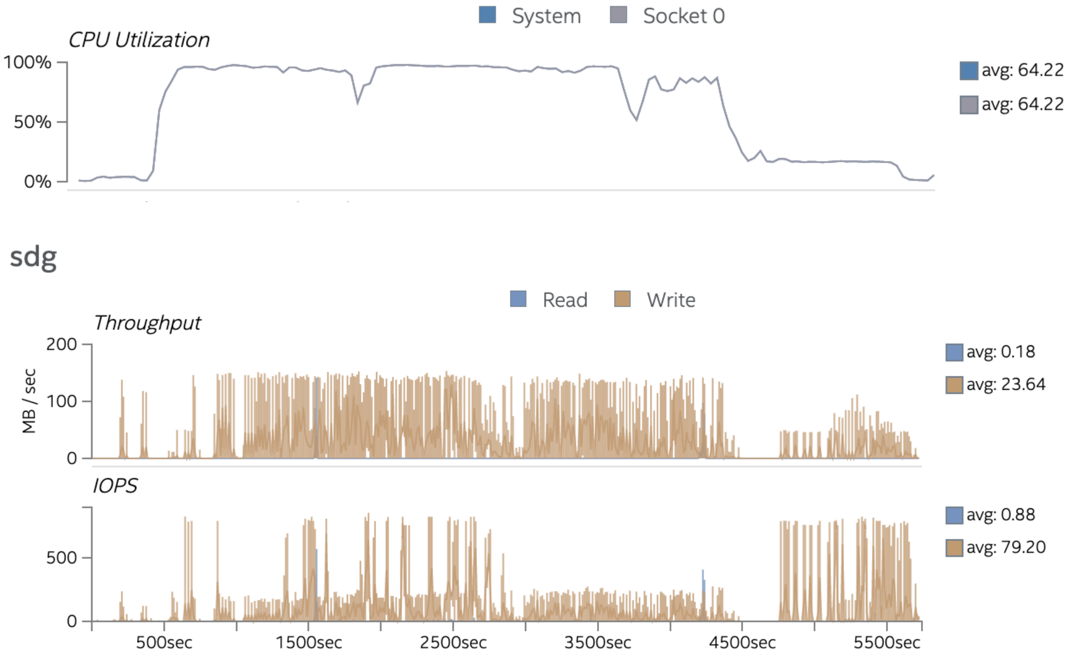
Terasort Runtime:

8330/1841 = 4.52X

We knew going into this that Terasort was more I/O bound and Gridmix was more CPU bound. ***But….***



Gridmix Loading, 1000 Jobs



New Configuration. Graphs from Intel vTune Amplifier Platform Profiler

The cluster wasn’t fully loading.

Rather long tail of low CPU utilization at each end.

Also, rather low throughput on HDD (24MB/s and average 80 IOPS)

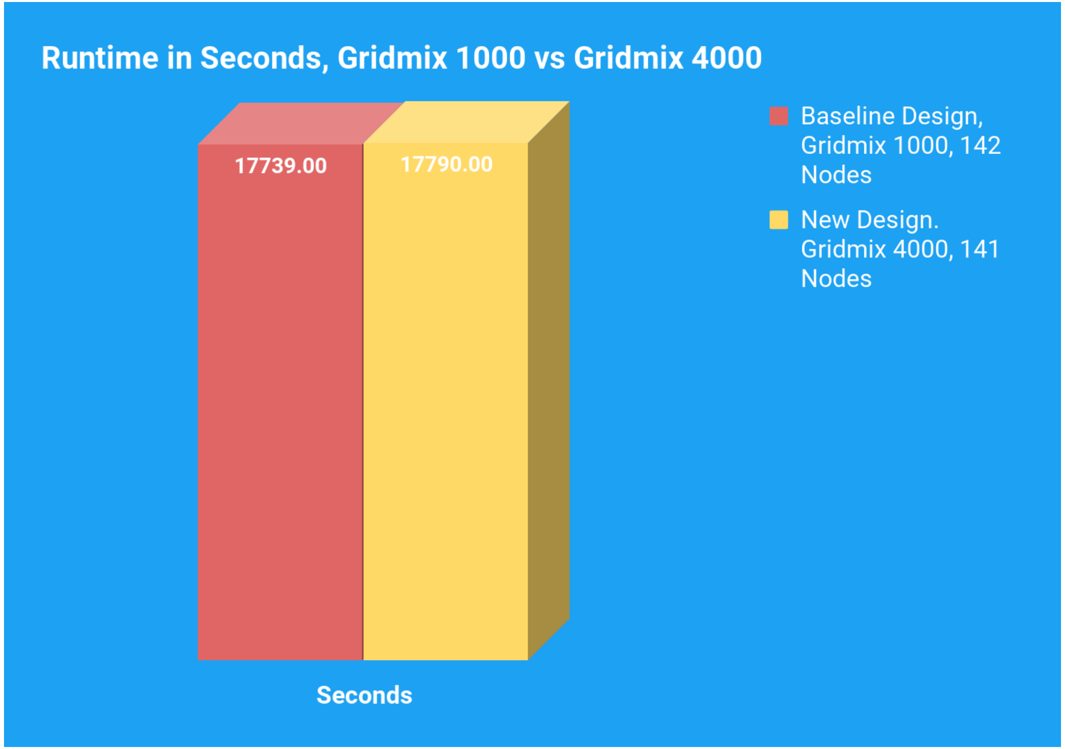
So, we switched our approach to loading with 4000 jobs (rather than 1000) to fully utilize the hardware.

Driving More Load with Gridmix

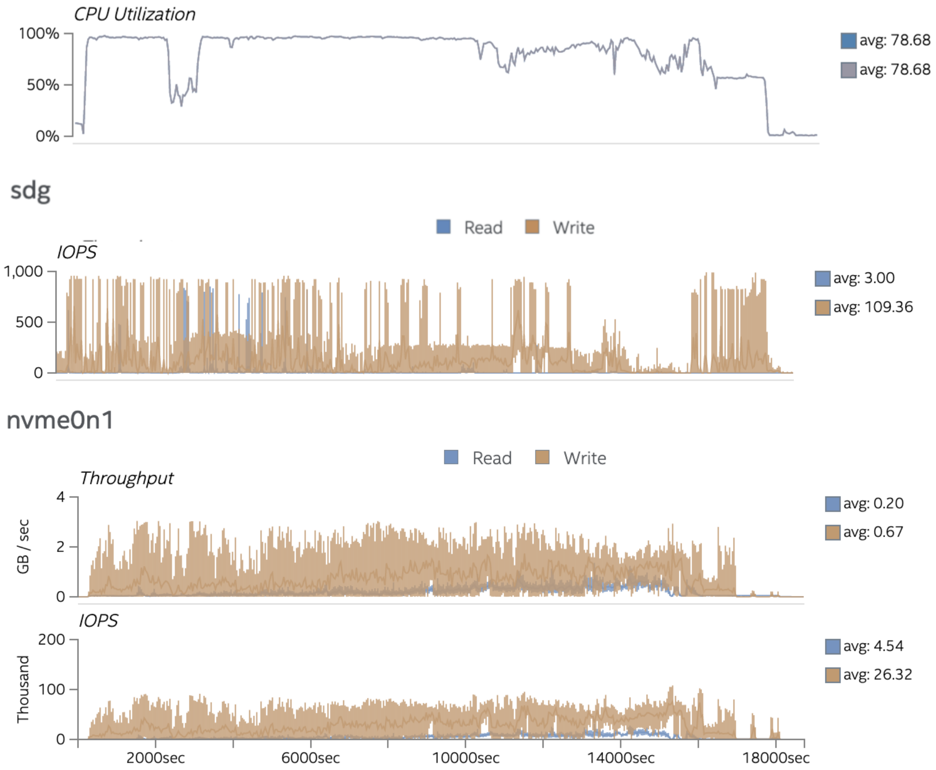
To drive more load, we decided to run equal node counts but place 4X the load on the new design.

Gridmix 1000 jobs vs Gridmix 4000 jobs Runtimes are 49 seconds different.

Adjusting for the 1 node difference, this is 4.01X



Gridmix Loading, 4000 Jobs



CPU Utilization averaged 78% (Was 64%)

HDDs averaged ~110 IOPS (Was 80)

But the most distinct thing you can see is that the NVMe SSD averaged a combined 30K IOPS. (or ~100 HDDs worth of IOPS)

New Configuration. Graphs from Intel vTune Amplifier Platform Profiler

TCO Impacts

Yes, every server gets more expensive due to having more CPU / DRAM / Flash.

But we'll end up buying ~75% less units.

And we'll have 75% lower OPEX because of that. (Space / Power / Maintenance)

TCO Impacts

At cluster scale, 75% fewer:

- Motherboards
- Power supplies
- NICs
- Switches

This can actually reduce the CAPEX of the cluster.

The major spend components are similar or better:

- Cost of CPU core
- Cost of DRAM/GB

Bigger HDDs are less expensive per capacity unit.

TCO Impacts

Let’s assume that your OPEX$ is on the same order as your CAPEX$ over some lifespan (unless you’re buying really, really expensive servers.)

If OPEX is 50% of your TCO spend, and if you cut OPEX by 75% you’ve saved 37.5% of TCO.

But you can probably reduce the CAPEX$ as well for even greater savings.

Challenges During Testing

- Needed a new strategy for load testing
  - From: Expecting ¼ the runtime
  - To: Run 4X workload and expect same runtime.
- Software integration issues with CAS.
- Configuring Hadoop for the new compute : storage : memory ratios
- Internal software interfering with metrics collections with Intel VTune Platform Profiler.

So What Did We Get?

- 1 new server can replace 4 old servers
  - Using the NVMe storage for YARN
- No (real) hard ceiling on YARN space
  - Using selective caching via Intel CAS
- 75% less servers to power and maintain.
- Big TCO savings!

Next Generation Musings

- Increase density again with more CPU cores in each machine.
- QLC Flash
  - Interesting opportunity to eliminate HDDs for very hot clusters.
- Coarse indirection flash. Will there be a write amplification issue with these loads?

Acknowledgments

- We collaborated with an extensive team at Intel spanning the hardware & software components of this project.
- Our management and colleagues that supported us:

Thank you!

Questions?

Thank you!