Scaling Hadoop at LinkedIn

Konstantin V Shvachko

Sr. Staff Software Engineer

Zhe Zhang

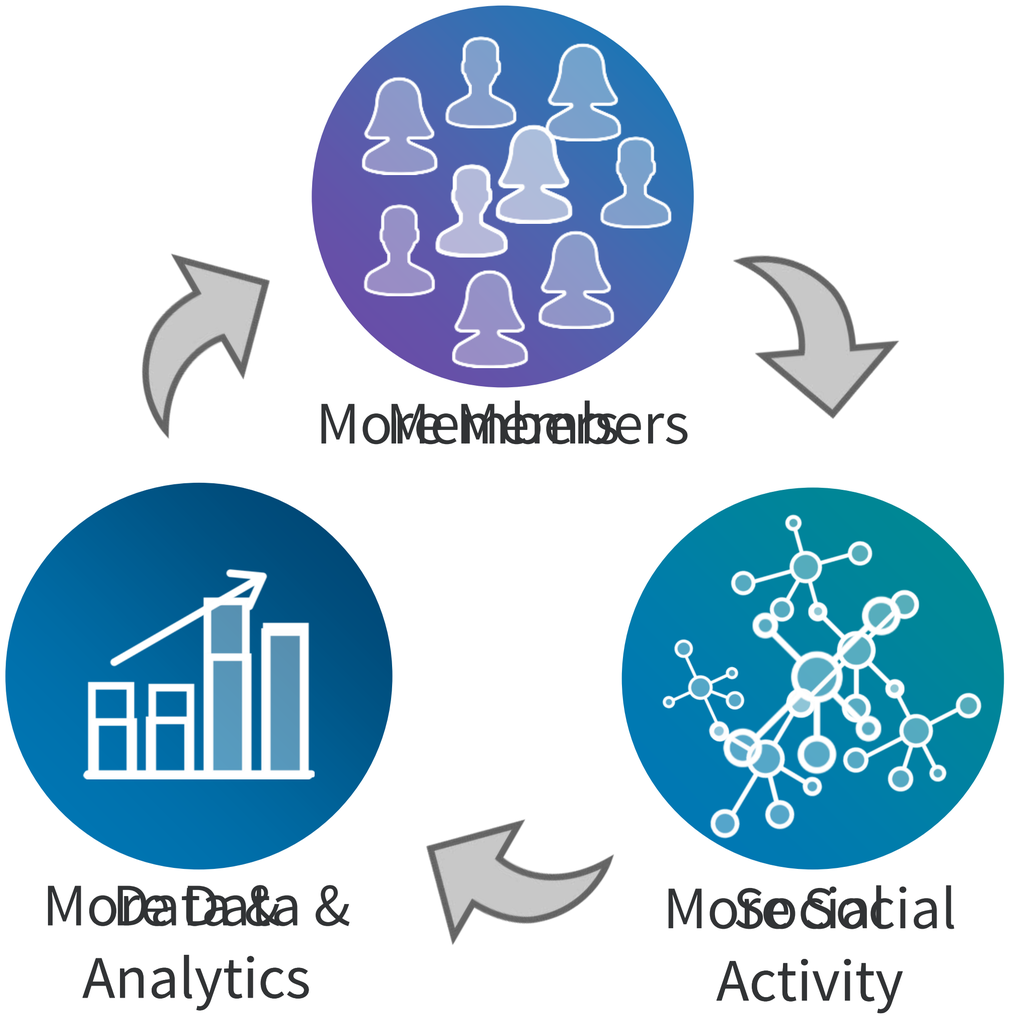
Engineering Manager

Erik Krogen

Senior Software Engineer

Continuous Scalability as a Service

LINKEDIN RUNS ITS BIG DATA ANALYTICS ON HADOOP



Hadoop Infrastructure

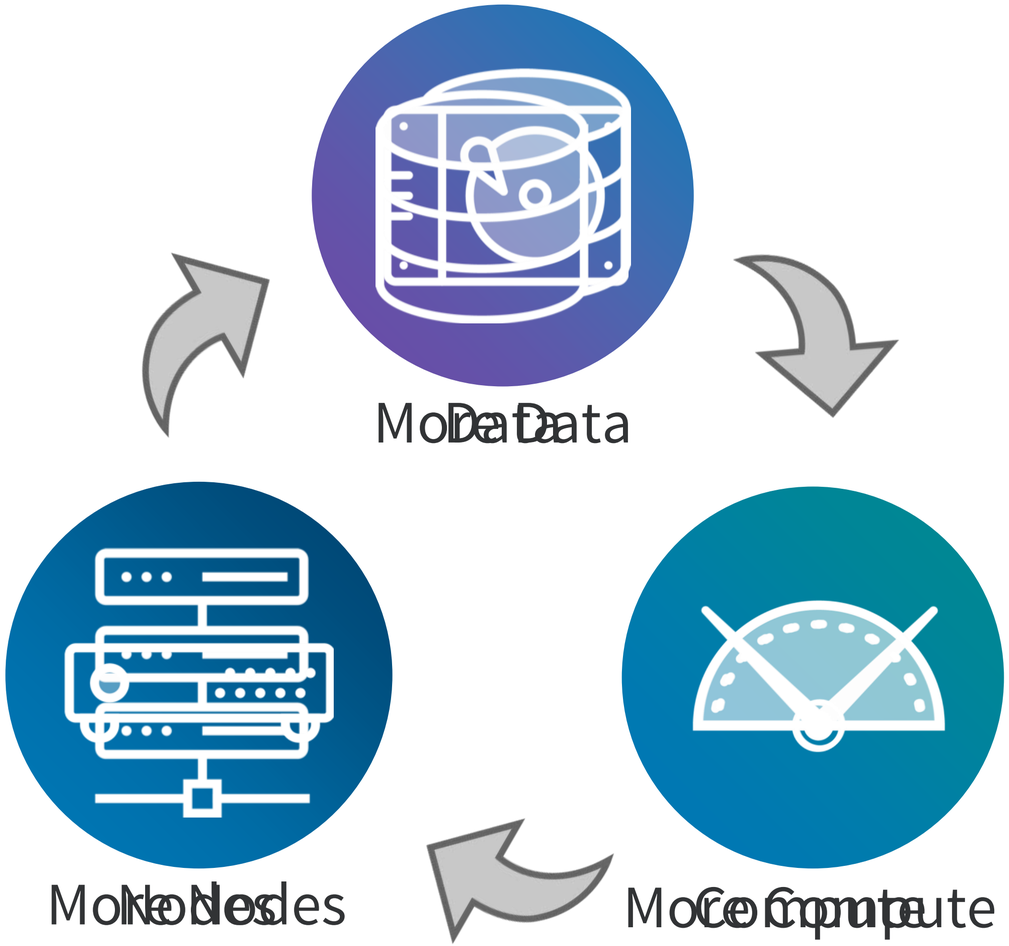
ECOSYSTEM OF TOOLS FOR LINKEDIN ANALYTICS

- 
  - Storage: HDFS
  - Compute: YARN and MapReduce
-   
-   **TensorFlow**
-  – workflow scheduler

- **Dali** – data abstraction and access layer for Hadoop
- ETL: 
-  Dr. Elephant – artificially intelligent, polite, but uncompromising bot
- presto – distributed SQL engine for interactive analytic

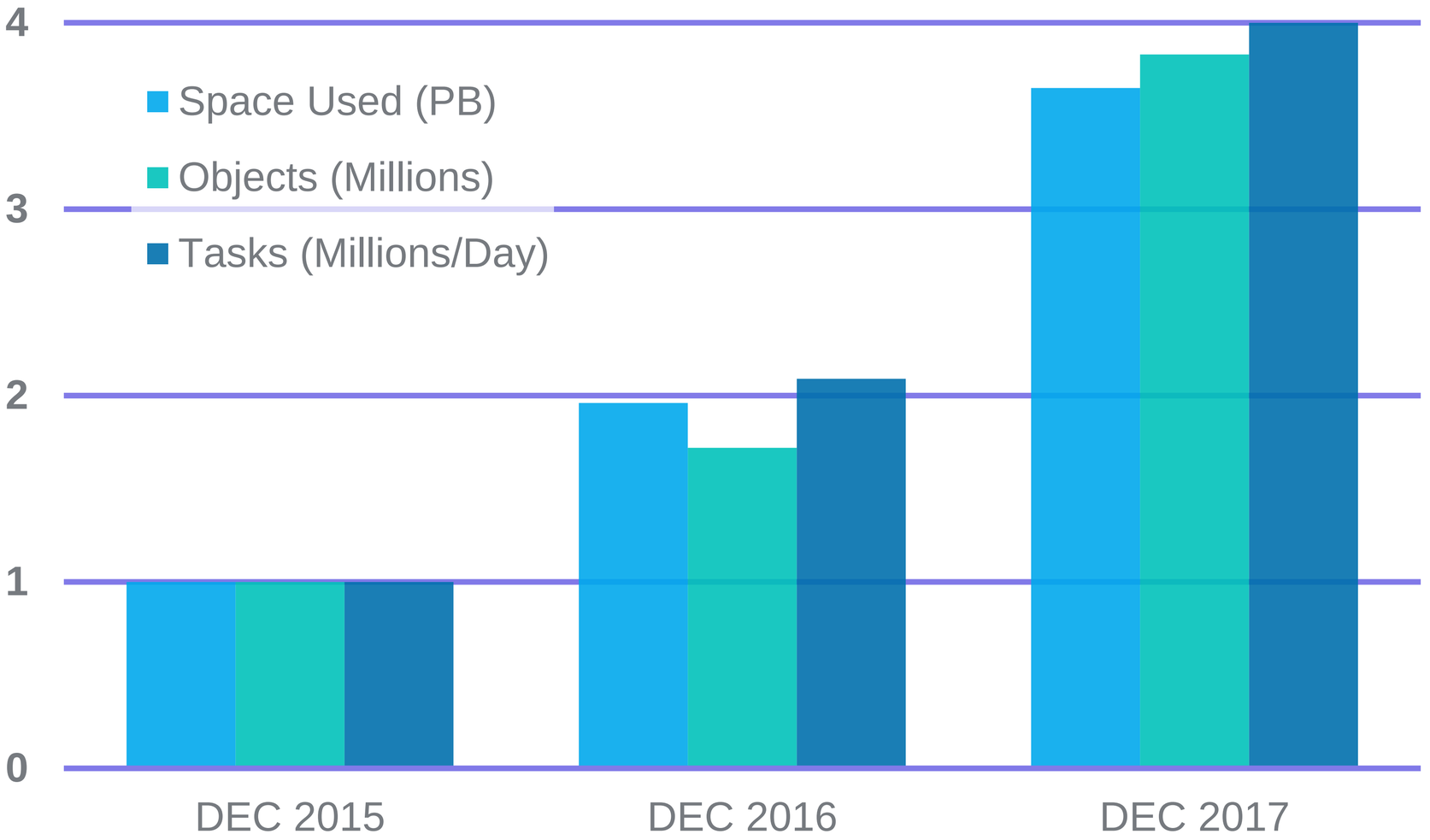
Growth Spiral

THE INFRASTRUCTURE IS UNDER CONSTANT GROWTH PRESSURE



Cluster Growth 2015 - 2017

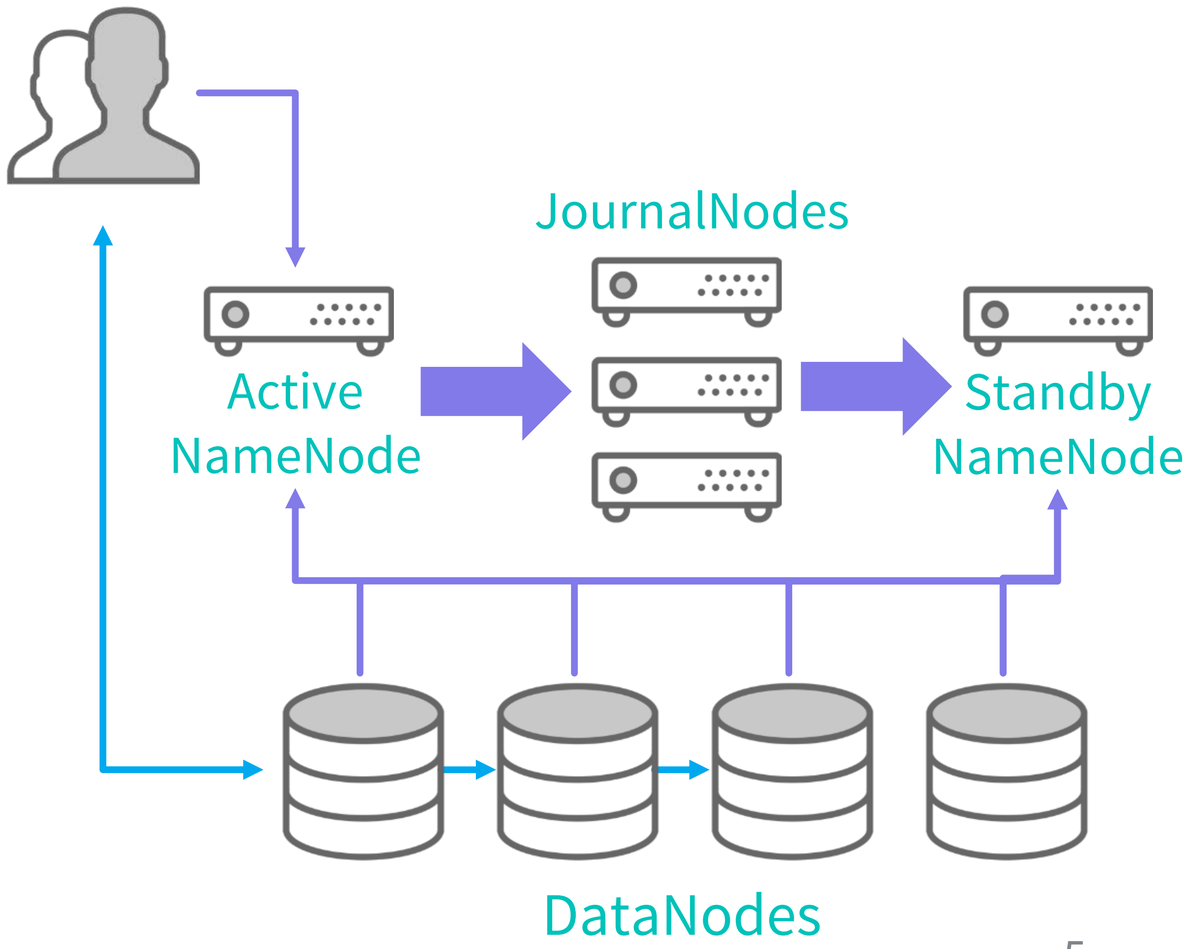
THE INFRASTRUCTURE IS UNDER CONSTANT GROWTH PRESSURE



HDFS Cluster

STANDARD HDFS ARCHITECTURE

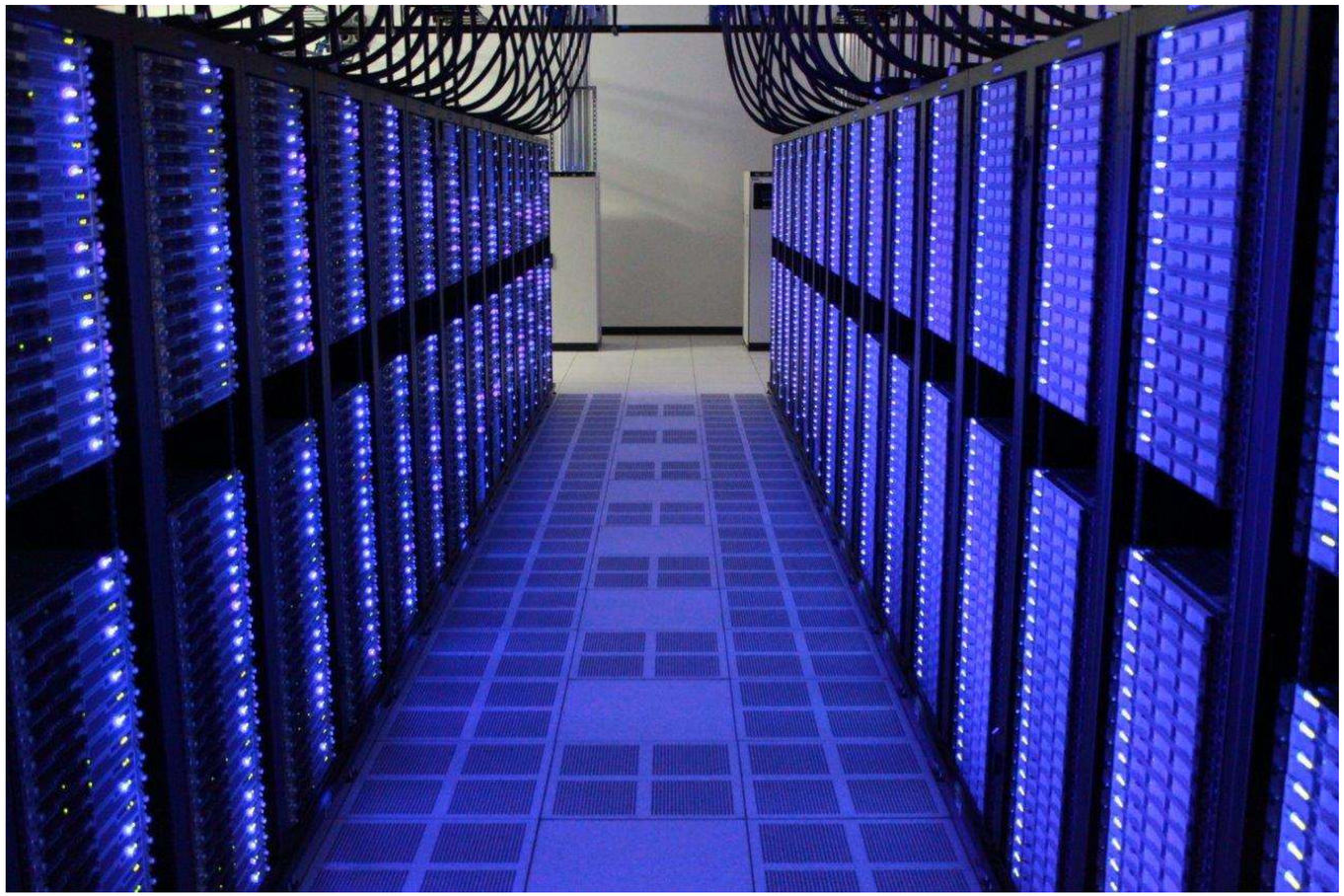
- HDFS metadata is decoupled from data
- **NameNode** keeps the directory tree in RAM
- Thousands of **DataNodes** store data blocks
- **HDFS clients** request metadata from active NameNode and stream data to/from DataNodes



Cluster Heterogeneity & Balancing

Homogeneous Hardware

ELUSIVE STARTING POINT OF A CLUSTER



Heterogeneous Hardware

PERIODIC CLUSTER EXPANSION ADDS A VARIETY OF HARDWARE

Balancer

MAINTAINS EVEN DISTRIBUTION OF DATA AMONG DATANODES

- DataNodes should be filled uniformly by % used space
  - Locality principle: more data => more tasks => more data generated

DataNodes usages% (Min/Median/Max/stdDev):	12.36% / 82.89% / 90.91% / 3.75%

- Balancer iterates until the balancing target is achieved
  - Each iteration moves some blocks from overutilized to underutilized nodes
  - Highly multithreaded. Spawns many dispatcher threads. In a thread:
    - getBlocks() returns a list of blocks of total size S to move out of sourceDN
    - Choose targetDN
    - Schedule transfers from sourceDN to targetDN

Balancer Optimizations

BALANCER STARTUP CAUSES JOB TIMEOUT

HDFS-11384

- **Problem 1**. At the start of each Balancer iteration threads hit NameNode at once
  - Increase RPC CallQueue length => user jobs timeout
  - Solution. Disperse Balancer calls at startup over 10 sec period. Restricts the number of RPC calls from Balancer to NameNode to 20 calls per second

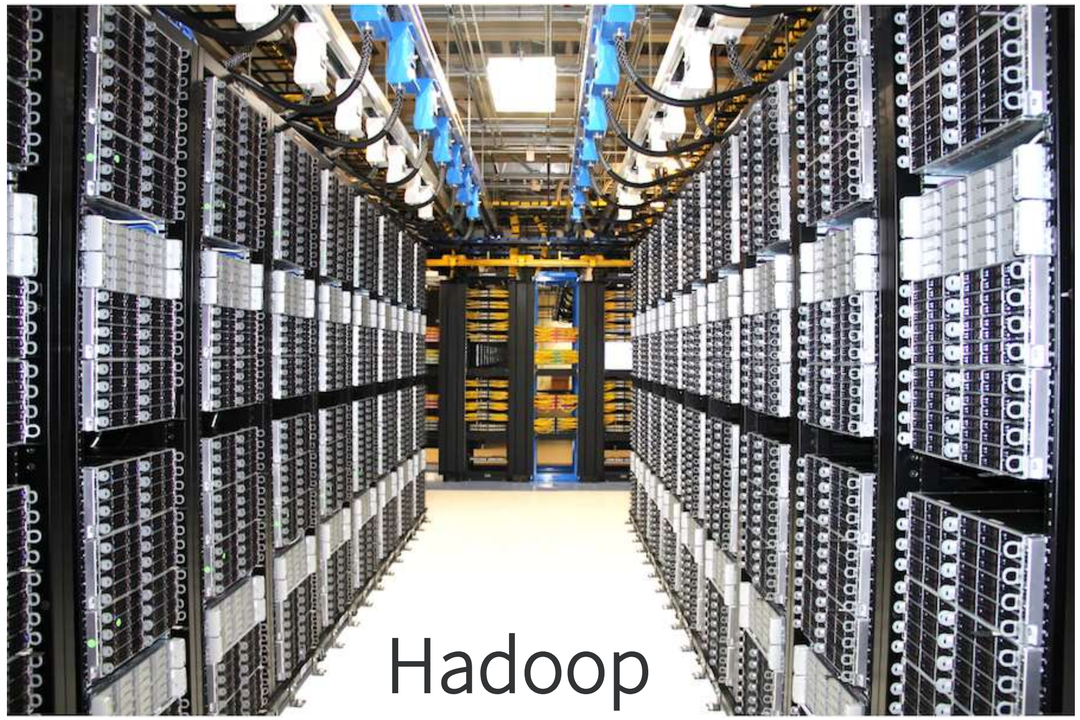
- **Problem 2**. Inefficient block iterator in getBlocks() HDFS-11634
  - NameNode iterates blocks from a randomly selected startBlock index.
  - Scans all blocks from the beginning, instead of jumping to startBlock position.
  - 4x reduction in exec time for getBlocks() from 40ms down to 9ms

- **Result**: Balancer overhead on NameNode performance is negligible

Block Report Processing

POP QUIZ

What do Hadoop clusters and particle colliders have in common?

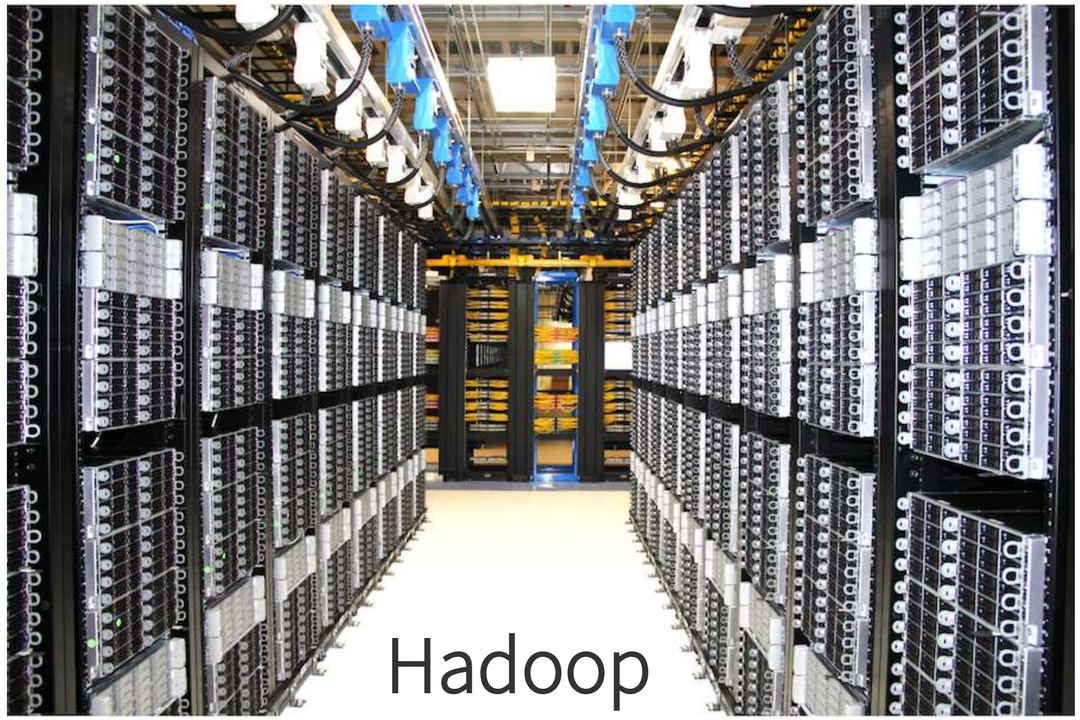


Hadoop

Large Hadron Particle Collider

POP QUIZ

What do Hadoop clusters and particle colliders have in common?



Hadoop

Large Hadron Particle Collider

Improbable events happen all the time!

Optimization of Block Report Processing

DATANODES REPORT BLOCKS TO THE NAMENODE VIA BLOCK REPORTS

- DataNodes send periodic block reports to the NameNode (6 hours)
  - Block report lists **all the block replicas** on the DataNode
  - The list contains block id, generation stamp and length for each replica
- Found a **rare race condition** in processing block reports on the NameNode
  - Race with repeated reports from the same node
  - The error recovers itself on the next successful report after six hours
- HDFS-10301, fixed the bug and simplified the processing logic
- Block reports are expensive as they **hold the global lock for a long time**
  - Designed segmented block report proposal HDFS-11313

Cluster Versioning

Upgrade to Hadoop 2.7

COMMUNITY RELEASE PROCESS

- Motivation: chose 2.7 as the most stable branch compared to 2.8, 2.9, 3.0
  - The team contributed majority of commits to 2.7 since 2.7.3
- Release process
  - Worked with the community to lead release
  - Apache Hadoop release v 2.7.4 (Aug 2017)
- Testing, performance benchmarks
  - Community testing: automated and manual testing tools
  - Apache BigTop integration and testing
  - Performance testing with Dynamometer
  - Benchmarks: TestDFSIO, Slive, GridMix

Upgrade to Hadoop 2.7

INTEGRATION INTO LINKEDIN ENVIRONMENT

- Comprehensive testing plan
  - Rolling upgrade
  - Balancer
  - OrgQueue
  - Per-component testing
    - Pig, Hive, Spark, Presto
    - Azkaban, Dr. Elephant, Gobblin
  - Production Jobs

- **What went wrong?**
- DDOS of InGraphs with new metrics
  - Introduced new metrics turned ON by default
  - Large scale required to cause the issue

Satellite Cluster Project

Small File Problem

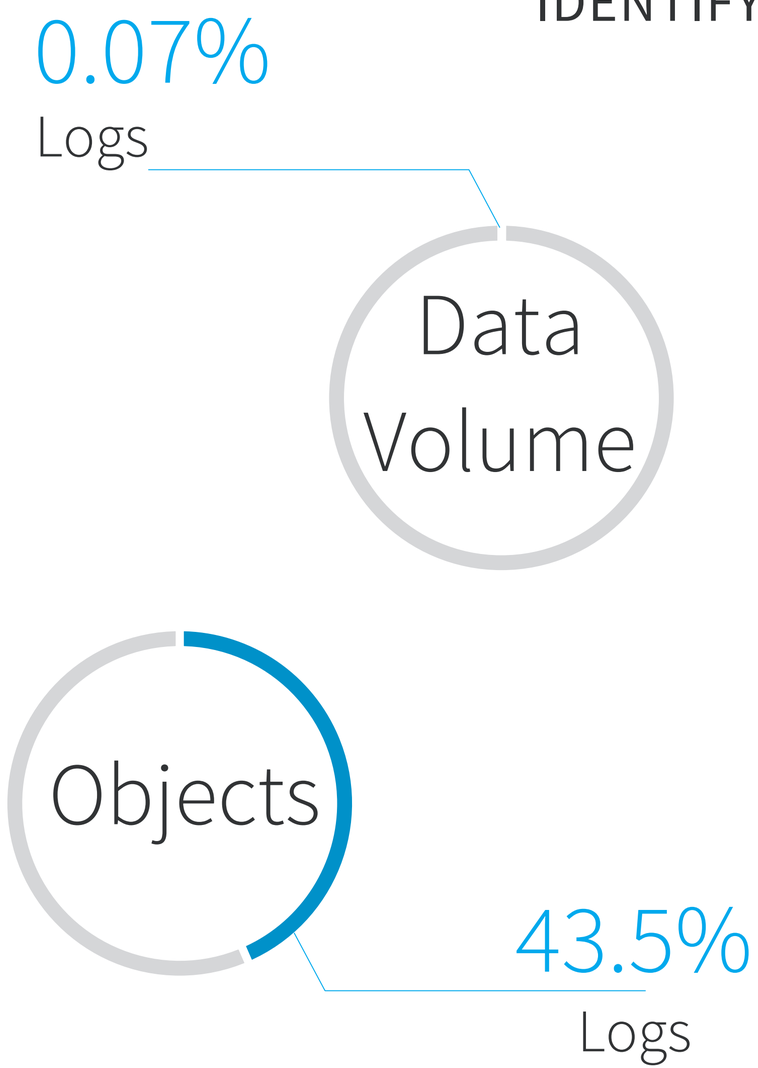
“Keeping the mice away from the elephants”

Small File Problem

- “Small file” is **less than one block**
  - Each requires at least two objects: block & inode
- Small files **bloat the memory usage** of NameNode & lead to **numerous RPC calls** to NameNode
- Block-to-inode ratio **steadily decreasing**; now at 1.11
  - **90% of our files are small!**

Satellite Cluster Project

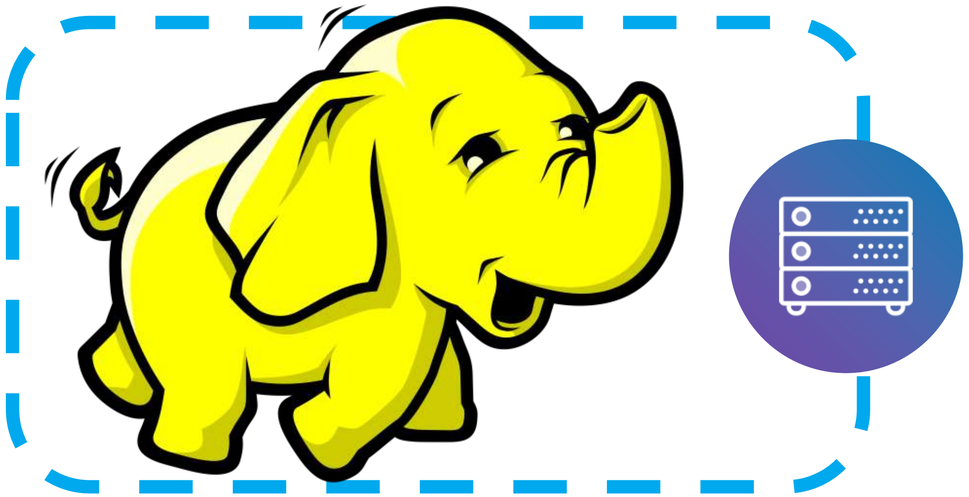
IDENTIFY THE MICE: SYSTEM LOGS

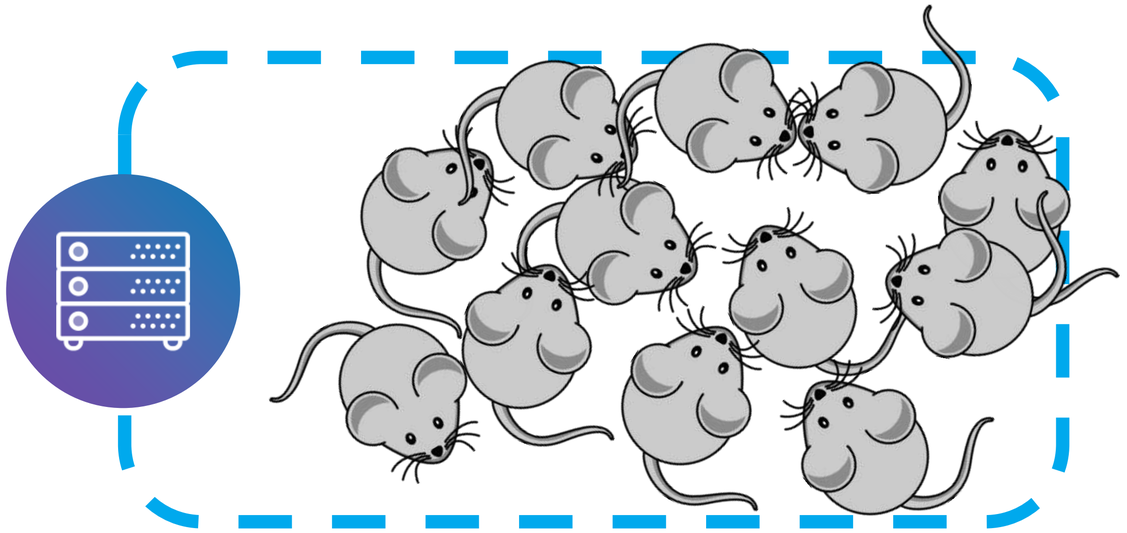


- Realization: many of these small files were logs (YARN, MR, Spark…)
- Average size of log files: **100KB**!
- Only accessed by **framework/system daemons**
  - NodeManager
  - MapReduce / Spark AppMaster
  - MapReduce / Spark History Server
  - Dr. Elephant

Satellite Cluster Project

GIVE THE MICE THEIR OWN CLUSTER





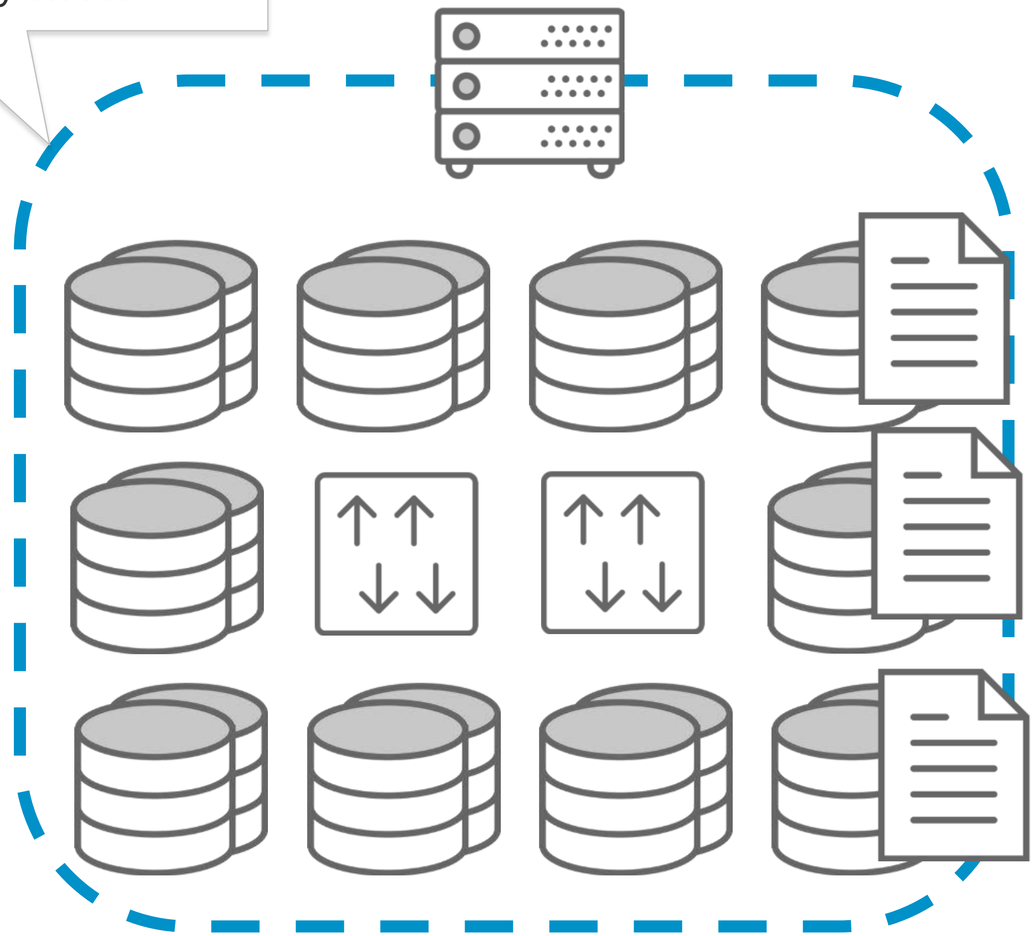
- **Two separate clusters**, one ~100x more nodes than the other
- Operational challenge: how to bootstrap? **> 100M files to copy**
- Write new logs to new cluster
- Custom FileSystem presents **combined read view** of both clusters' files
- Log retention policy eventually deletes all files on old cluster

* Additional details in appendix

Satellite Cluster Project

ARCHITECTURE

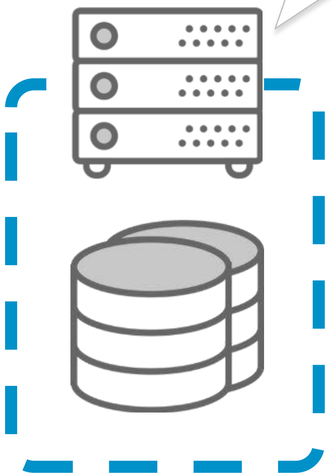
Bulk data transfer stays internal to the primary cluster



Primary Cluster

Log Files

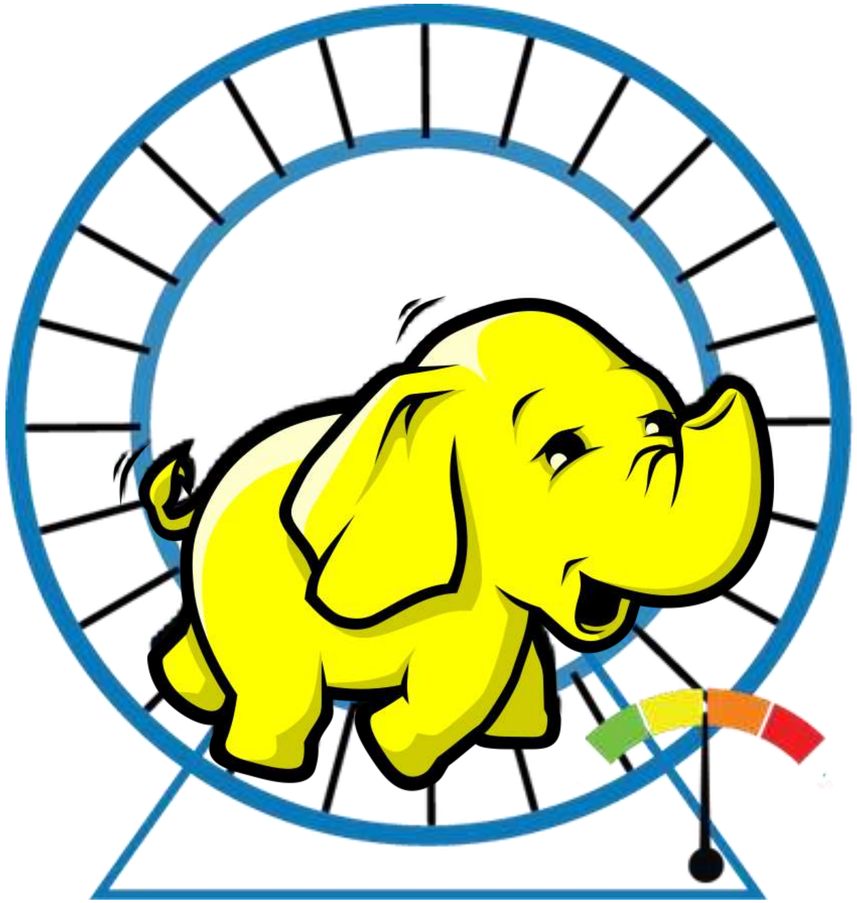
Same namespace capacity (one active NameNode) but much cheaper due to smaller data capacity (fewer DataNodes)



Satellite Cluster

Testing: Dynamometer

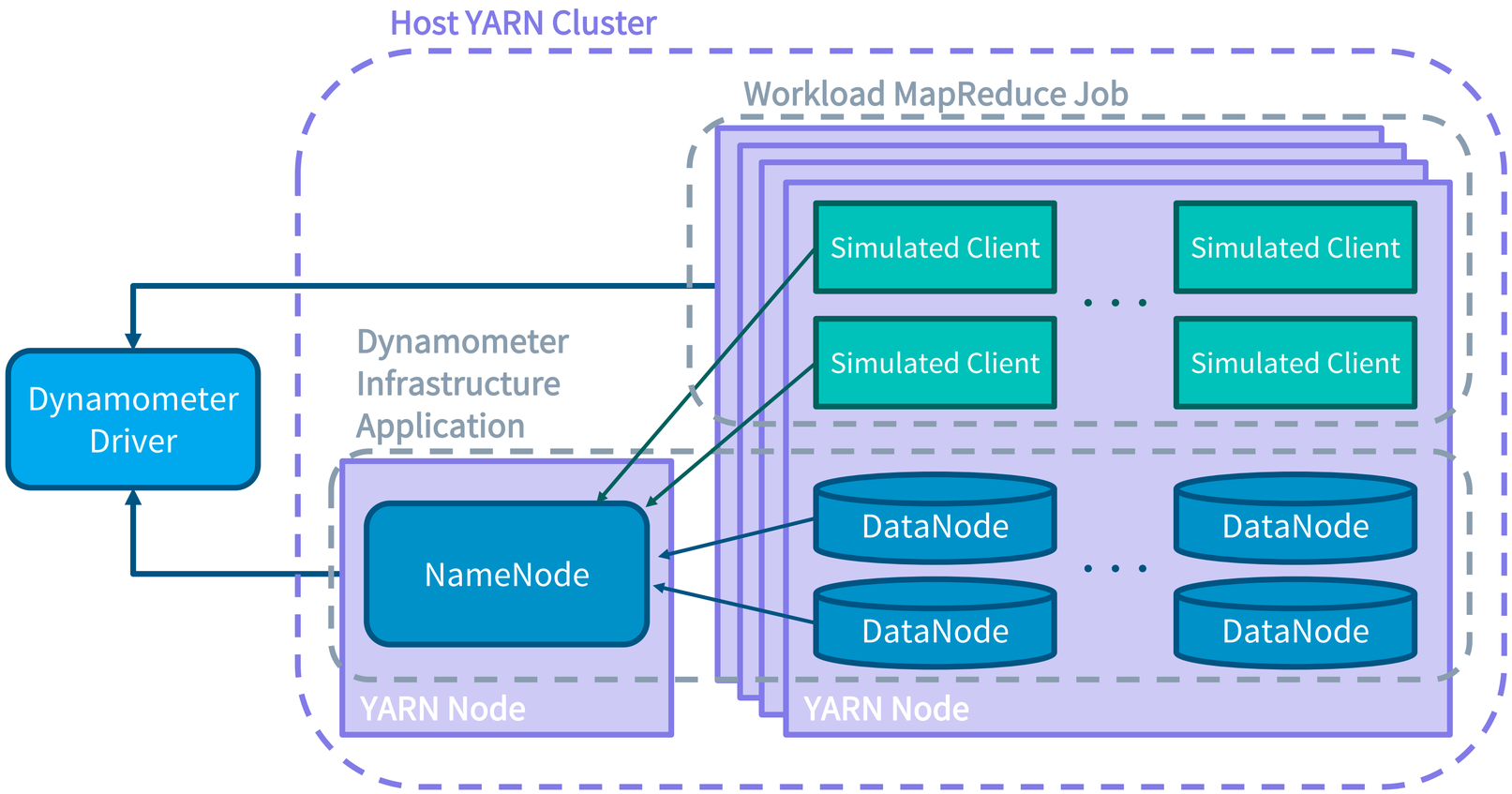
Dynamometer



- Realistic performance benchmark & stress test for HDFS
- Open sourced on LinkedIn GitHub, hope to contribute to Apache
- Evaluate **scalability limits**
- Provide **confidence** before new feature/config deployment

Dynamometer

SIMULATED HDFS CLUSTER RUNS ON YARN



- Real NameNode, fake DataNodes to run on **~5% the hardware**
- Replay **real traces** from production cluster audit logs

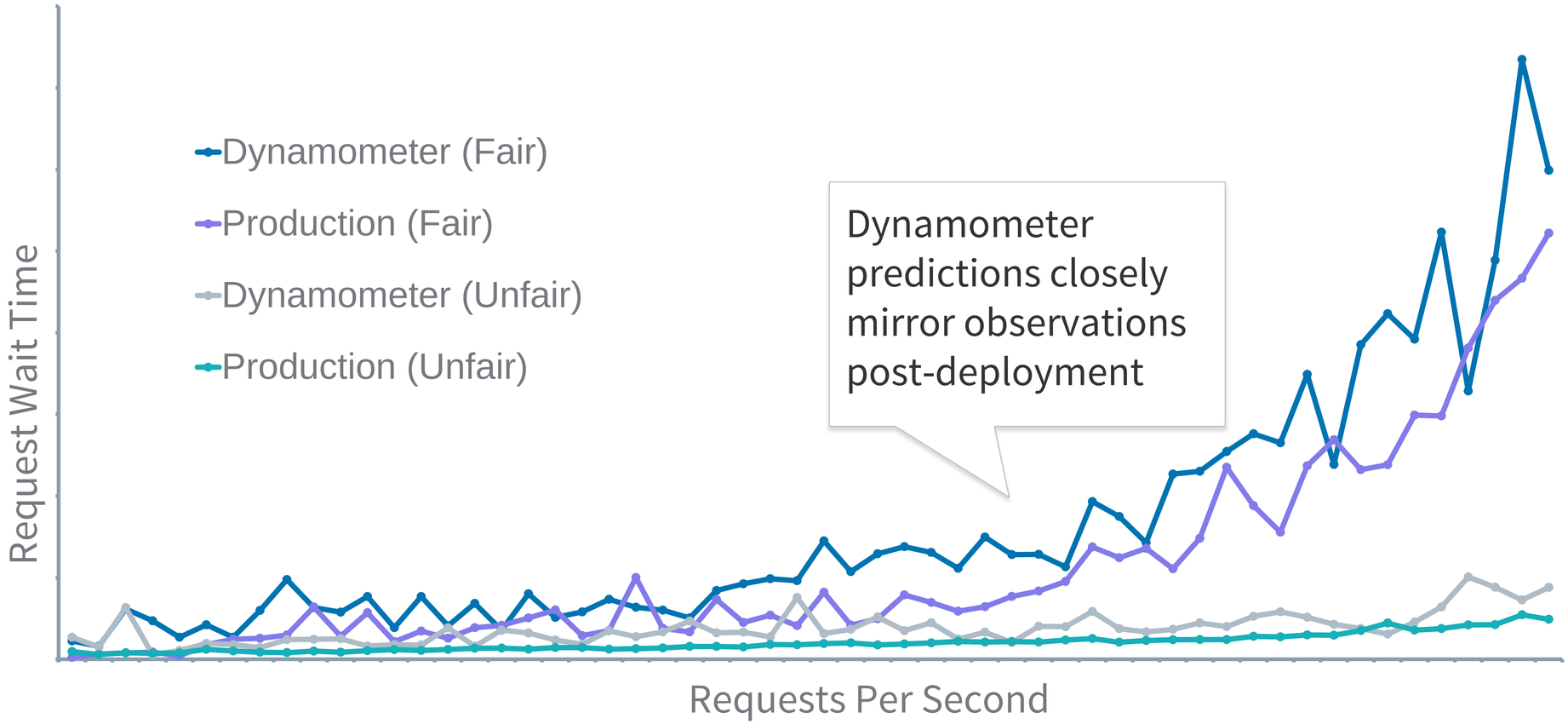
Namespace Locking

Nonfair Locking

- NameNode uses a **global read-write lock** which supports two modes:
  - **Fair**: locks are acquired in FIFO order (HDFS default)
  - **Nonfair**: locks can be acquired out of order
- Nonfair locking discriminates writes, but benefits reads via **increased read parallelism**
- NameNode operations are **> 95% reads**

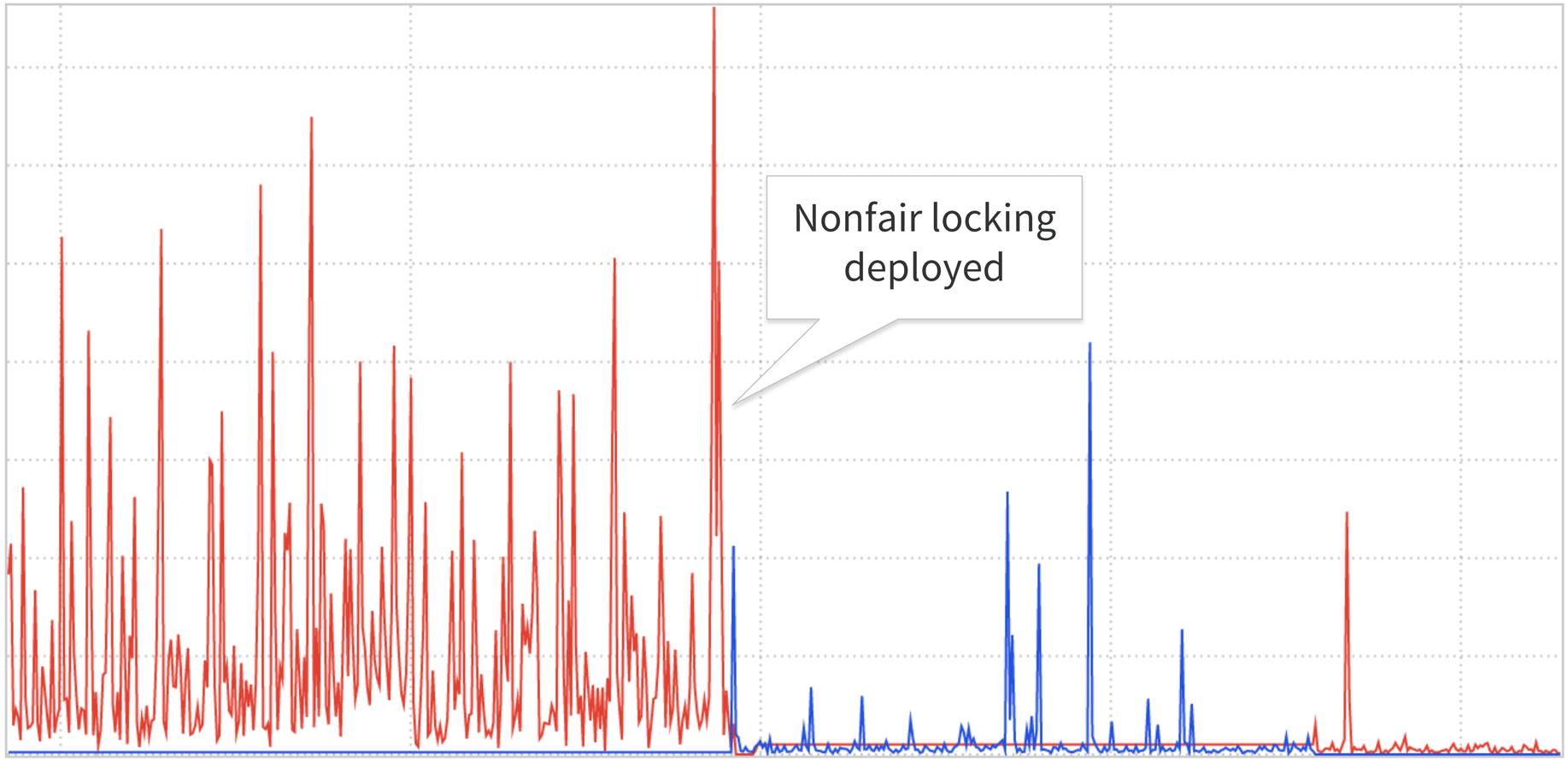
Dynamometer

EXAMPLE: PREDICTION OF FAIR VS NONFAIR NAMENODE LOCKING PERFORMANCE



Nonfair Locking

IN ACTION



RPCQueueTimeAvgTime

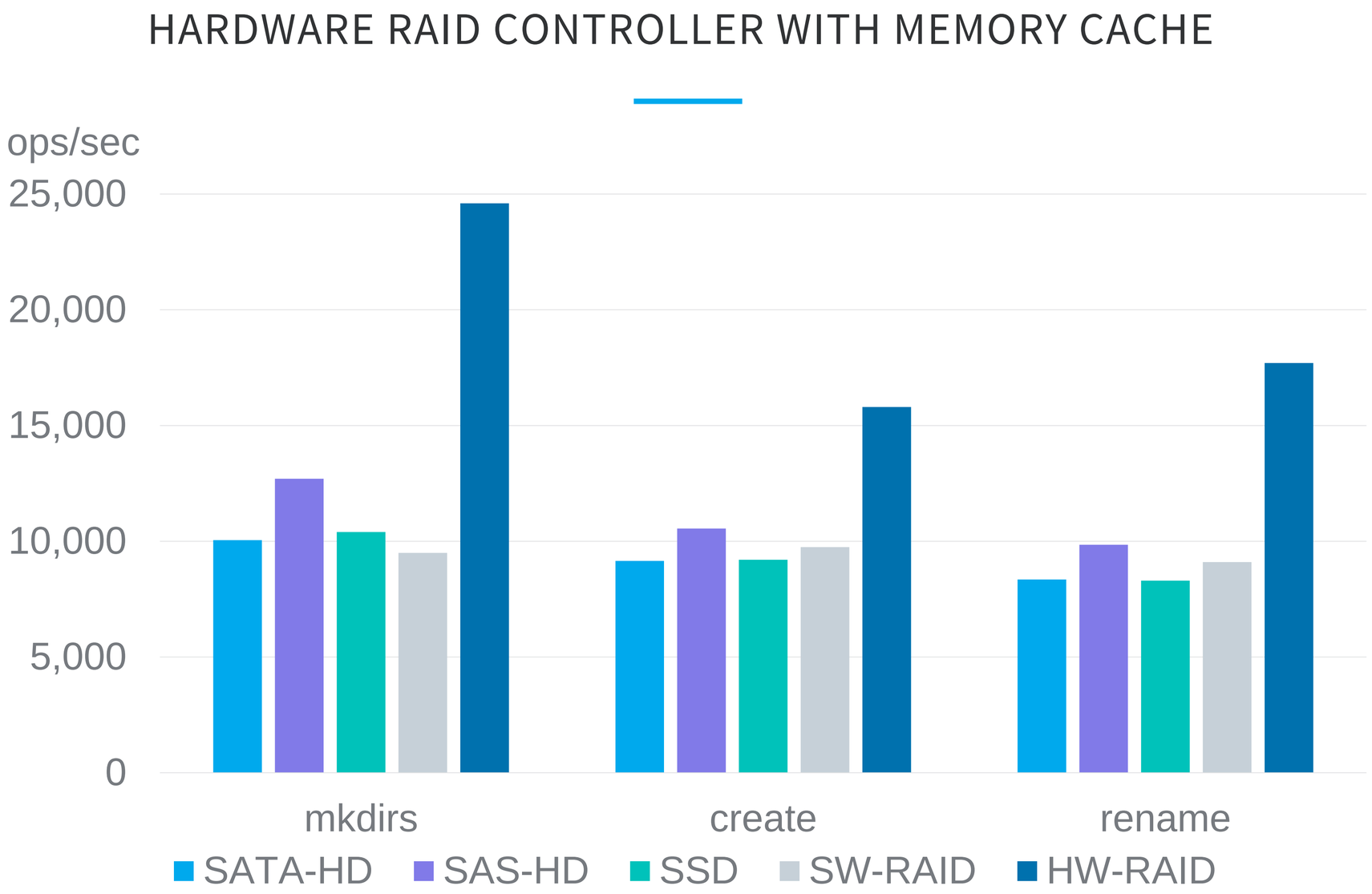
Optimal Journaling

Optimal Journaling Device

BENCHMARK METHODOLOGY

- Persistent state of NameNode
  - Latest checkpoint FSImage. Periodic 8 hour intervals
  - Journal EditsLog – latest updates to the namespace
- Journal IO workload
  - Sequential writes of 1KB chunks, no reads
  - Flush and sync to disk **after every write**
- **NNThroughputBenchmark** tuned for efficient use of CPU
  - Ensure throughput is **bottlenecked mostly by IOs** while NameNode is journaling
  - Operations: mkdir, create, rename

Optimal Journaling Device



Optimal Journaling Device

SUMMARY

- SATA vs SAS vs SSD vs software RAID vs hardware RAID
- SAS is 15-25% better than SATA
- SSD is on par with SATA
- SW-RAID doesn’t improve performance compared to single SATA drive
- **HW-RAID provides 2x performance gain vs SATA drives**

Open Source Community

Open Source Community

KEEPING INTERNAL BRANCH IN SYNC WITH UPSTREAM

- The commit rule:
  - Backport to all upstream branches before internal
  - Ensures future upgrades will have all historical changes
- Release management: 2.7.4, 2.7.5, 2.7.6
- Open sourcing of OrgQueue with Hortonworks
- 2.9+ GPU support with Microsoft
- StandbyNode reads with Uber & PayPal

Next Steps

What's Next?

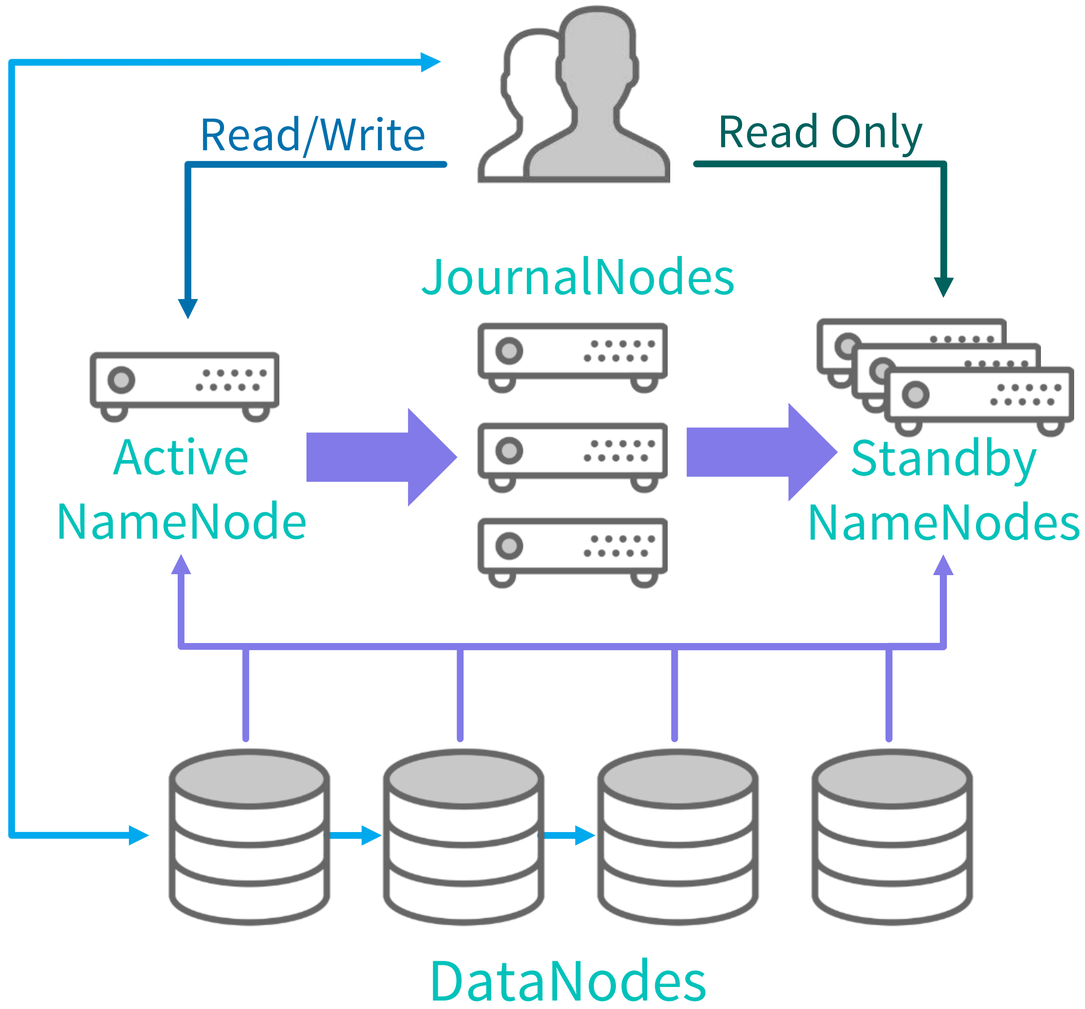
2X GROWTH IN 2018 IS IMMINENT

- **Stage I**. Consistent reads from standby — HDFS-12943
  - **Optimize for reads: 95% of all operations**
  - Consistent reading is a coordination problem
- **Stage II**. Eliminate NameNode's global lock — HDFS-10419
  - Implement namespace as a KV-store
- **Stage III**. Partitioned namespace
  - Linear scaling to accommodate increases RPC load

Consistent Reads from Standby Node

ARCHITECTURE

- **Stale Read Problem** HDFS-12943
  - Standby Node syncs edits from Active NameNode via Quorum Journal Manager
  - Standby state is always behind the Active
- **Consistent Reads Requirements**
  - Read your own writes
  - Third-party communication



Thank You!

Konstantin V Shvachko
Sr. Staff
Software Engineer

Zhe Zhang
Engineering
Manager

Erik Krogen
Senior
Software Engineer

LinkedIn

Appendix

Satellite Cluster Project

IMPLEMENTATION

- Two clusters *li-satellite-01* (thousands of nodes), *li-satellite-02* (32 nodes)
- *FailoverFileSystem* – transparent view of */system* directory during migration
  - Access *li-satellite-02* first. If not there go to *li-satellite-01*. *listStatus()* merges from both
- Configuration change for Hadoop-owned services/frameworks:
  - NodeManager, MapReduce / Spark AppMaster & History Server, Azkaban, Dr. Elephant

```
mapreduce.job.hdfs-servers =
    hdfs://li-satellite-02.grid.linkedin.com:9000
mapreduce.jobhistory.intermediate-done-dir =
    hdfs://li-satellite-02.grid.linkedin.com:9000/system/mr-history/intermediate
mapreduce.jobhistory.done-dir =
    hdfs://li-satellite-02.grid.linkedin.com:9000/system/mr-history/finished
yarn.nodemanager.remote-app-log-dir =
    hdfs://li-satellite-02.grid.linkedin.com:9000/system/app-logs
spark.eventLog.dir =
    hdfs://li-satellite-02.grid.linkedin.com:9000/system/spark-history
```

Satellite Cluster Project

CHALLENGES

- **Copy >100 million existing files** (< 100TB) from li-satellite-01 to li-satellite-02
  - Can take > 12 hours, but saturated NameNode before copying a single byte
- **Solution**: FailoverFileSystem created new history files on li-satellite-02
  - Removed /system from li-satellite-01 after log retention period passed
- **Very large block reports**
  - 32 DataNodes each holds 9 million block replicas (vs 200K on normal cluster)
  - Takes forever to process on NameNode; long lock hold times
- **Solution**: Virtual partitioning of each drive into 10 storages
  - With 6 drives, block report split into 60 storages per DataNode
  - 150K blocks per storage – good report size