Netflix Data Benchmark (NDBench)

Ioannis Papapanagiotou
Cloud Database Engineering

Databases and Caches

- >100M subscribers
- Running tens of thousands of nodes of
  a. Cassandra
  b. ElasticSearch
  c. EVCache (Memcache)
  d. Dynomite (Redis)

Requirements

- Single benchmark framework for all data stores
- Dynamically change the benchmark configuration while the test is running.
- Test a platform along with production microservices
- Be able to integrate with other platform cloud services
- Run for an infinite amount duration
  - Wanted to test failure scenarios
  - Test long running maintenance such as database repairs

What is NDBench?

Netflix Data Benchmark (NDBench) is a **Pluggable cloud-enabled** benchmarking tool that can be used across **any data store system**.

NDBench allows

- Run infinite horizon tests to identify potential memory leaks from long running processes
- Test out different failure scenarios in TEST
- Performance test heavy processes like repairs, compaction, backup/restores

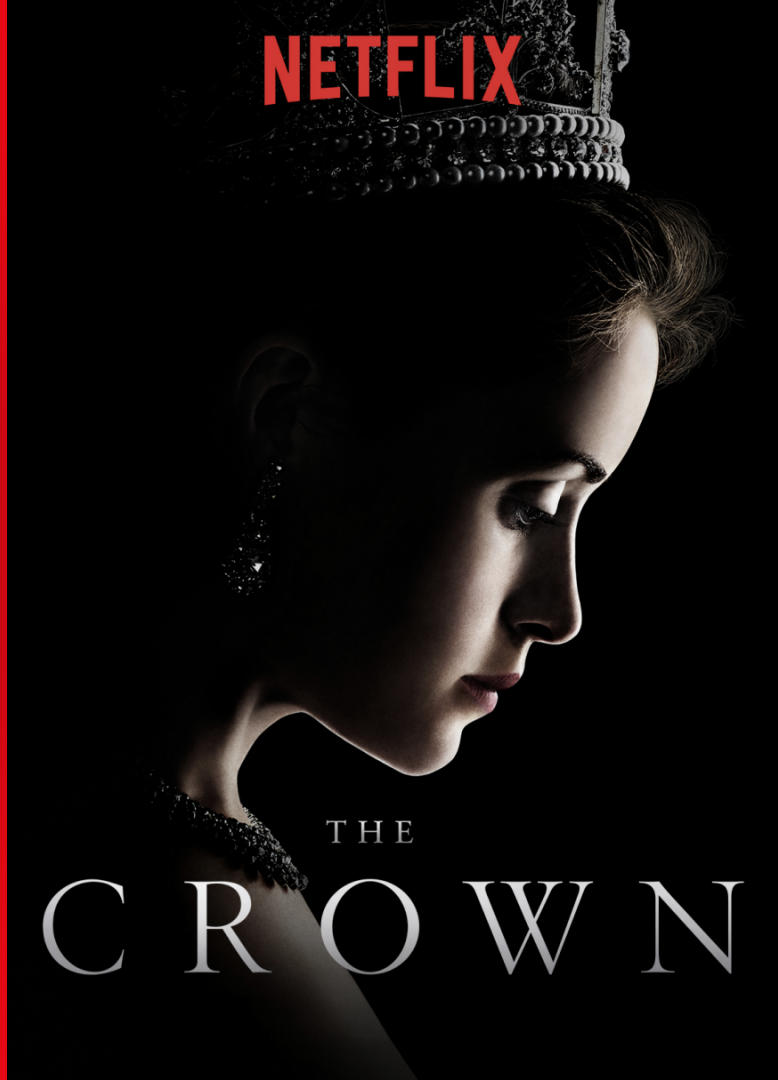
NDBench integration

Netflix OSS integration:

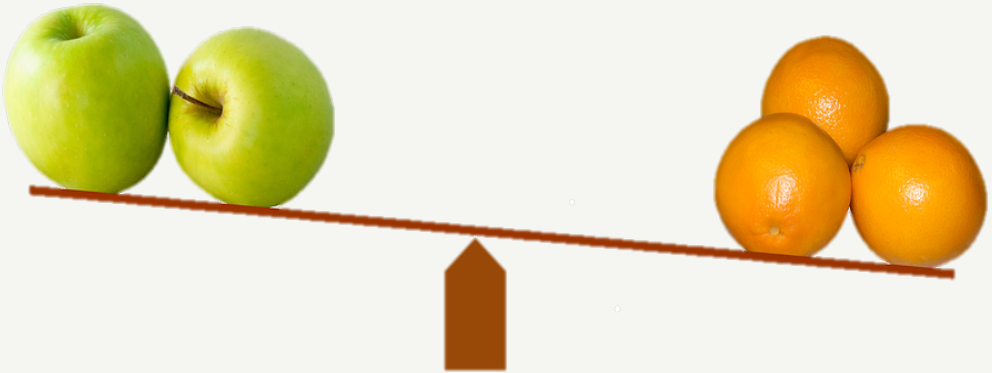
- Archaius 2: Dynamic configurations
- Spectator: Metrics
- Eureka: Middle-tier discovery service

NDBench is designed so that libraries are injected:

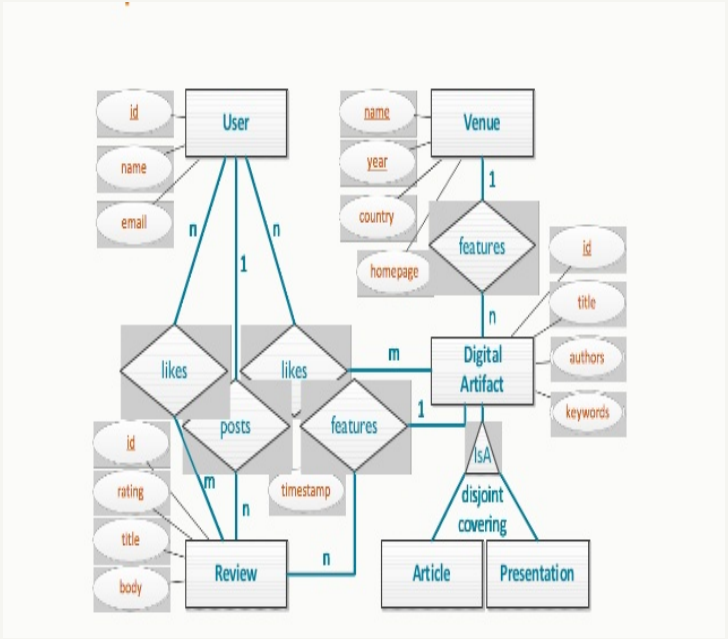
- Cross-cloud support



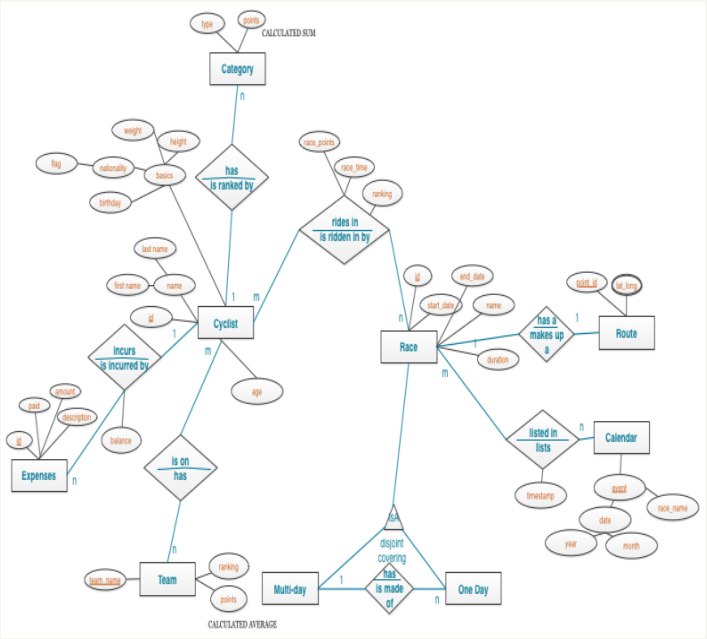
Side-by-Side comparison



Varying Data Models



VS

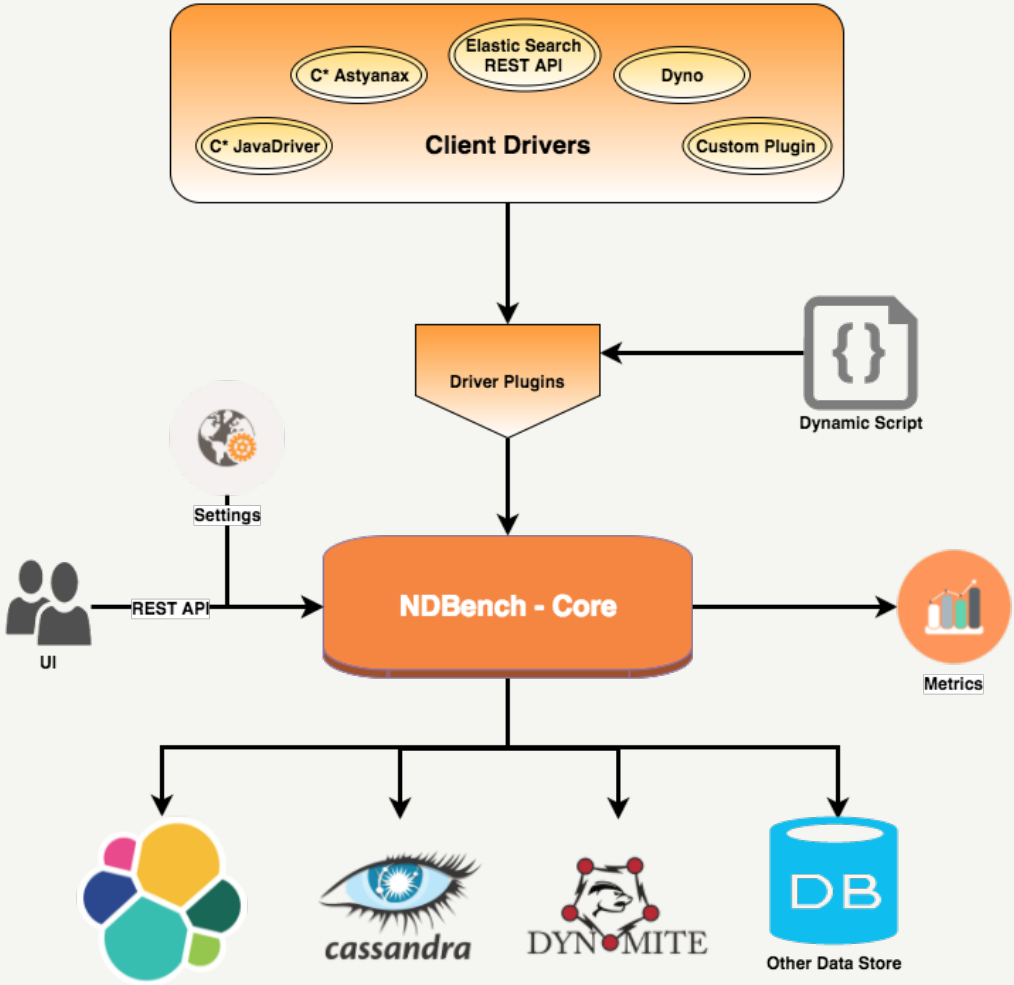


Pluggable Patterns and Loads

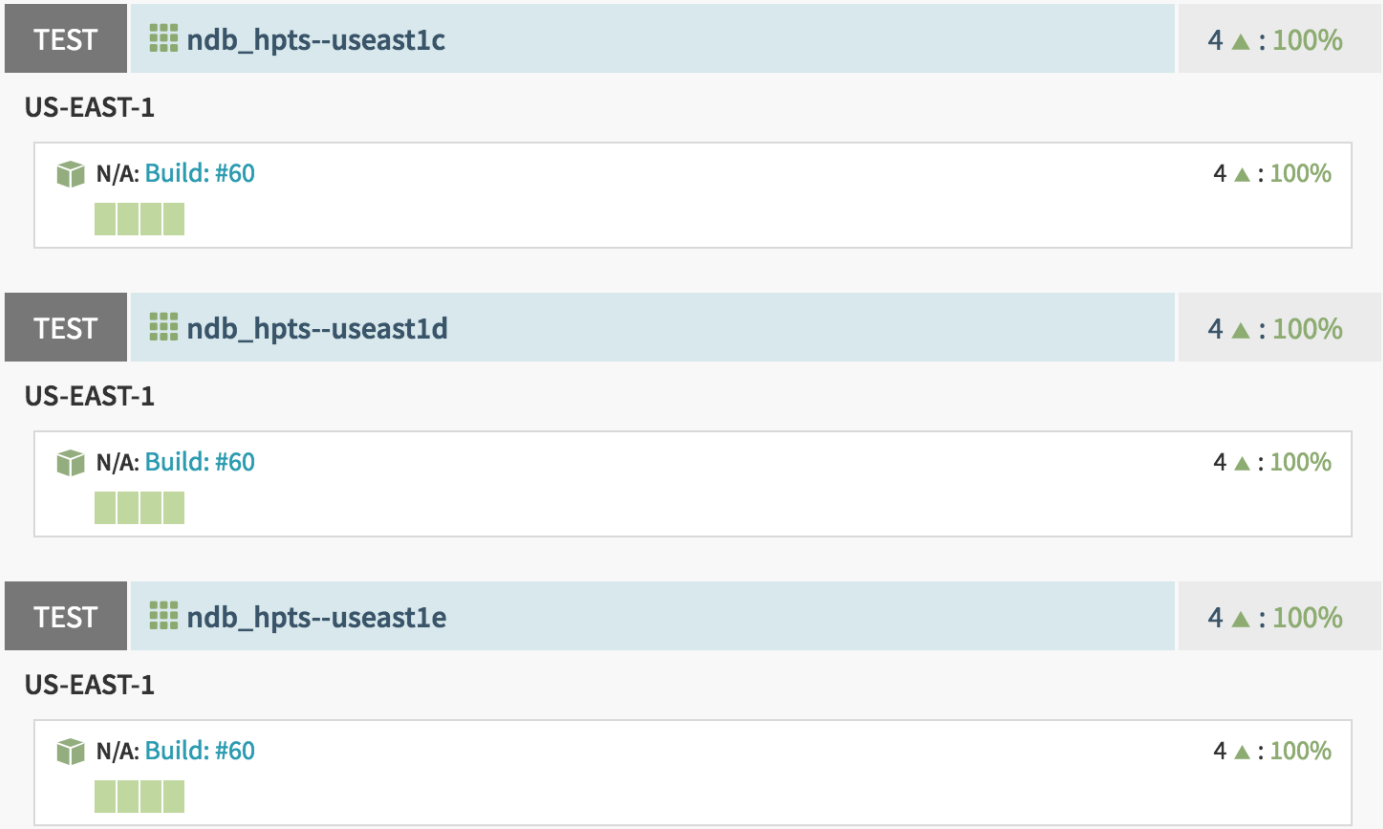
Different Client APIs

Architecture

- **Core**: Workload generator
- **API**: Adding plugins
- **Web**: UI and the servlet context listener



Configuring a cluster



NDBench Runner

1. Select a Cluster

Cluster
localhost

2. Connect a Driver

Driver
InMemoryTest
InMemoryTestPlugin - ConnectionInfo :: InMemoryMap Key Count: 0

HIDE SETTINGS | DISCONNECT DRIVER

Initial Settings

These settings can only be changed before connecting a Driver.

backfillStartKey	1
dataSize	128
dataSizeLowerBound	1000
dataSizeUpperBound	5000
numBackfill	1
numKeys	10000
numReaders	1
numValues	100
numWriters	1
readRateLimit	25
statsResetFreqSeconds	200
statsUpdateFreqSeconds	5
writeRateLimit	15
readEnabled	☑
useStaticData	☐
useVariableDataSize	☐
writeEnabled	☑

Runtime Settings

These settings can be changed at any time while a Load Test is running.

readRateLimit	25
writeRateLimit	15

SAVE SETTINGS

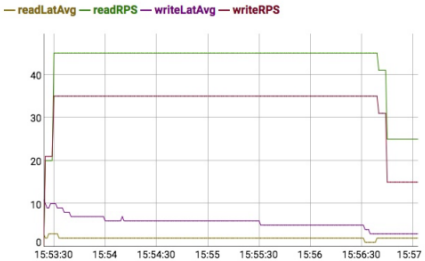
3. Run Load Tests

Load Pattern
RANDOM

Instance	Backfill	Writes	Reads
localhost:8080	▶	⏸ RandomString...	⏸ RandomString...

Instance Statistics

localhost:8080

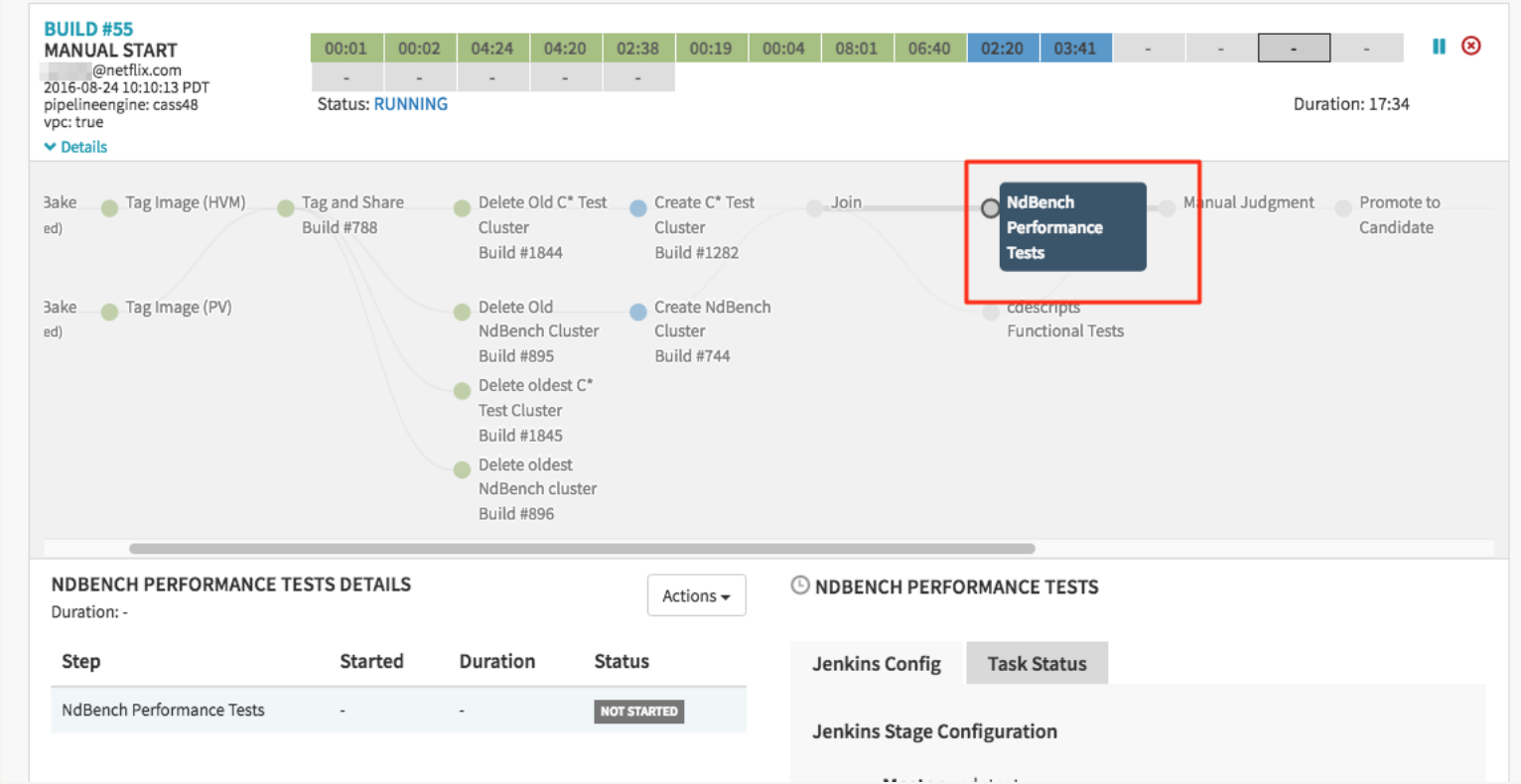


☐ cacheHitRatioInt ☐ cacheHits ☐ cacheMiss ☐ readFailure
☑ readLatAvg ☐ readLatP50 ☐ readLatP95 ☐ readLatP99
☐ readLatP995 ☐ readLatP999 ☑ readRPS ☐ readSuccess
☐ writeFailure ☑ writeLatAvg ☐ writeLatP50 ☐ writeLatP95
☐ writeLatP99 ☐ writeLatP995 ☐ writeLatP999 ☑ writeRPS
☐ writeSuccess

HIDE STATS JSON


```
{
    "cacheHitRatioInt": 29,
    "cacheHits": 2791,
    "cacheMiss": 6659,
    "readFailure": 0,
    "readLatAvg": 2,
    "readLatP50": 1,
    "readLatP95": 1,
    "readLatP99": 1,
    "readLatP995": 1,
    "readLatP999": 3,
    "readRPS": 25,
    "readSuccess": 9450,
    "writeFailure": 0,
    "writeLatAvg": 3,
    "writeLatP50": 3,
    "writeLatP95": 5,
    "writeLatP99": 6,
    "writeLatP995": 7,
    "writeLatP999": 14,
    "writeRPS": 15,
    "writeSuccess": 7267
}
```

C* AMI Certification Process



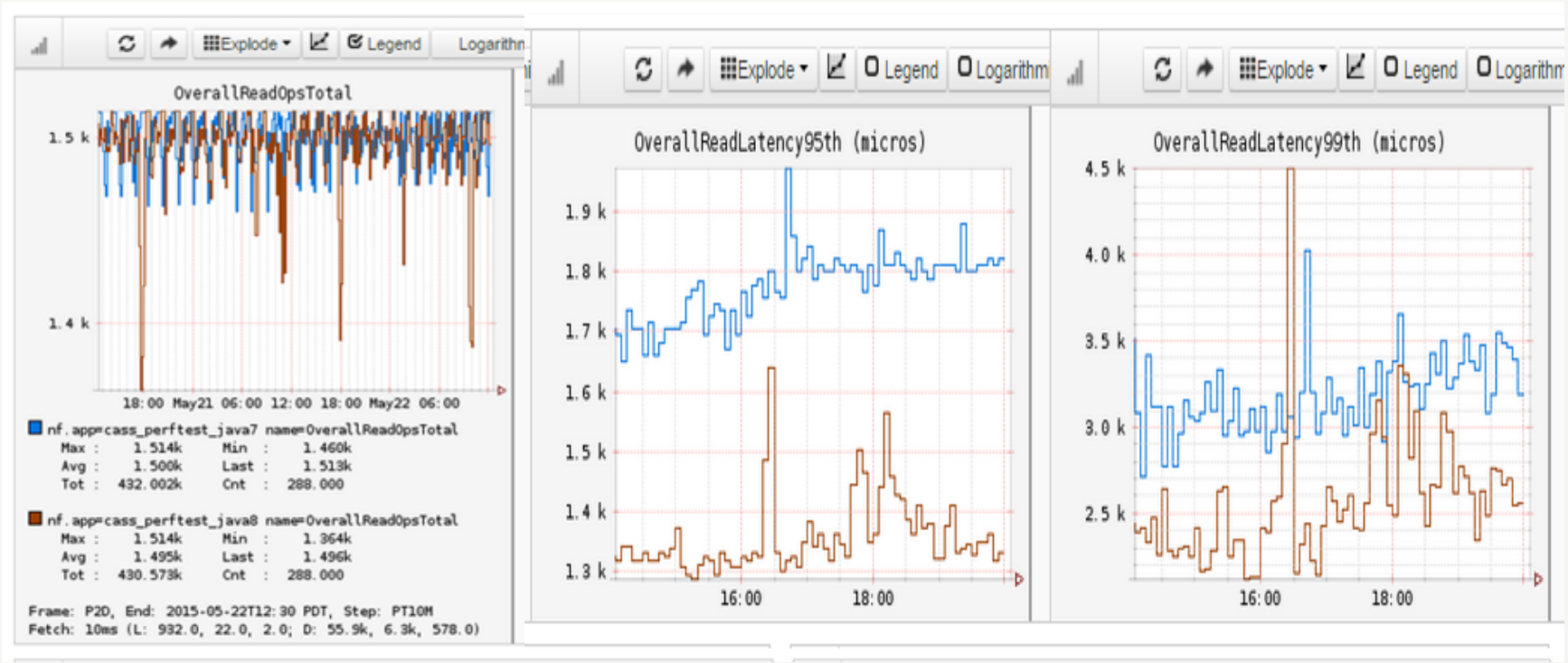
NDBench exemplar uses

C*

CentOS

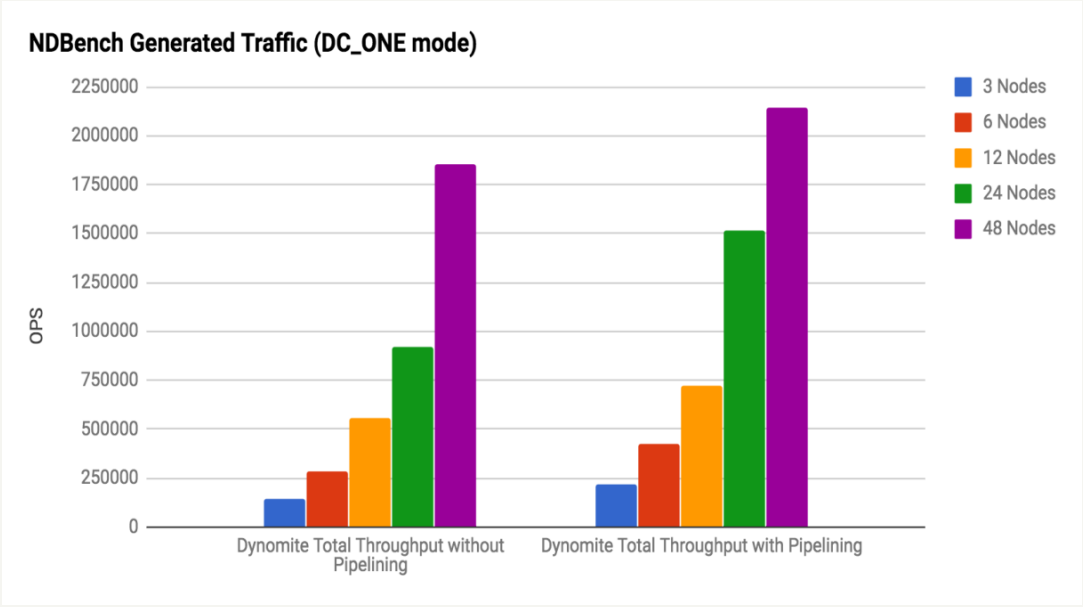
NETFLIX

C* Java 7 to Java 8 (reads)



Does our systems scale linearly?

Generating Millions of Ops/sec



Auto-tuning (Experimental)

- Requirements
  - Find the correct to load that the cluster can sustain
  - Evaluate different data stores
    - How many RPS/WPS they can go up to
    - Number of documents indexed per second per node
- Gradually increase load until we get failures
- After X% of failures, NDBench stops stepping up the load
    - Performs exponential backup to figure out if cluster is oversaturated.

Github: https://github.com/netflix/ndbench
Talk to us: https://gitter.im/Netflix/ndbench

Q & A

Ioannis Papapanagiotou
https://twitter.com/ipapapa