

#NoSQL Nation

SCYLLA

Real-Time processing of Big Data with ScyllaDB

Glauber Costa
VP Field Engineering, ScyllaDB

SCYLLA.

Today we will cover:

- + What's ScyllaDB; Why ScyllaDB
- + How ScyllaDB helps CERN.
- + What's under the hood, that allows that to happen

No clear winner in NoSQL



DynamoDB

Challenges:

- Cost
- Lock-in



mongoDB

Challenges:

- Scale
- Multi DC
- Latency



redis

Challenges:

- Not persistent
- Manageability



cassandra

Challenges:

- Price/performance
- Complexity
- JVM..

What we do: Scylla, towards the best NoSQL

Cassandra

- + 1,000-node cluster
- + Flexible replication
- + Multi Datacenter
- + CQL language
- + Auto sharding
- + Wide rows
- + Lightweight Transactions
- + Homogeneous nodes
- + Spark integration, Presto
- + Vibrant Open Source community
- + More

What we do: Scylla, towards the best NoSQL

- + > 1 million OPS per node
- + < 1ms 99% latency
- + Auto tuned
- + Scale up and out



Cassandra shares #1 rank in HA

- + 1,000-node cluster
- + Flexible replication
- + Multi Datacenter
- + CQL language
- + Auto sharding
- + Wide rows
- + Lightweight Transactions
- + Homogeneous nodes
- + Spark integration, Presto
- + Vibrant Open Source community
- + More

Some of Our Users

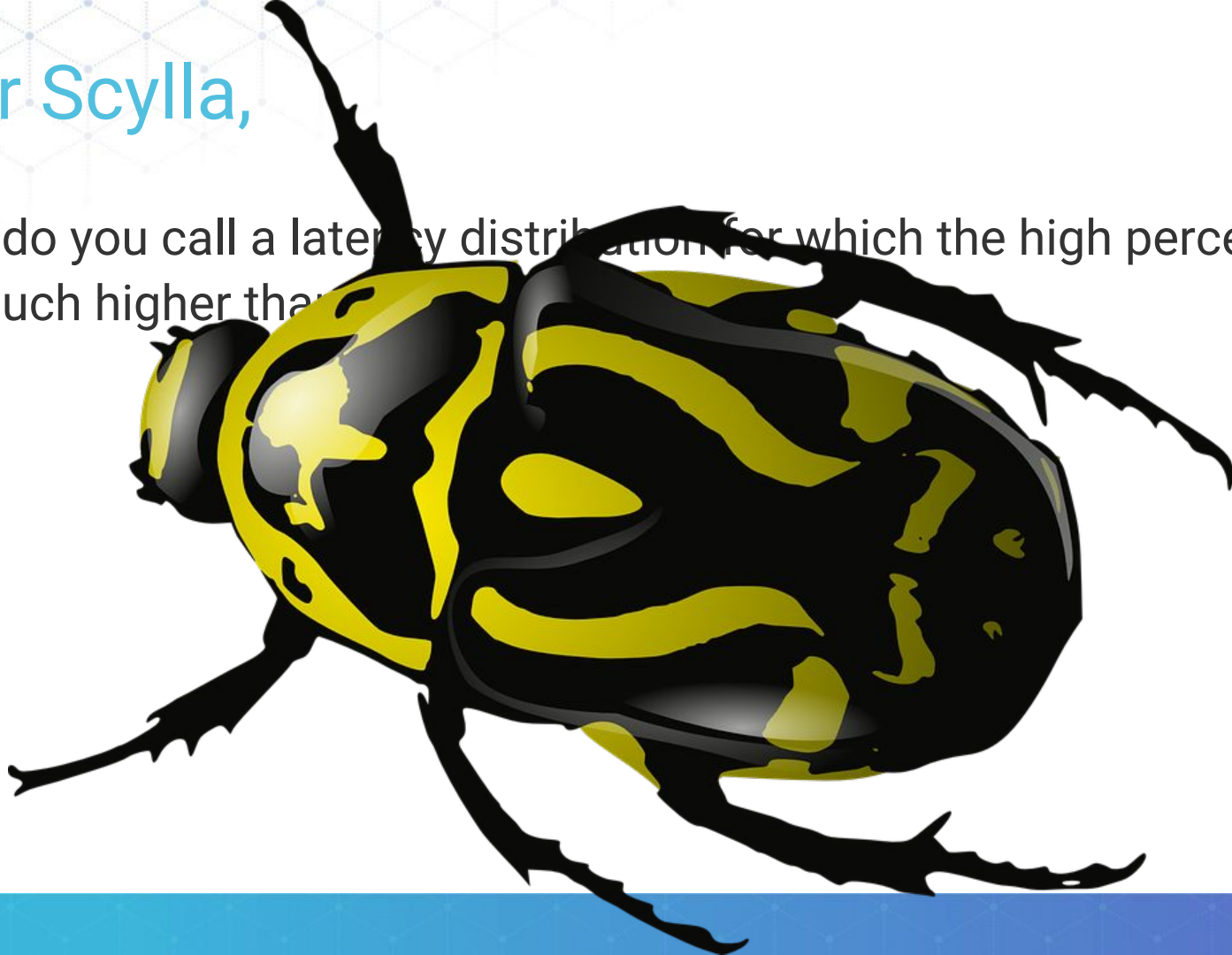


Dear Scylla,

What do you call a latency distribution for which the high percentiles are much higher than the average?

Dear Scylla,

What do you call a latency distribution for which the high percentiles are much higher than the low percentiles?



SCYLLA.

Please welcome Miguel!

Database use-case in



File Catalogue: metadata store for all the data of the experiment

- LFN namespace

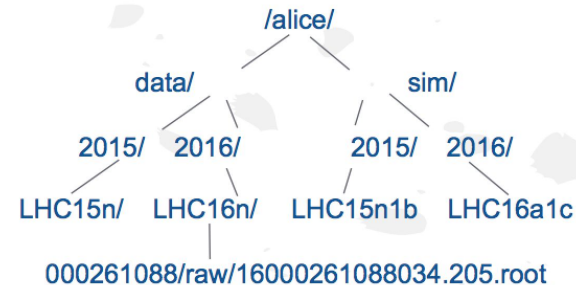
- /alice/data/2016/LHC16n/000261088/raw/16000261088034.205.root
- -rwxr-xr-x alidaq alidaq 264403565 Sep 09 22:10 0f24bce32446ea22840d188e035b11a9
- 1300+ tables, 5B entries, namespace split into tables

- GUID namespace

- 76CEBD12-76A0-11E6-9717-0D38A10ABEEF (Mac + TimeStamp based)
- 200+ tables, 5B entries, split by time intervals (append)

- Physical File Pointers

- 5B entries, 2B physical files, pointers to ZIP members, 110PB over 50 Storages



<root://alice-tape-se.gridka.de:1094//10/33903/76cebd12-76a0-11e6-9717-0d38a10abeef>

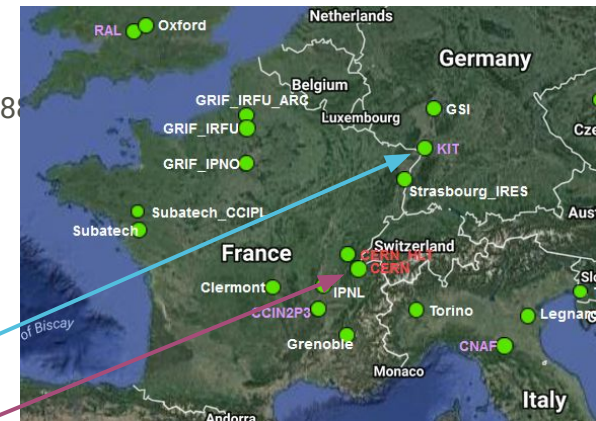
<root://voalice10.cern.ch//castor/cern.ch/.../16000261088034.205.root>

Database use-case in



File Catalogue: metadata store for all the data of the experiment

- LFN namespace
 - /alice/data/2016/LHC16n/000261088/raw/16000261088034.205.root
 - -rwxr-xr-x alidaq alidaq 264403565 Sep 09 22:10 0f24bce32446ea22840d18
 - 1300+ tables, 5B entries, namespace split into tables
- GUID namespace
 - 76CEBD12-76A0-11E6-9717-0D38A10ABEEF (Mac + TimeStamp based)
 - 200+ tables, 5B entries, split by time intervals (append)
- Physical File Pointers
 - 5B entries, 2B physical files, pointers to ZIP members, 110PB over 50 Storages

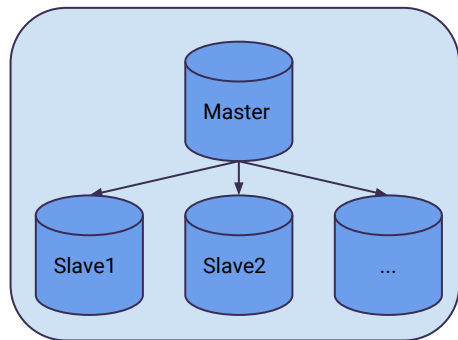


<root://alice-tape-se.gridka.de:1094//10/33903/76cebd12-76a0-11e6-9717-0d38a10abeef>

<root://voalice10.cern.ch//castor/cern.ch/.../16000261088034.205.root>

Looking towards the future

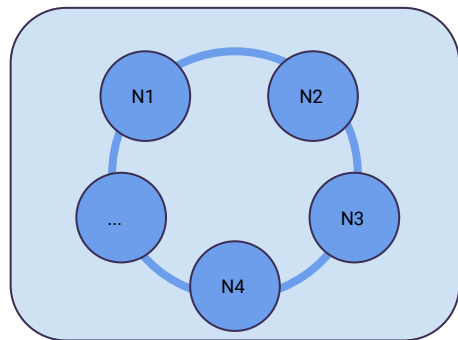
MySQL



- LHC Run3 **challenge**
 - **50B** entries
 - **O(100K)** ops/s



ScyllaDB/Cassandra



- Manual **sharding**
 - Split hierarchy into tables
- Single **point of failure**
- Rely on good **hardware** for performance
- Today:
 - **15B** entries
 - **O(10K)** ops/s
 - 6TB on disk

- Automatic **sharding**
- **No single point of failure**, HA
- **Horizontal scaling**, cheap hardware
- Consistency
- Paradigm change
 - SQL to **noSQL**

Data model in C*



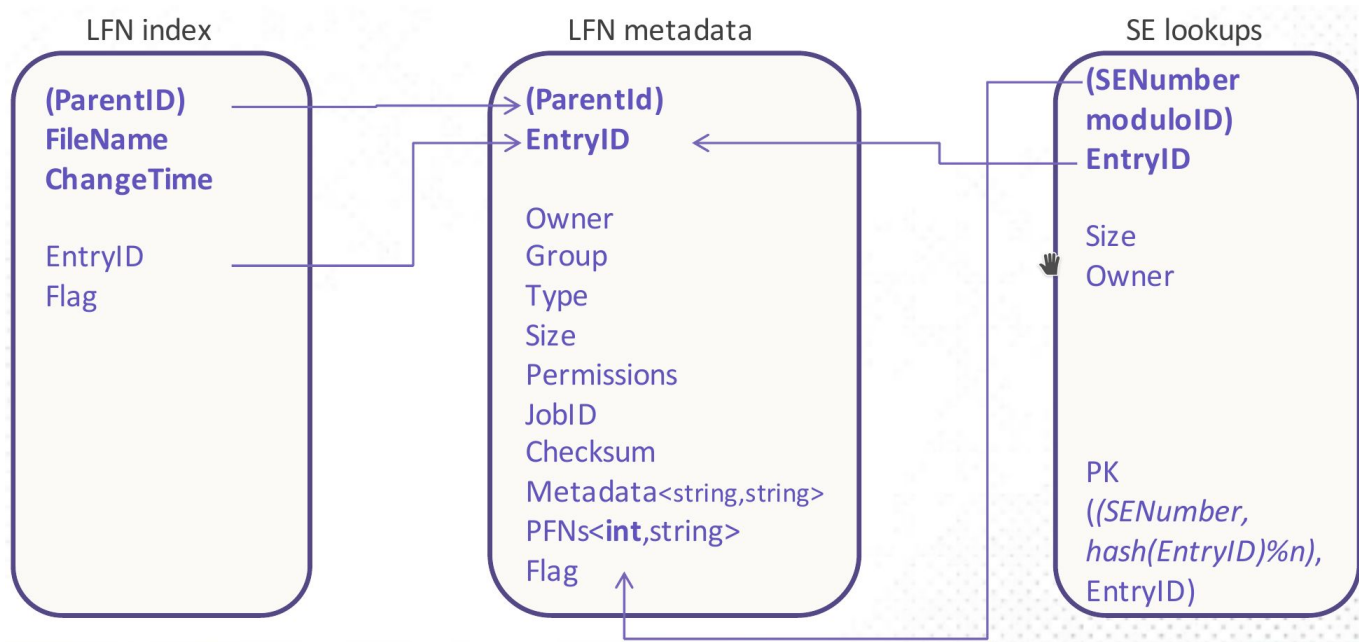
Find **balance** between **performance** and **functionality** => schema affects both drastically!



- Move/delete (folders) efficient, fast metadata retrieval, easy to keep trash bin

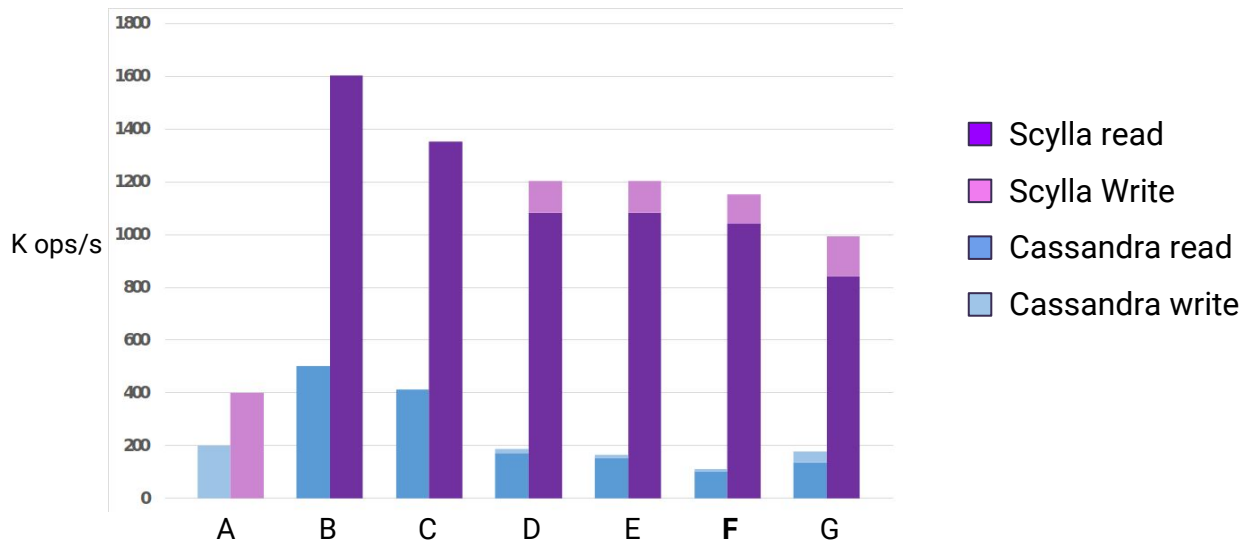


- Need to loop over lfn index (but cacheable)



Benchmarking results

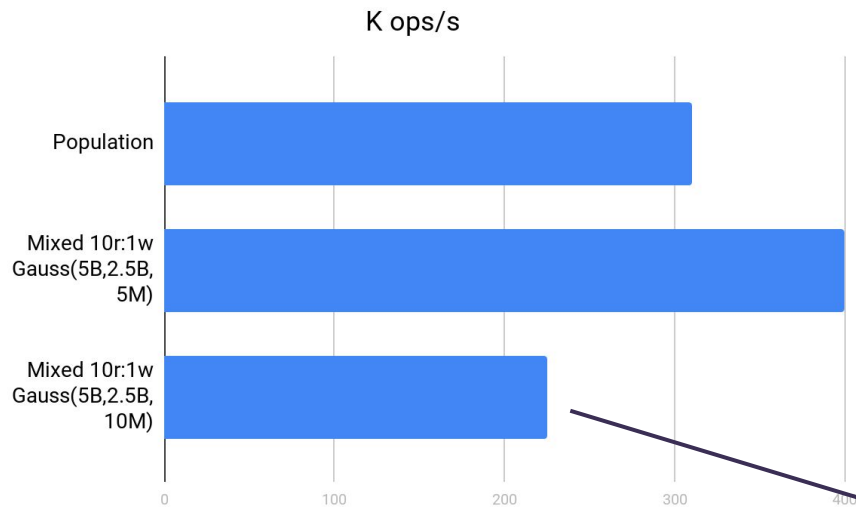
- **Cassandra-stress**
- **6-node** cluster
- Node type:
 - 56 HT-cores
 - 128GB RAM
 - Single SSD



	Benchmark (default cassandra-stress)	Cassandra	Scylla	Factor ops/s
A	Insert only	18 %util, 2% iowait	100 %util	x2
B	Read-only Gauss(5B,2.5B,10K)	Disk idle, 50% cpu	100 %cpu	x3
C	Read-only Gauss(5B,2.5B,1M)	11 %util, 40% cpu	11 %util, 100 %cpu	x3.28
D	Mixed (10r,1w) Gauss(5B,2.5B,100K)	2 %util, 45% cpu	10 %util, 100 %cpu	x5.8
E	Mixed (10r,1w) Gauss(5B,2.5B,1M)	5 %util, 50% cpu	16 %util, 100 %cpu	x6.4
F	Mixed (10r,1w) Gauss(5B,2.5B,10M)	9% util, 45% cpu	40 %util, 100 %cpu	x6.1
G	Mixed 2K thrd. read, 200 write, G(5B,2.5B,100K)	8 %util, 40% cpu, no iowait	26 %util, 100 %cpu	x5.62

Benchmarking results

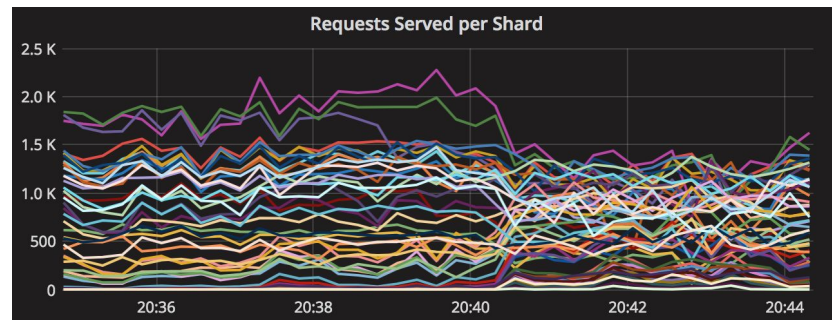
- **jAliEn benchmark** => synthetic representation of our real use-case
- **7-node** cluster
- Node type:
 - 56 HT-cores
 - 256GB RAM
 - Single SSD



- Room for **improvement**?
 - Hardware limitations (disk)

Busiest disk IO			
%util	IOPS	MB/s	dev
100	59442	254.8 MB/s	sda
100	63390	276.5 MB/s	sda
100	64107	267.1 MB/s	sda
100	57949	242.9 MB/s	sda
100	59811	253.6 MB/s	sda
100	61228	257 MB/s	sda
100	56202	377.7 MB/s	sda

- Workload imbalances



- Cache hits/insertions < 2

Initial Run3 **requirements** achieved

Personal notes/experiences

- The **internal implementation** of ScyllaDB and Cassandra differ vastly
 - Cassandra needs **manual tuning**: JVM, Kernel, Cassandra config => several factor improvement
- The **compactions** and **repairs** should be taken into account for performance and operations
 - Why some compaction strategies are *not recommended*
 - Repair is an obligatory periodic maintenance operation => reads/writes significant amount of data, data recovery
- Cassandra **daemon** can be run multiple times per node
 - Operations, commands and network setup get tricky
 - Performance increase is not a direct factor of number of instances in our tests => 30-40% performance increase
 - It deals with the lack of saturation of resources => locks on Java application level
- **Monitoring** is your best friend
 - Node/cluster problems
 - Imbalanced workloads
 - Provisioning, hardware profiling



Next steps

- Continue **optimisations** (within reason)
- Exercise fast catalogue **imports**
 - Sqoop, MySQL slaves
- Finalize application side **developments** and run first **production** cluster
 - Tightly coupled with the development of our new framework: [jAliEn](#)
 - Compare and validate Scylla/C* with MySQL
- Exercise efficient **full-table scans**
 - Apache Spark, Token-Range parallel exploration
 - Used to keep user and storage quotas, catalogue statistics, popularity?...
- Validate **maintenance** and **recovery** mechanisms
 - Backup, repair, restore
- Find a model for a new use-case: **CCDB** (Run3 conditions data backend)
 - Time-ranges queries not trivial to map without several indexes or a dedicated structure/type
 - DateRangeType in DataStax Enterprise Cassandra

SCYLLA.

HOW?

Two-level sharding - shard per core

Threads

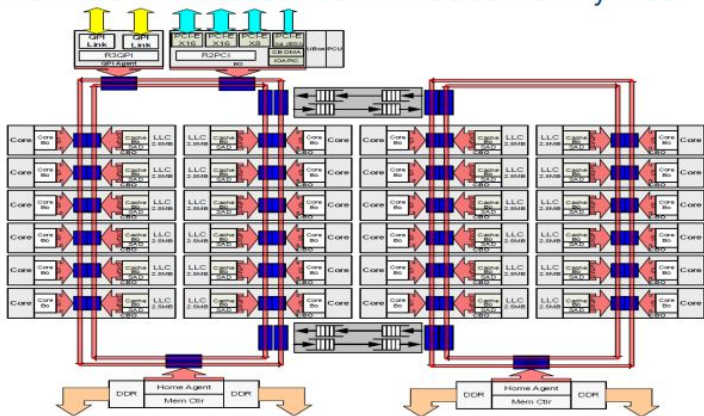


Shards



Seastar, Scylla's engine: "All things async"

Intel® Xeon® Processor E5 v4 Product Family HCC



Close to the hardware

- Our own memory allocator
- Our own Disk I/O Scheduler
- Our own CPU Scheduler
- Our own cache, bypasses Linux entirely.

The Autonomous NoSQL Database

- SLA for Requests over maintenance operations
- Automatic tuning
- Automatic backpressure
- Scale up/down easily and stream as fast as possible
- Ongoing repair
- Smooths complex data models

Autonomous operations - example 1

(Speed mismatch)



How fast is my system?

- There are two speeds:
 - Disk Speed
 - CPU/memory speed
- What happens when they are not in sync ?

```
latency mean           : 51.9
latency median         : 9.8
latency 95th percentile : 125.6
latency 99th percentile : 1184.0
latency 99.9th percentile : 1991.2
```

How fast is my system?

- There are two speeds:
 - Disk Speed
 - CPU/memory speed
- What happens when they are not in sync ?

```
latency mean           : 51.9
latency median         : 9.8
latency 95th percentile : 125.6
latency 99th percentile : 1184.0 (x 22)
latency 99.9th percentile : 1991.2 (x 38)
```

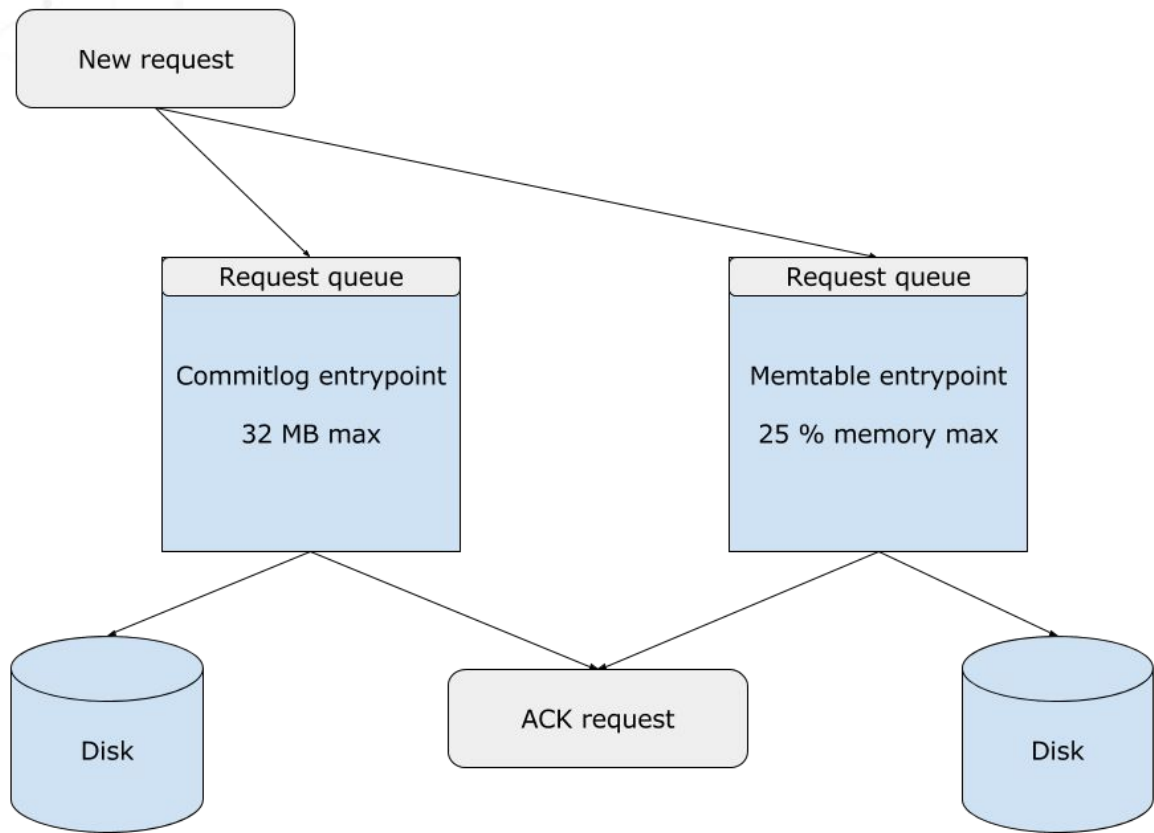
The Wall - where is it relevant?

- Disk speed slower than CPU speed
 - plain slow disk, large payloads

The Wall - where is it relevant?

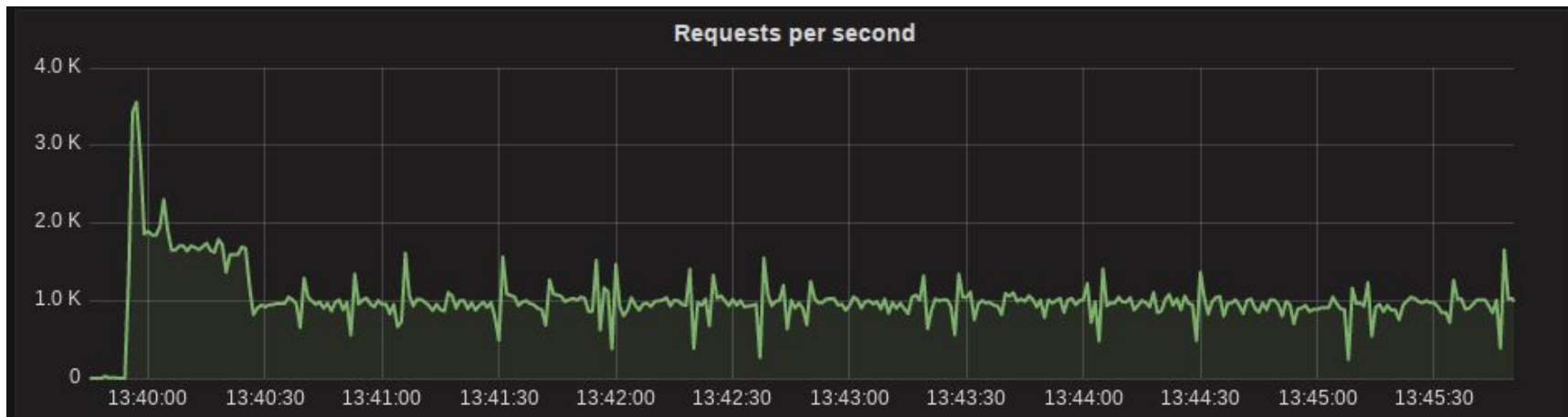
- Disk speed slower than CPU speed
 - plain slow disk, large payloads
- Any other mismatch between resources
 - For example, large memory capped by narrow network

The Wall



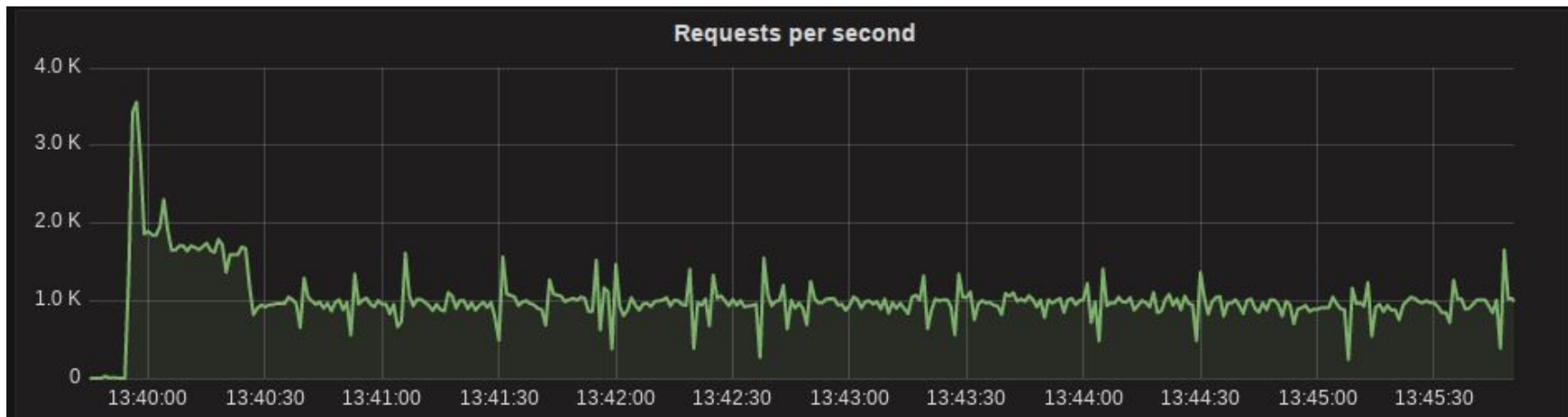
The Wall - Results

```
latency mean           : 54.9  
latency median        : 43.5  
latency 95th percentile : 126.9  
latency 99th percentile : 253.9  
latency 99.9th percentile : 364.6
```



The Wall - Results

```
latency mean           : 54.9  
latency median        : 43.5  
latency 95th percentile : 126.9  
latency 99th percentile  : 253.9 (x 4.6)  
latency 99.9th percentile : 364.6 (x 6.6)
```



Autotuning example 2

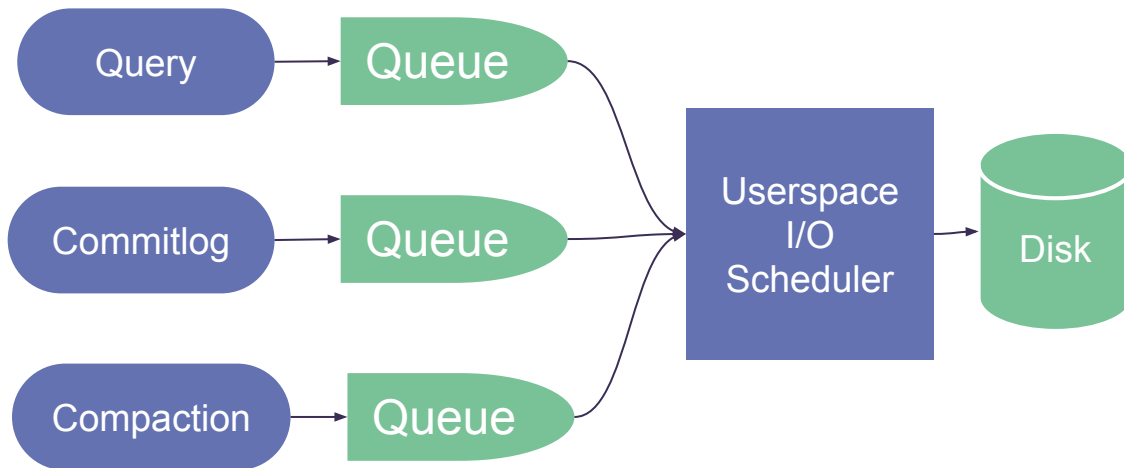
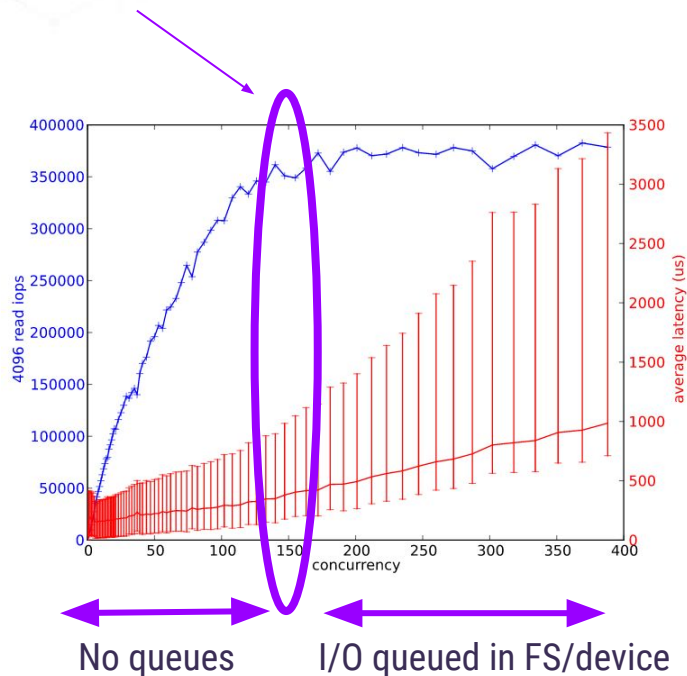
(Fighting imperfect Isolation)



AMCtv.com Photo by Gene Page © TW Production, LLC. All rights reserved

The I/O Scheduler

Max useful disk concurrency



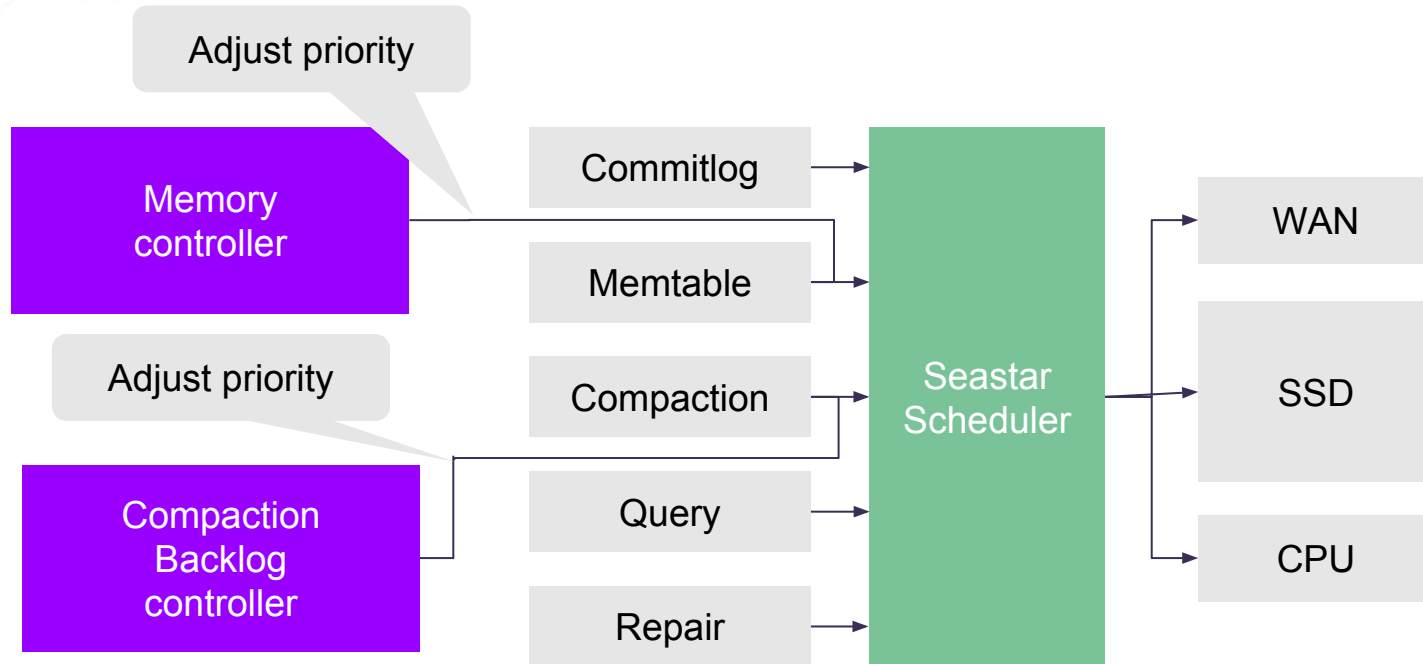
The I/O Scheduler

- Major component of Scylla since early versions
 - Central component in The Wall
 - Getting major improvements for latency workloads in Scylla 2.3

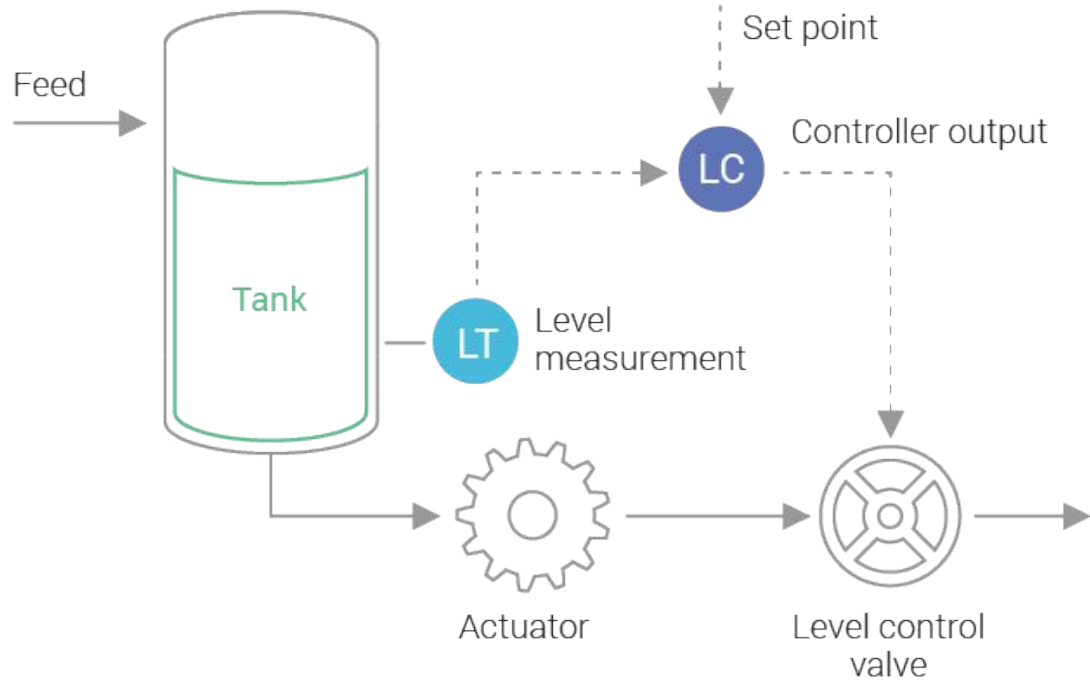
The CPU Scheduler

- Since Scylla 2.0, initial version
 - disabled by default, AdGear enables it.
 - enabled in our AWS AMI if using i3 instances.
- 2.2 ships with the full solution
 - Ships this week!
 - Enabled by default everywhere.
 - Much better isolation

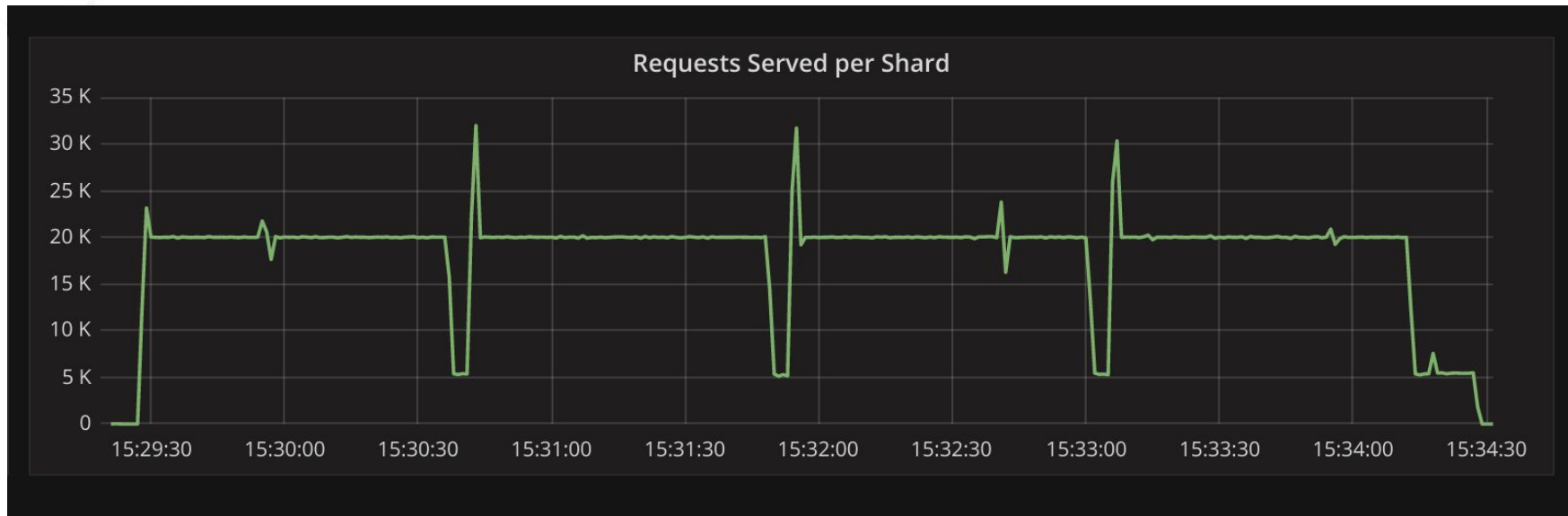
The Autonomous Database



The controllers



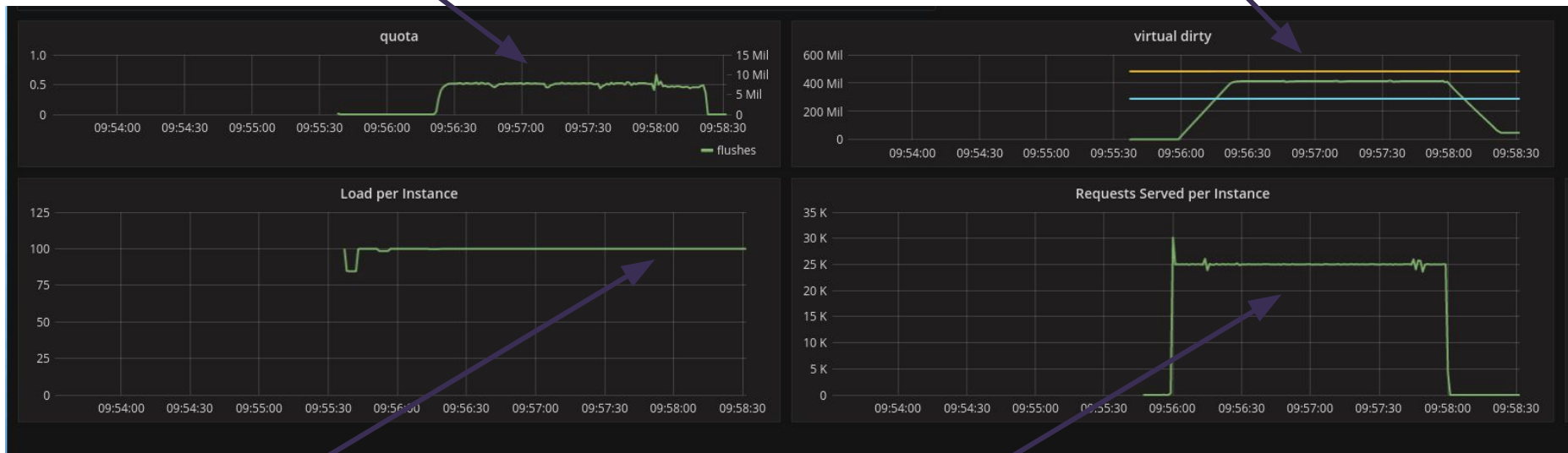
The controllers



The controllers - memtable

This is the CPU percentage needed (50 %)

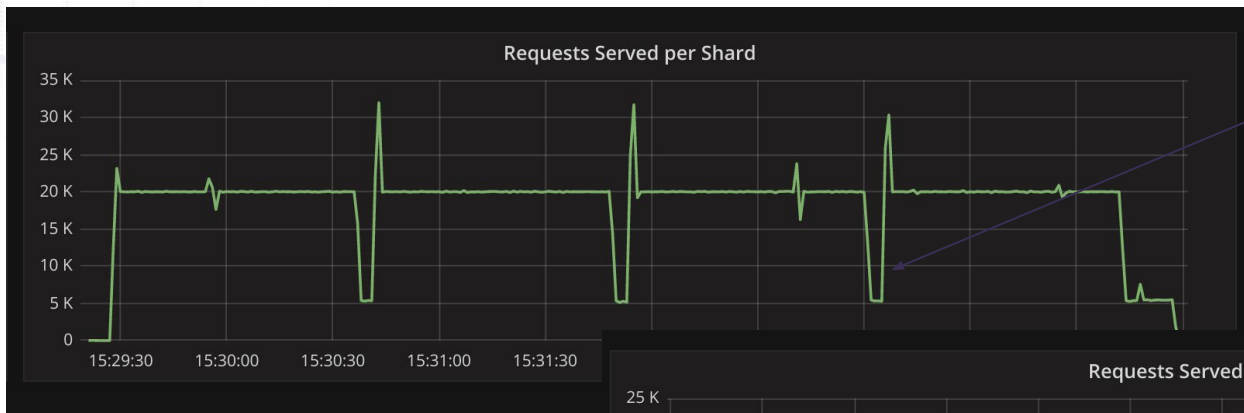
To keep the buffers at a stable level



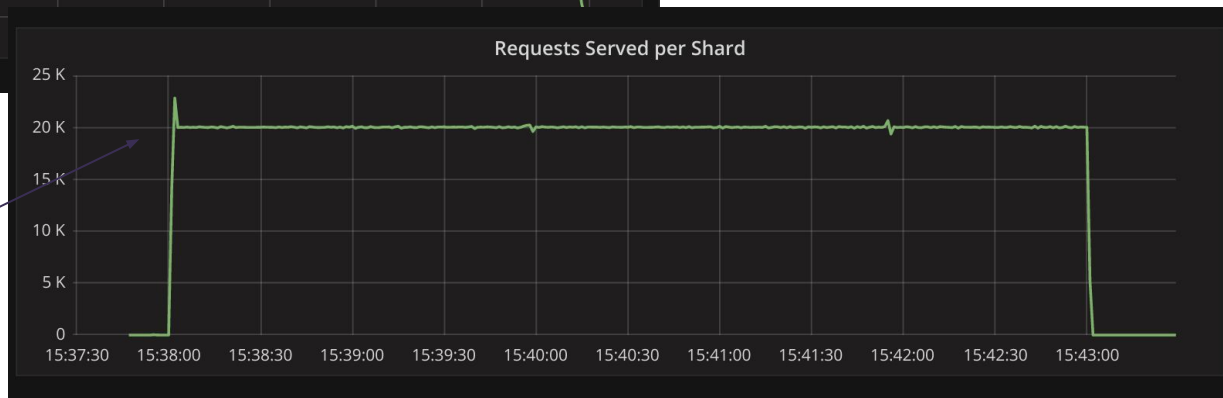
Total system CPU usage barely oscillates

Throughput barely oscillates

The controllers - memtable



without controller



with controller

The controllers - memtable

Before:

```
latency mean           : 0.6
latency median         : 0.5
latency 95th percentile : 0.8
latency 99th percentile : 3.6 (x 6.0)
latency 99.9th percentile : 4.5 (x 7.5)
```

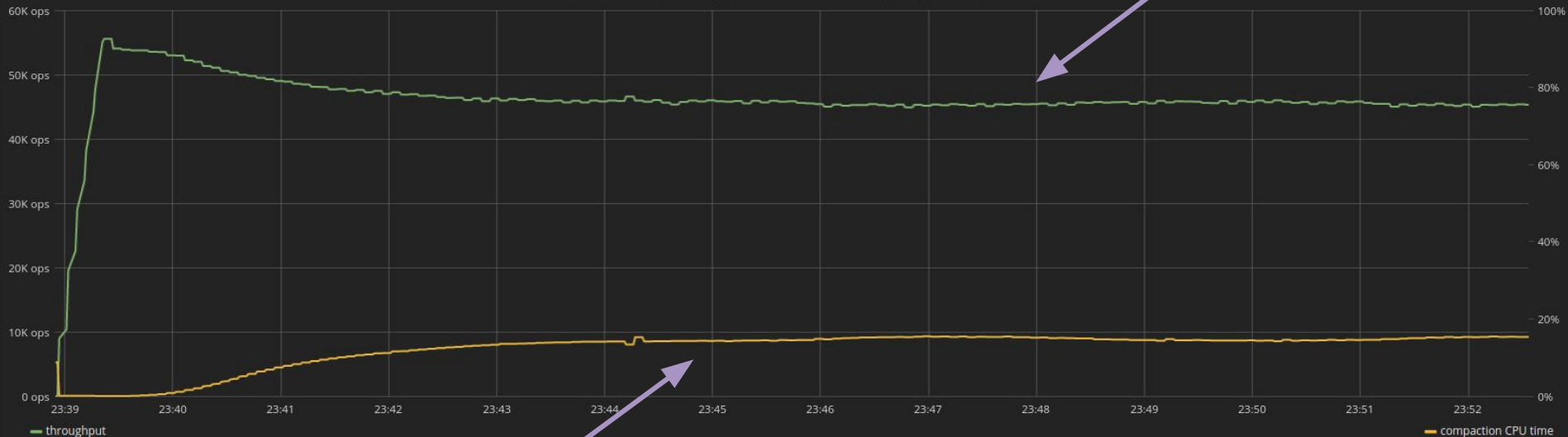
After:

```
latency mean           : 0.4
latency median         : 0.4
latency 95th percentile : 0.6
latency 99th percentile : 0.8 (x 2.0)
latency 99.9th percentile : 1.9 (x 4.7)
```

The controllers - compactions

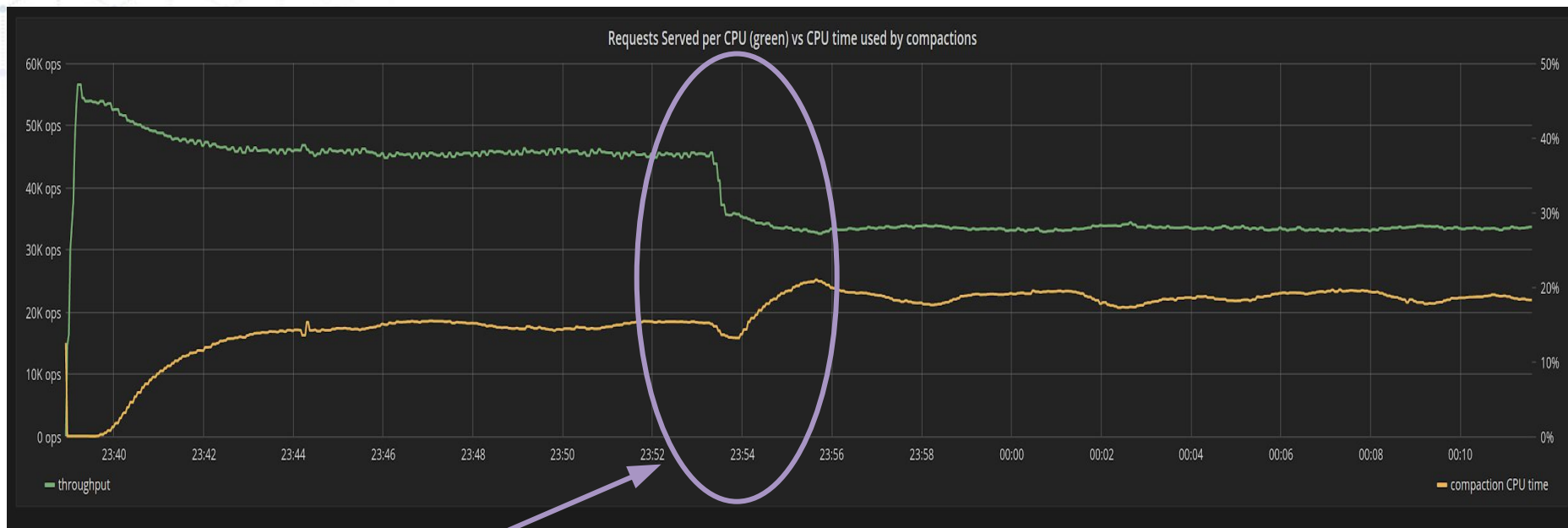
Throughput

Requests Served per CPU (green) vs CPU time used by compaction (yellow)



% CPU time used by Compactions

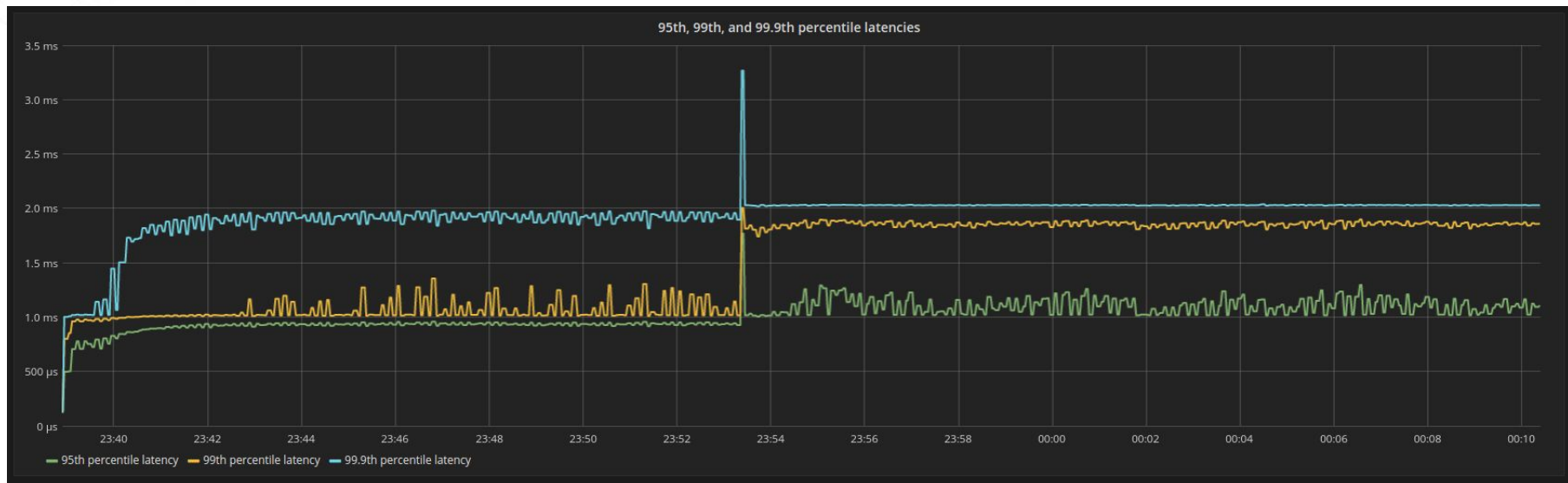
The controllers - compactions



workload changes:

- automatic adjustment
- new equilibrium

The controllers - compactions



2ms : 99.9 % latencies at 100 % load
< 2ms : 99 % latencies,
1ms : 95 % latencies.

The controllers - coming soon

- Scylla 2.2: SizeTiered compactions are controlled.
- Scylla 2.3: All compaction strategies are controlled.
- Repairs
 - Repairs already respect latencies very well, but are not as fast as they could be. Controllers will help unleash their full potential
 - Done: Scylla Enterprise Manager schedules repairs automatically, no human involvement needed

Summary

- Scylla inherits the user-visible architecture from Cassandra, a solution that is known to scale out very well
- Scylla employs a radically different internal architecture, allowing it to scale up as well as out while keeping latencies predictable
- Scylla reduces TCO across the board, by also minimizing operational expenses.

Thanks You!

Resources

glauber@scylladb.com (@glcst)

scylladb.com/blog



[@scylladb](https://twitter.com/scylladb)



<http://bit.ly/2oHAfok>



youtube.com/c/scylladb



slideshare.net/ScyllaDB