Improving Bad Partition Handling
In Apache Cassandra

Cheng Wang
Jordan West

Who We Are?

— **Jordan West**

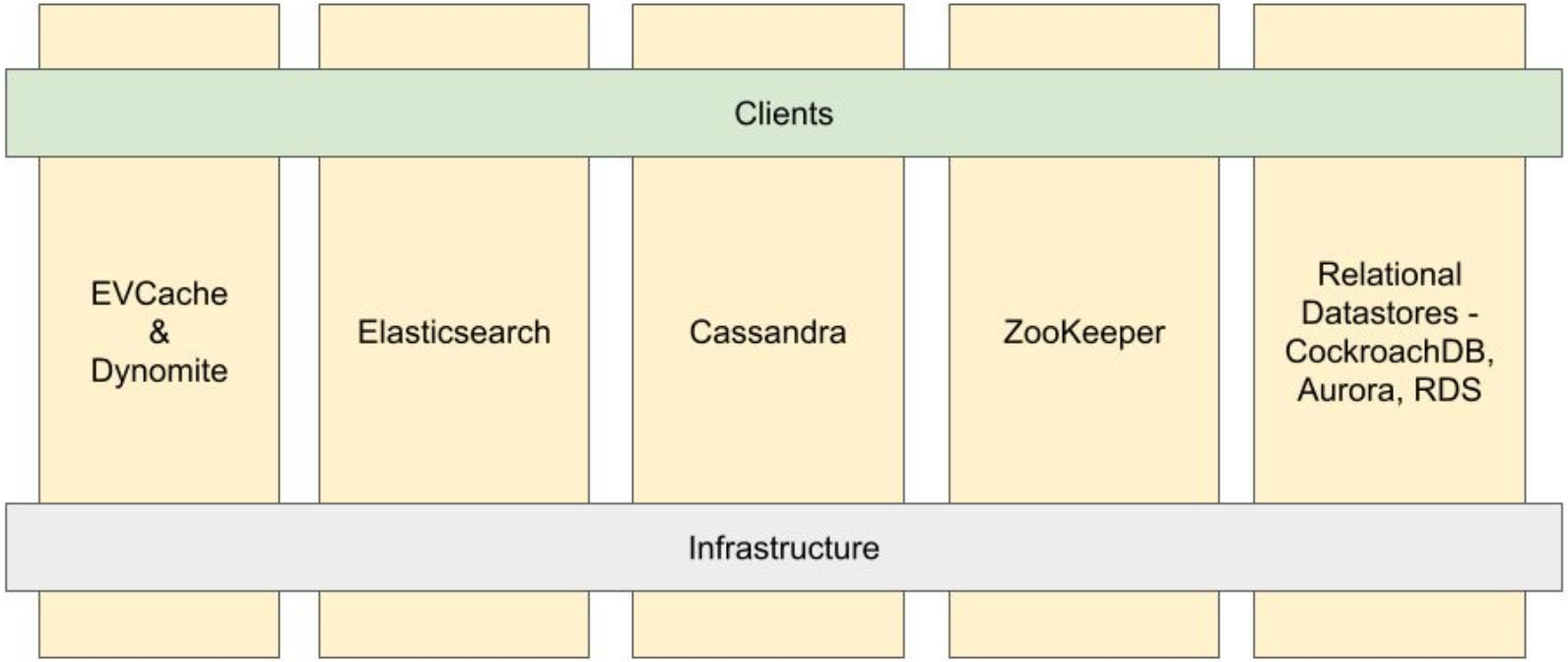
- SWE @ Netflix approx. 2 years
- Cassandra committer
- Been working with Cassandra for approx. 7 years total. Databases 10+.

— **Cheng Wang**

- SWE @ Netflix approx. 6 months
- Been working with database engine for approx. 6 years total

Online Datastore Team @Netflix

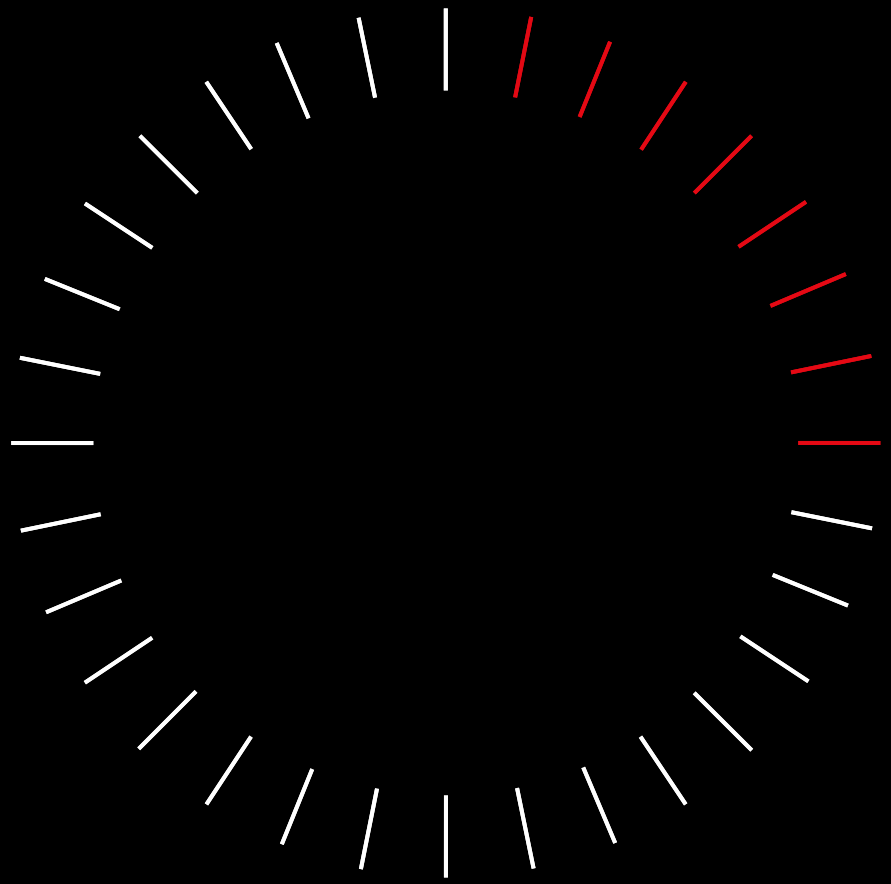
- Provide high leverage datastores as a managed service at scale, to support all operational data needs spanning across Streaming, Studio and Gaming businesses for Netflix.

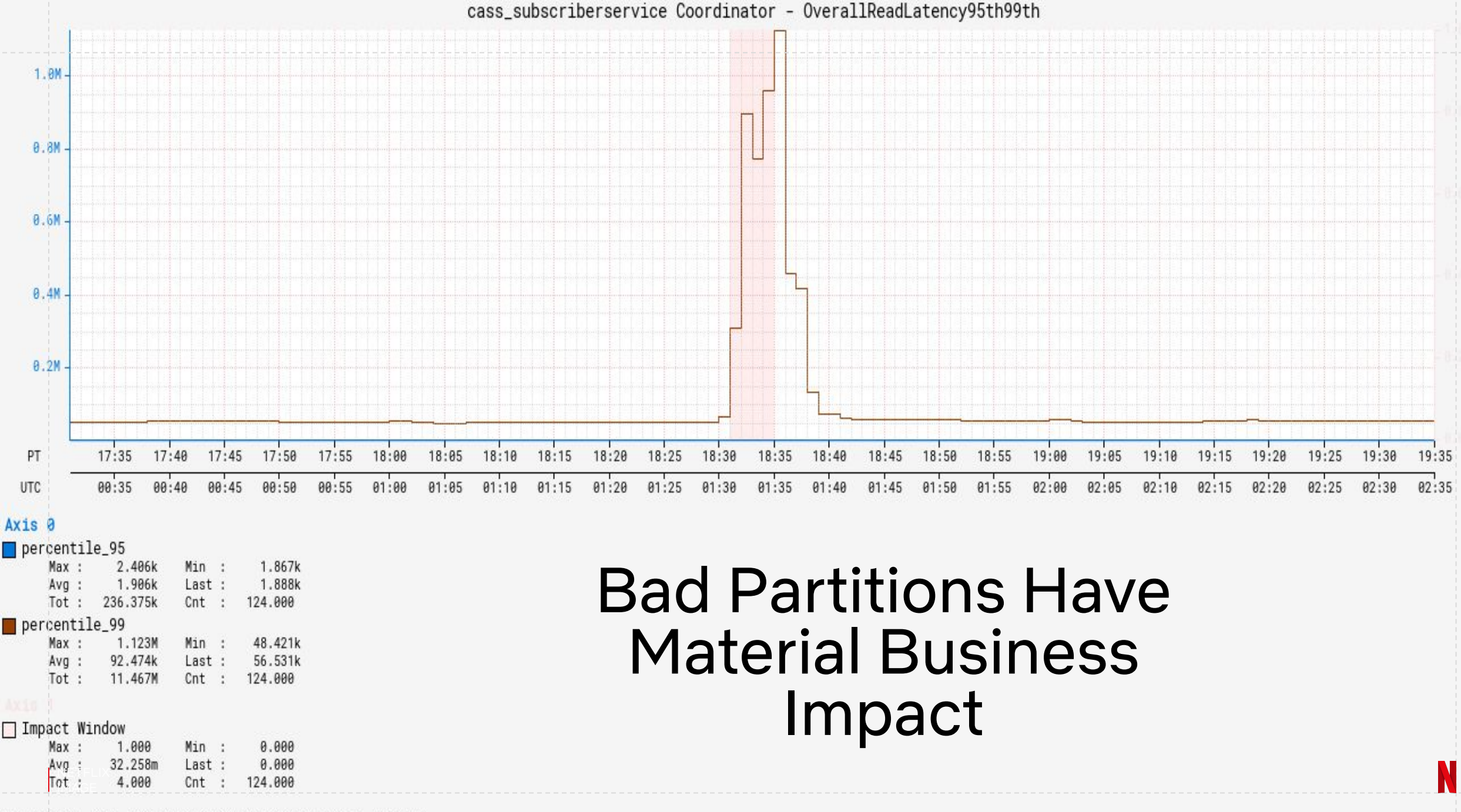


Cassandra @ Netflix

- Over 22,000 Nodes
- 900+ Clusters
- 12+ PB of Data
- 12M+ req/s (approx. 60-40 read-write)
- Cassandra 3.0.x (for now)
- Moving towards 4.x targeting rollout 2023

Why Are Bad Partitions Bad?





Bad Partitions Have Material Business Impact

playapi -> subscriberservice IPC errors by status

Axis 0

cancelled
Max : 133.333m Min : 0.000
Avg : 5.031m Last : 0.000
Tot : 266.667m Cnt : 53.000

throttled
Max : 2.670k Min : 2.033
Avg : 65.685 Last : 2.567
Tot : 8.145k Cnt : 124.000

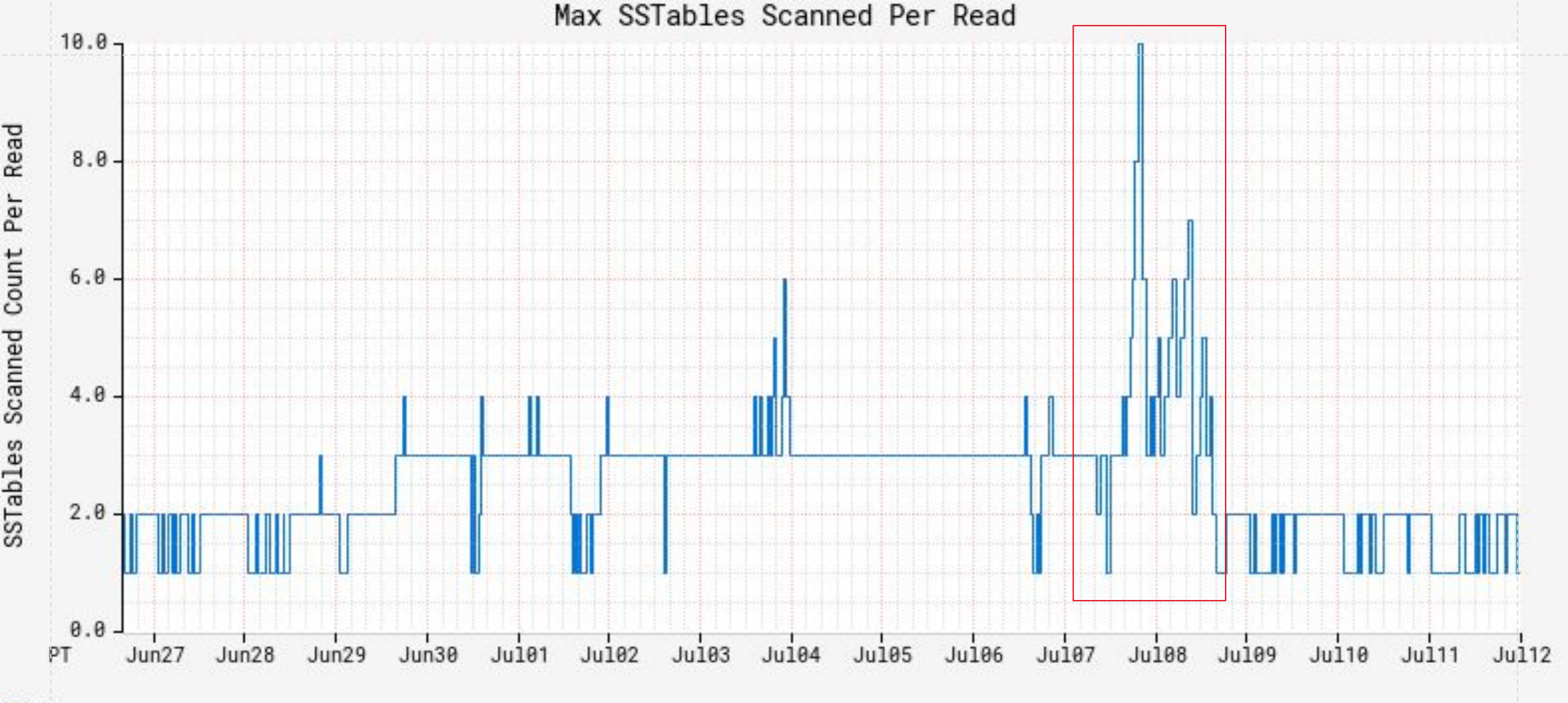
timeout
Max : 66.667m Min : 0.000
Avg : 3.333m Last : 0.000
Tot : 166.667m Cnt : 50.000

unexpected_error
Max : 42.550 Min : 0.000
Avg : 6.222 Last : 0.000
Tot : 143.100 Cnt : 23.000

Impact Window
Max : 1.000 Min : 0.000
Avg : 32.258m Last : 0.000
Tot : 4.000 Cnt : 124.000

Frame: 124m, End: 2021-07-06T19:36-07:00[US/Pacific], Step: 1m
Fetch: 15ms (L: 90.0, 44.0, 5.0; D: 5.4k, 5.5k, 620.0k)


cass_subscriberservice Replica Reads - Maximum Latency99th Per CF
300.0k
200.0k
100.0k
PT
17:35 17:40 17:45 17:50 17:55 18:00 18:05 18:10 18:15 18:20 18:25 18:30 18:35 18:40 18:45 18:50 18:55 19:00 19:05 19:10 19:15 19:20 19:25 19:30 19:35
UTC
00:35 00:40 00:45 00:50 00:55 01:00 01:05 01:10 01:15 01:20 01:25 01:30 01:35 01:40 01:45 01:50 01:55 02:00 02:05 02:10 02:15 02:20 02:25 02:30 02:35
Axis 0
Max : 943.127 Min : 943.127
Avg : 943.127 Last : 943.127
Tot : 116.948k Cnt : 124.000
Min : 943.127
Avg : 943.127 Last : 943.127
Tot : 116.948k Cnt : 124.000
Min : 943.127
Avg : 946.169 Last : 943.127
Tot : 117.325k Cnt : 124.000



Frame: 368h, End: 2021-07-12T00:30-07:00[US/Pacific], Step: 30m
Fetch: 14ms (L: 35.5k, 102.0, 1.0; D: 2.1M, 75.1k, 736.0k)

Types of Bad Partitions

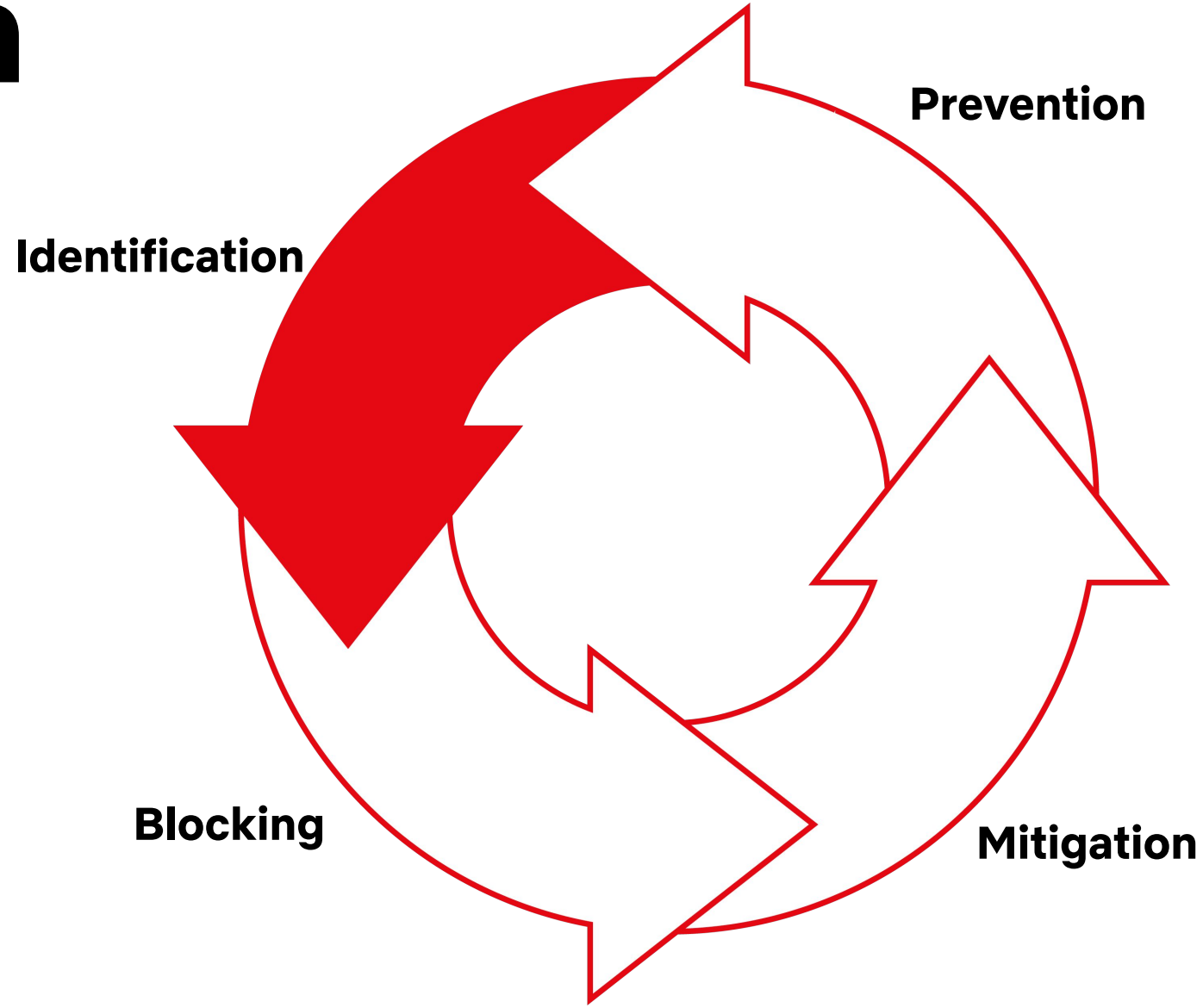
- **Total Size:** Partitions in the GB+ size range
- **Row Count:** Smaller partitions with Million+ rows
- **Tombstone Count:** Partitions with Million+ tombstones
- **SSTable Count:** A partition spread across many sstables is more expensive to read

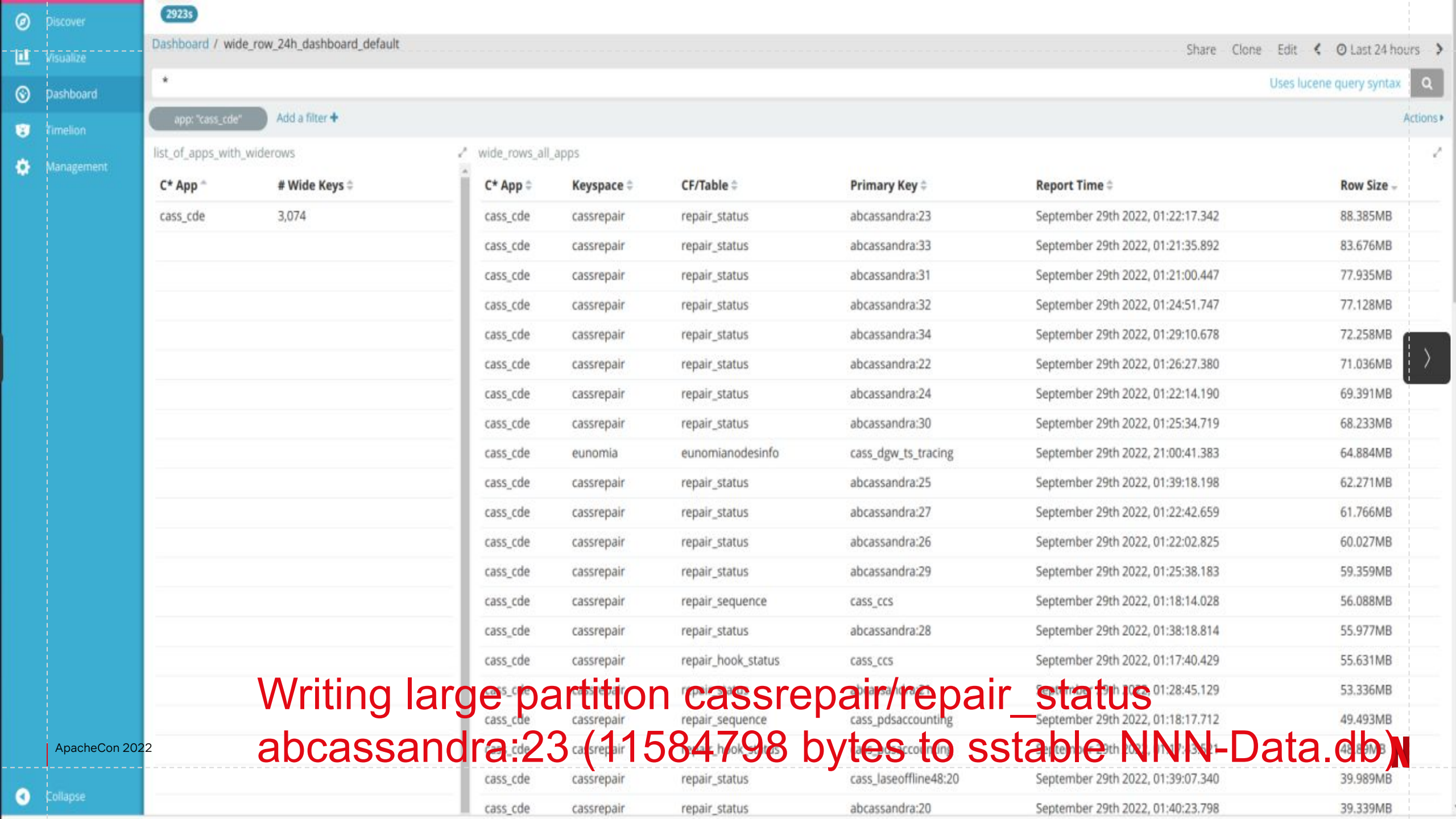
What Makes Bad Partitions Bad

- CPU Usage
- Memory Usage
  - Buffer allocations
  - Object overhead
  - GC
- Reading More Files
- Compaction Impacts
- Cascading Read Latency

How We Improve?

Identification





Writing large partition cassrepair/repair_status abcassandra:23 (11584798 bytes to sstable NNN-Data.db)

```
$ sstabledump me-1176537-big-Data.db -k josephl_test
[
 {
  "partition" : {
   "key" : [ "josephl_test" ],
   "position" : 0
  },
  "rows" : [
   {
    "type" : "row",
    "position" : 26,
    "clustering" : [ "0x68656c6c6f0a" ],
    "liveness_info" : { "tstamp" : "2021-10-23T00:17:11.252097Z" },
    "cells" : [
     { "name" : "value", "value" : "0xb068656c6c6f20776f726c64" },
     { "name" : "value_metadata", "value" : "0x" }
    ]
   }
  ]
 }
]
```

```
$ nodetool getendpoints foo bar josephl_test
```

```
$ nodetool getsstables foo bar josephl_test
```

```
$ sstabledump …. | grep row | wc -l
```

```
$ sstablemetadata foo bar ….
```

nodetool getsstables -l

- Small extension to nodetool getsstables
- Only works for LeveledCompactionStrategy
- Helps identify how partitions are spread across sstables

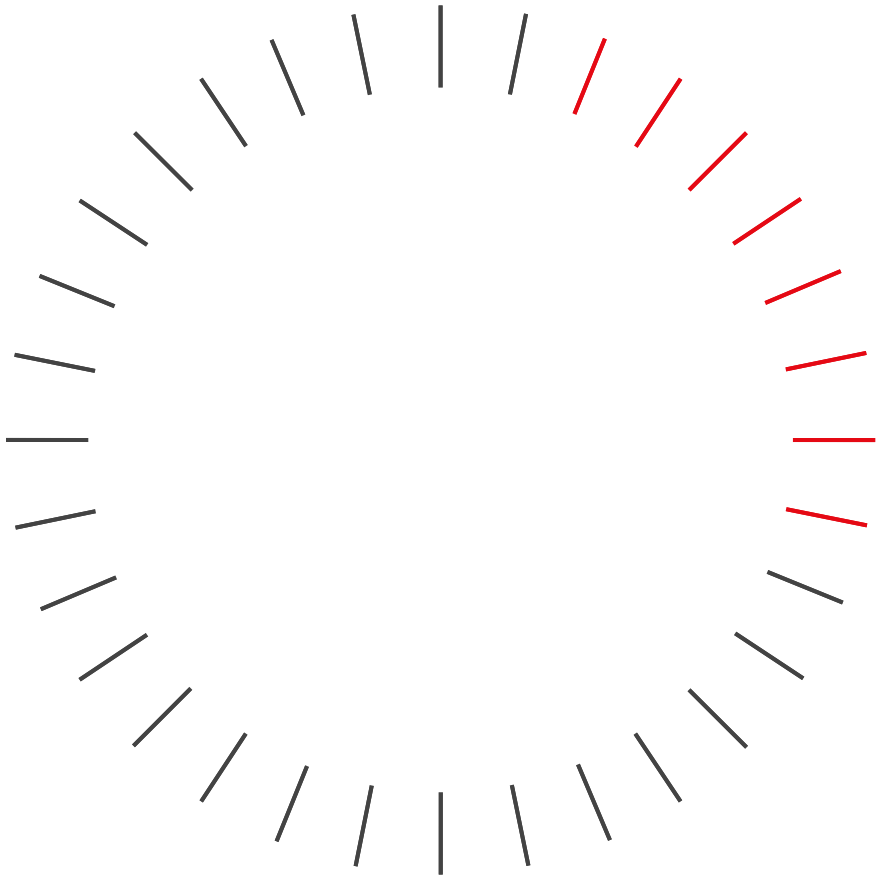
nodetool toppartitions -m MAX

- Extension to top partitions to find TopK in addition to counts
- Latency
- SSTables
- Rows
- Tombstones

fix intermittent failure in top partitions

- CASSANDRA-17254
- Incorrect use of ByteBuffers led to intermittent formatting errors when outputting top partitions results
- Would often cause a delay in our ability to identify problematic partitions

Live Demo



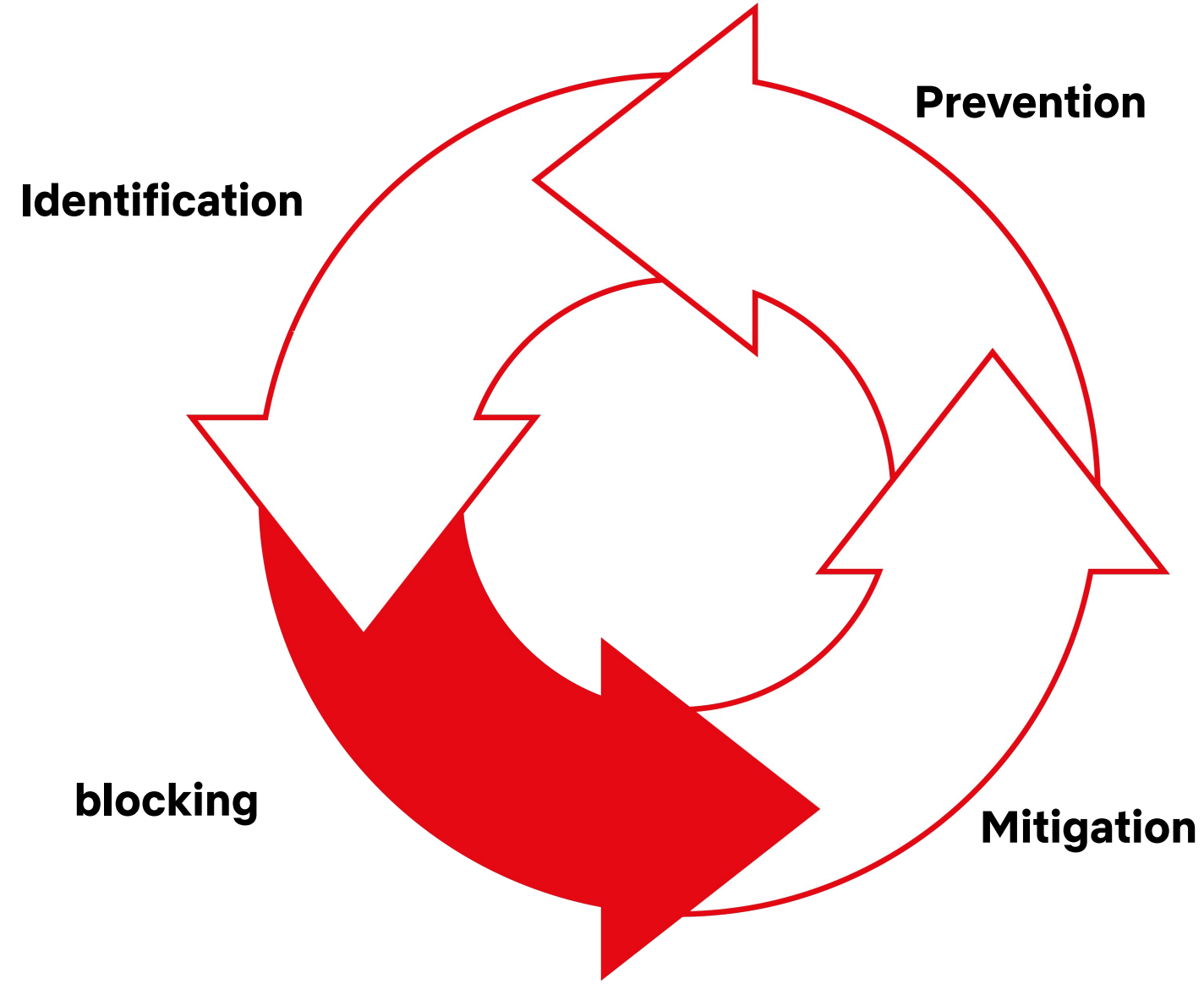
$ nt toppartitions -m MAX -a TOMBSTONES marken_01 startedannotationoperationid 10000
TOMBSTONES Max Sampler:
Top 10 partitions:
Partition Max
STARTED 5911

$ cqlsh
[cqlsh 5.0.1 | Cassandra 3.0.26.1 | CQL spec 3.4.0 | Native protocol v3]
Use HELP for help.
cqlsh> ALTER TABLE marken_01.startedannotationoperationid WITH gc_grace_seconds=3600;

$ nt compact marken_01 startedannotationoperationid

$ nt toppartitions -m MAX -a TOMBSTONES marken_01 startedannotationoperationid 10000
TOMBSTONES Max Sampler:
Top 10 partitions:
Partition Max
STARTED 2970

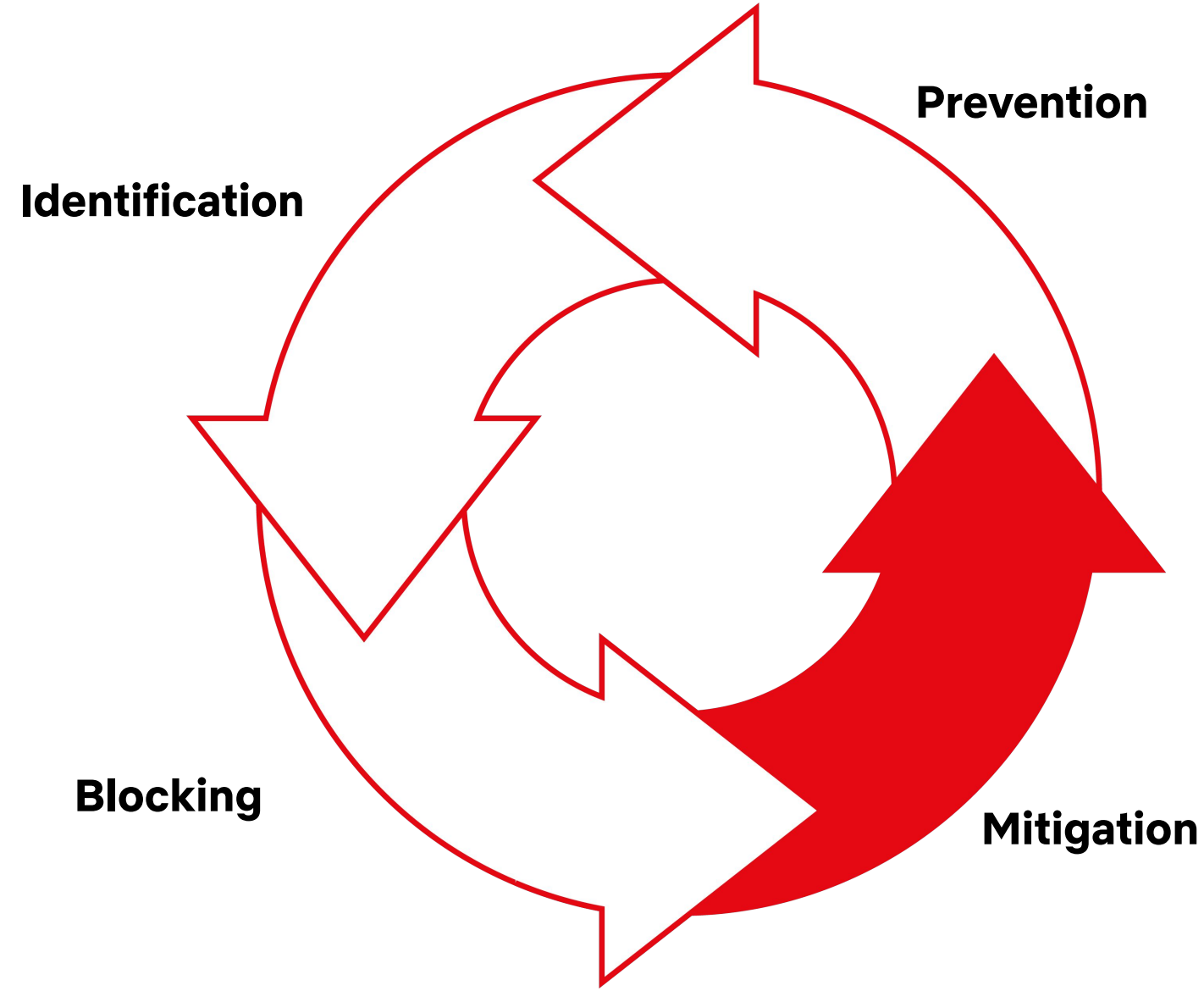
Blocking



Partition blocklist

- Backport **CASSANDRA-12106:** add ability to blocklist / denylist a CQL partition so all requests are ignored
- Prevent reads, range reads and writes (configurable) to given partition keys
- Write to system table via client or JMX to add or remove denied partitions
- Provides a tool to operators to control undesirable application behavior

Mitigation

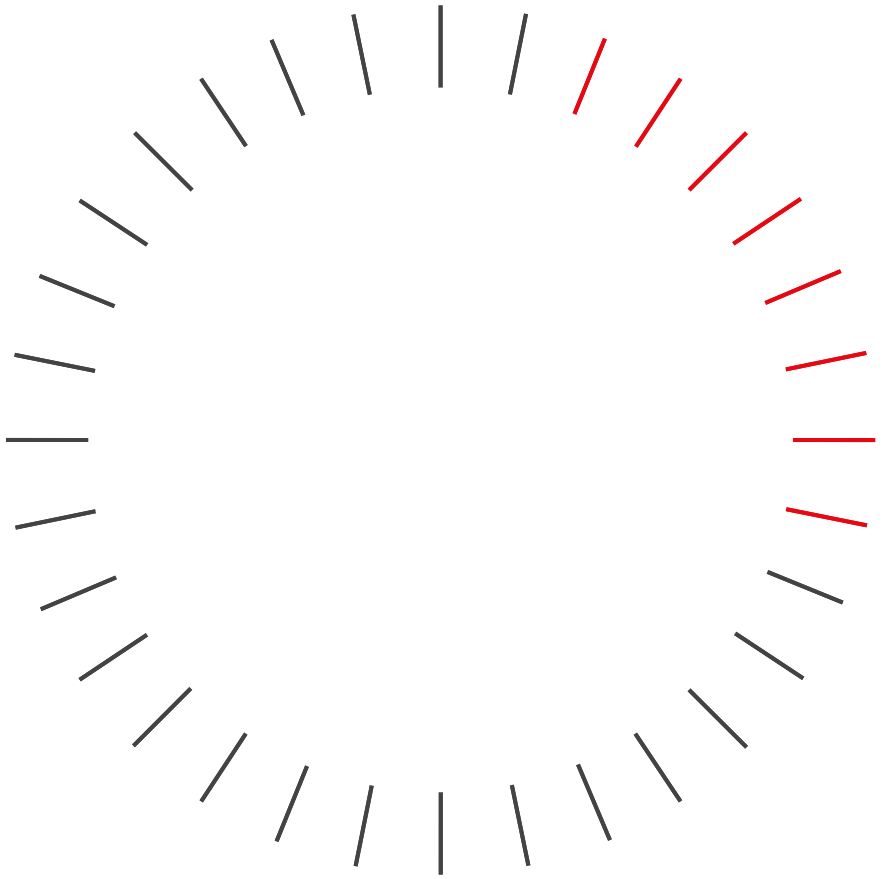


nodetool force compact

```
$ nodetool
forcecompact
keyspace table
list_of_partition_keys
```

- New tool developed internally - Cassandra
- Previously, we reduced the **gc_grace_seconds** and ran on the **nodetool compact**
  - Took hours even days to scan all the sstables
  - Ran at the risk of deleted(tomestone) data reappearing
- Force compact
  - Only scans the sstables for the keys given
  - Help to quickly mitigate the bad partitions and avoid scanning the whole table
  - **Ignore gc_grace_seconds:** For keys where we know it is safe to remove tombstoned or TTL'd data
- Target for OSS 4.x

Live Demo



Thank You.

Jordan West (jwest@apache.org)
Cheng Wang (superwangcheng@gmail.com)