- Cassandra for Time Series Data -

Joris Gillis, June 28, 2017

Joris Gillis

I am a software engineer at TrendMiner and focus on the enterprise scalability of our industrial analytics platform.

I studied at the University of Hasselt and the University of Antwerp in the field of Database theory and Data Mining.

My interests are:

- Big Data technology
- Functional Programming
- Athletics

TrendMiner - Agenda

1. Introduction
2. Why Cassandra?
3. How to model time series in Cassandra?
4. How to configure Cassandra for Time Series Data?
5. Q&A

TrendMiner

TrendMiner is the Leading Modelling Free Industrial Analytics Platform to Analyze, Monitor and Predict Asset and Process Performance.

With a proven track record in the (Petro-) Chemical and Oil & Gas industry to increase overall profitability, by improving production yield, lower costs, avoid unplanned process downtime, increase overall equipment efficiency and reduce safety risks.

About our company

- Started in 2008 as Spin-off K.U.Leuven
  - > 70 Man Year research behind TrendMiner
  - Spin-out Idea from Bayer MaterialScience (Covestro)
  - Patented several core technologies for US/EU

- Headquarter EMEA, Hasselt, Belgium
- Headquarter US, Houston, TX
- 60+ Employees and growing

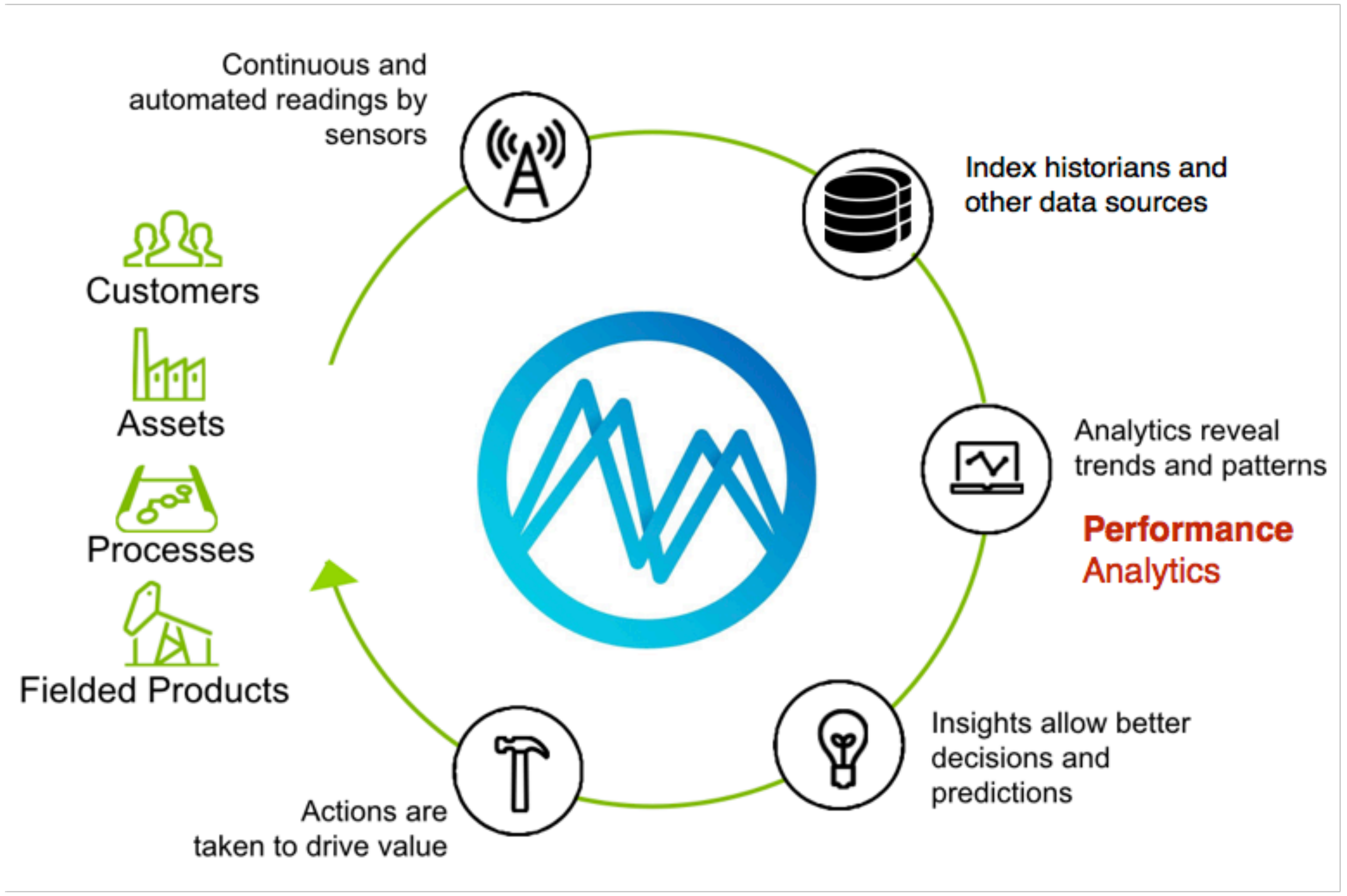
- Global OSIsoft PI ISV & OEM partner
- Platform Agnostic vendor
- Front runner in both Process & Asset Analytics

TrendMiner - Industry 4.0

CONNECTIVITY	Internet of Things	Wearables	Augmented Reality
BIG DATA & ANALYTICS	Optimization & Prediction	Machine Learning	Cyber Security
ADVANCED MANUFACTURING	Additive Manufacturing	Advanced Materials	Autonomous Robotics

Technologies that enable new ways of working and of doing business

About our software



Search

High performance and multi-dimensional search on process data

Ex. Search for identical patterns, correlations, drift, slopes, boolean conditions,…

Diagnose

Self-service and model free advanced diagnostics

Ex. Finding root causes, influence factors, comparing events,…

Capture

Capturing human input on process events in combination with automatic labeling of bookmarked situations

Monitor

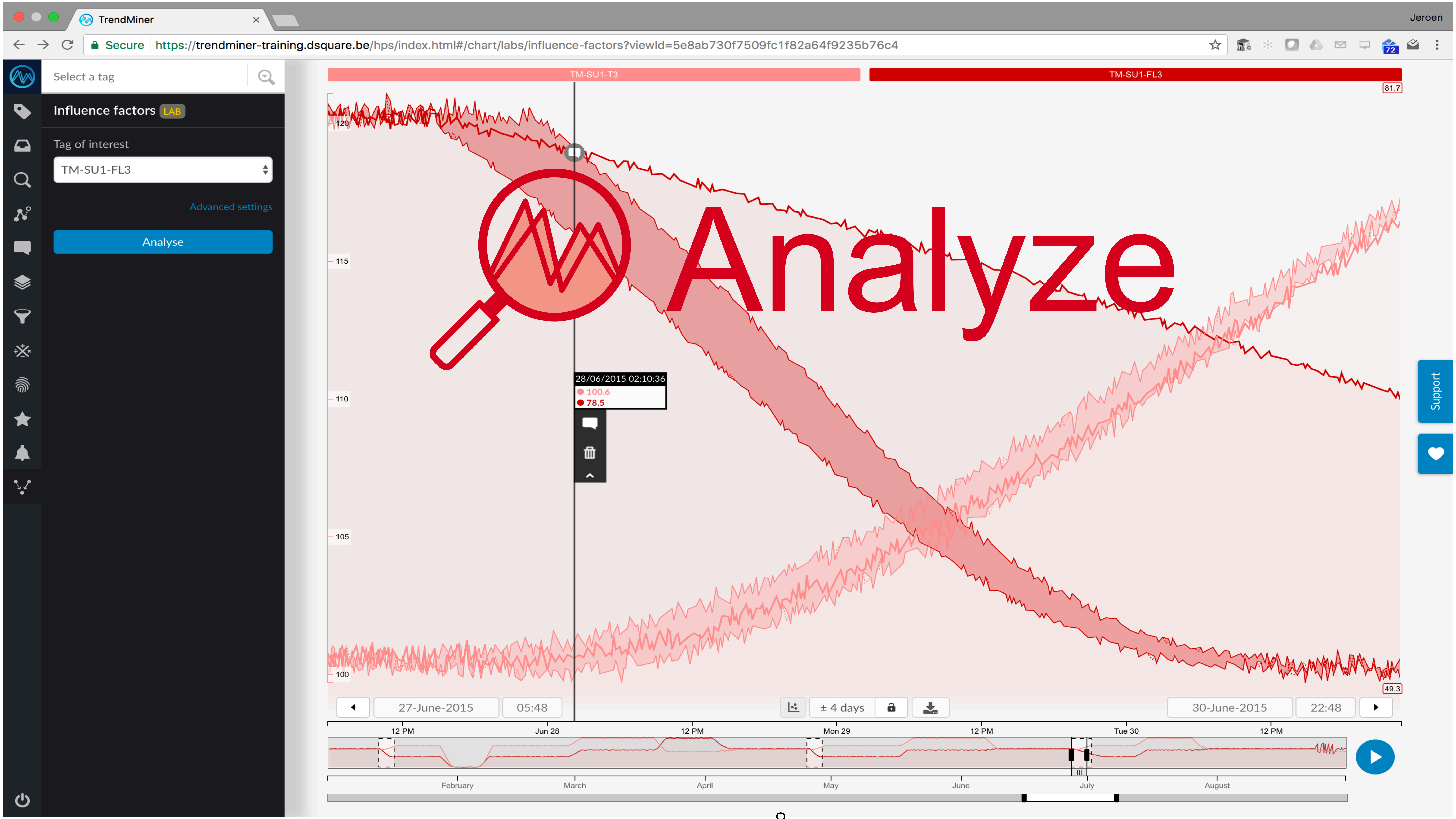
Monitoring for both simple and complex patterns

Ex. Golden Batch monitoring, Asset Health Monitoring,…

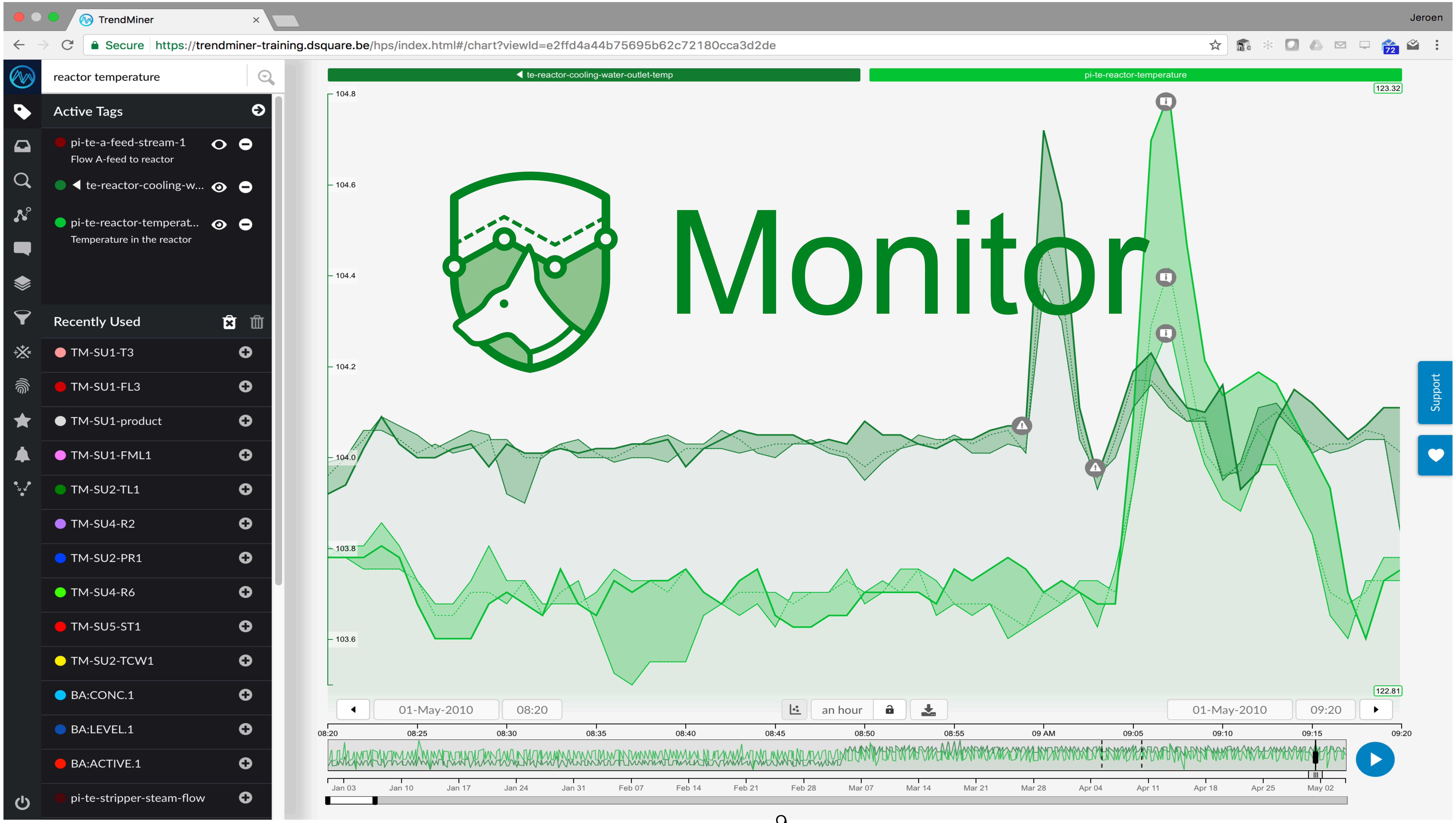
Predict

Predicting asset performance with process contextualisation

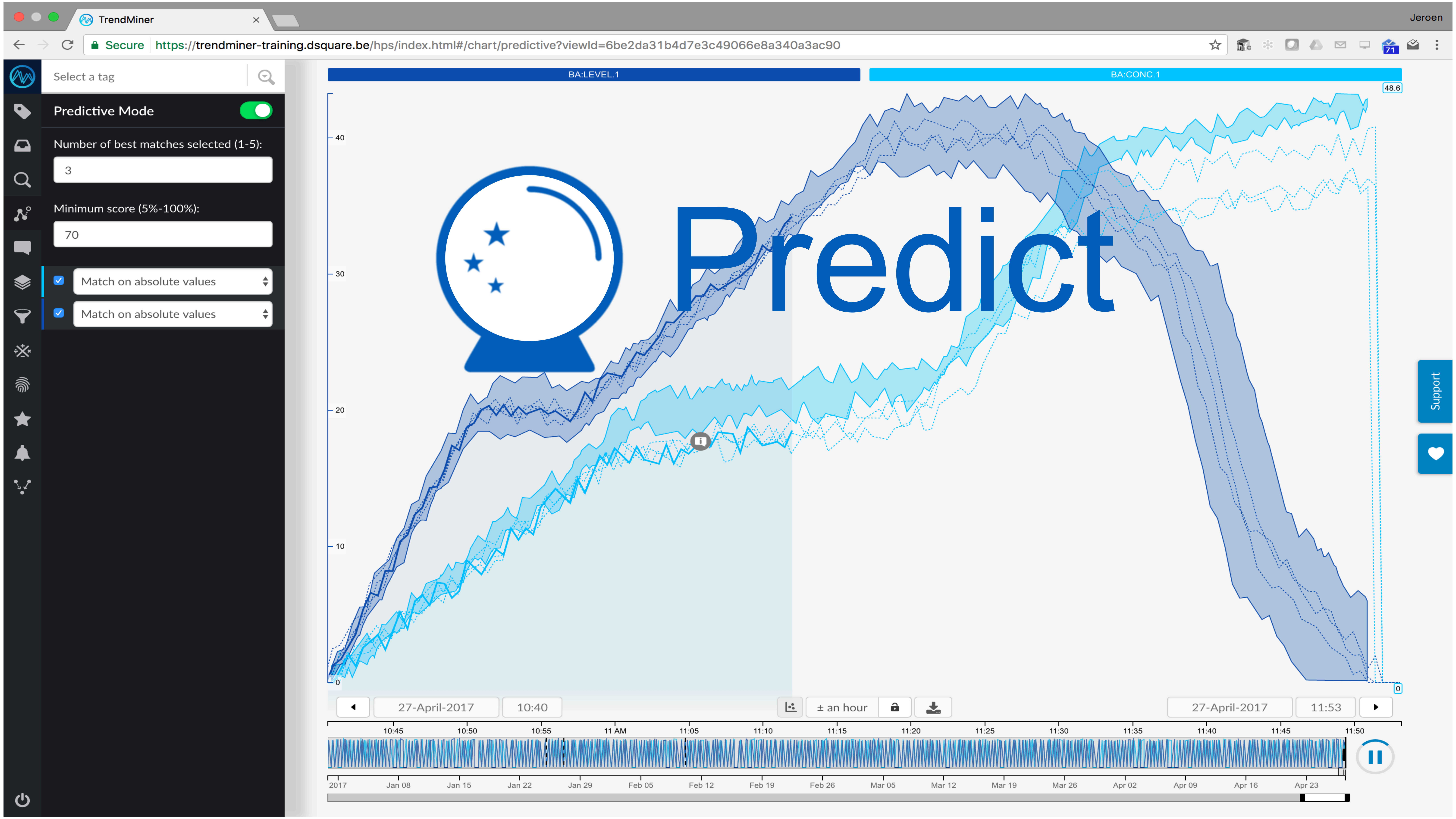
TrendMiner - About TrendMiner



TrendMiner - About TrendMiner



TrendMiner - About TrendMiner



TrendMiner - Problem statement

1 | Data

2 | Information

Real-time Decision Support

3 | Intelligence

Business/Operation Intelligence

Time series

- From thousands to millions
- Resolution between 5 minutes and 1 second
- E.g., 10 year history

Complex analyses

Plants across the globe

- A **time series** is a series of timestamped data points
  - Sometimes data points are spaced equidistantly
- `List<Tuple<Long, Float>>`

TrendMiner - Agenda

1. Introduction
2. **Why Cassandra?**
3. How to model time series in Cassandra?
4. How to configure Cassandra for Time Series Data?
5. Q&A

TrendMiner - Why Cassandra

KairosDB

elastic

influxdb

↓

In-store analytics too limited for our needs

Only HTTP interface

New technology

Horizontal scaling

Big Data => Big Index

↑

PostgreSQL

TIMESCALE

TrendMiner - Why Cassandra

- **ADVANTAGES**
  - Proven technology
  - Connectivity
    - JDBC connector (also to Spark)
  - Edge locations
    - Geographic distribution of data
  - Support for storing and querying time series data
    - E.g., KairosDB uses Cassandra as underlying store

- **DISADVANTAGES**
  - Overhead vs custom optimised format
  - No time series specific optimisations
    - E.g., Gorilla/Beringei
    - Delta-delta encoding for semi-equidistant points
    - Delta encoding for stable values

TrendMiner - Agenda

1. Introduction
2. Why Cassandra?
3. **How to model time series in Cassandra?**
4. How to configure Cassandra for Time Series Data?
5. Q&A

Keys

- Primary key
  - One or more columns identifying a row
    - PRIMARY KEY (A)
    - PRIMARY KEY (A, B)
      - Compound primary key
- Partition key
  - First column(s) of primary key
  - E.g., PRIMARY KEY ((A, B), C)
    - A & B are composite partition key

Partitioning & Clustering

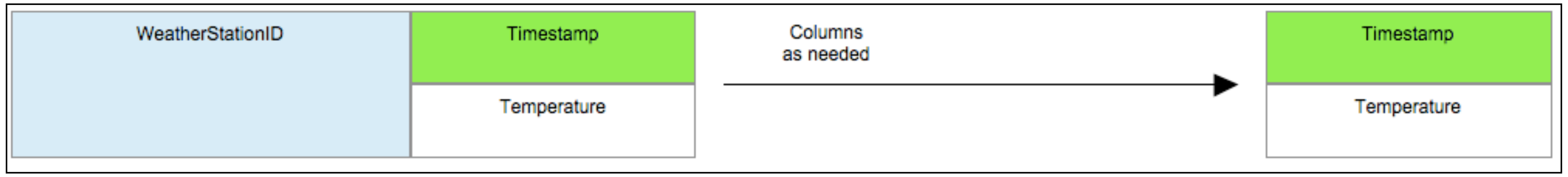
- A partition is mapped to a Cassandra node
  - All rows with same partition key on same node(s)
- Clustering columns
  - Part of compound primary key
  - Define sorting inside partition

```
Map<byte[], SortedMap<Clustering, Row>>
```

Partition Key **Clustering columns** **Other columns**

Modelling: Simple

```
CREATE TABLE temperature (
  weatherstation_id uuid,
  event_time timestamp,
  temperature float,
  PRIMARY KEY (weatherstation_id, event_time)
);
```



https://academy.datastax.com/resources/getting-started-time-series-data-modeling

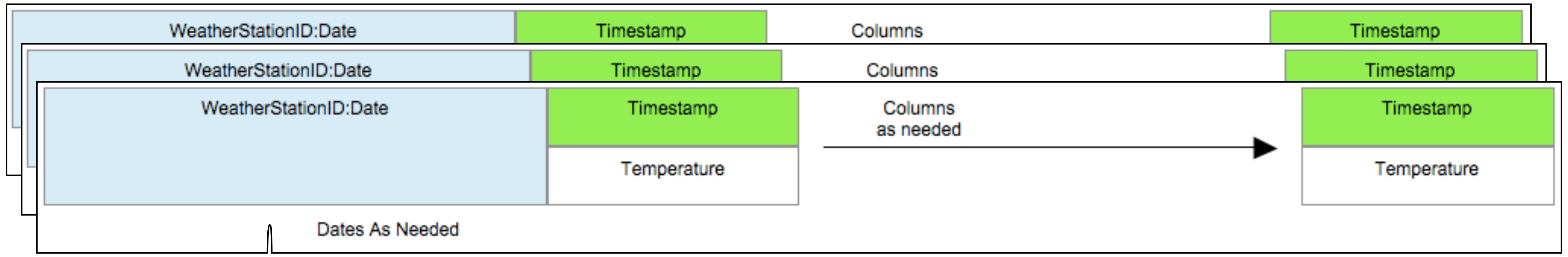
Modelling: Simple

- Advantage
  - Easy to understand
  - Simple to query
- Disadvantage
  - All data for one time series in one partition
  - Max 2 billion rows per partition

```
SELECT temperature
FROM temperature
WHERE weatherstation_id='1234ABCD'
AND event_time > '2013-04-03 07:01:00'
AND event_time < '2013-04-03 07:04:00';
```

Modelling: Partitioned

```
CREATE TABLE temperature_by_day (
  weatherstation_id uuid,
  day date,
  event_time timestamp,
  temperature float,
  PRIMARY KEY ((weatherstation_id, date), event_time)
);
```



https://academy.datastax.com/resources/getting-started-time-series-data-modeling

Modelling: Partitioned

- Advantage
  - Virtually no storage limitation
- Disadvantage
  - Crossing bucket boundary => multiple queries
    - Need to specify id and day; otherwise unpredictable performance
  - If data comes in burst => uneven partition sizes

TrendMiner - Agenda

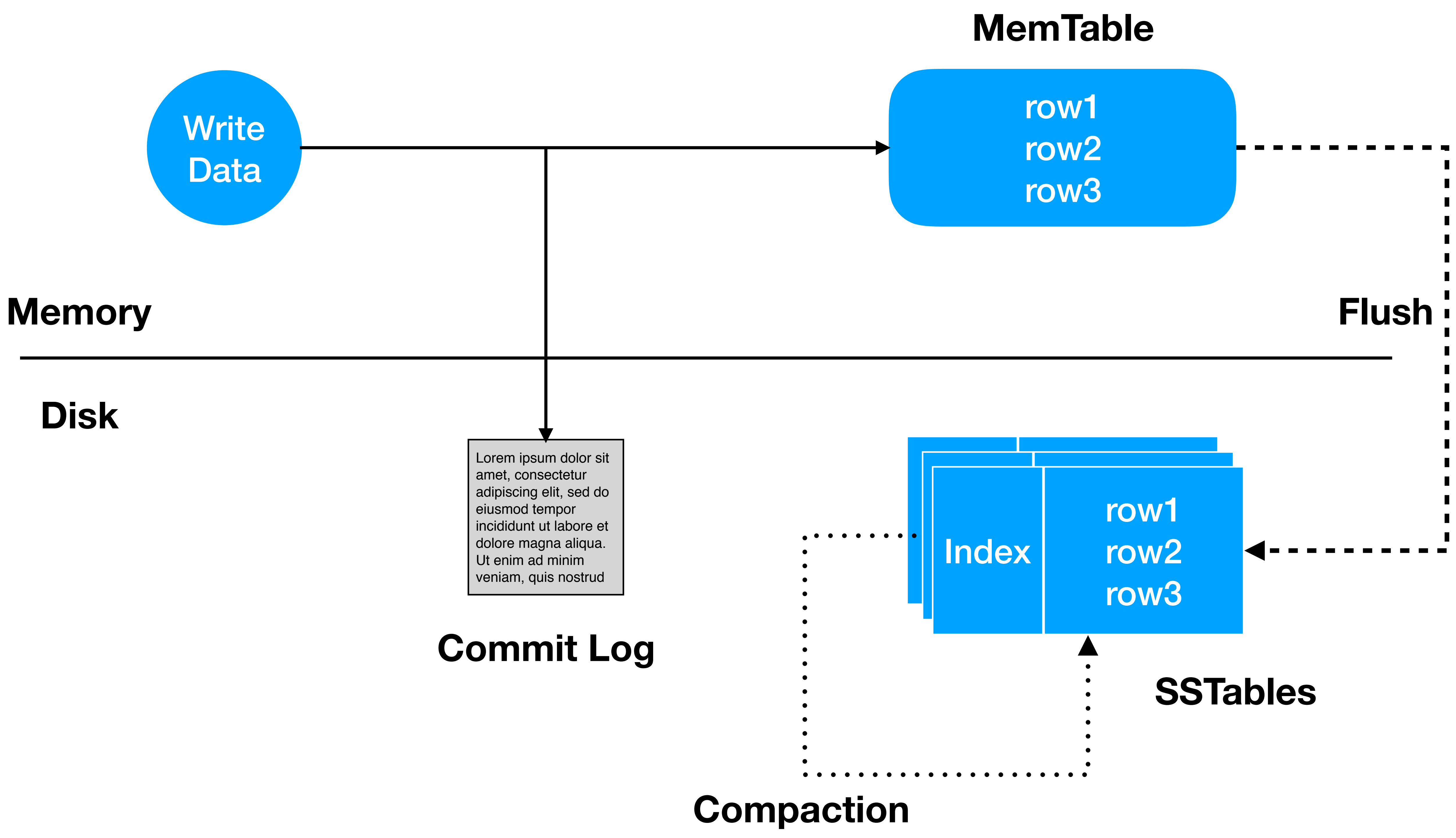
1. Introduction
2. Why Cassandra?
3. How to model time series in Cassandra?
4. **How to configure Cassandra for Time Series Data?**
5. Q&A

TrendMiner - How to configure Cassandra for Time Series Data?

- Cassandra v3
  - Storage engine refactored compared to v2
- Options to influence read and write performance
  - Compression
  - **Compaction**

How Cassandra Writes Data

- Commit log
  - Durability
- Memtable
  - Cache writes in memory
  - Regularly flushed to disc
- Sorted Strings Table (SSTable)
  - Compaction
    - Re-organise
    - Cleanup



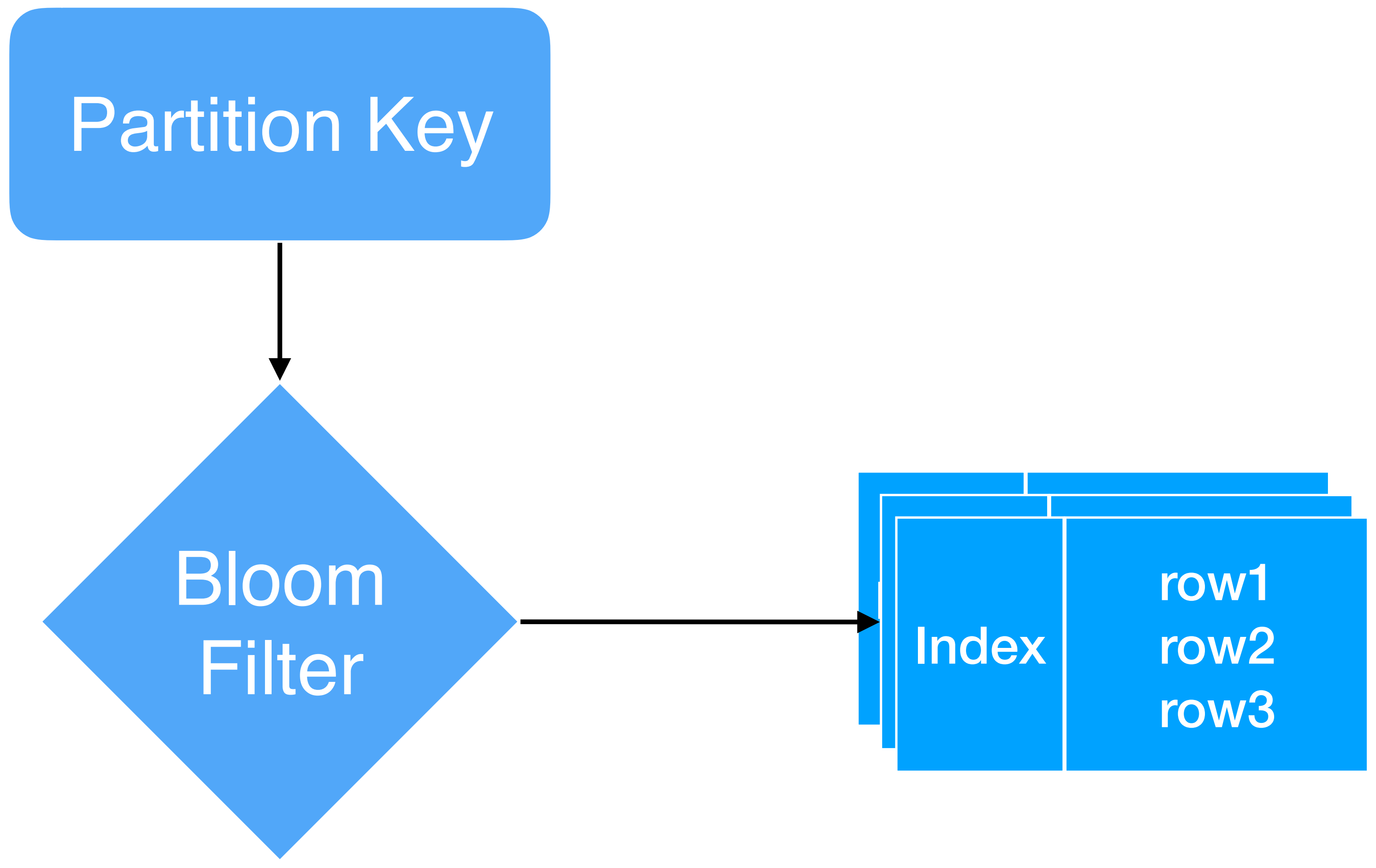
http://docs.datastax.com/en/cassandra/3.0/cassandra/dml/dmlHowDataWritten.html

How Cassandra Maintains Data

- SSTable = immutable
  - Updates => Timestamped version of row
  - Deletes => Tombstones
- Clean up old versions and tombstones
  - Compaction

How Cassandra Reads Data (Simplified)

- How to get relevant data
  - SSTables
    - Partition key => node(s)
    - Bloom filter
      - Returns list of SSTables that *might contain* rows for query
    - Partition index on SSTable
      - Keeps offset for each partition key
  - Memtable
    - Extract qualifying rows
- Resolve
  - Timestamped rows
  - Tombstones



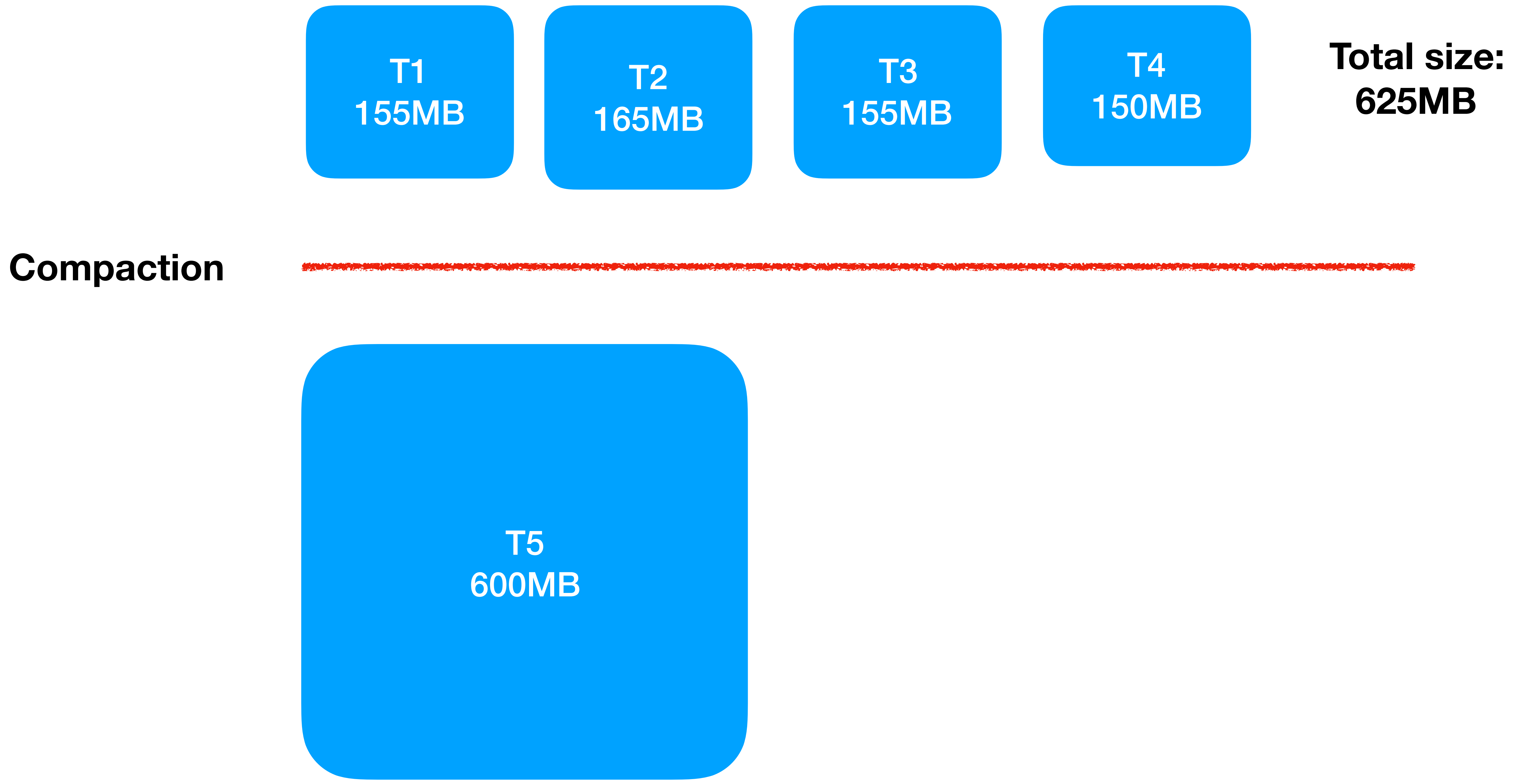
Compaction options

- Size tiered compaction (default)
- Levelled compaction
- Time window compaction

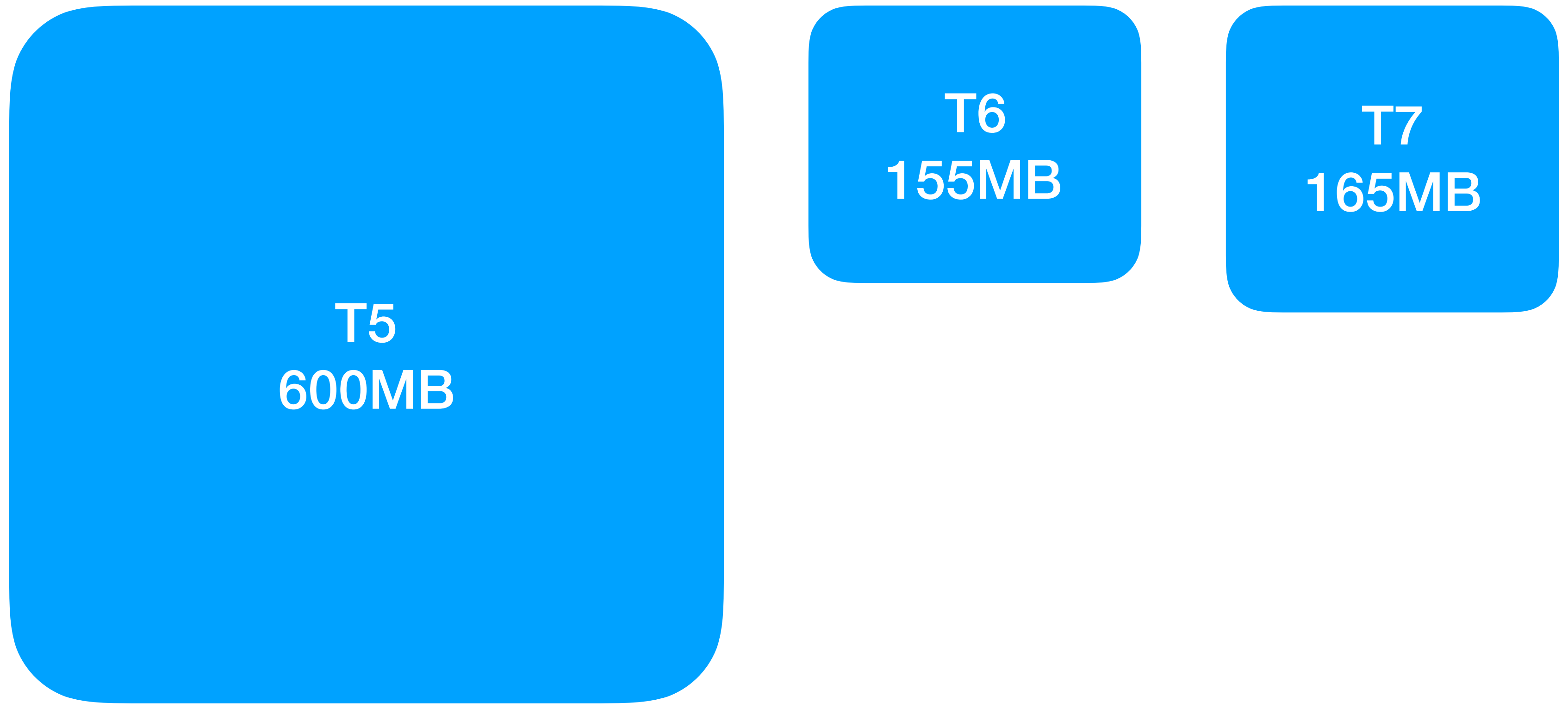
Size tiered Compaction

- Default strategy
  - Optimised for write heavy workloads
- Compaction
  - When # similarly sized SSTables
  - Merge into one new file

Size tiered Compaction Example



Size tiered Compaction Example



Size tiered Compaction

- Advantage
  - Write optimised
- Disadvantage
  - Rows of a partition are spread across multiple SSTables
  - Holds on to stale data for a long time
  - A lot of memory needed as SSTables grow in size

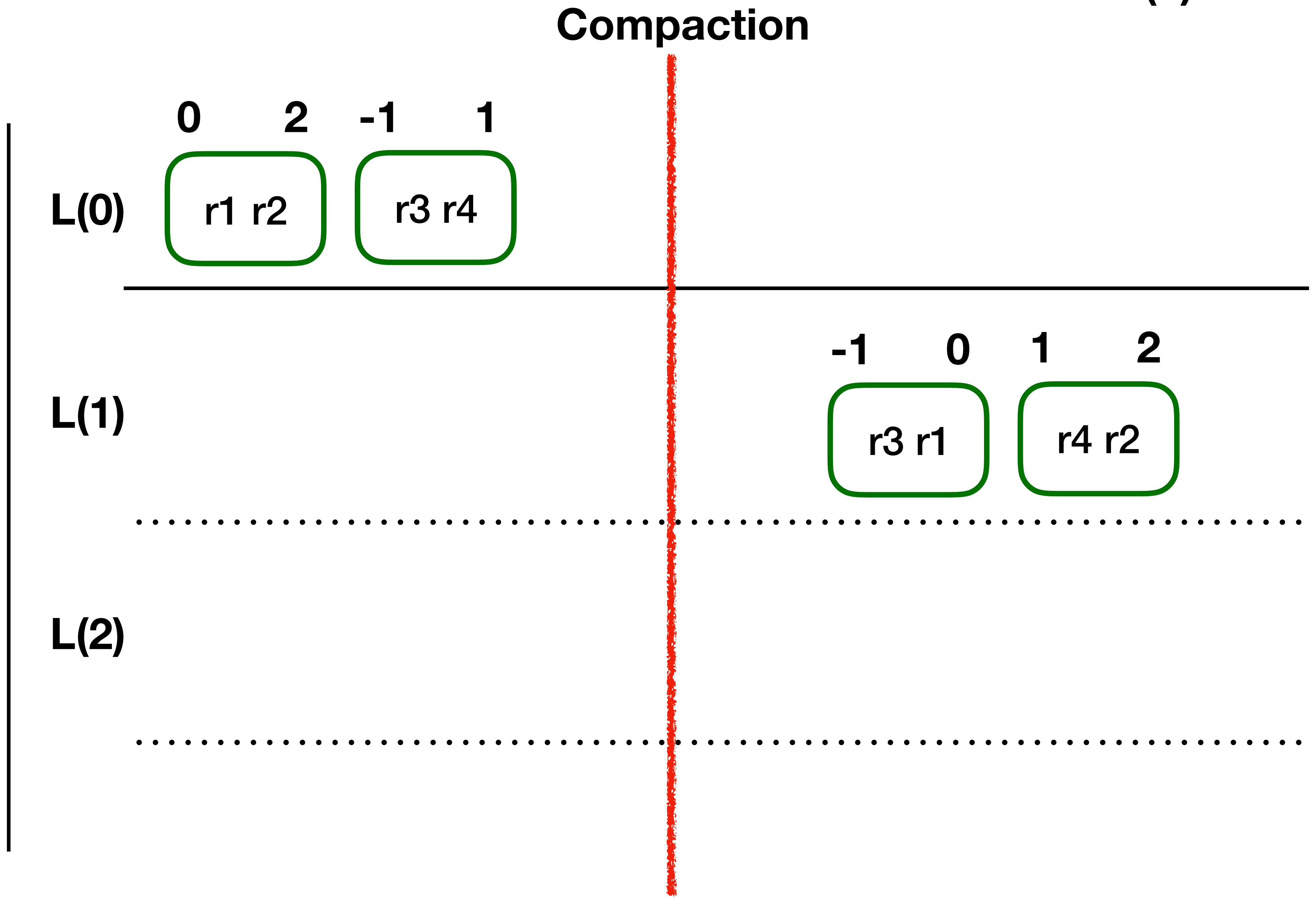
Levelled Compaction

- L(0)
  - Flushes from memtable
- L(N > 0)
  - Fixed size SSTables (default: 160MB)
  - Each SSTable has range of partitions => NO OVERLAP!
  - L(1) holds at most 10 SSTables
  - L(N+1) can hold 10x more SSTables than L(N)

Levelled Compaction Example

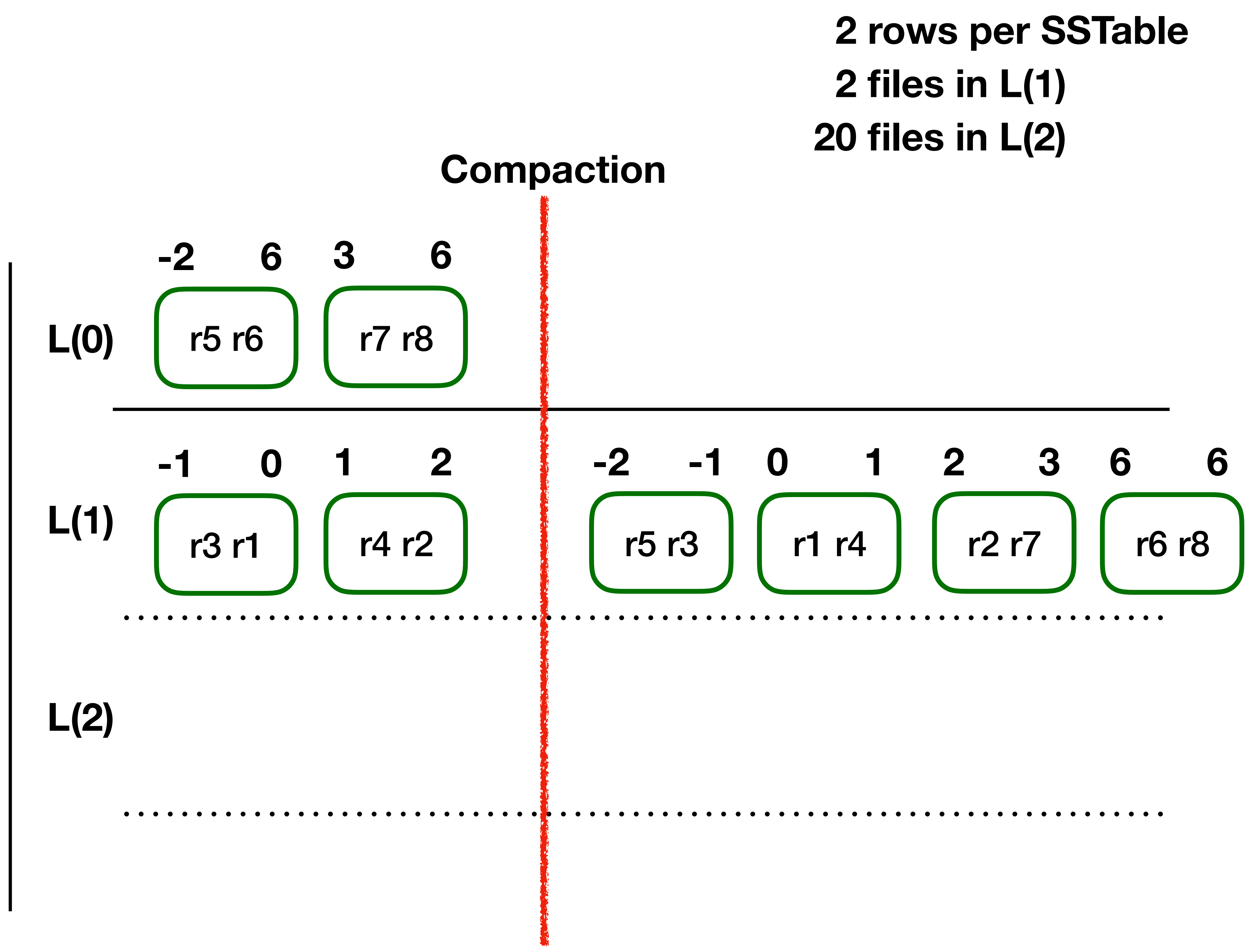
2 rows per SSTable
2 files in L(1)
20 files in L(2)

Row	Partition
r1	#0
r2	#2
r3	#-1
r4	#1
r5	#-2
r6	#6
r7	#3
r8	#6
r9	#-1
r10	#-2



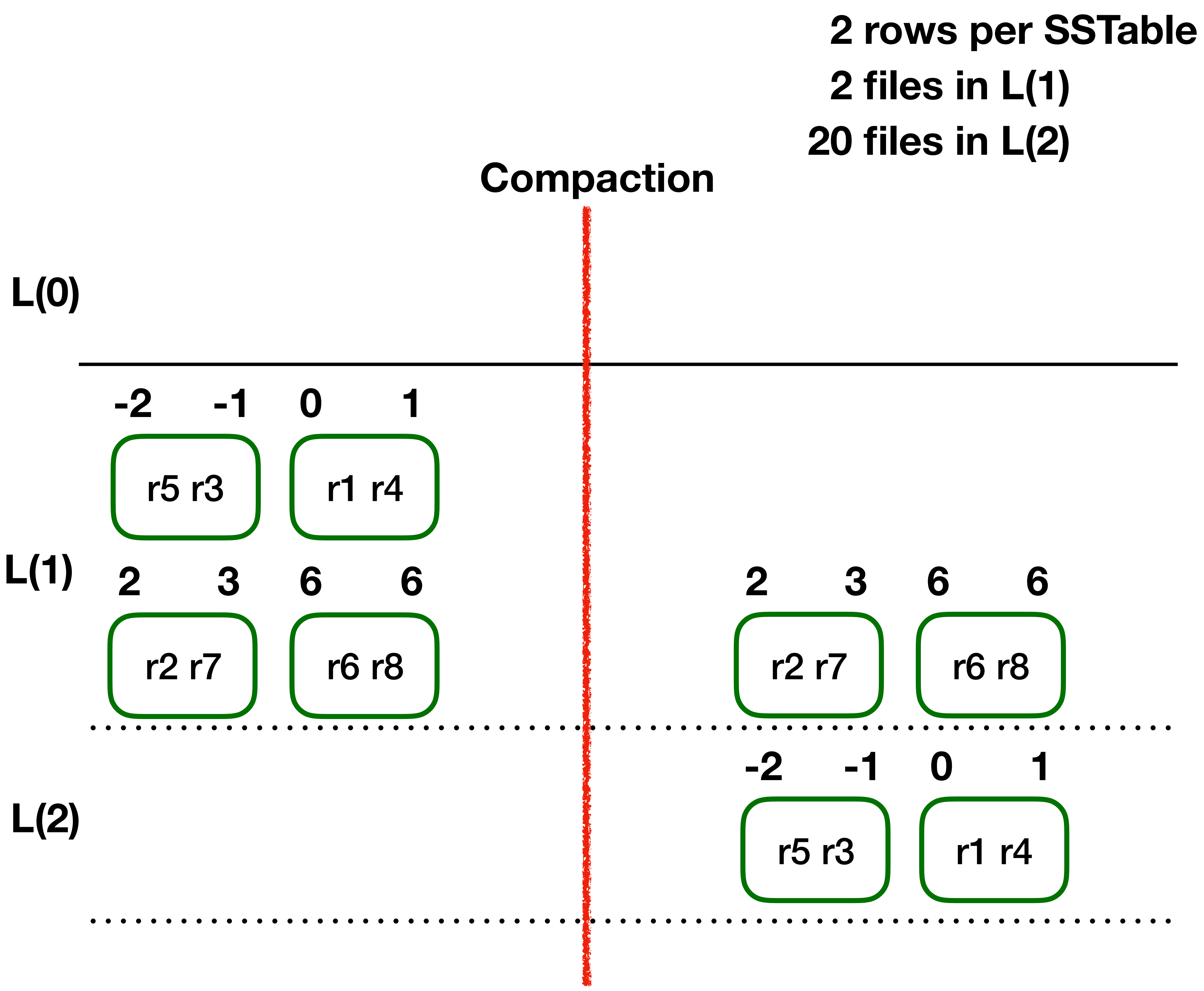
Example ctd.

Row	Partition
r1	#0
r2	#2
r3	#-1
r4	#1
r5	#-2
r6	#6
r7	#3
r8	#6
r9	#-1
r10	#-2



Example ctd.

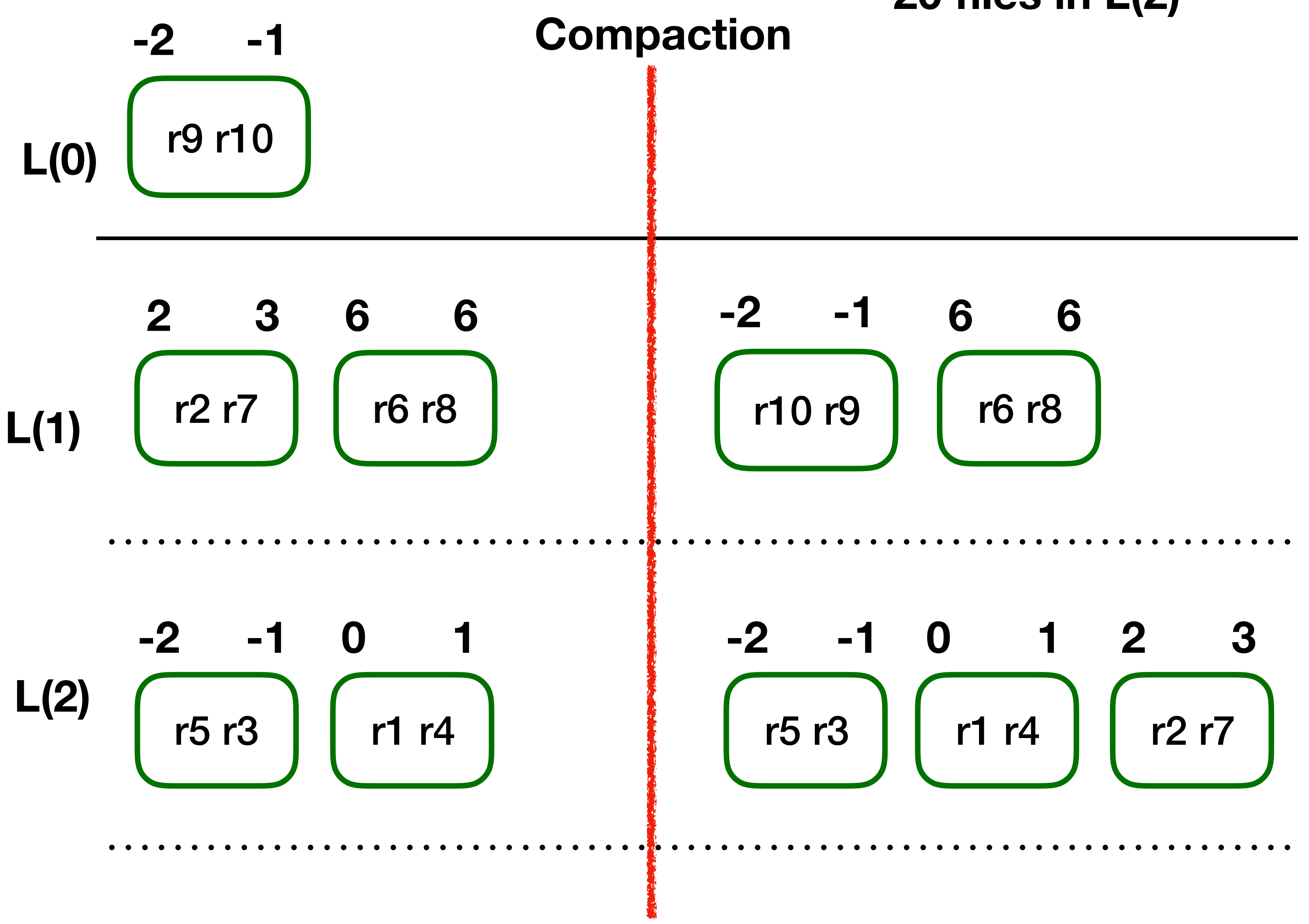
Row	Partition
r1	#0
r2	#2
r3	#-1
r4	#1
r5	#-2
r6	#6
r7	#3
r8	#6
r9	#-1
r10	#-2



Example ctd.

2 rows per SSTable
2 files in L(1)
20 files in L(2)

Row	Partition
r1	#0
r2	#2
r3	#-1
r4	#1
r5	#-2
r6	#6
r7	#3
r8	#6
r9	#-1
r10	#-2



Levelled Compaction Configuration Trade-off

- max_sstable_size (default: 160MB)
  - From L1 on, all SSTable are of this size
  - **One partition must fit in one SSTable!**
- Bigger SSTable size:
  - Fewer levels => Faster reads
- Smaller SSTable size:
  - Faster compaction
  - Write amplification

Level	Max Data size
L(1)	1.6GB
L(2)	16GB
L(3)	160GB
L(4)	1.6TB

Levelled Compaction Conclusion

- Advantages
  - No overlap => at most 1 SSTable per level for a partition => Consistent read performance
  - Quicker removal of tombstones
- Disadvantages
  - Max SSTable size must be bigger than biggest partition
  - More CPU and IO used during compaction (write amplification)
- When to use
  - Limited/stable number of partitions
  - Compaction speed > write speed

Time Window Compaction

- Fixed time window
- Compaction at end of time window
  - Older windows never compacted again!
- Ideal for immutable, in order, fixed rate time series
  - No compaction needed
  - ± same sized SSTables
- Downside
  - Disk usage only grows

Compaction Comparison

- Size tiered compaction
  - Heavy write load
- Levelled compaction
  - Read load
- Time window compaction
  - Immutable, equally space time series
- Configurable per table!

Conclusion

- Cassandra is mature and powerful technology
- Devil is in the details
  - Key: understanding of storage engine
  - Modelling
    - Start from queries
  - Configuration
    - Compaction? Compression? Caching? Replication? Heap size?

References

- DataStax
  - Docs: http://docs.datastax.com/en/landing_page/doc/index.html
  - Academy: https://academy.datastax.com/
- Outworkers three part blog post on Cassandra:
  - http://outworkers.com/blog/post/a-series-on-cassandra-part-1-getting-rid-of-the-sql-mentality
  - http://outworkers.com/blog/post/a-series-on-cassandra-part-2-indexes-and-keys
  - http://outworkers.com/blog/post/a-series-on-cassandra-part-3-advanced-features
- TimeScaleDB: https://blog.timescale.com/when-boring-is-awesome-building-a-scalable-time-series-database-on-postgresql-2900ea453ee2
- Beringei: https://github.com/facebookincubator/beringei
- Gorilla In-memory Time Series store: http://www.vldb.org/pvldb/vol8/p1816-teller.pdf
- Time Series overview: https://docs.google.com/spreadsheets/d/1sMQe9oOKhMhIVw9WmuCEWdPtAoccJ4a-IuZv4fXDHxM/edit#gid=0
- LevelledCompactionStrategy: https://www.slideshare.net/DataStax/the-missing-manual-for-leveled-compaction-strategy-wei-deng-datastax-cassandra-summit-2016

TrendMiner

Questions & Answers

TrendMiner

Thank you for attending!

For more information you can contact:

joris.gillis@trendminer.com or

Visit our website www.trendminer.com