instaclustr

Using Apache Cassandra and Apache Kafka to Scale Next Gen Applications

Ben Slater, Chief Product Officer, Instaclustr

Agenda

- The basics of Kafka and Cassandra
- Some example use cases
- Partitioning - the foundation of scale
- Fitting and architecting for your use case

About Instaclustr

- Experts in open source reliability at scale
- Over 50 million node-hours of experience managing Cassandra, Kafka, Spark and Elasticsearch
- Customers from start-ups to international banks and the most popular online games
- Platform providers automated provisioning, monitoring and management
- Available on AWS, GCP, Azure, IBM Cloud and on-prem
- Enterprise support and consulting also offered

Apache Cassandra

Scaleable, Always On, Multi-Region Data Store

Apache Cassandra is the leading NoSQL operational database for high-scale and high-reliability applications

Shared nothing and peer to peer architecture provides reliability with Instaclustr SLAs up to 100%

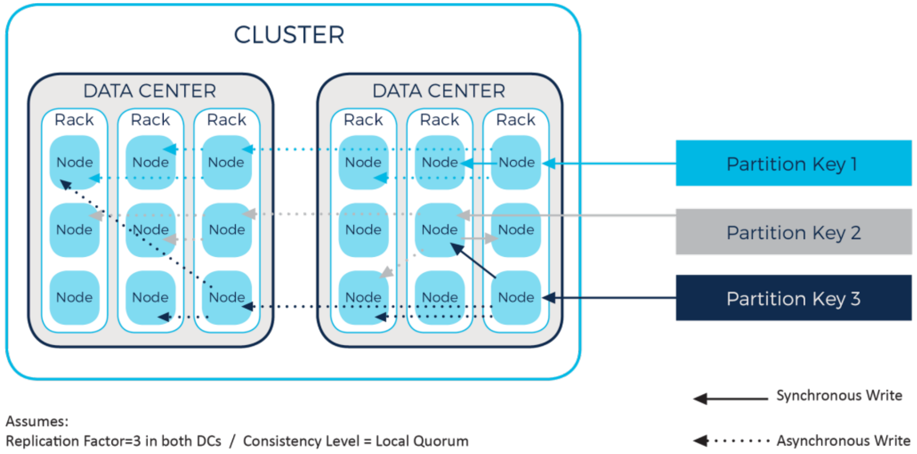
- Replicated data and multiple nodes capable of fulfilling a query means no failover, service just keeps running
- Full online maintenance and upgrades

Low latency for operational applications - < 10ms p95 reads and writes achievable

Native active-active multi data center support for:

- Geographic distribution to meet latency requirements
- Disaster resilience
- Workload isolation (analytic DCs)

In your application architecture, Cassandra is for storing and retrieving data, not an analytic query engine or place to run logic (stored procs).



Assumes:
Replication Factor=3 in both DCs / Consistency Level = Local Quorum

Apache Kafka

Queuing and Streaming at Scale

Apache Kafka is a distributed streaming platform meaning it provides three key capabilities:

- It lets you publish and subscribe to streams of records. In this respect it is similar to a message queue or enterprise messaging system.
- It lets you store streams of records in a fault-tolerant way.
- It lets you process streams of records as they occur or randomly access data at any position in the stream.

Apache Kafka has a similar (at a very high level) shared-nothing, replicated architecture to Cassandra which also allows it to operate with similar extreme levels of scalability and reliability

Typical use cases include:

- as a message bus - provide loose coupling between producers and consumers of messages
- as a store of logical transactions for populating a analytical data stores or edge caches
- as a buffer to manage back pressure is systems subject to workload spikes; and
- (along with Kafka Streams or Spark Streaming) as the basis of a streaming architecture for real-time analytics.

worldpay

Payment processing business spun out of RBS in 2010

- Merged with Vantive in US in Jan 2018 for USD 10.4B to form WorldPay Inc.

Processes:

- >40 Million transactions per day
- for 400,000 merchants
- 42% of all UK non-cash transactions.

worldpay

Re-architecting XML Payment API which facilitates ~40M transactions per month

- New architecture based on open source technologies including Cassandra and Kafka to provide scalability, availability and reduced costs
- New Idempotency service first project to use the new architecture, provides capabilities to ensure payments are not repeated

Challenges

- Tight deployment timeframes
- Very high availability expectations
- Low latency requirements

worldpay

Used Cassandra to provide highest levels of availability and scalability

- 18 Node cluster running over 3 AWS regions in Europe
- Cassandra consistency levels to provide strong consistency across regions while still being able to operate with a whole region unavailable
- Simple data model and atomic reads/writes fits well with Cassandra capability

Used Instaclustr to accelerate development and time to stable service:

- Consulting engagement assisted with data model design
- Cassandra cluster run in Instaclustr Managed Service - production ready in weeks
- Initial preference was to run on-prem for ease of security compliance and did not expect cloud to meet latency requirements. However, timeframes did not allow establishment of internal deployment.
  - Used Instaclustr on AWS for initial go-live and are now satisfied as a long-term solution

Australia’s leading online home loan lender, processing over 90% of Australia’s online lending enquiries.

Implementing a customer and data-centric re-architecture of their platform following a major funding round

Integration-heavy environment due to interfaces with banks, etc.

Moving to microservices with Kafka as a message bus enabled:

- Decoupling application code from embedded data sets from various business applications.
- Unifying data models from the various point solutions and market segments.
- Enabling extensive scale to support rapid and large growth in data as the consumer base grows.

Partitioning

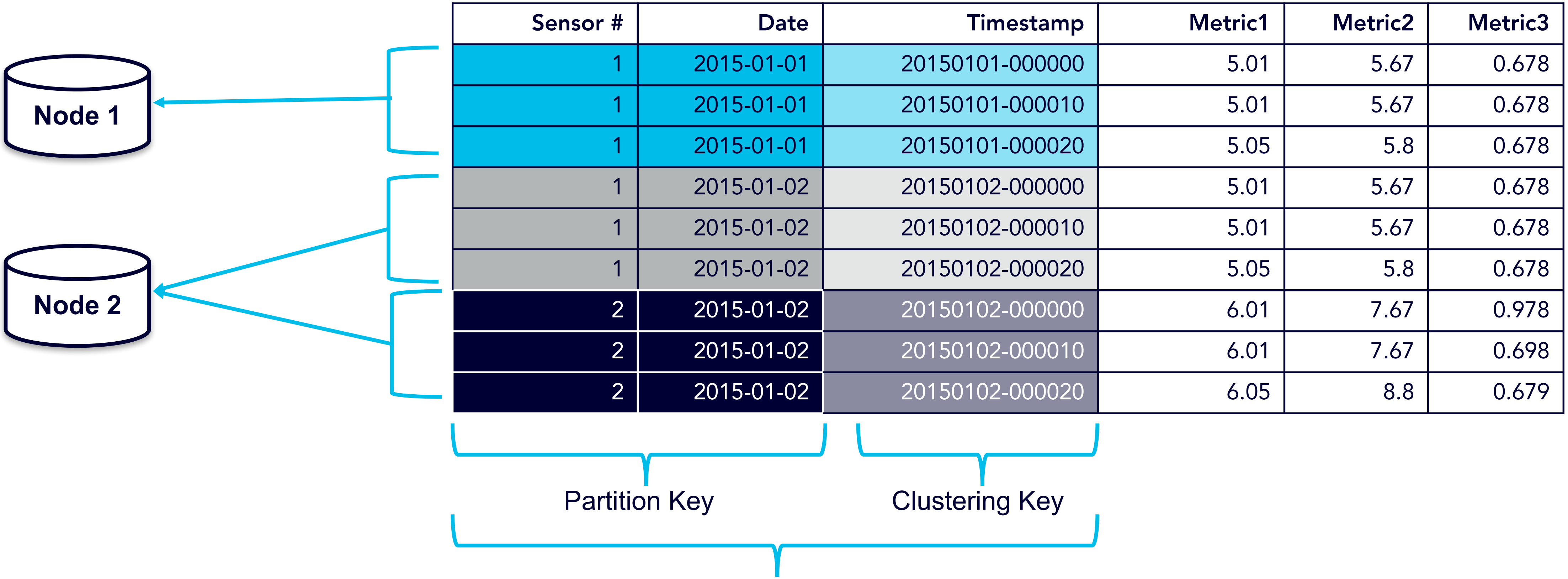
the foundation
of scale

Partitioning = using a key in your data to split the data across multiple servers

Manual partition is possible but painful

Technologies like Cassandra and Kafka make partitioning transparent but ….

Cassandra Partitions



Node 1

Node 2

Sensor #	Date	Timestamp	Metric1	Metric2	Metric3
1	2015-01-01	20150101-000000	5.01	5.67	0.678
1	2015-01-01	20150101-000010	5.01	5.67	0.678
1	2015-01-01	20150101-000020	5.05	5.8	0.678
1	2015-01-02	20150102-000000	5.01	5.67	0.678
1	2015-01-02	20150102-000010	5.01	5.67	0.678
1	2015-01-02	20150102-000020	5.05	5.8	0.678
2	2015-01-02	20150102-000000	6.01	7.67	0.978
2	2015-01-02	20150102-000010	6.01	7.67	0.698
2	2015-01-02	20150102-000020	6.05	8.8	0.679

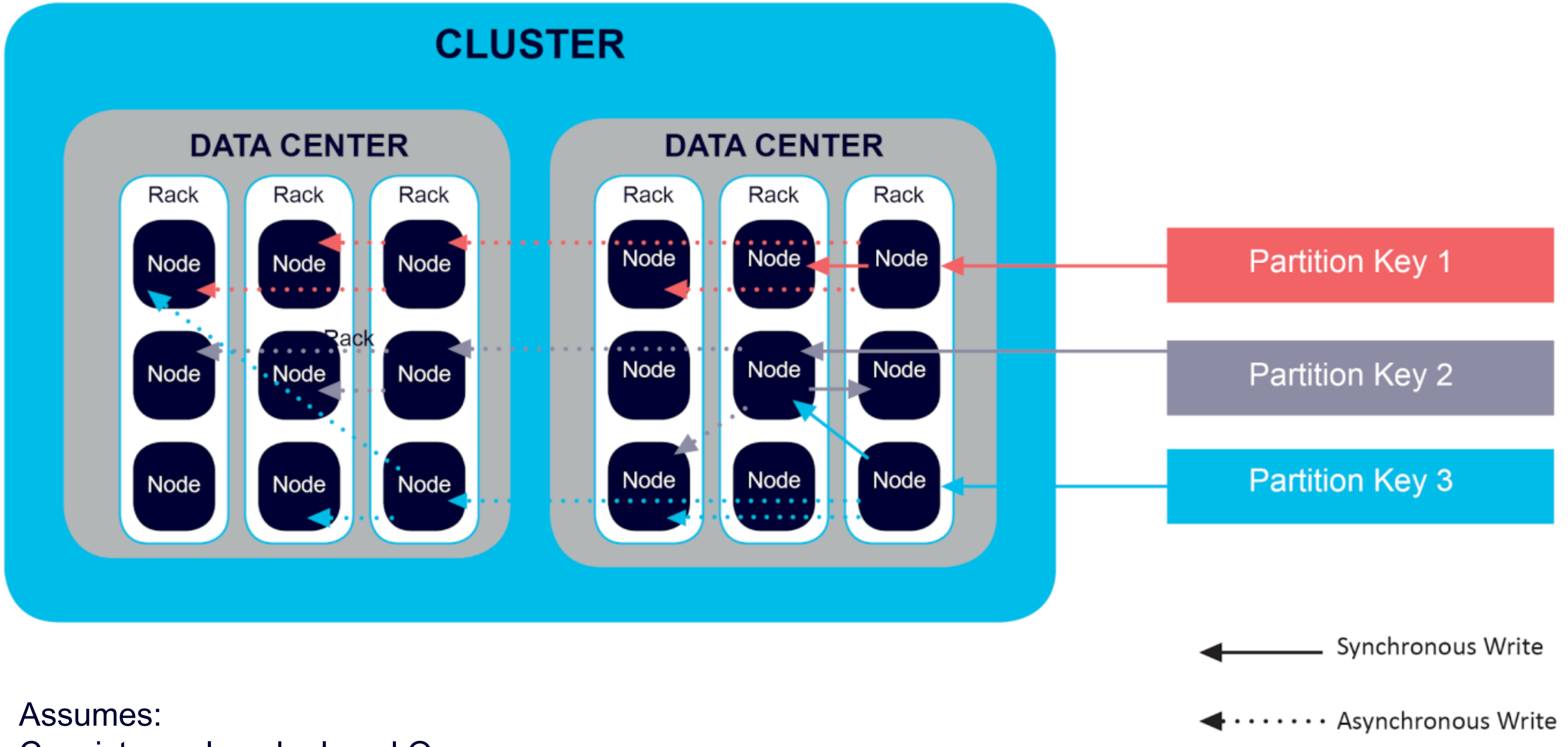
Partition Key

Clustering Key

Primary Key

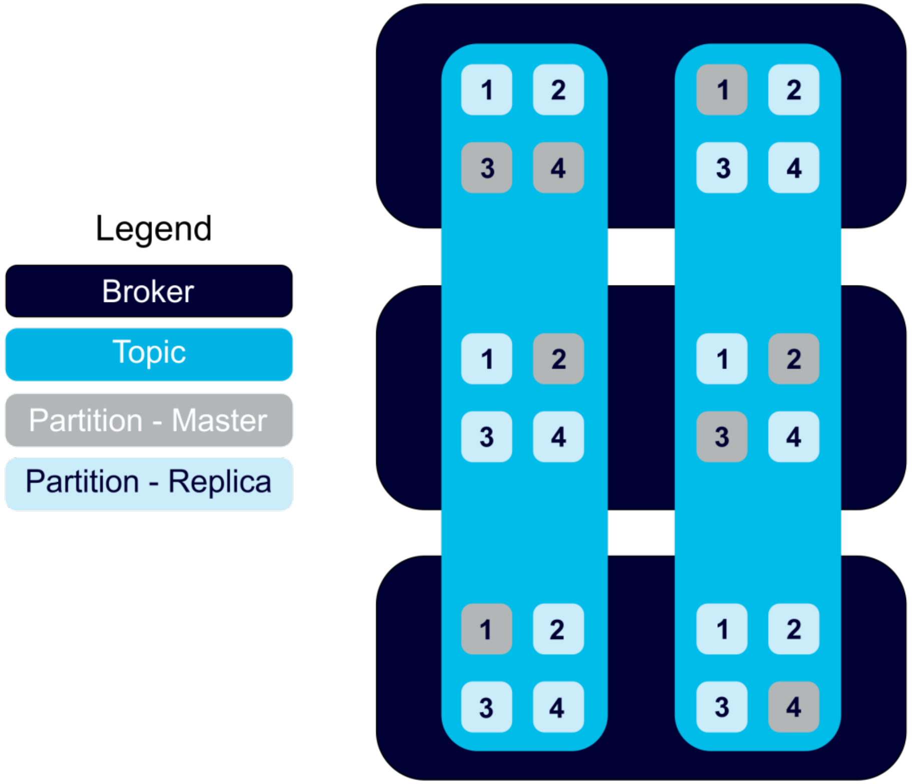
PRIMARY KEY ((Sensor,Date),Timestamp)

Cassandra Partitions in Action



Assumes:
Consistency Level = Local Quorum
Replication Factor=3 in both DCs
NetworkTopologyStrategy

Kafka Topics & Partitions



Topic

- Logical grouping of data
- Settings such as replication, num partitions, log retention, compaction, etc controllable at topic level

Partition

- Subset of messages in a partition that:
  - Have a single master broker
  - Guarantee ordered delivery within that subset
  - Within consumer groups, 1 consumer is assigned to read from each partition
- Number of partitions is set on topic creation
- Messages are mapped to partition by key

Fitting to your use case

Cassandra:

Big data = one or more individually big (>1TB) tables

Need to pre-defined read pattern (at least to partition key)

Very low cost writes - great for high write/read ratio use cases

Ideal for small reads - 1, 10, 100, 1000 rows at a time

No limits to horizontal scaling (data size or ops/sec) provided you can find a partition that fits

Kafka:

Big data = 10k+ message/topic/second

Any Questions?

instaclustr

Ben Slater

Chief Product Officer, Instaclustr

Tim Palmer
APAC Sales Director, Instaclustr