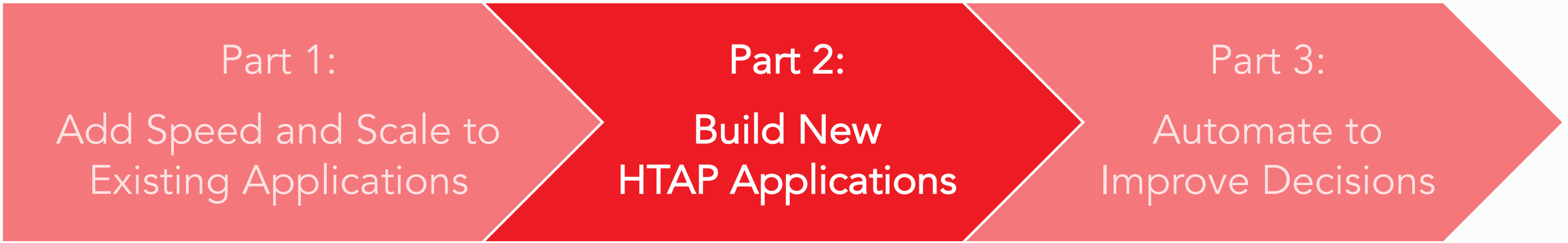
GridGain

apache Ignite

Building New HTAP Applications:
With Apache Ignite

Denis Magda
Apache Ignite PMC Chair
GridGain Product Management

In-Memory Computing Best Practices
Part 2: Build New HTAP Applications



Agenda

- HTAP Principles

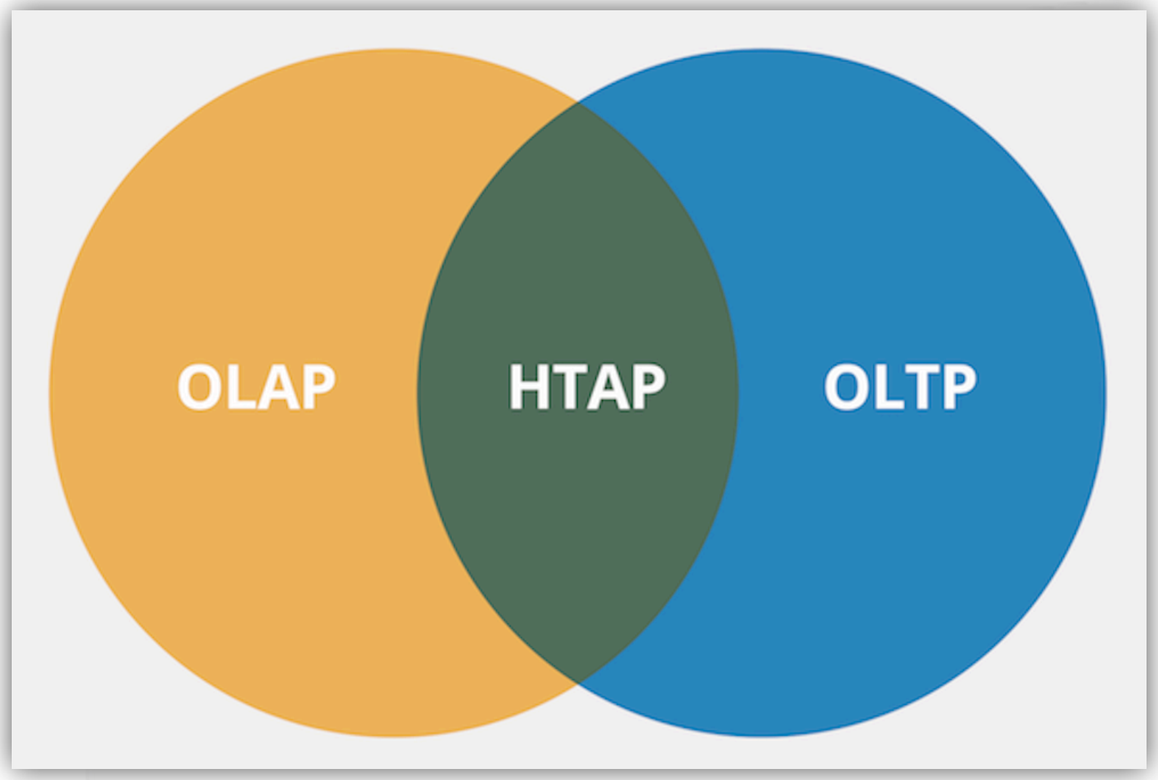
- Apache Ignite for HTAP
  - Memory-Centric Storage
  - Collocated Processing
  - Real-Time Streaming
  - Scalable Machine and Deep Learning
- Demo (Cloud Native Solution)

- Q & A

Hybrid Transactional/Analytical Processing Principles

HTAP Principles

- One Platform Multiple Workloads
  - OLTP and OLAP
  - Real-Time Streaming and Batching
  - No ETL
- Distributed Everything
  - Scalability
  - No Single Point of Failure
  - SQL, Calculations, ML, etc.
- Cloud Native



Apache Ignite for HTAP

Apache Ignite Overview

Financial Services

Telco

Travel & Logistics

E-Commerce

Pharma & Healthcare

IoT

SQL | Key/Value | Transactions | Compute | Services | Streaming | ML

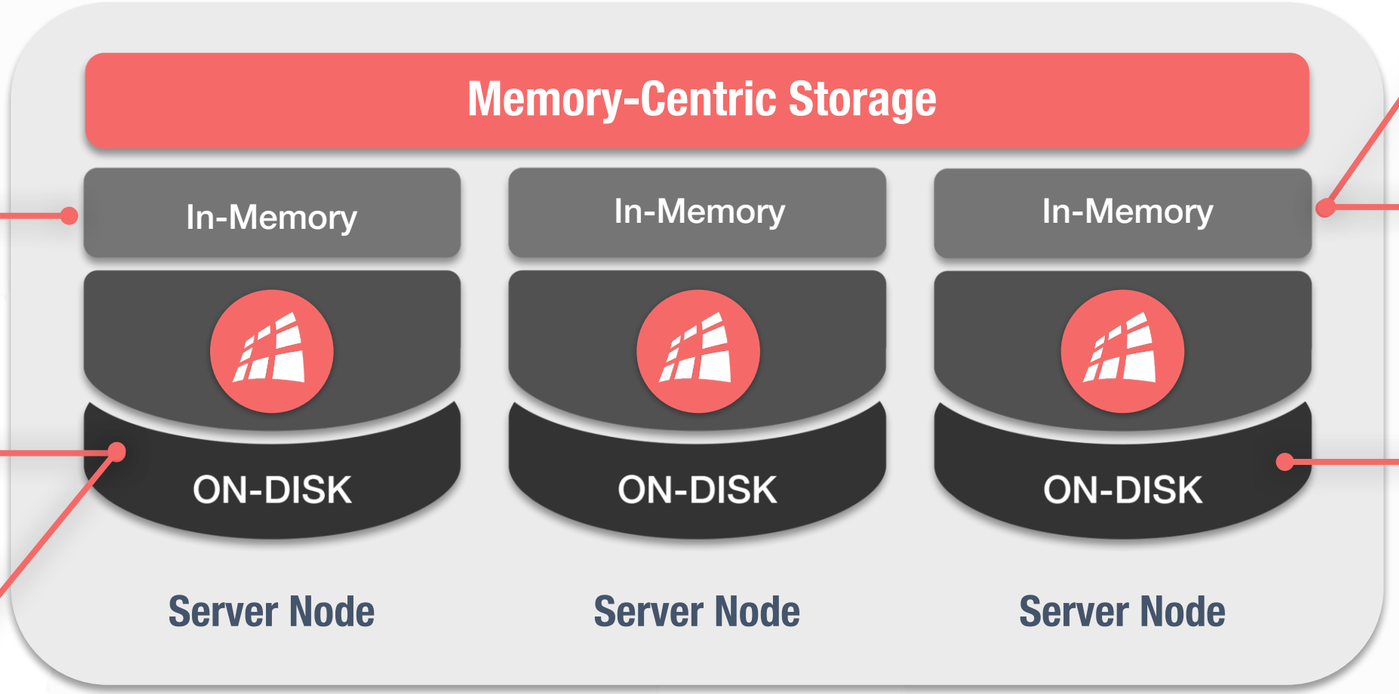
Memory-Centric Storage
Scale to 1000s of Nodes & Store TBs of Data

Ignite Native Persistence
(Flash, SSD, Intel 3D XPoint)

Third-Party Persistence
Keep Your Own DB
(RDBMS, HDFS, NoSQL)

Memory-Centric Storage

Memory-Centric Storage



Predictable Memory Consumption

Fully Transactional WAL (write Ahead Log)

Instantaneous Restarts

Automatic Defragmentation

Off-heap Removes Noticeable GC Pauses

Stores Superset of Data

GridGain

Memory Usage Modes

Mode	Description	Major Advantage
In-Memory	Pure In-Memory Storage	Maximum perfomance possible (data is never written to disk)
In-Memory + 3rd Party DB	Ignite as a caching layer (aka. data grid) above an existing databases – RDBMS, NoSQL.	Reads acceleration and scale of existing solutions
In-Memory + Full Copy on Disk	The whole data set is stored both in memory and on disk	Reads/Writes acceleration and fast restarts
100% on Disk + In-Memory Cache	100% of data is in Ignite native persistence and a subset is in memory	Above + Unlimited scale beyond RAM capacity

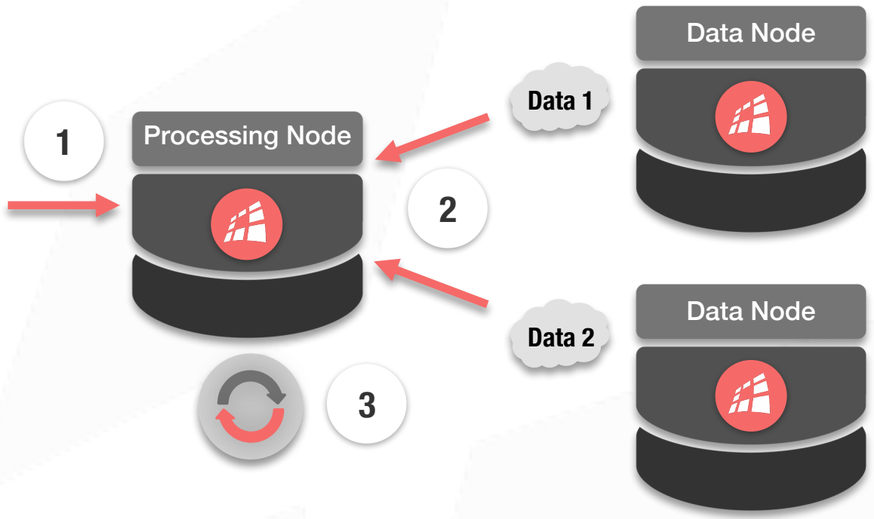
Collocated Processing

Affinity Collocation

- Data-to-Data Collocation
  - Countries and Cities, Vendors and Cars
  - Efficient SQL JOINS

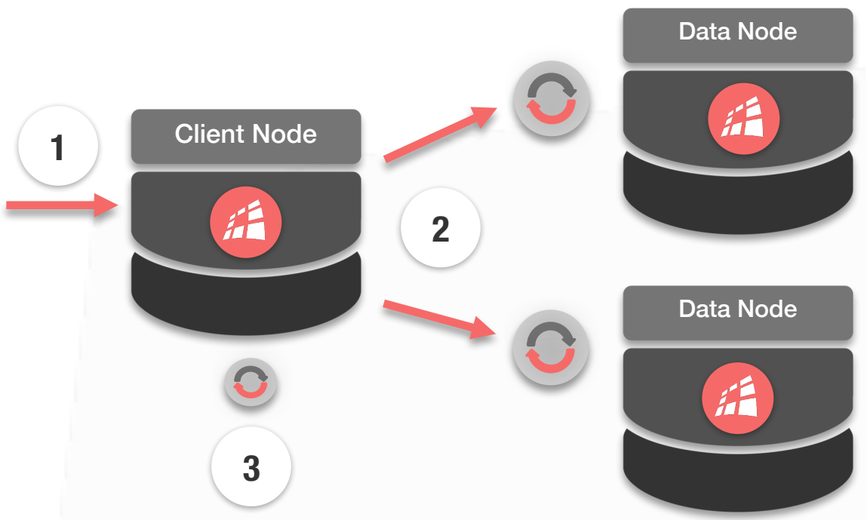
- Compute-to-Data Collocation
  - Send Compute to Data (Not Data to Compute)
  - Reduced Network Traffic -> Better Performance

Client-Server Processing



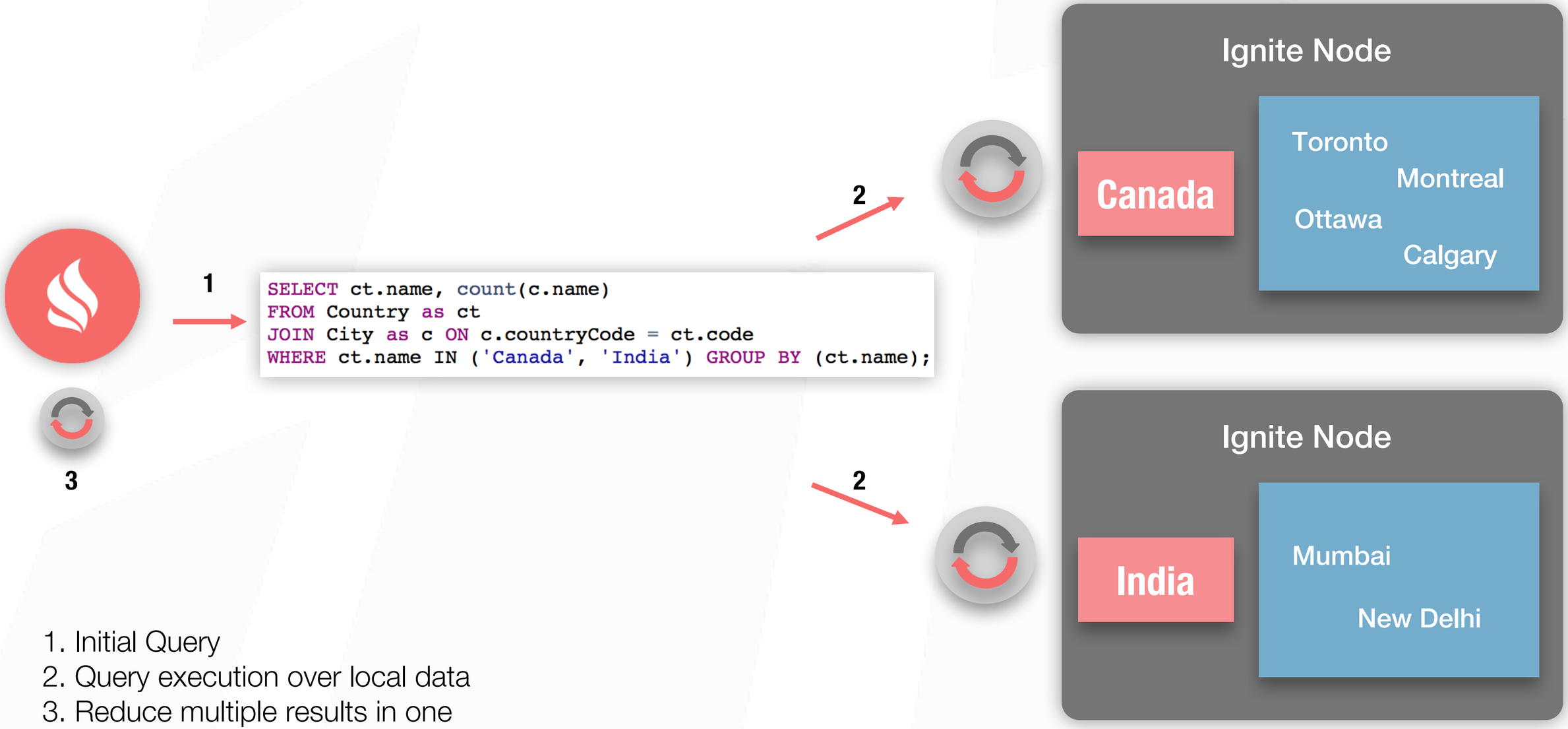
1. Initial Request
2. Fetch data from remote nodes
3. Process the entire data-set

Co-located Processing



1. Initial request
2. Co-locate processing with data
3. Reduce multiple results into one

Collocated Joins

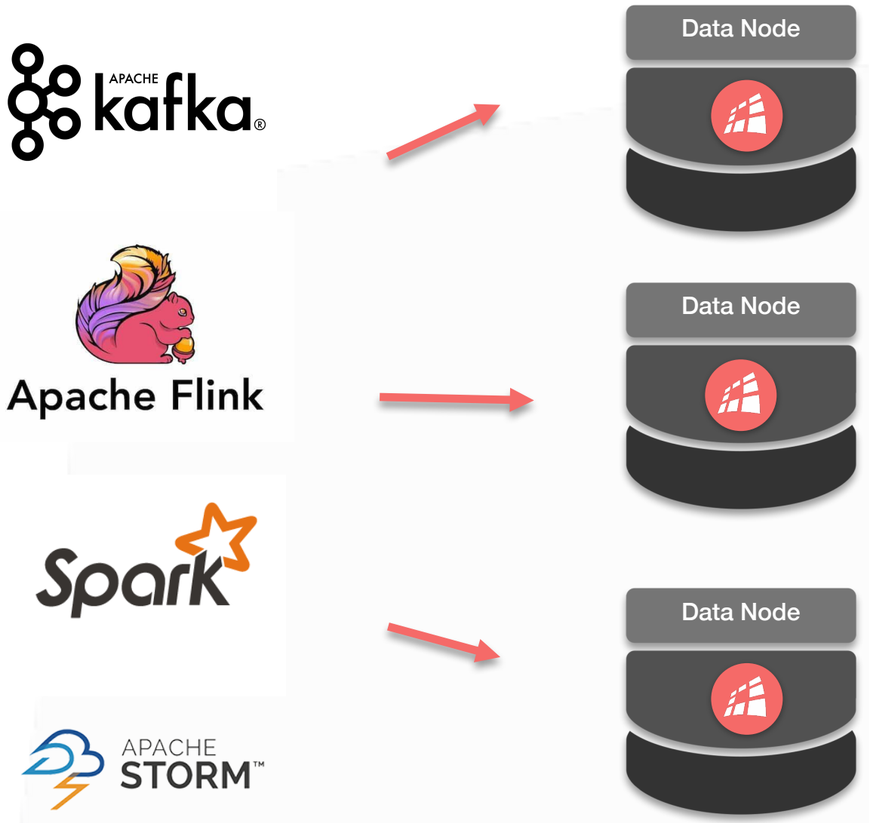


1. Initial Query
2. Query execution over local data
3. Reduce multiple results in one

Real-Time Streaming

Real-Time Streaming

- Various Streaming Technologies
  - Kafka, Spark, Flink, Storm, etc.
  - Process, Enrich and then =>
- Ignite as a store for streaming pipelines
  - Streaming Analytics

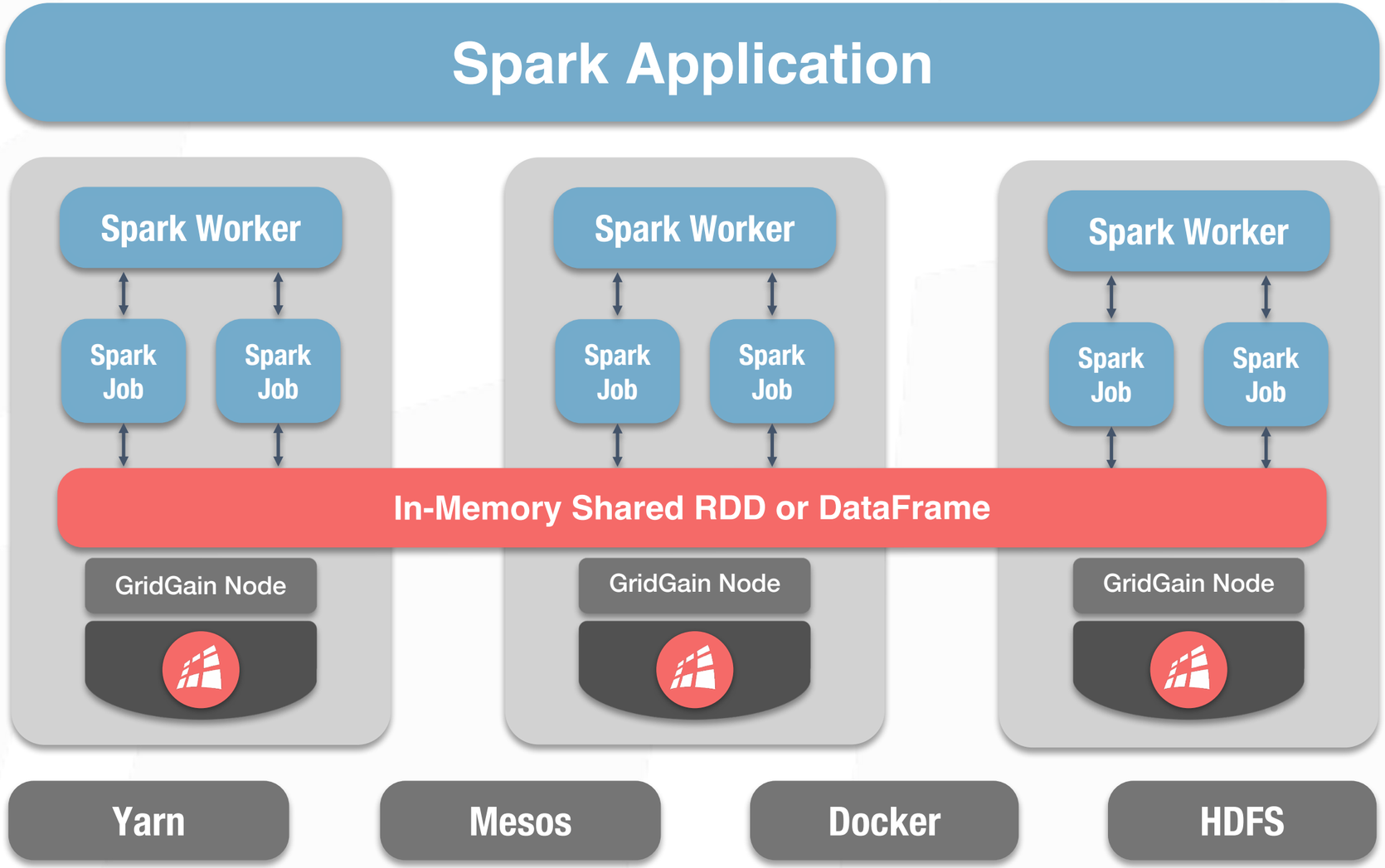


Ignite + Spark

Share state and data among Spark jobs

No data movement

In-place queries execution



Boost DataFrame and SQL Performance

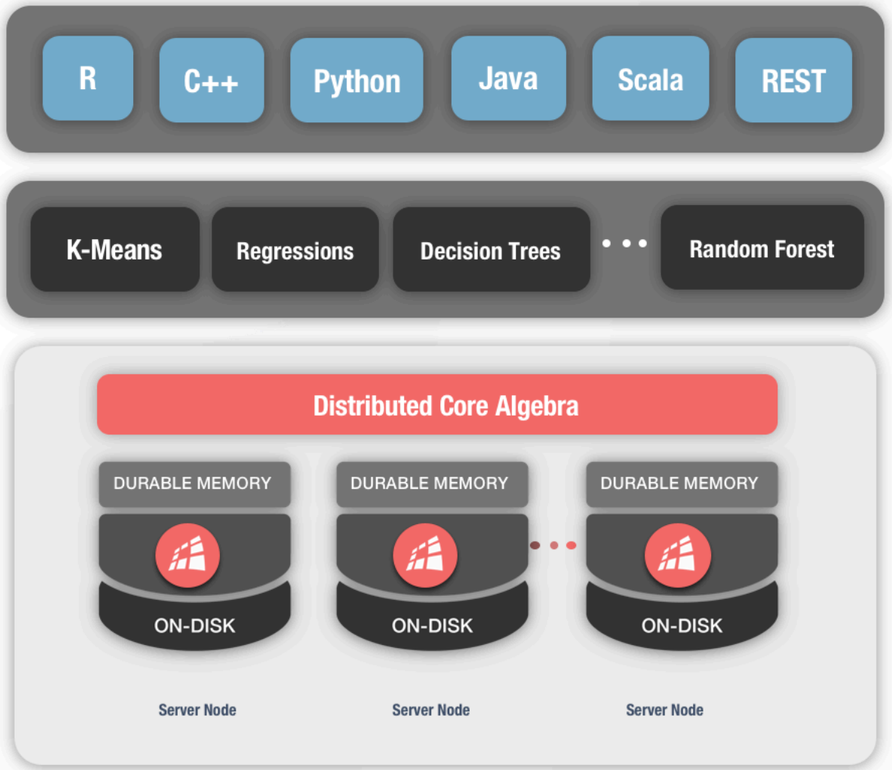
SQL on top of RDDs

GridGain

Scalable ML and DL

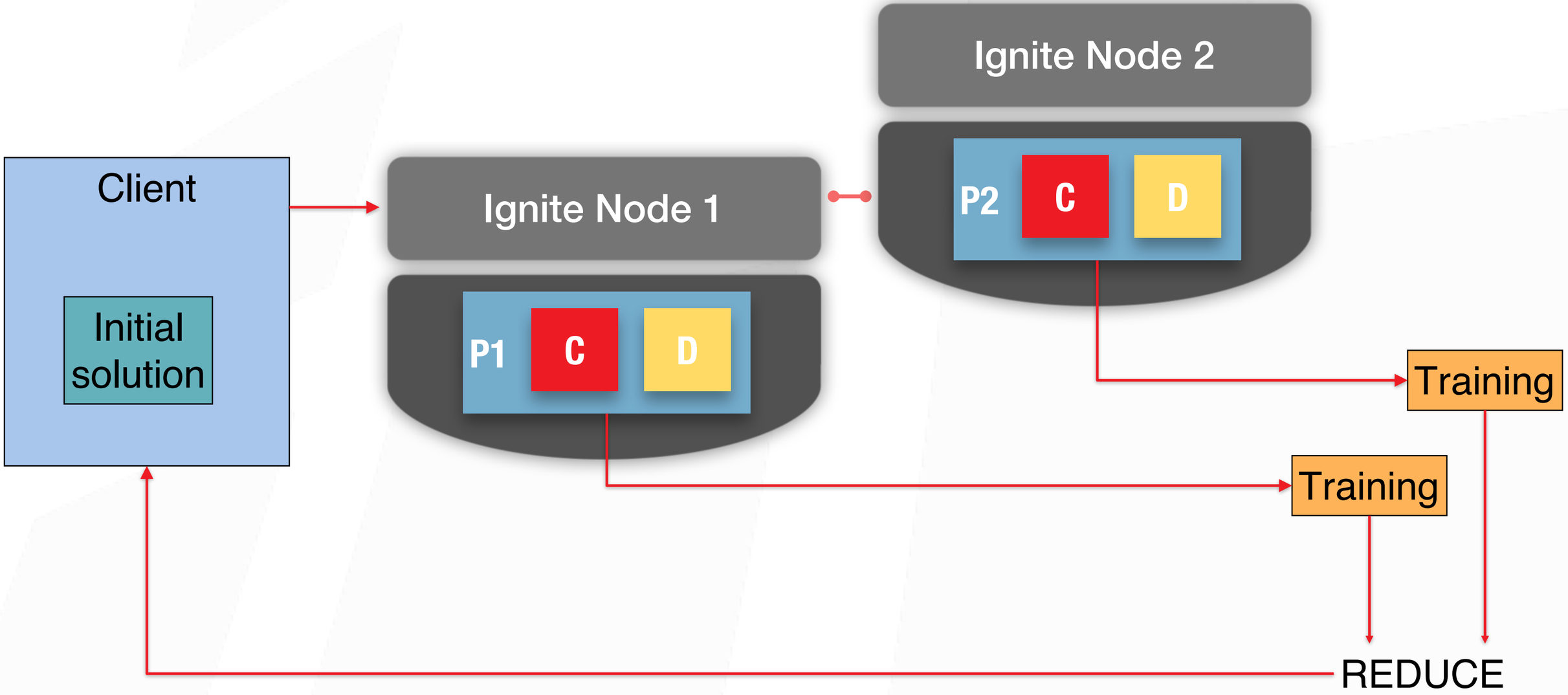
ML/DL for HTAP

- Massive Scalability
  - Scale Beyond Single Server Unit
  - Horizontal + Vertical
  - RAM + Disk
- Zero-ETL
  - Train Models and run Algorithms in Place
- Fault Tolerance and Continuous Learning
  - Partition-based dataset

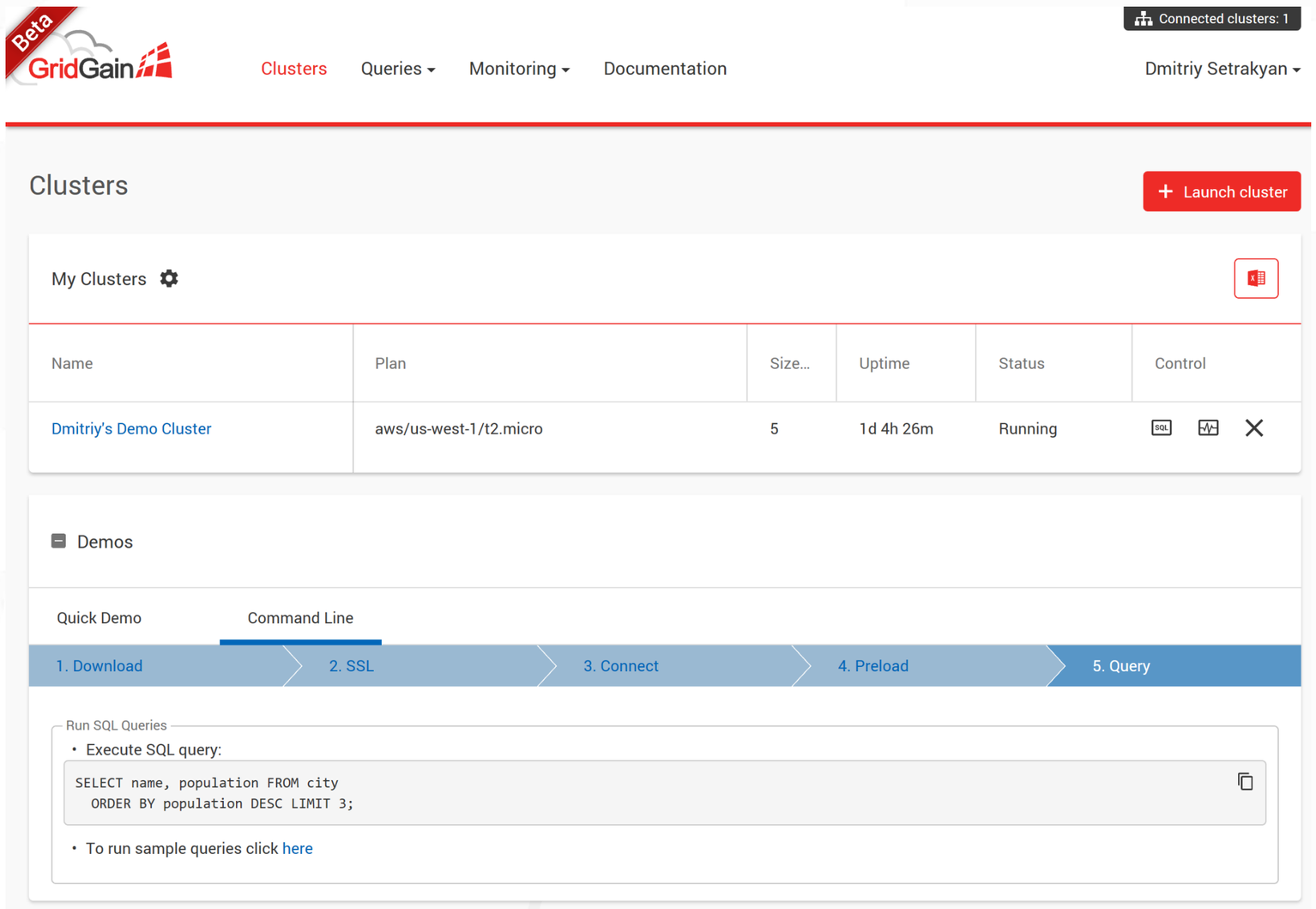


GridGain

Partition-Based Dataset



Demo - Cloud Native Solutions



https://cloud.gridgain.com

GridGain

Summary: HTAP and Apps Architecture

- Single storage and platform
  - OLAP, OLTP, Streaming
- No ETL
- Horizontal Scalability is at the Core
- Cloud Deployment Options

Thank You!!!

Thank you for joining us. Follow the conversation.
https://ignite.apache.org

@denismagda
#apacheignite
#gridgain