



In-Memory Performance
Durability of Disk





Ignite the Fire in your SQL App

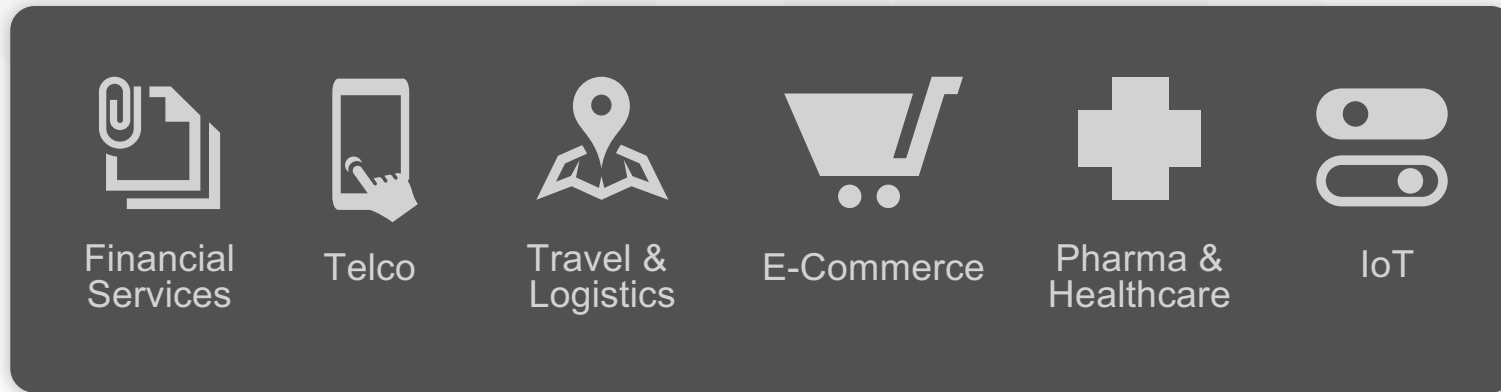


Akmal B. Chaudhri
Technology Evangelist
GridGain Systems

Agenda

- SQL Capabilities
 - Connectivity
 - Data Definition Language
 - Data Manipulation Language
- Caching from a 3rd Party Storage
- Ignite as Memory-Centric Database
- Q&A

GridGain In-Memory Computing Platform



Financial Services Telco Travel & Logistics E-Commerce Pharma & Healthcare IoT

SQL Key/Value Transactions Compute Services Streaming ML

Memory-Centric Storage

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

Third-Party Persistence
(RDBMS, HDFS, NoSQL)

Data Snapshots & Recovery

Monitoring & Management

Security & Auditing

Data Center Replication

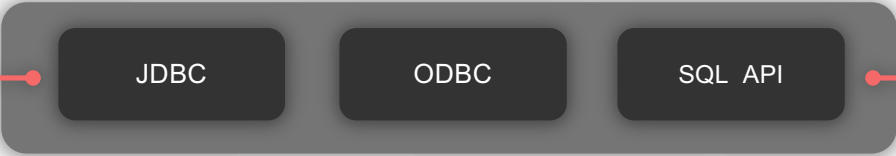
SQL Capabilities

Distributed SQL

Cross-platform
Compatibility

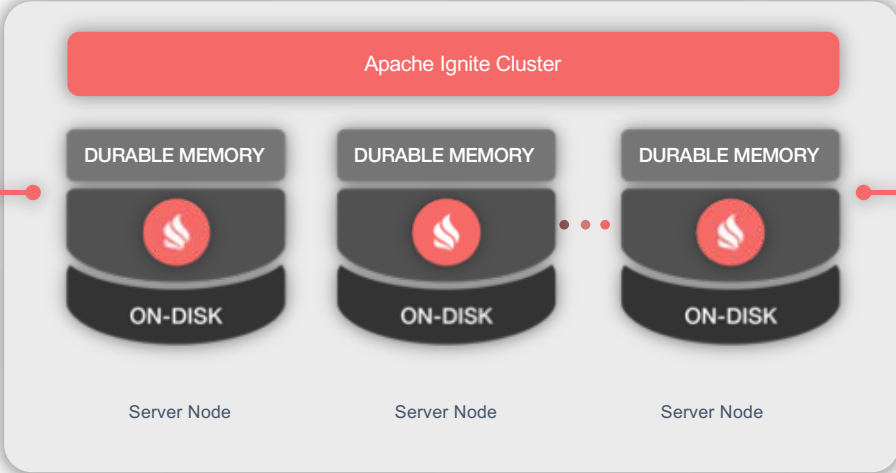


DDL, DML Support



SELECT, UPDATE,
INSERT, MERGE,
DELETE, CREATE
and ALTER

Indexes in
RAM or Disk



Dynamic
Scaling

Connectivity

- JDBC
- ODBC
- REST
- Java, .NET and C++ APIs

```
// Register JDBC driver.  
Class.forName("org.apache.ignite.IgniteJdbcThinDriver");  
  
// Open the JDBC connection.  
Connection conn = DriverManager.getConnection("jdbc:ignite:thin://192.168.0.50");
```

```
./sqlline.sh --color=true --verbose=true -u jdbc:ignite:thin://127.0.0.1/
```

Data Definition Language

- CREATE/DROP TABLE
- CREATE/DROP INDEX
- ALTER TABLE
- Changes Durability
 - Ignite Native Persistence

```
CREATE TABLE `city` (  
  `ID` INT(11),  
  `Name` CHAR(35),  
  `CountryCode` CHAR(3),  
  `District` CHAR(20),  
  `Population` INT(11),  
  PRIMARY KEY (`ID`, `CountryCode`)  
) WITH "template=partitioned, backups=1, affinityKey=CountryCode";
```

<https://apacheignite-sql.readme.io/docs/ddl>

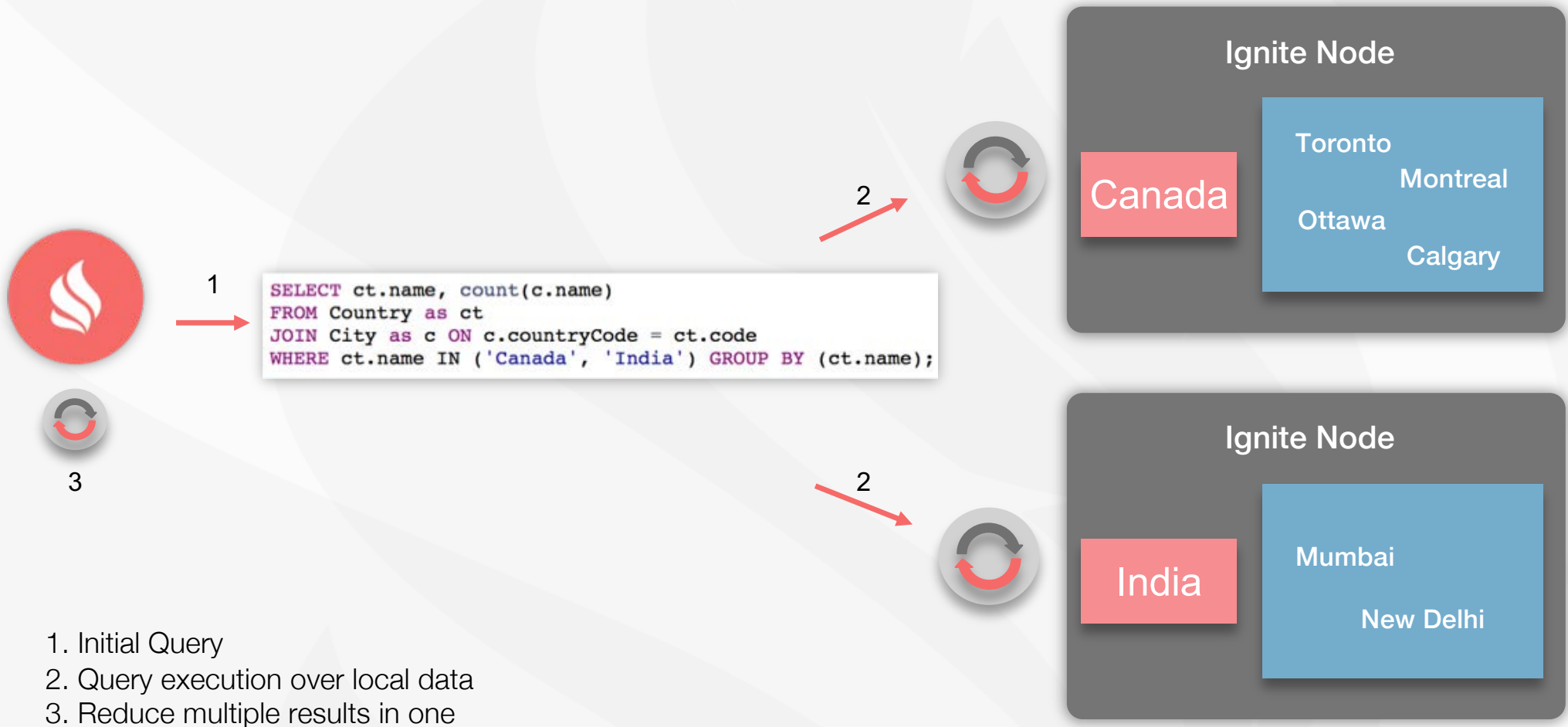
Data Manipulation Language

- ANSI-99 specification
- Fault-tolerant and consistent
- INSERT, UPDATE, DELETE
- SELECT
 - JOINS
 - Subqueries

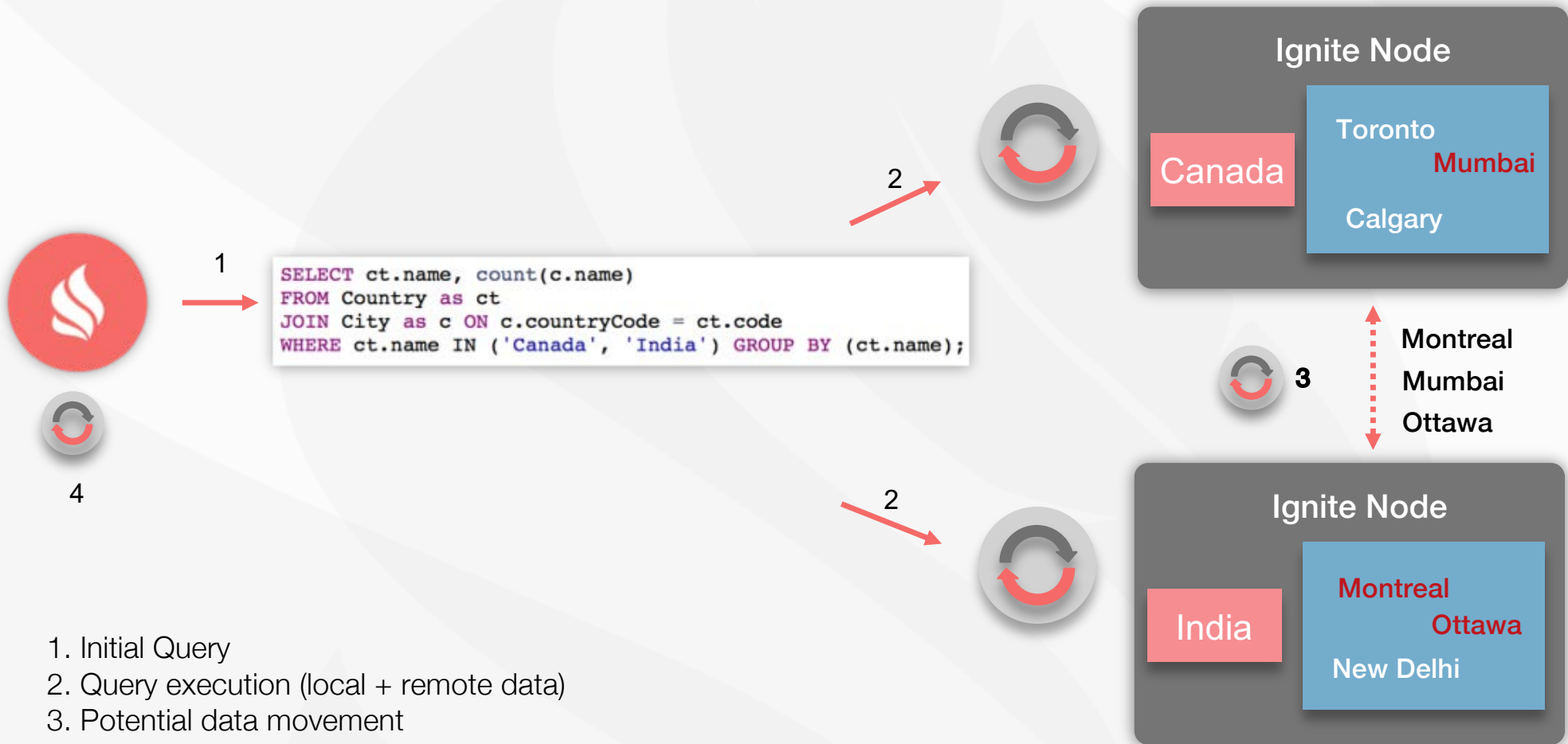
```
SELECT country.name, city.name, MAX(city.population) as max_pop
FROM country JOIN city ON city.countrycode = country.code
WHERE country.code IN ('USA', 'RUS', 'CHN')
GROUP BY country.name, city.name ORDER BY max_pop DESC LIMIT 3;
```

<https://apacheignite-sql.readme.io/docs/dml>

Collocated Joins



Non-Collocated Joins

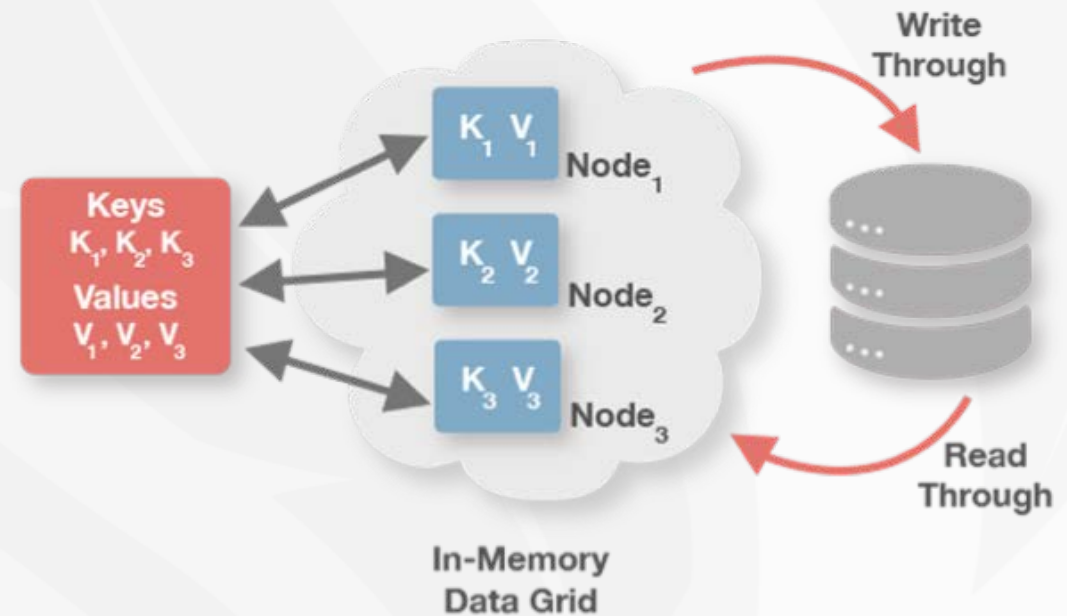


1. Initial Query
2. Query execution (local + remote data)
3. Potential data movement
4. Reduce multiple results in one

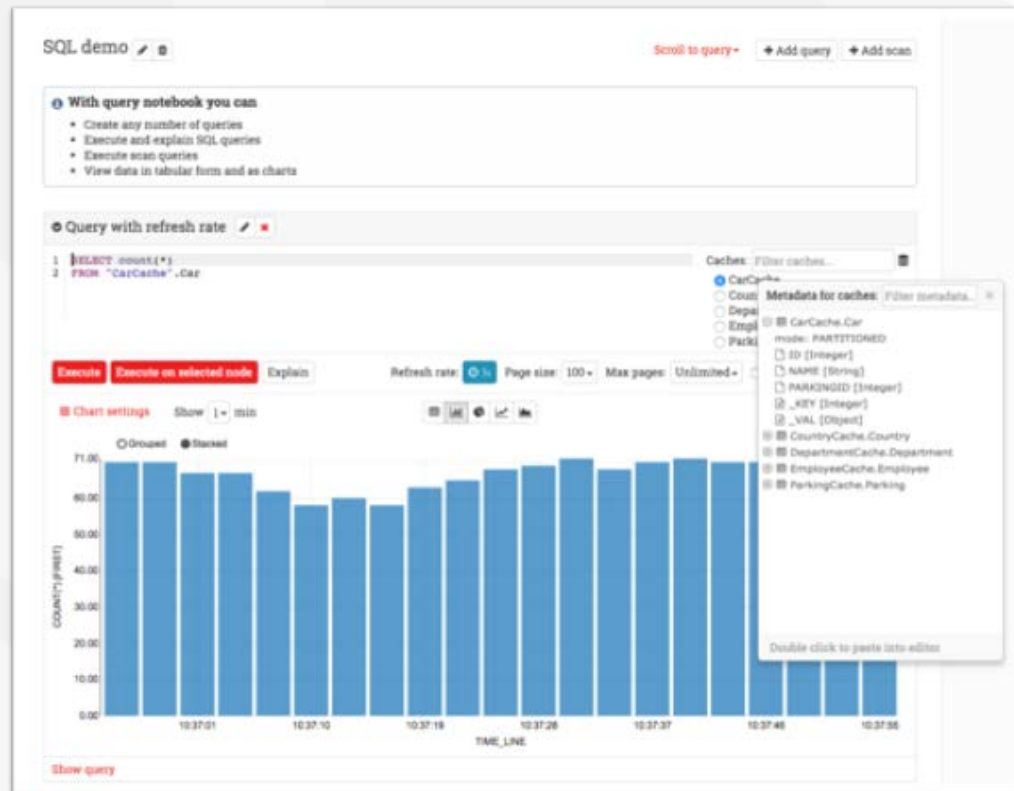
Caching from a 3rd Party Storage

Turbocharging Database System

- Database Caching Use Case
 - Slide Ignite in between Database System and applications
- No 'rip and replace' Performance Boost
 - Keep data both in memory and Database System
 - Scale to 1000s of nodes
- Automatic Read-Through and Write-Through
 - Key-Value Operations Only
- ANSI-99 SQL
 - Over in-memory data sets



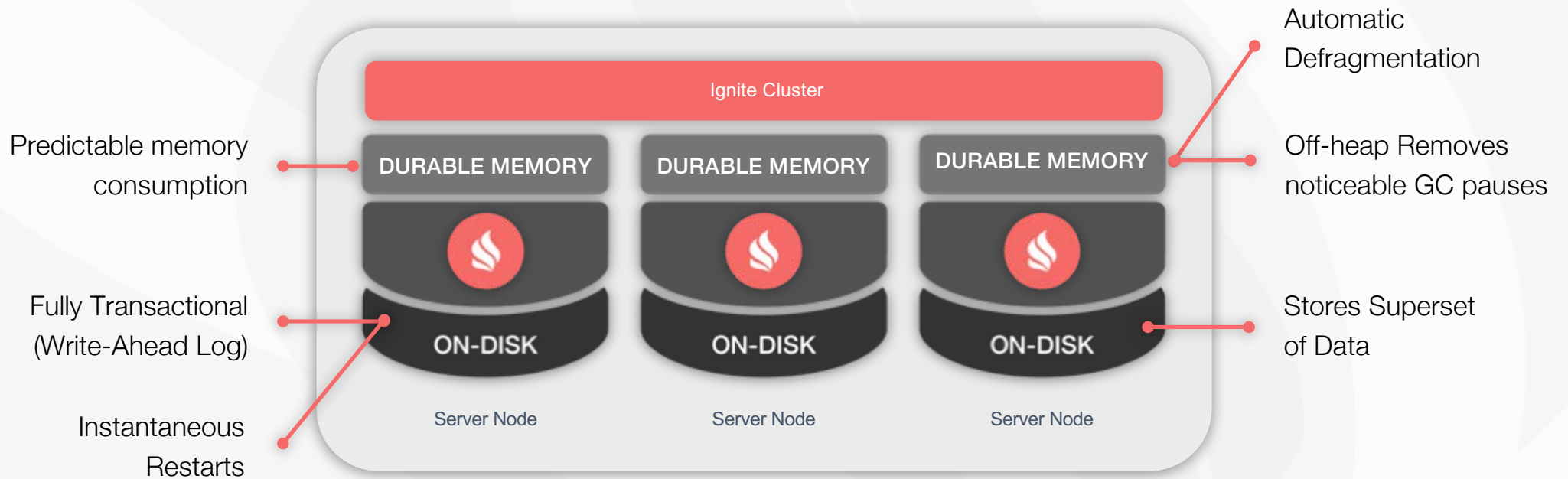
Apache Ignite Web Console



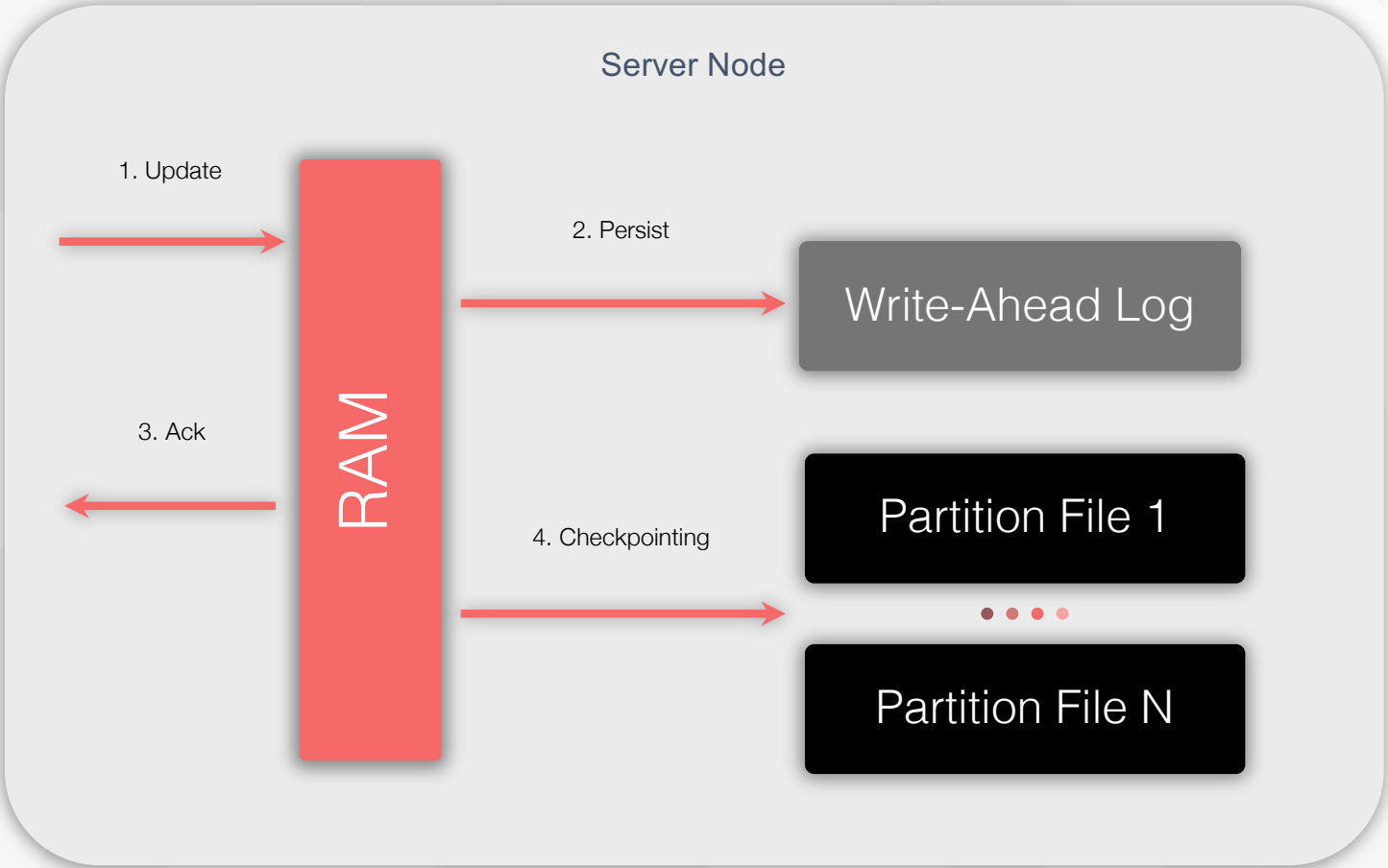
Demo

Ignite as Memory-Centric Database

Durable Memory



Ignite Native Persistence



Enabling Ignite Native Persistence

```
<bean class="org.apache.ignite.configuration.IgniteConfiguration">
  <!-- Enabling Apache Ignite native persistence. -->
  <property name="dataStorageConfiguration">
    <bean class="org.apache.ignite.configuration.DataStorageConfiguration">
      <property name="defaultDataRegionConfiguration">
        <bean class="org.apache.ignite.configuration.DataRegionConfiguration">
          <property name="persistenceEnabled" value="true"/>
        </bean>
      </property>
    </bean>
  </property>
</bean>

<!-- Additional setting. -->

</bean>
```

<https://apacheignite.readme.io/docs/distributed-persistent-store>

Demo



Any Questions?

Thank you for joining us. Follow the conversation.
<http://ignite.apache.org>



#apacheignite