Hands-On: Make Sure You Are Ready

Step 0: Get what you need.

- Java Development Kit, release 8 or later
- Apache Maven 3.0 or later
- Your favorite IDE or text editor

Step 1: Download the training project: https://github.com/GridGain-Demos/imc-essentials-in-90-minutes.

O'REILLY®

In-Memory Computing Essentials

November/2020

Your Instructor: Denis Magda

- Distributed in-memory system
  - Apache Ignite Committer and PMC Member
  - Head of DevRel at GridGain
- Java engineering and architecture
  - Java engineering at Oracle
  - Technology evangelism at Sun Microsystems

Training Flow

- Introduction
  - Why In-Memory Computing? (5 mins)
  - Apache Ignite, Brief Overview (15 mins)
- Data Partitioning (25 mins)

- Break (5 mins)

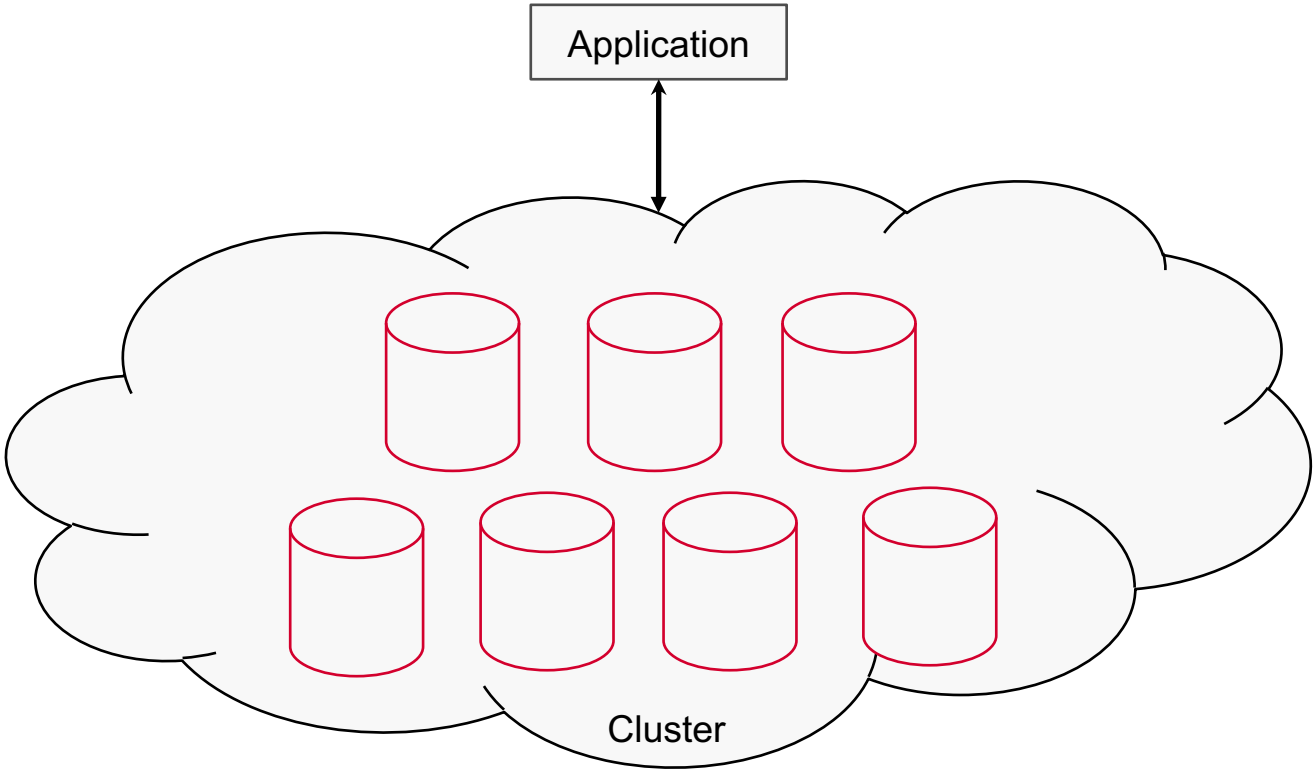
- Affinity Co-location (25 mins)
- Co-located Computations (25 mins)

Introduction

Why In-Memory Computing?

Speed and Scale

In-Memory Software Is Distributed



Comparing System-Event Latencies

System Event	Actual Latency	Scaled Latency
One CPU cycle	0.4 ns	1 s
Level 1 cache access	0.9 ns	2 s
Level 2 cache access	2.8 ns	7 s
Level 3 cache access	28 ns	1 min
Main memory access (DDR DIMM)	~100 ns	4 min
Intel Optane DC persistent memory access	~350 ns	15 min
Intel Optane DC SSD I/O	<10 µs	7 hrs
NVMe SSD I/O	~25 µs	17 hrs
SSD I/O	50 – 150 µs	1.5 – 4 days
Rotational disk I/O	1 – 10 ms	1 – 9 months
Internet: SF to NY	65 ms	5 years

Memory Versus Disk Latency

System Event	Actual Latency	Scaled Latency
One CPU cycle	0.4 ns	1 s
Level 1 cache access	0.9 ns	2 s
Level 2 cache access	2.8 ns	7 s
Level 3 cache access	28 ns	1 min
Main memory access (DDR DIMM)	~100 ns	4 min
Intel Optane DC persistent memory access	~350 ns	15 min
Intel Optane DC SSD I/O	< 10 µs	7 hrs
NVMe SSD I/O	~25 µs	17 hrs
SSD I/O	50 – 150 µs	1.5 – 4 days
Rotational disk I/O	1 – 10 ms	1 – 9 months
Internet: SF to NY	65 ms	5 years

Poll 1

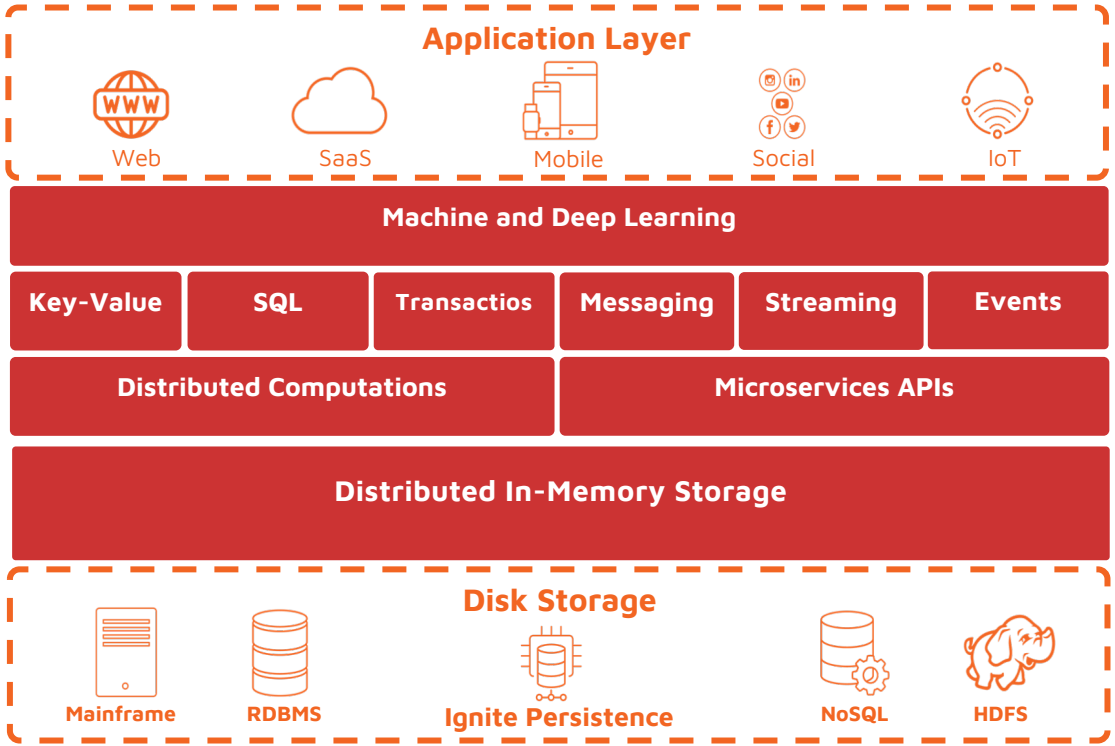
What type of in-memory software do you use?

- Caches — Redis, Memcached,...
- Databases — MemSQL, VoltDB,...
- Grids and platforms — Ignite, Hazelcast, Gemfire,...
- Not there yet

Apache Ignite

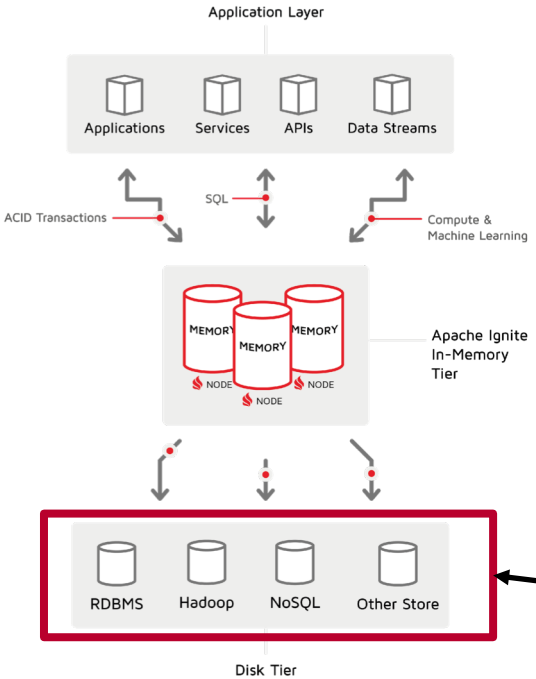
Brief Overview

Apache Ignite Platform



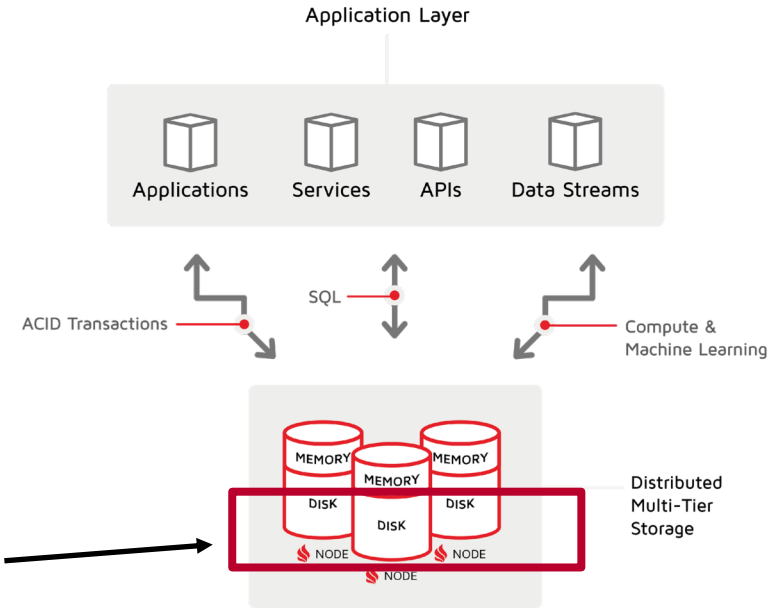
How Ignite Is Used

Ignite as a Cache



difference

Ignite as a Database



Of more than 350 Apache Projects

Top 10 Dev Mailing Lists	Top 5 User Mailing Lists
1. Flink	1. Flink
...	2. Lucene
8. Apache Ignite	3. Apache Ignite
9. Apache Arrow	4. Apache Tomcat
10. Apache Airflow	5. kafka

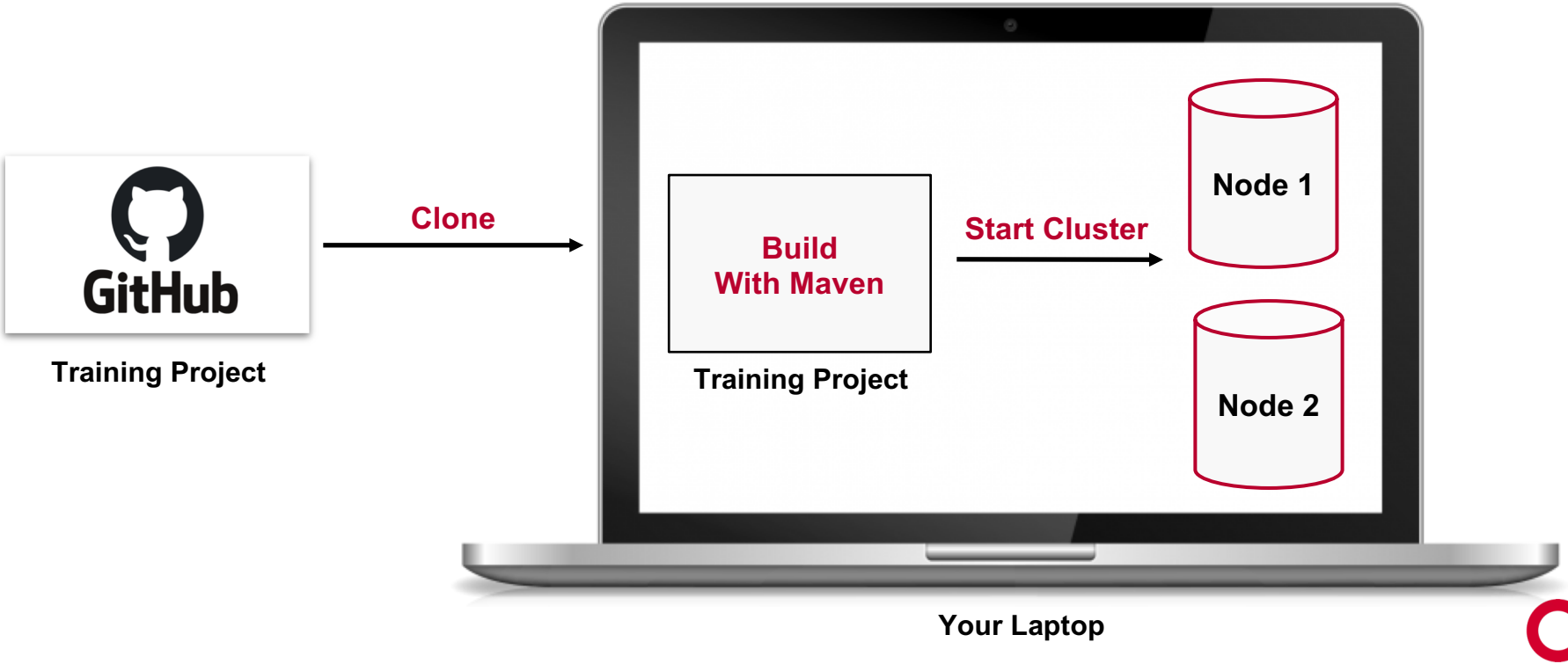
15M+

downloads

Apache Ignite is used by thousands of companies.

Bloomberg

Hands-On: What You'll Do Next



Hands-On: Start the Cluster

Step 0: Get what you need.

- Java Development Kit, version 8 or later
- Apache Maven 3.0 or later
- Your favorite IDE or text editor

Step 1: Download the training project:
https://github.com/GridGain-Demos/imc-essentials-in-90-minutes.

Step 2: Unzip the project and go to its root directory.

Hands-On: Start the Cluster

Step 3: Build the core module of the project.

```
mvn clean package -P core
```

Step 4: Start the first cluster node.

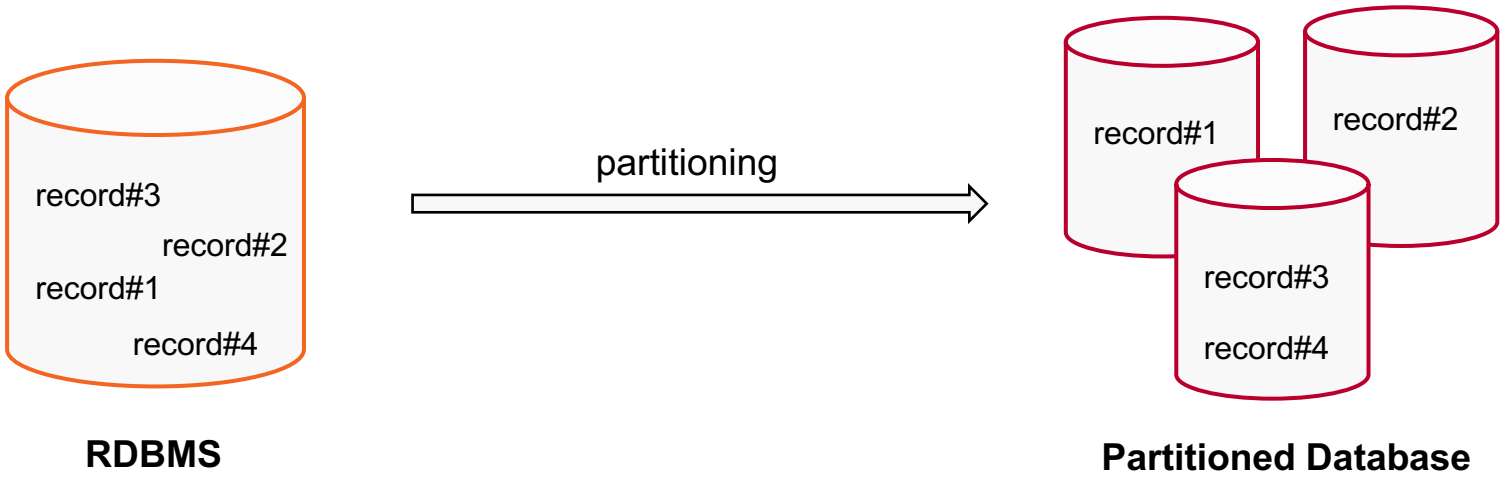
```
java -cp libs/core.jar training.ServerStartup
```

Step 5: In another terminal window, start the second node.

Data Partitioning

Using all of the cluster’s memory and CPUs

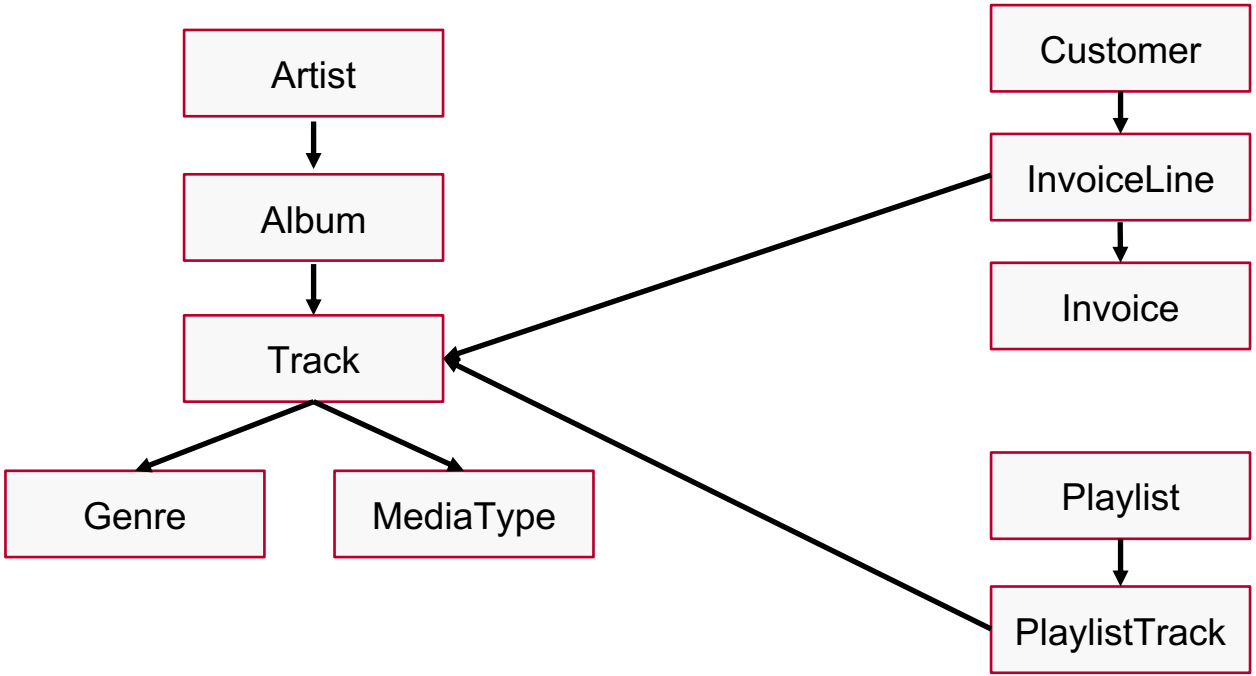
Partitioning and Horizontal Scalability



RDBMS

Partitioned Database

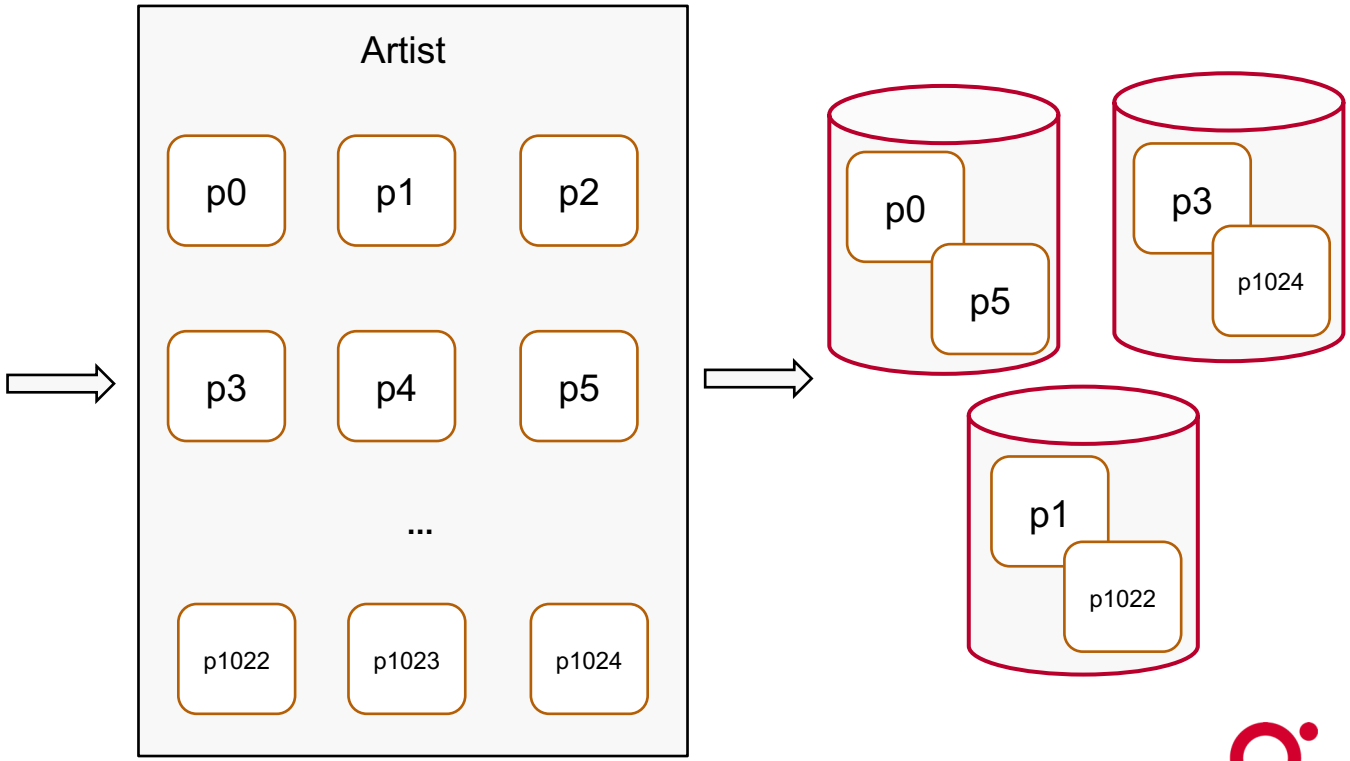
Chinook Database: Digital Media Store



https://github.com/lerocha/chinook-database

Partitions

```
CREATE TABLE Artist {
...
}
```

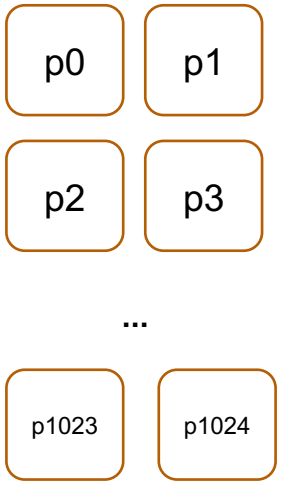


Record->Partition->Node Mapping

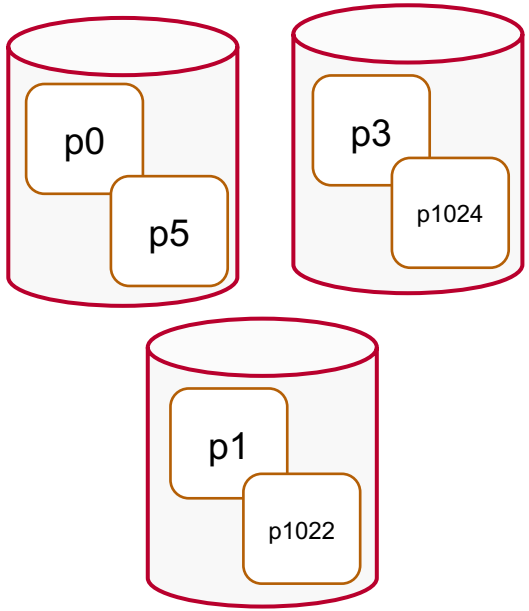
```
INSERT INTO Artist
(ArtistId, Name)
VALUES (10, 'The Beatles')
```

1. key to partition (artistId)

{10, ‘The Beatles}



2. request to partition owner

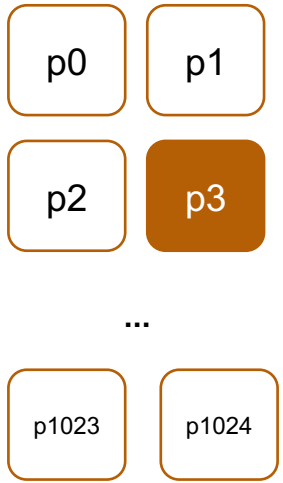


Record->Partition->Node Mapping

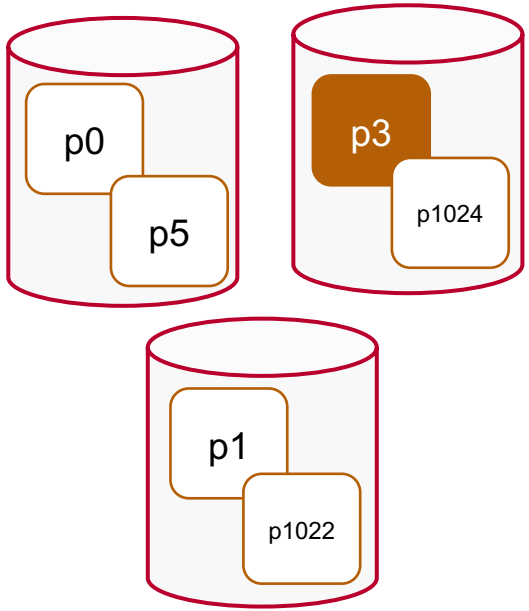
```
INSERT INTO Artist
(ArtistId, Name)
VALUES (10, 'The Beatles')
```

1. key to partition (artistId)

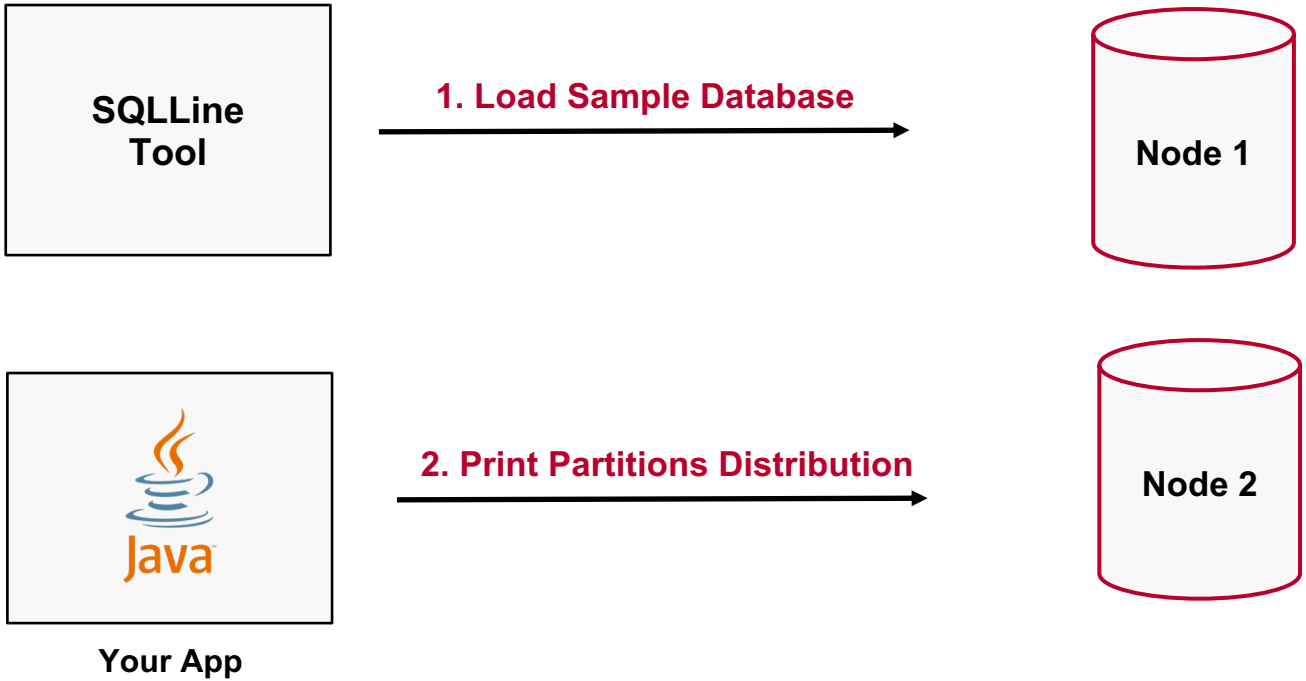
{10, ‘The Beatles}



2. request to partition owner



Hands-On: What You'll Do Next



Hands-On: Load the Chinook Database

Step 1: Start SQLLine tool.

```
java -cp libs/core.jar sqlline.SqlLine
```

Step 2: Use SQLLine to connect to the cluster.

```
!connect jdbc:ignite:thin://127.0.0.1/ ignite ignite
```

Hands-On: Load the Chinook Database

Step 3: Load the database.

```
!run config/media_store.sql
```

Step 4: List the created tables.

```
!tables
```

Hands-On: Print Partitions' Distribution

Step 1: Update the oreilly.training.KeyValueApp logic.

```
Affinity<Integer> affinity = ignite.affinity("Artist");

int part = affinity.partition(artistKey);
UUID nodeId = affinity.mapPartitionToNode(part).id();

System.out.println(artist + " [part=" + part + ", node=" + nodeId);
```

Hands-On: Print Partitions Distribution

Step 2: Build a version of the project that has application classes:

```
mvn clean package -P apps
```

Step 3: Start the application and check partitions' distribution

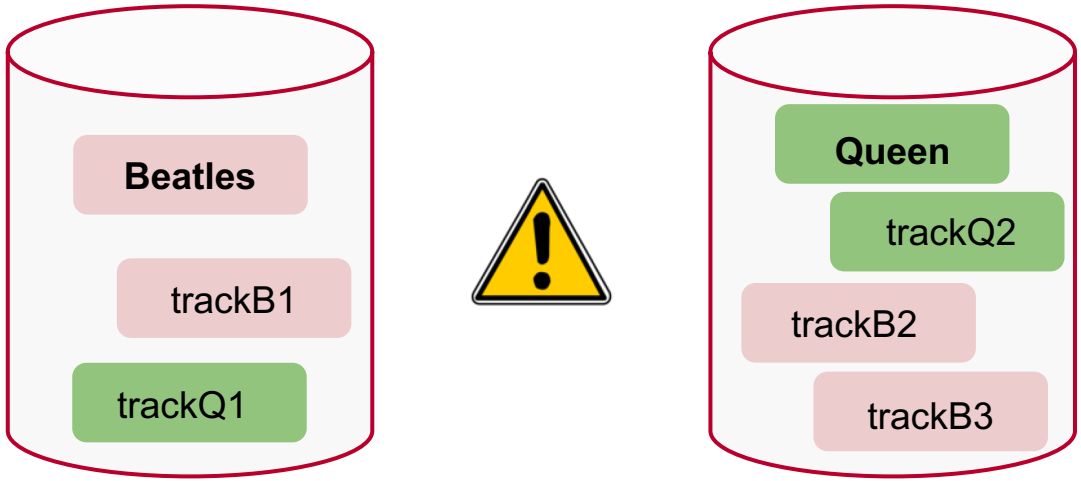
```
java -cp libs/apps.jar training.KeyValueApp
```

Five-Minute Break

Affinity Co-location

Reducing network utilization for complex requests

Random Data Distribution

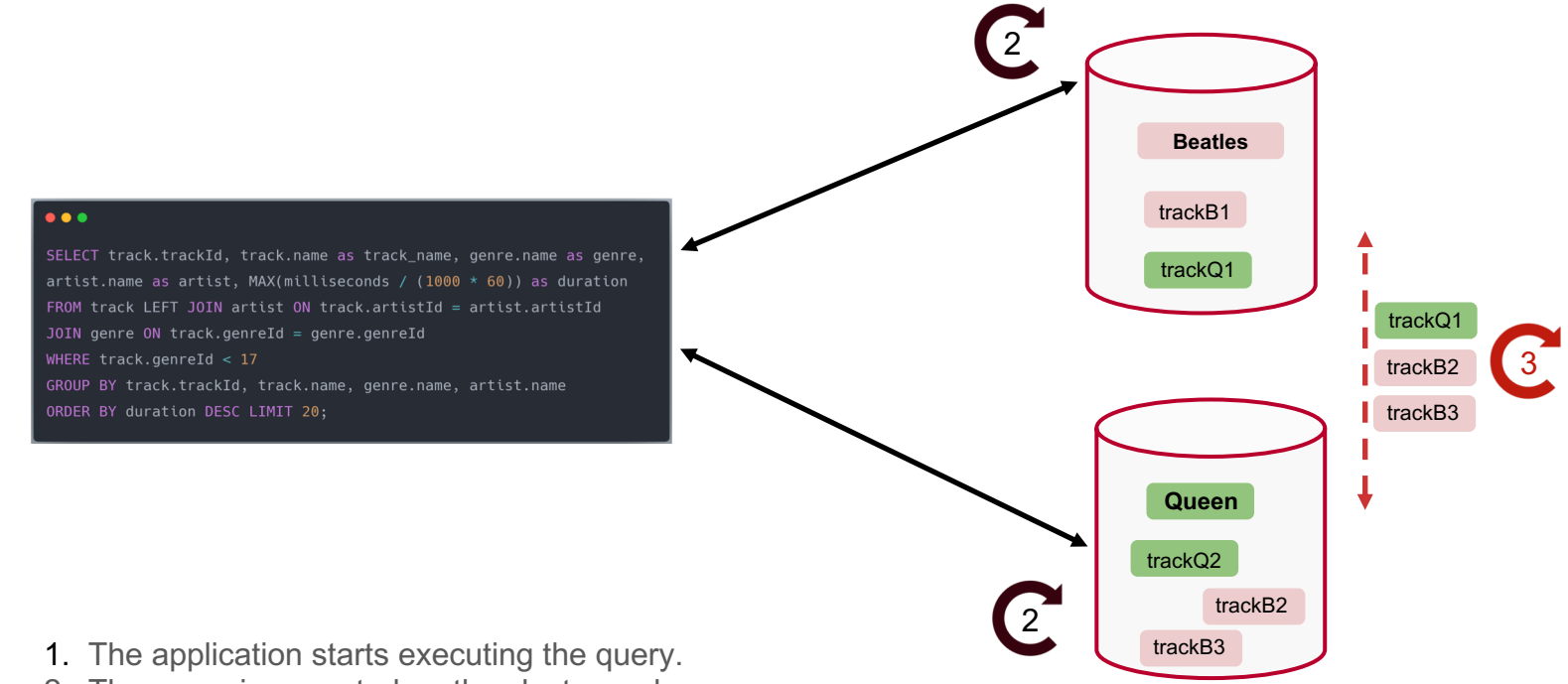


Data Is Shuffled During the JOIN phase

Application

1 & 4

```
SELECT track.trackId, track.name as track_name, genre.name as genre,
artist.name as artist, MAX(milliseconds / (1000 * 60)) as duration
FROM track LEFT JOIN artist ON track.artistId = artist.artistId
JOIN genre ON track.genreId = genre.genreId
WHERE track.genreId < 17
GROUP BY track.trackId, track.name, genre.name, artist.name
ORDER BY duration DESC LIMIT 20;
```

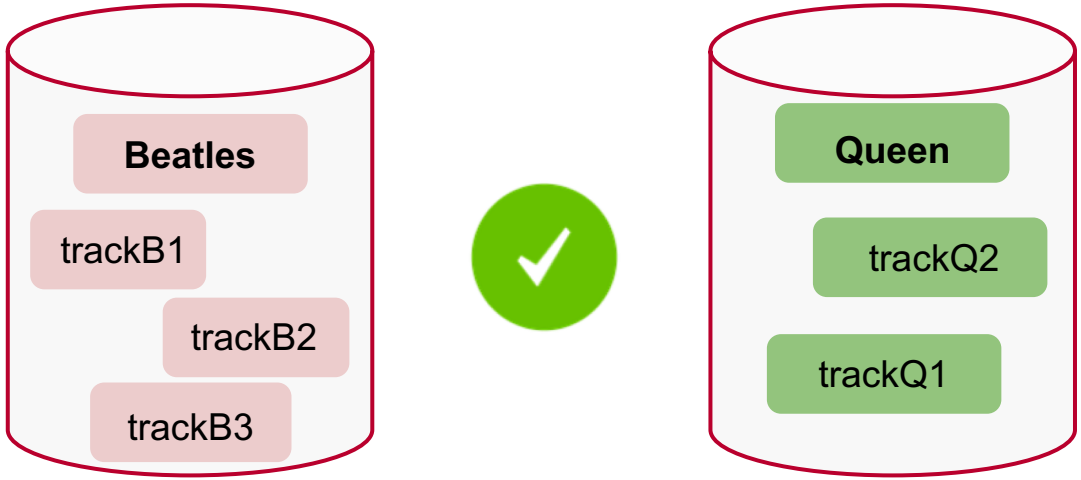


1. The application starts executing the query.
2. The query is executed on the cluster nodes.
3. Data is shuffled between the nodes on JOIN.
4. The result set is reduced.

Disk Versus Network Latency

System Event	Actual Latency	Scaled Latency
One CPU cycle	0.4 ns	1 s
Level 1 cache access	0.9 ns	2 s
Level 2 cache access	2.8 ns	7 s
Level 3 cache access	28 ns	1 min
Main memory access (DDR DIMM)	~100 ns	4 min
Intel Optane DC persistent memory access	~350 ns	15 min
Intel Optane DC SSD I/O	<10 µs	7 hrs
NVMe SSD I/O	~25 µs	17 hrs
SSD I/O	50-150 µs	1.5 – 4 days
Rotational disk I/O	1 – 10 ms	1 – 9 months
Internet: SF to NY	65 ms	5 years

Co-located Data Distribution



How to Group Related Data

```
CREATE TABLE Artist
(
    ArtistId INT,
    Name VARCHAR(120),
    PRIMARY KEY (ArtistId)
);
```

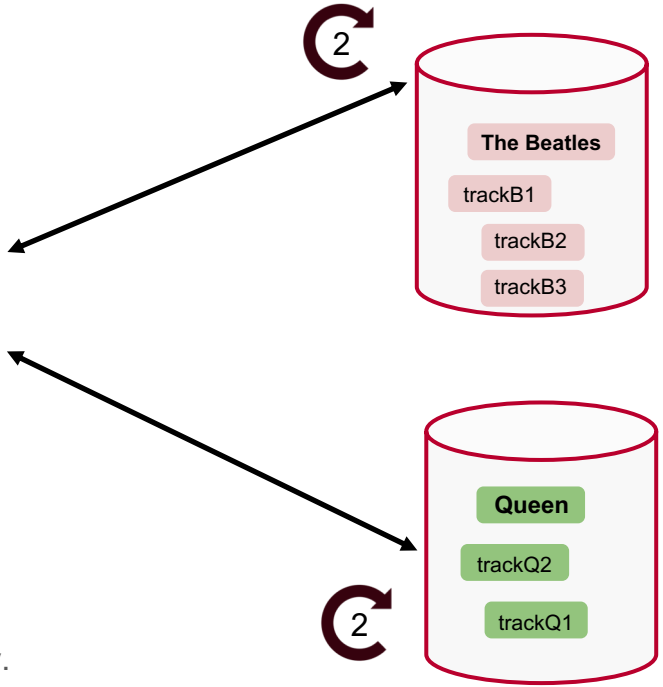
```
CREATE TABLE Track
(
    TrackId INT,
    Name VARCHAR(200),
    AlbumId INT,
    ArtistId INT,
    GenreId INT,
    Composer VARCHAR(220),
    Milliseconds INT,
    PRIMARY KEY (TrackId, ArtistId)
) WITH "affinityKey=ArtistId";
```

JOINs with Co-Located Data

Application

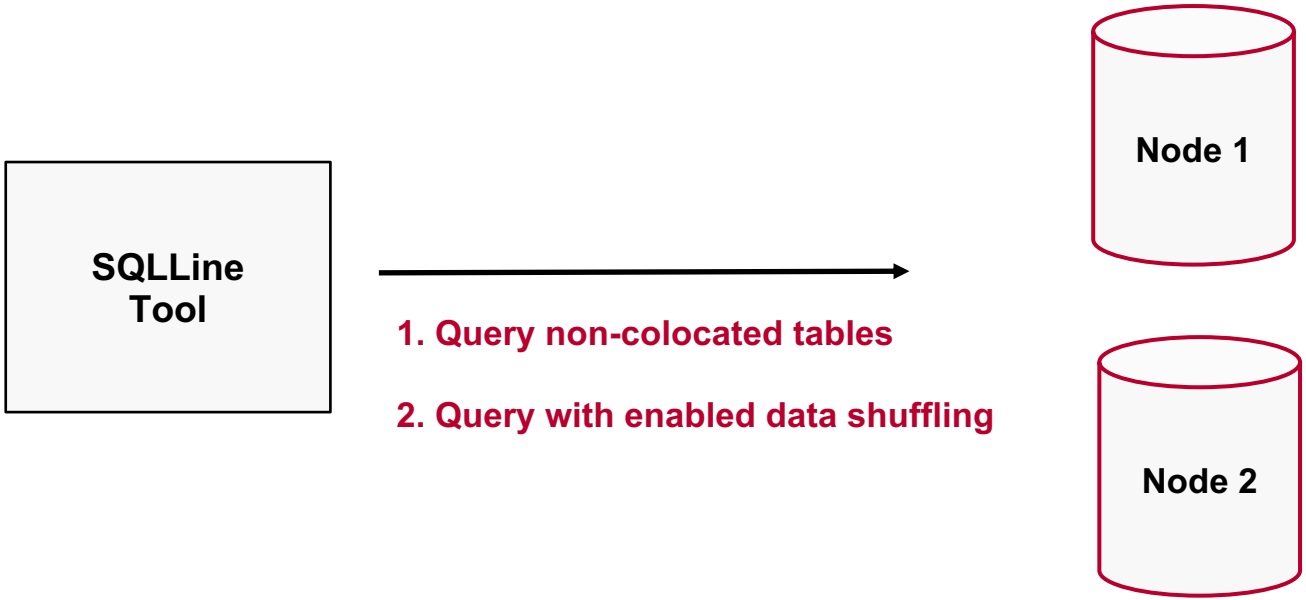
1 & 4

```
SELECT track.trackId, track.name as track_name, genre.name as genre,
artist.name as artist, MAX(milliseconds / (1000 * 60)) as duration
FROM track LEFT JOIN artist ON track.artistId = artist.artistId
JOIN genre ON track.genreId = genre.genreId
WHERE track.genreId < 17
GROUP BY track.trackId, track.name, genre.name, artist.name
ORDER BY duration DESC LIMIT 20;
```



1. The application starts executing the query.
2. The query is executed on the cluster nodes.
3. The result set is reduced.

Hands-On: What You'll Do Next



Hands-On: JOIN Without Data Co-location

Step 1: Make sure that you are connected to the cluster via SQLline.

```
java -cp libs/core.jar sqlline.SqlLine

!connect jdbc:ignite:thin://127.0.0.1/ ignite ignite
```

Step 2: Get the top 20 longest tracks.

```
!run sql/top_20_longest_tracks.sql
```

Hands-On: JOINs Without Data Co-location

Step 3: Get the top 20 with details about the artists and identify data inconsistencies:

```
!run sql/top_20_longest_tracks_with_authors.sql
```

Step 4: Allow data shuffling for JOINs and run the query again:

```
!close

!connect jdbc:ignite:thin://127.0.0.1?distributedJoins=true ignite ignite
```

Demo by Instructor: Co-locating Related Data

Step 1: Co-locate tracks with trtists.

- Change Tracks' primary key to PRIMARY KEY (TrackId, ArtistId).
- Set Track's affinity key to affinityKey=ArtistId.

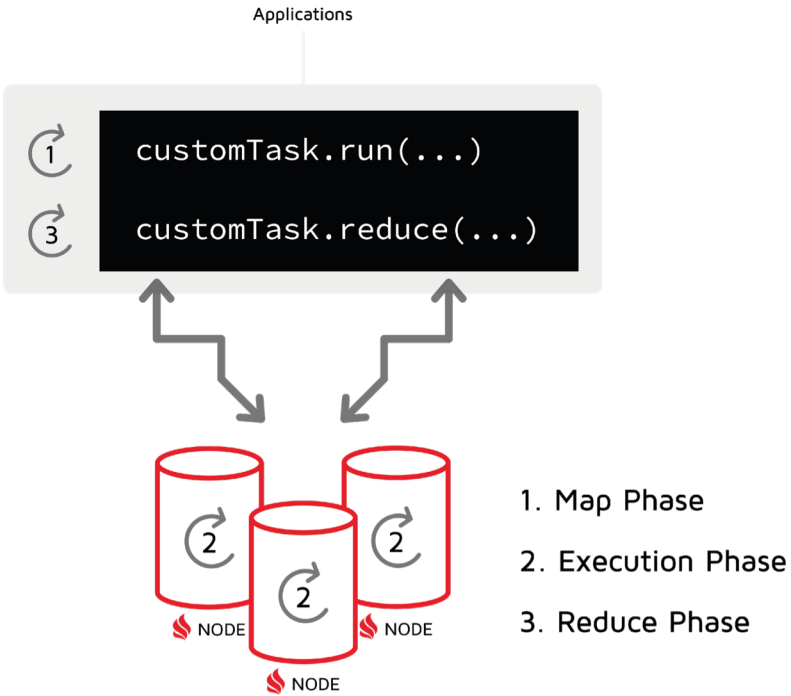
Step 2: Clean the Ignite work directory and restart the cluster.

Step 3: Reconnect with SQLLine and run the last SQL query.

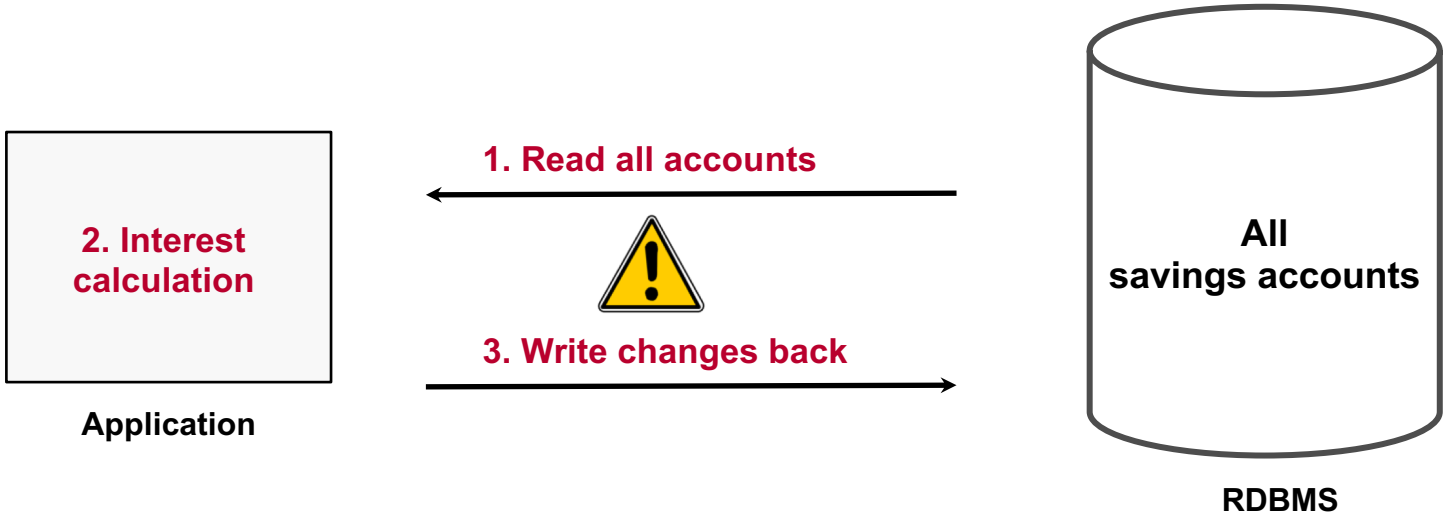
Co-located Computations

Executing data-intensive logic on cluster nodes

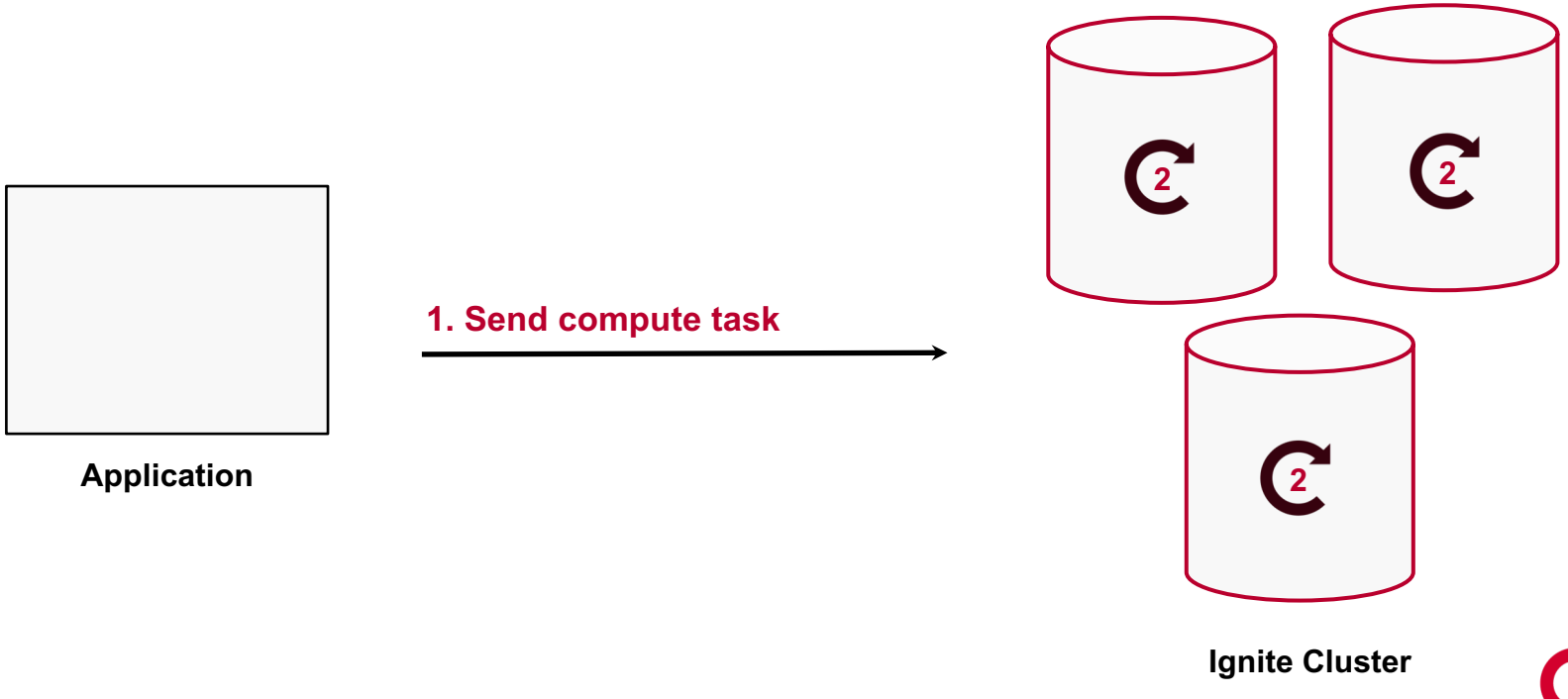
Executing Custom Logic in the Cluster



Case With Millions Savings Accounts



Case With Millions Savings Accounts



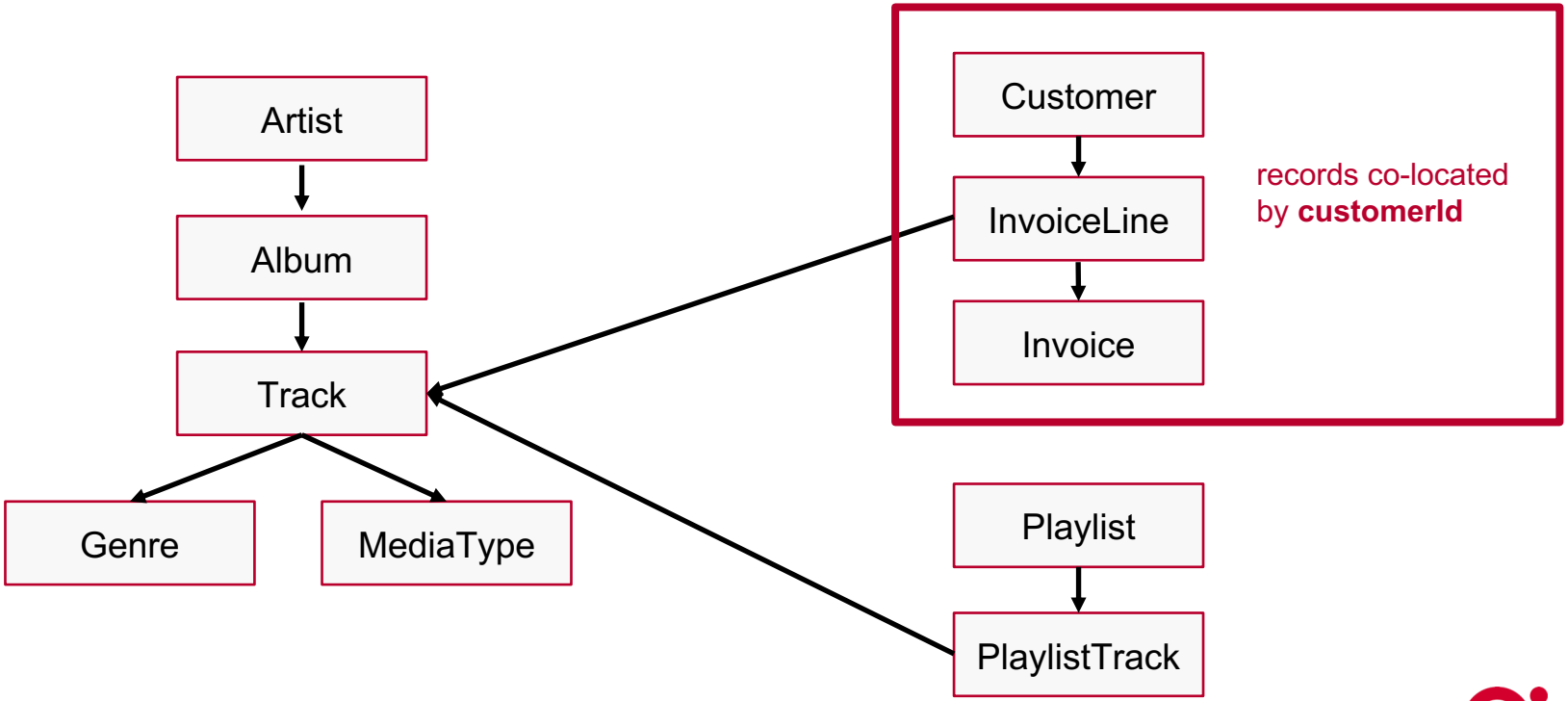
Poll 2

What was the new execution time of the business operation once it was implemented as a compute task?

- 40 - 50 minutes
- 10 - 15 minutes
- 1 - 5 minutes
- Less than a minute

Hands-On: What You'll Do Next

Top-5 Paying Listeners of the Media Store



https://github.com/lerocha/chinook-database

Hands-On: Find Top Paying Listeners

Step 1: Execute the Java logic that returns top-5 paying listeners.

```
java -cp libs/apps.jar training.ComputeApp
```

Step 2: Update the source code to return top-10 paying listeners.

Hands-On: Find the Top-5 Paying Listeners

Step 3: Rebuild the apps' artifact of the project:

```
mvn clean package -P apps
```

Step 4: Execute the application.
The servers will reload the updated version of the computation.

```
java -cp libs/apps.jar training.ComputeApp
```

Summary

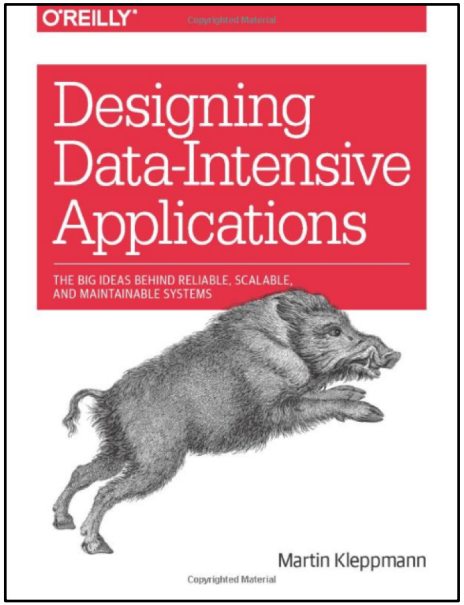
In-memory computing essentials

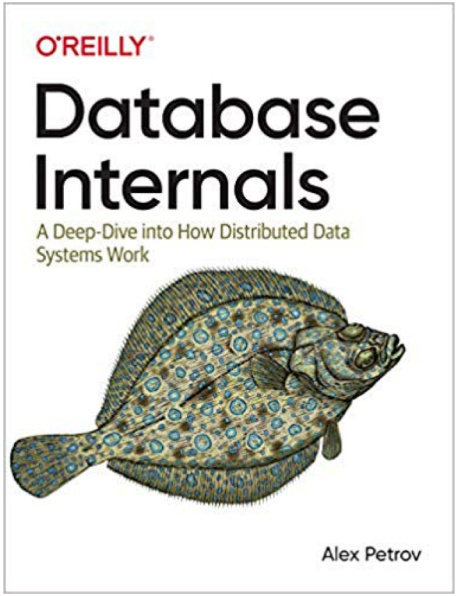
It’s Not Enough to Put Data in RAM

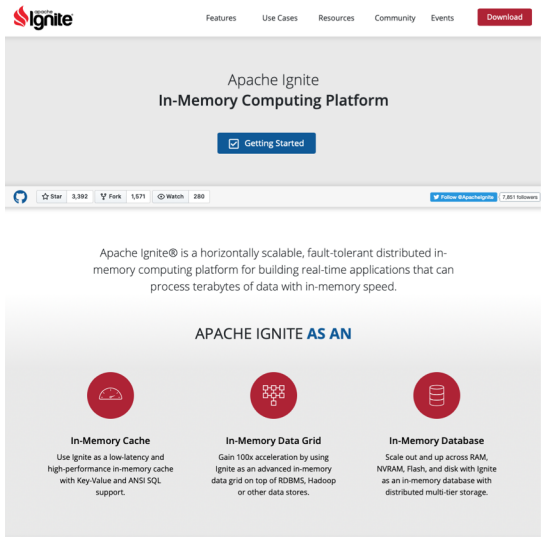
- Scale out to spread data evenly and to balance the load.
  - Understand partitioning basics
  - Know how to tweak default partitioning behavior

- Avoid any negative impact that the network might have on performance
  - Co-locate the related data.
  - Run data-intensive logic on the cluster nodes.

Want to Learn More?







@denismagda

O'REILLY®

Thank you