Better Together – Apache Ignite & Apache Spark

Fast Data Meets Open Source

DMITRIY SETRAKYAN
GridGain Founder & Chief Product Officer
Apache Ignite PMC

VALENTIN KULICHENKO
GridGain Lead Architect
Apache Ignite PMC

http://ignite.apache.org


@apacheignite


@dsetrakyan

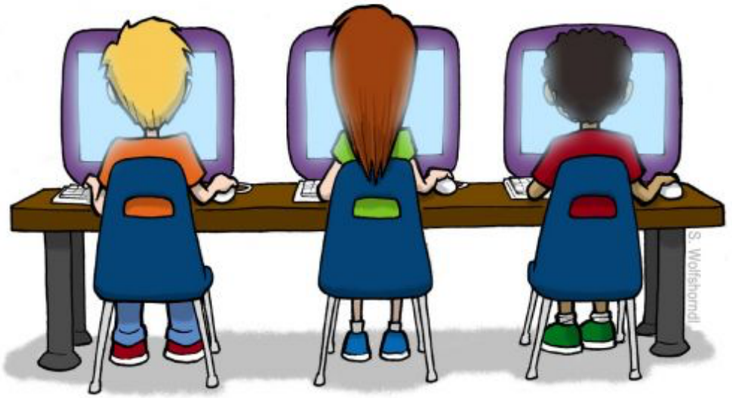
GridGain

Agenda

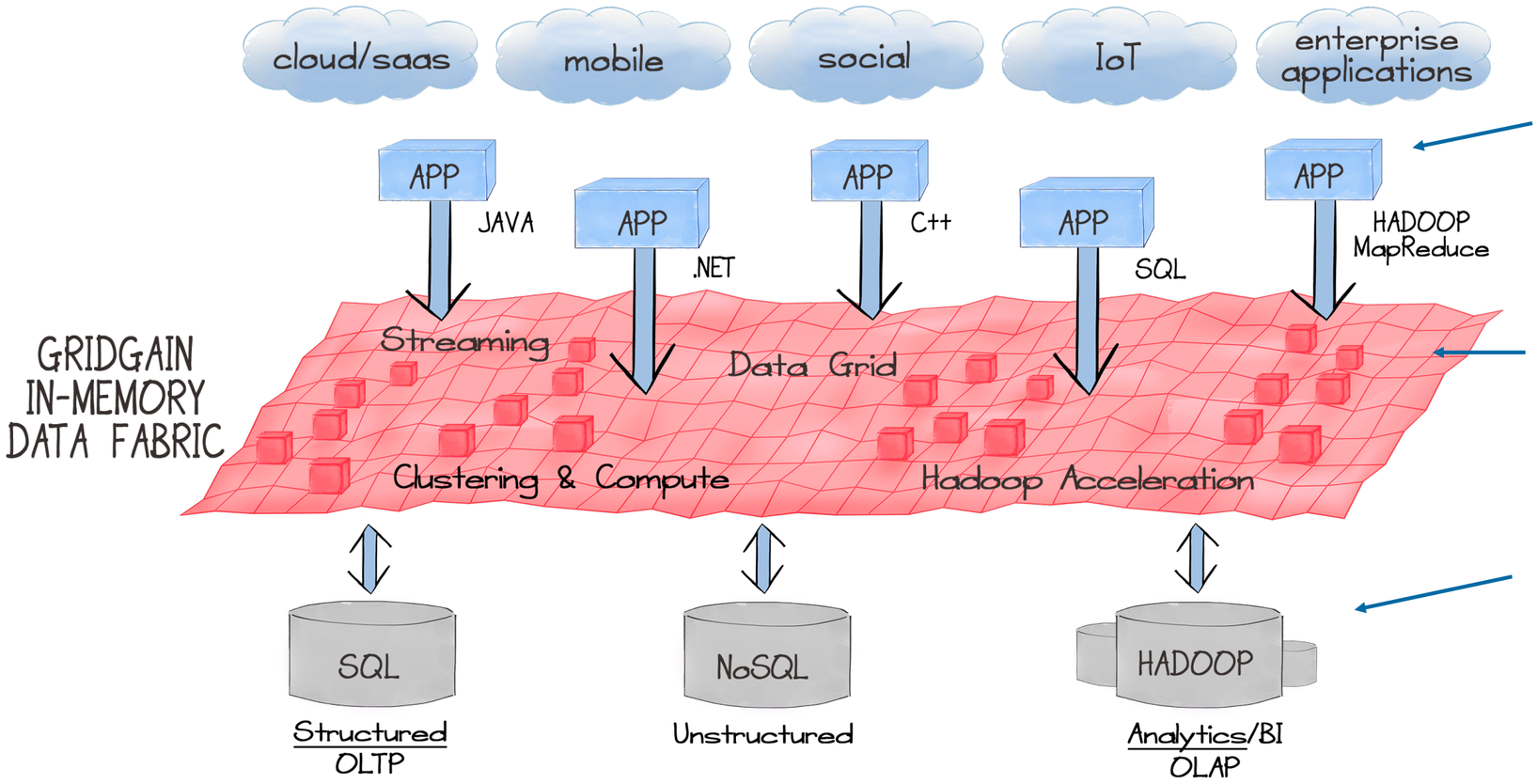
- Apache Ignite(tm) Overview
- Data Grid
  - Partitioning Schemes
  - SQL
- Shared Memory Layer
  - Share Spark RDDs
  - In-Memory File System
  - DevOps: Yarn and Mesos
- Faster MapReduce & Hive
  - Ignite MapReduce
- Demo - Shared Ignite RDDs
- Demo - SQL using Apache Zeppelin
- Q & A

Apache Ignite - We Are Hiring!

- Very Active Community
- Great Way to Learn Distributed Computing
- How To Contribute:
  - https://ignite.apache.org/community/contribute.html#contribute
  - https://cwiki.apache.org/confluence/display/IGNITE/How+to+Contribute



Apache Ignite™ In-Memory Data Fabric: Strategic Approach to IMC

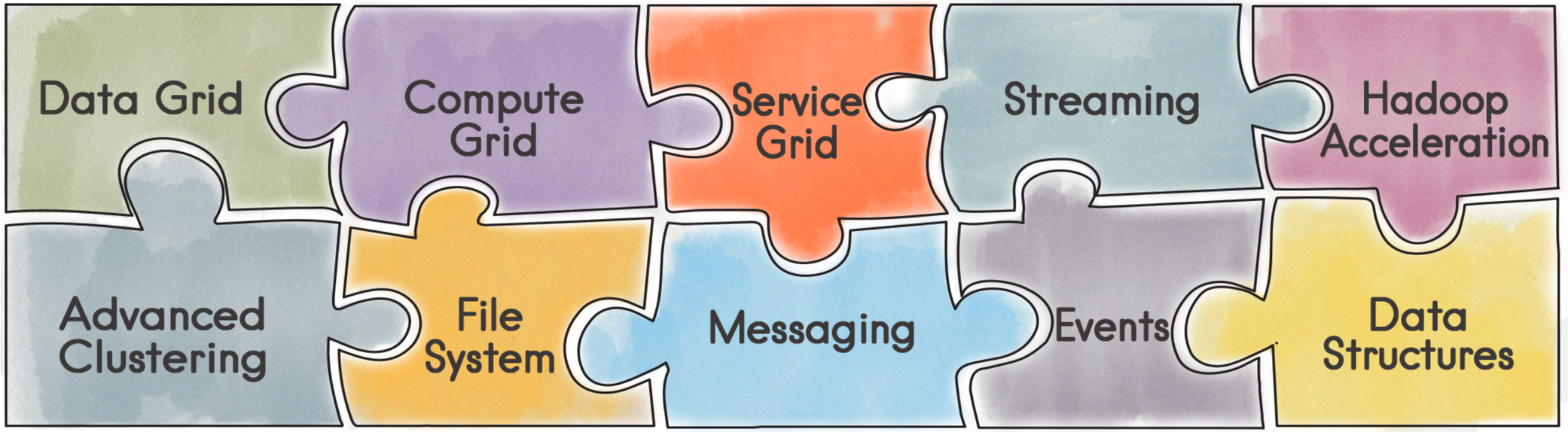


- Supports Applications of various types and languages

- Open Source – Apache 2.0
- Simple Java APIs
- 1 JAR Dependency
- High Performance & Scale
- Automatic Fault Tolerance
- Management/Monitoring
- Runs on Commodity Hardware

- Supports existing & new data sources
- No need to rip & replace

Apache Ignite In-Memory Data Fabric



Why Share State in Spark?

- Long Running Applications
  - Passing State Between Jobs
- Disk File System (HDFS?)
  - Convert RDDs to Disk Files and Back
  - Argh#$%
- Share RDDs In-Memory
  - Native Spark API
  - Native Spark Transformations

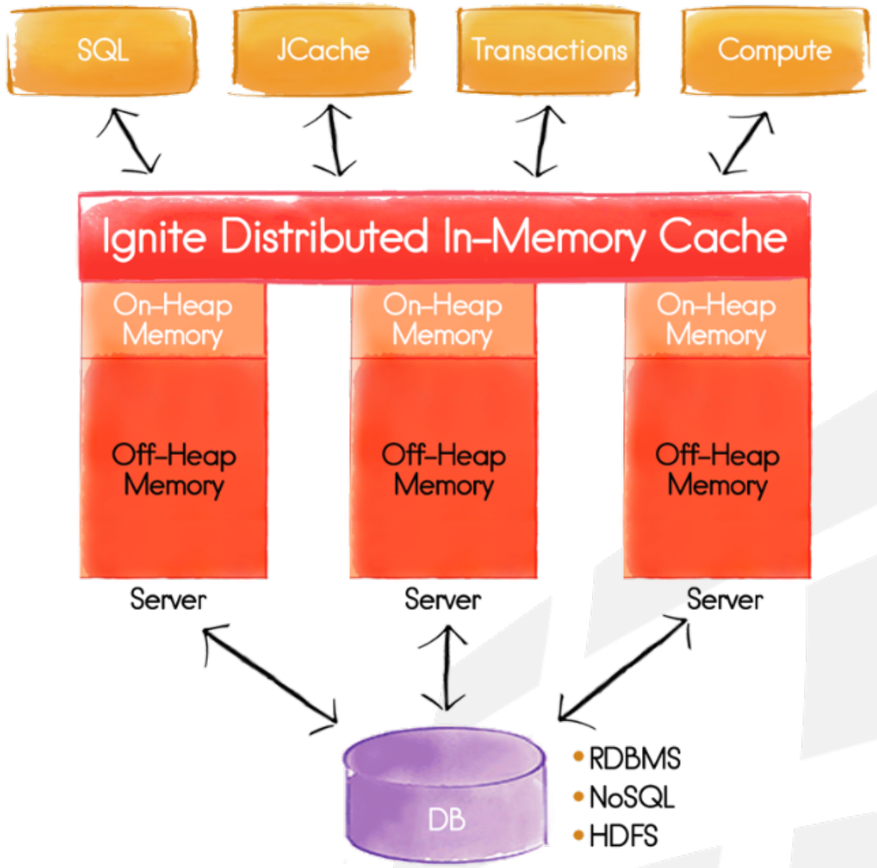
Why Ignite Data Grid?

- In-Memory Key-Value Store
  - Good for Caching Tuples
- Foundation for Shared Memory State
  - IgniteRDD is based on Data Grid
  - Ignite File System is based on Data Grid
- On-Heap & Off-Heap Memory
- In-Memory Indexes
  - **Fast SQL**
- Built for High Throughput and Low Latencies

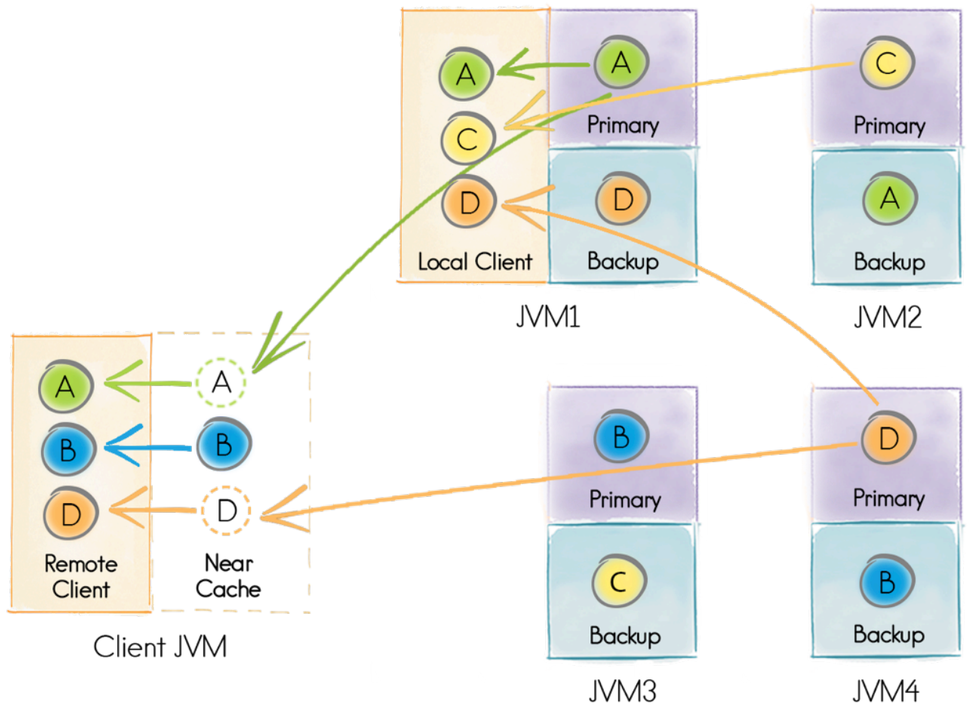
Data Grid: JCache (JSR 107)

- Key-Value Store (JCache, JSR 107)
  - In-Memory Key-Value Store
  - Basic Cache Operations
  - ConcurrentMap APIs
  - Collocated Processing (EntryProcessor)
  - Events and Metrics
  - Pluggable Persistence
- Data Grid
  - ACID Transactions
  - SQL Queries (ANSI 99)
  - In-Memory Indexes
  - On-Heap & Off-Heap Memory
  - Automatic RDBMS Integration

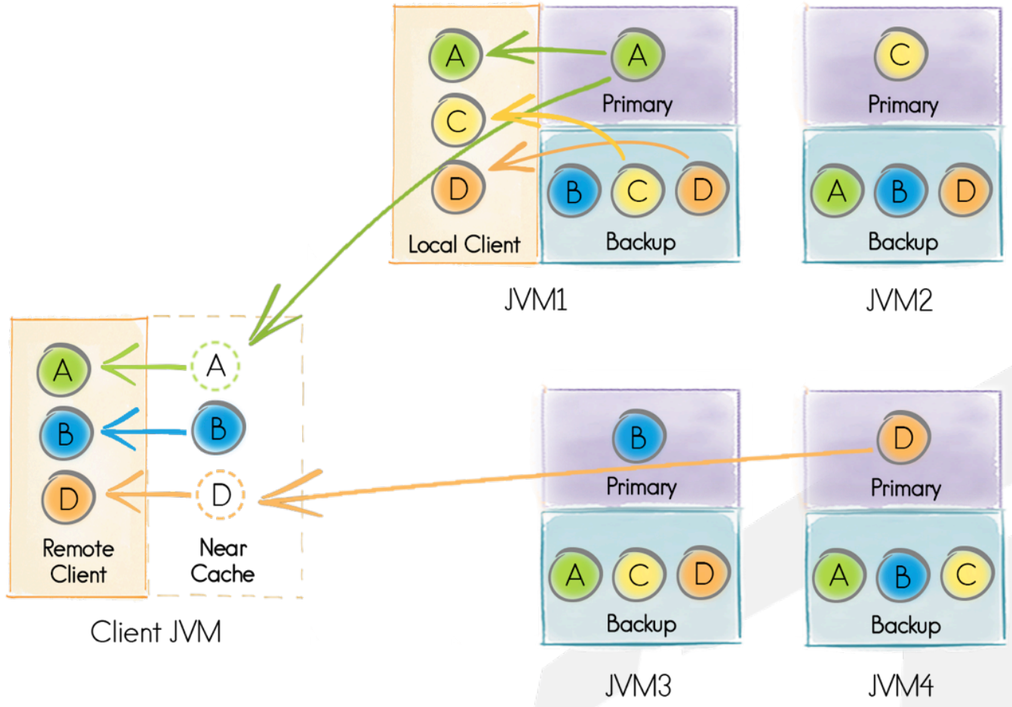
JCache



Data Grid: Distributed Caching



Partitioned Cache



Replicated Cache

Data Grid: Ad-Hoc SQL (ANSI 99)

- ANSI-99 SQL
- Always Consistent
- Fault Tolerant
- In-Memory Indexes (On-Heap and Off-Heap)
- Automatic Group By, Aggregations, Sorting
- Cross-Cache Joins, Unions, etc.
- Ad-Hoc SQL Support

SQL Cross-Cache GROUP BY Example

```
IgniteCache<AffinityKey<UUID>, Person> cache = ignite.cache("persons");

// Query to get salaries grouped by organization.
SqlFieldsQuery qry = new SqlFieldsQuery(
    "select org.name, avg(salary), max(salary), min(salary) " +
        "from Person, \"Organizations\".Organization as org " +
        "where Person.orgId = org.id " +
        "group by org.name " +
        "order by org.name");

QueryCursor<List<?>> cursor = cache.query(qry);

List<List<?>> res = cursor.getAll();
```

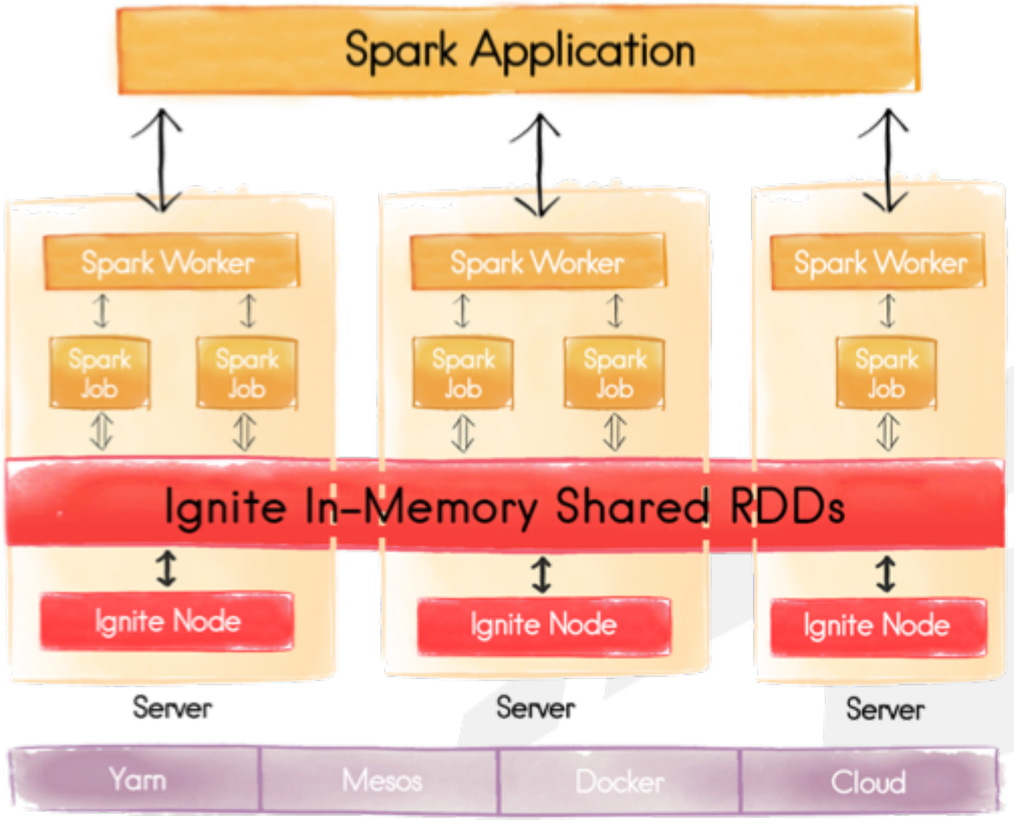
Apache Ignite for Spark and Hadoop

DevOps: Integration with Yarn and Mesos

- Automatic Resource Management
- Easy Data Center Installation
- Easy Data Center Configuration
- On-Demand Elasticity

Share RDDs Across Spark Jobs

- IgniteRDD Deployment Modes
  - Share RDD across tasks on the host
  - Share RDD across tasks in the application
  - Share RDD globally
  - Embedded vs External Deployments
- Faster SQL
  - In-Memory Indexes
  - SQL on top of Shared RDD



IgniteContext

- Main Entry Point from Spark to Ignite
- Specify Different Ignite Configurations
- Embedded vs External Deployments
  - Client vs Server Modes

```
val igniteContext = new IgniteContext[Integer, Integer](sparkContext,
    () => new IgniteConfiguration())
```

IgniteRDD

- Implementation of SparkRDD
- Mutable (unlike native RDDs)
- Partitioned over Ignite Partitioned Caches
- Indexed SQL
  - Spark only does Full Scans
  - Indexes are 1000x faster

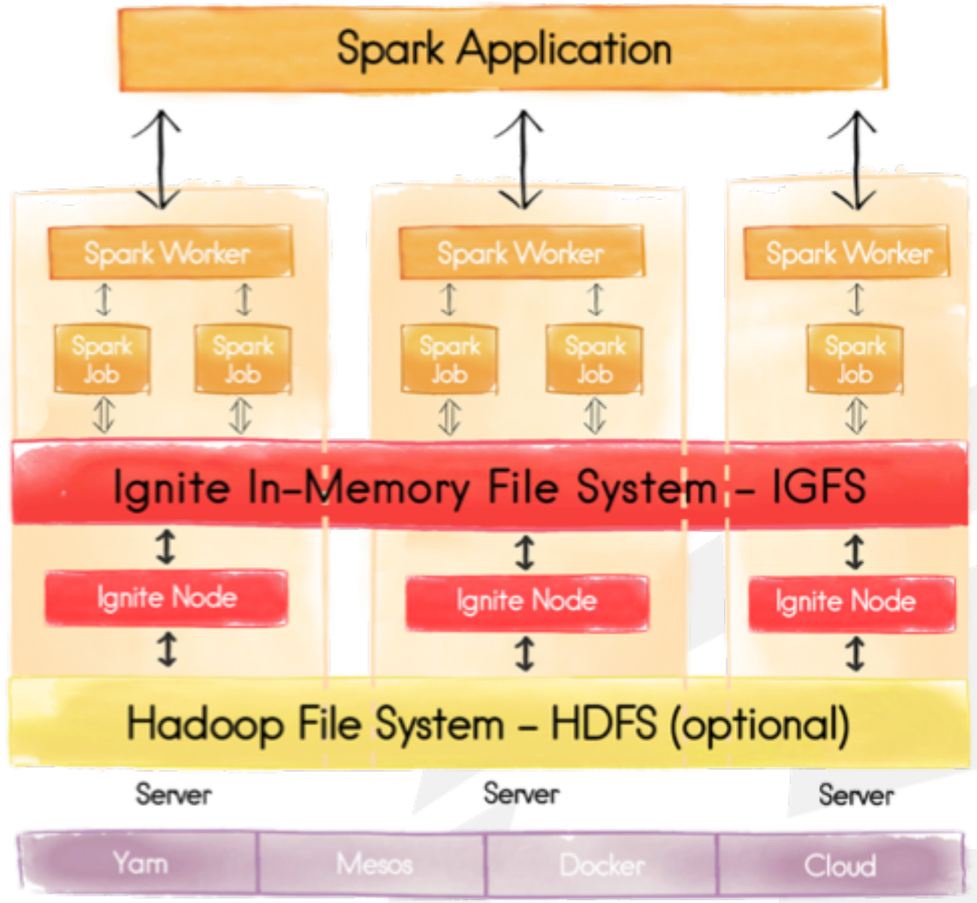
```
val cacheRdd = igniteContext.fromCache("partitioned")

cacheRdd.savePairs(sparkContext.parallelize(1 to 10000, 10).map(i => (i, i)))

val result = cacheRdd.sql(
  "select _val from Integer where val > ? and val < ?", 10, 100)
```

Ignite In-Memory File System

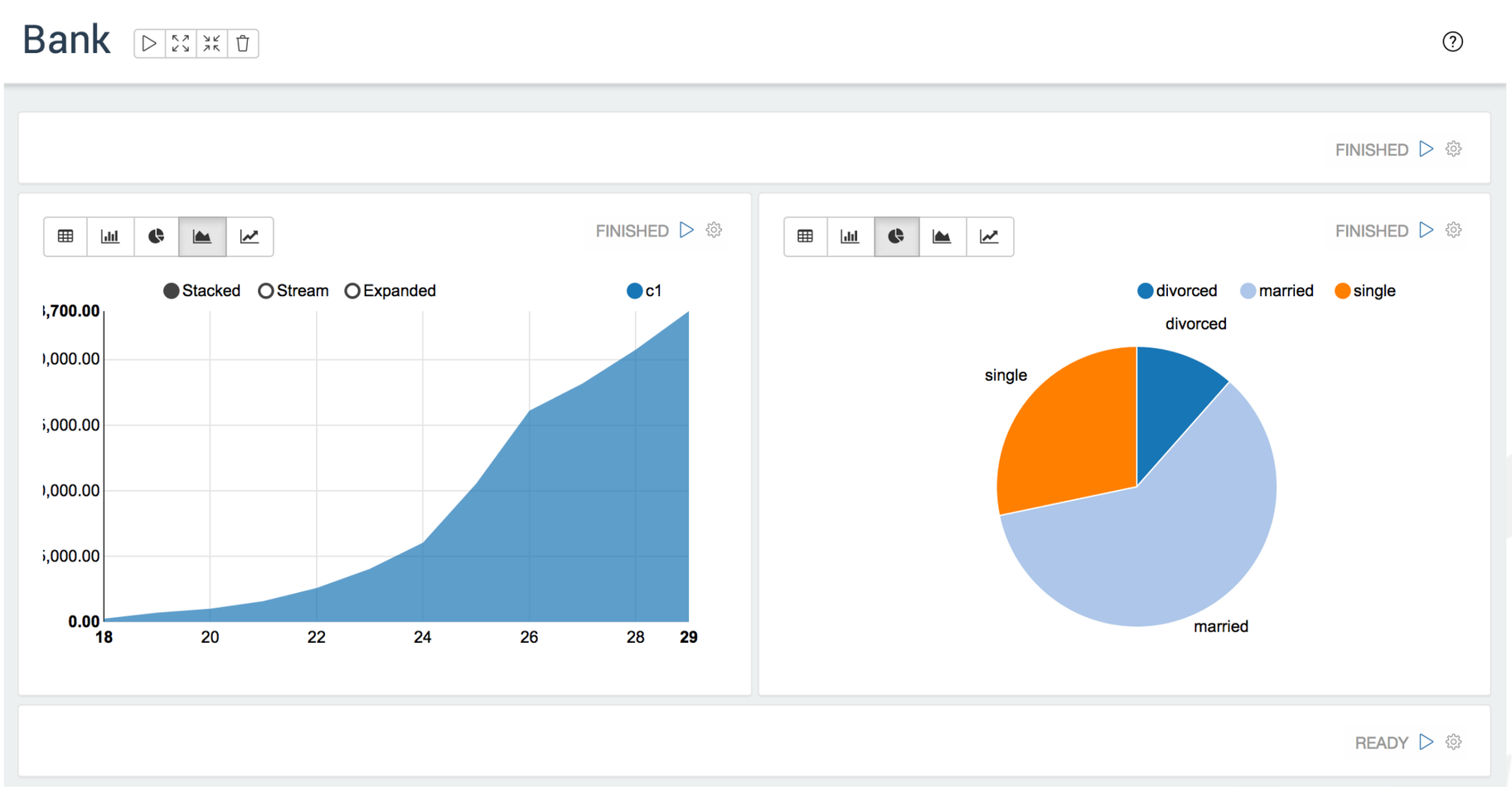
- Ignite In-Memory File System (IGFS)
  - Hadoop-compliant
  - Easy to Install
  - On-Heap and Off-Heap
  - Caching Layer for HDFS
  - Write-through and Read-through HDFS
  - Performance Boost



Apache Ignite Roadmap

- Non-Collocated Joins (released in 1.7)
- Data Modification Language (DML in 2.0)
  - INSERT, UPDATE, DELETE
- Data Definition Language (DDL in 2.1)
  - CREATE, ALTER, DROP
- More IGFS Performance
- Native Data Frame Integration

Interactive SQL with Apache Zeppelin



GridGain

ANY QUESTIONS?

Thank you for joining us. Follow the conversation.

http://www.ignite.apache.org

 @apacheignite

 @dsetrakyan