Better Together: Fast Data with Ignite & Spark

Christos Erotocritou - Spark Summit EU 2016

Agenda

- GridGain & Apache Ignite Project

- Ignite In-Memory Data Fabric

- Apache Ignite vs. Apache Spark

- Hadoop & Spark Integration

- Q & A

GridGain

Apache Ignite Project

- **2007:** First version of GridGain

- **Oct. 2014:** GridGain contributes Ignite to ASF

- **Aug. 2015:** Ignite is the second fastest project to graduate after Spark

- **Today:**

  - 82+ contributors and growing rapidly

  - Huge development momentum - Estimated 233 years of effort since the first commit in February, 2014
    [Openhub]
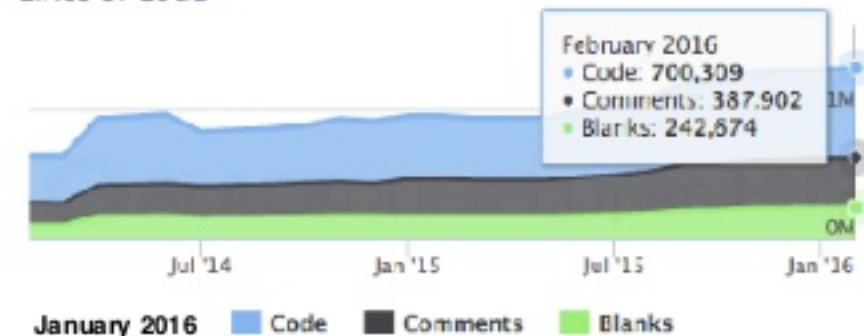
  - Mature codebase: 840k+ SLOC & more than 17k commits

Contributors per Month

November 2015
Contributors: 34

Jul '14   Jan '15   Jul '15   Jan '16

Languages

Java 82%   C# 7%
15 Other 11%

Lines of Code

February 2016
Code: 700,309
Comments: 387,902
Blanks: 242,874

Jul '14   Jan '15   Jul '15   Jan '16

January 2016   Code   Comments   Blanks

GridGain

- **GridGain Enterprise Edition**
  - Is a binary build of Apache Ignite™ created by GridGain
  - Added enterprise features for enterprise deployments
  - Earlier features and bug fixes by a few weeks
  - Heavily tested

Customer Use Cases

Automated Trading Systems
Real time analysis of trading positions & market risk. High volume transactions, ultra low latencies.

Financial Services
Fraud Detection, Risk Analysis, Insurance rating and modelling.

Online & Mobile Advertising
Real time decisions, geo-targeting & retail traffic information.

Big Data Analytics
Customer 360 view, real-time analysis of KPIs, up-to-the-second operational BI.

Online Gaming
Real-time back-ends for mobile and massively parallel games.

SaaS Platforms & Apps
High performance next-generation architectures for Software as a Service Application vendors.

Travel & E-Commerce
High performance next-generation architectures for online hotel booking.

Ignite   GridGain
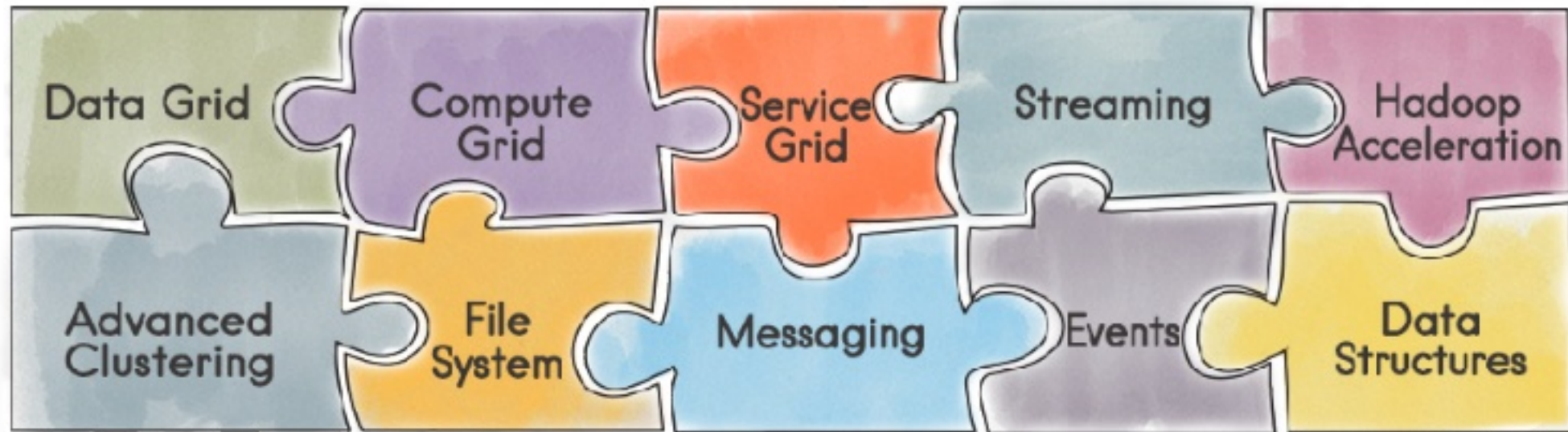
What is an IMDF?

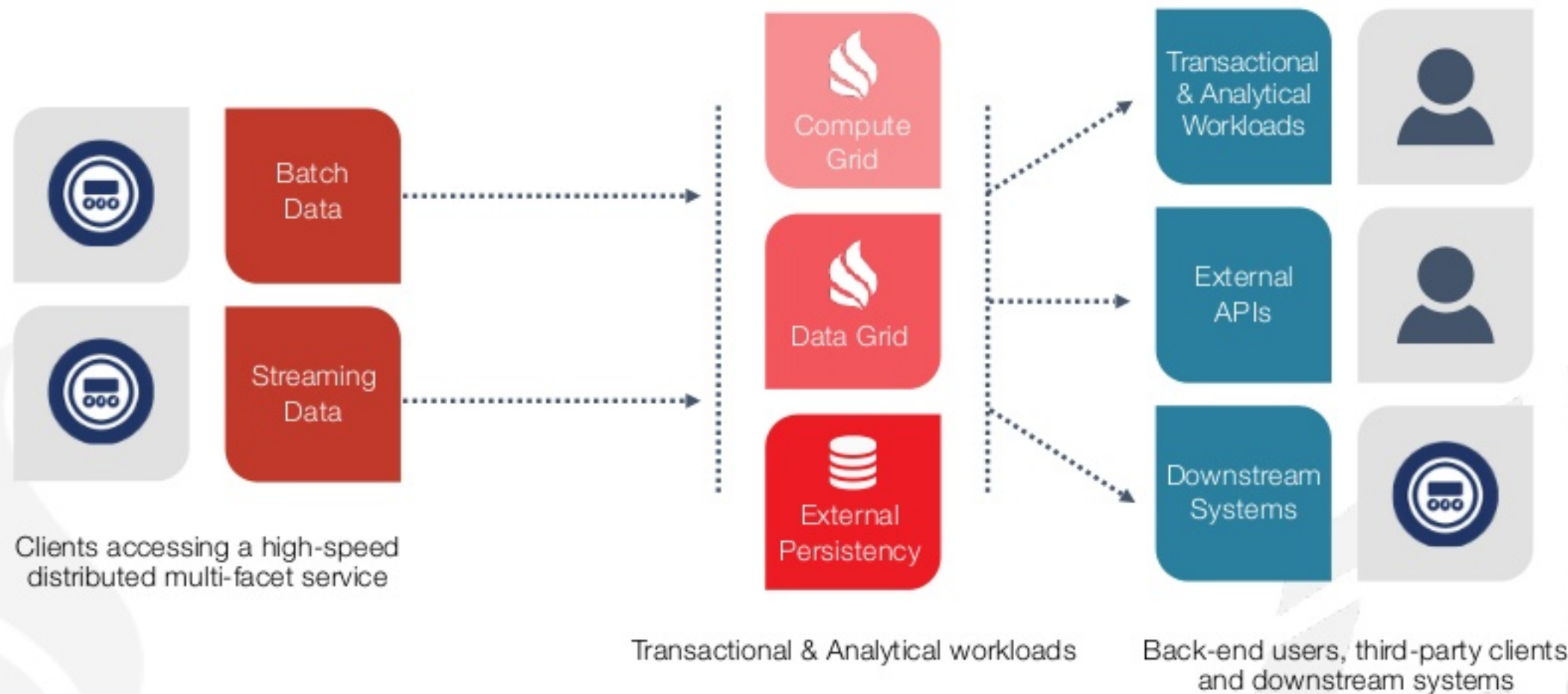
High-performance distributed in-memory platform for computing and transacting on large-scale data sets in near real-time.
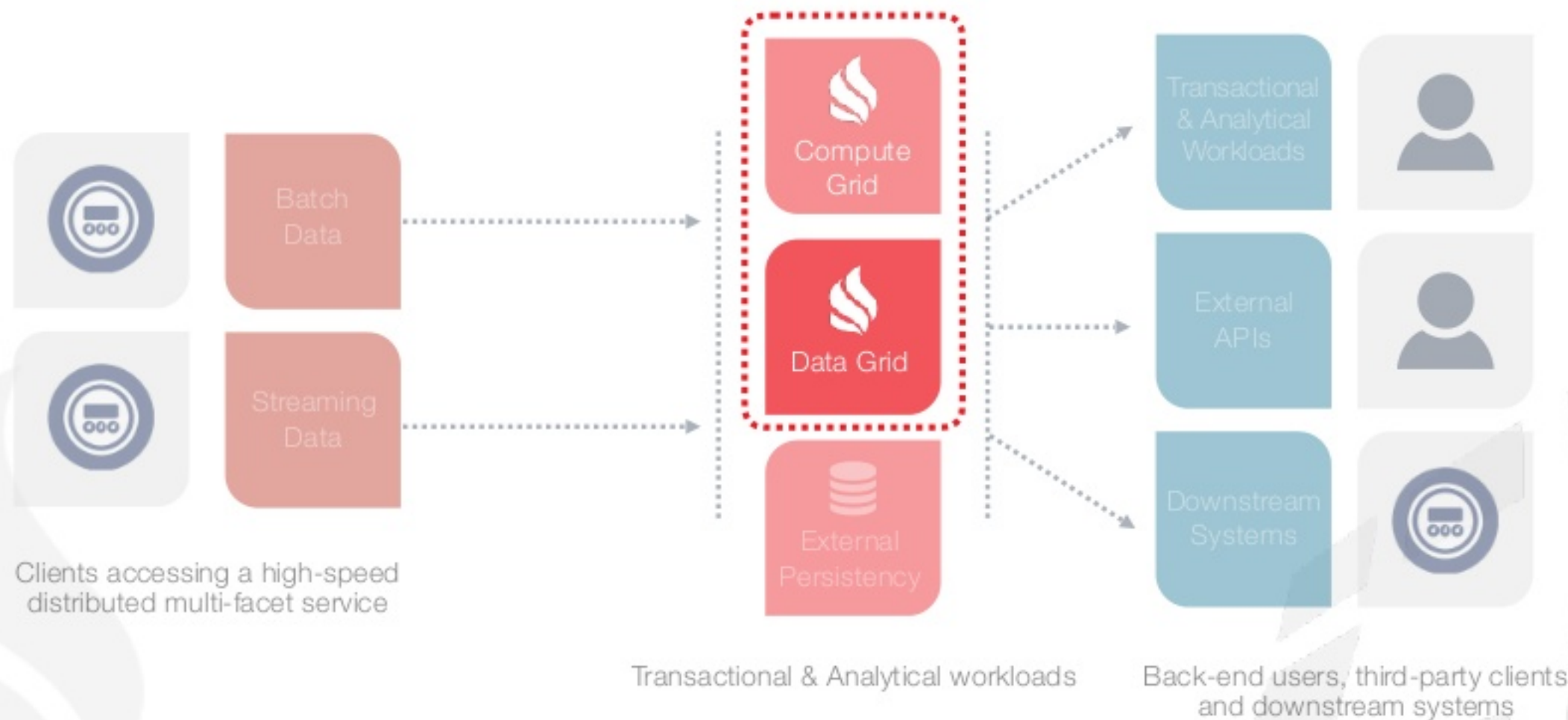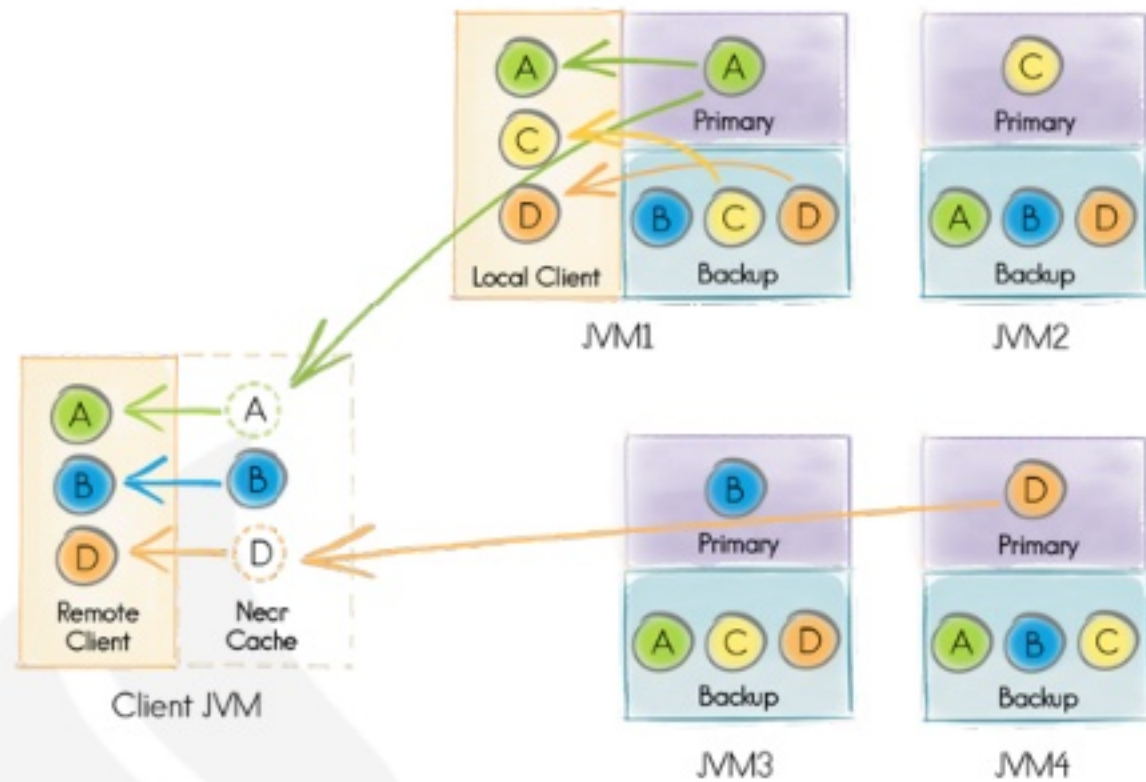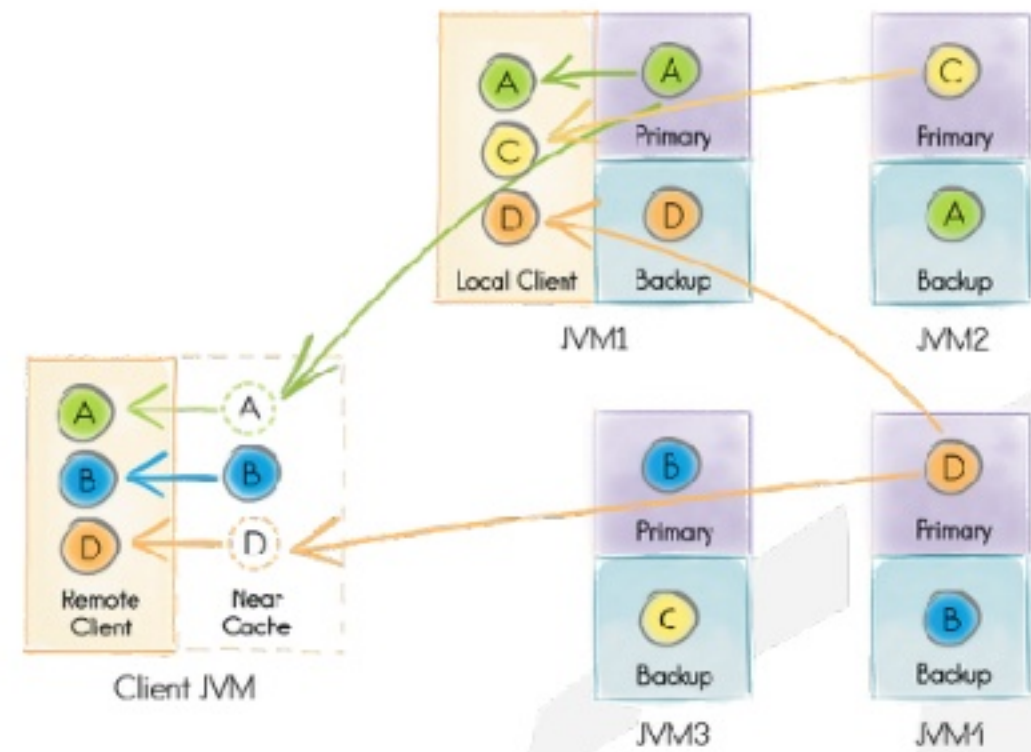
What is an IMDF?

- Web-session clustering
- Distributed caching
- In-Memory SQL

- HPC
- Machine learning
- Risk analysis
- Grid computing

- HA API Services
- Scalable Middleware

- Real-time Analytics
- Big Data
- Monitoring tools

- Big Data
- Realtime Analytics
- Batch processing

Data Grid | Compute Grid | Service Grid | Streaming | Hadoop Acceleration

Advanced Clustering | File System | Messaging | Events | Data Structures

- Fault Tolerance
- Multiple backups
- Cluster groups
- Auto Rebalancing

- Distributed In-Memory File System

- Node2Node & Topic-based Messaging

- Complex event processing
- Event driven design

- Distributed queues
- Atomic variables
- Dist. Semaphore

Ignite   GridGain

In-Memory Computing Platform

Batch Data

Streaming Data

Clients accessing a high-speed distributed multi-facet service

Compute Grid

Data Grid

External Persistency

Transactional & Analytical workloads

Transactional & Analytical Workloads

External APIs

Downstream Systems

Back-end users, third-party clients and downstream systems

Ignite   GridGain

Scalability & Resilience with Ignite

Clients accessing a high-speed distributed multi-facet service

Compute Grid

Data Grid

External Persistency

Transactional & Analytical workloads

Batch Data

Streaming Data

Transactional & Analytical Workloads

External APIs

Downstream Systems

Back-end users, third-party clients and downstream systems

Fault Tolerance & Horizontal Scalability

Replicated Cache

Partitioned Cache

Local Store & Vertical Scale

- Tiered Memory
  - On-Heap -> Off-Heap -> Disk
- Persistent On-Disk Store
- Fast Recovery
- Local Data Reload
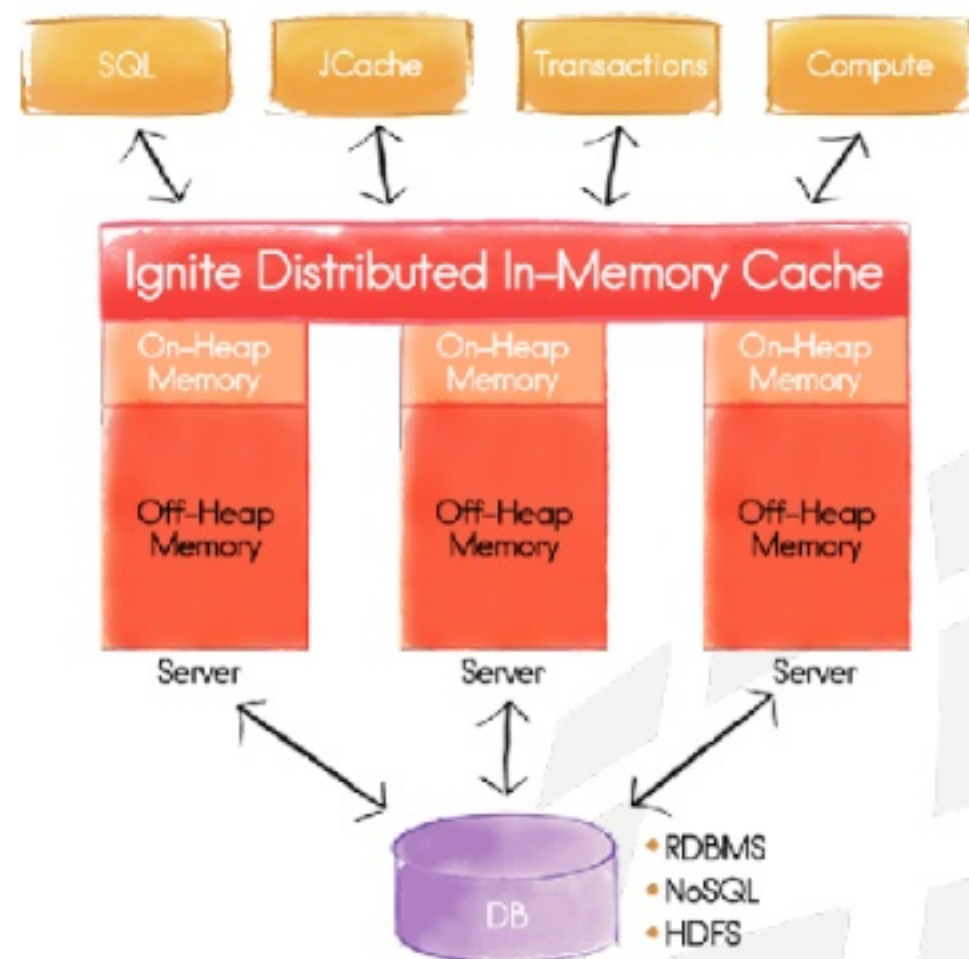 - Eliminate Network and Db impacts when reloading in-memory store

GRIDGAIN
TIERED STORGAGE MODEL

LATENCY

DRAM   HDD, FLASH

LOCAL SWAP SPACE

OFF-HEAP JVM

ON-HEAP JVM

CAPACITY

LOCAL RESTARTABLE STORE
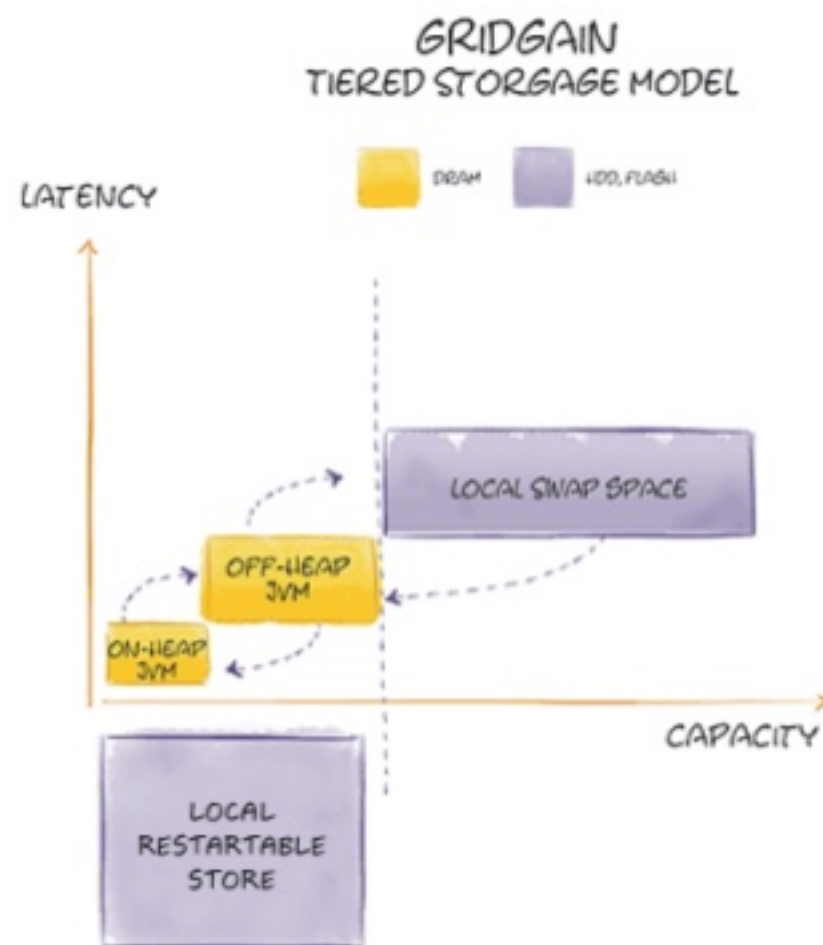
Storage and Caching using Ignite

Clients accessing a high-speed distributed multi-facet service

Transactional & Analytical workloads

Back-end users, third-party clients and downstream systems
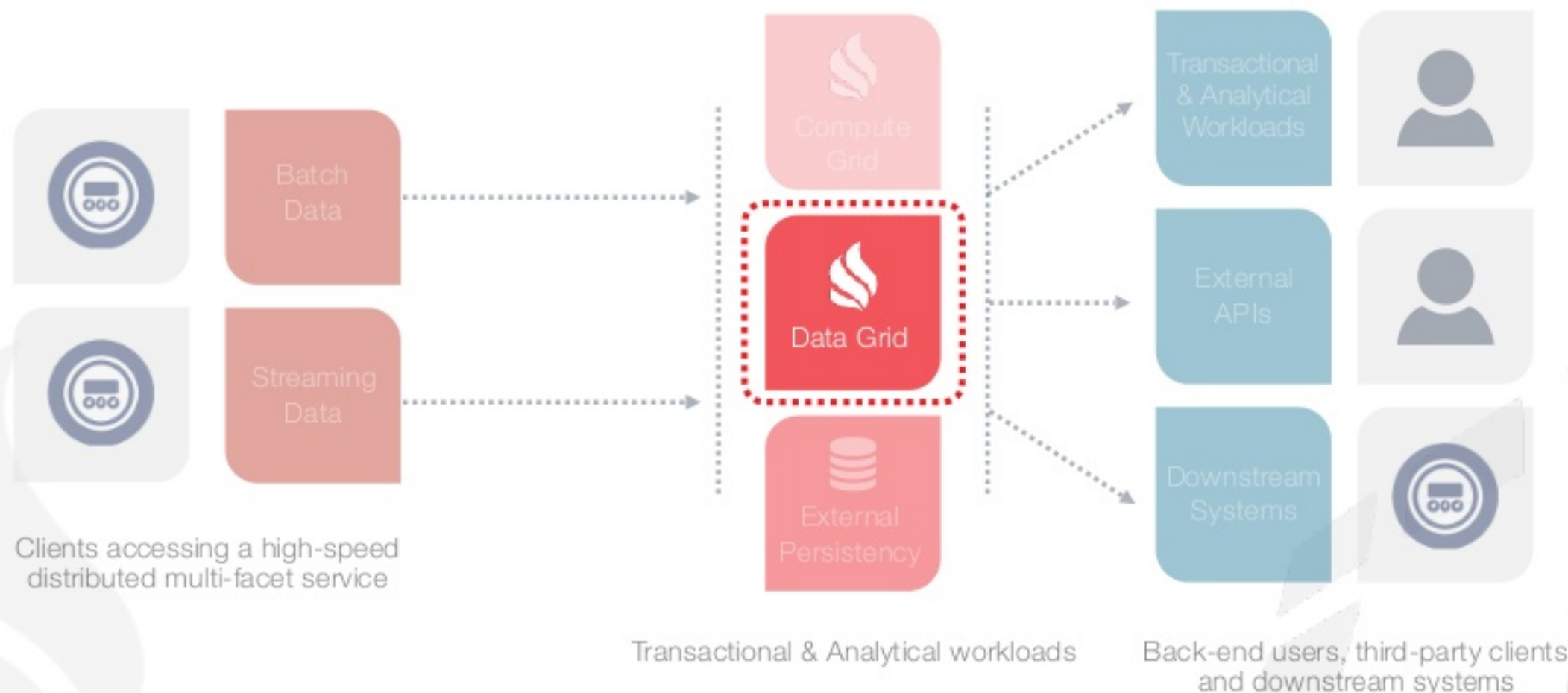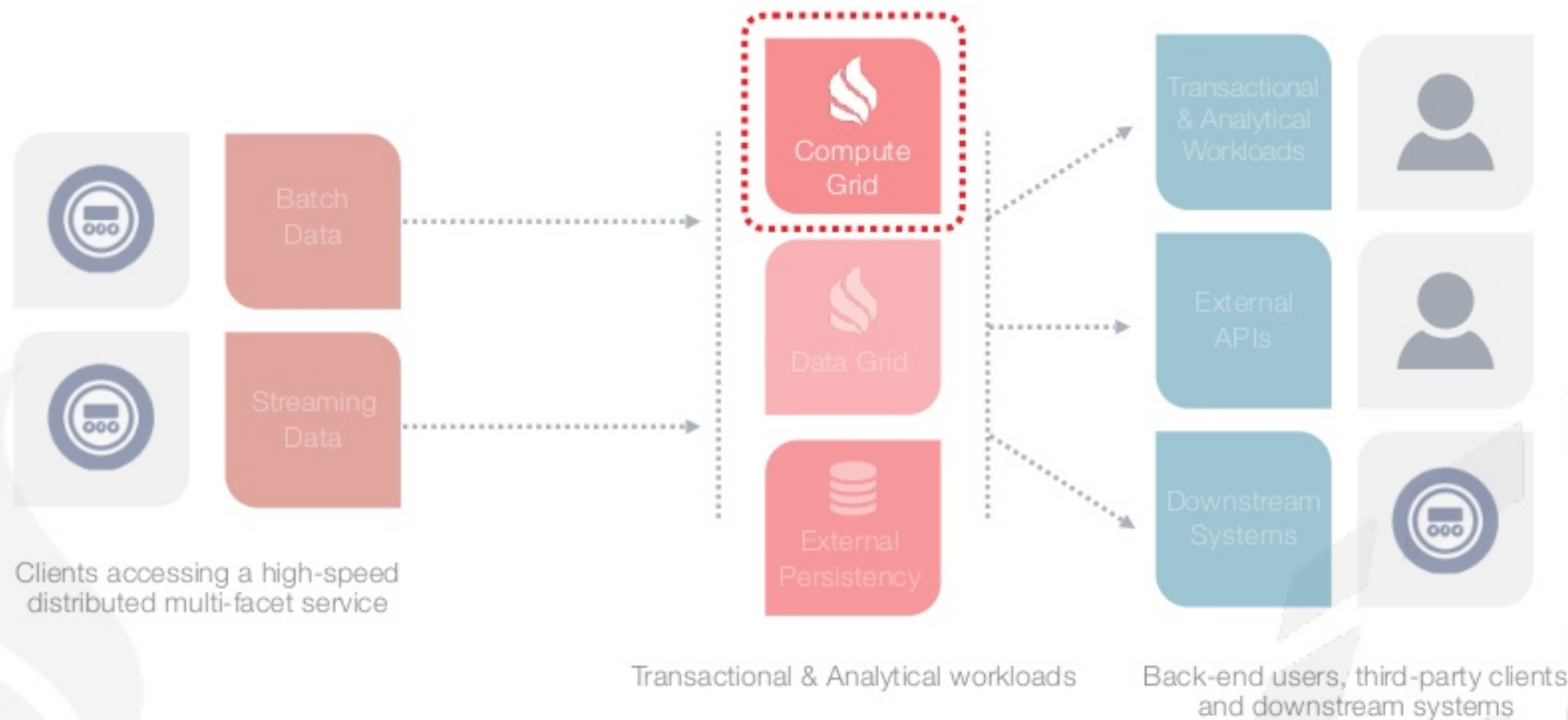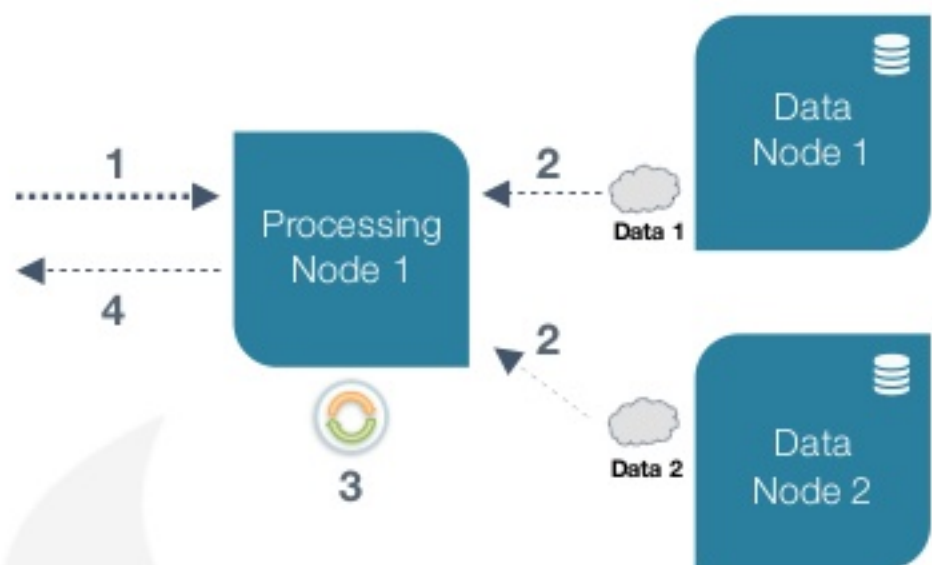
In-Memory Data Grid

- 100% JCache Compliant (JSR 107)
  - Basic Cache Operations
  - Concurrent Map APIs
  - Collocated Processing (EntryProcessor)
  - Events and Metrics
  - Pluggable Persistence

- Ignite Data Grid
  - Fault Tolerance and Scalability
  - Distributed Key-Value Store
  - SQL Queries (ANSI 99)
  - ACID Transactions
  - In-Memory Indexes
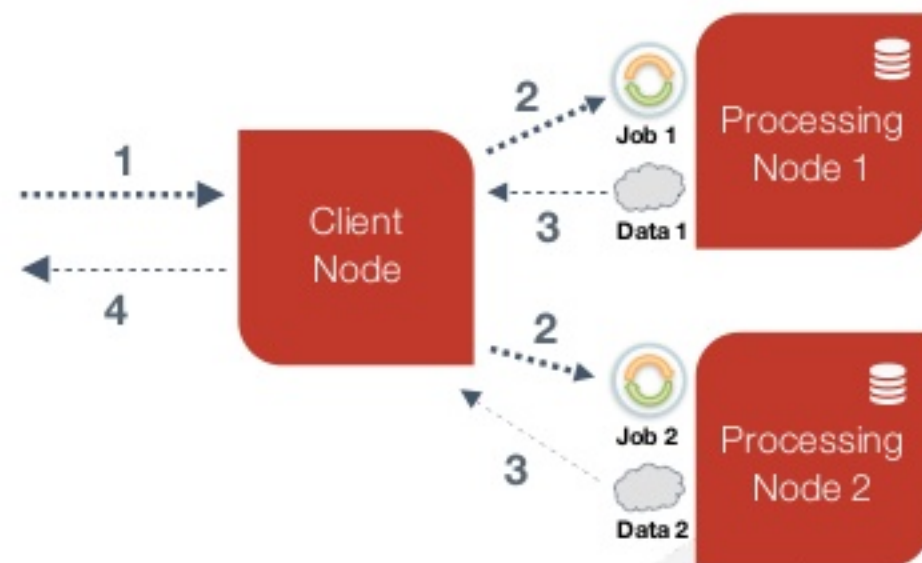 - RDBMS / NoSQL Integration

Distributed Computing with Ignite

Clients accessing a high-speed distributed multi-facet service

Transactional & Analytical workloads

Back-end users, third-party clients and downstream systems

# Client-Server vs. Affinity Colocation

1. Initial Request
2. Fetch data from remote nodes
3. Process entire data-set
4. Return to client

1. Initial Request
2. Co-locating processing with data
3. Return partial result
4. Reduce & return to client

In-Memory Compute Grid

- Direct API for MapReduce
- Cron-like Task Scheduling
- State Checkpoints
- Load Balancing
  - Round-robin
  - Random & weighted
- Automatic Failover
- Per-node Shared State
- Zero Deployment
  - Distributed class loading
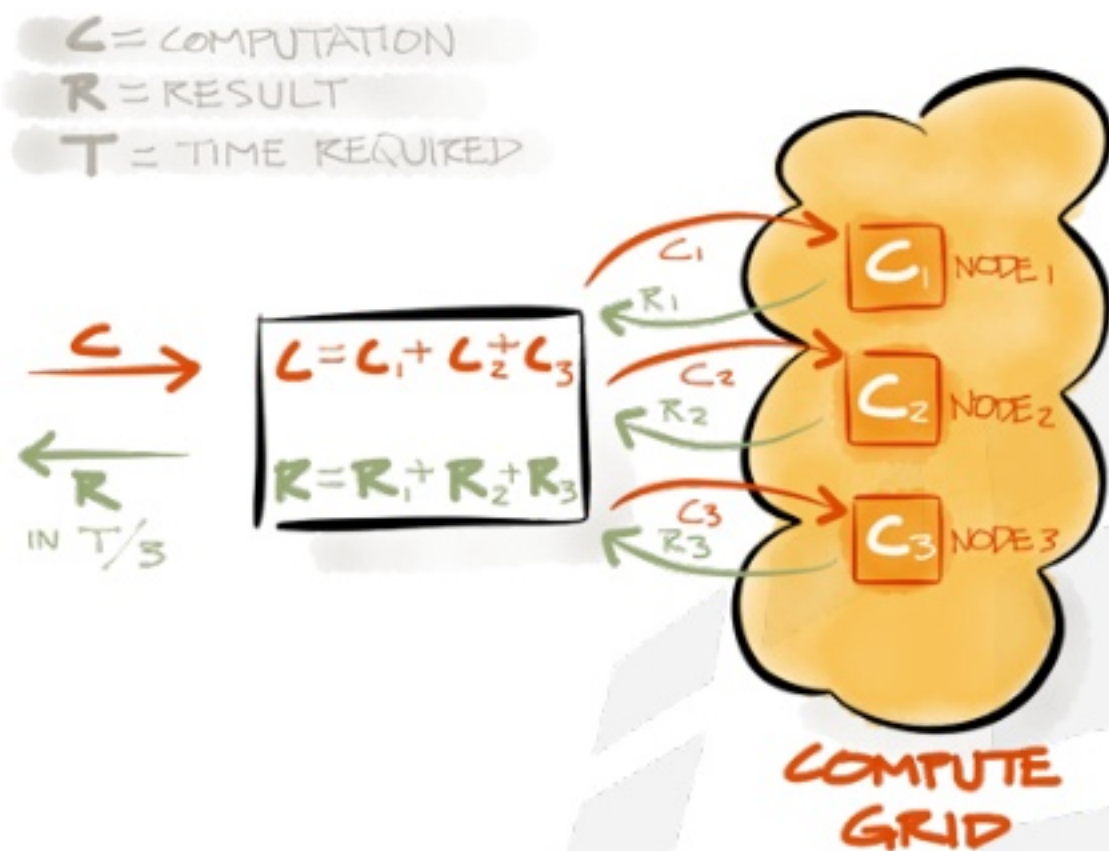
# Hadoop & Spark Integration

Apache Ignite

- Data source agnostic
- Fully fledged compute engine and resilient data storage in-memory for OLAP & OLTP
- Zero-deployment
- In-Memory SQL support
- Fully ACID transactions across memory and disk
- Broader in-memory system that is less focused on Hadoop
- Off-heap memory to avoid GC pauses
- In production since 2007

Apache Spark

- Ingests data from HDFS or another distributed file system
- Inclined towards analytics (OLAP) and focused on MR-specific payloads
- Requires the creation of RDD and data and processing operations are governed by it
- Basic disk-based SQL support
- Strong ML libraries
- Big community

Spark & Ignite Integration

- IgniteRDD
  - Share RDD across jobs on the host
  - Share RDD across jobs in the application
  - Share RDD globally
- Faster SQL
  - In-Memory Indexes
  - SQL on top of Shared RDD
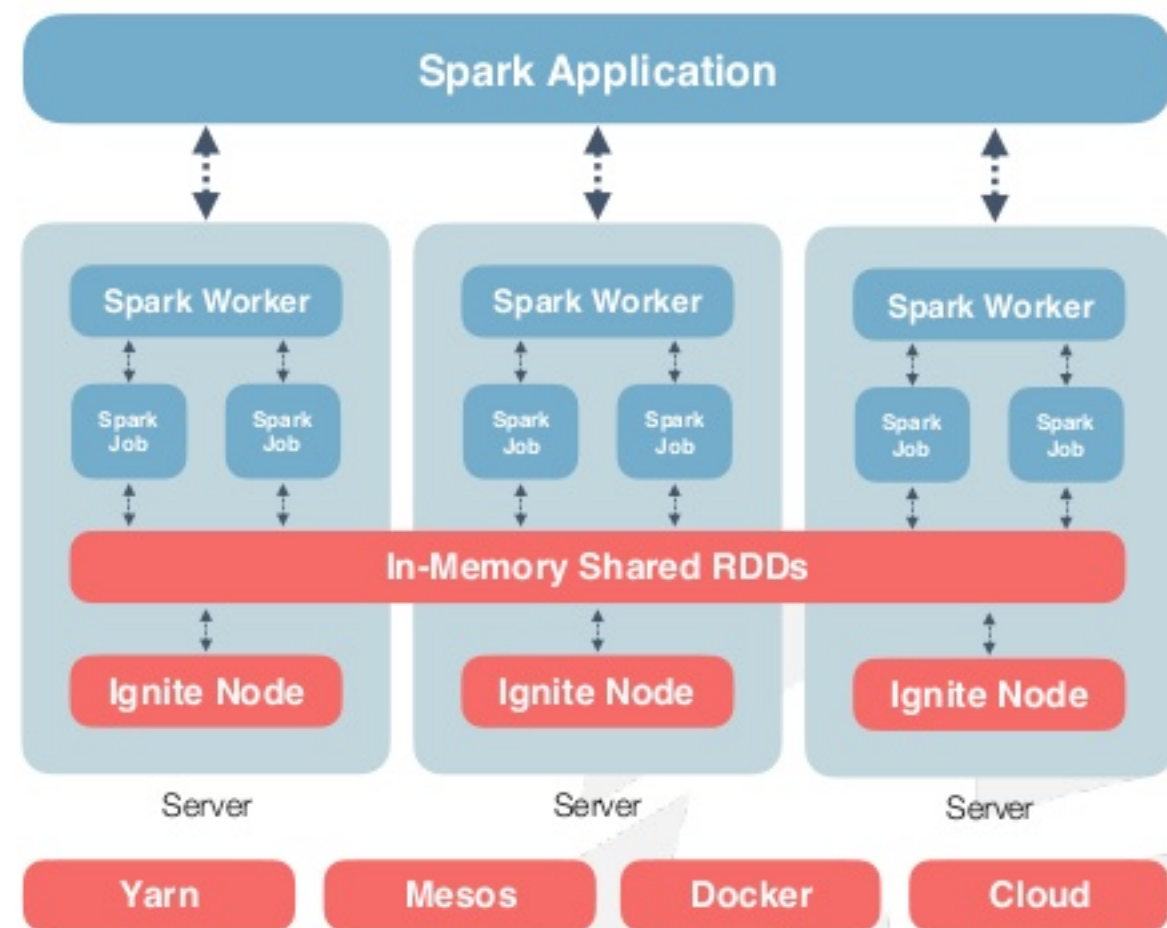
Spark & Ignite Integration: IgniteRDD

- IgniteContext is the main entry point to Spark-Ignite integration:

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

- Reading values from Ignite:

```
val cache = igniteContext.fromCache("myRdd")
val result = cache.filter(_._2.contains("Ignite")).collect()
```

- Saving values to Ignite:

```
val cacheRdd = igniteContext.fromCache("myRdd")
  cacheRdd.savePairs(sparkContext.parallelize(1 to 10000, 10).map(i => (i, i)))
```

- Running SQL queries against Ignite Cache:

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

Spark Integration: Using Dataframes from IgniteRDDs

```scala
// Create an IgniteRDD

val companyCacheIgnite = new IgniteContext[Int, String](sc, () =>
new IgniteConfiguration()).fromCache("CompanyCache")

// Create company DataFrame

val dfCompany = sqlContext.createDataFrame(companyCacheIgnite.map(p =>
Company(p._1, p._2)))

// Register DataFrame as a table

dfCompany.registerTempTable("company")
```
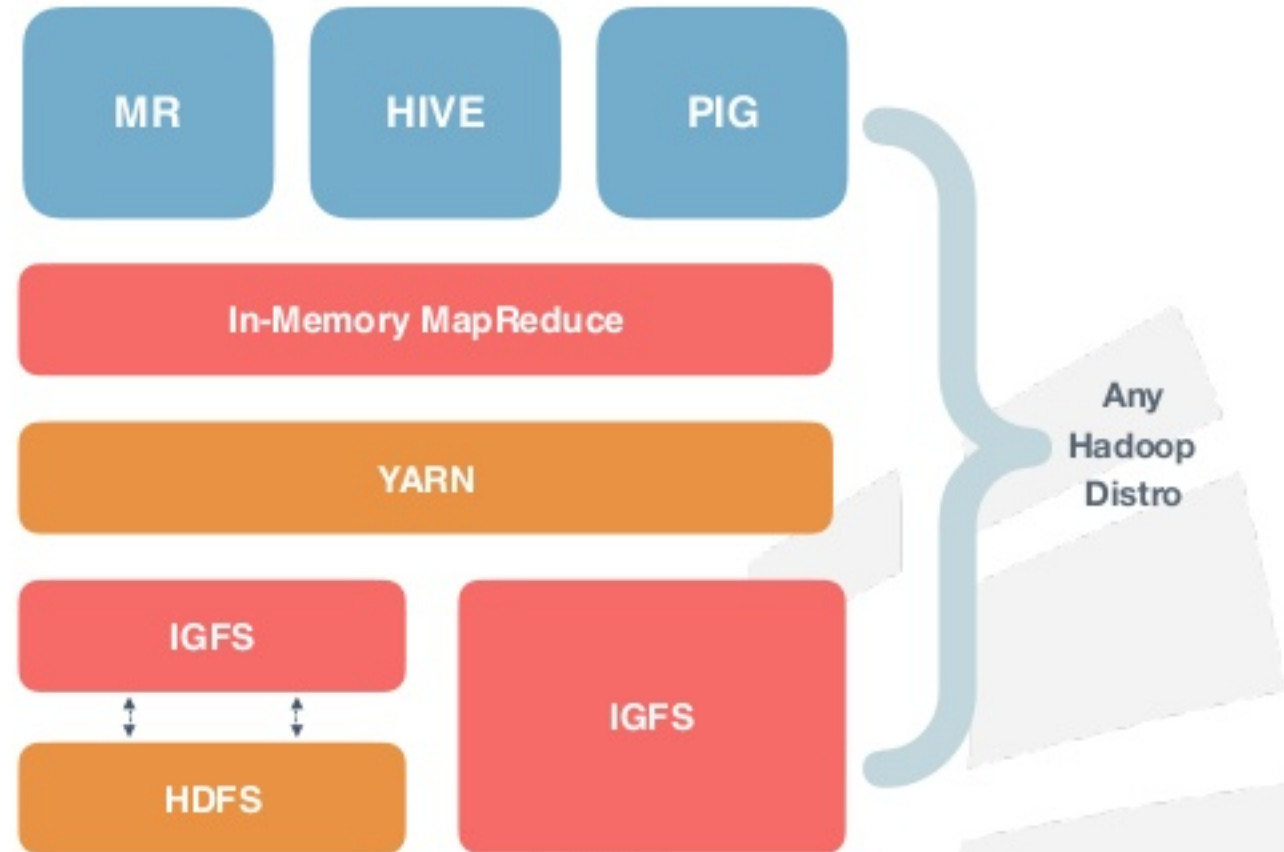
Ignite   GridGain

IGFS: 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

MR	HIVE	PIG

In-Memory MapReduce

YARN

IGFS

IGFS

HDFS

Any Hadoop Distro

Hadoop Accelerator: Map Reduce

- In-Memory Performance
- Zero Code Change
  - Use existing MR code
  - Use existing Hive queries
- No Name Node
- No Network Noise
- In-Process Data Colocation
- Eager Push Scheduling
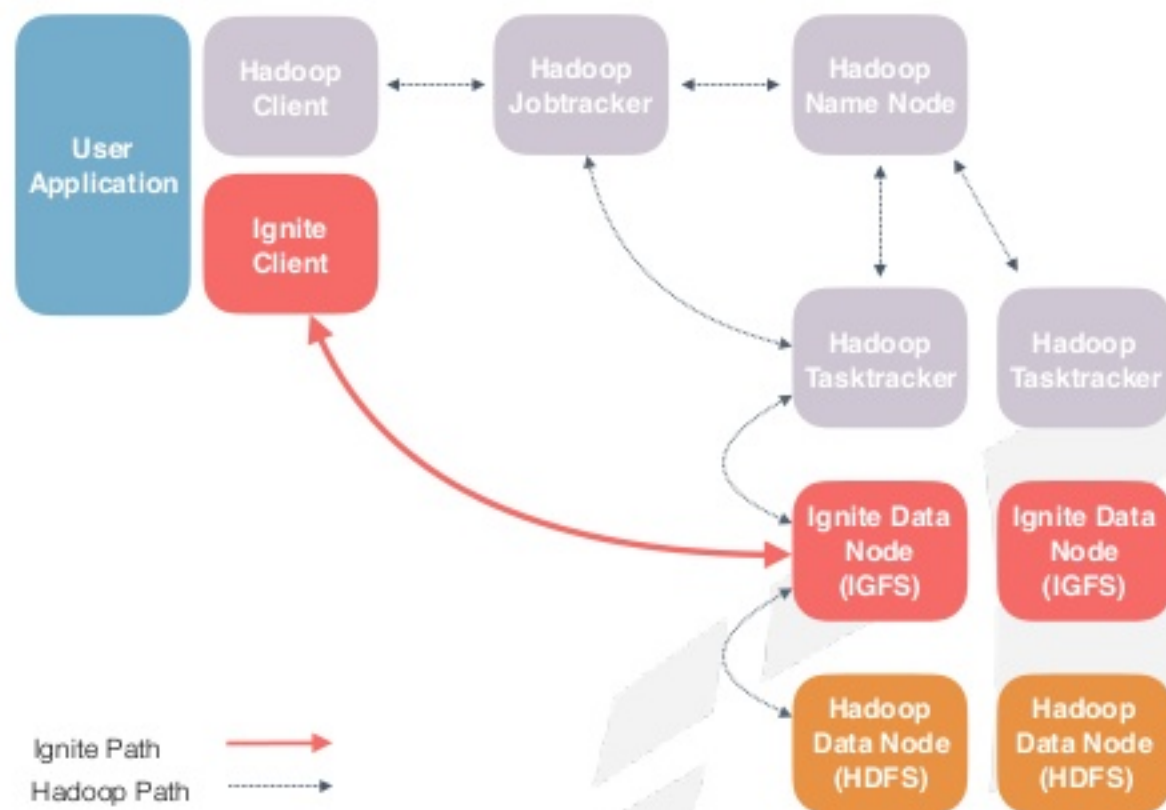
Deployment

- Docker
- Amazon AWS
- Azure Marketplace
- Google Cloud
- Apache JClouds
- Mesos
- YARN
- Apache Karaf (OSGi)

Thank You!

Author: Christos Erotocritou

github.com/kemiz/SparkIgniteSimpleExample

Thank you for joining us. Follow the conversation.

www.gridgain.com

@gridgain @ApacheIgnite
#gridgain #ApacheIgnite