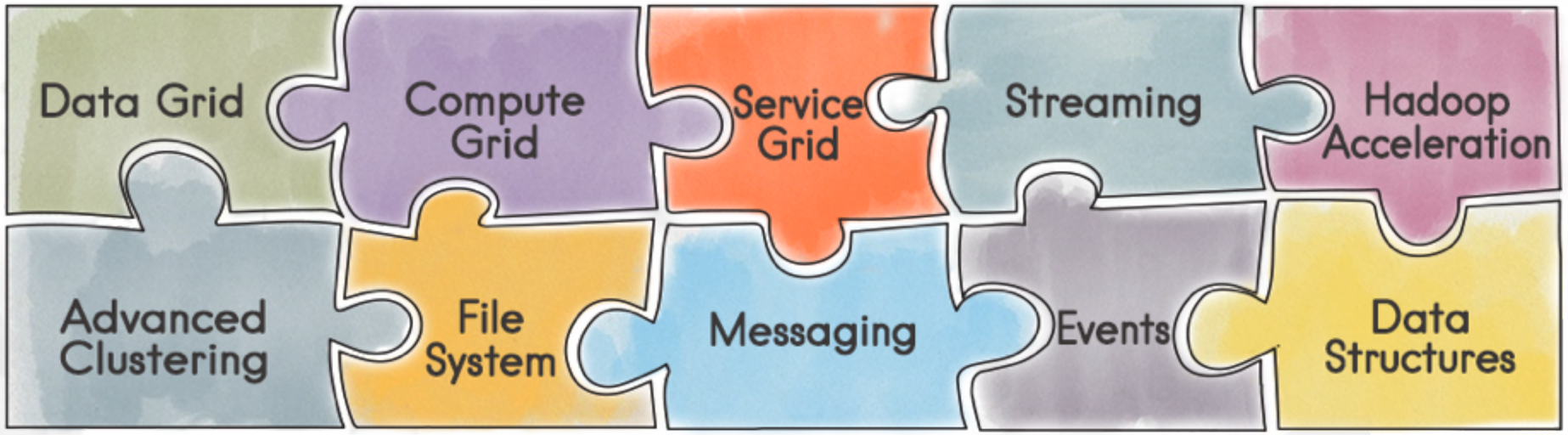
Introduction

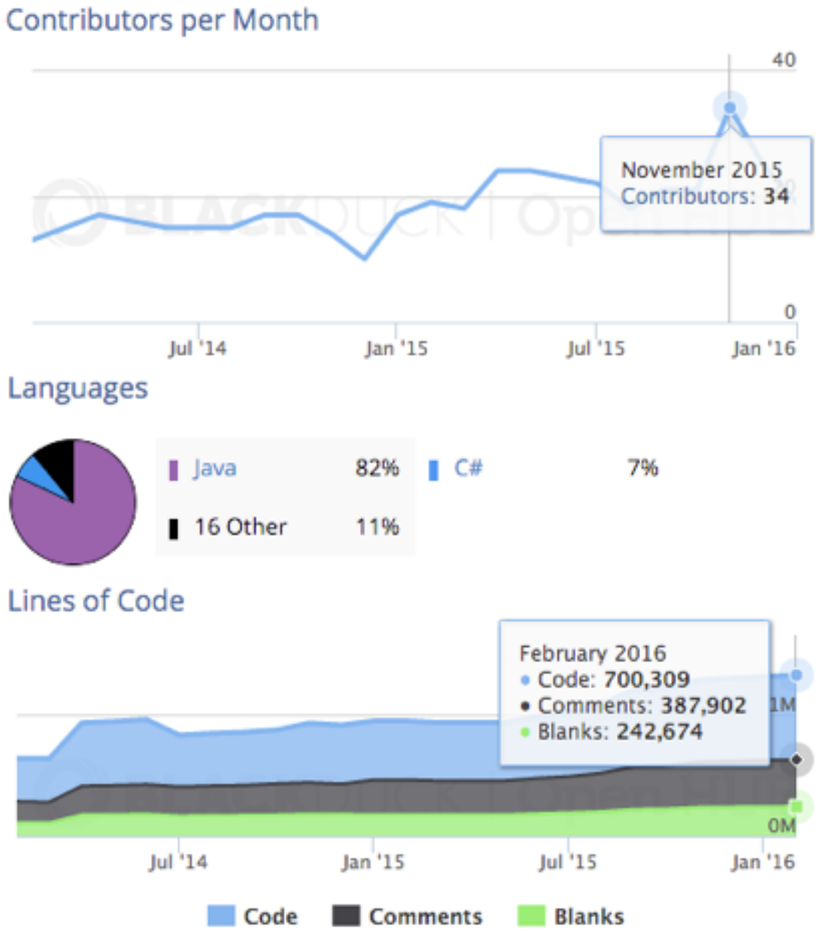


Agenda

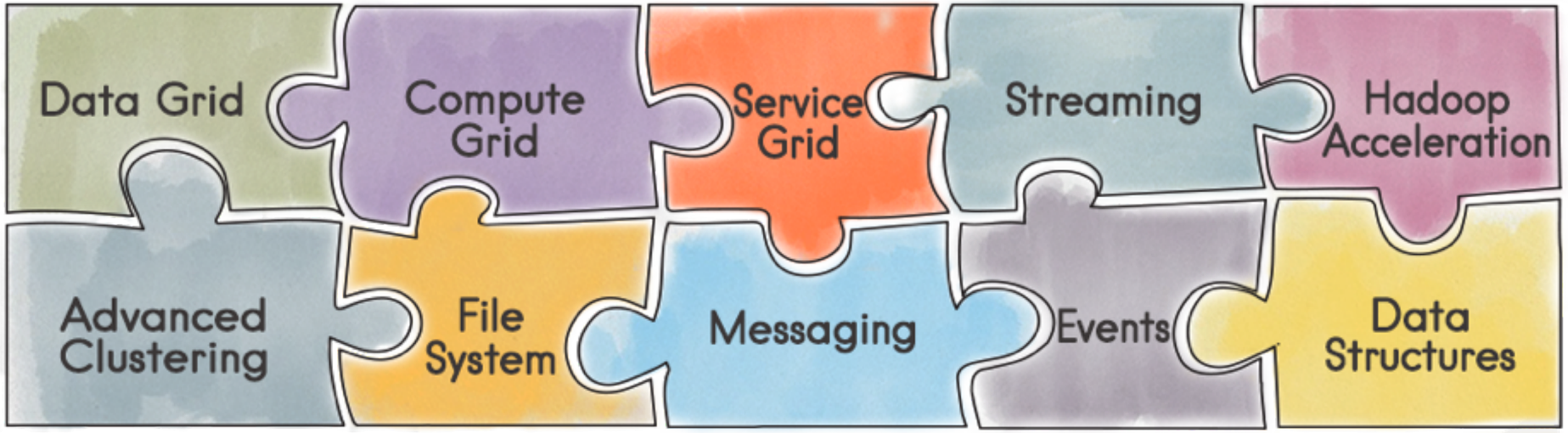
- Apache Ignite Project
- Apache Ignite Data Fabric:
  - Data Grid
  - HPC & Compute
  - Streaming & CEP
  - Hadoop & Spark Integration
- Use Cases
- Demo
- Q & A

Apache Ignite Project

- **2007:** Nikita & Dmitriy create the first version of GridGain
- **Oct. 2014:** GridGain contributes Ignite to ASF
- **Aug. 2015:** Ignite is the second fastest project to graduate after Spark
- **Today:**
  - 60+ contributors and growing rapidly
  - Huge development momentum - Estimated 192 years of effort since the first commit in February, 2014 [Openhub]
  - Mature codebase: 700k+ SLOC & more than 16k commits

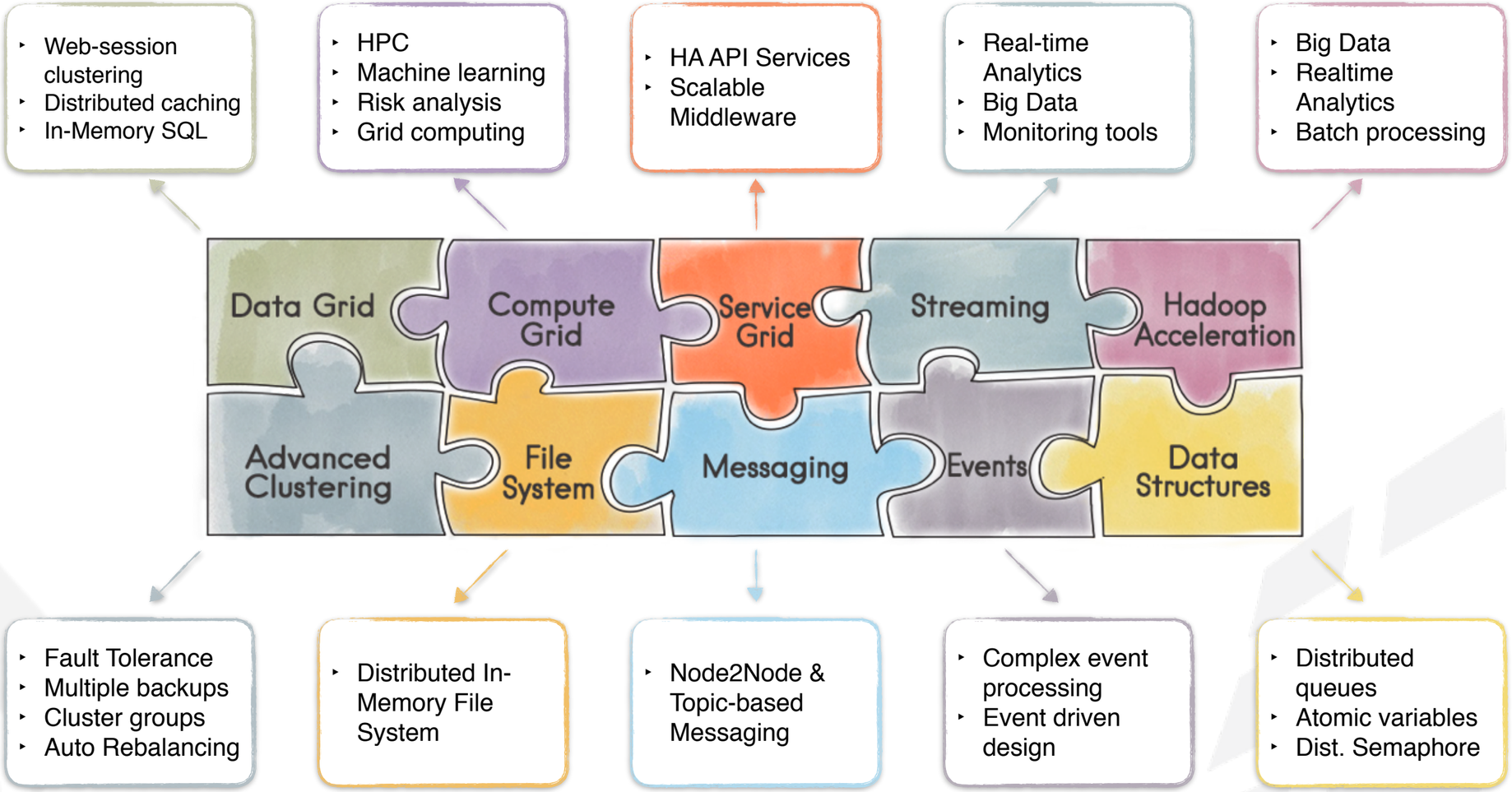


What is Apache Ignite?



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

What is Apache Ignite?



Customer Use Cases

Data Velocity, Data Volume, Real-Time Performance

Automated Trading Systems

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

Online Gaming

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

Financial Services

Fraud Detection, Risk Analysis, Insurance rating and modeling.

SaaS Platforms & Apps

High performance next-generation architectures for Software as a Service Application vendors.

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.

citi

markit

SBERBANK

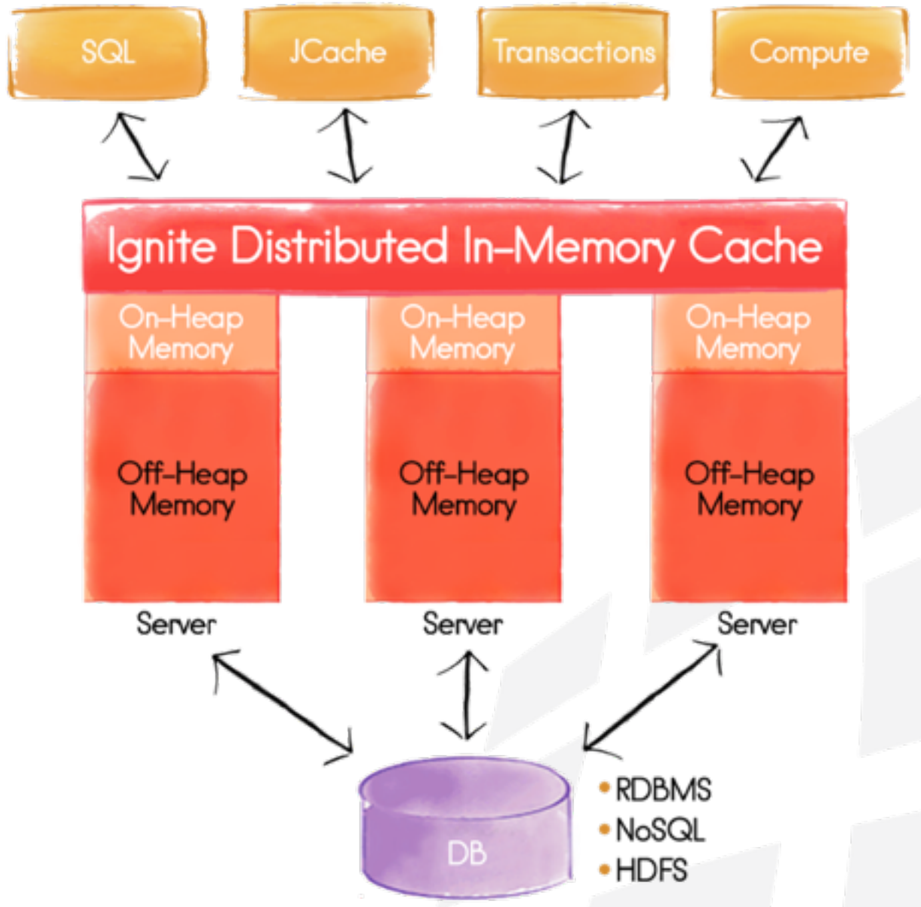
CHRONOTRACK SYSTEMS

SONY

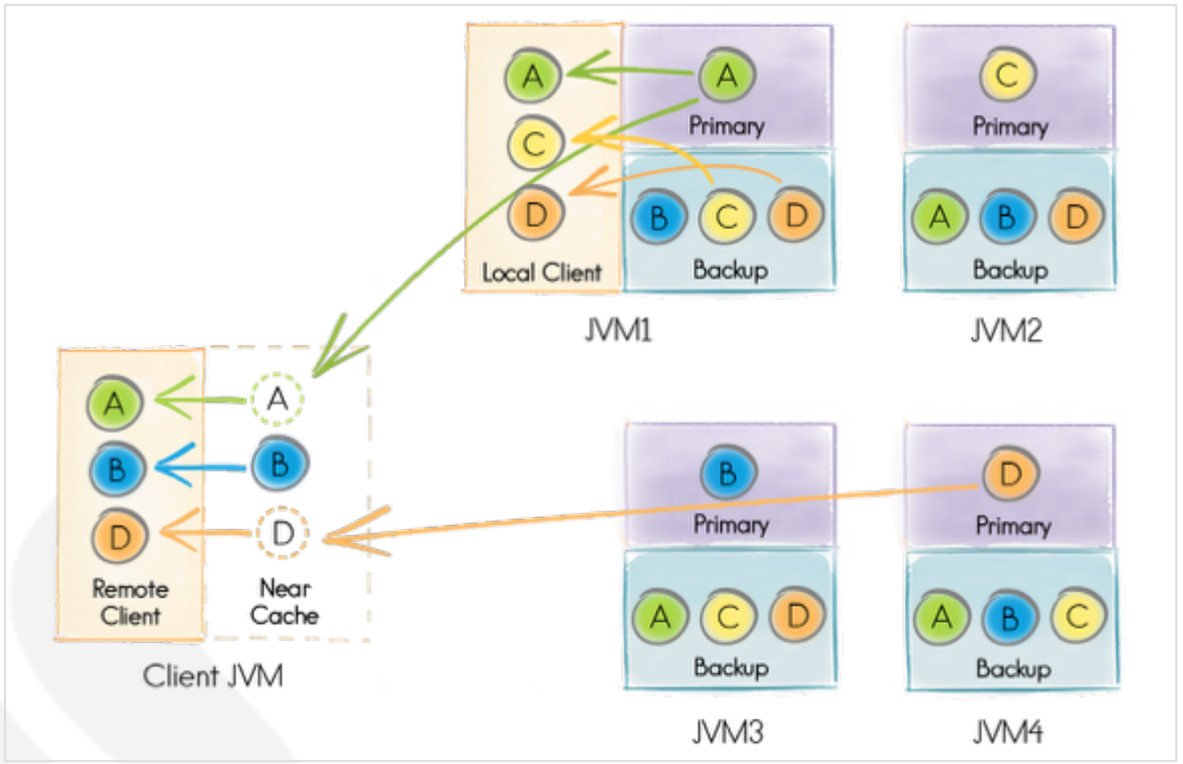
THOMSON REUTERS

In-Memory Data Grid

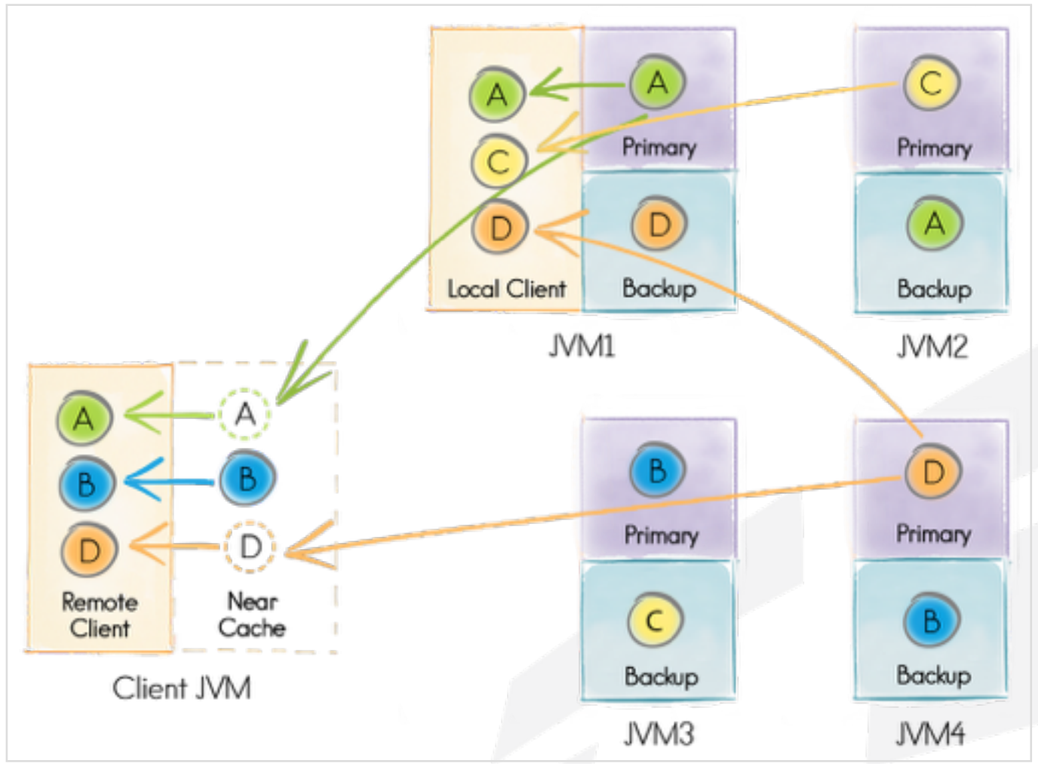
- 100% JCache Compliant (JSR 107)
  - Basic Cache Operations
  - Concurrent Map APIs
  - Collocated Processing (EntryProcessor)
  - Events and Metrics
  - Pluggable Persistence
- Ignite Data Grid
  - Fault Tolerance and Scalability
  - Distributed Key-Value Store (Cache queries)
  - SQL Queries (ANSI 99)
  - ACID Transactions
  - In-Memory Indexes
  - RDBMS / NoSQL Integration



Data Grid: Fault Tolerance & Scalability



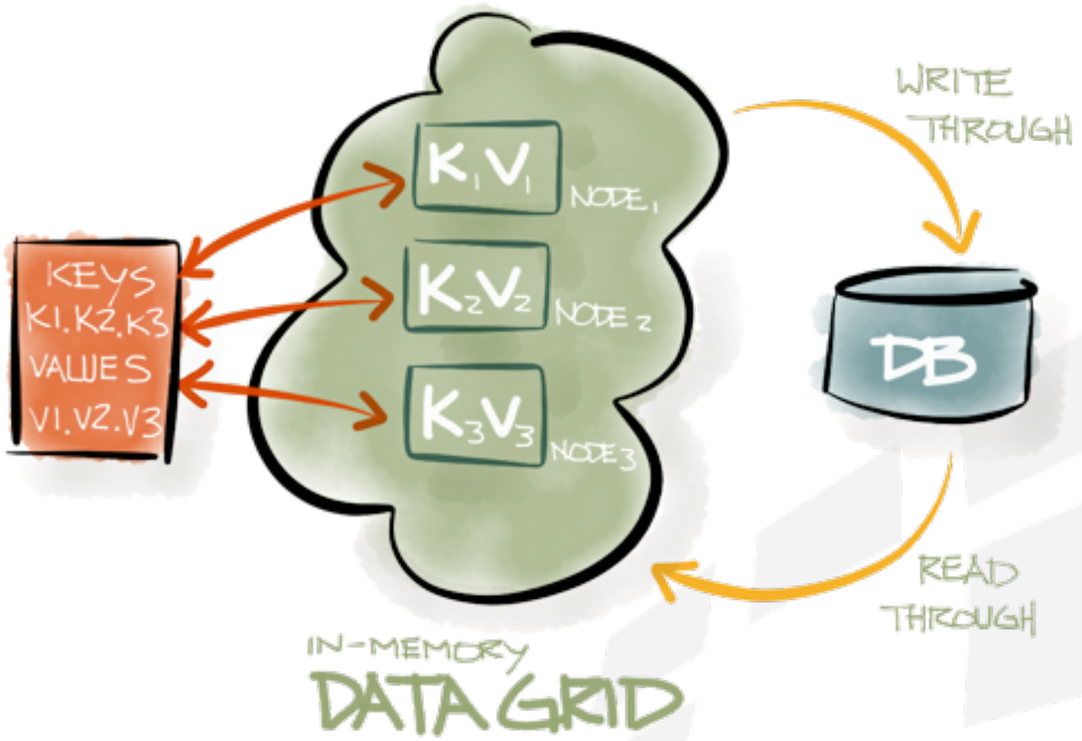
Replicated Cache



Partitioned Cache

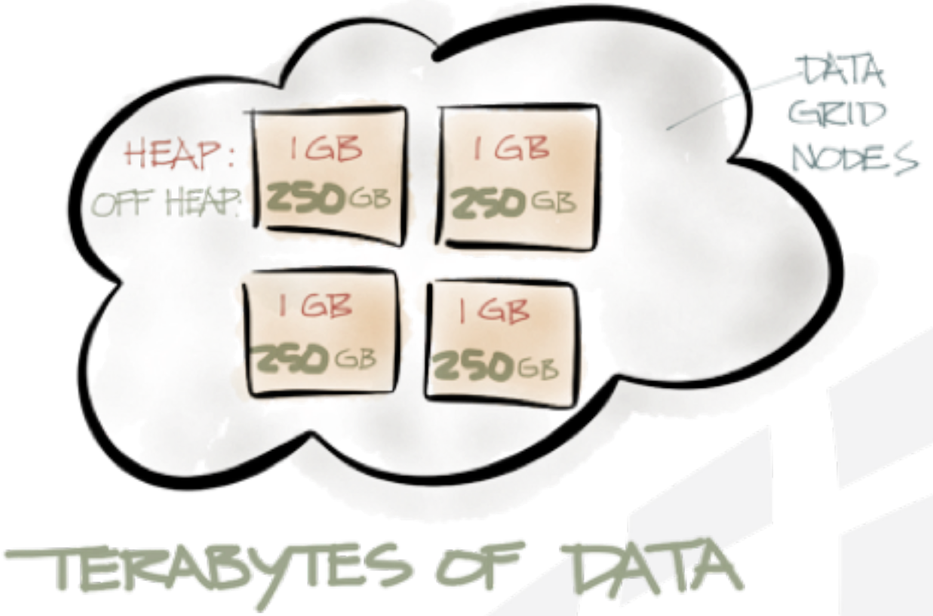
Data Grid: External Persistence

- Read-through & Write-through
- Support for Write-behind
- Configurable eviction policies
- DB schema mapping wizard:
  - Generates all the XML configuration and Java POJOs



Data Grid: Off-Heap Memory

- Unlimited Vertical Scale
- Avoid Java Garbage Collection Pauses
- Small On-Heap Footprint
- Configurable eviction policies
- Off-Heap Indexes
- Full RAM Utilization
- Simple Configuration



Data Grid: Cache APIs & Queries

- Predicate-based Scan Queries
- Text Queries based on Lucene indexing
- Query configuration using annotations, Spring XML or simple Java code
- SQL Queries
- Memcached (PHP, Java, Python, Ruby)
- HTTP REST API
- JDBC

```
IgniteCache<Long, Person> cache = ignite.cache("mycache");

// Find only persons earning more than 1,000.
try (QueryCursor cursor = cache.query(new ScanQuery((k, p) -> p.getSalary() > 1000)) {
  for (Person p : cursor)
    System.out.println(p.toString());
}
```

```
// Query for all people with "Master Degree" in their resumes.
TextQuery txt = new TextQuery(Person.class, "Master Degree");

try (QueryCursor<Entry<Long, Person>> masters = cache.query(txt)) {
  for (Entry<Long, Person> e : cursor)
    System.out.println(e.getValue().toString());
}
```

```
/** Person ID (indexed). */
@QuerySqlField(index = true)
private long id;
```

```
// Listing indexes.
Collection<QueryIndex> indexes = new ArrayList<>(3);

indexes.add(new QueryIndex("id"));
indexes.add(new QueryIndex("orgId"));
indexes.add(new QueryIndex("salary"));

queryEntity.setIndexes(indexes);
```

Data Grid: SQL Support (ANSI 99)

- ANSI-99 SQL
- In-Memory Indexes (On and Off-Heap)
- Automatic Group By, Aggregations, Sorting
- Cross-Cache Joins, Unions
- Use local H2 engine

```
IgniteCache<Long, Person> cache = ignite.cache("mycache");

// SQL join on Person and Organization.
SqlQuery sql = new SqlQuery(Person.class,
  "from Person, Organization "
  + "where Person.orgId = Organization.id "
  + "and lower(Organization.name) = lower(?)");

// Find all persons working for Ignite organization.
try (QueryCursor<Entry<Long, Person>> cursor = cache.query
(sql.setArgs("Ignite"))) {
  for (Entry<Long, Person> e : cursor)
    System.out.println(e.getValue().toString());
}
```

Data Grid: Transactions

- Fully ACID
- Support for **Transactional** & **Atomic**
- Cross-cache transactions
- **Optimistic** and **Pessimistic** concurrency modes with multiple isolation levels
- Deadlock protection
- JTA Integration

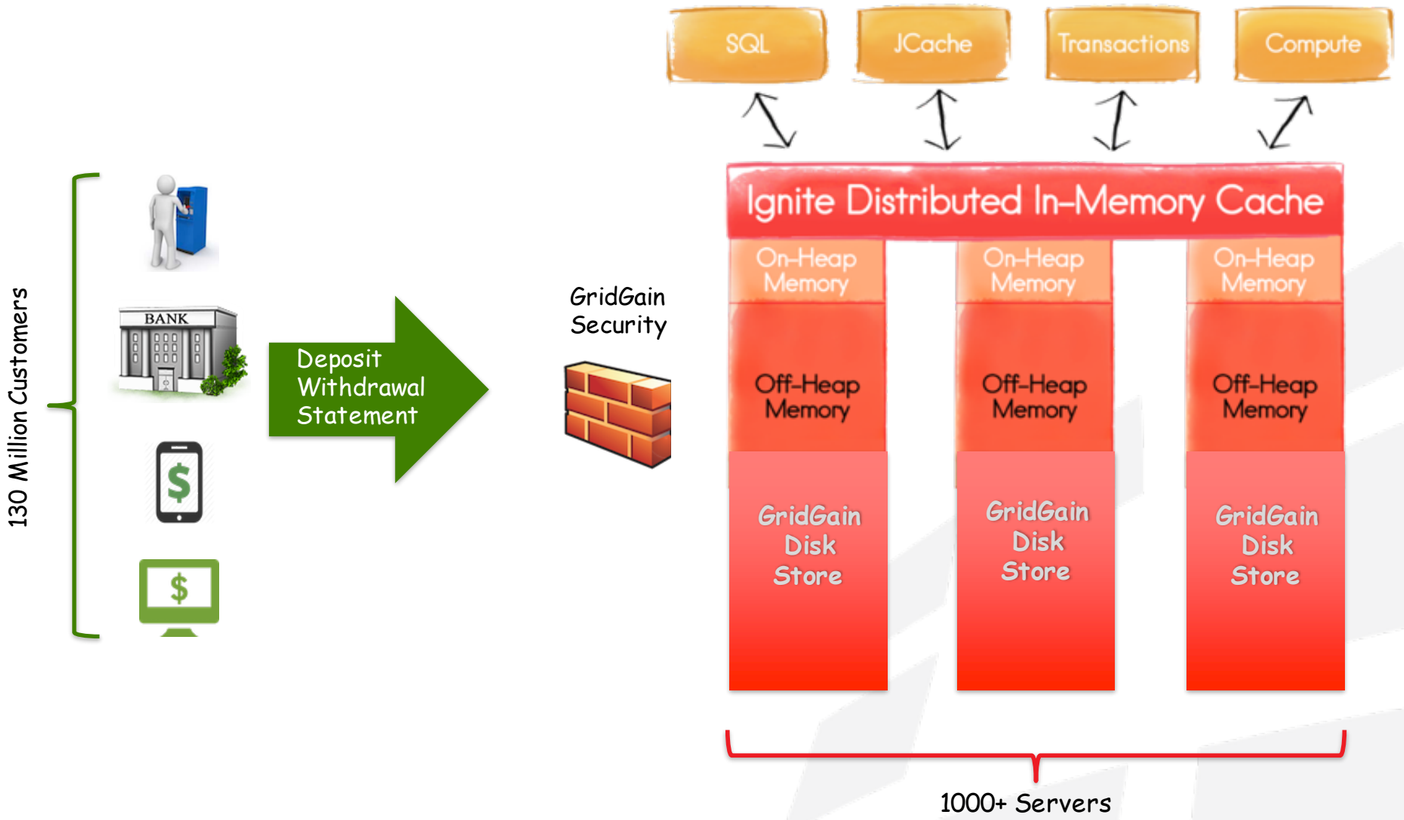
```
try (Transaction tx = transactions.txStart()) {
    Integer hello = cache.get("Hello");

    if (hello == 1)
        cache.put("Hello", 11);

    cache.put("World", 22);

    tx.commit();
}
```

```
IgniteTransactions txs = ignite.transactions();

// Start transaction in optimistic mode with repeatable read is
olation level.
Transaction tx = txs.txStart(TransactionConcurrency.OPTIMISTIC,
TransactionIsolation.REPEATABLE_READ);
```

Use Case: SBERBANK

Largest bank in Russia & Eastern Europe & third largest in Europe

- Sberbank Requirements:
  - Migrate to data grid architecture
  - Minimize dependency on Oracle
  - Move to open source
- Why GridGain Won:
  - Best performance
    - 10+ competitors evaluated
  - Demonstrated best:
    - Fault tolerance & scalability
    - ANSI-99 SQL Support
    - Transactional consistency
  - Strict SLAs
    - Less then 5 min cluster restart



Data Grid: Continuous Queries

- Execute a query and get notified on data changes captured in the filter
- Remote filter to evaluate event and local listener to receive notification
- Guarantees exactly once delivery of an event

```
IgniteCache<Integer, String> cache = ignite.cache("mycache");

// Create new continuous query.
ContinuousQuery<Integer, String> qry = new ContinuousQuery<>();

// Optional initial query to select all keys greater than 10.
qry.setInitialQuery(new ScanQuery<Integer, String>((k, v) -> k > 10));

// Callback that is called locally when update notifications are received.
qry.setLocalListener((evts) ->
  evts.stream().forEach(e -> System.out.println("key=" + e.getKey() + ", val="
+ e.getValue())));

// This filter will be evaluated remotely on all nodes.
// Entry that pass this filter will be sent to the caller.
qry.setRemoteFilter(e -> e.getKey() > 10);

// Execute query.
try (QueryCursor<Cache.Entry<Integer, String>> cur = cache.query(qry)) {
  // Iterate through existing data stored in cache.
  for (Cache.Entry<Integer, String> e : cur)
    System.out.println("key=" + e.getKey() + ", val=" + e.getValue());

  // Add a few more keys and watch a few more query notifications.
  for (int i = 5; i < 15; i++)
    cache.put(i, Integer.toString(i));
}
```

Distributed Java Structures

- Distributed Map (cache)
- Distributed Set
- Distributed Queue
- CountDownLatch
- AtomicLong
- AtomicSequence
- AtomicReference
- Distributed ExecutorService

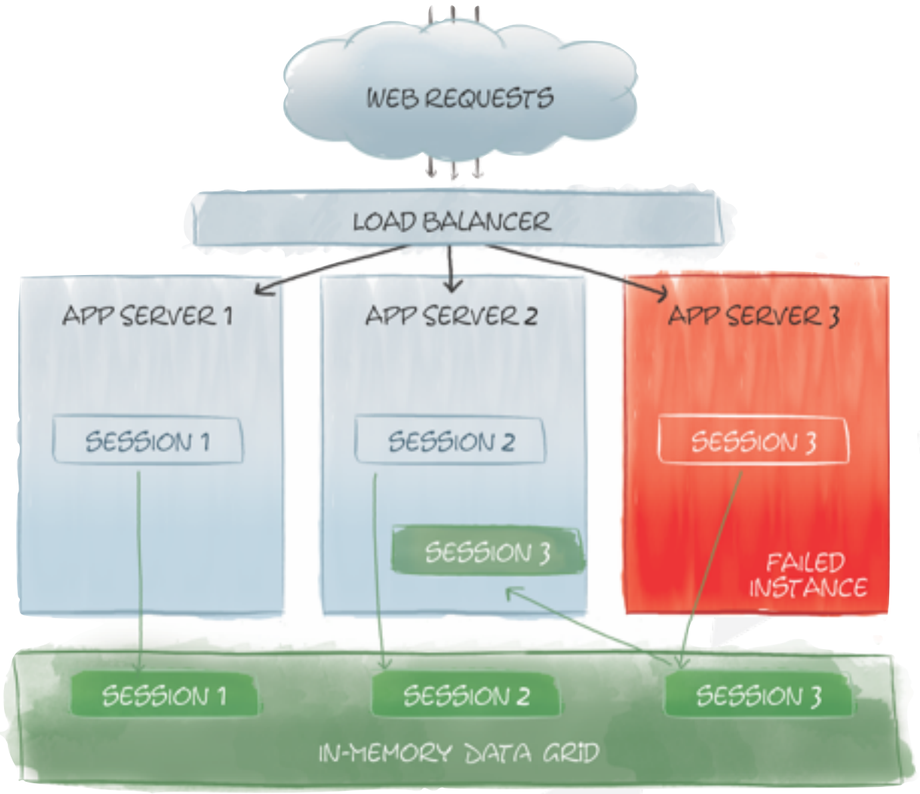
```
Ignite ignite = Ignition.ignite();

CollectionConfiguration colCfg = new Colle
ctionConfiguration();

colCfg.setCollocated(true);

// Create collocated queue.
IgniteQueue<String> queue = ignite.queue
("queueName", 0, colCfg);
```

```
// Initialize atomic long.
final IgniteAtomicLong atomicLong = ignite.atomic
Long("atomicName", 0, true);

// Increment atomic long on local node.
System.out.println("Incremented value: " + atomic
Long.incrementAndGet());
```

```
IgniteAtomicLong atomicLong = ignite.atomicLong(
    "atomicName", // Atomic long name.
    0,            // Initial value.
    false         // Create if it does not exist.
)
```

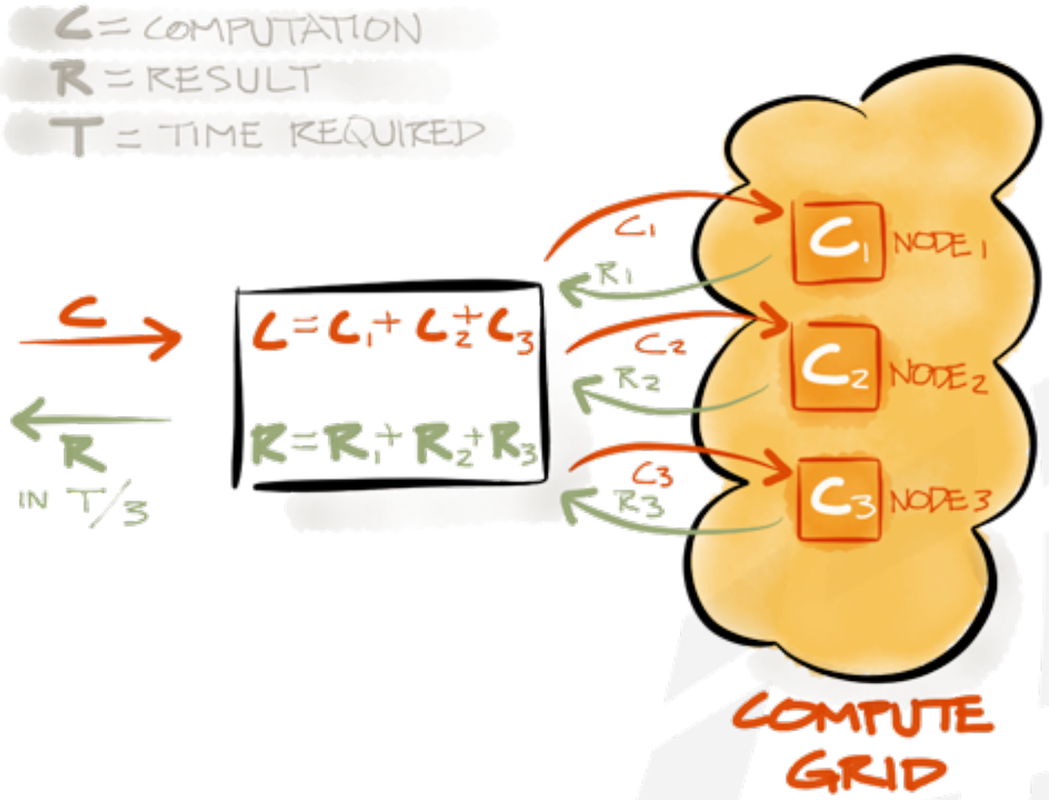
Data Grid: Web Session Clustering

- No need for sticky sessions
- Shared session between app servers
- Fault tolerance
- Scalability



In-Memory Compute Grid

- Direct API for MapReduce
- Cron-like Task Scheduling
- State Checkpoints
- Load Balancing
- Automatic Failover
- Zero Deployment
  - Distributed class loading



In-Memory Compute Grid

- Distributed Closures
  - Java lambda expressions (JSR 335)
- Distributed ExecutorService
- Sync or Async
- Task Deployment (GAR)

```
// Execute closure on all cluster nodes.
Collection<Integer> res = compute.apply(
    String::length,
    Arrays.asList("How many characters".split(" "))
);

// Add all the word lengths received from cluster nodes.
int total = res.stream().mapToInt(Integer::intValue).sum();
```

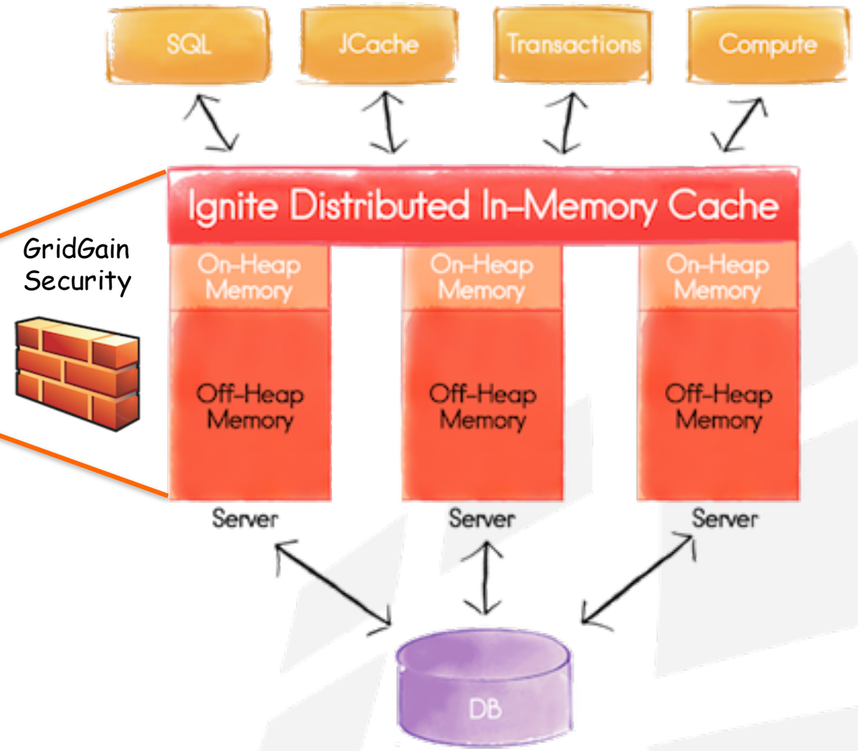
```
// Limit broadcast to remote nodes only.
IgniteCompute compute = ignite.compute(ignite.cluster
().forRemotes());

// Print out hello message on remote nodes in the clust
er group.
compute.broadcast(() -> System.out.println("Hello Node:
" + ignite.cluster().localNode().id()));
```

```
// Iterate through all words and print
// each word on a different cluster node.
for (String word : "Print words on different cluster no
des".split(" "))
    // Run on some cluster node.
    compute.run(() -> System.out.println(word));
```

Use Case: SilverSpring NETWORKS

Smart metering & utilities - comprehensive IoT platform

- SilverSpring Requirements:
  - Migrate to in-memory processing
  - Add scalability & elasticity
  - Move to open source
- Why GridGain Won:
  - Strong compute
    - Colocated compute & data
  - Demonstrated best:
    - On-demand elasticity
    - ANSI-99 SQL Support
    - Transactional consistency

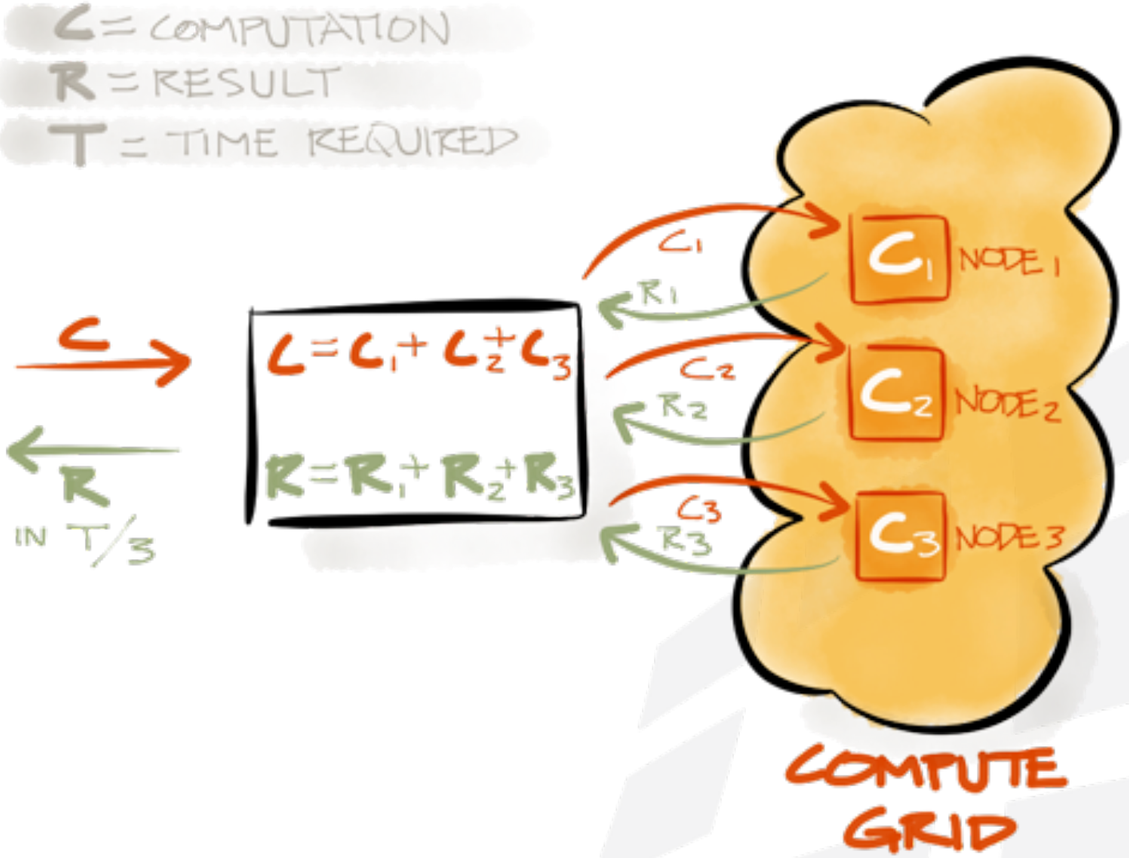


Use Case: e-Therapeutics plc

systems biology drug discovery

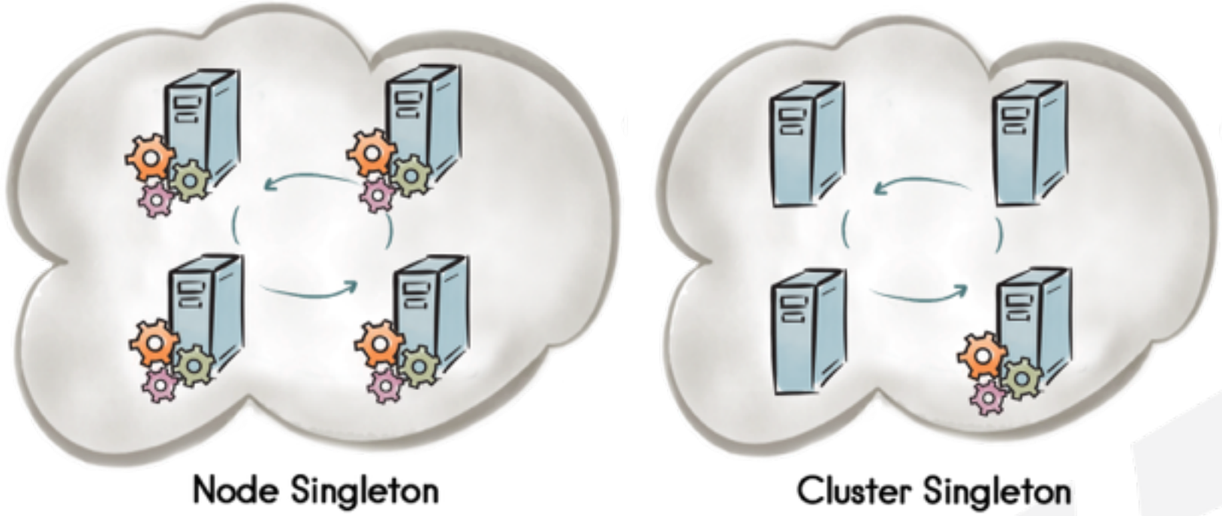
Finds new medicines for diseases that are still poorly treated

- SilverSpring Requirements:
  - Run in-memory computations
  - Add scalability
  - Agile discovery platform
- Why GridGain Won:
  - Strong compute
    - Colocated compute & data
  - Demonstrated best:
    - Fault tolerance & scalability



In-Memory Service Grid

- Singletons on the Cluster
  - Cluster Singleton
  - Node Singleton
  - Key Singleton
- Guaranteed Availability
  - Auto Redeployment in Case of Failures




```
svcs.deployNodeSingleton("myNodeSingleton", new MyService());
```

```
svcs.deployClusterSingleton("myClusterSingleton", new MyService());
```

```
svcs.deployKeyAffinitySingleton("myKeySingleton", new MyService(), "myCache", new MyCacheKey());
```

Messaging & Events

- Topic-based messaging
- Ordered & Unordered messages
- Local & Remote message listeners
- Local & Remote event listeners
  - Trigger actions from any cluster events or operations
- Query events via IgniteEvents API

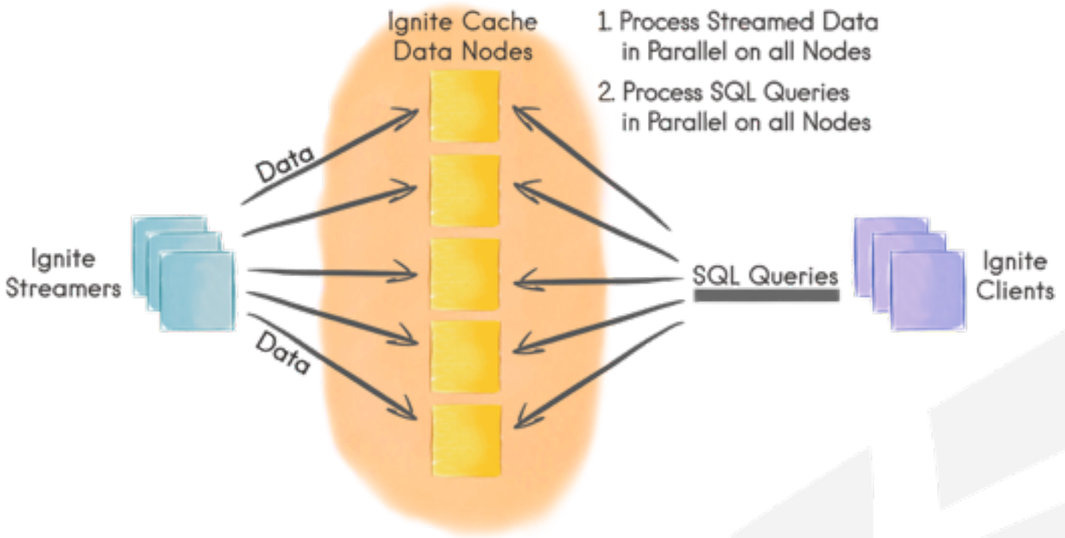
```
// Subscribe to specified cache events
occuring on local node.
ignite.events().localListen(locLsnr,
  EventType.EVT_CACHE_OBJECT_PUT,
  EventType.EVT_CACHE_OBJECT_READ,
  EventType.EVT_CACHE_OBJECT_REMOVED);
```

```
// Subscribe to specified cache events
on all nodes that have cache running.
ignite.events(ignite.cluster().forCache
Nodes("cacheName")).remoteListen(null,
rmtLsnr,
EventType.EVT_CACHE_OBJECT_PUT,
  EventType.EVT_CACHE_OBJECT_READ,
  EventType.EVT_CACHE_OBJECT_REMOVED);

// Generate cache events.
for (int i = 0; i < 20; i++)
  cache.put(i, Integer.toString(i));
```

```
// Add listener for unordered messages on all remote nodes.
rmtMsg.remoteListen("MyOrderedTopic", (nodeId, msg) -> {
    System.out.println("Received ordered message [msg=" + m
sg + ", from=" + nodeId + ']');

    return true; // Return true to continue listening.
});
```

In-Memory Streaming and CEP

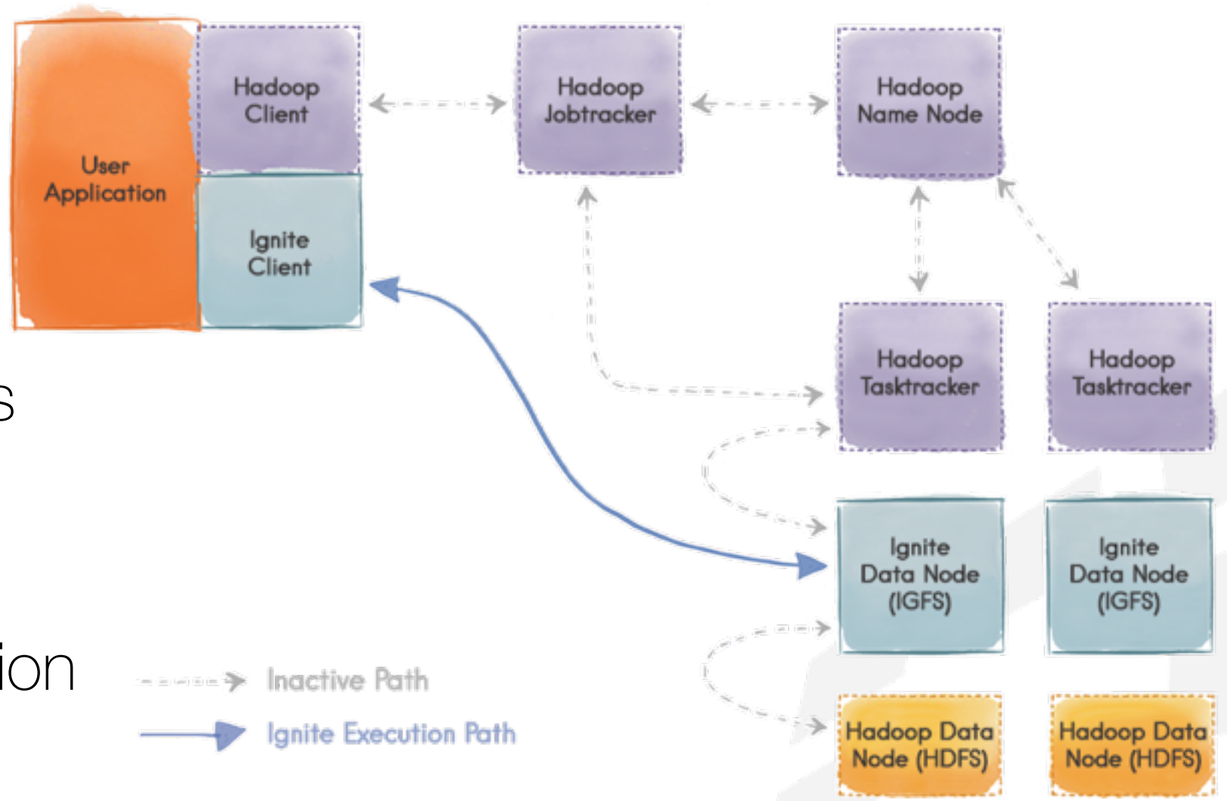
- Branching Pipelines
- Pluggable Routing
- Sliding Windows for CEP/ Continuous Query
- Real Time Analysis




```
CacheConfiguration<Integer, Long> cfg = new CacheConfiguration<>("myStreamCache");

// FIFO window holding 1,000,000 entries.
cfg.setEvictionPolicyFactory(new FifoEvictionPolicy(1_000_000));
```

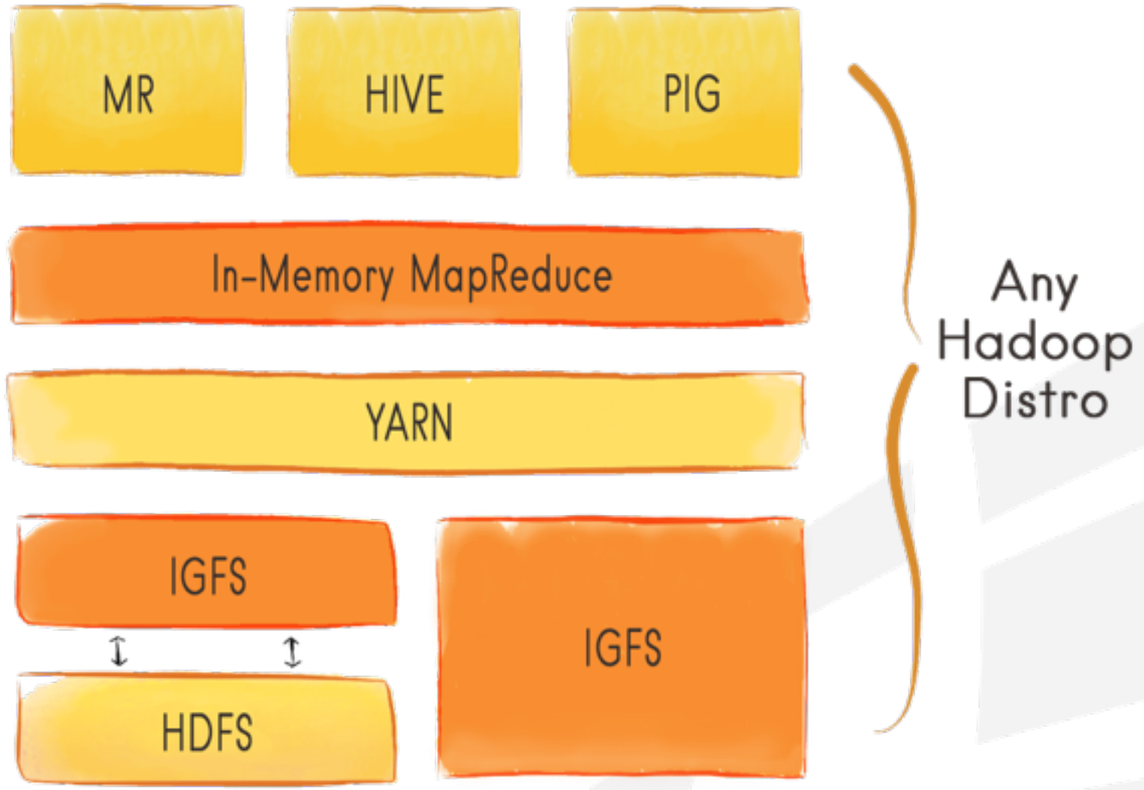
Hadoop Accelerator: Map Reduce

- In-Memory Native 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



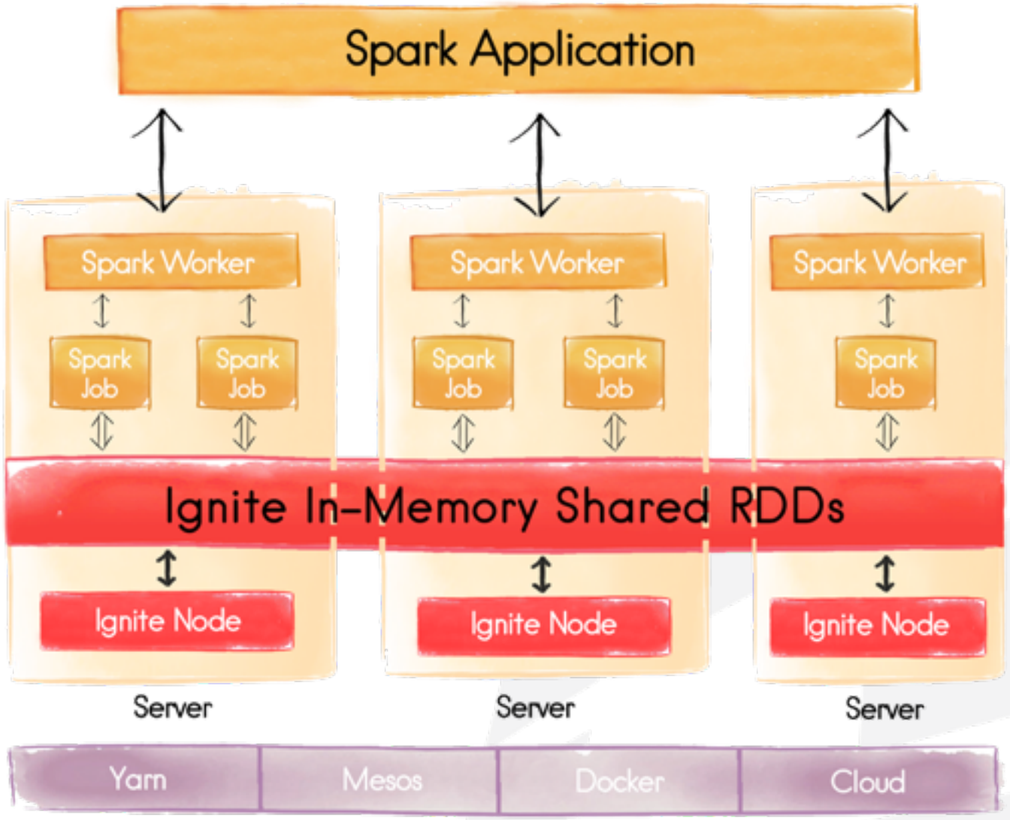
IGFS: Ignite In-Memory File System

- 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



Spark Integration: Shared RDDs & Improved SQL

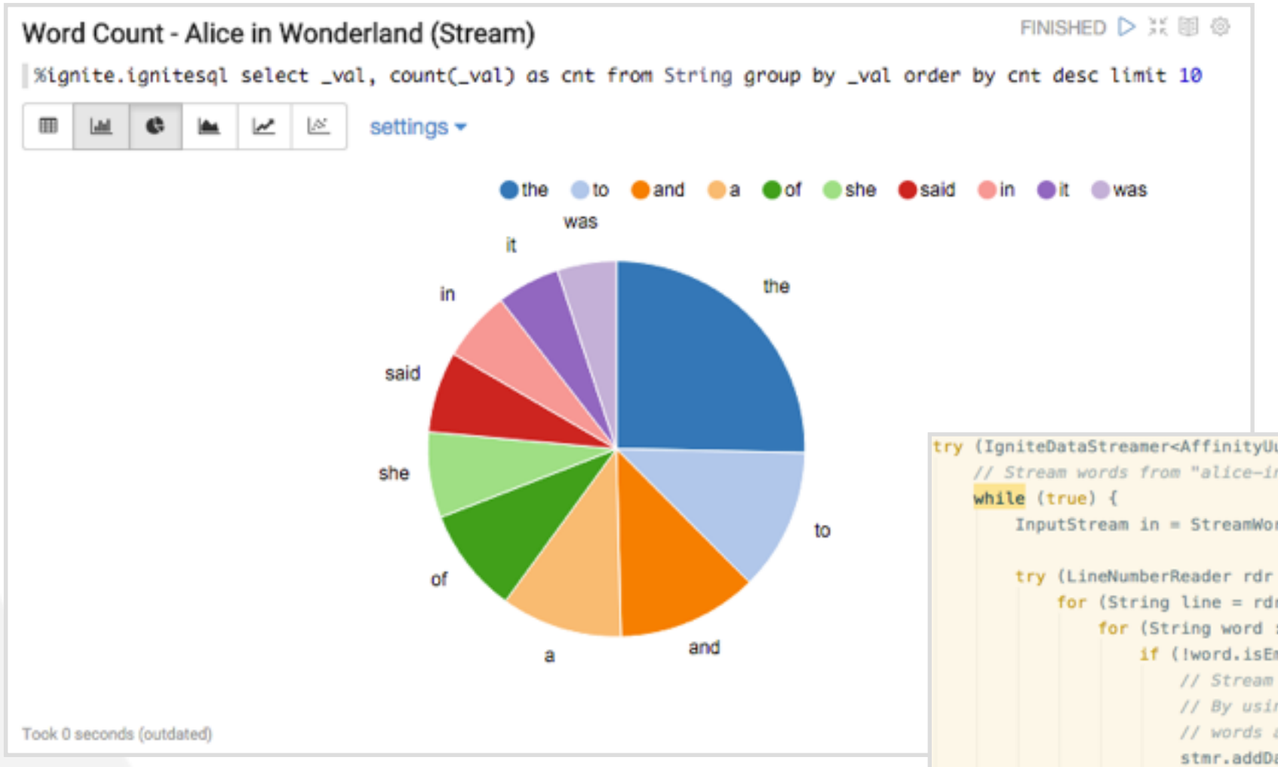
- IgniteRDD
  - Share RDD across jobs on the host
  - Share RDD across jobs in the application
  - Share RDD globally
- Faster SQL
  - In-Memory Indexes
  - SQL on top of Shared RDD



Deployment

- Docker
- Amazon AWS
- Google Cloud
- Apache JClouds
- Mesos
- YARN
- Apache Karafe (OSGi)

In-Memory Streaming Demo

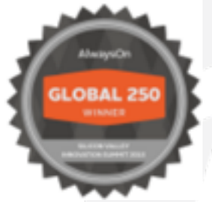


- Apache Zeppelin

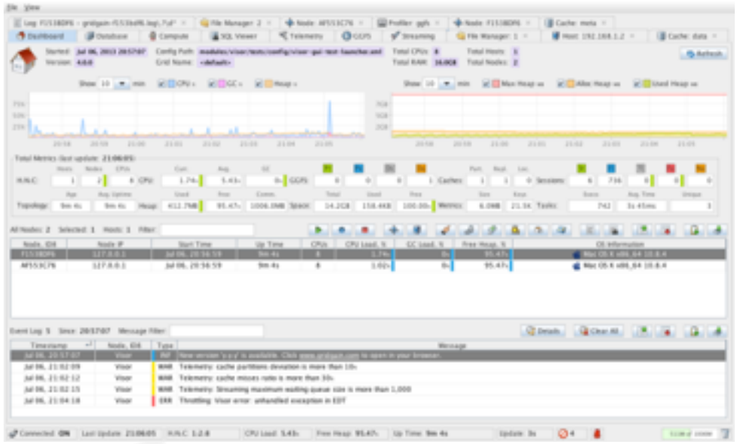
```
try (IgniteDataStreamer<AffinityUuid, String> stmr = ignite.dataStreamer(stmCache.getName())) {
    // Stream words from "alice-in-wonderland" book.
    while (true) {
        InputStream in = StreamWords.class.getResourceAsStream("alice-in-wonderland.txt");

        try (LineNumberReader rdr = new LineNumberReader(new InputStreamReader(in))) {
            for (String line = rdr.readLine(); line != null; line = rdr.readLine()) {
                for (String word : line.split(" "))
                    if (!word.isEmpty())
                        // Stream words into Ignite.
                        // By using AffinityUuid we ensure that identical
                        // words are processed on the same cluster node.
                        stmr.addData(new AffinityUuid(word), word);
            }
        }
    }
}
```

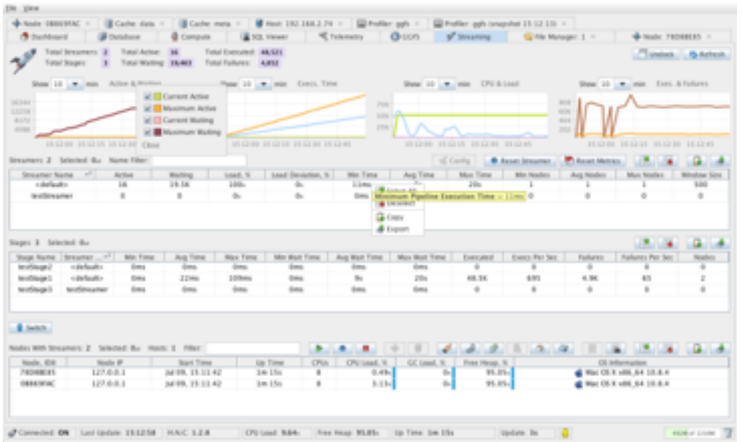
GridGain

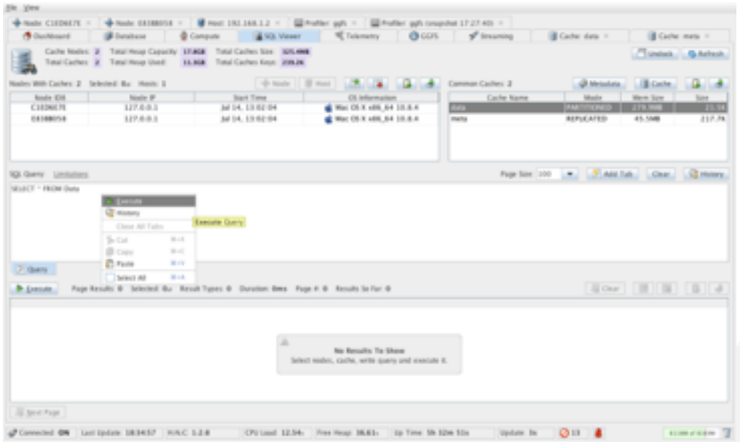
- GridGain Enterprise Edition
  - Is a binary build of Apache Ignite™ created by GridGain, which includes optional LGPL dependencies, such as Hibernate L2 cache integration and Geospatial Indexing.
  - Added enterprise features for enterprise deployments
  - Earlier features and bug fixes by a few weeks.
  - More testing.

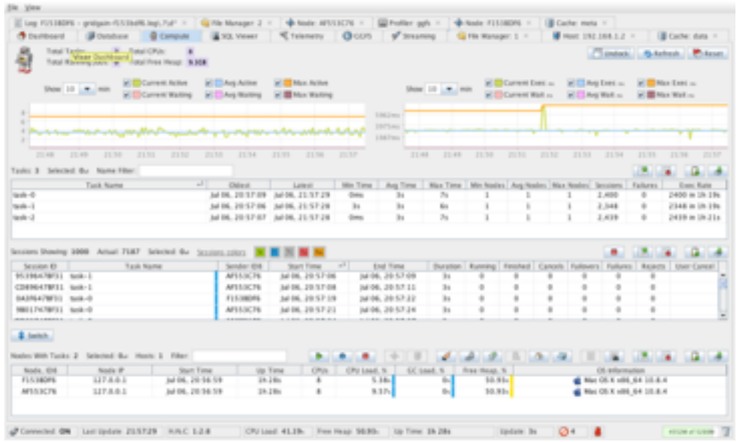


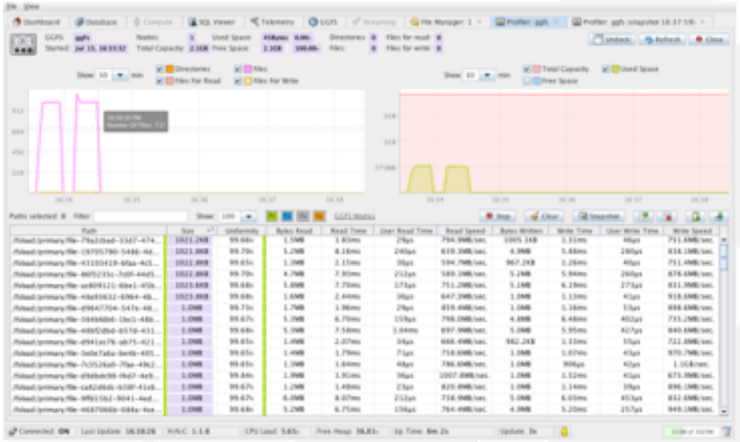
Visor: Monitoring & Mgmt for DevOps







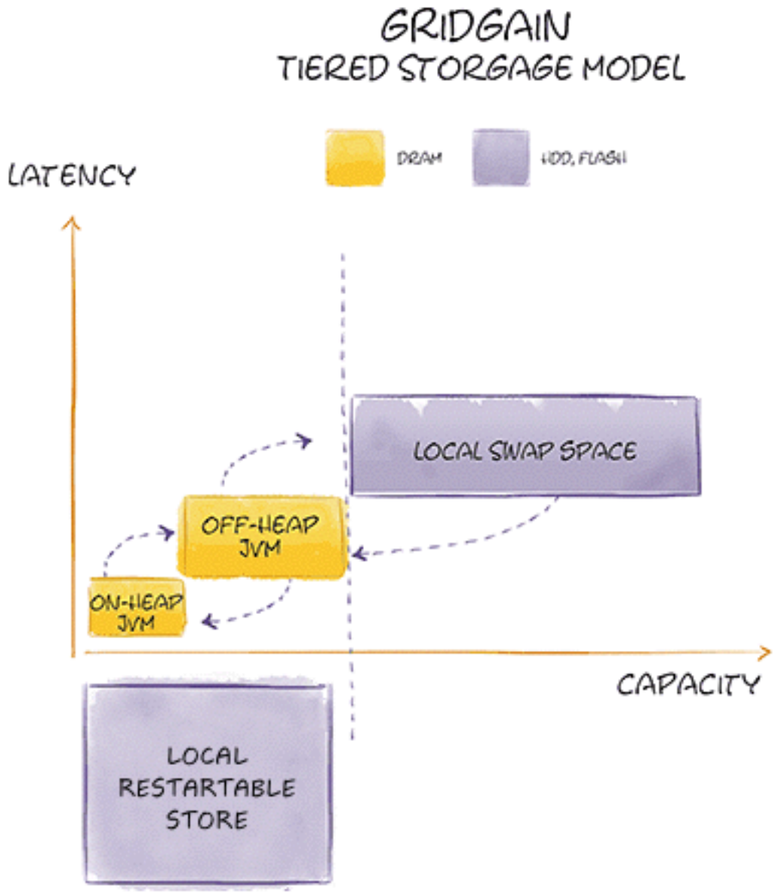




✶ *Enterprise Edition Only*

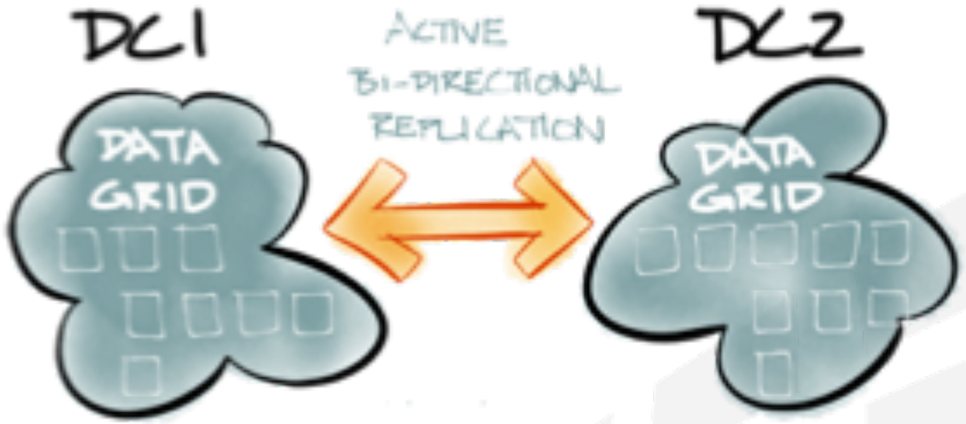
Tiered Memory & Local Store

- 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
- ✳ Enterprise Edition Only



In-Memory Data Fabric: Data Center Replication

- Multiple (up to 32) Data Centers
- Complex Replication Technologies
- Active-Active & Active-Passive
- Smart Conflict Resolution
- Durable Persistent Queues
- Automatic Throttling
- ✳ Enterprise Edition Only

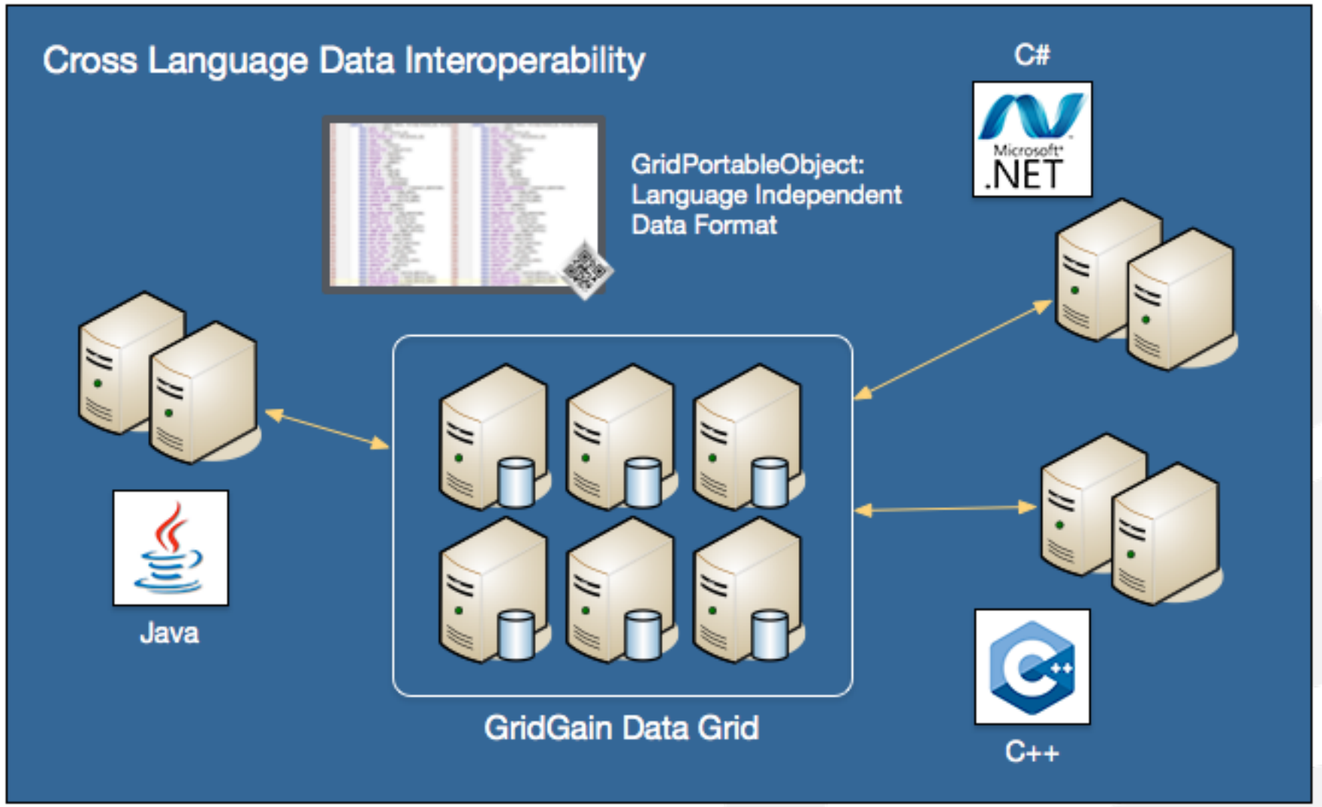


In-Memory Data Fabric: Security

- Pluggable Auth & Auth
  - JAAS, LDAP, JNDI, Kerberos
- In-Cluster Node Authentication
- Client Authentication
- Multi-Tenancy
- Fine-Grained Authorization
- Comprehensive Auditing
  - Who? What? When?
- ✳ Enterprise Edition Only

Cross-Language Interoperability

- Portable Objects
- Performance Across Languages
- Client Feature Parity
- Dynamic Schema Changes
- Searchable/Indexable
- Version Independent
- ✳ Enterprise Edition Only



GridGain's Open Core Business Model
Apache Ignite vs. GridGain Enterprise

GridGain Enterprise Subscriptions include:

- GridGain Enterprise Edition
- Bug fixes, patches, updates and upgrades
- 9x5 or 24x7 Support
- Ability to procure Training and Consulting Services from GridGain
- Confidence and protection, not provided under Open Source licensing, that only a commercial vendor can provide, such as indemnification

Features	Apache Ignite	GridGain Enterprise
In-Memory Data Grid	√	√
In-Memory Compute Grid	√	√
In-Memory Service Grid	√	√
In-Memory Streaming	√	√
In-Memory Hadoop Acceleration	√	√
Distributed In-Memory File System	√	√
Advanced Clustering	√	√
Distributed Messaging	√	√
Distributed Events	√	√
Distributed Data Structures	√	√
Portable Binary Objects	√	√
Management & Monitoring GUI		√
Enterprise-Grade Security		√
Network Segmentation Protection		√
Recoverable Local Store		√
Rolling Production Updates		√
Data Center Replication		√
Integration with Oracle GoldenGate		√
Basic Support (9×5)	√	√
Enterprise Support (9×5 and 24×7)		√
Security Updates		√
Maintenance Releases & Patches		√
	Free w/ optional Paid Support	Annual License Subscription

GridGain apache Ignite

Thank you!

Questions?


Thank you!