In-Memory Computing:

From Disk-First Architecture to Memory-First

YAKOV ZHDANOV
Director R&D

www.gridgain.com

@gridgain

© 2014 GridGain Systems, Inc.

GridGain: In-Memory Computing Leader

- More than 5 years in production
- 100s of customers & users
- Starts every 10 seconds worldwide

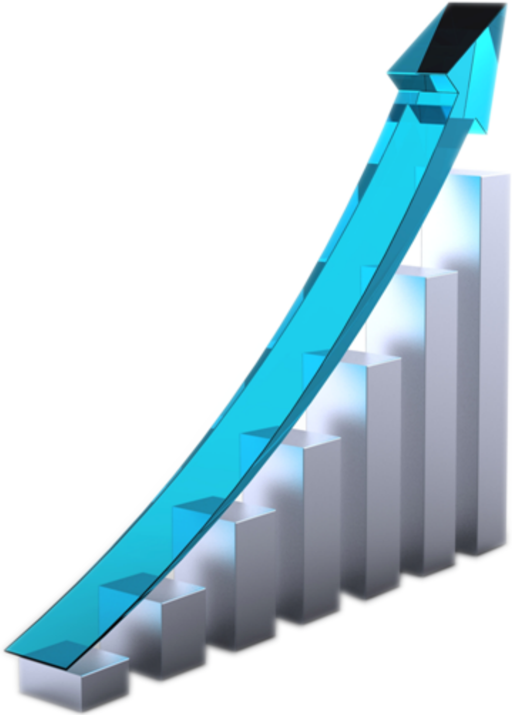
© 2014 GridGain Systems, Inc.

Agenda

- [R]Evolution of Data: Extreme Volume Growth
- In-Memory Computing: Faster Apps on Larger Data Sets
- Example System: Moving Application “In-Memory”
- GridGain: Overview And “Live Coding” Sessions
- QA

© 2014 GridGain Systems, Inc.

[R]Evolution of Data



2.5 exabytes* of data was generated every day in 2012 (IBM)

This is 62.5 km high stack of 1Tb 3.5” HDD

* 1 exabyte = 10^{18} bytes

© 2014 GridGain Systems, Inc.

Why Speeding Up Data Processing?

- 2008: Amazon found every 100ms of latency cost them 1% in sales
- Analysts demand sub-second, near real-time query results
- On-line traders want ultra-low latencies

© 2014 GridGain Systems, Inc.

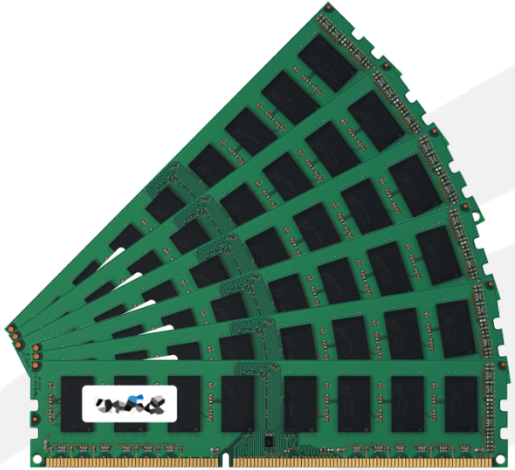
Problem

Data sets are large and complex – new tools needed.

© 2014 GridGain Systems, Inc.

Solution: In-Memory Computing

- Data is in-memory – no disk IO
- Mostly distributed – multi-node topologies
- Parallel processing
- Deal with operational data set
- Map-Reduce support
- Middleware software



RAM storage and parallel distributed processing are two fundamental pillars of in-memory computing.

© 2014 GridGain Systems, Inc.

In-Memory Computing: The Best Use Cases*

- Investment banking
- Insurance claim processing & modeling
- Real-time ad platforms
- Merchant platform for online games
- Geospatial/GIS processing
- Medical imaging processing
- Complex event processing of streaming sensor data

SBERBANK

SONY

worldpay markit

CHRONOTRACK SYSTEMS

*I can only speak for GridGain which has production customers in a wide variety of industries to be statistically significant

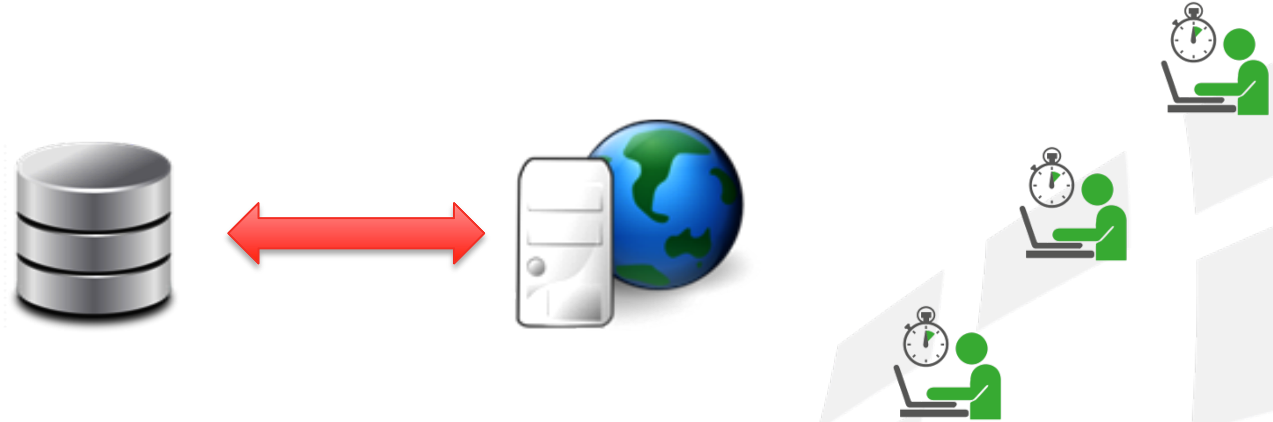
© 2014 GridGain Systems, Inc.

Example System

- Imagined system in the beginning of its lifecycle
- Growing: handling more users and data
- Moving to memory for better characteristics
- Comparison: before and after

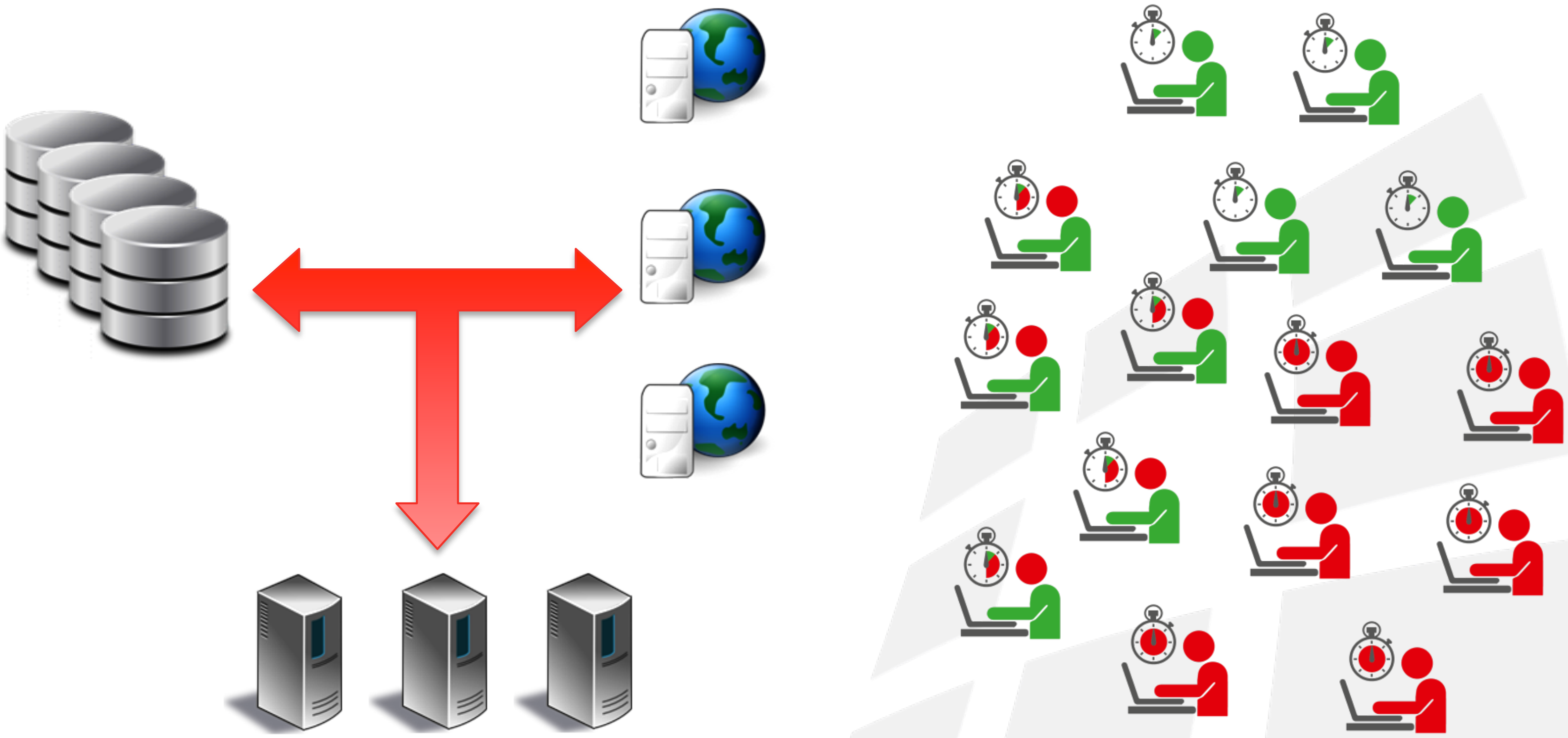
© 2014 GridGain Systems, Inc.

Example System: Just Launched



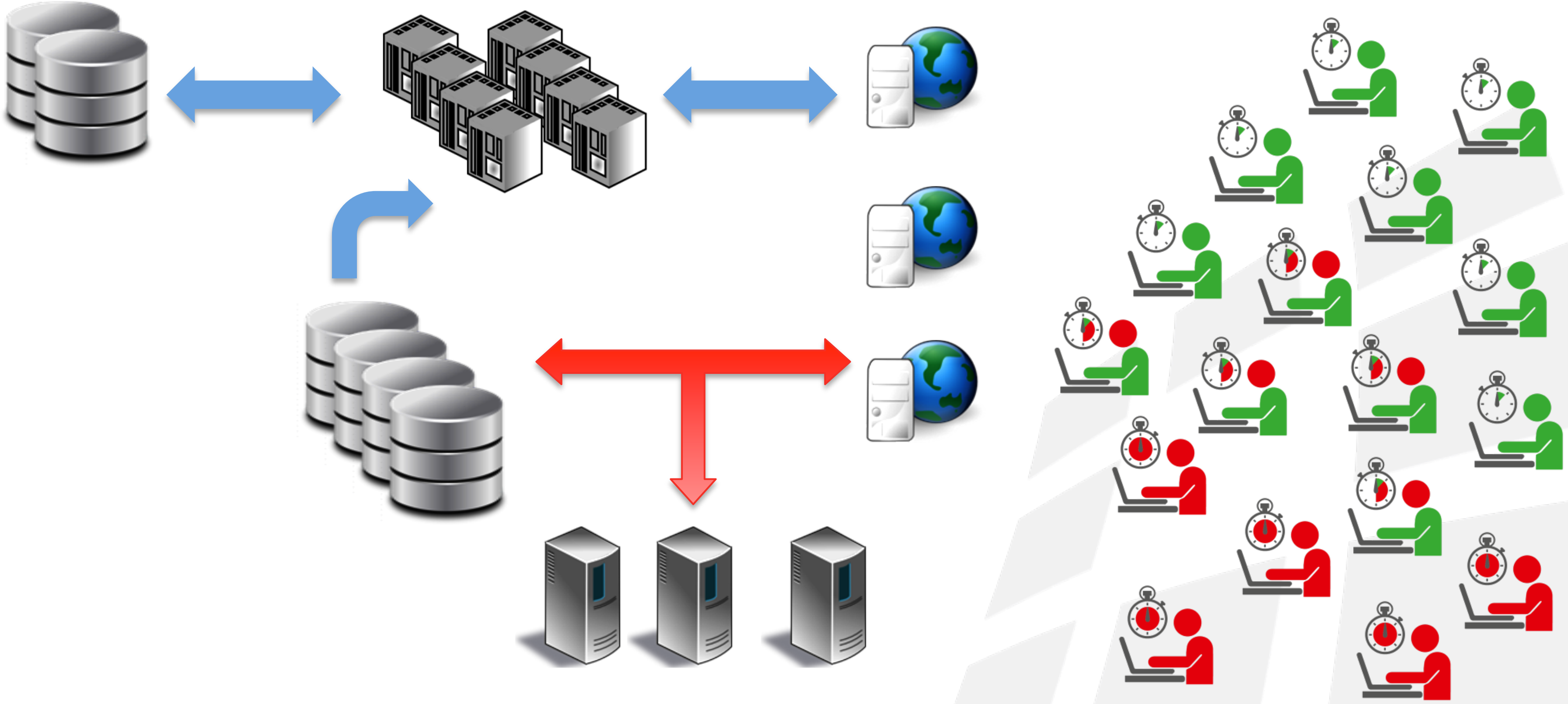
© 2014 GridGain Systems, Inc.

Example System: Growing



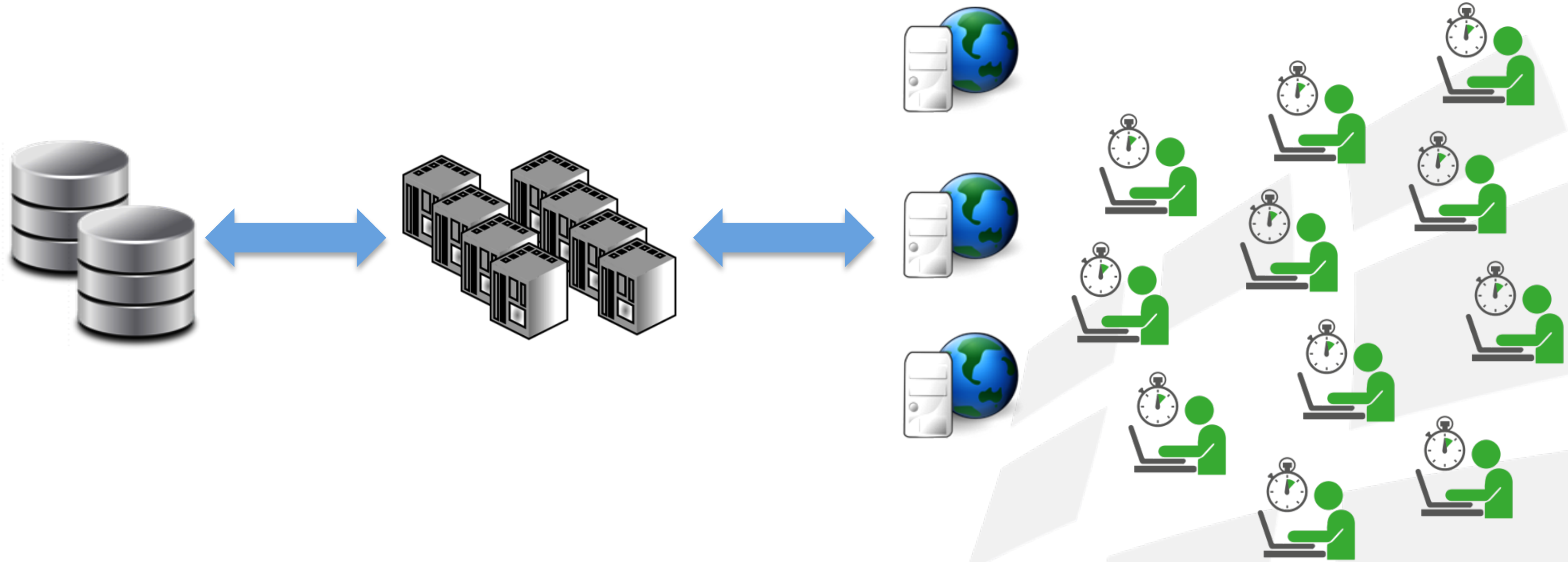
© 2014 GridGain Systems, Inc.

Example System: Moving to Memory (Step 1)



© 2014 GridGain Systems, Inc.

Example System: Moving to Memory (Step 2)



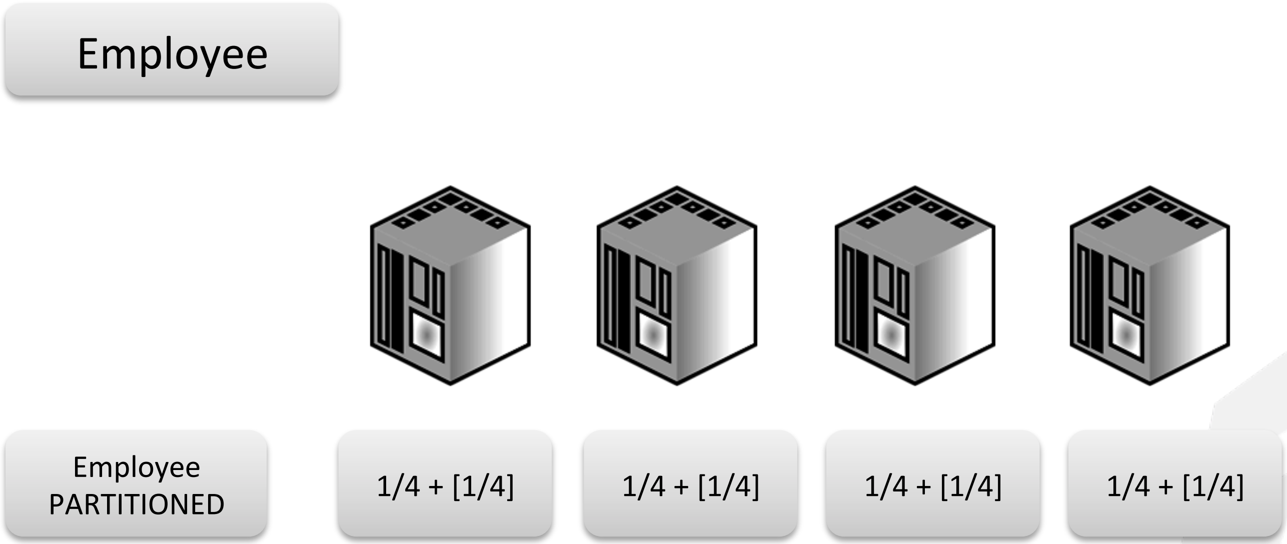
© 2014 GridGain Systems, Inc.

Comparison: Before And After

- Data distribution in in-memory system
- Simple data update scenarios
- Long running queries
- Scalability
- Failure tolerance

© 2014 GridGain Systems, Inc.

Comparison: Data Distribution



Employee

Employee PARTITIONED

1/4 + [1/4]

1/4 + [1/4]

1/4 + [1/4]

1/4 + [1/4]

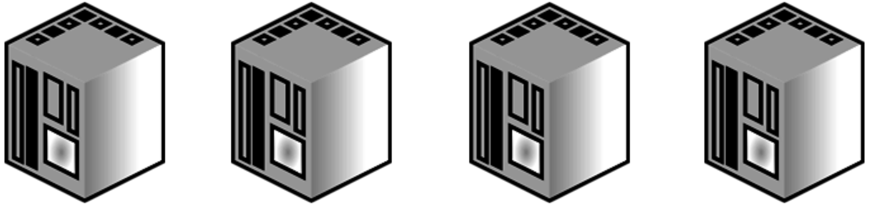
Larger data sets stored in PARTITIONED manner

Each server is PRIMARY for 1/4 of Employee objects and BACKUP for another 1/4.

© 2014 GridGain Systems, Inc.

Comparison: Data Distribution

Company → Employee



Company REPLCATED	FULL	FULL	FULL	FULL
Employee PARTITIONED	1/4 + [1/4]	1/4 + [1/4]	1/4 + [1/4]	1/4 + [1/4]

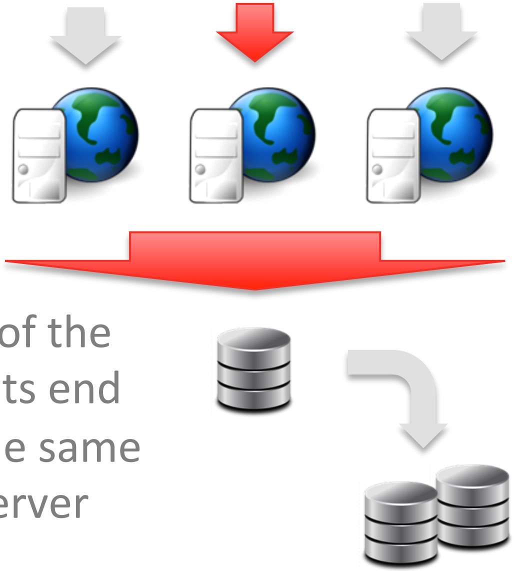
- Smaller data sets may be REPLICATED for colocation
- Larger data sets stored in PARTITIONED manner

Companies and employees are colocated

© 2014 GridGain Systems, Inc.

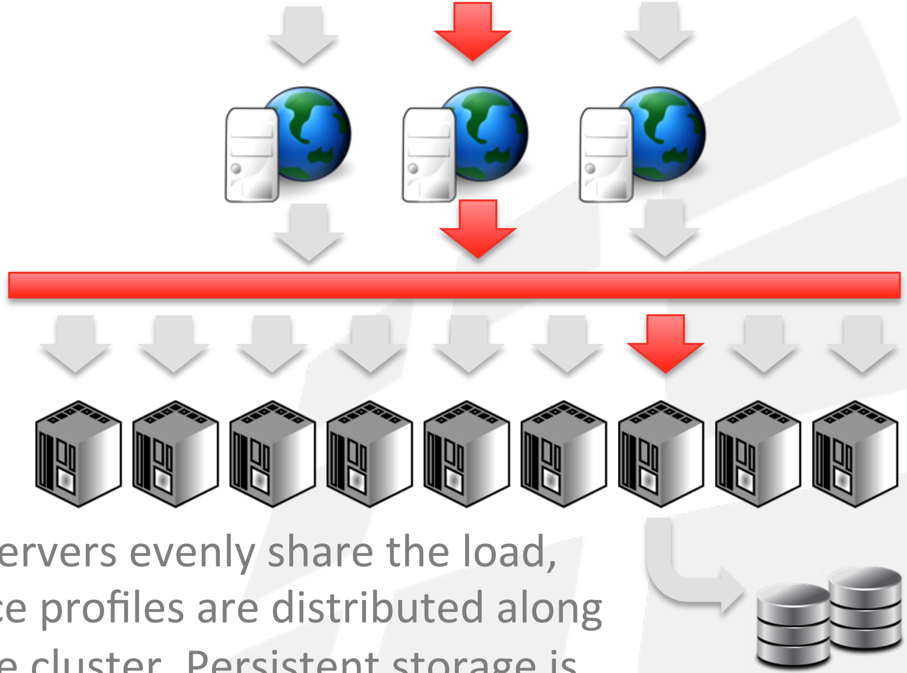
Comparison: Data Access And Update

Before: Update User Profile



100% of the requests end up to the same DB server

After: Update User Profile

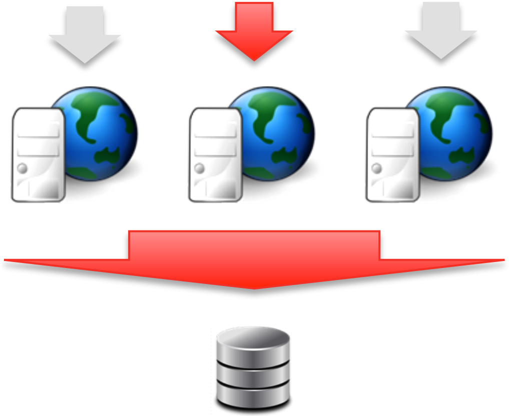


Servers evenly share the load, since profiles are distributed along the cluster. Persistent storage is updated in async manner

© 2014 GridGain Systems, Inc.

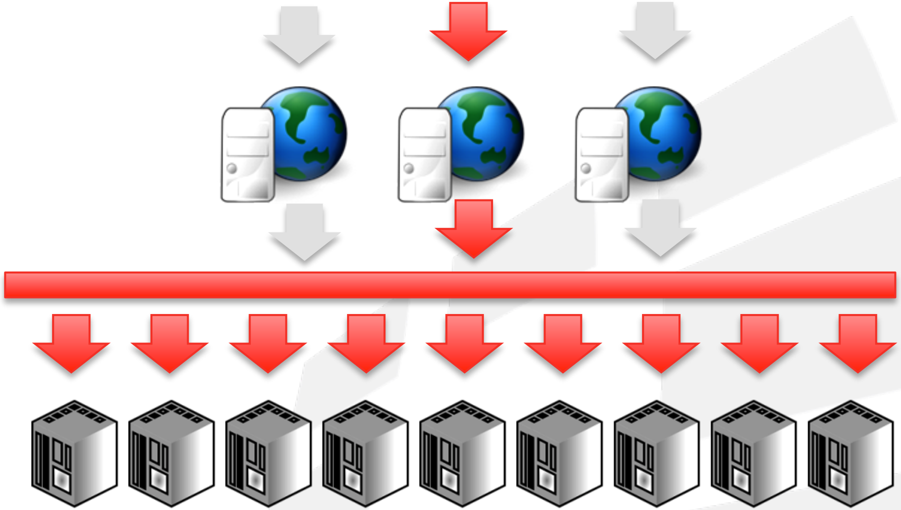
Comparison: Long Running Queries

Before: **select** avg(sum) **from** Orders



100% of the requests end up to the same DB server which is responsible for scanning the entire data set.

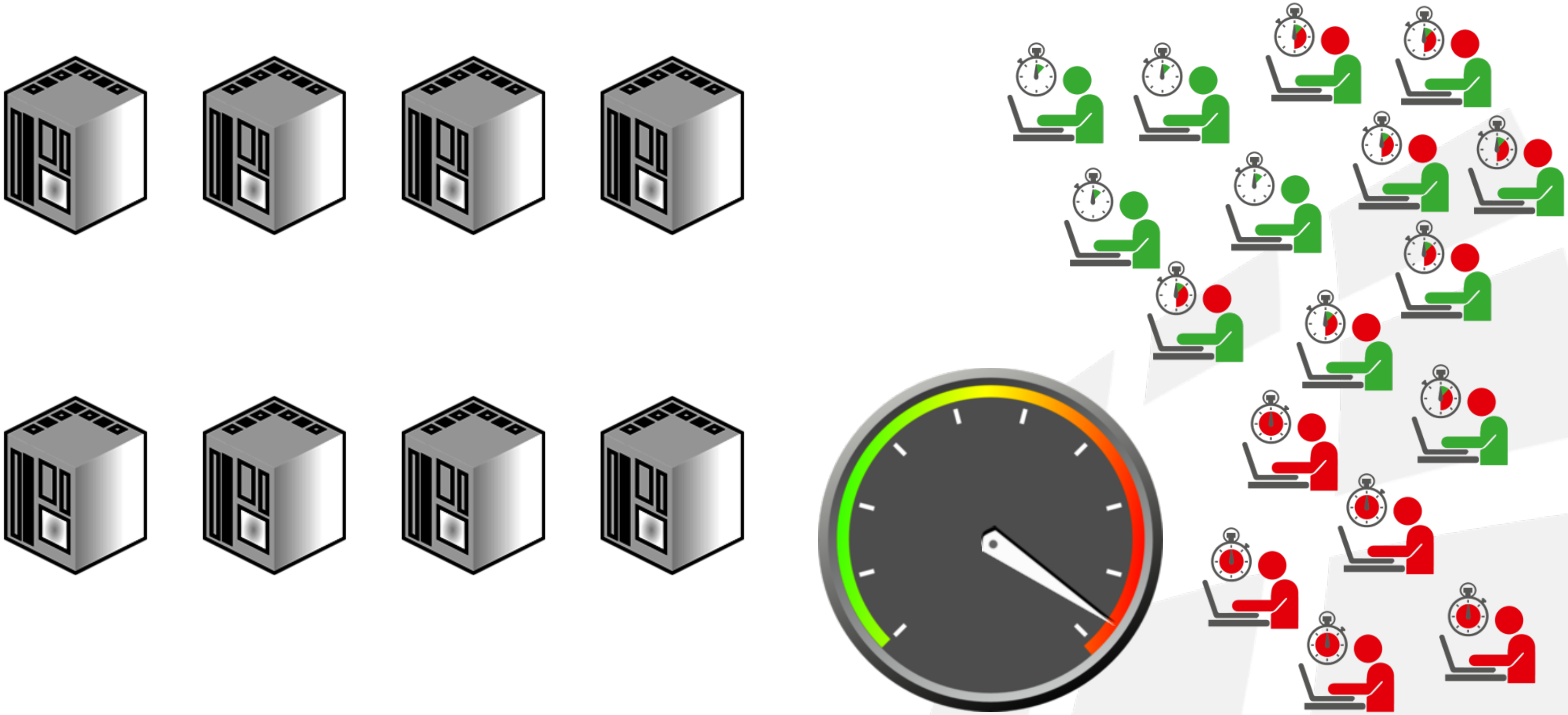
After: **select** avg(sum) **from** Orders



Servers evenly share the load running query against partitioned data. Each server has smaller data set to process.

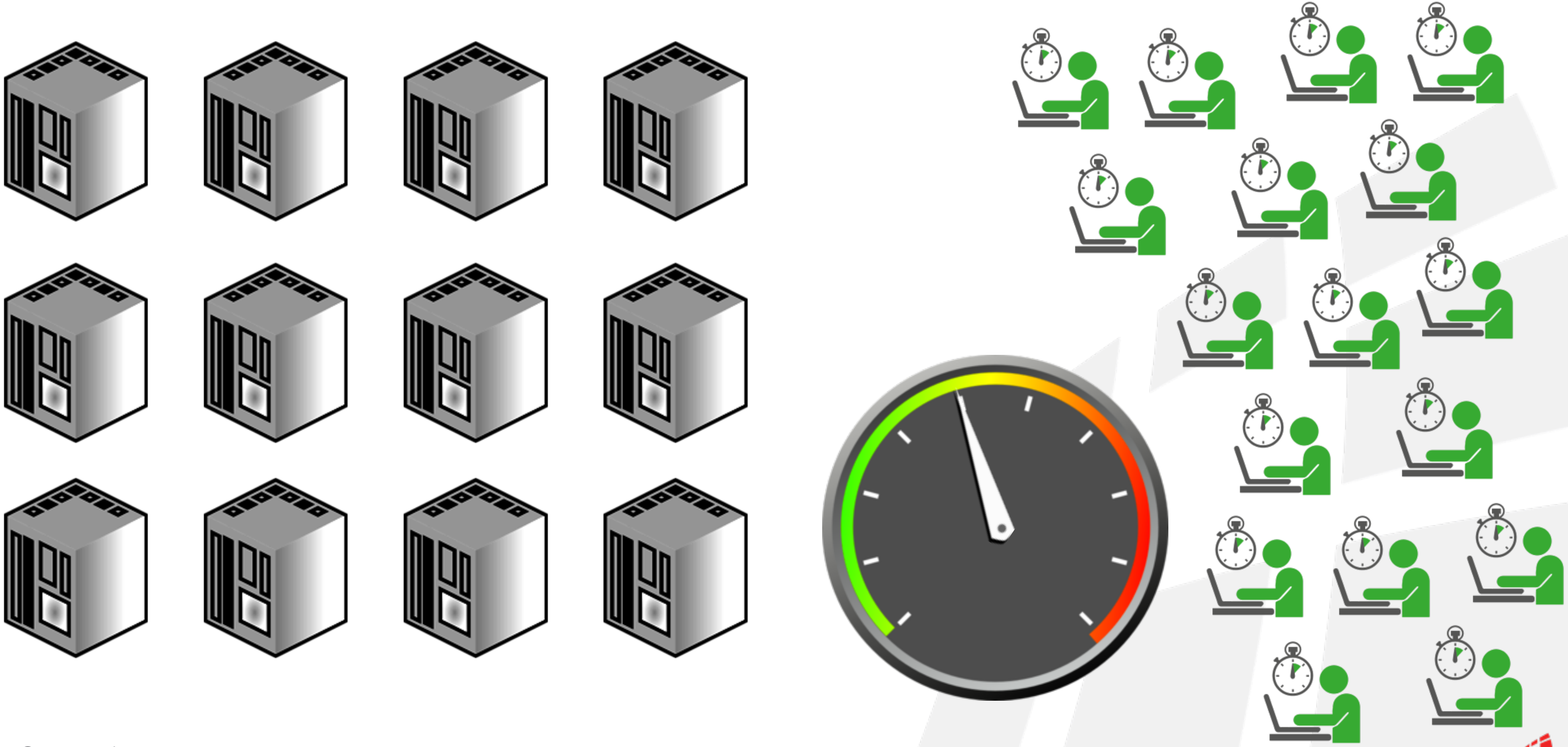
© 2014 GridGain Systems, Inc.

Comparison: Scalability



© 2014 GridGain Systems, Inc.

Comparison: Scalability

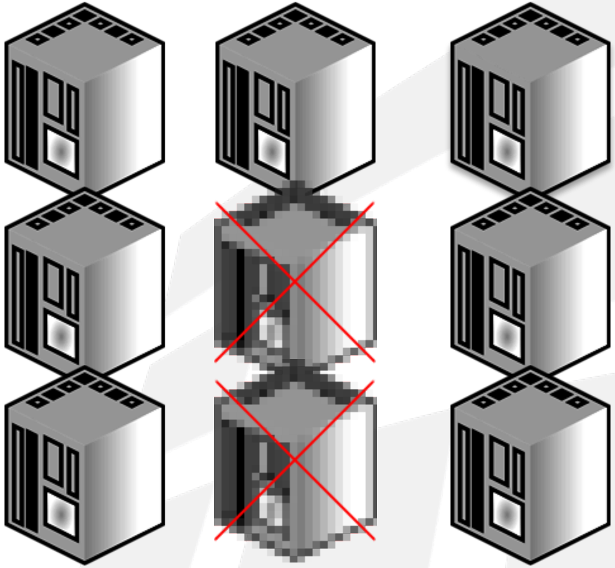


© 2014 GridGain Systems, Inc.

Comparison: Failure Tolerance

Failure of master DB server or certain servers in NoSQL deployment threatens the app.

Failure of 1-2-3-N nodes or planned shutdown does not threaten the cluster.



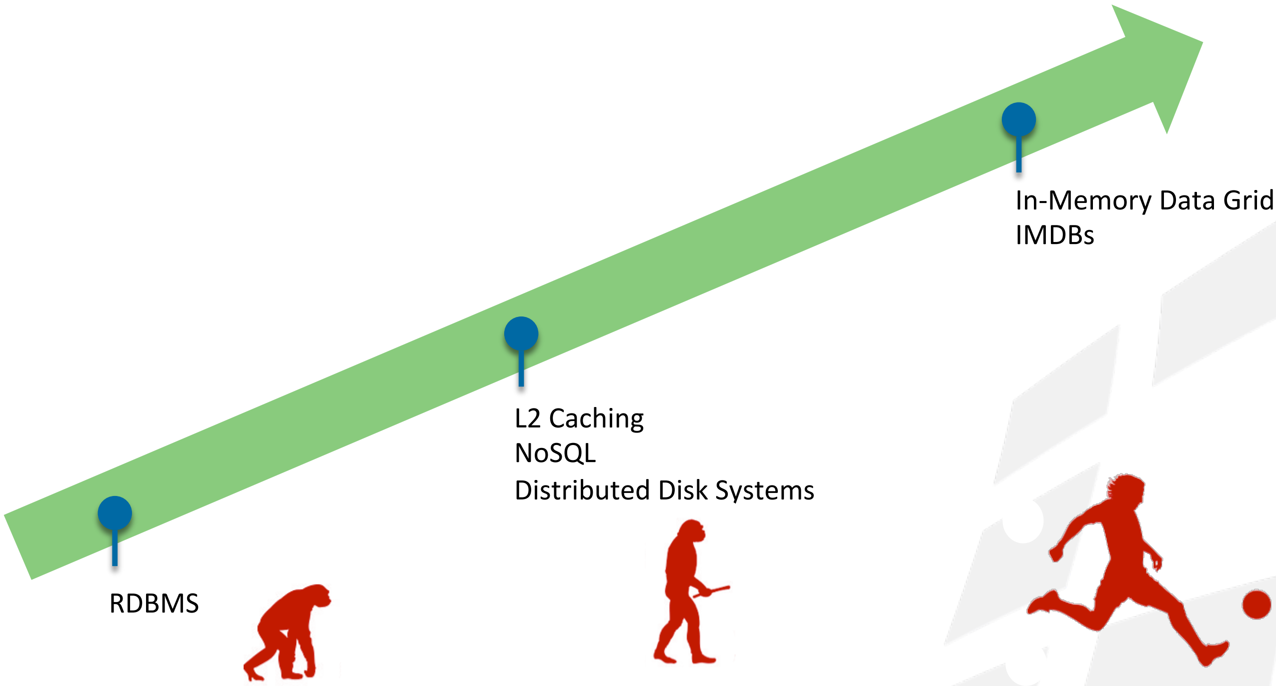
© 2014 GridGain Systems, Inc.

Economics of In-Memory Computing

- High Performance and low Latencies
- RAM + Network Faster than Disk or Flash
- Volatile and Persistent
- Cost Effective
- OLAP and OLTP Use Cases
- Distributed or Not
- Caching, Streaming, Computations
- Data Querying – SQL or Unstructured

© 2014 GridGain Systems, Inc.

Where Is Your Application?



RDBMS

L2 Caching
NoSQL
Distributed Disk Systems

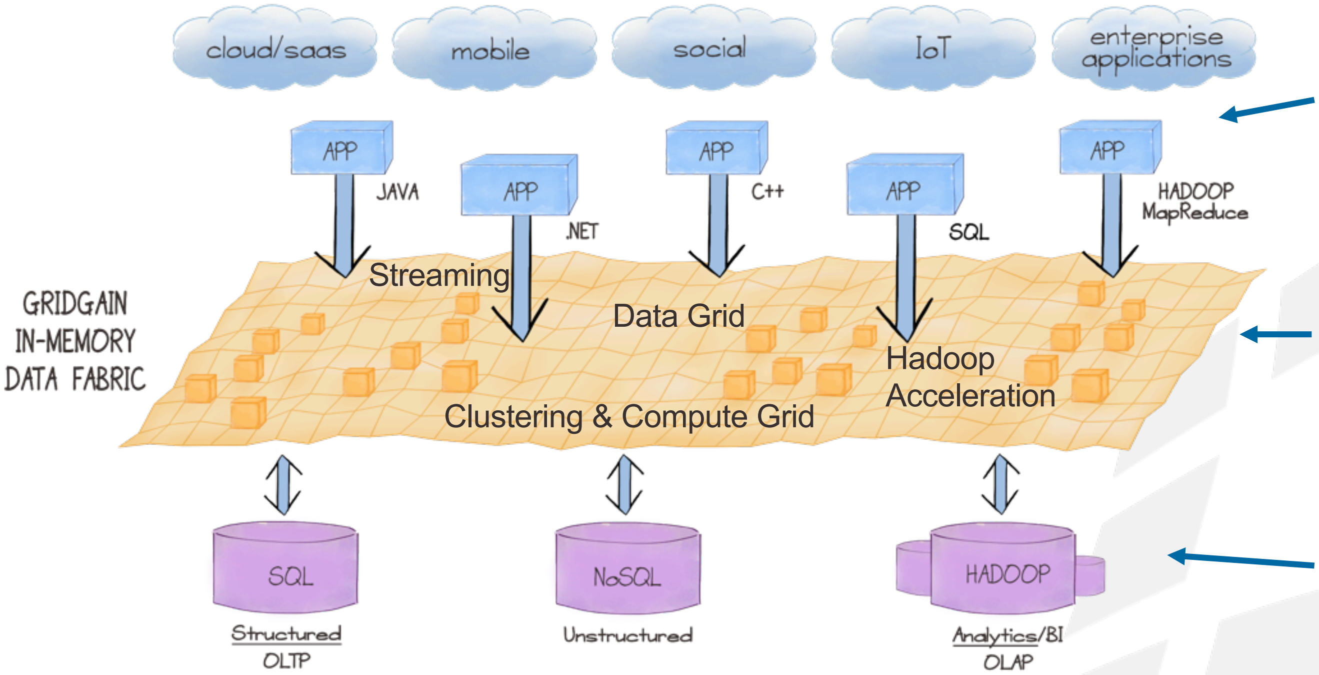
In-Memory Data Grids
IMDBs

© 2014 GridGain Systems, Inc.

GridGain: Try In-Memory Computing

- Full in-memory stack: compute, caching, streaming
- Intuitive APIs – easy to start with
- Open-sourced under Apache 2.0 license
- Hosted and developed on GitHub – fork and enjoy!

GridGain: In-Memory Data Fabric Strategic Approach to IMC



- Supports all Apps

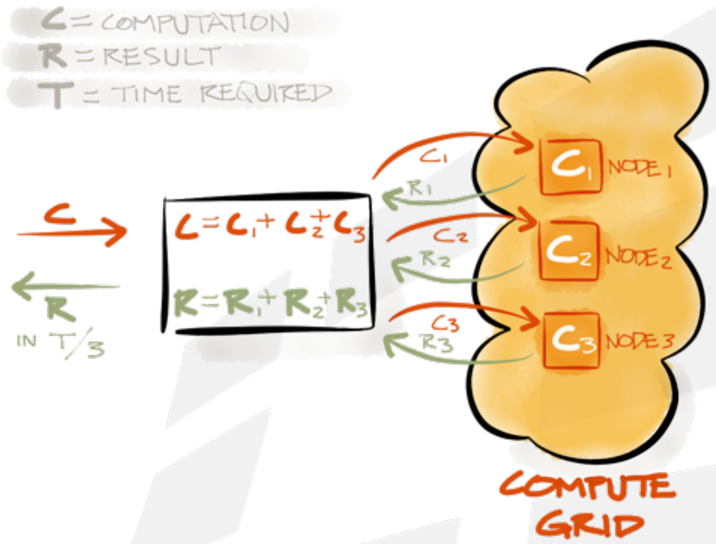
- Simple Java APIs
- 1 JAR Dependency
- High Performance & Scale
- Automatic Fault Tolerance
- Management/Monitoring
- Runs on Commodity Hardware

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

© 2014 GridGain Systems, Inc.

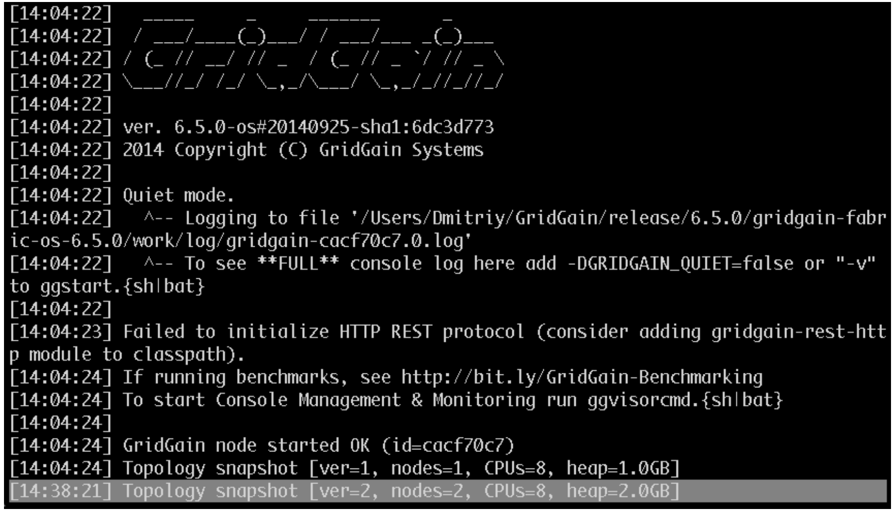
GridGain: Clustering And Compute

- Direct API for MapReduce
- Direct API for Fork/Join
- Zero Deployment
- Cron-like Task Scheduling
- State Checkpoints
- Early and Late Load Balancing
- Automatic Failover
- Full Cluster Management
- Pluggable SPI Design

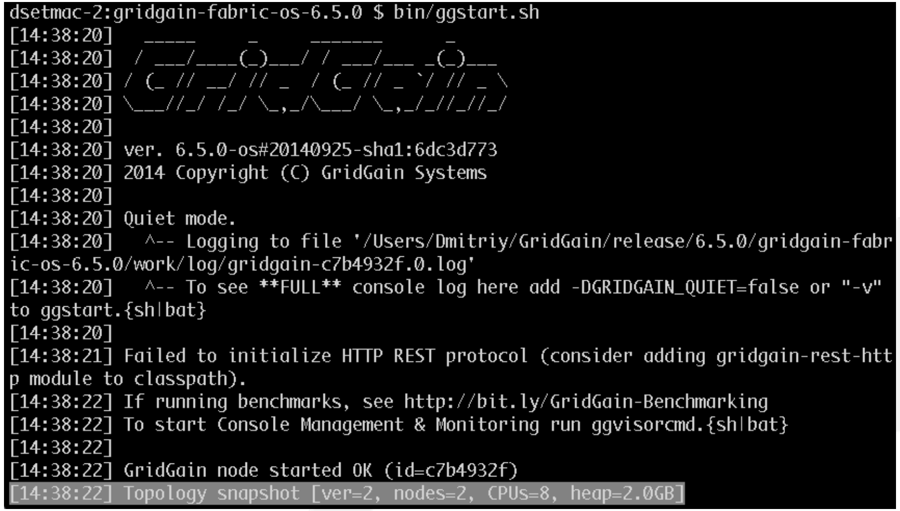


© 2014 GridGain Systems, Inc.

GridGain: Automatic Cluster Discovery



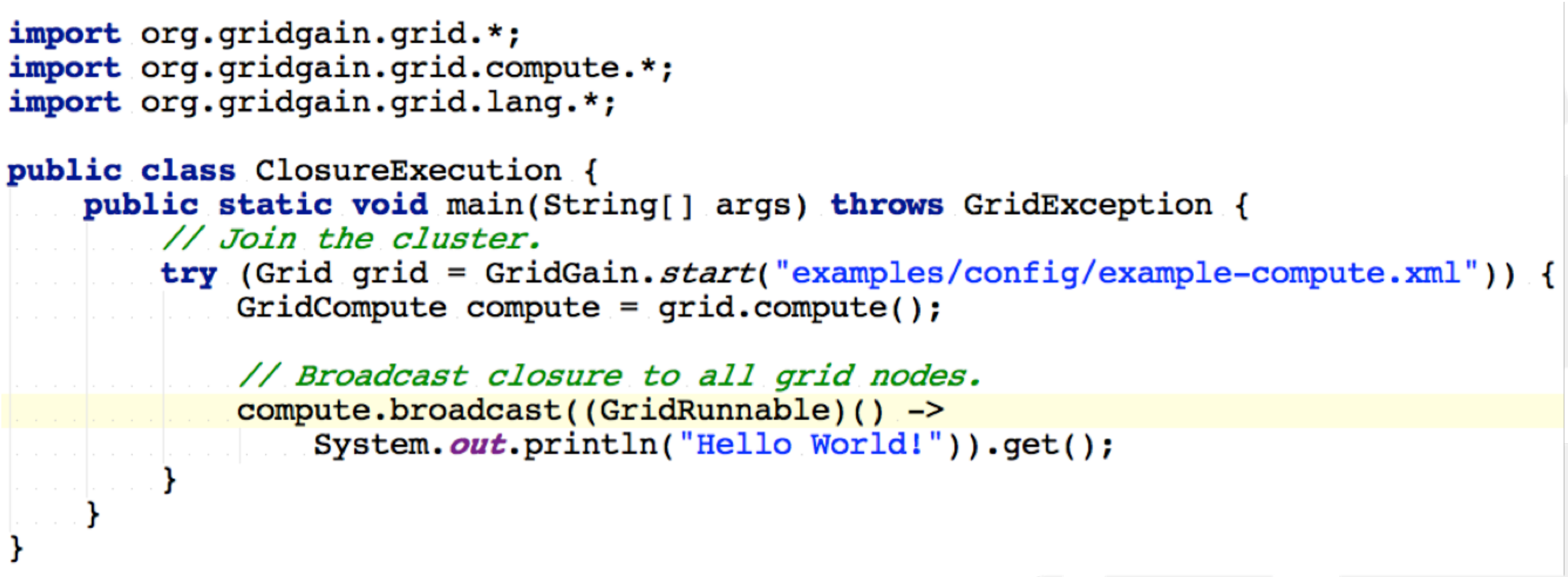
```
[14:04:22]    _____     _     _______      _
[14:04:22]   / ___/____(_)___/ / ___/___ _(_)___
[14:04:22]  / (_ // __/ // _  / (_ // _ `/ // _ \
[14:04:22]  \___//_/ /_/ \_,_/\___/ \_,_/_//_//_/
[14:04:22]
[14:04:22] ver. 6.5.0-os#20140925-sha1:6dc3d773
[14:04:22] 2014 Copyright (C) GridGain Systems
[14:04:22]
[14:04:22] Quiet mode.
[14:04:22]   ^-- Logging to file '/Users/Dmitriy/GridGain/release/6.5.0/gridgain-fabr
ic-os-6.5.0/work/log/gridgain-cacf70c7.0.log'
[14:04:22]   ^-- To see **FULL** console log here add -DGRIDGAIN_QUIET=false or "-v"
to ggstart.{sh|bat}
[14:04:22]
[14:04:23] Failed to initialize HTTP REST protocol (consider adding gridgain-rest-htt
p module to classpath).
[14:04:24] If running benchmarks, see http://bit.ly/GridGain-Benchmarking
[14:04:24] To start Console Management & Monitoring run ggvisorcmd.{sh|bat}
[14:04:24]
[14:04:24] GridGain node started OK (id=cacf70c7)
[14:04:24] Topology snapshot [ver=1, nodes=1, CPUs=8, heap=1.0GB]
[14:38:21] Topology snapshot [ver=2, nodes=2, CPUs=8, heap=2.0GB]
```



```
dsetmac-2:gridgain-fabric-os-6.5.0 $ bin/ggstart.sh
[14:38:20]    _____     _     _______      _
[14:38:20]   / ___/____(_)___/ / ___/___ _(_)___
[14:38:20]  / (_ // __/ // _  / (_ // _ `/ // _ \
[14:38:20]  \___//_/ /_/ \_,_/\___/ \_,_/_//_//_/
[14:38:20]
[14:38:20] ver. 6.5.0-os#20140925-sha1:6dc3d773
[14:38:20] 2014 Copyright (C) GridGain Systems
[14:38:20]
[14:38:20] Quiet mode.
[14:38:20]   ^-- Logging to file '/Users/Dmitriy/GridGain/release/6.5.0/gridgain-fabr
ic-os-6.5.0/work/log/gridgain-c7b4932f.0.log'
[14:38:20]   ^-- To see **FULL** console log here add -DGRIDGAIN_QUIET=false or "-v"
to ggstart.{sh|bat}
[14:38:20]
[14:38:21] Failed to initialize HTTP REST protocol (consider adding gridgain-rest-htt
p module to classpath).
[14:38:22] If running benchmarks, see http://bit.ly/GridGain-Benchmarking
[14:38:22] To start Console Management & Monitoring run ggvisorcmd.{sh|bat}
[14:38:22]
[14:38:22] GridGain node started OK (id=c7b4932f)
[14:38:22] Topology snapshot [ver=2, nodes=2, CPUs=8, heap=2.0GB]
```

© 2014 GridGain Systems, Inc.

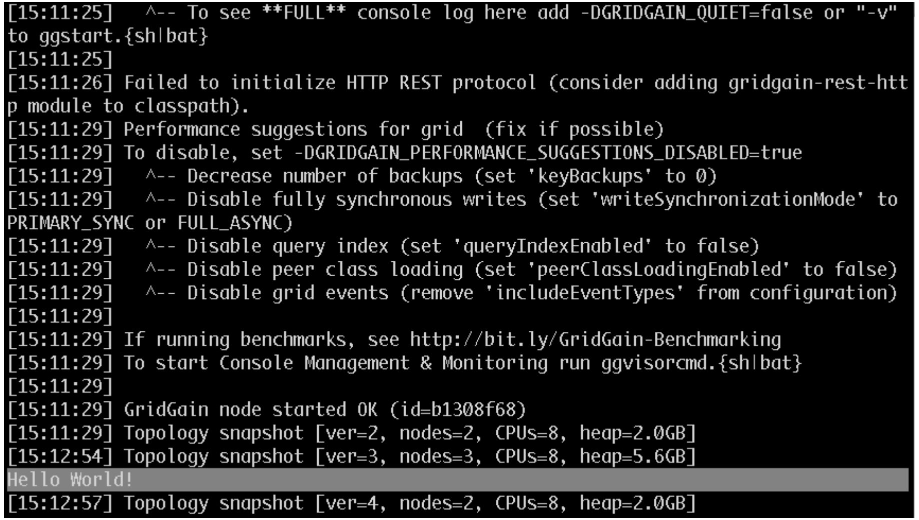
GridGain: Closure Execution



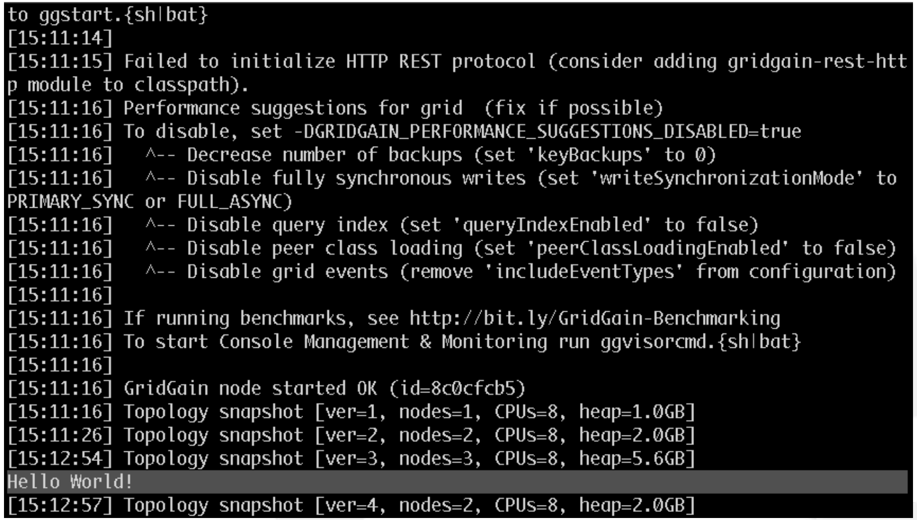
```java
import org.gridgain.grid.*;
import org.gridgain.grid.compute.*;
import org.gridgain.grid.lang.*;

public class ClosureExecution {
    public static void main(String[] args) throws GridException {
        // Join the cluster.
        try (Grid grid = GridGain.start("examples/config/example-compute.xml")) {
            GridCompute compute = grid.compute();

            // Broadcast closure to all grid nodes.
            compute.broadcast((GridRunnable)() ->
                System.out.println("Hello World!")).get();
        }
    }
}
```

© 2014 GridGain Systems, Inc.

GridGain: Closure Execution



```
[15:11:25]   ^-- To see **FULL** console log here add -DGRIDGAIN_QUIET=false or "-v"
to ggstart.{sh|bat}
[15:11:25]
[15:11:26] Failed to initialize HTTP REST protocol (consider adding gridgain-rest-htt
p module to classpath).
[15:11:29] Performance suggestions for grid  (fix if possible)
[15:11:29] To disable, set -DGRIDGAIN_PERFORMANCE_SUGGESTIONS_DISABLED=true
[15:11:29]   ^-- Decrease number of backups (set 'keyBackups' to 0)
[15:11:29]   ^-- Disable fully synchronous writes (set 'writeSynchronizationMode' to
PRIMARY_SYNC or FULL_ASYNC)
[15:11:29]   ^-- Disable query index (set 'queryIndexEnabled' to false)
[15:11:29]   ^-- Disable peer class loading (set 'peerClassLoadingEnabled' to false)
[15:11:29]   ^-- Disable grid events (remove 'includeEventTypes' from configuration)
[15:11:29]
[15:11:29] If running benchmarks, see http://bit.ly/GridGain-Benchmarking
[15:11:29] To start Console Management & Monitoring run ggvisorcmd.{sh|bat}
[15:11:29]
[15:11:29] GridGain node started OK (id=b1308f68)
[15:11:29] Topology snapshot [ver=2, nodes=2, CPUs=8, heap=2.0GB]
[15:12:54] Topology snapshot [ver=3, nodes=3, CPUs=8, heap=5.6GB]
Hello World!
[15:12:57] Topology snapshot [ver=4, nodes=2, CPUs=8, heap=2.0GB]
```

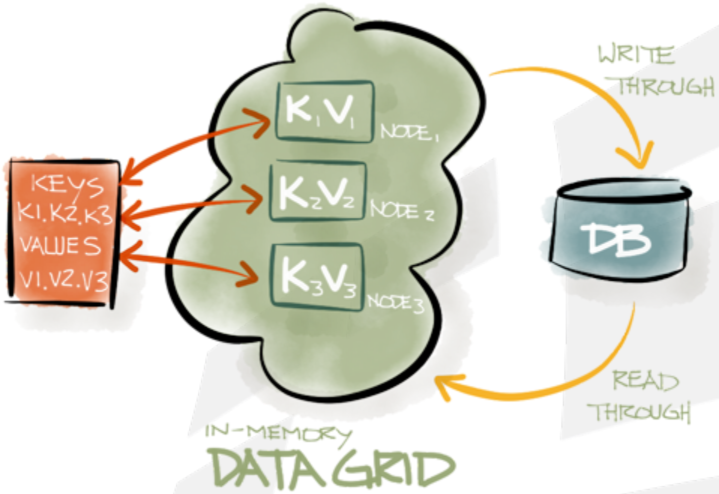


```
to ggstart.{sh|bat}
[15:11:14]
[15:11:15] Failed to initialize HTTP REST protocol (consider adding gridgain-rest-htt
p module to classpath).
[15:11:16] Performance suggestions for grid  (fix if possible)
[15:11:16] To disable, set -DGRIDGAIN_PERFORMANCE_SUGGESTIONS_DISABLED=true
[15:11:16]   ^-- Decrease number of backups (set 'keyBackups' to 0)
[15:11:16]   ^-- Disable fully synchronous writes (set 'writeSynchronizationMode' to
PRIMARY_SYNC or FULL_ASYNC)
[15:11:16]   ^-- Disable query index (set 'queryIndexEnabled' to false)
[15:11:16]   ^-- Disable peer class loading (set 'peerClassLoadingEnabled' to false)
[15:11:16]   ^-- Disable grid events (remove 'includeEventTypes' from configuration)
[15:11:16]
[15:11:16] If running benchmarks, see http://bit.ly/GridGain-Benchmarking
[15:11:16] To start Console Management & Monitoring run ggvisorcmd.{sh|bat}
[15:11:16]
[15:11:16] GridGain node started OK (id=8c0cfcb5)
[15:11:16] Topology snapshot [ver=1, nodes=1, CPUs=8, heap=1.0GB]
[15:11:26] Topology snapshot [ver=2, nodes=2, CPUs=8, heap=2.0GB]
[15:12:54] Topology snapshot [ver=3, nodes=3, CPUs=8, heap=5.6GB]
Hello World!
[15:12:57] Topology snapshot [ver=4, nodes=2, CPUs=8, heap=2.0GB]
```

© 2014 GridGain Systems, Inc.

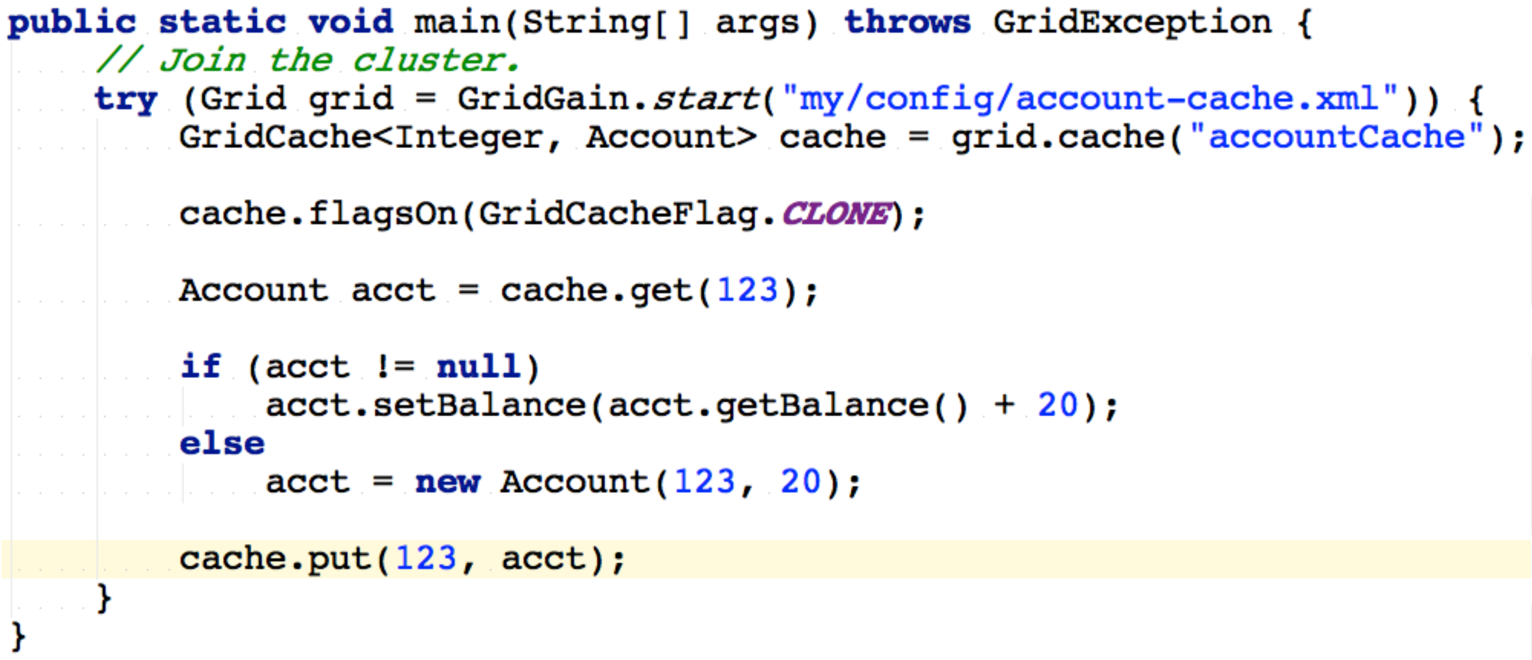
GridGain: In-Memory Caching and Data Grid

- Distributed In-Memory Key-Value Store
- Replicated and Partitioned
- TBs of data, of any type
- On-Heap and Off-Heap Storage
- Backup Replicas / Automatic Failover
- Distributed ACID Transactions
- SQL queries and JDBC driver
- Colocation of Compute and Data



© 2014 GridGain Systems, Inc.

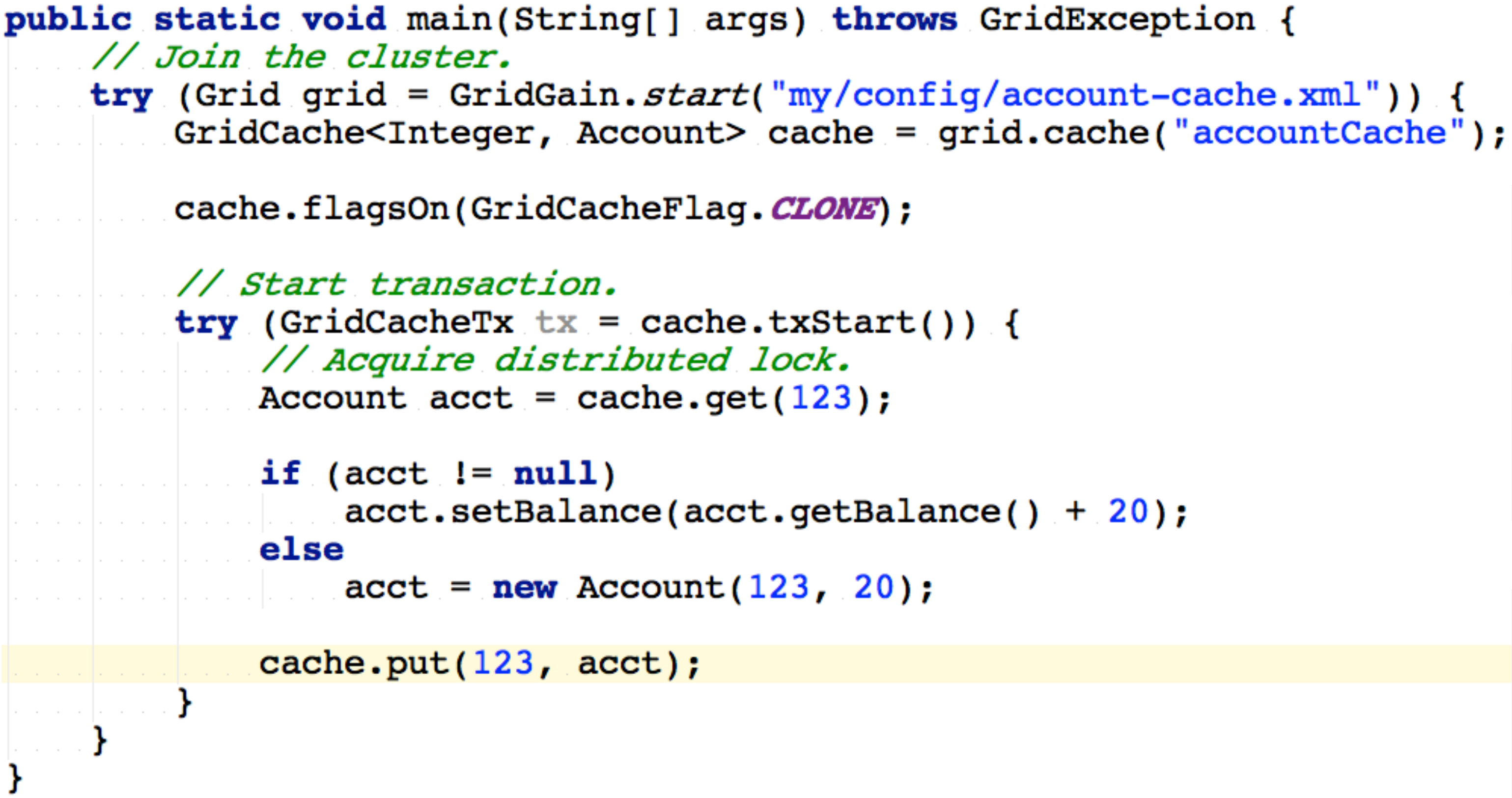
GridGain: Cache Operations



```
public static void main(String[] args) throws GridException {
    // Join the cluster.
    try (Grid grid = GridGain.start("my/config/account-cache.xml")) {
        GridCache<Integer, Account> cache = grid.cache("accountCache");

        cache.flagsOn(GridCacheFlag.CLONE);

        Account acct = cache.get(123);

        if (acct != null)
            acct.setBalance(acct.getBalance() + 20);
        else
            acct = new Account(123, 20);

        cache.put(123, acct);
    }
}
```

© 2014 GridGain Systems, Inc.

GridGain: Cache Transaction

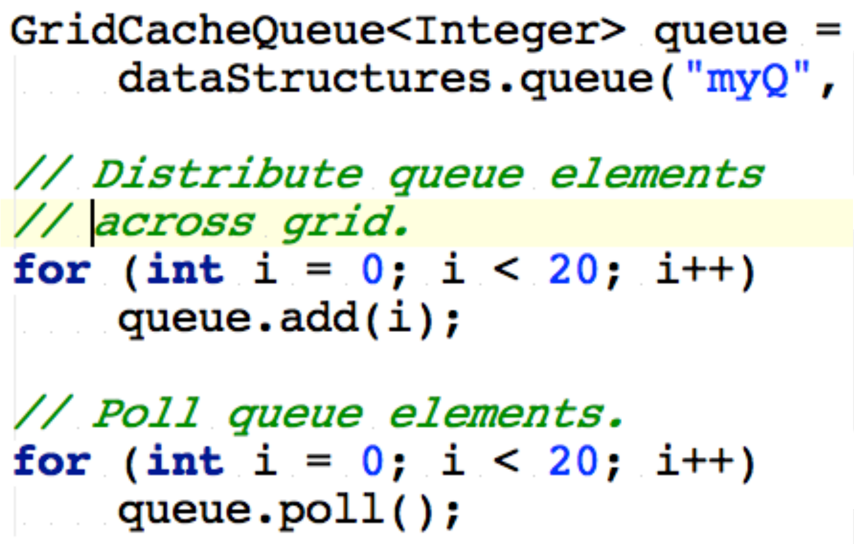


```java
public static void main(String[] args) throws GridException {
    // Join the cluster.
    try (Grid grid = GridGain.start("my/config/account-cache.xml")) {
        GridCache<Integer, Account> cache = grid.cache("accountCache");

        cache.flagsOn(GridCacheFlag.CLONE);

        // Start transaction.
        try (GridCacheTx tx = cache.txStart()) {
            // Acquire distributed lock.
            Account acct = cache.get(123);

            if (acct != null)
                acct.setBalance(acct.getBalance() + 20);
            else
                acct = new Account(123, 20);

            cache.put(123, acct);
        }
    }
}
```

© 2014 GridGain Systems, Inc.

GridGain: Distributed Java Data Structures

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



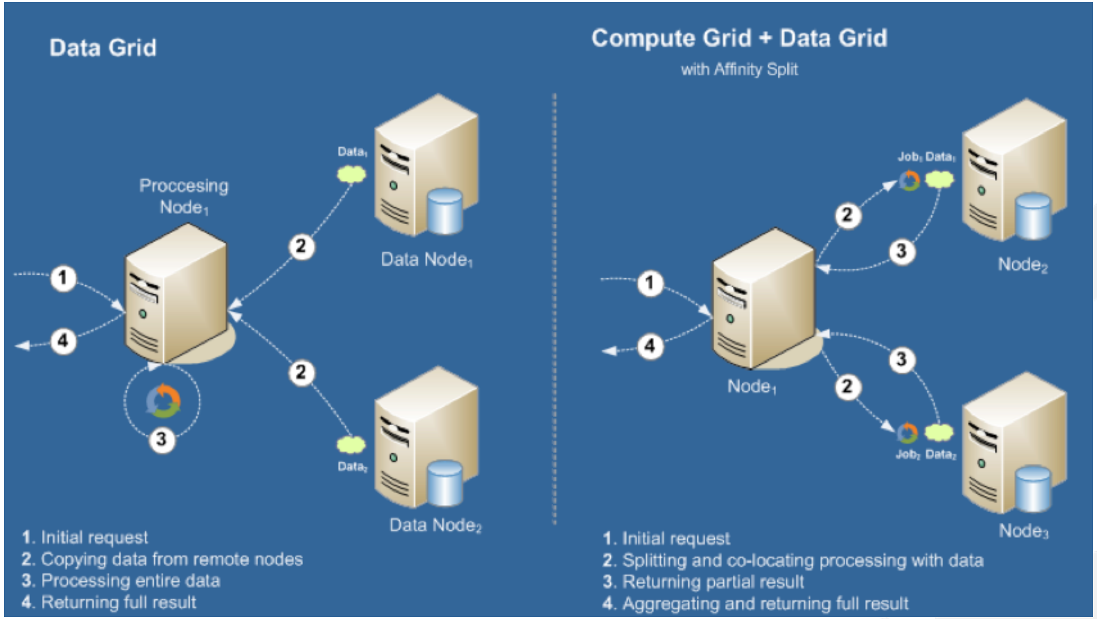
```
GridCacheQueue<Integer> queue =
    dataStructures.queue("myQ",

// Distribute queue elements
// across grid.
for (int i = 0; i < 20; i++)
    queue.add(i);

// Poll queue elements.
for (int i = 0; i < 20; i++)
    queue.poll();
```

© 2014 GridGain Systems, Inc.

Client-Server vs Affinity Colocation

Client-Server

Affinity Colocation



1. Initial request
2. Copying data from remote nodes
3. Processing entire data
4. Returning full result

1. Initial request
2. Splitting and co-locating processing with data
3. Returning partial result
4. Aggregating and returning full result

© 2014 GridGain Systems, Inc.

GridGain

THANK YOU

www.gridgain.com 🐦 @gridgain

© 2014 GridGain Systems, Inc.