GridGain – Java Grid Computing Made Simple

Dmitriy Setrakyan
www.gridgain.org

Agenda

- GridGain
  - What is Grid Computing and why
  - GridGain In a Glance
  - Key Concepts
- Demos
  - Grid Application in 15 Minutes

What is Grid Computing?

- Compute Grids
  - Parallelize execution
- Data Grids
  - Parallelize data storage
- **Grid Computing = Compute Grids + Data Grids**
  - a.k.a. Data Partitioning + Affinity Map/Reduce
- Utility, on-demand, cloud computing...?

Why Grid Computing?

- Ask Google, Yahoo, eBay, Amazon
  - Amazon: 100ms latency cost 1% of sales
  - Google: 500ms latency drops traffic 20%
  - Financial: $4M/ms lose if 5ms behind
- Google’s VP Marissa Mayer:
  “Users Really Respond to Speed”
- Solves problems often unsolvable otherwise
  - Google has ~1,000,000 nodes in its grid
- Uniformed programming paradigm
  - Scales from garage to Google

GridGain In a Glance

Open Source Java Grid Computing

- Grid Computing
  - Innovative Compute Grid
  - Integration with Data Grids
- Java
  - Built in Java and for Java
- Open Source
  - LGPL and Apache 2.0

Elegant Simplicity with Powerful Features

Professional Open Source

- GridGain - Professional Open Source
  - Free and Open Source licenses: LGPL and Apache 2.0
  - Commercial support, training and consulting
- Best business model for software middleware
- Like JBoss, Spring Source, Mule Source…

GridGain Statistics

In 12 months since the 1st release:

- Over 20,000 downloads
- Starts every 60 seconds around the globe
- One of the largest Amazon EC2 clouds – 512 nodes
- Over 2000 different individuals, projects and organizations

Fastest Growing Java Grid Computing Middleware

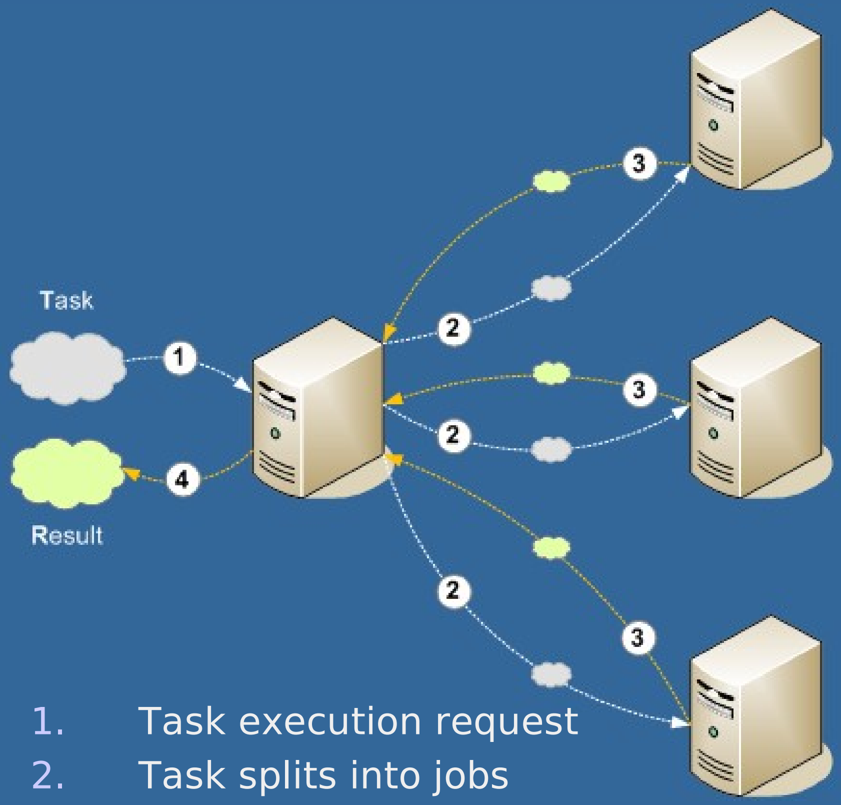
Key Concepts

- MapReduce
- Zero Deployment
- On Demand Scalability
- Fault Tolerance
- LEGO-like Integration
- Transparent Grid Enabling
- Data Grids Integration
- JMX Monitoring

MapReduce

Features:

- Direct API support for MapReduce
- Pluggable failover resolution
- Pluggable topology resolution
- Distributed task session
- Annotation-based execution
- Asynchronous execution
- Redundant mapping
- Partial asynchronous reduction
- Adaptive split
- Checkpoints for long running tasks
- Early and late load balancing
- Affinity co-location with data grids



1. Task execution request
2. Task splits into jobs
3. Result of job execution
4. Aggregation of job results

Zero Deployment

- Peer-to-Peer Grid Class Loading technology
  - No Ant scripts to run
  - No JARs to copy or FTP
  - No need to restart
- Develop in EXACTLY the same way as locally
  - Change ► Compile ► Run on the grid
- Start many grid nodes in
  - Single JVM – debug grid apps locally (!)
  - Single computer – run grid on your workstation

=> Biggest developer's productivity boost

On Demand Scalability

- Early and late load balancing:
  - Optimal scalability for non-deterministic execution on the grid
- Load Balancing SPI
  - Early load balancing
- Collision SPI
  - Late load balancing

=> Most comprehensive scalability support

Fault Tolerance

- Customizable failover resolution
  - Automatic failover
  - Fail-fast, fail-slow implementation
- Failure – is result too
- Redundant jobs
- Asynchronous results processing
  - Policy-based continuation
- Checkpoints for long-running tasks
  - “Smart” restart in case of failover
- => Most comprehensive fault tolerance functionality

LEGO-Like Integration

- Service Provider Interface (SPI)-based architecture
  - Plug in and customize almost any aspect of grid computing framework
  - LEGO-like assembly of custom grid infrastructure
  - Design approach enabling transparent usability for HPC, traditional grid computing and cloud computing
- Grid computing framework aspect that are fully pluggable:
    - Communication
    - Discovery
    - Tracing
    - Startup
    - Event storage
    - Marshalling
    - OnDemand
    - Checkpoints
    - Failover
    - Collision Resolution
    - Topology management
    - Load balancing
    - Deployment

LEGO-like Integration

"Out-of-the-box" integration with:

Application Servers
- JBoss AS
- BEA Weblogic
- IBM Websphere
- Glassfish
- Tomcat

Data Grids
- JBoss Cache
- Coherence
- GigaSpaces

AOP
- JBoss AOP
- Spring AOP
- AspectJ

Messaging Middleware
- Mule
- JMS
  - ActiveMQ
  - SunMQ
- Jgroups
- Email
- TCP, IP-Multicast

Others
- Spring
- Junit
- JXInsight

Transparent Grid Enabling

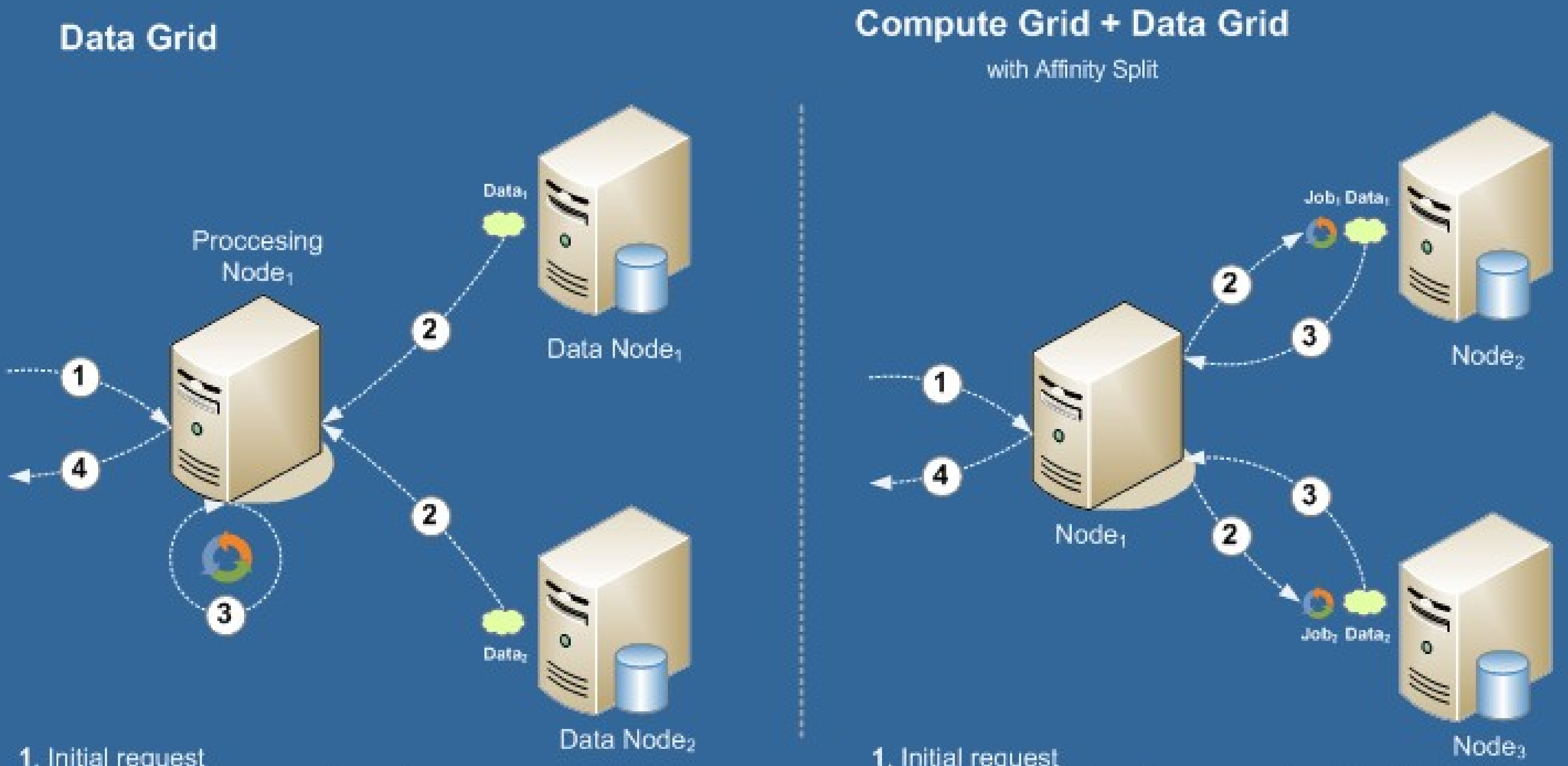
```
class BizLogic {
  @Gridify(...)
  public static Result process(String param) {
    ...
  }
}

class Caller {
  public static void Main(String[] args) {
    GridFactory.start();

    try {
      BizLogic.process(args[0]);
    }
    finally {
      GridFactory.stop();
    }
  }
}
```

Execution of process() method will be performed on the grid

Data Grids Integration

- Integration with Data Grids – key to ultimate scalability
- Affinity MapReduce – ability to co-locate processing logic and the data
  - a.k.a. Data-aware routing
  - Minimizes “noise” traffic
  - Optimal grid load and performance
- Out-of-the-box support:
  - JBoss Cache
  - Oracle Coherence

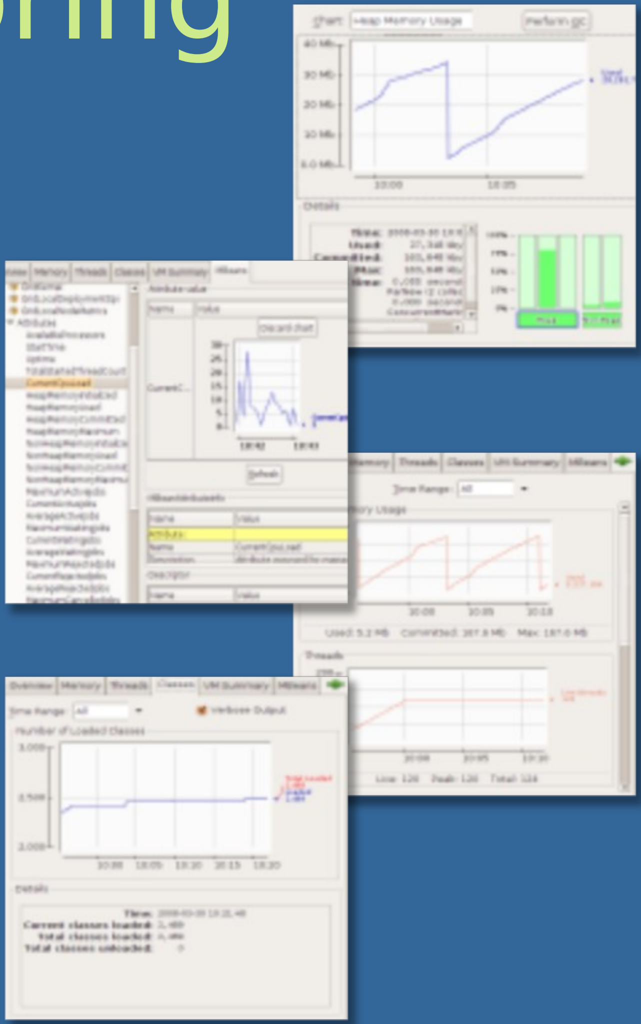
Data Grid Integration



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

JMX Monitoring



- Full JMX instrumentation
  - Every SPI
  - Kernal
  - Public APIs
- Flexible access
  - Programmatic via JMX API
  - From GUI JMX console
    - Jboss Management
    - Hyperic
    - Jconsole/VisualVM

Roadmap

- GridGain 1.5 - July 2007
- GridGain 2.0 - February 2008
- GridGain 3.0 - Q109
  - Improved support for cloud computing with Amazon EC2
  - Web 2.0 Grid Computing: REST + JSON
  - Enhanced Management and Monitoring

Demos

- Java 5/Eclipse 3.2/Windows Vista
- GridGain 2.0

Q & A

Thanks for your time!

Dmitriy Setrakyan: dsetrakyan@gridgain.com

GridGain: www.gridgain.org