



Grid Computing with AOP – Fun, Simple and Productive

Nikita Ivanov

www.gridgain.org





Introduction

- Nikita Ivanov
 - Over 15 years of experience
 - Last 7 years developing grid computing and distributed middleware
 - JSR-107 “JCache” Expert Group member
- GridGain Founder, www.gridgain.org
 - Computational grid
 - Java 5
 - LGPL professional open-source



Presentation Overview

- What is Grid Computing and why?
- Why Java?
- Why Open Source?
- What is AOP and why?
- What is wrong with existing products?
- Live coding demo – fun :-)



What is Grid Computing?

- **Computational Grid Computing**
 - **Parallelize work load**
- **Data Grids**
 - **Parallelize data load**
- **Management Grids**
 - **Oracle: Fault tolerance**
- **Utility or On Demand Computing**
 - **Data centers**



Computational Grid Computing

- Split/Aggregate *a.k.a.* Map/Reduce
- Topology management
- Fault tolerance
- Resource management
- Integration



Why Grid Computing?

- Ask Google, Yahoo, eBay, Amazon
- Solves problems often unsolvable otherwise
- Allows smaller businesses to compete with big corporations
 - Many examples in financial sector
- Ideal technology for Web 2.0, mash-ups, geo-processing, *etc.*
 - Start small – grow with the business
 - Uniform programming model throughout life-cycle of business



Why Java?

- Most of today grids are running C/C++
 - HPC/MPI legacy
 - Fortran/C/C++ libraries
- Yet Java has a number of intrinsic benefits:
 - Much more productive than C/C++
 - Performance gap is minimal
 - Easily call out to existing Fortran/C/C++ libraries
 - Extensive server-side software stack
 - Commercial
 - Professional open-source
 - Cross-platform (at least for server side systems)
 - Makes heterogeneous grids a simple reality



Why Open Source?

- Best way to **own** a middleware product
 - No per-CPU license penalties
 - Free to try and free to use
- Developers friendly
 - Feel like a part of the team
- LGPL Business friendly
 - JBoss, Spring, Mule, *etc.* successes
 - Selling services vs. licenses



What is AOP?

- Encapsulating cross-cutting concern that cannot be expressed in OO language
- Mature technology with many implementations:
 - JBoss AOP, AspectJ, Spring AOP
- No standards yet – but well compatible
- Perfect match for Java 5 annotations



Why AOP?

- Grid enabling Java application = grid enabling a Java method call
 - AOP is ideal for cross-cutting methods
- Grid enabling is a cross-cutting concern
 - Application should functionally work the same with or without grid
 - Grid enabling can be added independently
- Transparent grid enabling



Problems with existing Grid Computing products

- Hard to use
- Hard to learn
- Not Java friendly
- Not enterprise Java friendly
- Geared towards academics *vs.* commercial world



Live coding demo

- Windows XP, Eclipse, Java 5, GridGain
- We will grid enable the following code:

```
public class HelloWorldExample {  
    public static void main(String[] args) {  
        sayIt("Hello World!");  
    }  
  
    public static void sayIt(String msg) {  
        System.out.println(msg);  
    }  
}
```



Conclusion

- Grid app in less than 15 minutes from scratch
 - Everything you need – nothing you don't
- Transparent grid enabling
 - Annotation based AOP
- Transparent deployment
 - Work as if you develop local non-distributed app
- Grid logic is separated from business logic
 - Clear separation of concerns
- **Fun, simple and productive :-)**



Thanks for your time!

Nikita Ivanov: nivanov@gridgain.com

GridGain: www.gridgain.org