Pluggable Grid Aspects for Scientific Applications

Universidade do Minho – Departamento de Informática

AspectGrid

Non-Invasive Gridification through an Aspect-Oriented Approach

Edgar Sousa

edgar@di.uminho.pt

Rui C. Gonçalves, Diogo T. Neves, João L. Sobral

{rgoncalves, dneves, jls}@di.uminho.pt

Outline

- Motivation
- Gridification
- Aspect Oriented Programming
- AspectGrid Framework
- Evaluation
- Future Work
- Conclusion

Motivation

- Gridify existing scientific codes
  - Gridification: make an application runnable (efficiently) on a Grid environment
- Current approaches limitations:
  - Impose unnecessary burden
  - Trade-off: non-invasive vs fine grained
  - Lack of specific support for multicores
  - Discourage application-specific enhancements

Some Gridification Issues

- Algorithm parallelisation
  - Exploit availability of resources
- Distributed data (non)awareness
- Application-level fault tolerance
- Application interaction/composition
- Deployment

Gridification Taxonomy

	Coarse-Grained	**Fine-Grained**
Invasive	ProActive, PAGIS, …	Ibis (Satin), …
Non-Invasive	GEMLA, GRASG, …	***AspectGrid***

Aspect Oriented Programming

- Modularises scattered and/or tangled functionality
- Based on the concept of *joinpoint*
  - “A well defined point in the program execution”
- Defines actions to perform before, after or instead the joinpoint

Aspect Oriented Programming

```
class Line {
  Point p1,p2;
  int length;
  public void setP1(Point p) {
    p1=p;
    length=calcLength();
  }
  public void setP2(Point p) {
    p2=p;
    length=calcLength();
  }
  //(…)
}
```

Aspect Oriented Programming

```
class Line {
   Point p1,p2;
   int length;
   public void setP1(Point p) {
      p1=p;
   }
   public void setP2(Point p) {
   //(…)
}
```

Aspect:

```
after(Line l) : call(* setP*(Point))
&& target(l){
      l.length=calcLength();
}
```

AspectGrid Framework Overview

- Gridification
  - Non-invasive
  - Fine-grained
- Lightweight
- Services
  - (Un)Pluggable
  - Composable
  - Extensible

AspectGrid

Pluggable Grid Aspects for Scientific Applications

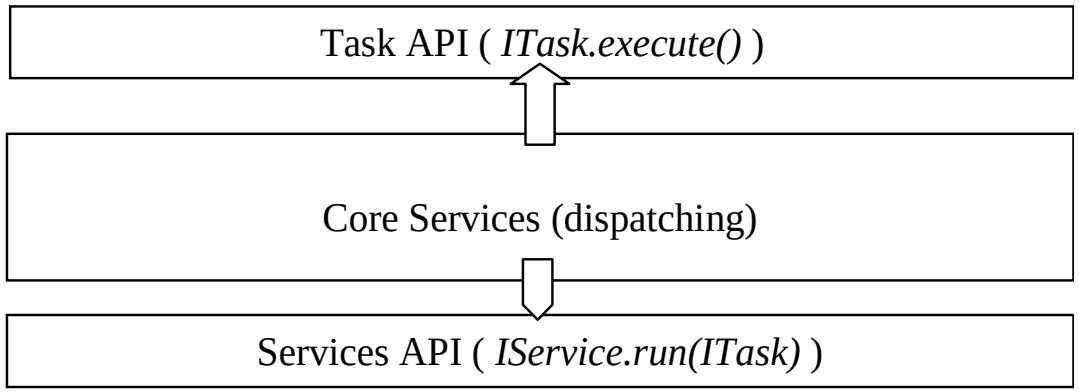
Universidade do Minho – Departamento de Informática

AspectGrid Framework

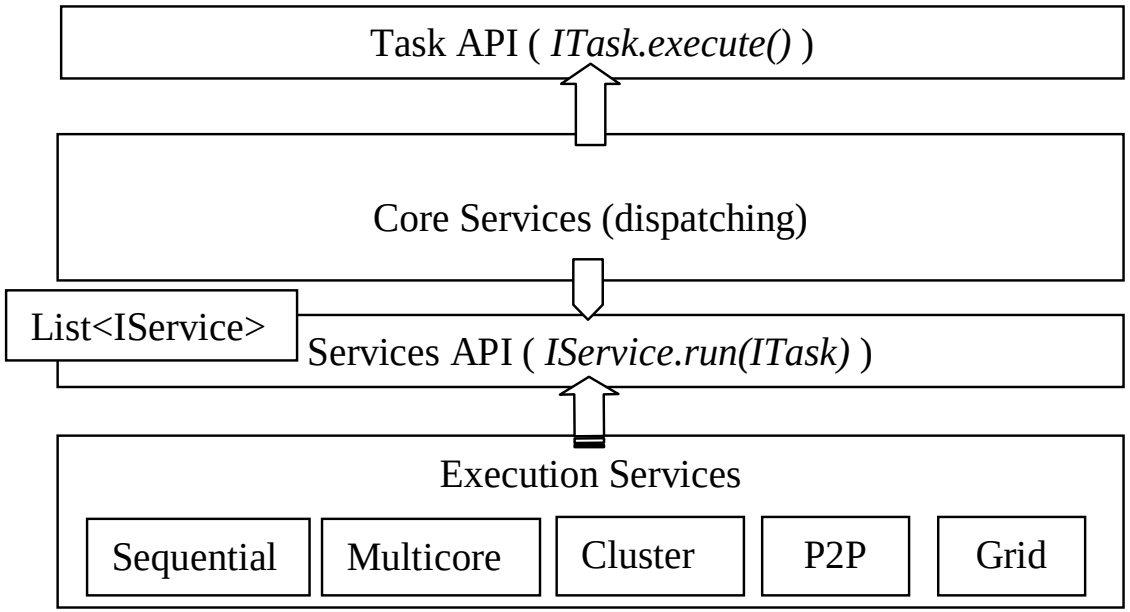
AspectGrid Framework

Core Services (dispatching)

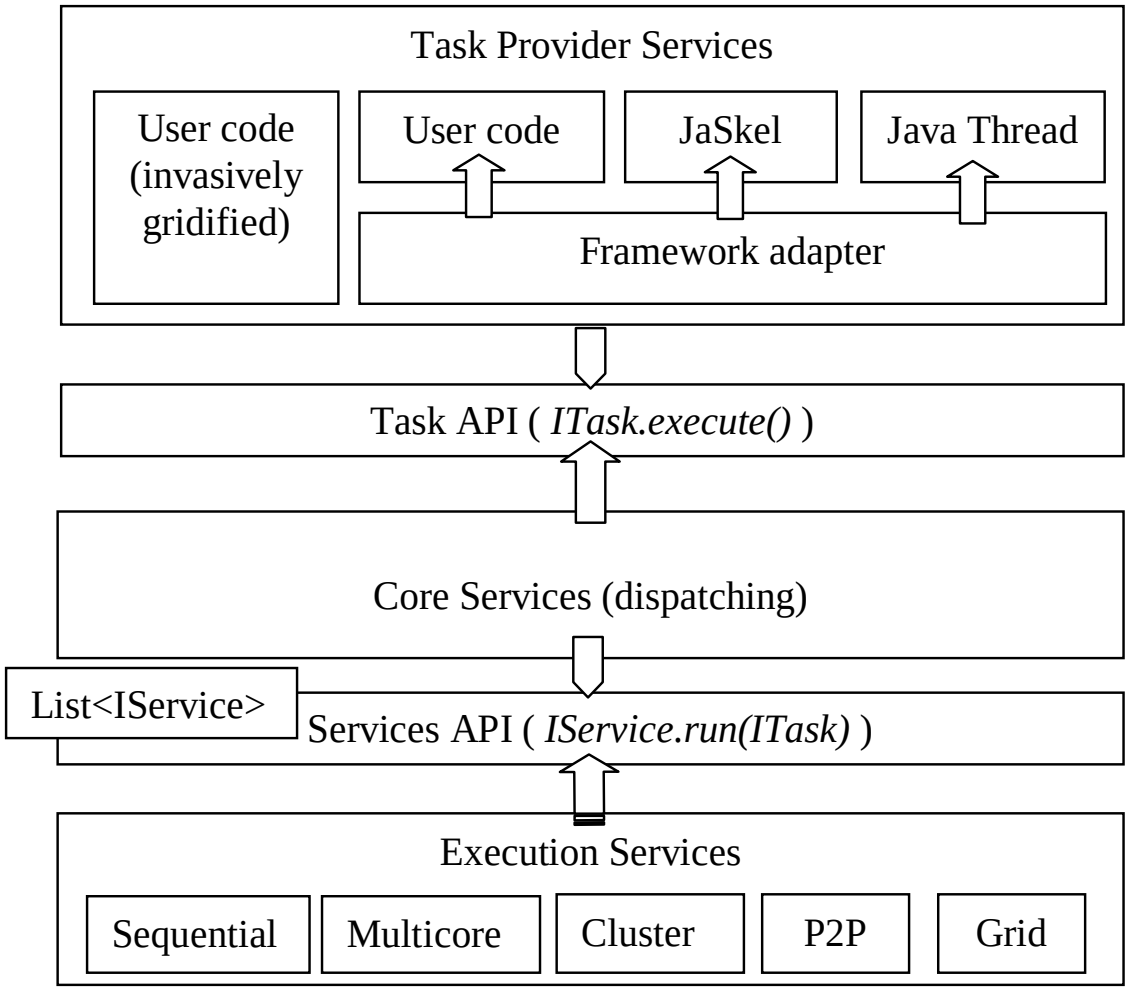
AspectGrid Framework



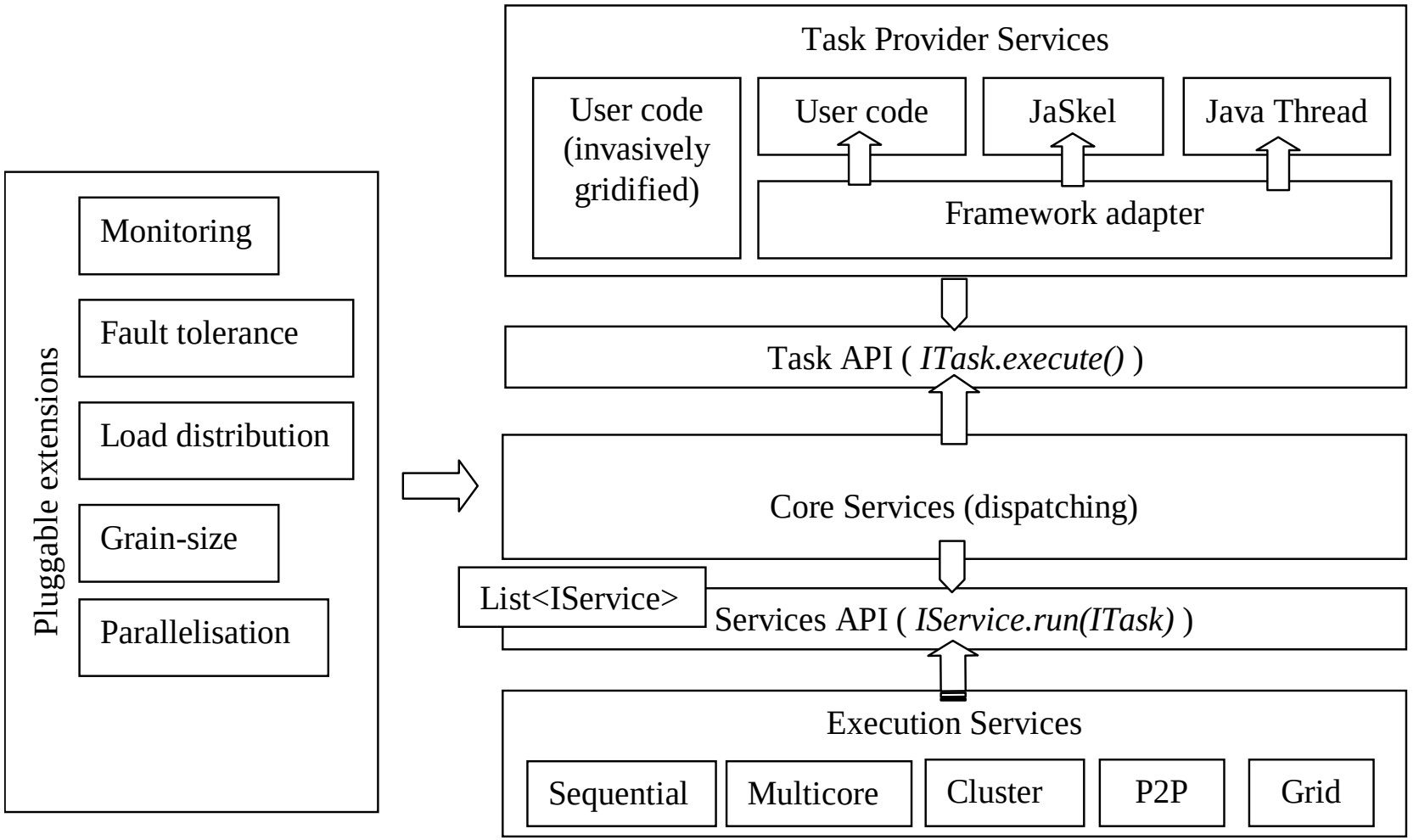
AspectGrid Framework



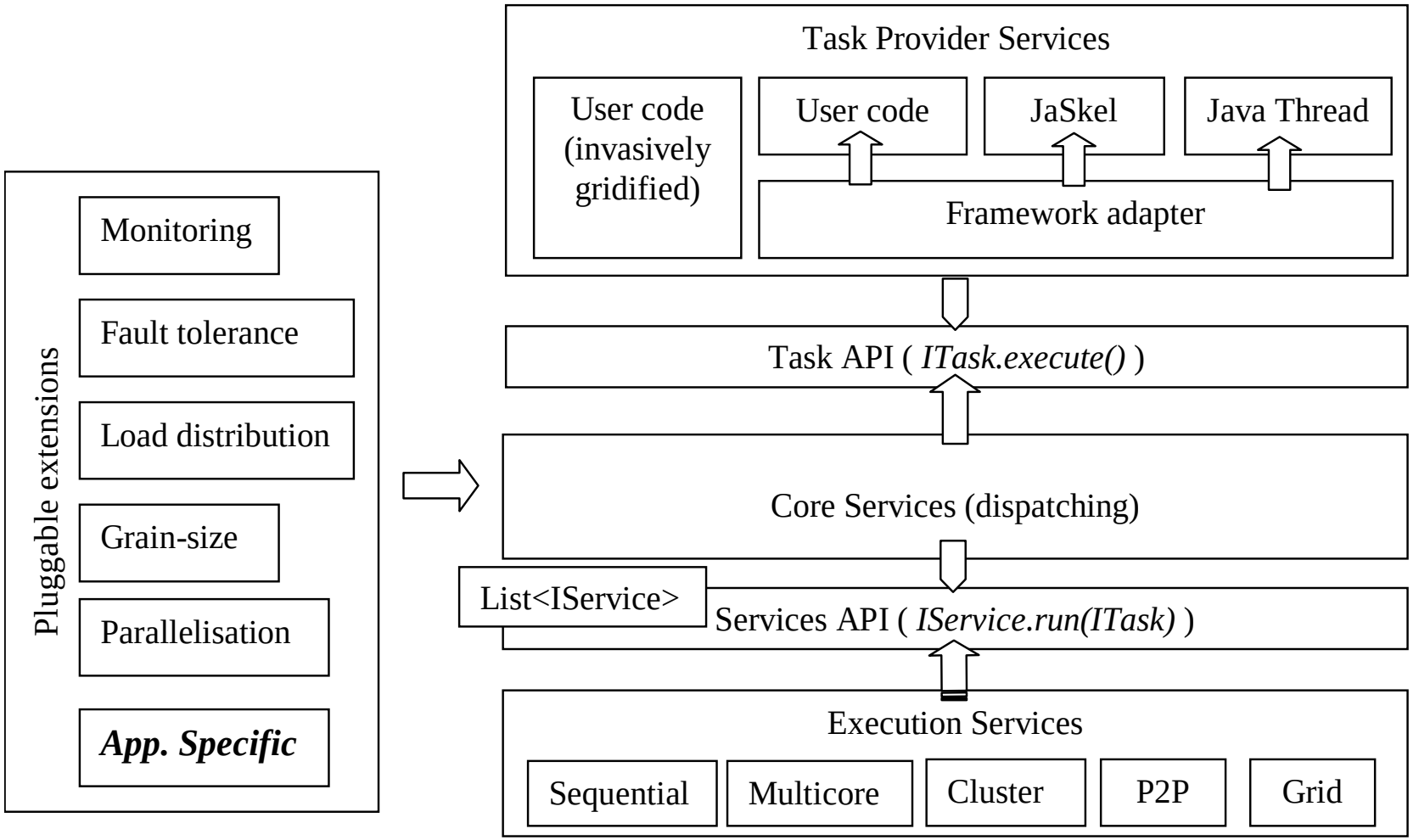
AspectGrid Framework



AspectGrid Framework



AspectGrid Framework



AspectGrid Framework

- Example 1: **Dispatcher**
  - On joinpoint *ITask.execute* dispatch *ITask* to an available resource

```
Object around(ITask task, Object i):
    call(* ITask.execute(..))&& … ;

  IService server = getService();
  …
  return server.run(task);
}
```

AspectGrid Framework

- Example 2: **Fault Tolerance**
  - Re-submit task for execution after a certain time-out

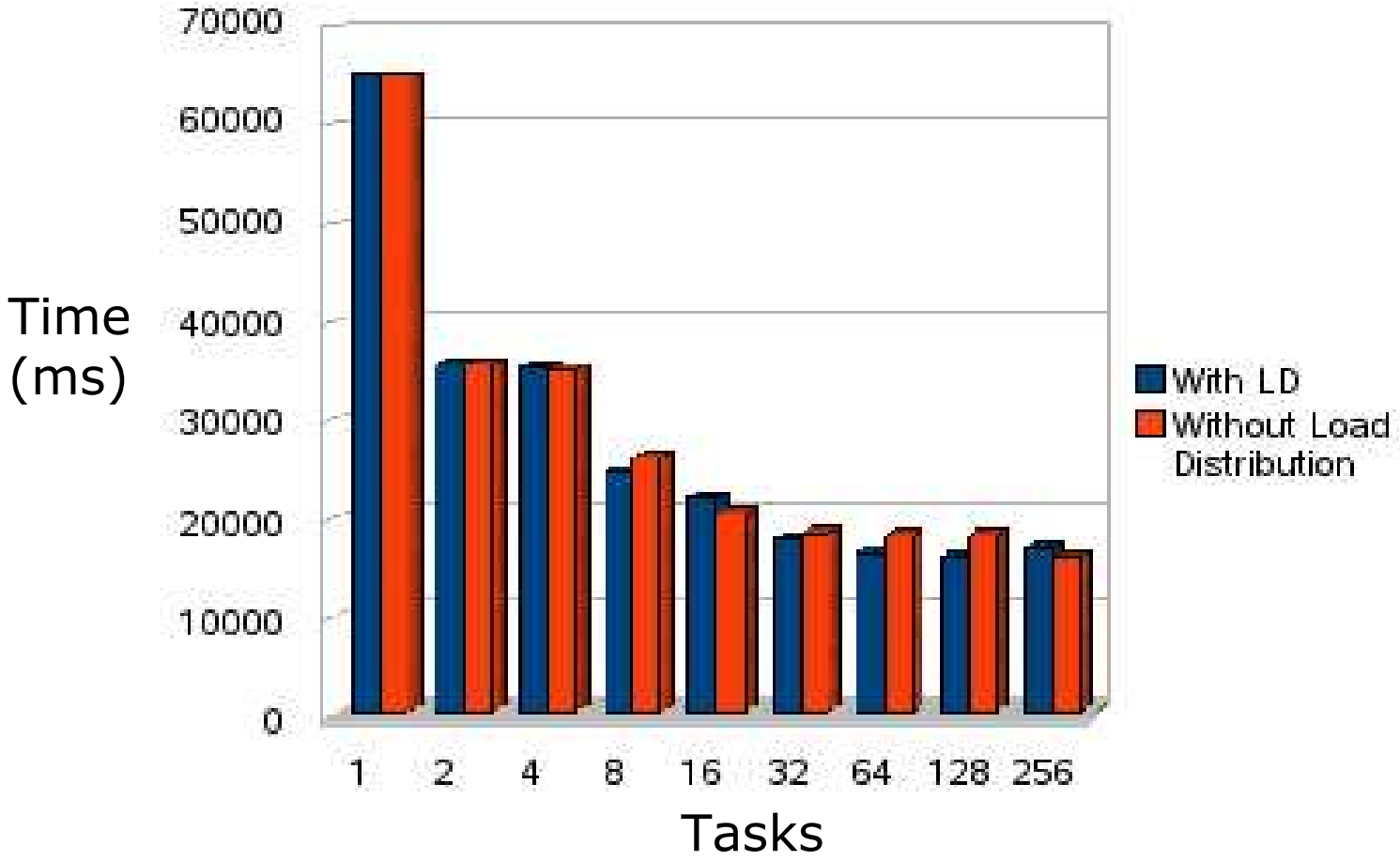
```
pointcut task_run(IService s, ITask t) :
  execution(* IService.run(ITask))
  && this(s) && args(t);

before(IService s, ITask t) : task_run(s,t) {
  … //start timeout mechanism
}
after(IService s, ITask t) : task_run(s,t) {
  … //task completed-> interrupt timer
}
```

Preliminary Evaluation

- Mandelbrot set
  - 2048x2048, 5000 iterations
  - SeARCH cluster http://www.di.uminho.pt/search
- Shared memory (dual Xeon, quad-core)
- Distributed memory (16 nodes)

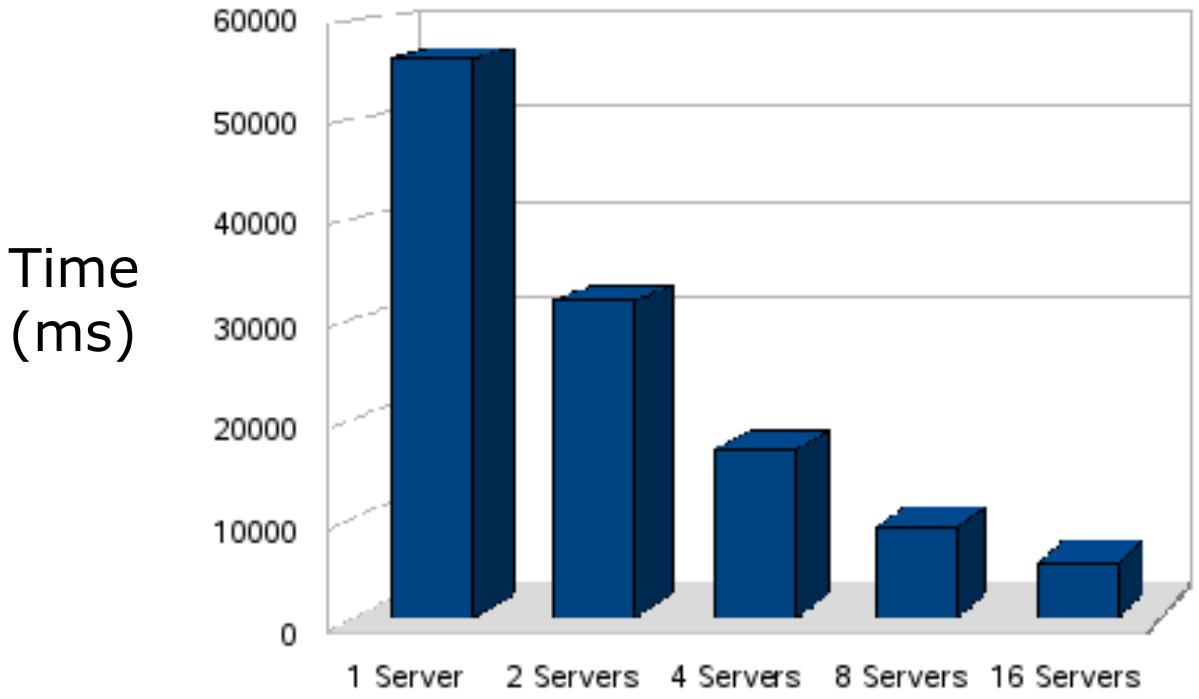
Preliminary Evaluation

- Execution on a quad-core machine



Preliminary Evaluation

- Execution on a cluster



Conclusion

- AspectGrid: lightweight framework to “gridify” scientific applications
  - Fine-grained, non-invasive gridification
  - Pluggable services
- More info:
  - AspectGrid project
    http://gec.di.uminho.pt/aspectgrid

Future work

- Fully implement all proposed modules (e.g., GLite binding)
- Support more parallelisation patterns
- Tool to assist *FrameworkAdapter*'s development
- Support for non-Java applications
- Distributed data (non)awareness