Components for grids
(an evolutionary process)

Marco Danelutto
Dept. of Computer Science Univ. of Pisa
&
Programming model Institute - CoreGRID

Outline

- Introduction
  - EchoGrid Roadmap
    - Key aspects in (grid) sw development
- Autonomic aspects in GCM
- Solving the Service / Component *dicotomia*
- Perspectives

Introduction

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- For a given problem or applications, *specification of the orchestration should be done using high-level abstractions* having the following properties: intuitive so that *non-expert programmers can use them*, generic to handle a large spectrum of applications and *parallelism should be implicit and fully hidden to the programmers*. Moreover, to cope with the large scale and unreliable dimension of the grid, programming languages, providing these high-level abstractions, have to be *associated with a distributed execution model to avoid any bottleneck*.

Roadmap, page 6

Introduction

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

If we could efficiently handle the predicted complexity derived from the large-scale interactions between future Grid applications, we could conclude to the following visionary ideas that may be proved useful as a guide in the technological advance of this area:

1. Produce component model to support *autonomic software development* so as to improve adaptation and reusability at a high level.
2. Develop a unified component model, that will fit the needs of future industrial use cases as well as those of today's Grid applications.
3. Clearer *separation of concerns regarding the functional and non-functional requirements* to ease the maintenance and the flexibility.
4. Standardize software certification models (such as the testing model for Grid software) at the component model level in order to improve software quality assurance.
5. The construction of reliable and scalable software systems require better behavioural guarantees. Elements towards this goal include: *more expressive specification formalisms, development environment providing "correct by construction" code, dynamic adaptation techniques*, etc.
6. Ensure QoS at the component level which will allow component applications to provide support SLA requirements.

Roadmap, pp 26

Key aspects in sw development

- Incremental design
  - separate test & debugging (functional & non functional)
  - "composability" (syntactic & semantic)
  - fundamental in the past (Unix!)
- Fast prototyping
  - ability to move from concept to code (tools, tools, tools)
- Interoperability
  - syntactic (easy), semantic (!?!?!)

Key aspects in *grid* sw development

- Resource management
  - initial mapping/scheduling,
  - adaptivity (fault tolerance)
- Fault tolerance
  - checkpointing, ...
- Security
  - active, passive

Component models

Component models

- Component = unit of deployment
  - functionality encapsulated much better than objects
- Programs = component assemblies
  - usually (hopefully): tested, third party components
- Composite components = components
  - more and more abstraction levels supported

Components & Grids

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

Components & Grids

- Deployment, life cycle, etc.
  - managed by the framework/tools
- Non functional aspects in component (assembly) execution
  - not in charge of the application programmer !
- Interoperability
  - guaranteed with major grid frameworks (WS)

Components & Grids

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

desiderata...

- Deployment, life cycle, etc.
  - managed by the framework/tools
- Non functional aspects in component (assembly) execution
  - not in charge of the application programmer !
- Interoperability
  - guaranteed with major grid frameworks (WS)

Autonomic domains

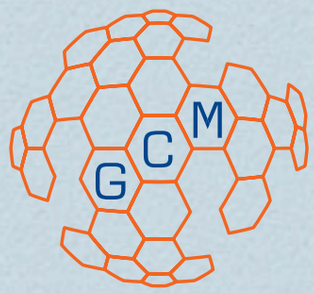
ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

Autonomic domains

- performance tuning
  - resource management, parallelism degree management, subcomputation rearranging, ...
- fault tolerance
  - checkpoint and restart on failure sympthoms, management of distributed checkpoints, ...
- security
  - security channel proxying, alternative communication media handling

GCM

- Hierarchical component composition
- Collective + data/stream ports
- Autonomic management of notable composite components
- XML based ADL
- Reference implementation in GridCOMP
  - Fractal based, in ProActive



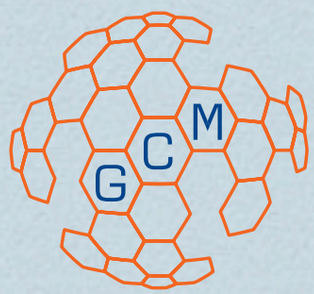
GCM

the Grid Component Model by CoreGRID

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- Hierarchical component composition
- Collective + data/stream ports
- Autonomic management of notable composite components
- XML based ADL
- Reference implementation in GridCOMP
  - Fractal based, in ProActive

GridCOMP
Effective Components for the Grids



GCM autonomic managers

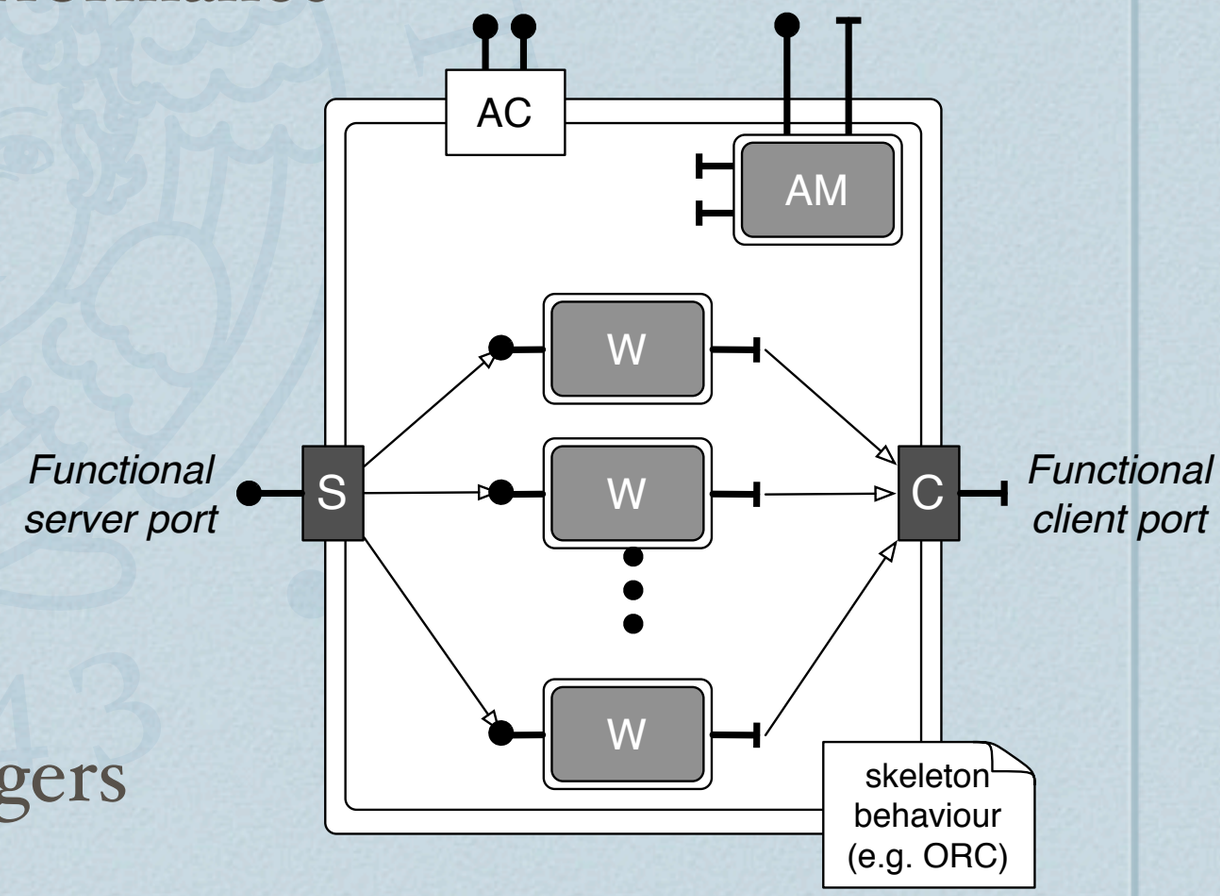
ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- Combining skeleton technology with autonomic management of performance
- Behavioural skeletons (BS)
- In GridCOMP/CoreGRID:
  - functional replication BS (task farm, data parallel)
  - rule based autonomic managers

GCM autonomic managers

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- Combining skeleton technology with autonomic management of performance
- Behavioural skeletons (BS)
- In GridCOMP/CoreGRID:
  - functional replication BS (task farm, data parallel)
  - rule based autonomic managers



Sample autonomic manager: performance

- Master/worker implementation of embarrassingly parallel computations
  - performance model
  - policies
  - autonomic cycle applying policies:
    monitor ⇒ *analyse* ⇒ *plan* ⇒ *execute*

$\max\{T_S, T_W/N_W, T_C\}$

P1:: if (Ts > max{...} && Ts > UserContract) then addWorker

P2::if(Ts <UserContract) then removeWorker

P3:: ...

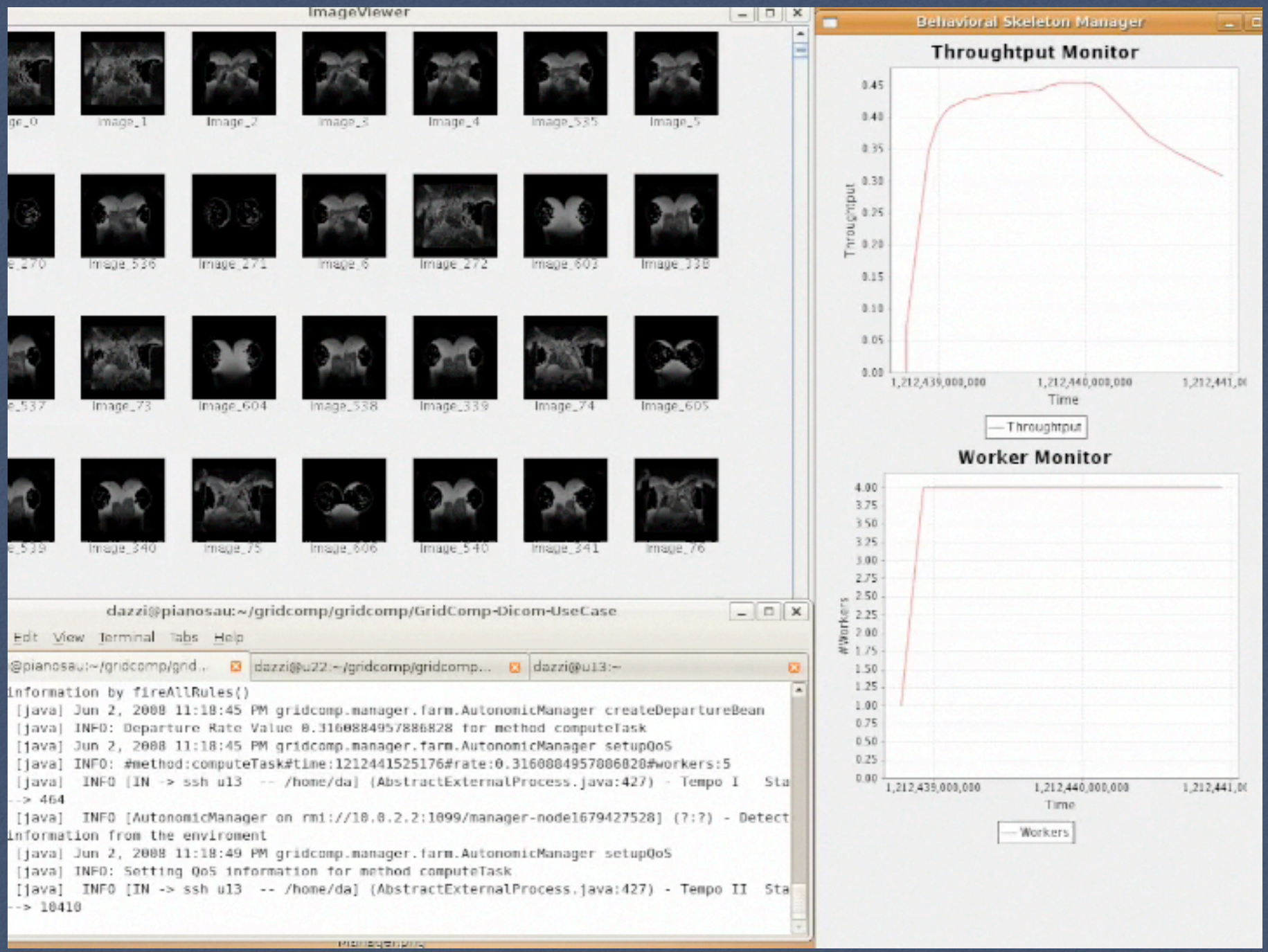
Sample autonomic manager: security

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

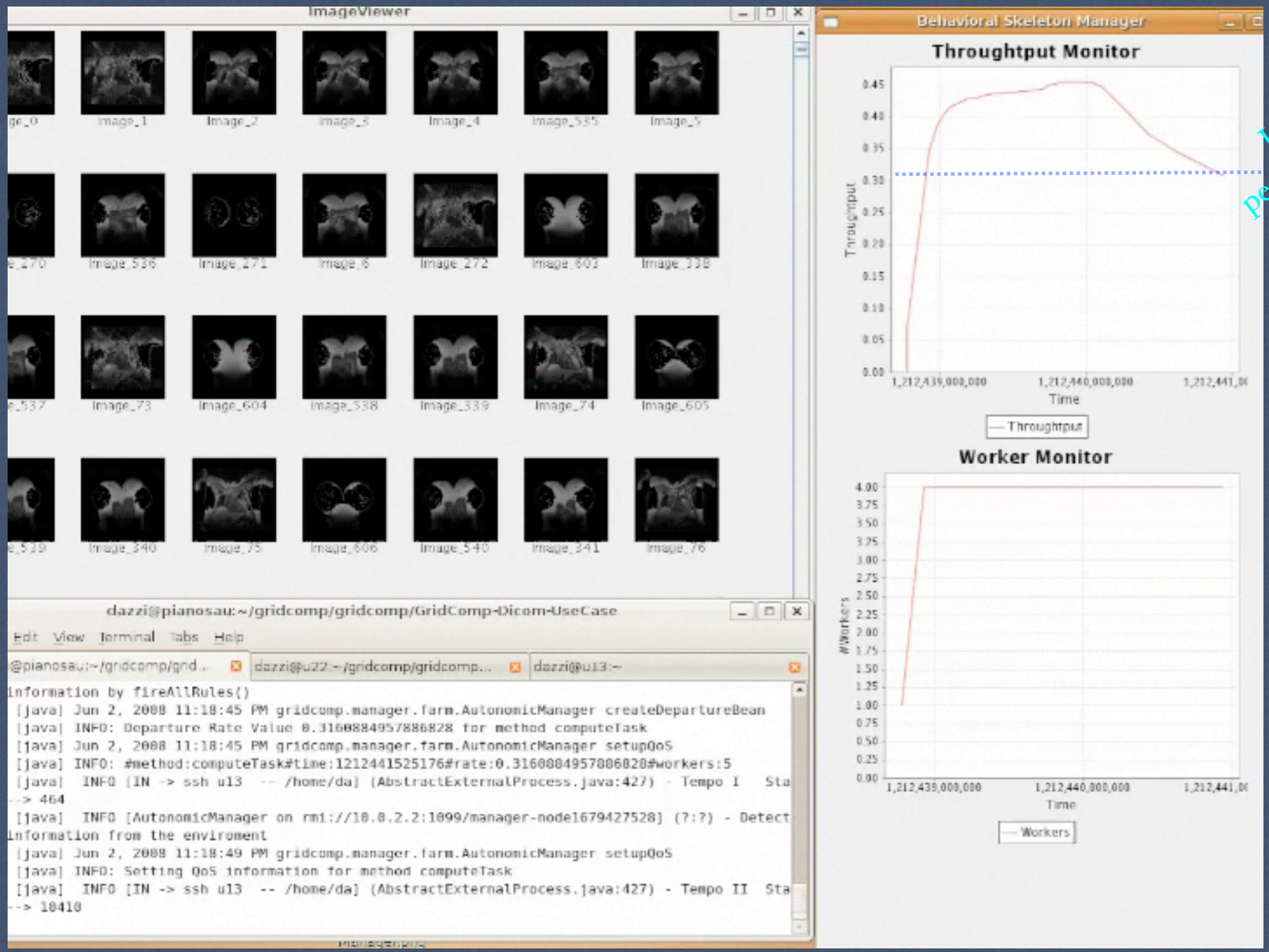
- Use/Provide component interaction
- Deployment info (with secure/insecure link taggings)
- Pre-defined proxy components (cypher, de-cypher)
  - policies
  - autonomic cycle applying policies:
    monitor ⇒ *analyse* ⇒ *plan* ⇒ *execute*

```
P1:: if (unsec(src,dst))
       then addProxyies
P2:: ...
```

Sample run

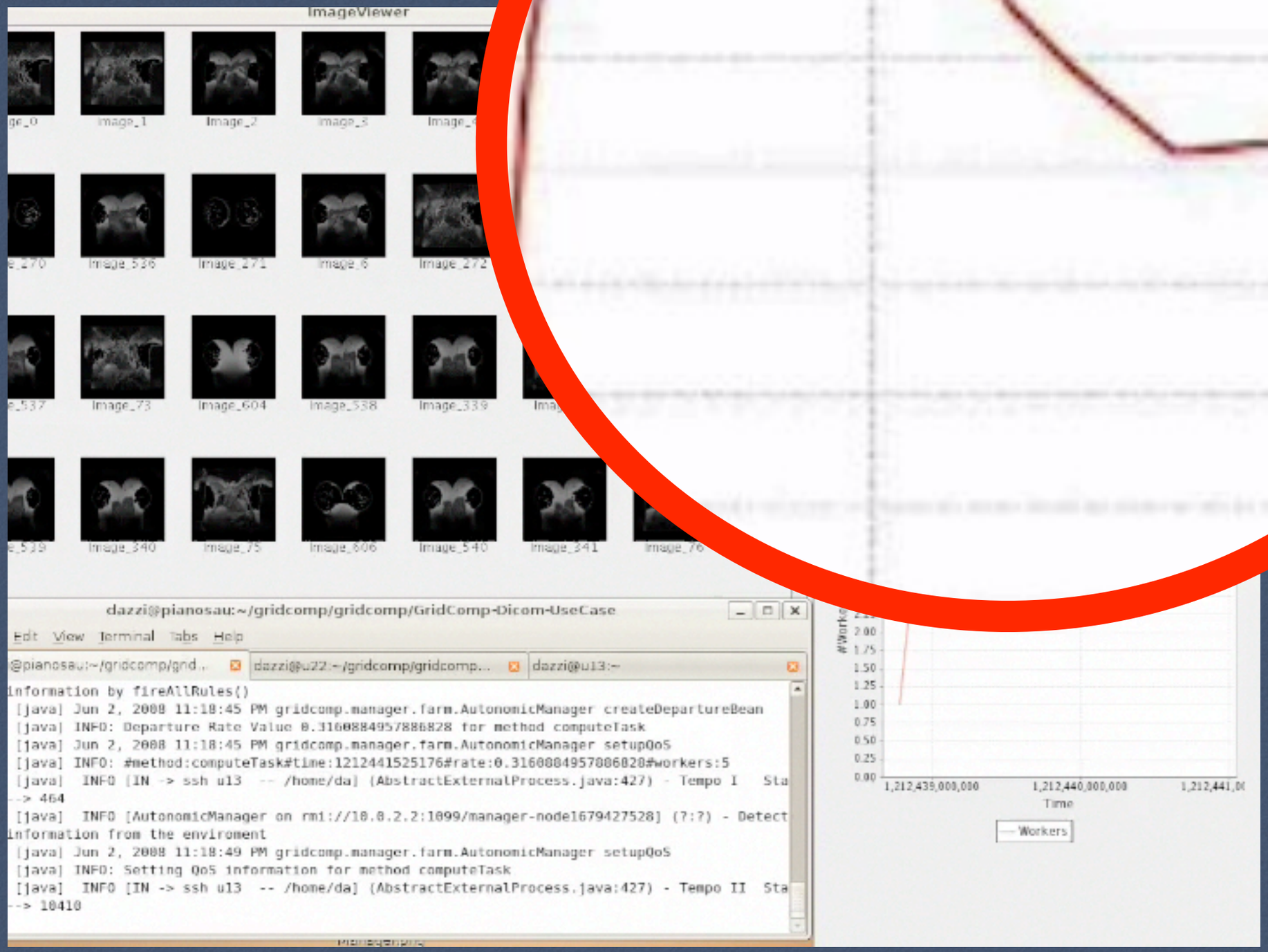

ImageViewer
dazzi@pianosau:~/gridcomp/gridcomp/GridComp-Dicom-UseCase
Edit View Terminal Tabs Help
[java] Jun 2, 2008 11:18:45 PM gridcomp.manager.farm.AutonomicManager createDepartureBean
[java] INFO: Departure Rate Value 0.3160884957886828 for method computeTask
[java] Jun 2, 2008 11:18:45 PM gridcomp.manager.farm.AutonomicManager setupQoS
[java] INFO: #method:computeTask#time:1212441525176#rate:0.3160884957886828#workers:5
--> 464
[java] INFO [AutonomicManager on rmi://10.0.2.2:1099/manager-node1679427528] (?:?) - Detect
information from the enviroment
[java] Jun 2, 2008 11:18:49 PM gridcomp.manager.farm.AutonomicManager setupQoS
[java] INFO: Setting QoS information for method computeTask
--> 10410
Behavioral Skeleton Manager
Throughtput Monitor
Throughput
Time
Throughput
Worker Monitor
#Workers
Time
Workers

Sample run



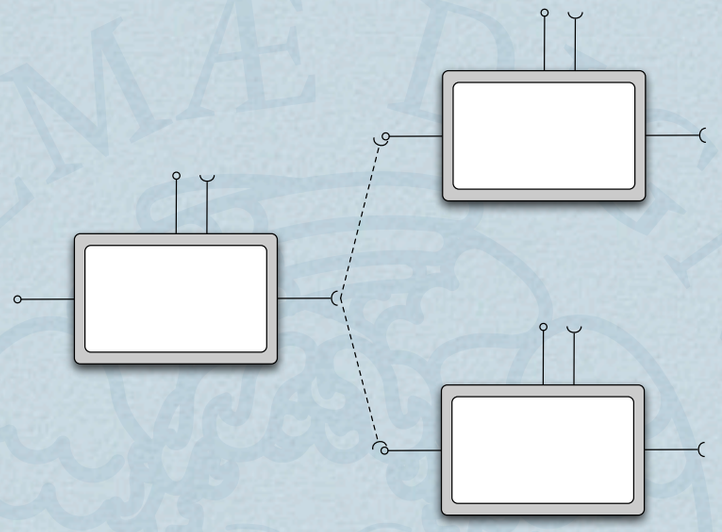
User supplied performance contract

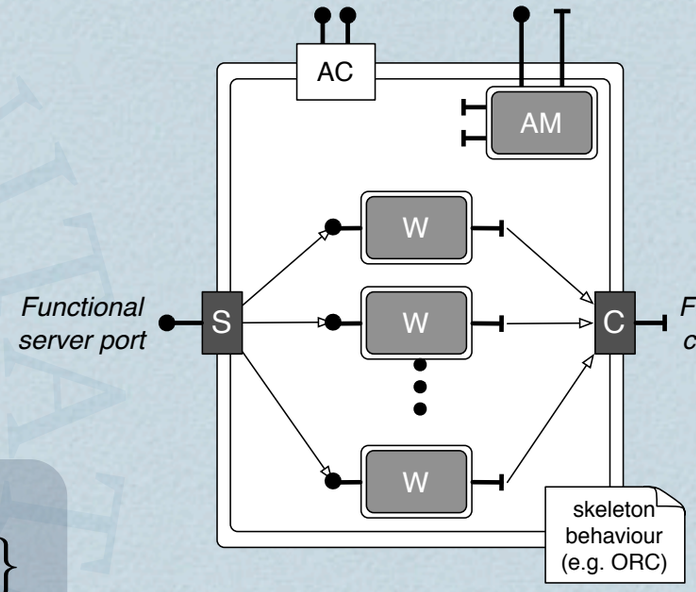
Sample run


ImageViewer
dazzi@pianosau:~/gridcomp/gridcomp/GridComp-Dicom-UseCase
Edit View Terminal Tabs Help
dazzi@u22:~/gridcomp/gridcomp...
dazzi@u13:~
information by fireAllRules()
[java] Jun 2, 2008 11:18:45 PM gridcomp.manager.farm.AutonomicManager createDepartureBean
[java] INFO: Departure Rate Value 0.3160884957886828 for method computeTask
[java] Jun 2, 2008 11:18:45 PM gridcomp.manager.farm.AutonomicManager setupQoS
[java] INFO: #method:computeTask#time:1212441525176#rate:0.3160884957886828#workers:5
[java] INFO [IN -> ssh u13 -- /home/da] (AbstractExternalProcess.java:427) - Tempo I Sta
--> 464
[java] INFO [AutonomicManager on rmi://10.0.2.2:1099/manager-node1679427528] (?:?) - Detect
information from the enviroment
[java] Jun 2, 2008 11:18:49 PM gridcomp.manager.farm.AutonomicManager setupQoS
[java] INFO: Setting QoS information for method computeTask
[java] INFO [IN -> ssh u13 -- /home/da] (AbstractExternalProcess.java:427) - Tempo II Sta
--> 10410
1,212,439,000,000
1,212,440,000,000
Time
Workers

This is :

- GCM
- Behavioural skeletons
- Business rule engine based autonomic manager
- Approximate performance models





$$T_s = max\{T_e, \frac{T_w}{n_w}, T_c\}$$

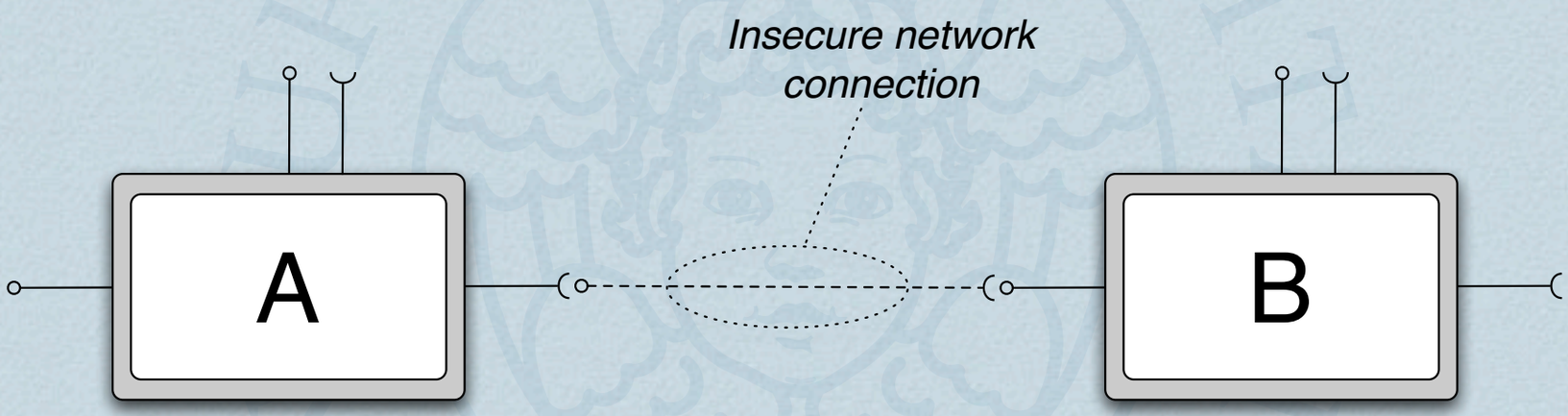
```
[methodMonitor="searchMatch"]
rule "CheckHigherBound"
        when
            $arrivalBean : PartitionSizeBean(value >=10)
        then
            $arrivalBean.fireOperation(ManagerOperation.ADD_EXECUTOR);
 end
[methodMonitor="getService"]
rule "CheckLowerBound"
        when
            $arrivalBean : PartitionSizeBean(value < 9)
        then
            $arrivalBean.fireOperation(ManagerOperation.REMOVE_EXECUTOR);
 end
```

Appl. programmers:

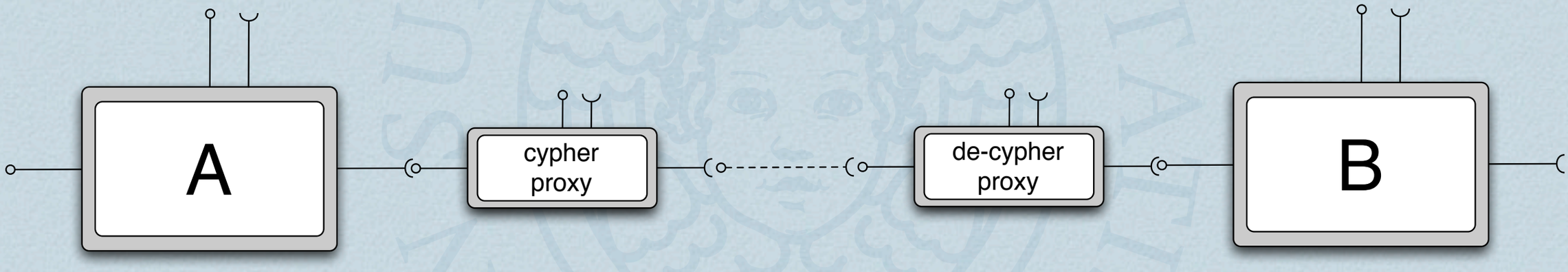
- Provide “worker” components
- Instantiate a task farm behavioural skeleton
  - through proper ADL (XML) file

And that’s it!

Security management



Security management

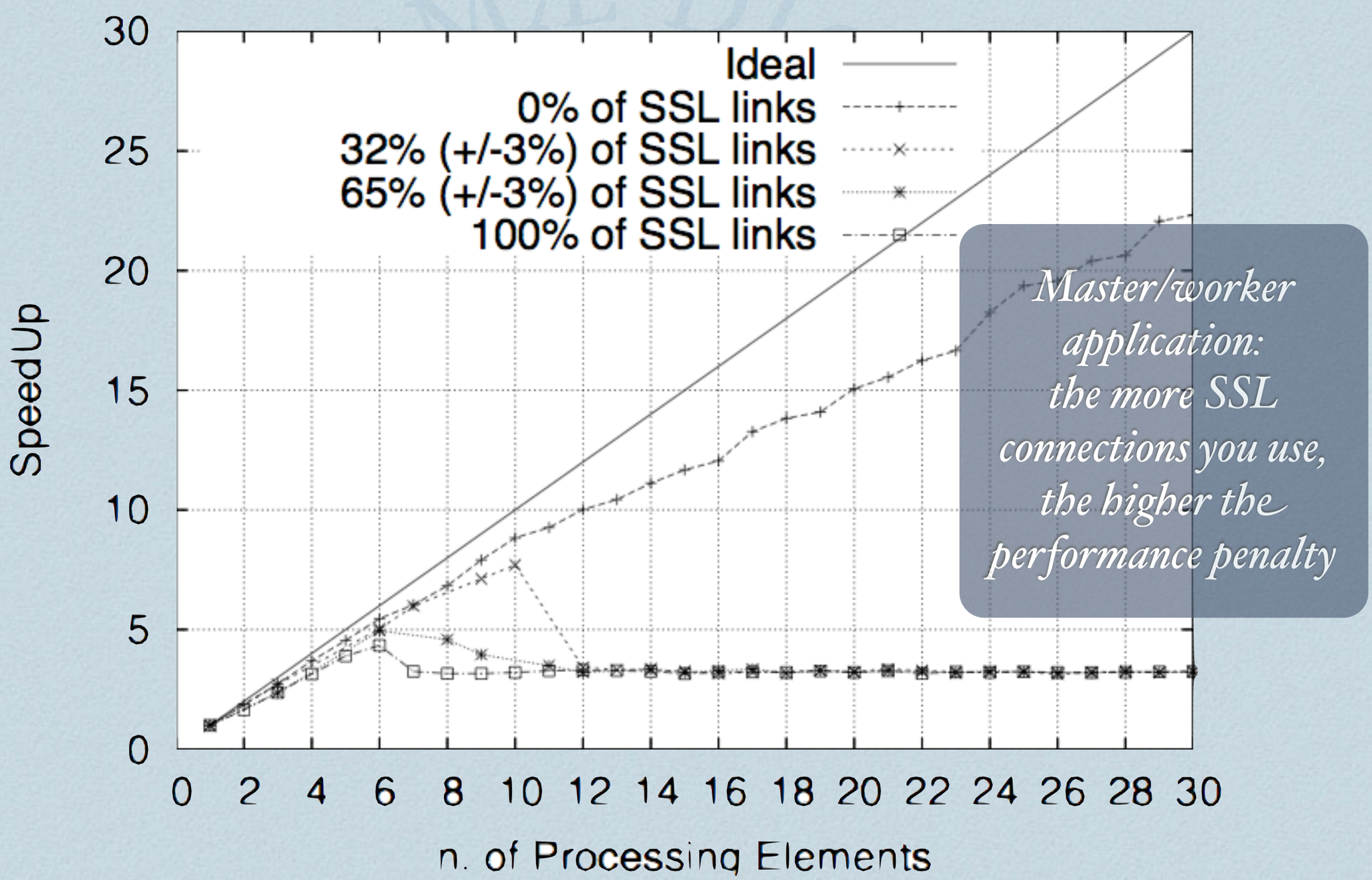


- Information about deployment is *transparent to the user*
- Pipelined proxies (if possible)

Performance impact...

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

Performance impact...



Master/worker application: the more SSL connections you use, the higher the performance penalty

Components vs. services ...

- services ≈ component ???
  - mostly **YES**, but:
    - no info of those needed to *move* services
    - service = component - use ports
- as a consequence:
  - ideal to sell *static* applications
  - if the case, *dynamic* stuff is programmed *ad hoc*

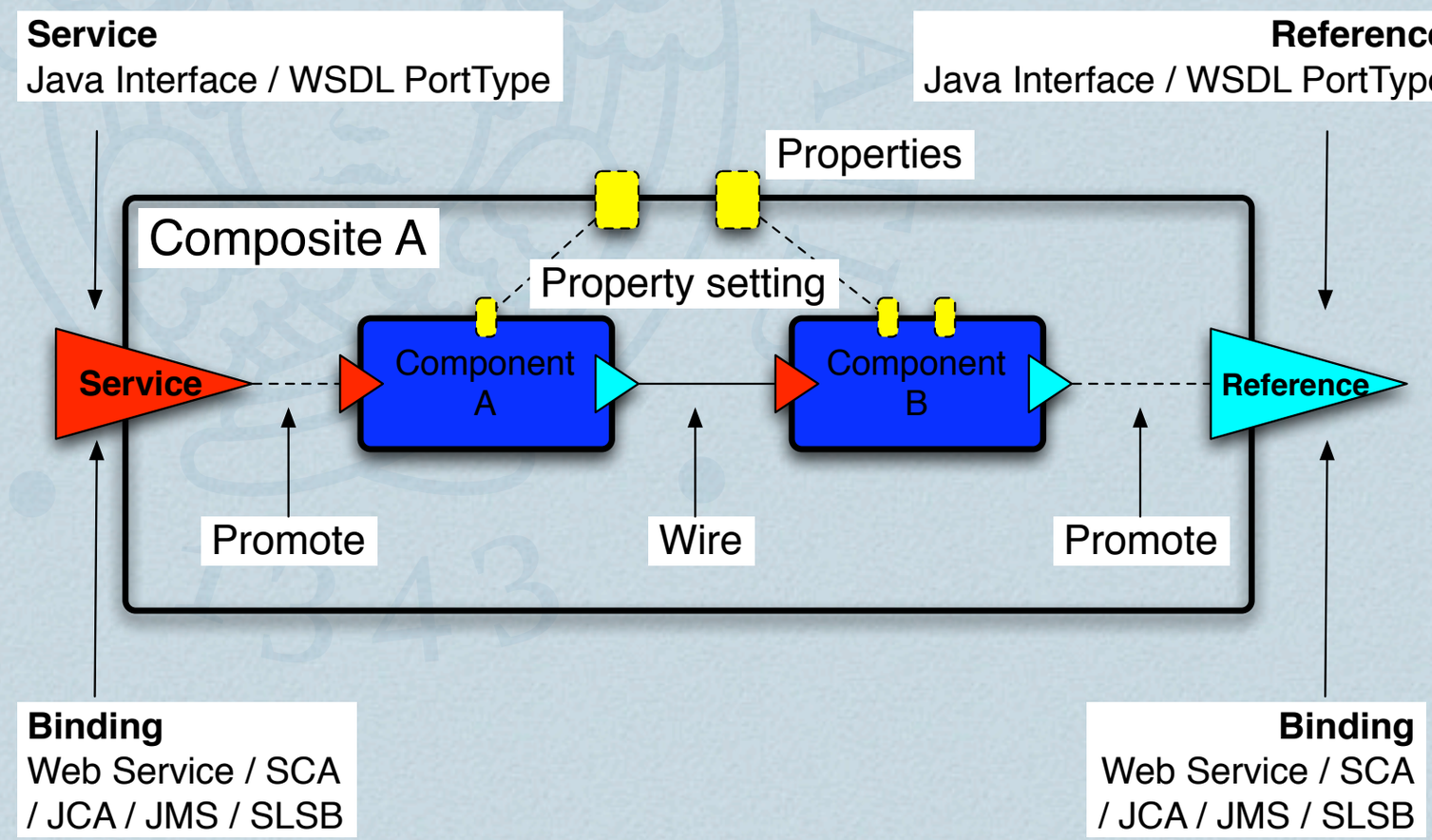
Services are moving ...

- SCA (Service Component Architecture)
  - composite components out of plain services
  - composites are plain components

Services are moving ...

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- SCA (Service Component Architecture)
  - composite components out of plain services
  - composites are plain components



GCM xx SCA

xx = vs. *or* xx = with?

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

GCM

- Hierarchical component composition
- Collective + data/stream ports
- Autonomic management of notable composite components
- XML based ADL
- Reference implementation in GridCOMP
  - Fractal based, in ProActive

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

GridCOMP

GCM xx SCA

xx = vs. *or* xx = with?

Already there, primitive

GCM

- Hierarchical component composition
- Collective + data/stream ports
- Autonomic management of notable composite components
- XML based ADL
- Reference implementation in GridCOMP
  - Fractal based, in ProActive

GCM xx SCA

xx = vs. *or* xx = with?

Already there, primitive

Can be implemented

GCM

- Hierarchical component composition
- Collective + data/stream ports
- Autonomic management of notable composite components
- XML based ADL
- Reference implementation in GridCOMP
  - Fractal based, in ProActive

GCM xx SCA

xx = vs. *or* xx = with?

Already there, primitive

Can be implemented

Experimented!

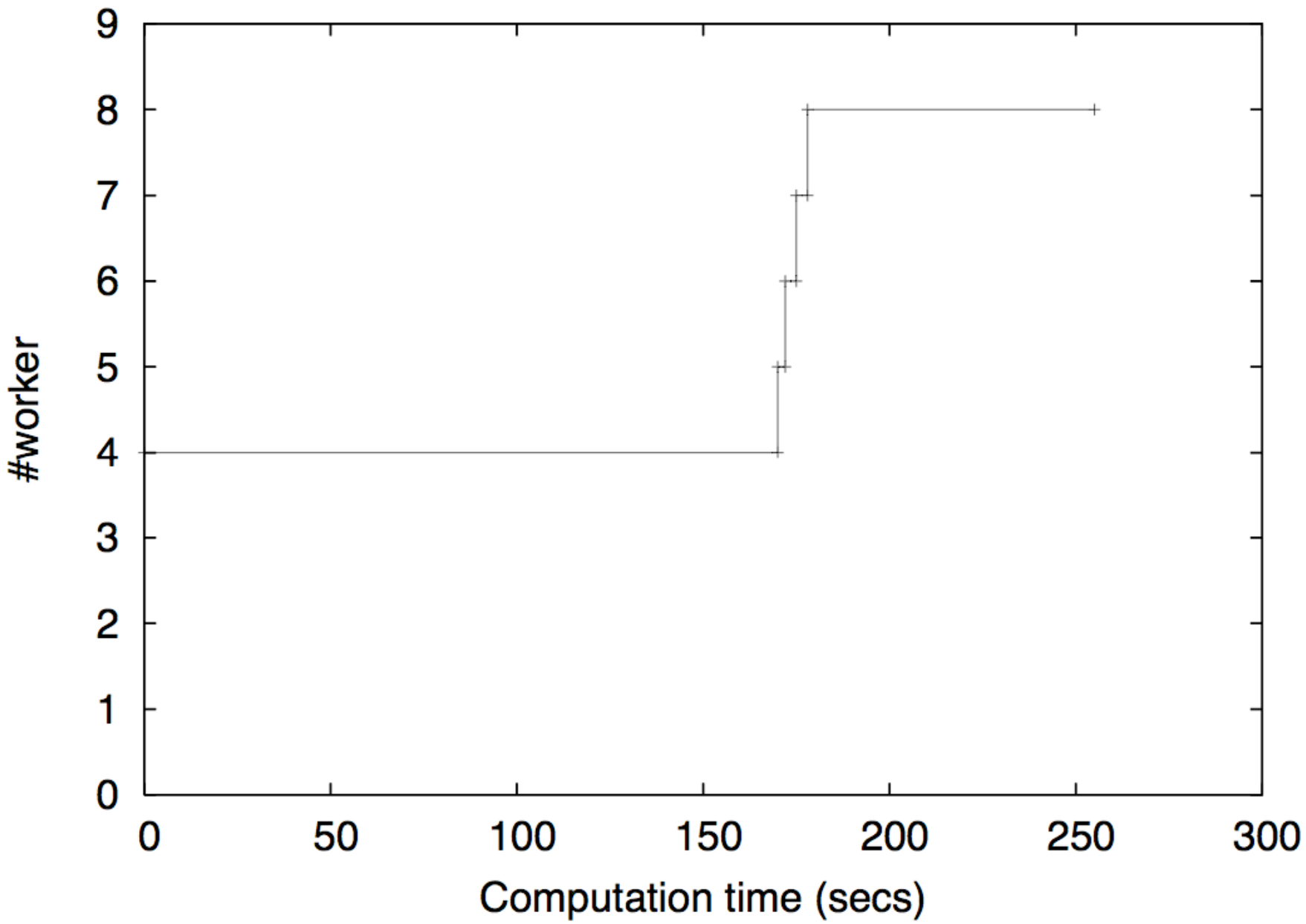
GCM

- Hierarchical component composition
- Collective + data/stream ports
- Autonomic management of notable composite components
- XML based ADL
- Reference implementation in GridCOMP
  - Fractal based, in ProActive

SCA task farm GCM behavioural skeleton

- Behavioural skeleton in SCA:
  - JBoss rule based manager + task farm skeleton
  - preliminary experimental results: *feasibility* & *scalability*
  - JBoss rule based manager moved in GridCOMP GCM reference implementation (ProActive/Fractal)
  - GCM task farm BS as a service:
    - generic, optimized, batch task processor (user defined tasks)

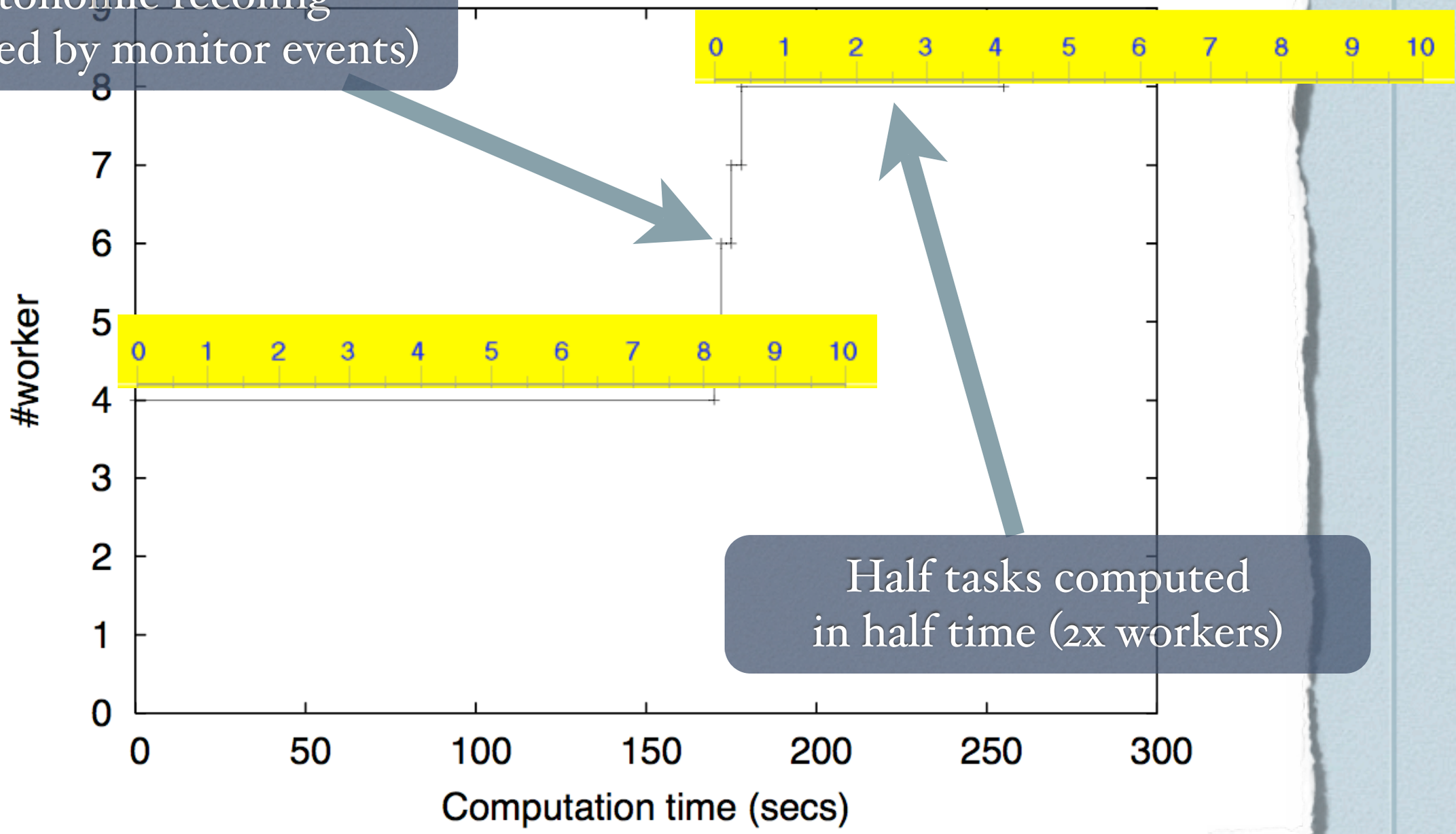
SCA GCM BS results



SCA GCM BS results

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

Autonomic reconfig
(triggered by monitor events)



Half tasks computed
in half time (2x workers)

Further, ongoing SCA/GCM activity

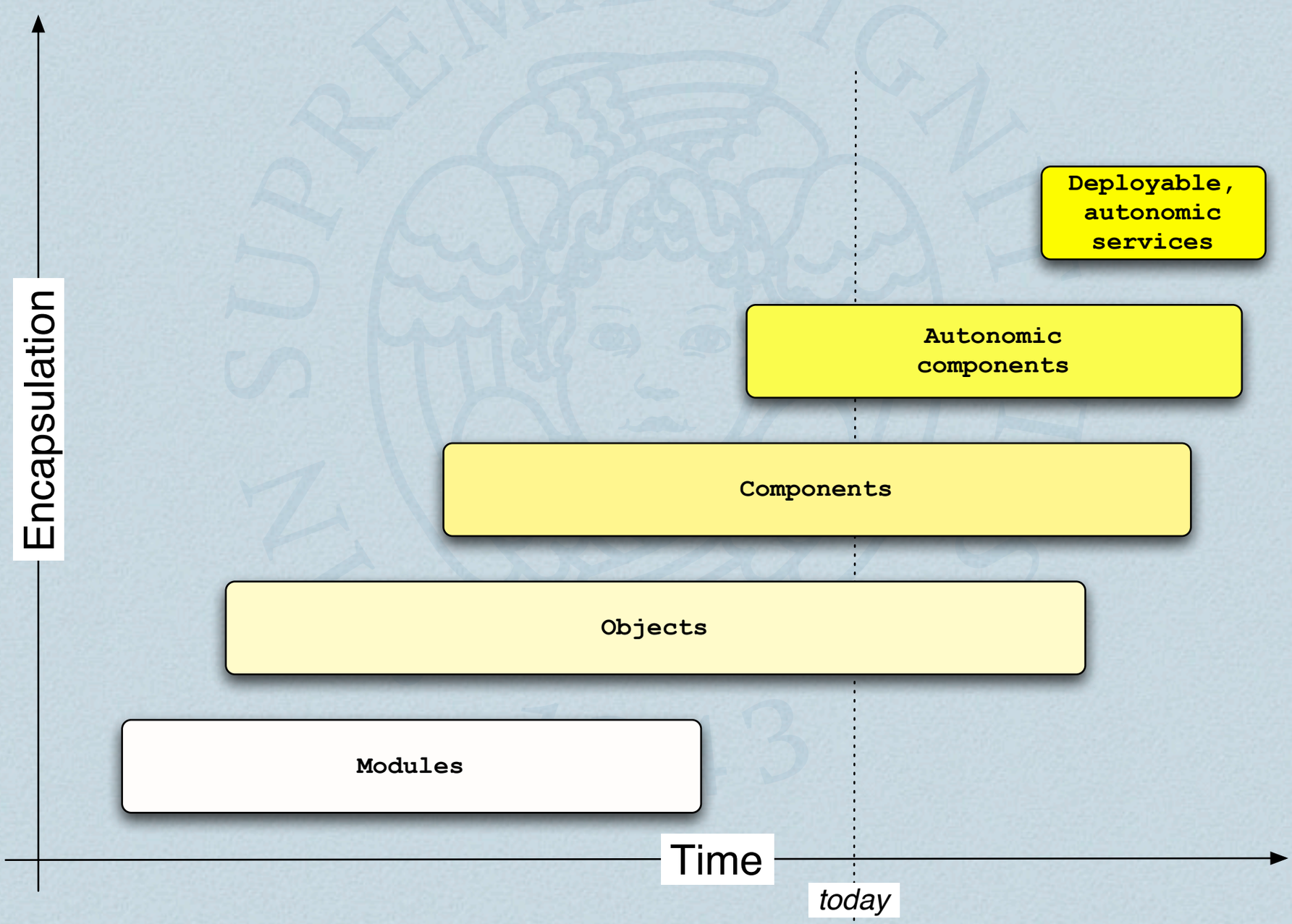
- providing collective *connector* components in SCA
  - collective out component (one to *N*, configurable distribution policy: scatter, {multi,broad,uni}cast)
  - collective in component (*N* to one, configurable gathering policy: gather, reduce, combine)
  - preliminary implementation ready, going to collect experimental results next month

Lessons learned

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

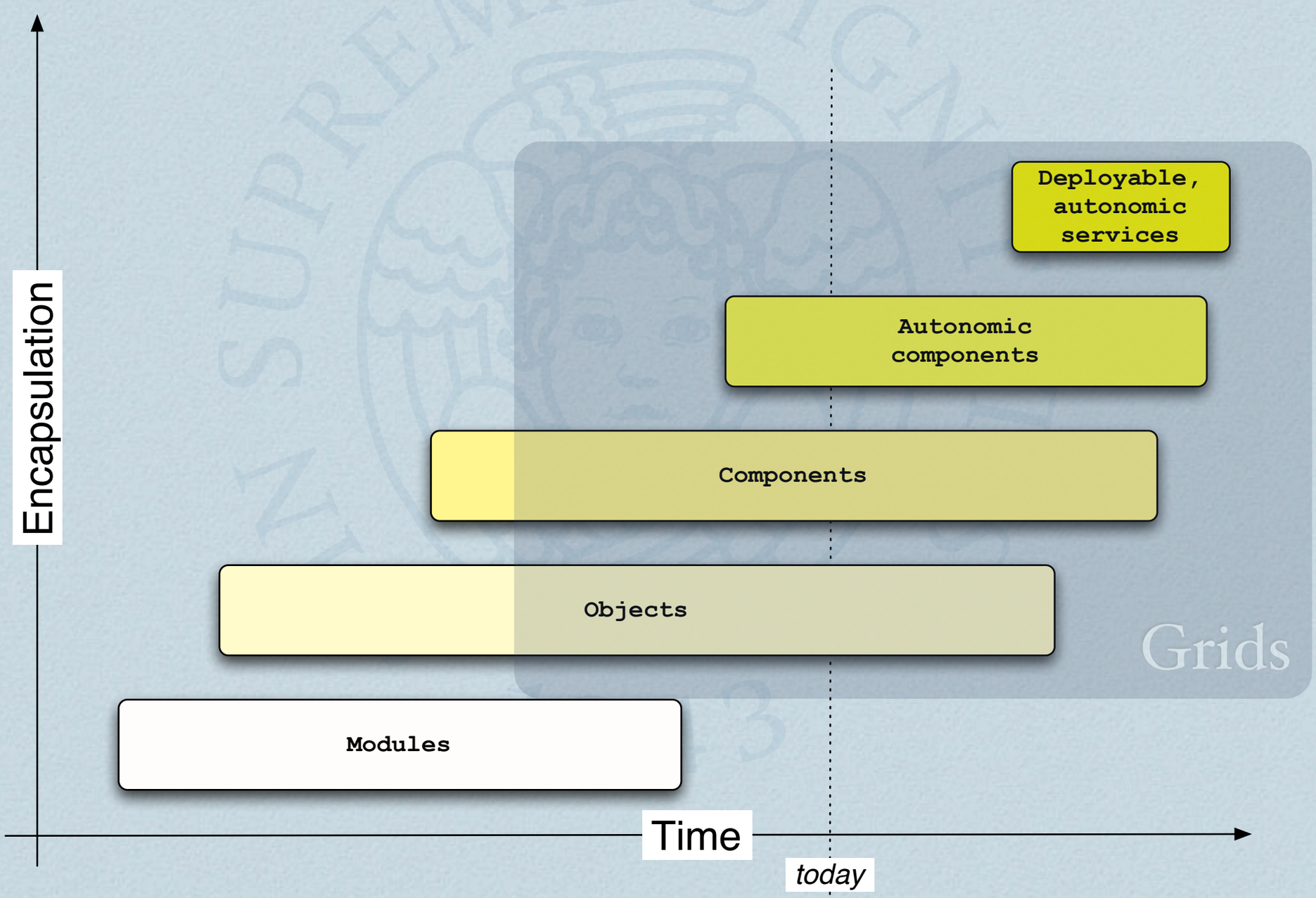
- Still poor support for *dynamic* component assemblies in SCA
  - 1.0 had something, looked like to improved in 1.1, almost disappeared in 1.2 ... (as perceived from the developer team)
- Service / component integration more or less perfect
- Support for a variety of host languages, bindings, etc.
- Open source project, with nice community behind, with recent official involvement of RedHat (pros&cons!)

Our "vision"


Encapsulation
Deployable, autonomic services
Autonomic components
Components
Objects
Modules
Time
today

Our “vision”

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID



Our vision (2)

- today:

algorithms
complexity concerns
reusability issues
object code portability
...

Our vision (2)

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- today:

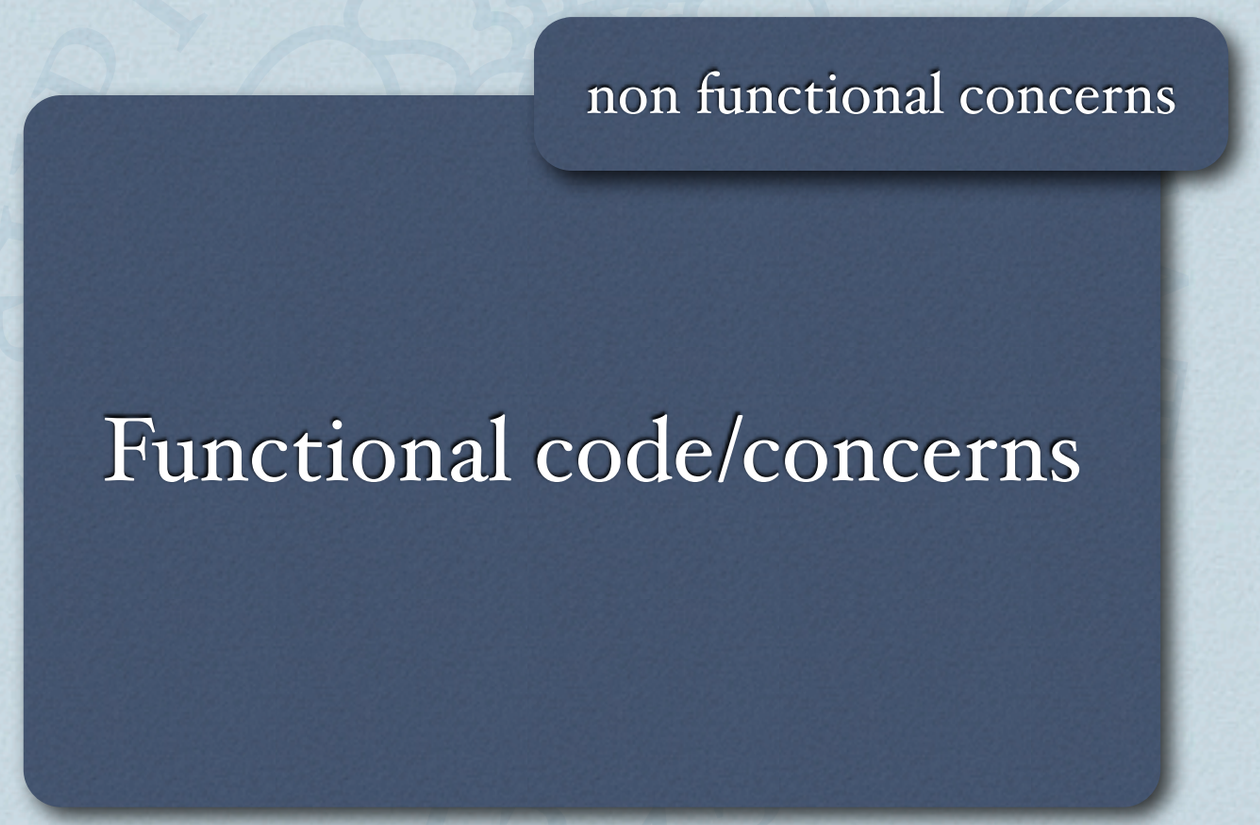
Functional code/concerns

Perfectly working code,
not so impressive global achievements

Our vision (2)

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- today:



Perfectly working code,
fairly better global achievements

Our vision (2)

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- today:

Perfectly working code,
fairly better global achievements

Our vision (2)

- tomorrow:

algorithms
complexity concerns
reusability issues
object code portability
...

Our vision (2)

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- tomorrow:

Non functional concerns, application orchestration, autonomic management

Perfectly working application schema

Our vision (2)

- tomorrow:

Non functional concerns, application orchestration, autonomic management

functional co-processor

Perfectly working application

Our vision (2)

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

- tomorrow:

Perfectly working application

EchoGRID roadmap

- Already identified key "components" in this vision
- *Innovative* programming models in the roadmap fit the vision
- Autonomic management & services look like kind of *first class citiziens* in the roadmap ...
  - *and this **must** be the focus*

Thank you for your attention