A RISC approach to the GRID

M. Danelutto

University of Pisa

IFIP 10.3 Nov 1°, 2003

Contents

- GRID: current status
- A RISC grid core
- Current experiments
- Conclusions

Contents

- *GRID: current status*
- A RISC grid core
- Current experiments
- Conclusions

GRID: current status
Contents

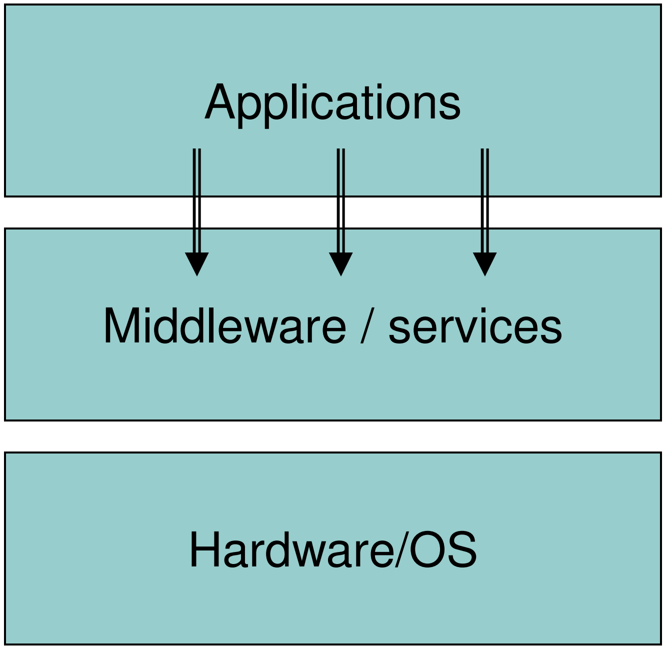
- Characterization of GRID
- Tools
- Abstract machine view
- Current, “GRID aware” applications

Characterization of GRIDs

- Target architectures for complex computations
  - Complex, multidisciplinary, multilanguage …
- Heterogeneous
  - HW & SW heterogeneous
- Dinamic
  - Instant (latency!) and long range (node up&down)
- Distributed
  - Geographic scale networks

Tools

- Middleware
  - Between hw & sw !
- Many flavours
  - One winner
- Complex functionalties
  - Scheduling, resource manag., data/code staging, monitoring, etc.
- All is in charge to the appl. programmer !



Abstract machine view

- Layer 0
  - OS services, common core (TCP, POSIX)
- Layer 1
  - Middleware (resource discovery/management, code/data staging, remote execution, security, monitoring)
- Layer 2
  - Programming environment (PSE, in some cases)
- Layer 3
  - Applications

GRID aware applications

- Case A
  - Embarassingly parallel computations (CONDOR like), no heterogenity, dinamicity
- Case B
  - Esplicitly needed resources (compiler, CPU power, …), hand made placement, no dinamicity (but “instant” one)
- Case C
  - Data intensive/data driven applications (no dinamicity, heterogenity)

GRID aware applications

- Case D
  - High level parallel code, high level requirement specs, automatical [illegible] discovery, automatic adap[illegible]a[illegible] restuc[illegible]ing [illegible]ic[illegible] lov[illegible]ring [illegible]

where ???

Contents

- GRID: current status
- *A RISC grid core*
- Current experiments
- Conclusions

RISC GRID core (RGC)

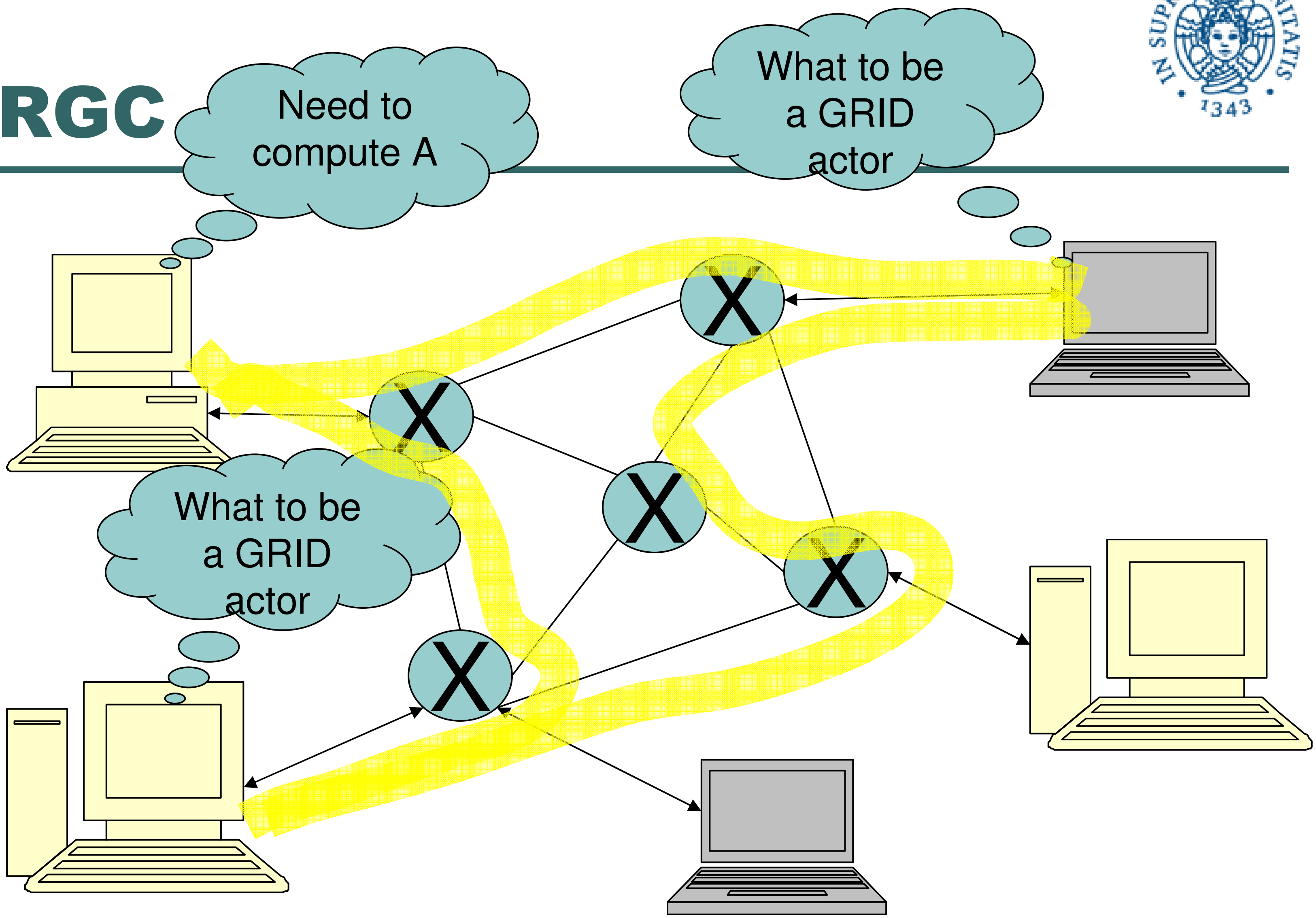
- Perspective
- Basic functionalities
- Implementation

RGC: perspective

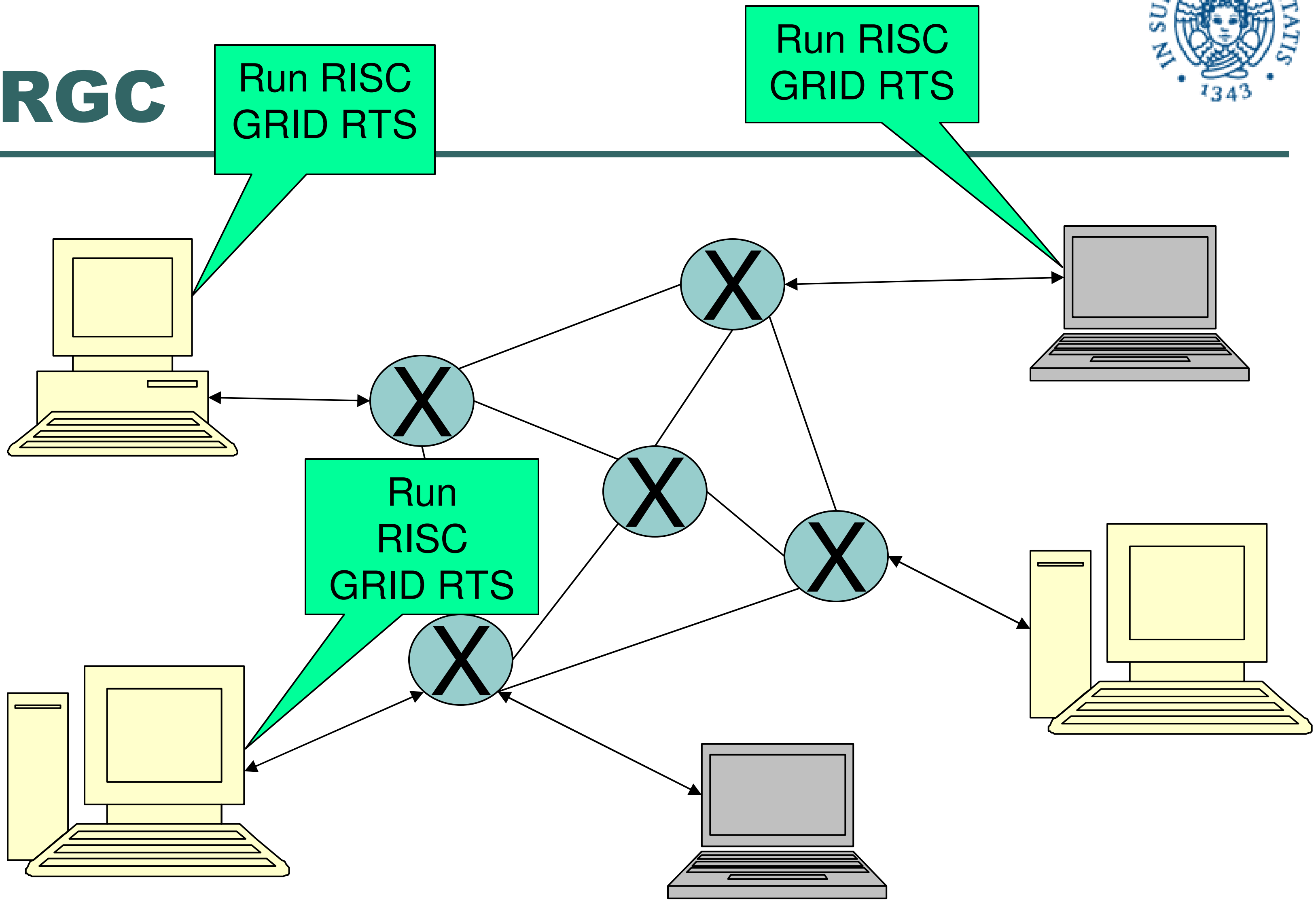
- GRID components actively participate to offer services
- Subject
  - Entity needing services → Entity providing services
- RGC basic service:

  → computing services

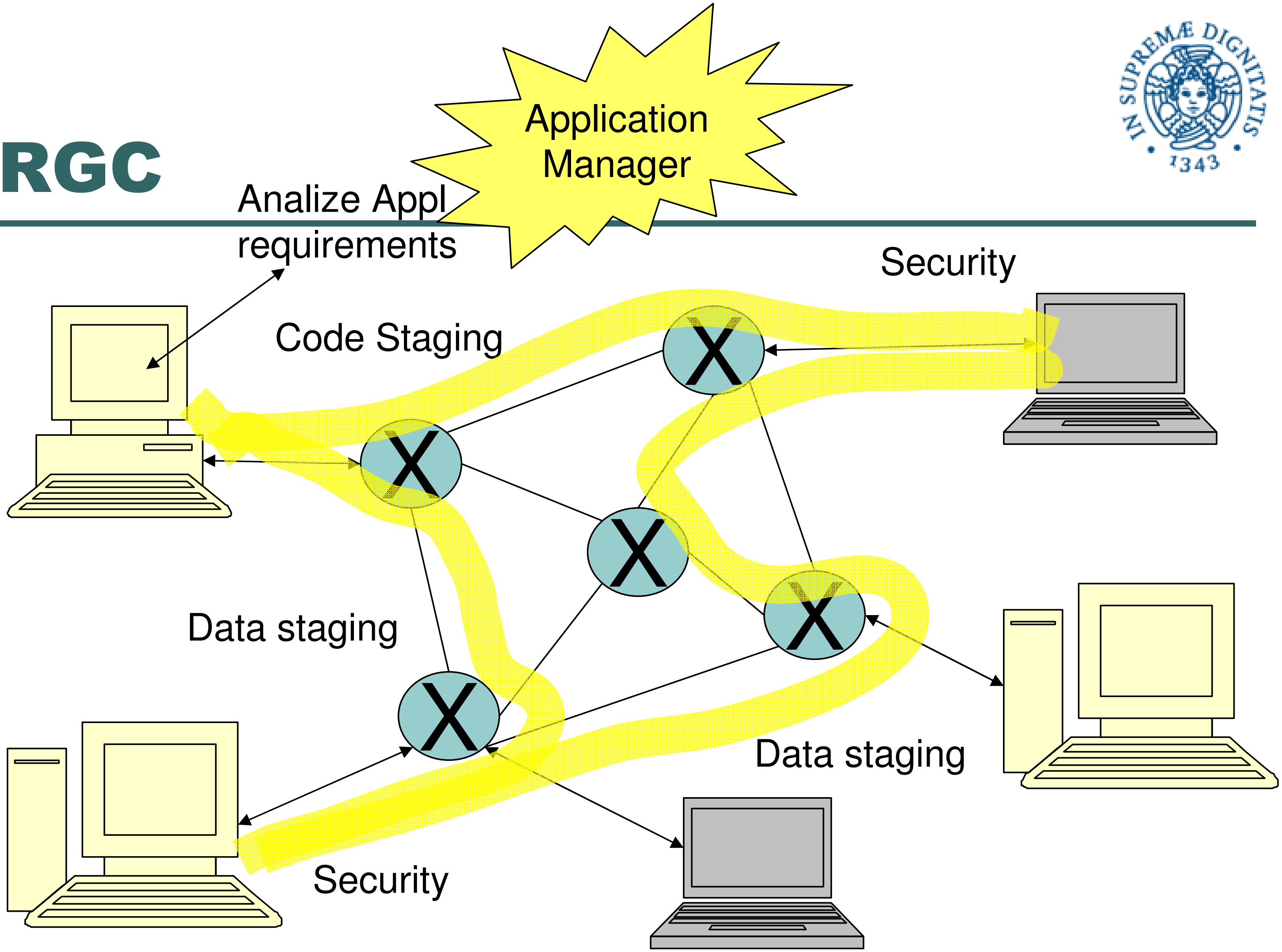
RGC


Need to compute A
What to be a GRID actor
What to be a GRID actor
X
X
X
X
X

RGC



RGC


Application
Manager
Analize Appl
requirements
Code Staging
Security
Data staging
Data staging
Security

RGC: *basic functionalities*

- *Secure access*
  (user classes, certificates, etc.)
- *Code & Data staging* (on demand!)
- *Secure data/code staging*
- *Introspection/reflection/meta info*
- *Remote control*
  (monitor, checkpoint, interrupt&resume)
- *Accounting*
- All services managed from application !

RGC bf: *secure access*

- Identify class of users
- Allow different sandbox levels
- Guarantee unique id & certificates
- Use session certificates for the single computation

RGC bf: *staging*

- On demand
- No long range consistency
- Completely application driven
- Caching allowed/enforced upon application directives
  - To enhance appl. performance as well as multiple runs performance

RGC bf: *secure staging*

- Session certificates
- Data **&** code
  - on demand, default → encode
- Performance issues
  - Latency/bandwidth vs. cypher/decypher time
  - Critical ? Very often latency much larger!

RGC bf: *introspection*

- Needed to gather compatibility info
- Needed to set up application deployment
- Open format
  - E.g. XML
- Service
  - To be asked from application manager
- Announce
  - When announcing node availability w.r.t. grid

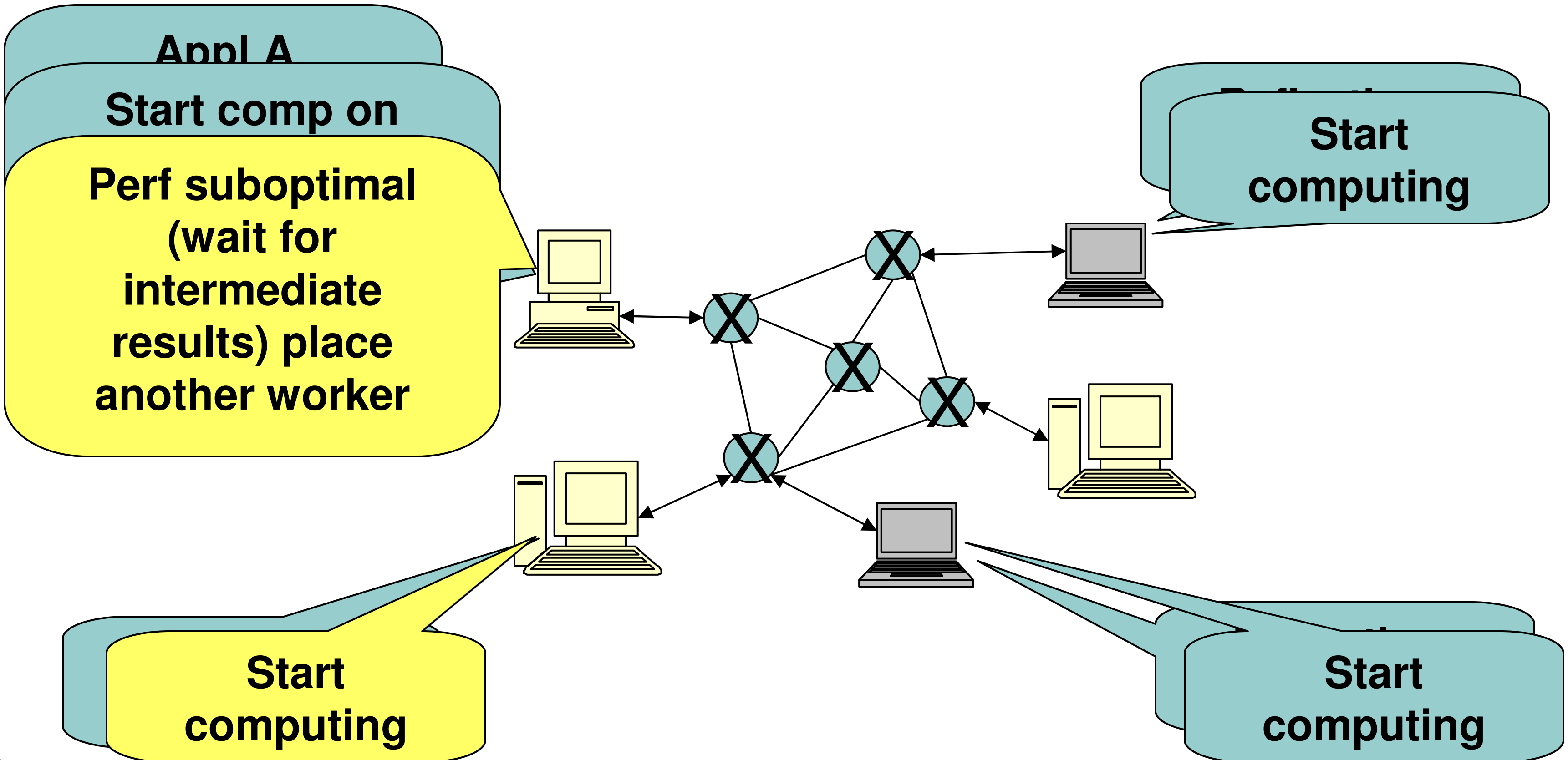
RGC bf: *remote control*

- Available to the application manager
  - To monitor code execution
  - To preempt no more useful computations
  - To force different behaviour upon performance changes (loss w.r.t. theoretical model)

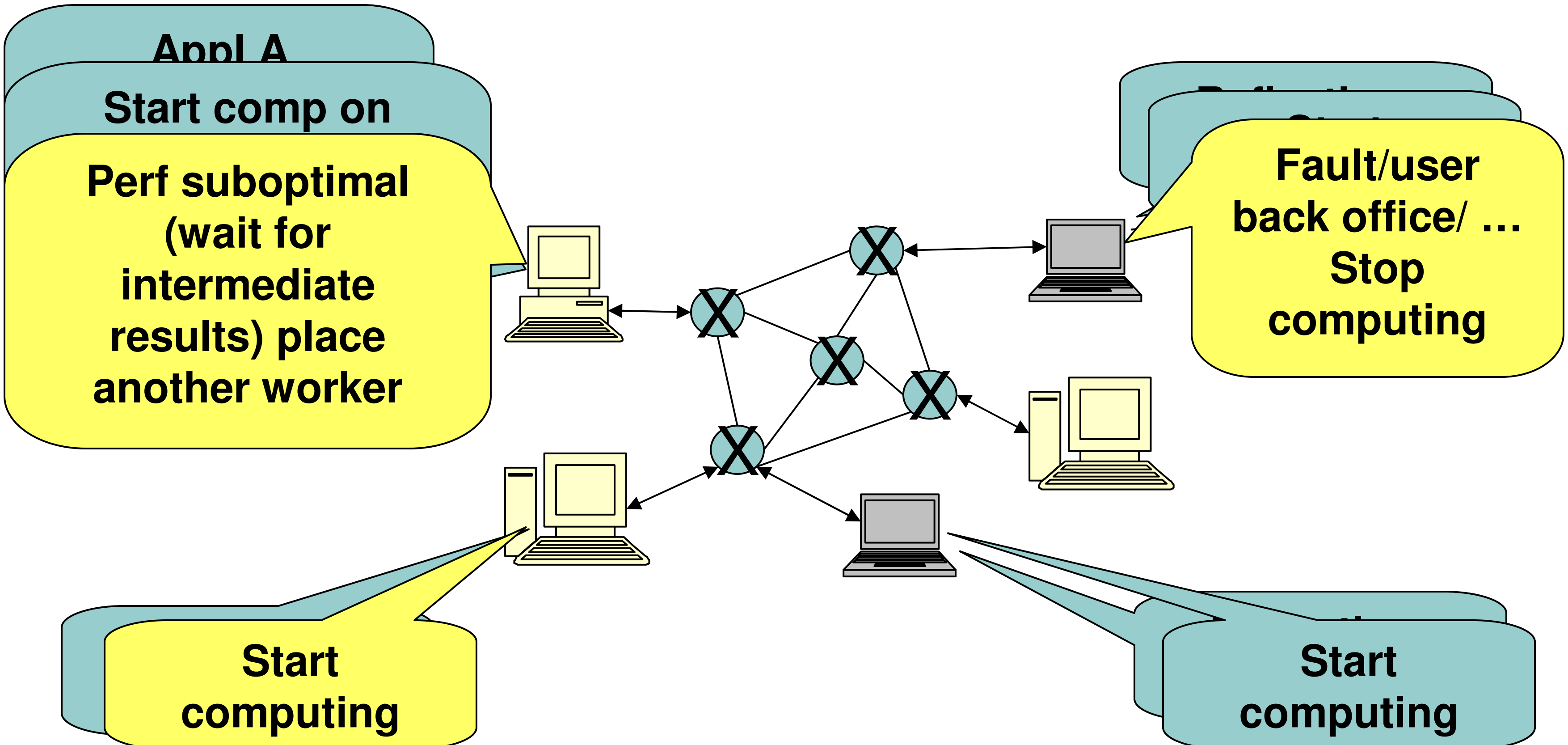
RGC bf: *accounting*

- Needed to bill users!
- Account CPU time
  - Different policies per user class
  - Different policies per type (idle time, …)
- *Must be present to make the approach acceptable*
  - Differently from other points above !

RGC scenario



RGC scenario (2)



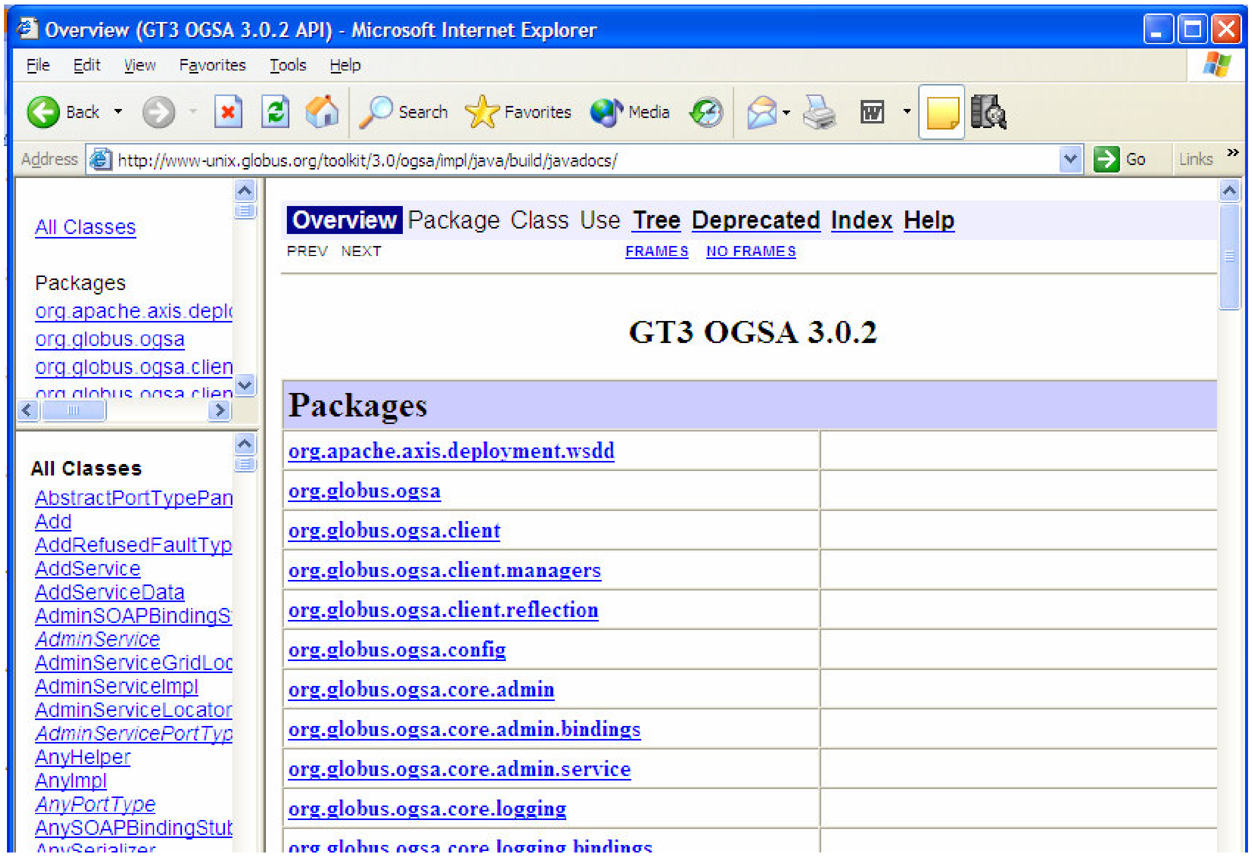
RGC implementation

- Server process on well known port
  - Peer2peer discovery/announce
  - Implements all basic RGC services
  - Possibly built on top of existing middleware
    - Globus
    - JXTA
    - …
  - Explicitly run by machine owner !

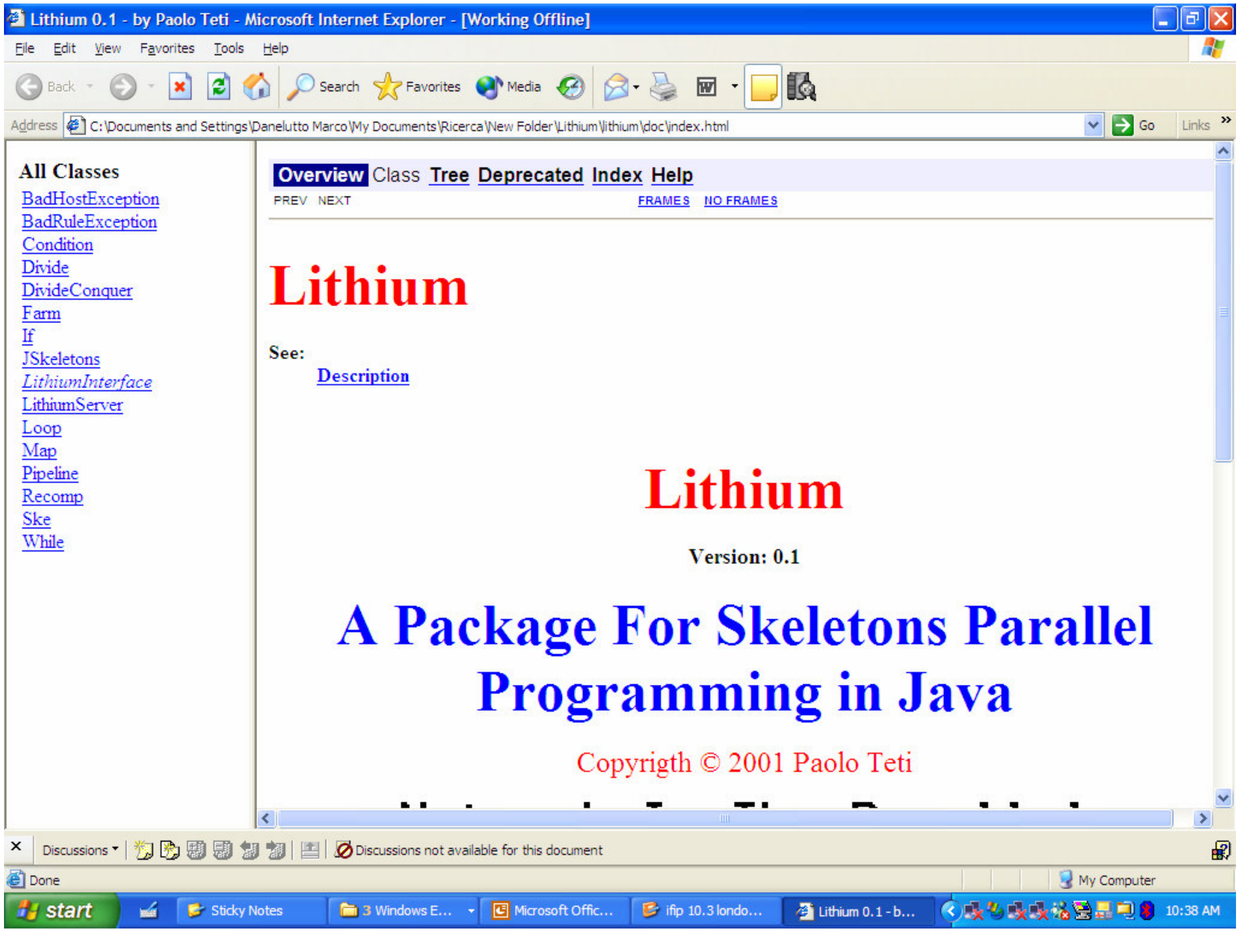
Why RISC ?

- Compare Globus services with our ones!
- Compare the amount of code in the p2p server placed on the machines!
- Compare the usage of services (application driven, even concerning server code staging)!

GT3 online man ...

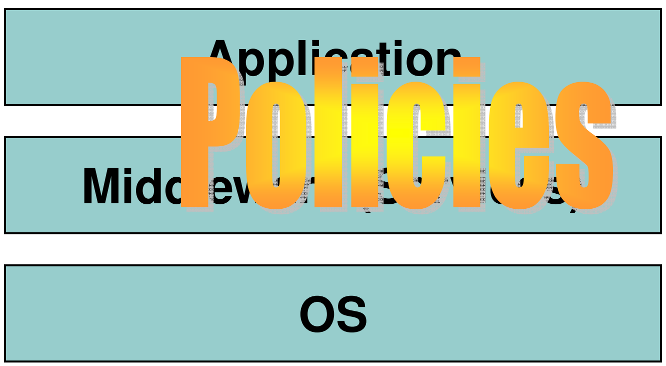


… vs. Lithium man !

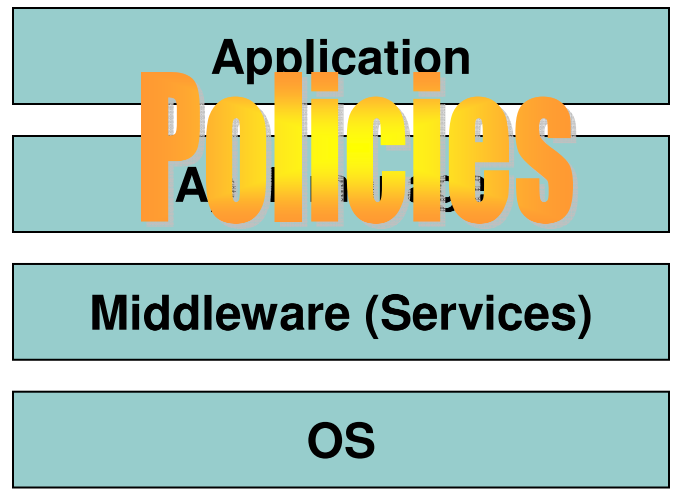


Why RISC (2)

- Control/policies moved up to application manager



currently



RGC

Contents

- GRID: current status
- A RISC grid core
- *Current experiments*
- Conclusions

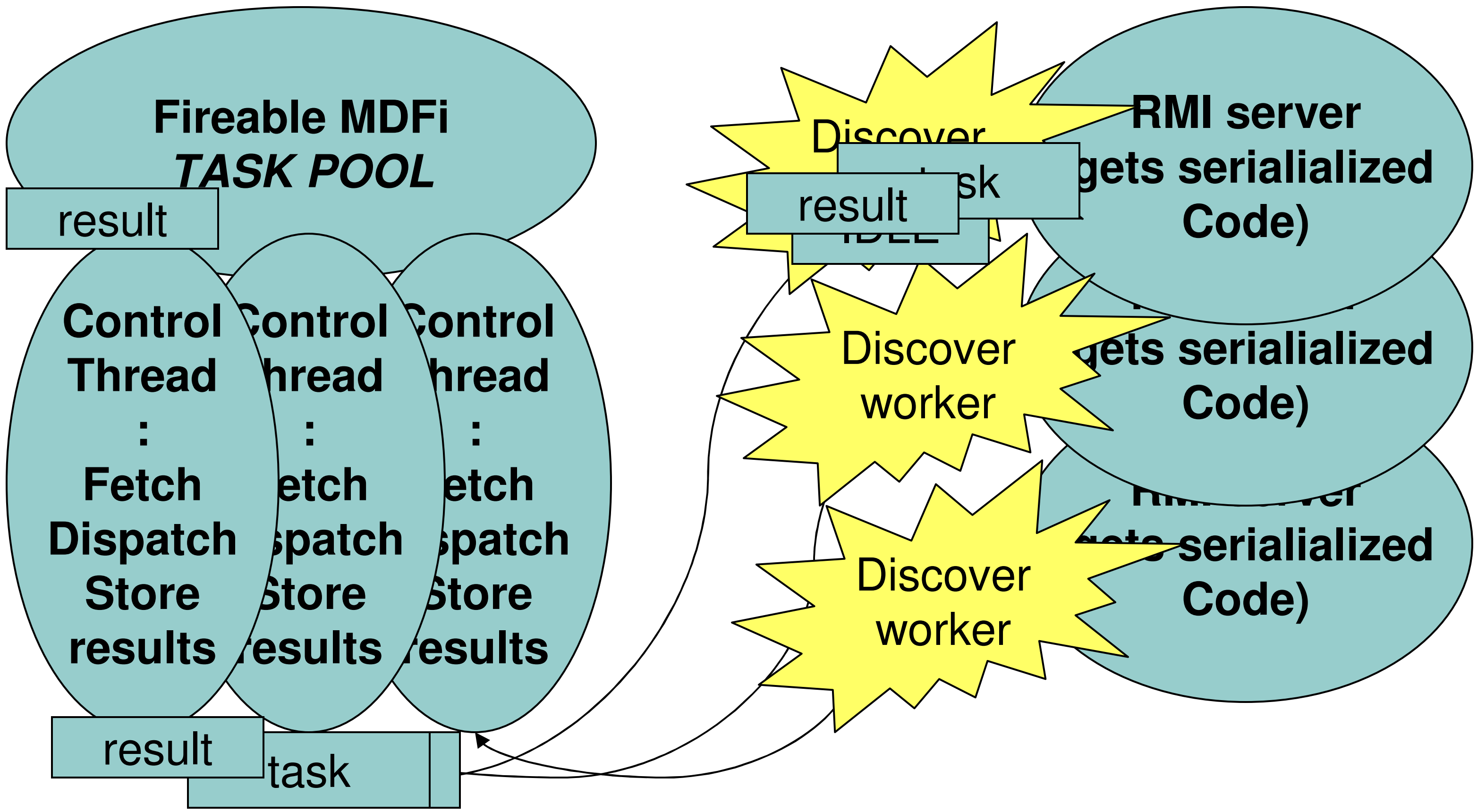
Framework

- FIRB GRID.it
  - Three year Italian National Project
  - Basic research on grids
  - WP8 “advanced, structured, component based parallel programming model for grids”
- In the meanwhile
  - Several proptotypes already available
    - P3L, Lithium, ASSIST, …

Lithium

- Java based, parallel, structured programming environment
  - RMI
  - Algoritmical skeletons (including farms, pipelines & divide&conquer)
  - Full Java library
  - On since 2001 (experimental)

Lithium structure



Lithium GRID evolution

- Discovery → *JINI, JXTA, …*
- Code/Data staging → *Java serialization*
- Security → *SSL*
- Reflection → *Beans*
- Remote Control → *RMI*
- Accounting → *???*

Application manager sample

- Embarassingly parallel computations
- Data at the application site
- Results at the application site
- *TASK FARM dynamic* template
  - *Compute tasks ASAP*
  - *On the available resources*
  - This is a CONDOR-like, Case A ***application schema***!

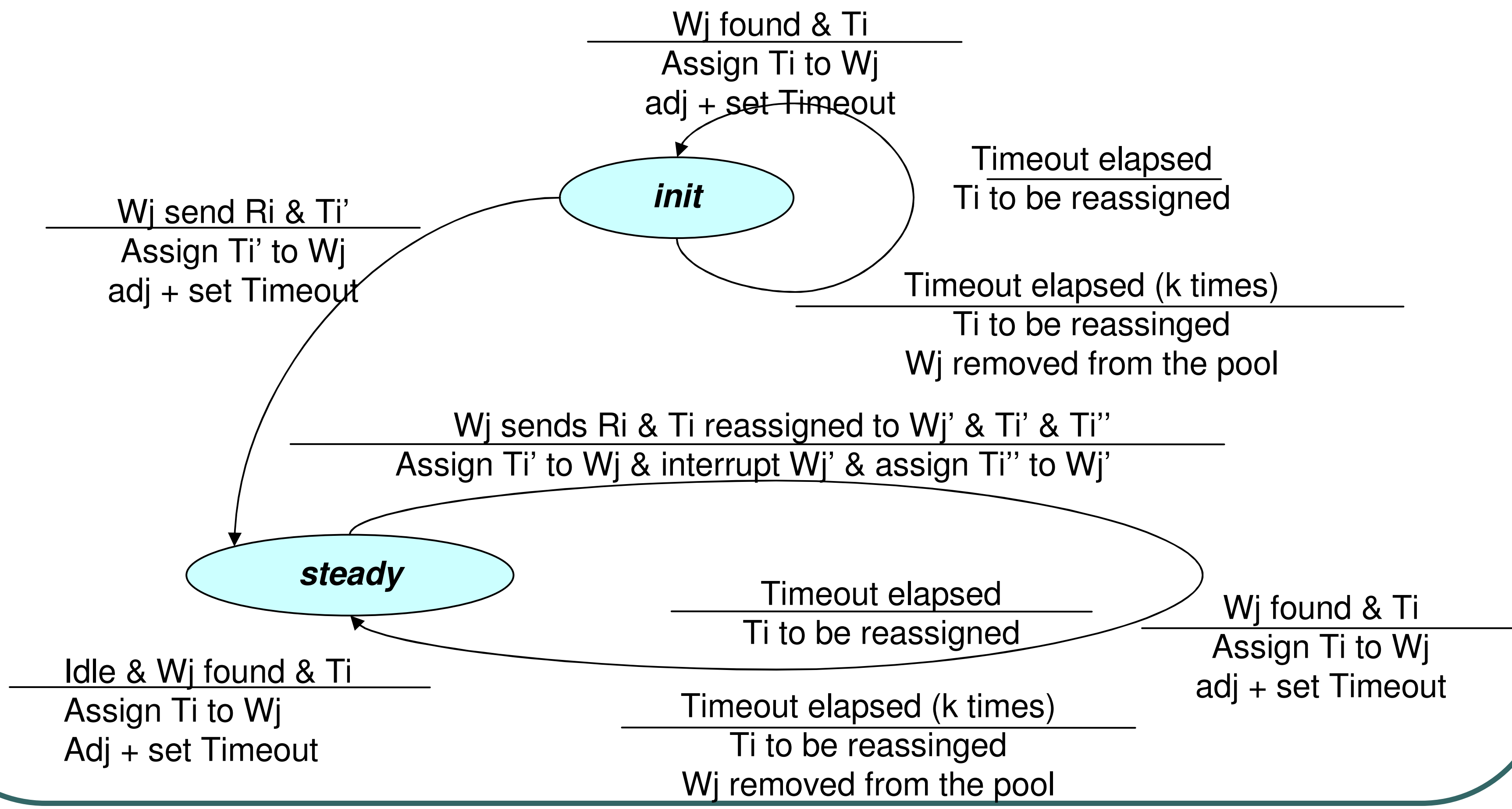
Usually ...

- Task list (pool)
- Code computing f
- List of possible workers (static, dynamic required to the GRID resource manager, available once and for all)
- … go
- & wait for completion

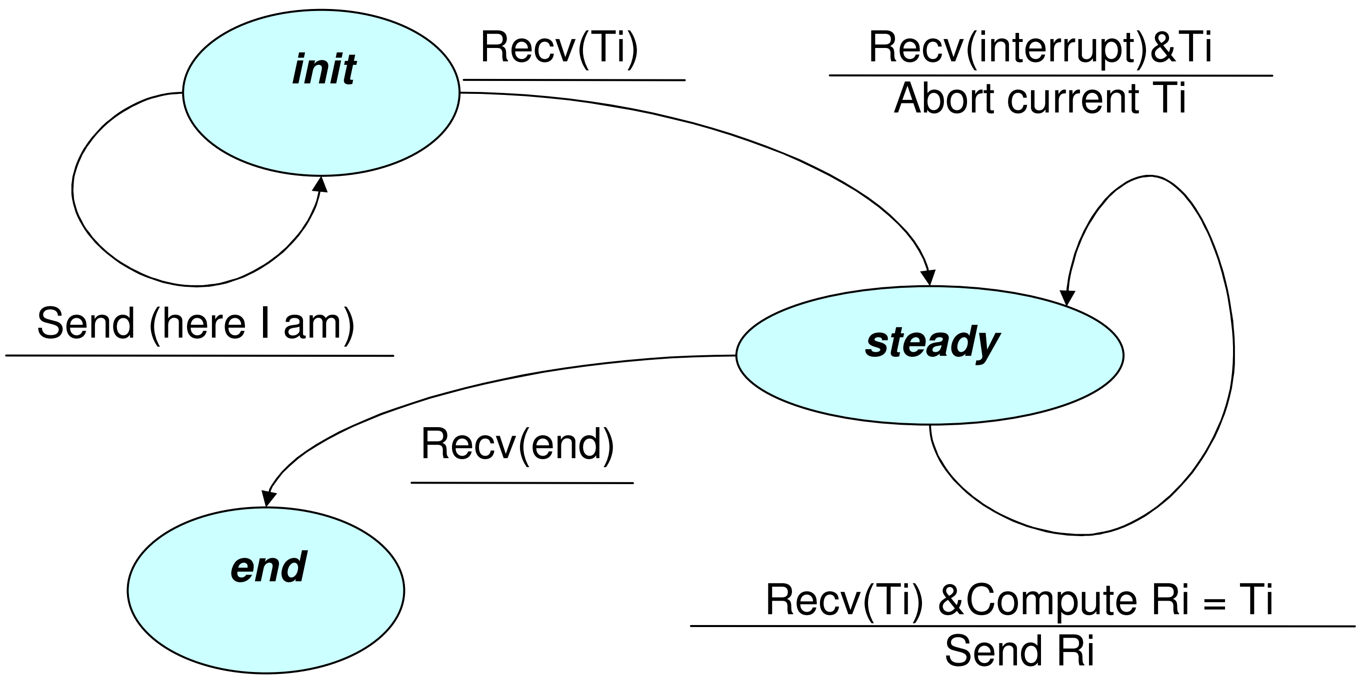
RISC approach

- Task farm manager
  - Looks for workers (p2p)
  - Assigns tasks for execution
  - When results come back
    - Stops looking for new workers (steady state)
    - Assigns tasks for execution to idle workers
- Workers
  - Just receive work to be computed
    (RGC: computation services only at the GRID peers!)

Task manager (FSA)



Worker (FSA)



init
Recv(Ti)
Recv(interrupt)&Ti
Abort current Ti
Send (here I am)
steady
Recv(end)
end
Recv(Ti) &Compute Ri = Ti
Send Ri

therefore …

- Adaptivity
  - To faulty workers
  - To dinamic changes in network performance
- Fault tolerance
  - Faulty workers
- Heterogeneity
  - Java …

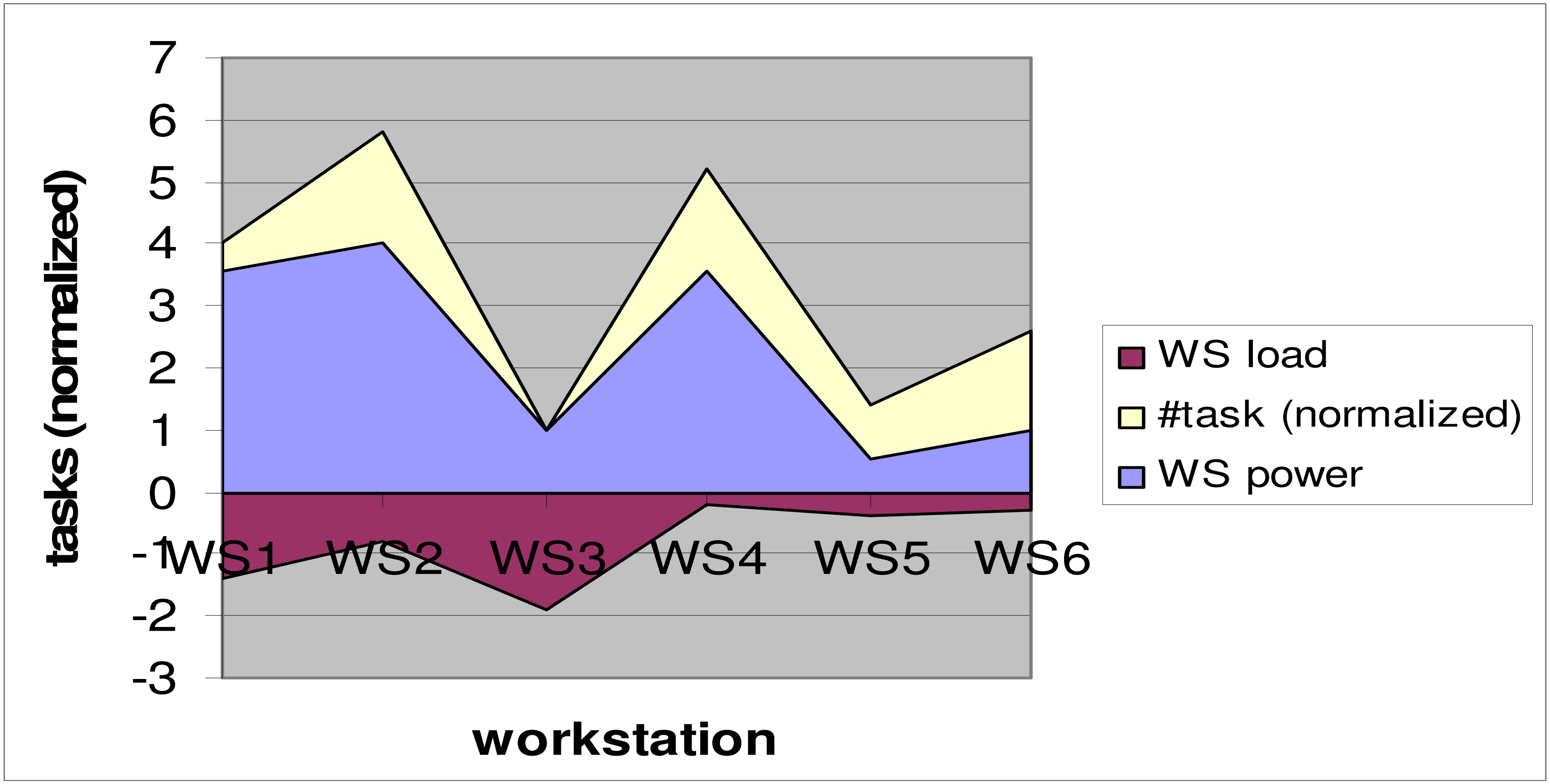
Experiments

- Modified version of the Lithium prototype
  - RMI, Serialization, …
- Simulator
  - Java program
  - Exact knowledge
  - … compared to measured

Results: *load balancing*

- Execute
  - a stream of independent task
    - Average execution time not known
    - Distribution not known
  - On a set of *production* workstations
    - Different Hw
    - Different load

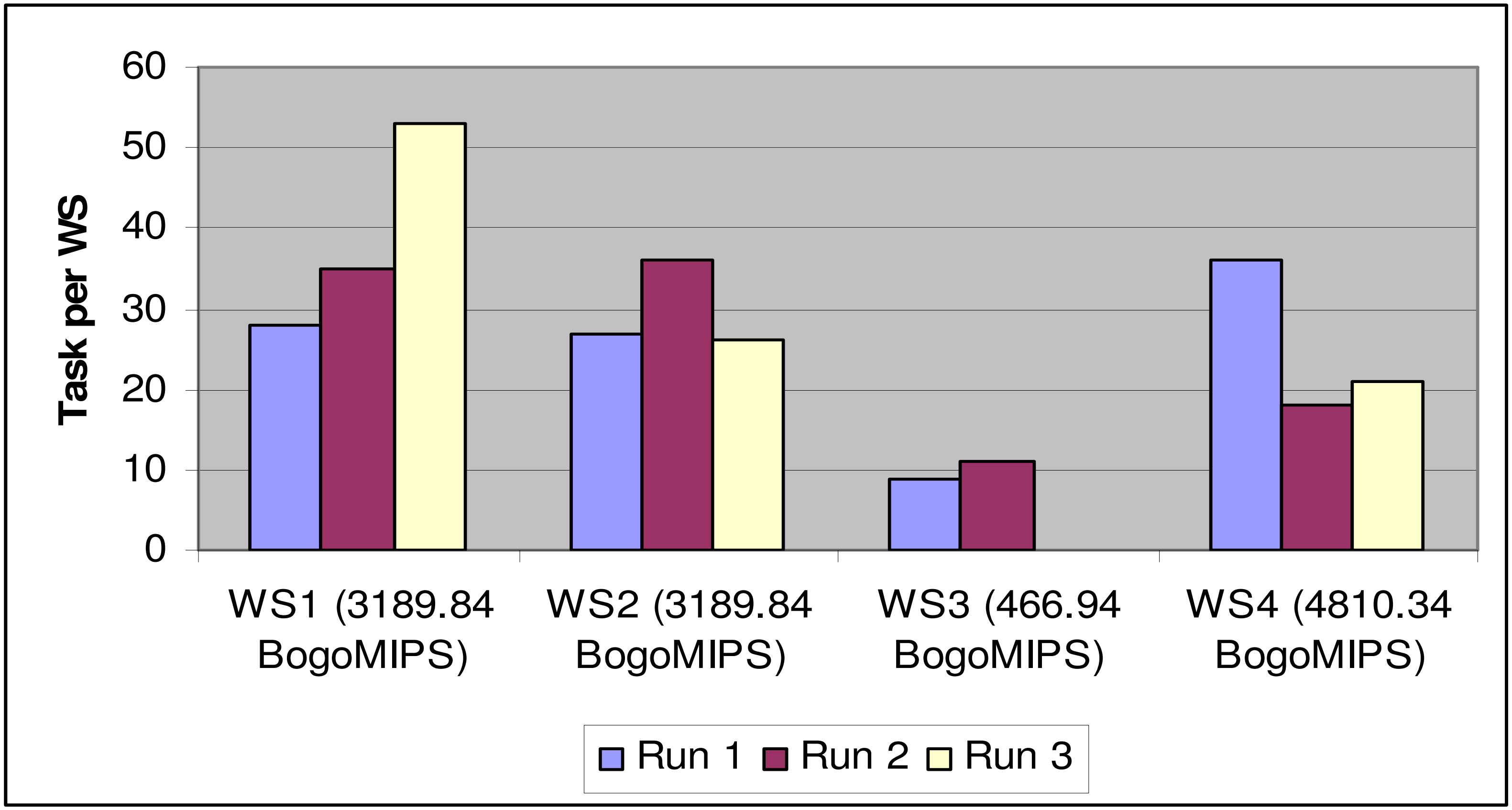
Results: *load balancing*



Results: *discovery*

- Run a stream of tasks
- On a set of production workstations
- First run
  - All discovered since T0
- Secon run
  - WS4 discovered after ½ tasks
- Third run
  - WS3 never discovered
  - WS2 discovered after 1/3 tasks
  - WS4 discovered after ½ tasks

Results: *discovery*



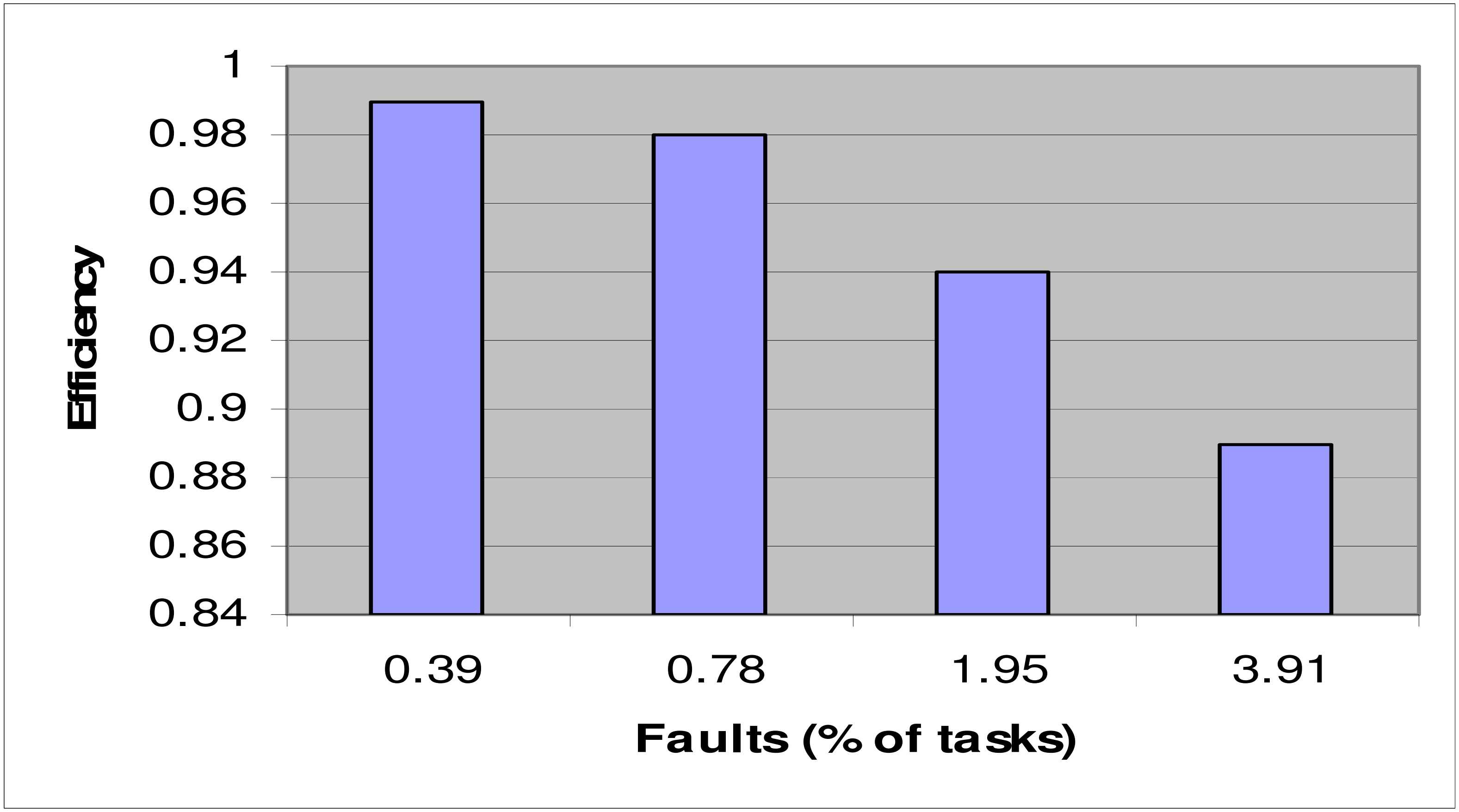
Results: *fault tolerance*

Simulation

- Workers assumed to fail
  - Due to internal problems
  - Due to network problems/delays
- Faults % to the total number of tasks executed
- Efficiency measured

Results: *fault tolerance*

Simulation

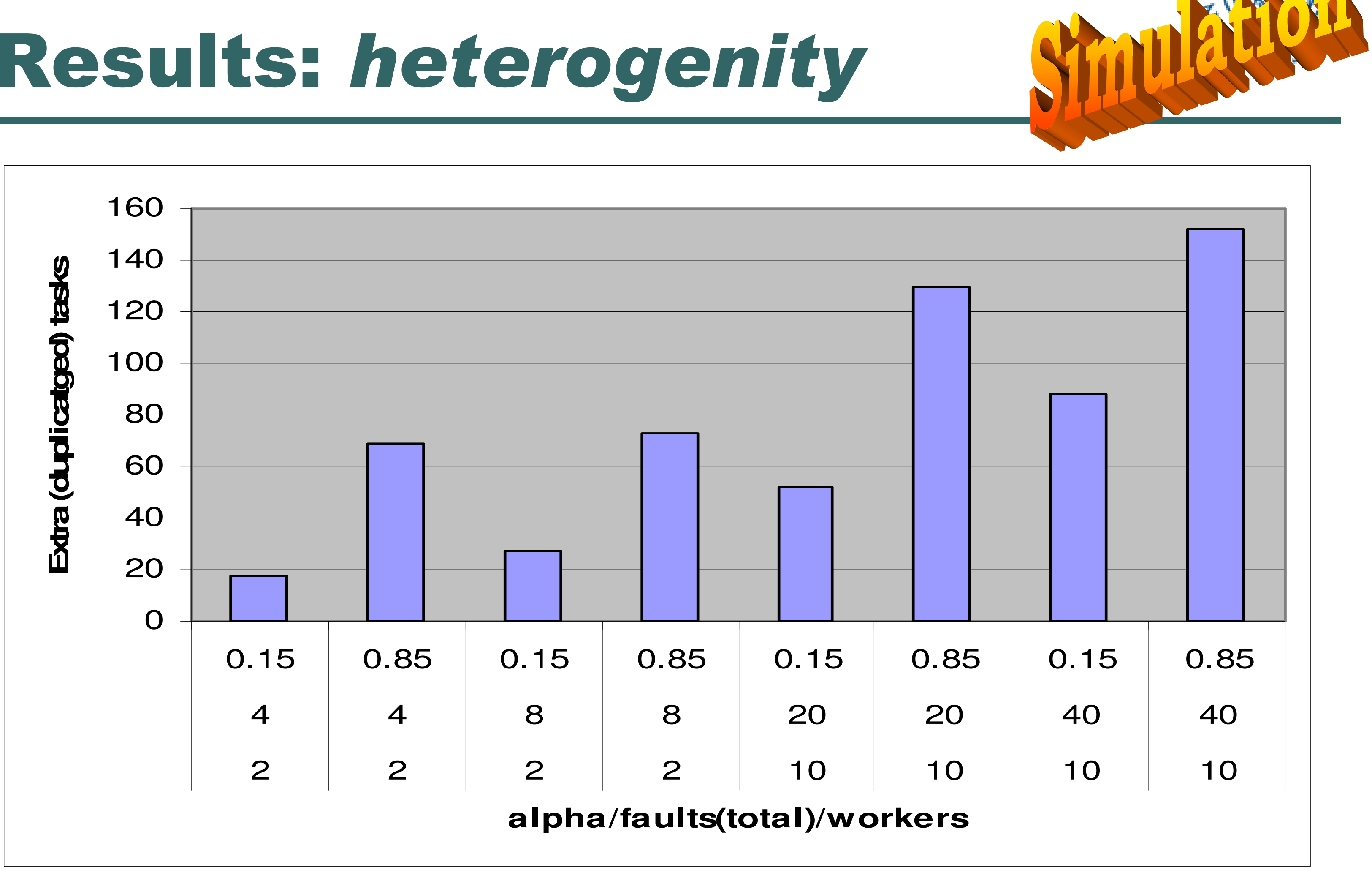


Results: *heterogeneity*

Simulation

- A stream of 1024 tasks executed
- With 2 or 10 workers
- 2 or 4 faults per worker (at random time)
- $\alpha = 0.15$ or $\alpha = 0.85$
  $RTT = \alpha\ RTT + (1-\alpha)\ RTTcurrent$

Results: *heterogenity*



Contents

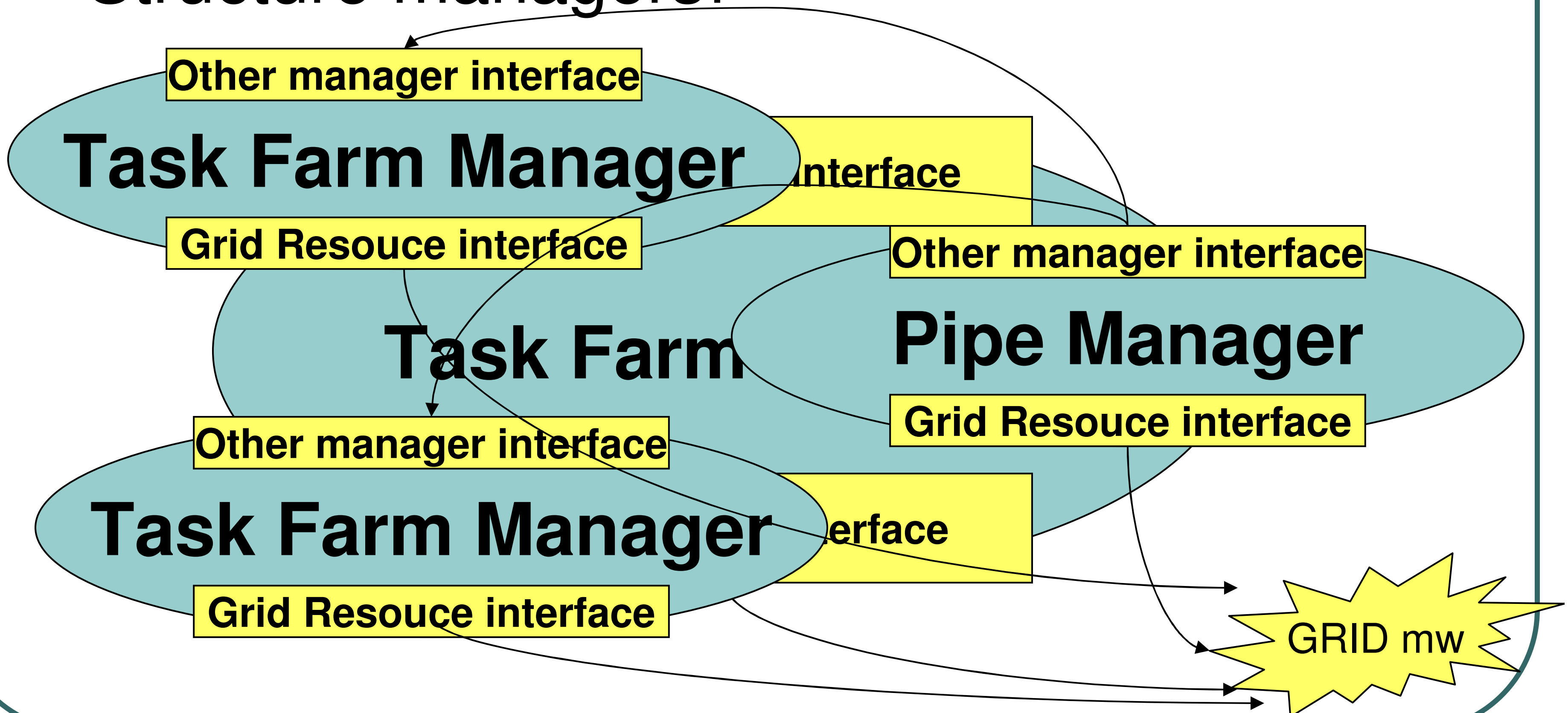
- GRID: current status
- A RISC grid core
- Current experiments
- Conclusions

Evolution

- Task farm manager
  → application schema manager
- Other managers (e.g. pipeline)
  → more complex application schemas
- Become an *environment*
  - Suitable to handle common grid aware applications

Evolution (2)

- Structure managers!



Evolution (3)

- Programming environment
  - Target GRIDs
  - GRID aware
  - Structured
  - Component based
  - With application managers
  - Sitting on top of *common sense* middleware (say GT3, Java/Jini/JXTA, …)

Related work

- So much activity on GRID … !
- Want to cite CONDOR
  - Limited application schema
  - Very efficient RTS
  - Included in recent toolkits
- Whereas RGC:
  - Unlimited application schemas
  - (hopefully) efficient RTS
  - ???

Conclusions

- New methodology for GRID tool development proposed
- Join skeleton/design pattern results with the GRID world
- Preliminary results
- Currently evolving
- …

marcod@di.unipi.it
www.di.unipi.it/~marcod

A RISC approach to the GRID

any questions ?