Autonomic management of performance concerns in GCM

Marco Danelutto

Dept. Computer Science, Univ. of Pisa
& CoreGRID ERCIM working group

OGF25/EGEE Forum
Catania 2-6, 2009
Workshop
“From GRID monitoring to analysis”

Contents

- Parallel computational patterns on grids
- Behavioural skeletons
    - Skeletons to model parallel computation pattern
    - Autonomic management to take care of non functional features
    - BS implementation in GCM
- Conclusions

Parallel computational patterns on GRIDS

- Two classes of application patterns (most succesfull/used)
  - embarrassingly parallel computations
    - bunch of tasks, task farm, master/worker, master/slave, parameter sweeping, map, forall independent, ...
  - workflows
    - small parallelism degree
    - although nodes/tasks with huge internal parallelism
- More patterns are known/studied (from algorithmic skeleton/parallel design pattern communities)

Skeleton concepts

- Algorithmic skeletons (Cole PhD thesis ’88)
  - several research groups followed (London, Pisa, Muenster, Orleans, Malaga, La Laguna, Tokio, Sophia Antipolis, ...)
- A skeleton is a *parallelism exploitation pattern*
  - *parametric*
    - par degree, code parameters (either skeletons or seq), ...
  - *reusable*
    - not bounded to application logic, general purpose
  - *known*
    - recognizable in common applications
  - *efficient*
    - efficient implementations exists on several distinct target architectures

CoreGRID experience

- Investigate feasibility of migrating skeleton concept to GRIDs
  - which skeletons
  - which implementation
  - which impact of peculiar grid features
    - dynamicity
    - hetereogeneity
    - non dedicated nodes
    - ....
- How do skeletons fit the component framework (GCM)

CoreGRID experience

- Investigate feasibility of migrating skeleton concept to GRIDs
  - which skeletons
  - which implementation
  - which impact of peculiar grid features
    - dynamicity
    - hetereogeneity
    - non dedicated nodes
    - ....
- How do skeletons fit the component framework (GCM)

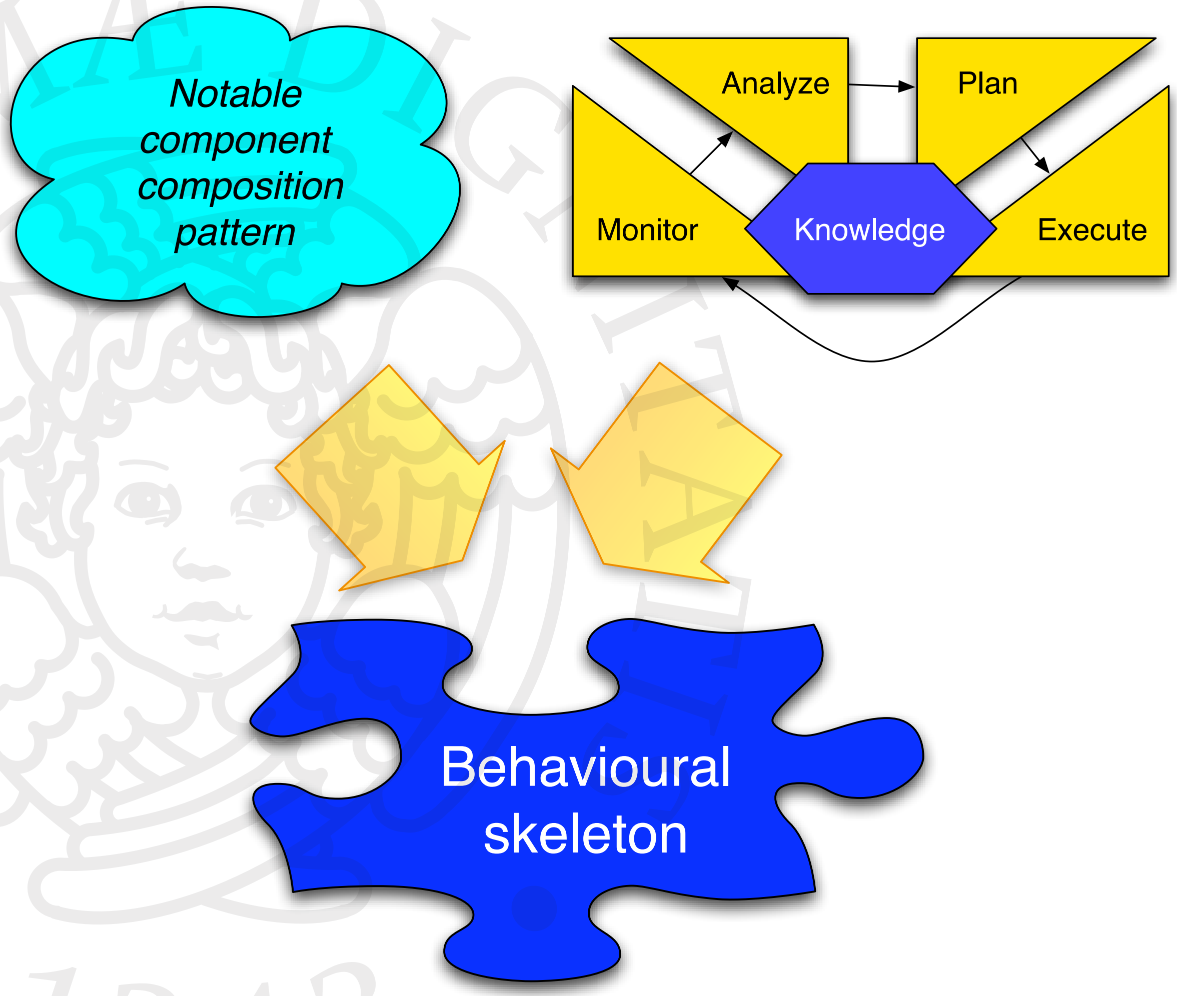
implementation related concerns (should be dealt with by the compiler/run time system rather than by programmers)

Complete separation of concerns

- Functional concerns
  - in charge to application programmers
- Non functional concerns
  - performance
  - security
  - fault tolerance
  - “green” computing
  - in charge to system programmers

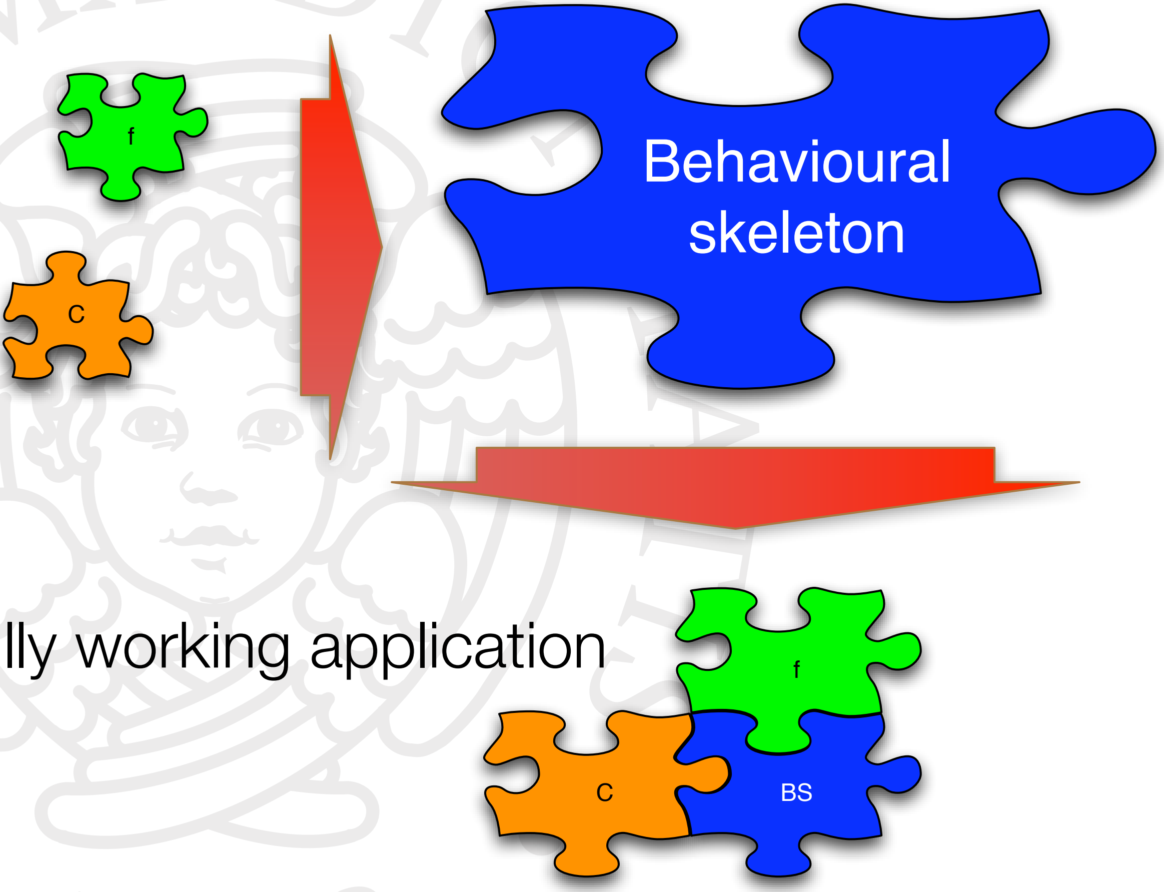
Complete separation of concerns

- Functional concerns
  - in charge to application programmers
- Non functional concerns
  - performance
  - security
  - fault tolerance
  - “green” computing
  - in charge to system programmers



Complete separation of concerns

- Functional concerns
  - in charge to application programmers
- Non functional concerns
  - performance
  - security
  - fault tolerance
  - “green” computing
  - in charge to system programmers

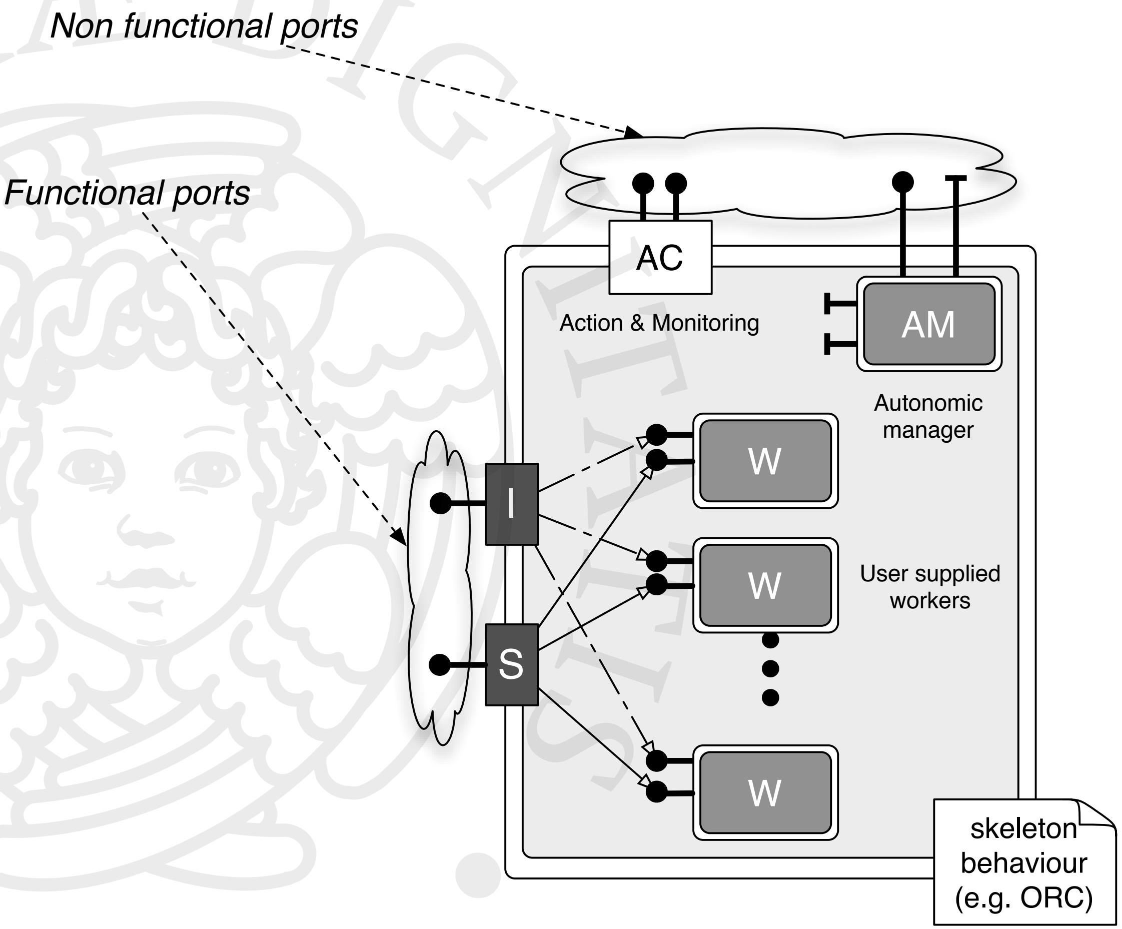


Behavioural skeletons the GCM way

- Composite components
  - including skeleton
  - including autonomic manager
- Ports to
  - configure (set up code params)
  - compute (tasks)
  - configure (SLA contracts)

Behavioural skeletons the GCM way

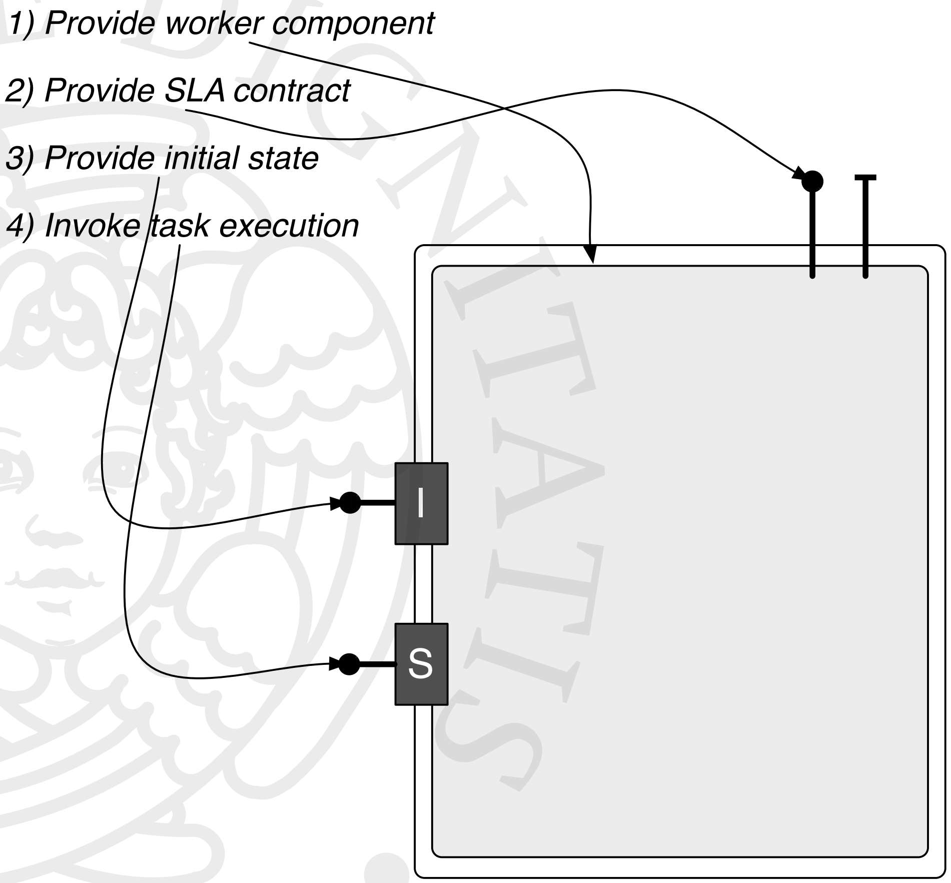
- Composite components
  - including skeleton
  - including autonomic manager
- Ports to
  - configure (set up code params)
  - compute (tasks)
  - configure (SLA contracts)



Functional replication BS (with shared state)

Behavioural skeletons the GCM way

- Composite components
  - including skeleton
  - including autonomic manager
- Ports to
  - configure (set up code params)
  - compute (tasks)
  - configure (SLA contracts)



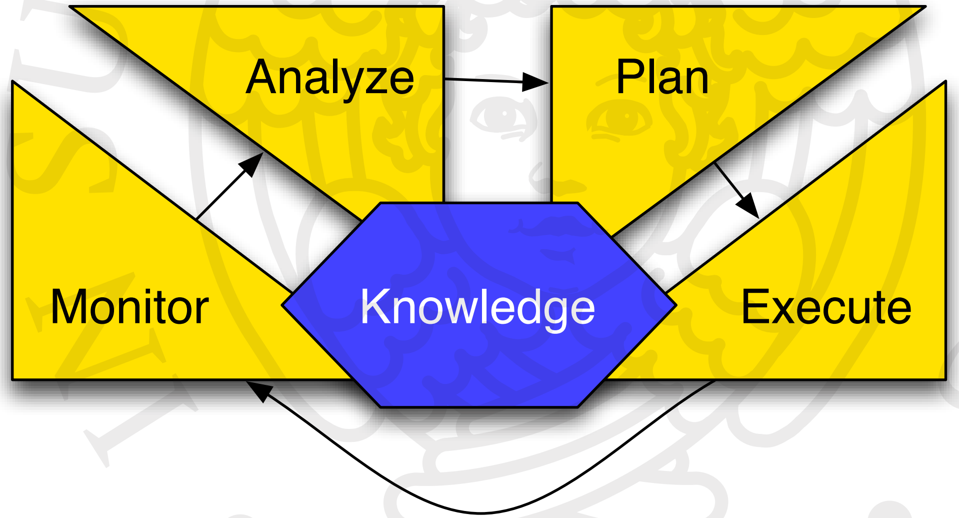
1) Provide worker component

2) Provide SLA contract

3) Provide initial state

4) Invoke task execution

Autonomic management of non functional concerns

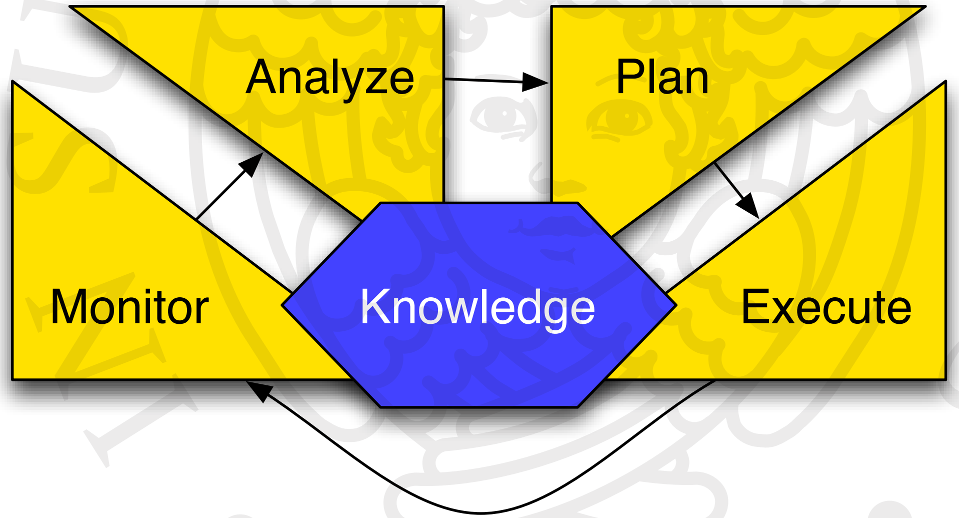


Autonomic management of non functional concerns

Analyse status w.r.t. user SLA

Consult policies and plan correction actions (if any)

Monitor current execution status



Operate corrective actions

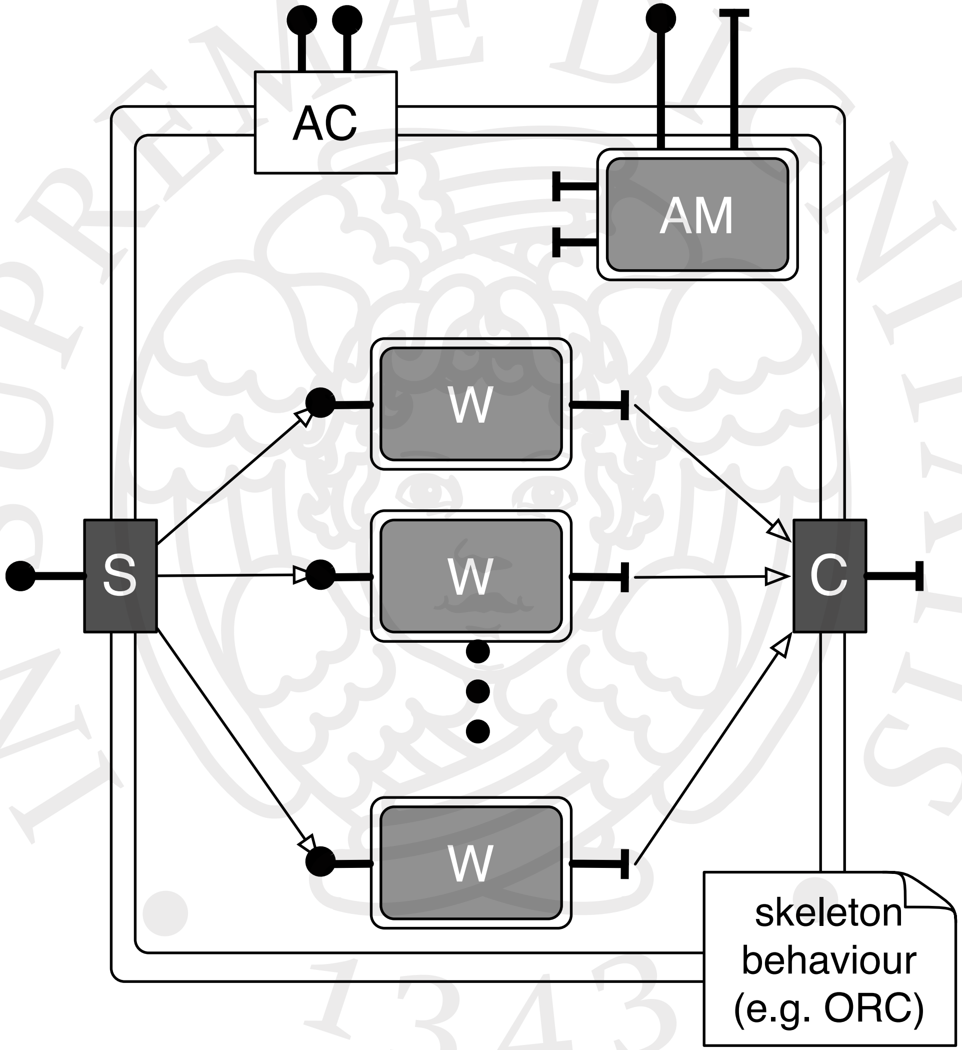
Complete separation of concerns

- *Application programmer*
    - uses pre-defined, parallel, composite GCM components modelling skeletons
    - provides parameters to instantiate the composite GCM component to serve the application at hand
    - provides a SLA contract, establishing the pretended behaviour of the composite GCM component
        - performance, fault tolerance, security, ...

Complete separation of concerns (2)

- *System programmer*
  - implements the GCM composite exploiting the skeleton parallel pattern
    - using further “system” components and user supplied components (functional part)
    - programming a set of *monitoring features* to inspect component behaviour
    - programming a set of *actions* to intervene when component behaviour does not match user expectations (SLA contract)
  - uses monitoring and actions to implement *precondition-action rules* that manage autonomically the component behaviour

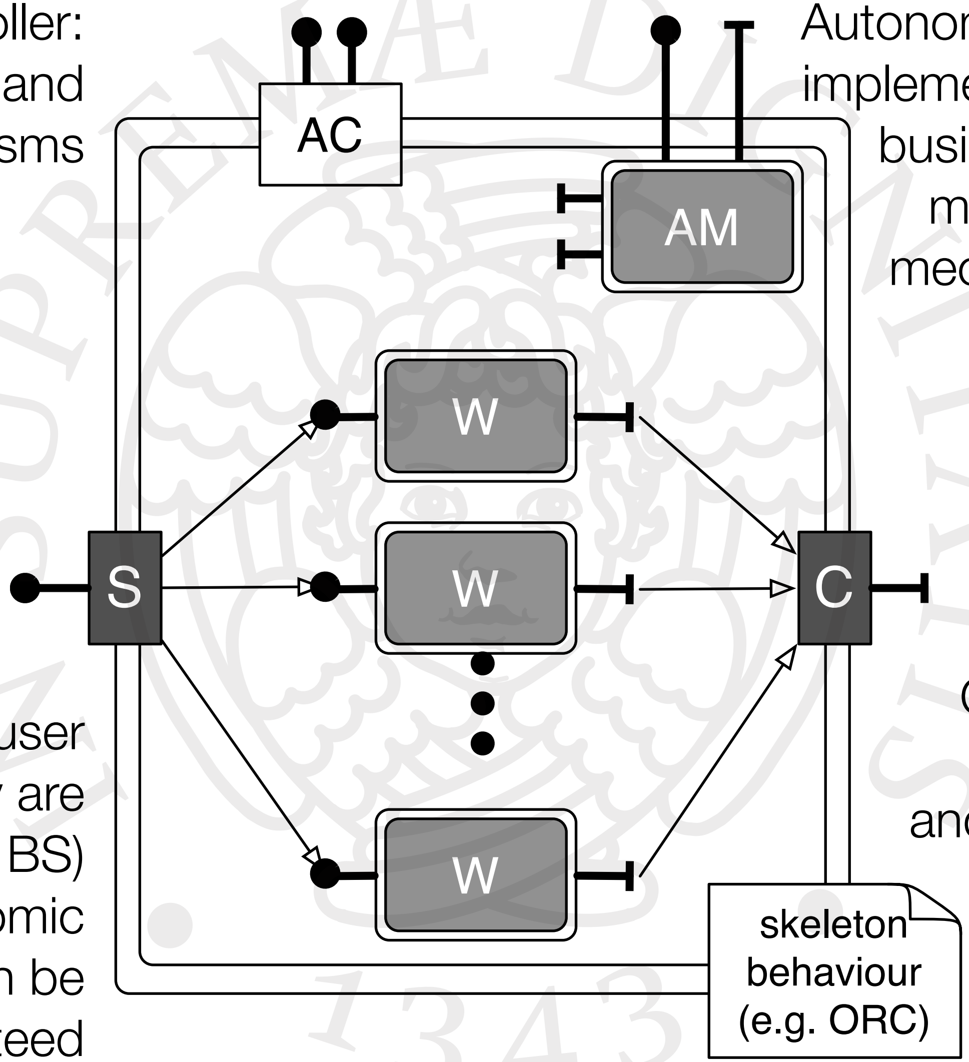
Behavioural skeleton in GCM (closer look)



Behavioural skeleton in GCM (closer look)

Autonomic controller: implements action and monitoring mechanisms

Autonomic manager: control loop implemented on top of the JBoss business rule engine; uses monitoring and action mechanisms from the AC



Inner components: user supplied; in case they are plain components (not BS) lower levels of autonomic management can be gauranteed

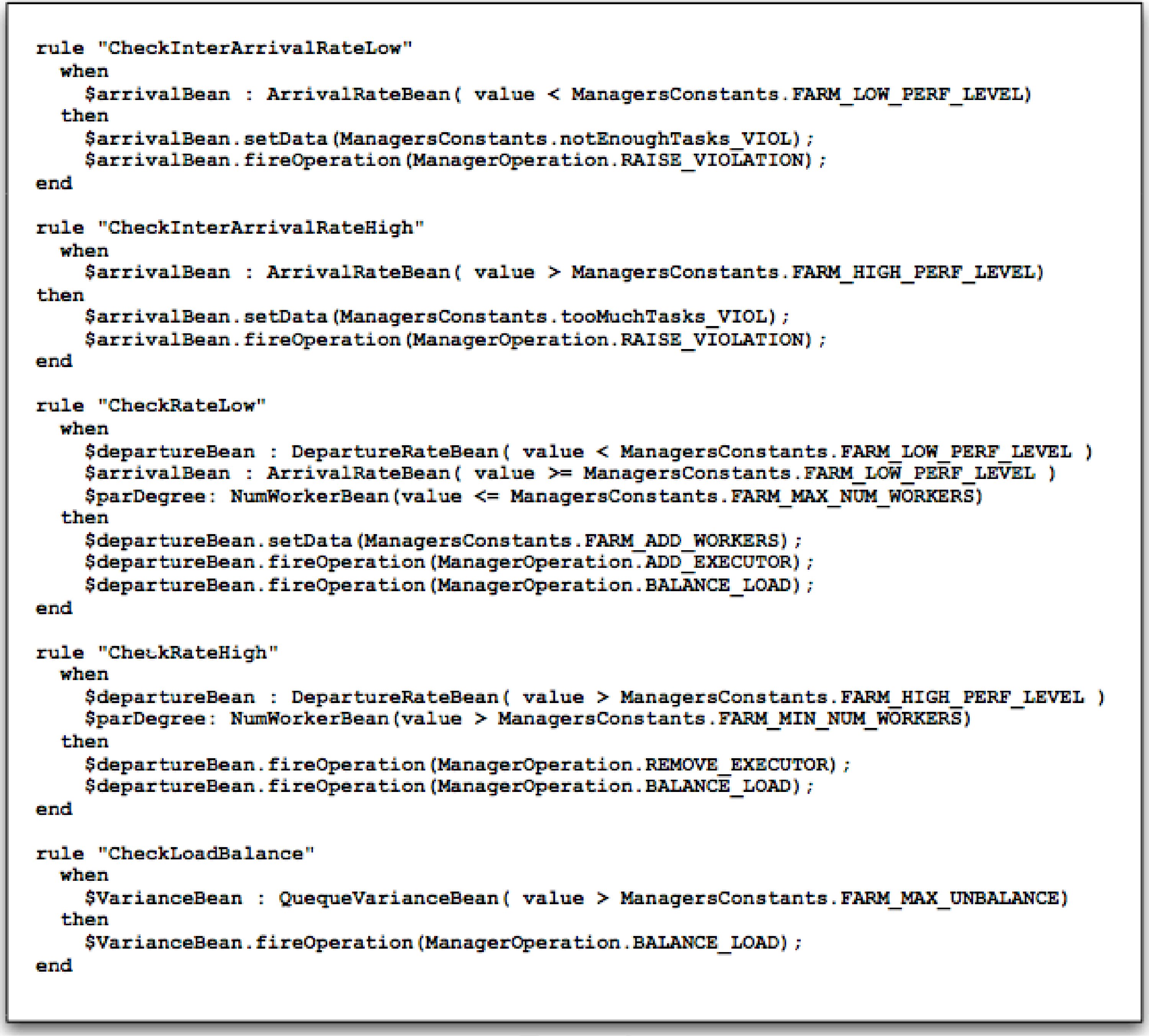
ORC formal specification: used to reason about BS and to drive implementation process

Performance non functional concerns

- SLA contract
  - expected service time
- Monitoring
  - service time of inner components (workers)
  - inter-arrival time
  - length of the queues (async port calls due to ProActive)
- Actions
  - add / remove worker
  - rebalance load

Performance non functional concerns (2)

- Sample rules programmed in the autonomic manager
  - `when ( service time > inter arrival time & SLA not satisfied)` → `add worker`
  - `when (service time < SLA )` → `remove worker`
  - `when ( unbalanced worker task queue)` → `rebalance`
- when clause
  - triggered with monitoring events
- then clause
  - operated through actions



```
rule "CheckInterArrivalRateLow"
  when
    $arrivalBean : ArrivalRateBean( value < ManagersConstants.FARM_LOW_PERF_LEVEL)
  then
    $arrivalBean.setData(ManagersConstants.notEnoughTasks_VIOL);
    $arrivalBean.fireOperation(ManagerOperation.RAISE_VIOLATION);
end

rule "CheckInterArrivalRateHigh"
  when
    $arrivalBean : ArrivalRateBean( value > ManagersConstants.FARM_HIGH_PERF_LEVEL)
then
    $arrivalBean.setData(ManagersConstants.tooMuchTasks_VIOL);
    $arrivalBean.fireOperation(ManagerOperation.RAISE_VIOLATION);
end

rule "CheckRateLow"
  when
    $departureBean : DepartureRateBean( value < ManagersConstants.FARM_LOW_PERF_LEVEL )
    $arrivalBean : ArrivalRateBean( value >= ManagersConstants.FARM_LOW_PERF_LEVEL )
    $parDegree: NumWorkerBean(value <= ManagersConstants.FARM_MAX_NUM_WORKERS)
  then
    $departureBean.setData(ManagersConstants.FARM_ADD_WORKERS);
    $departureBean.fireOperation(ManagerOperation.ADD_EXECUTOR);
    $departureBean.fireOperation(ManagerOperation.BALANCE_LOAD);
end

rule "CheckRateHigh"
  when
    $departureBean : DepartureRateBean( value > ManagersConstants.FARM_HIGH_PERF_LEVEL )
    $parDegree: NumWorkerBean(value > ManagersConstants.FARM_MIN_NUM_WORKERS)
  then
    $departureBean.fireOperation(ManagerOperation.REMOVE_EXECUTOR);
    $departureBean.fireOperation(ManagerOperation.BALANCE_LOAD);
end

rule "CheckLoadBalance"
  when
    $VarianceBean : QuequeVarianceBean( value > ManagersConstants.FARM_MAX_UNBALANCE)
  then
    $VarianceBean.fireOperation(ManagerOperation.BALANCE_LOAD);
end
```

Green computing non functional concerns

- with the same logic shown before (w.r.t. performance)
  - stress the "remove worker" actions
    - higher priority
    - less stringent constrains to activate
  - add some "switch off to standby" actions
    - to be added to the remove worker actions
- overall
  - keep alive (and consuming) only those machines actually needed to satisfy the user SLA contract

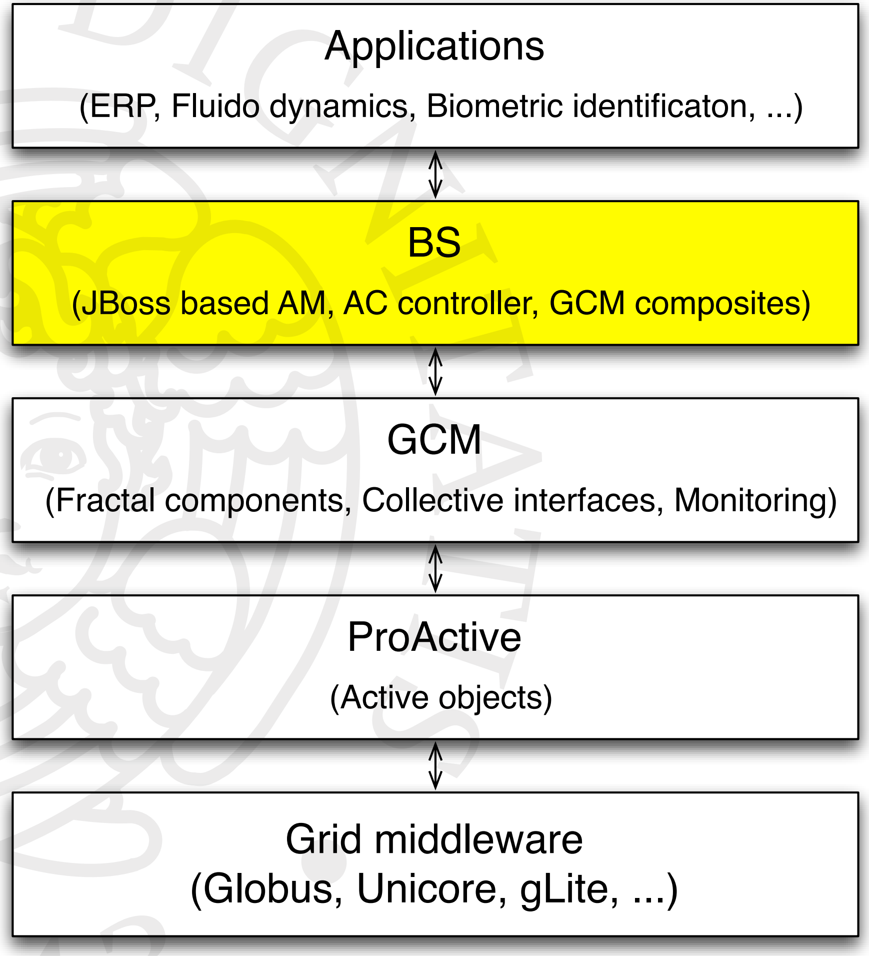
Reference Implementation (GCM BS)

- GridCOMP project
  - reference implementation of GCM + BS on top of ProActive middleware
  - STREP EU funded project 2006-2008
  - positive final review meeting in Pisa, last month of February
- Fully layered implementation

Reference Implementation (GCM BS)

GridCOMP
Effective Components for the Grids

- GridCOMP project
  - reference implementation of GCM + BS on top of ProActive middleware
  - STREP EU funded project 2006-2008
  - positive final review meeting in Pisa, last month of February
- Fully layered implementation



Applications
(ERP, Fluido dynamics, Biometric identificaton, ...)

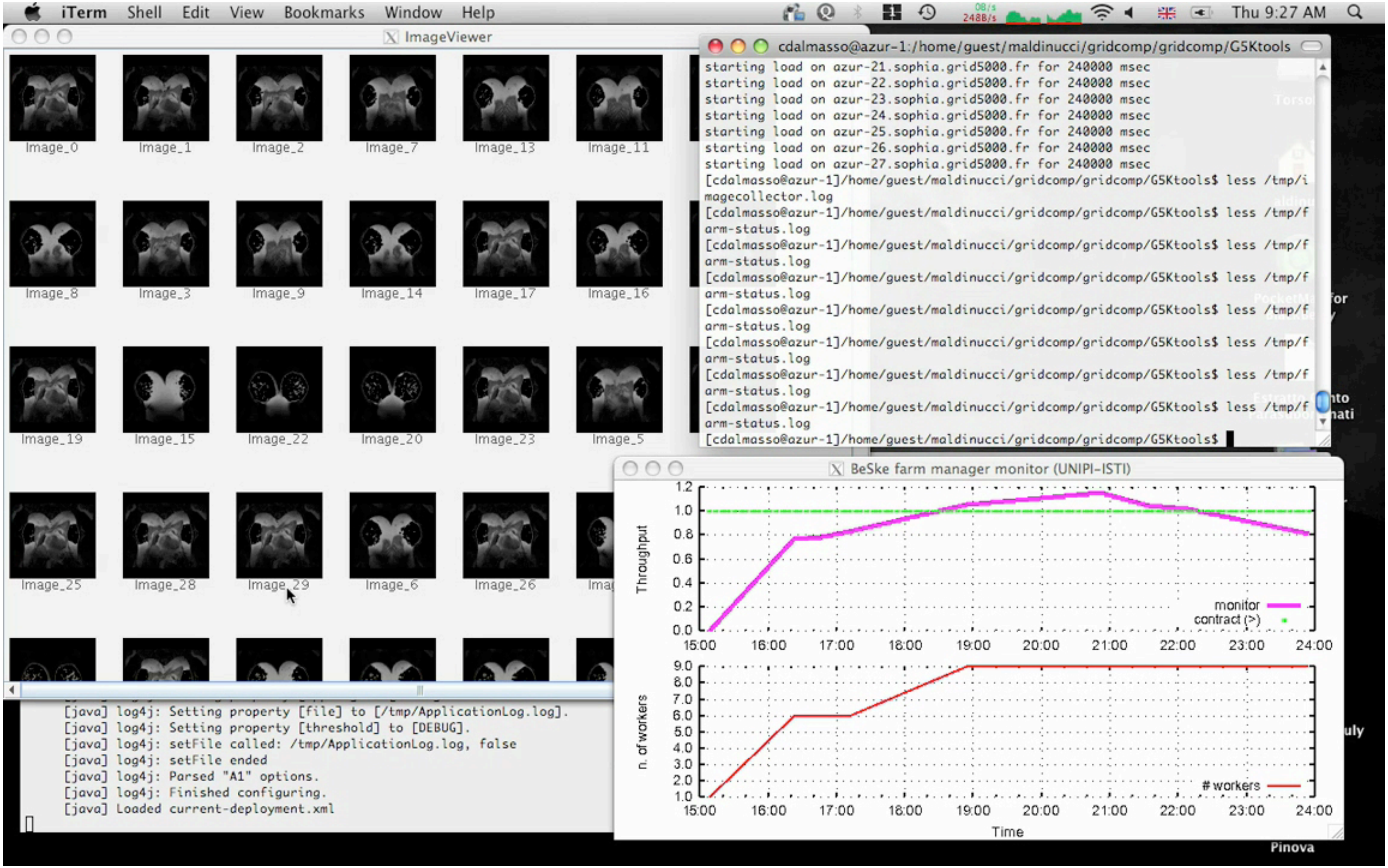
BS
(JBoss based AM, AC controller, GCM composites)

GCM
(Fractal components, Collective interfaces, Monitoring)

ProActive
(Active objects)

Grid middleware
(Globus, Unicore, gLite, ...)

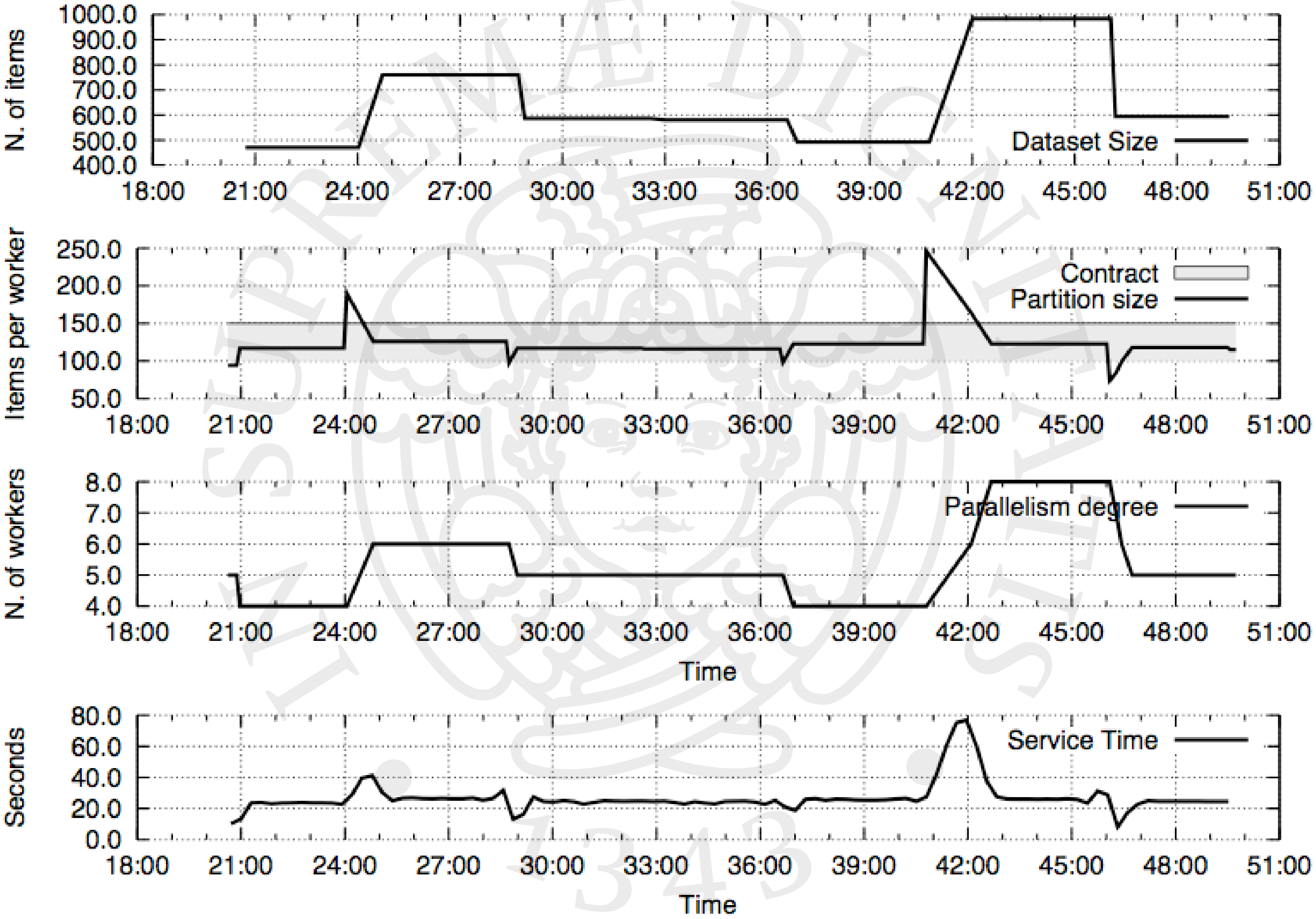
GCM BS (task farm) @ work (GridCOMP review’08)



```
starting load on azur-21.sophia.grid5000.fr for 240000 msec
starting load on azur-22.sophia.grid5000.fr for 240000 msec
starting load on azur-23.sophia.grid5000.fr for 240000 msec
starting load on azur-24.sophia.grid5000.fr for 240000 msec
starting load on azur-25.sophia.grid5000.fr for 240000 msec
starting load on azur-26.sophia.grid5000.fr for 240000 msec
starting load on azur-27.sophia.grid5000.fr for 240000 msec
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/i
magecollector.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/f
arm-status.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/f
arm-status.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/f
arm-status.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/f
arm-status.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/f
arm-status.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/f
arm-status.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$ less /tmp/f
arm-status.log
[cdalmasso@azur-1]/home/guest/maldinucci/gridcomp/gridcomp/G5Ktools$
```

```
[java] log4j: Setting property [file] to [/tmp/ApplicationLog.log].
[java] log4j: Setting property [threshold] to [DEBUG].
[java] log4j: setFile called: /tmp/ApplicationLog.log, false
[java] log4j: setFile ended
[java] log4j: Parsed "A1" options.
[java] log4j: Finished configuring.
[java] Loaded current-deployment.xml
```

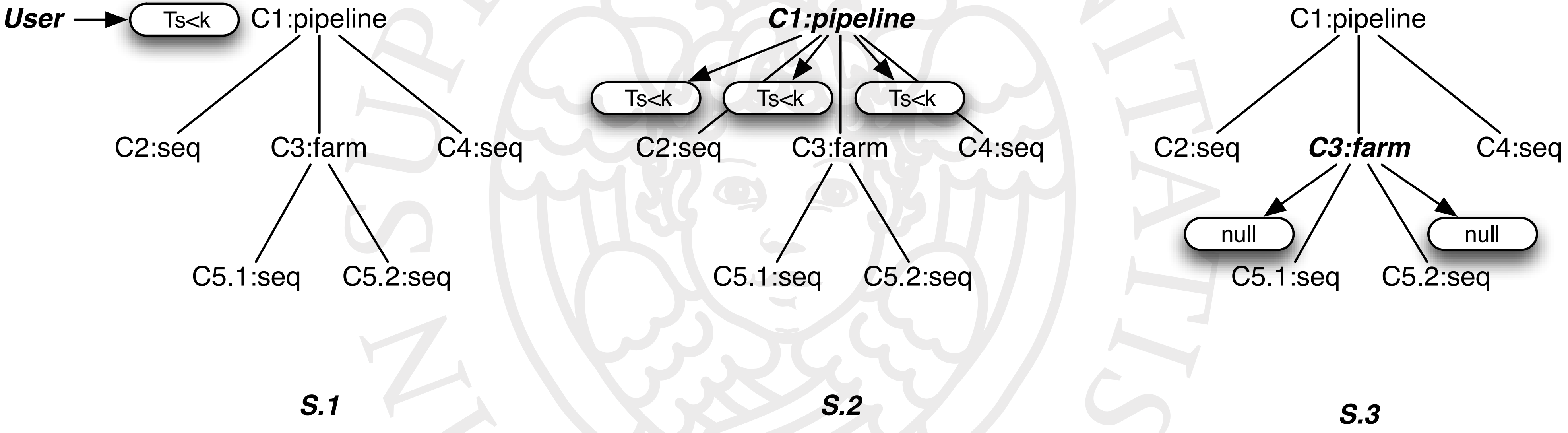
Data parallel BS @ work



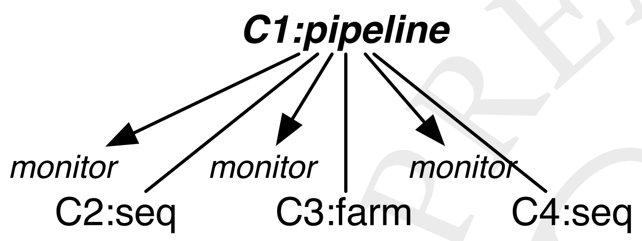
Single BS working ... then

- Managing hierarchies of managers
  - Propagation of contracts along the BS tree
  - Managing interaction among managers
    - managers supplying SLA contracts to other managers
    - managers reporting status (violations) to other managers

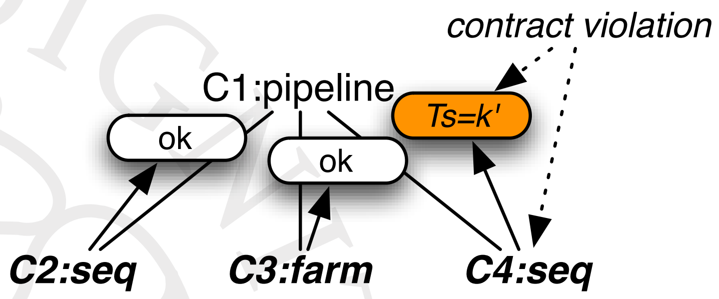
Contract propagation



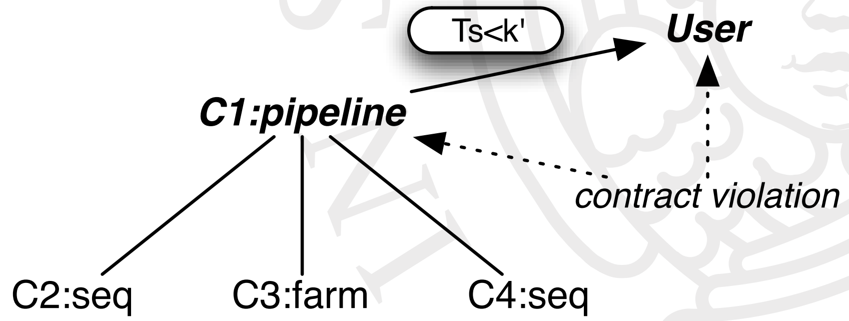
Reporting violations (top level: user)



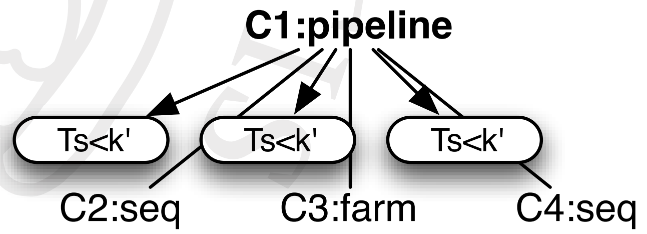
S.1



S.2

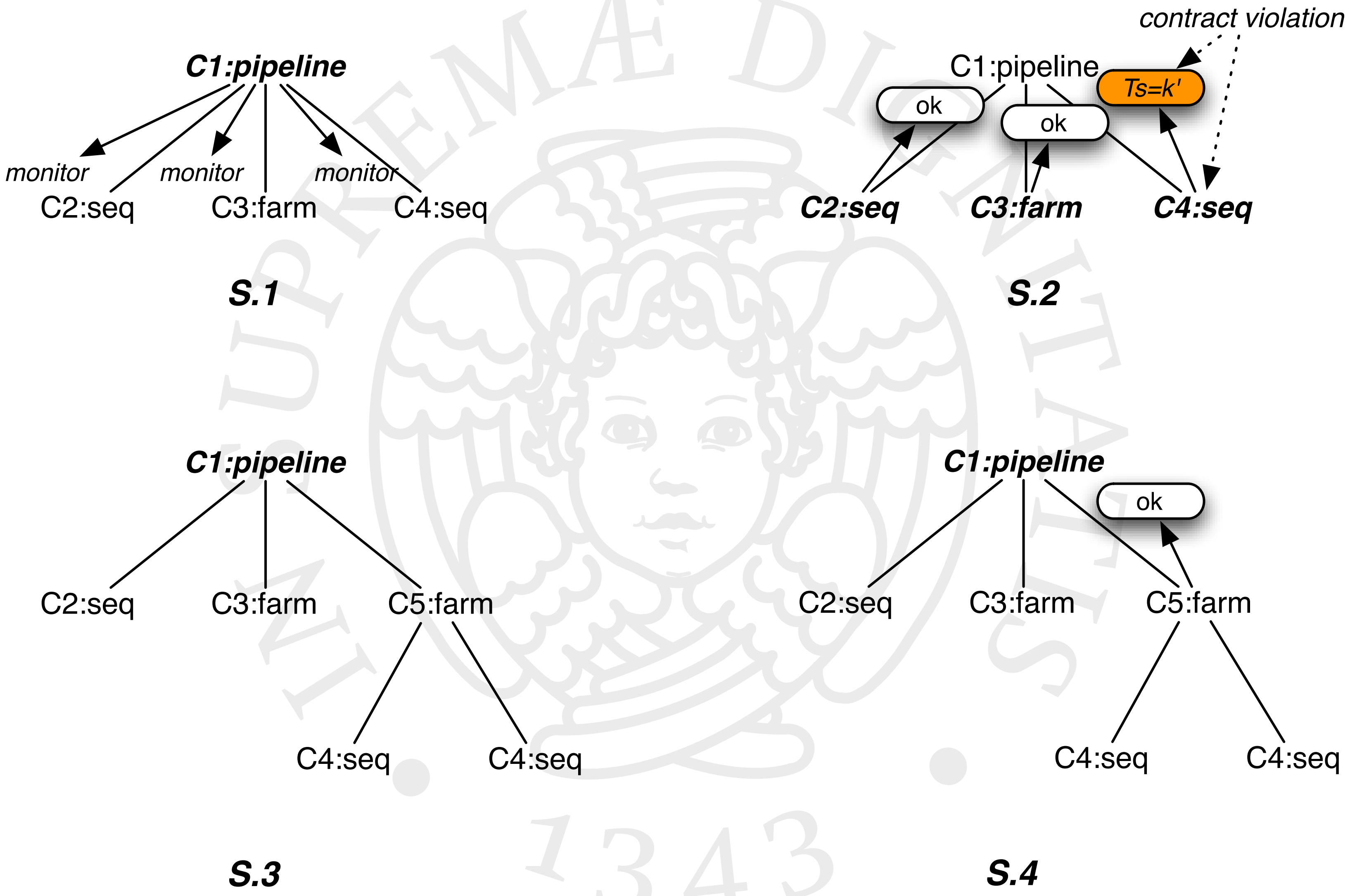


S.3

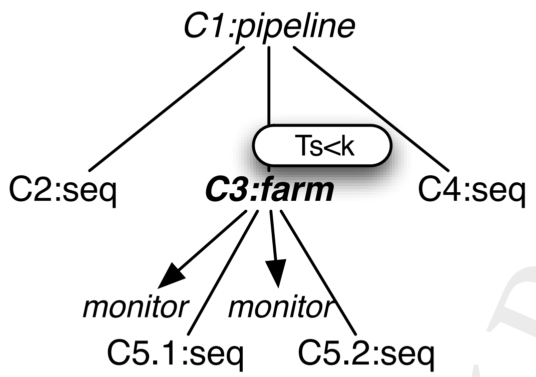


S.4

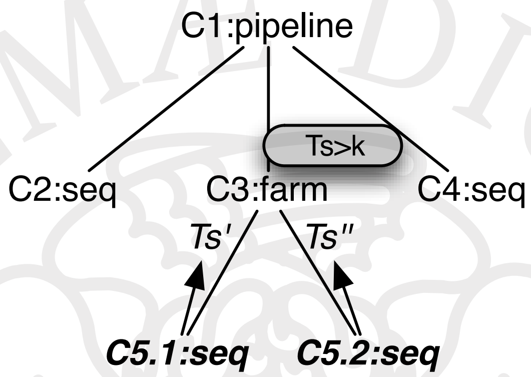
Reporting violations (general manager action)



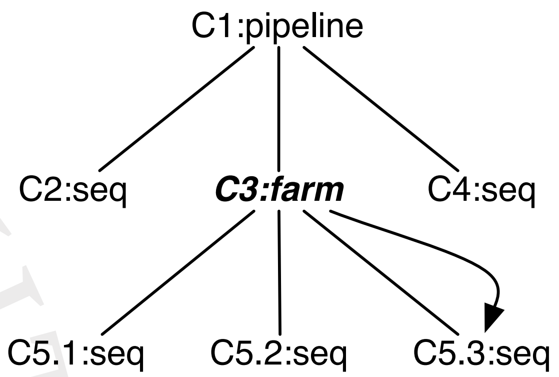
Contract violation (inner manager action)



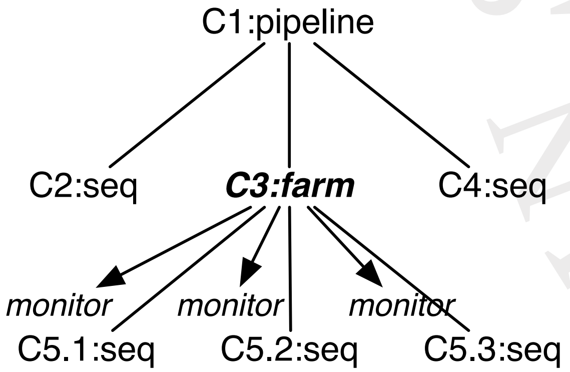
S.1



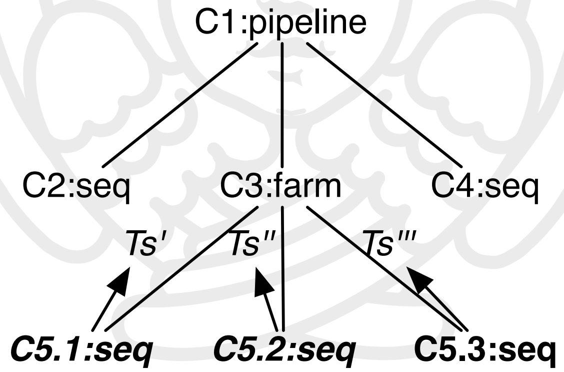
S.2



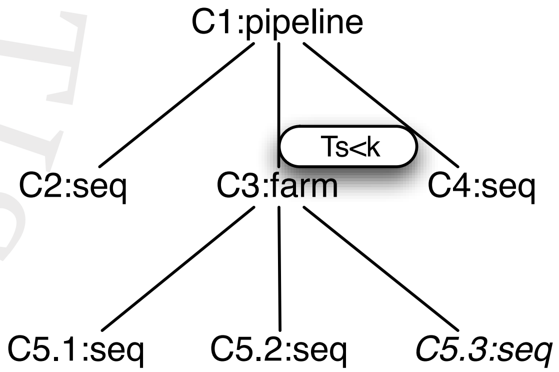
S.3



S.4

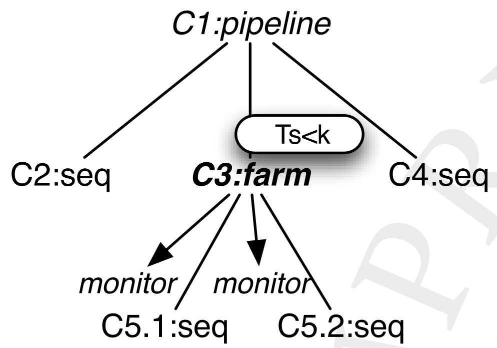


S.5

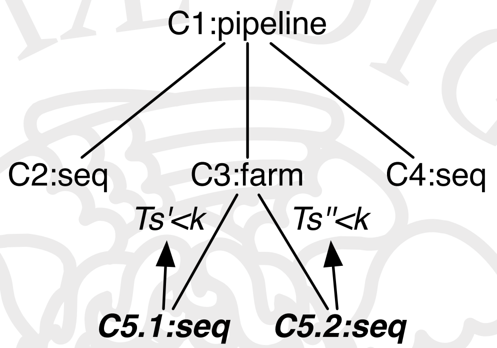


S.6

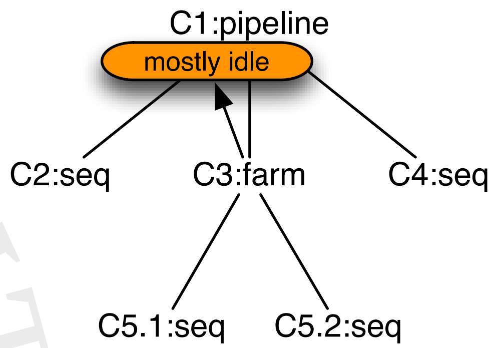
Contract violation (combined action)



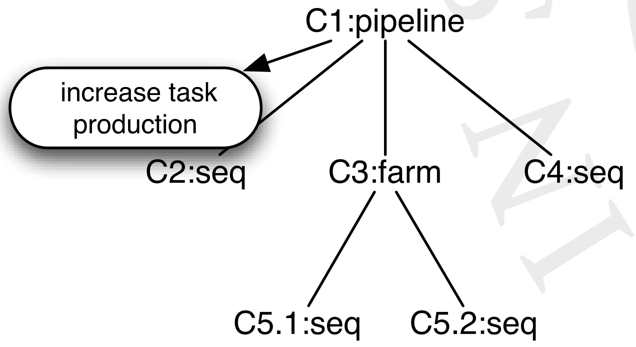
S.1



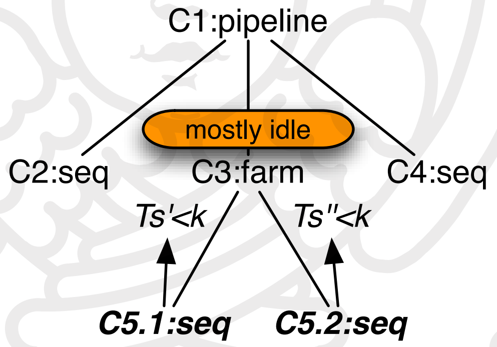
S.2



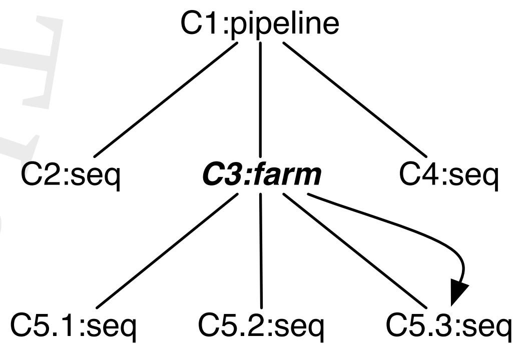
S.3



S.4

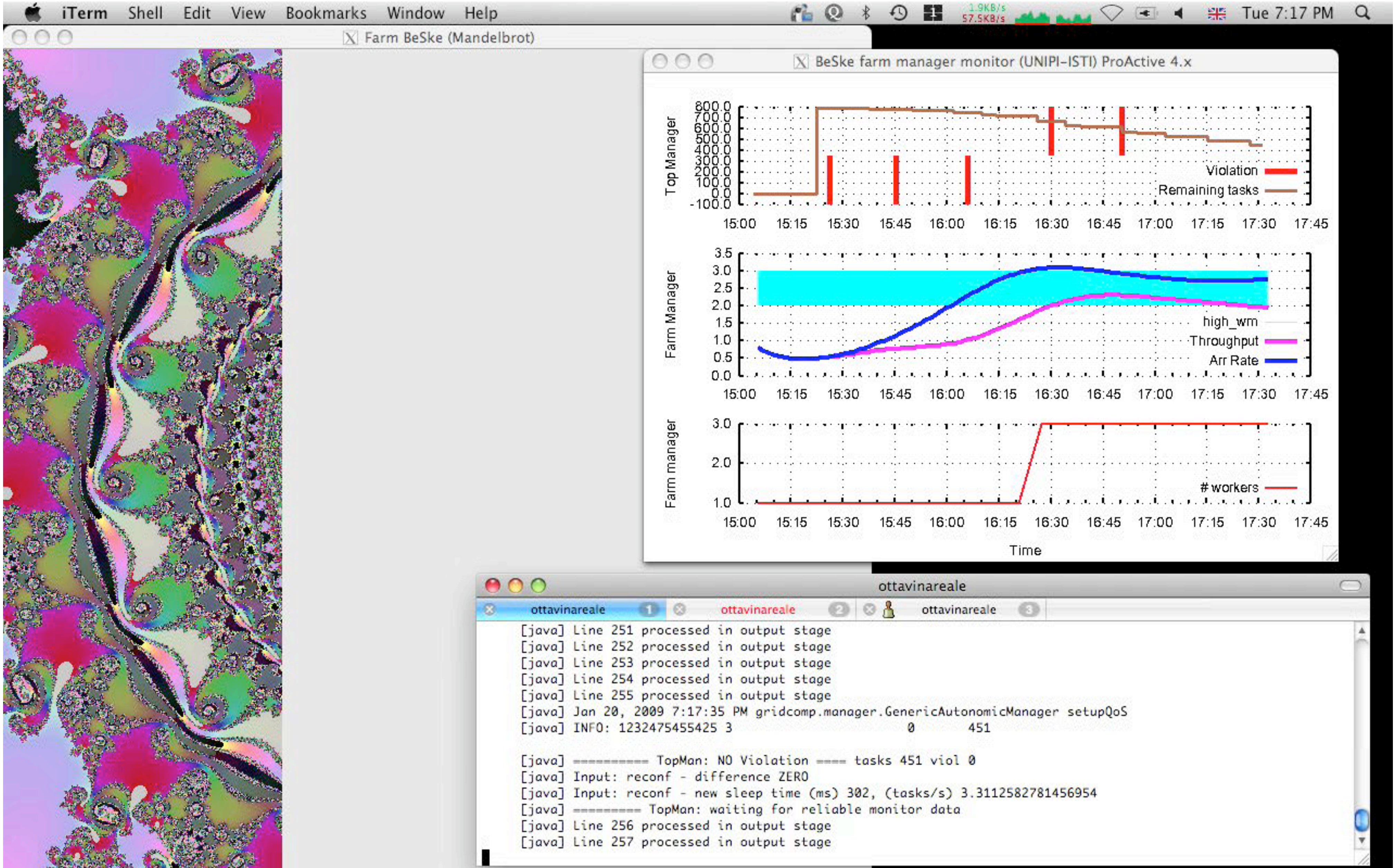


S.5



S.6

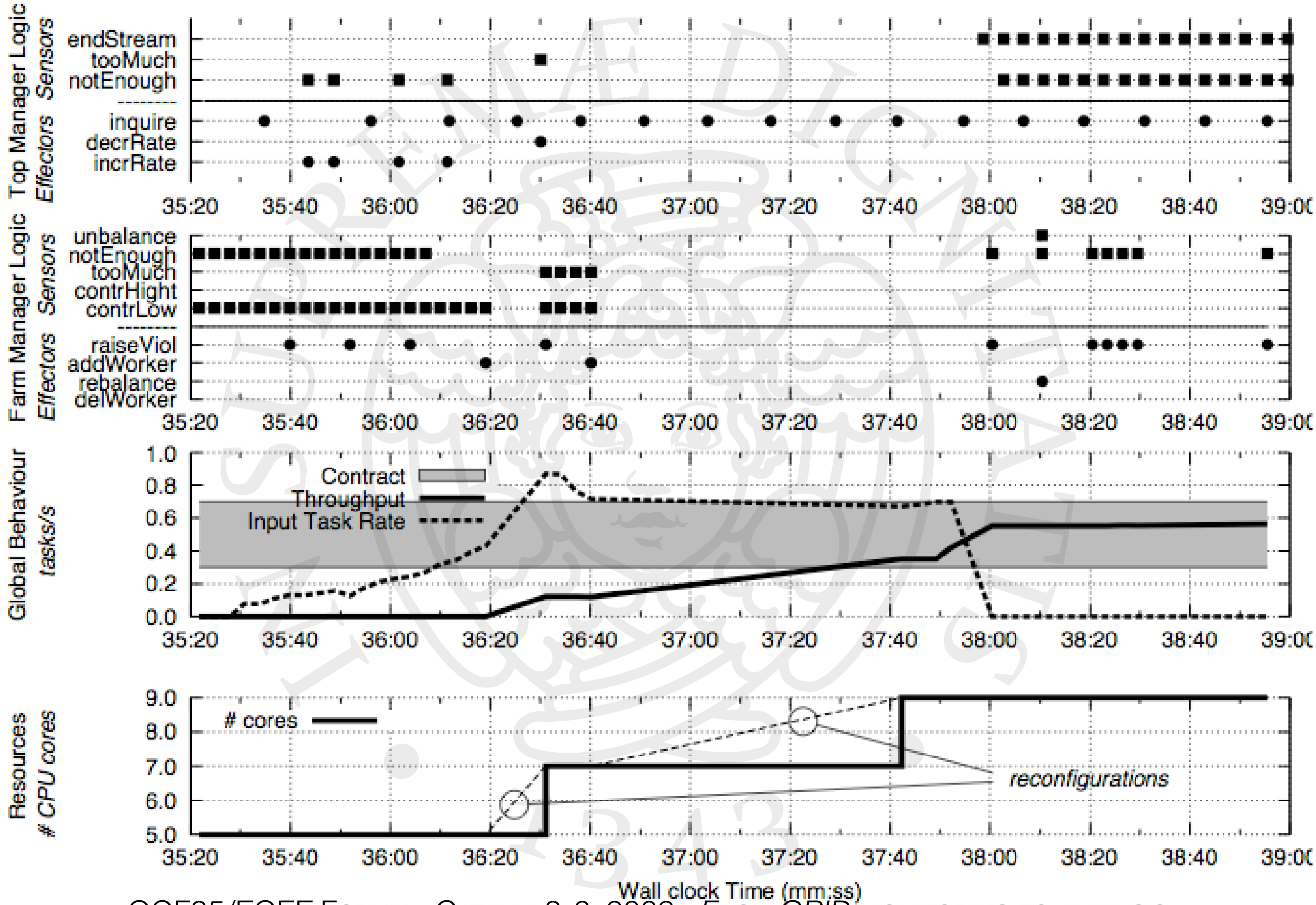
Sample BS run (hierarchical)



```
[java] Line 251 processed in output stage
[java] Line 252 processed in output stage
[java] Line 253 processed in output stage
[java] Line 254 processed in output stage
[java] Line 255 processed in output stage
[java] Jan 20, 2009 7:17:35 PM gridcomp.manager.GenericAutonomicManager setupQoS
[java] INFO: 1232475455425 3                         0       451

[java] ========== TopMan: NO Violation ==== tasks 451 viol 0
[java] Input: reconf - difference ZERO
[java] Input: reconf - new sleep time (ms) 302, (tasks/s) 3.3112582781456954
[java] ========= TopMan: waiting for reliable monitor data
[java] Line 256 processed in output stage
[java] Line 257 processed in output stage
```

Post processed trace ...



Conclusions

joint work with M. Aldinucci (UNITO), P. Kilpatrick (QUB), + S. Campa, P. Dazzi, N. Tonellotto, G. Zoppi (UNIPI + ISTI/CNR-PI)

- Autonomic management
  - effective control of performance concerns
  - related to typical dynamic features of grids
  - responsibility moved from application to system programmers
- Behavioural skeleton encapsulate
  - parallel pattern + autonomic management
  - demonstrated effective for single parallel pattern and hierarchical composition of patterns

Any questions?

gridcomp.ercim.org

marcod@di.unipi.it