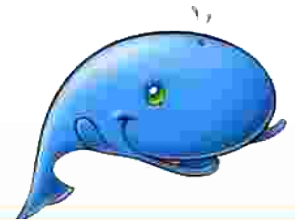




*JASMINe* makes the management of java ee cluster easier

*Benoit.Pelletier@bull.net*

*Laurent.Ruaud@serli.com*



<http://jasmine.ow2.org>

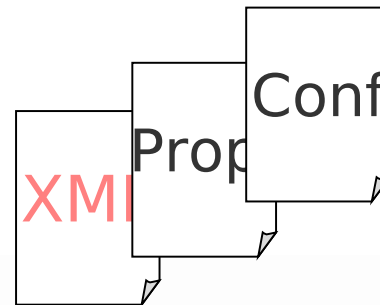
# Agenda

- Rationale
  - Towards autonomic system
  - JASMINe Design/Deploy
  - JASMINe Monitoring
  - JASMINe Self-management
  - To conclude



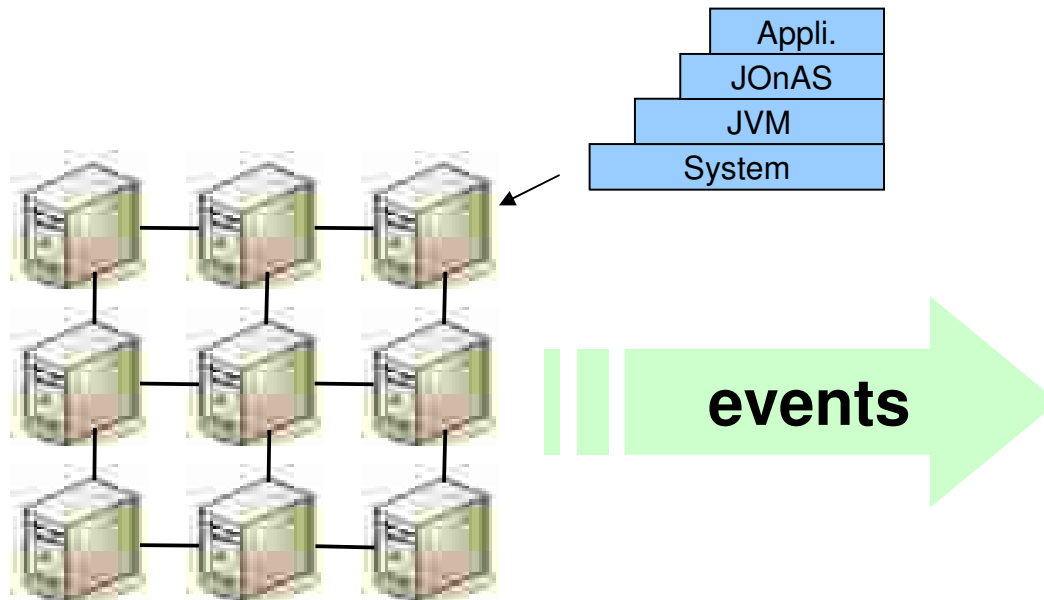
# Complex distributed system : configuration is a difficult task

- Distributed configuration
  - SOA platform => Numerous complex software, each being an assembly of different modules
  - Java EE Cluster => several nodes
- Configuration facilities are generally proprietary (heterogeneous files, XML, properties, ...)
- Numerous files and parameters per node
  - JOnAS = 46 files, Apache configuration file =160 lines



# Complex distributed system : monitoring is a difficult task

- Huge number of events to monitor
  - Distributed architecture
  - Multiple levels

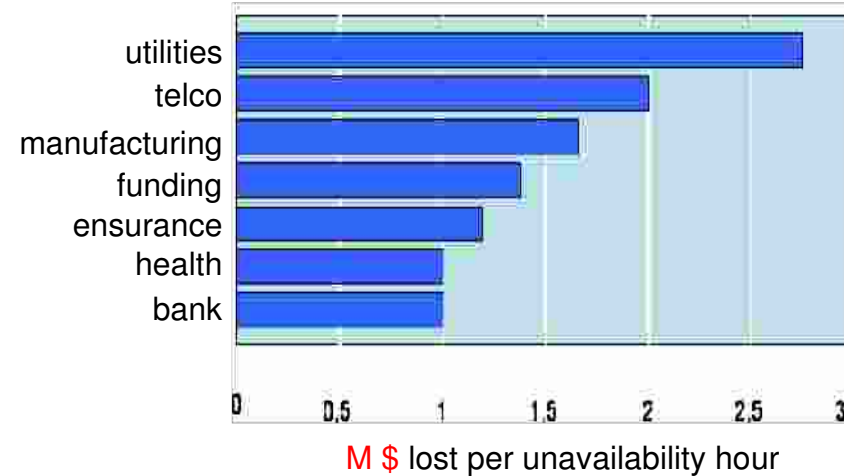


**!!! Not manageable by humans !!!**



# Consequence

- Human actions are error prone
  - 40% of the interruptions of service
  - Service interruption cost
- Low reactivity
- Administration tasks costly
  - Consumes a lot of resources
    - Human resources : Required skills are rare, > hardware cost
    - Material resources : Overbooking (estimation in the worst case)



# Strong requirements for a smart tool

- Simplify the administration tasks
- Reduce the errors (avoid human errors)
- Improve the reactiveness/QoS
  - And so the applications availability
- Minimize the human and material resources cost



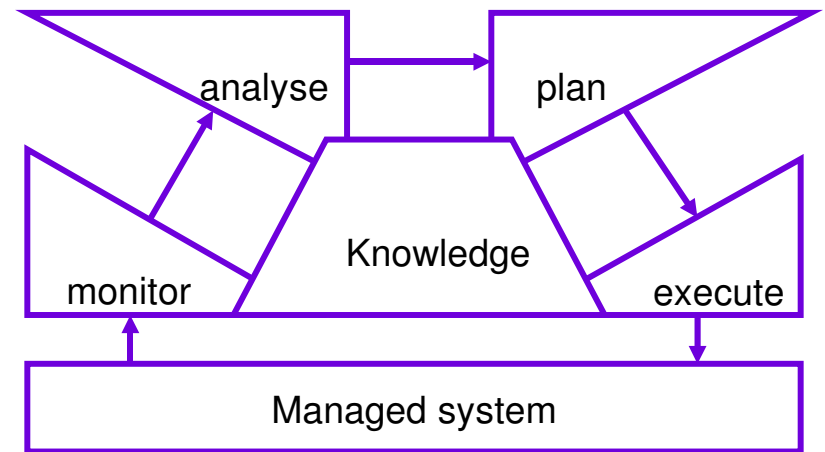
# Agenda

- Rationale
- Towards autonomic system
- JASMINe Design/Deploy
- JASMINe Monitoring
- JASMINe Self-management
- To conclude



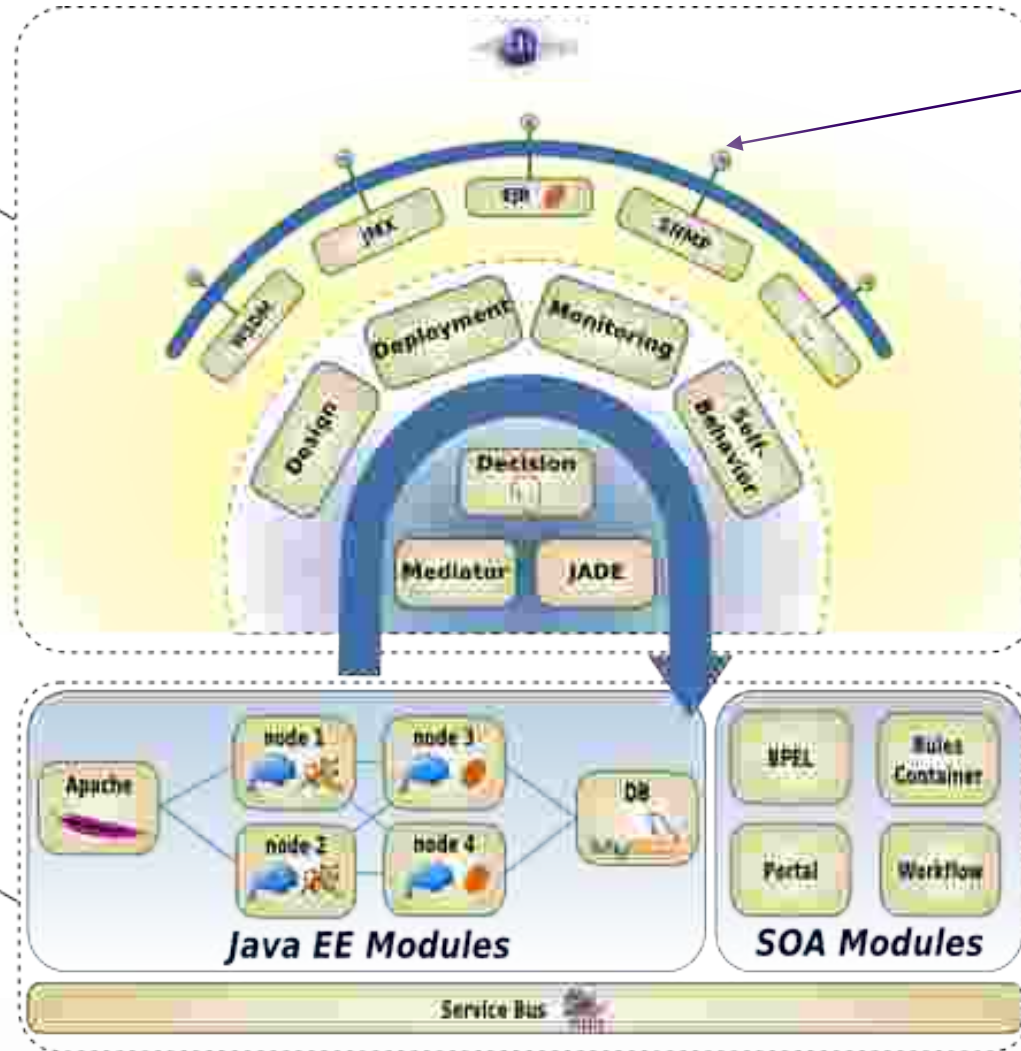
# Autonomic system

- Concept initiated by IBM in 2001
- According to the Butler Group analysts, will change the face of infrastructure softwares
- Scope
  - Control loop with Jade framework (INRIA)



# JASMINe, big picture

 JASMINe



**Nagios**

**SOA Platform**

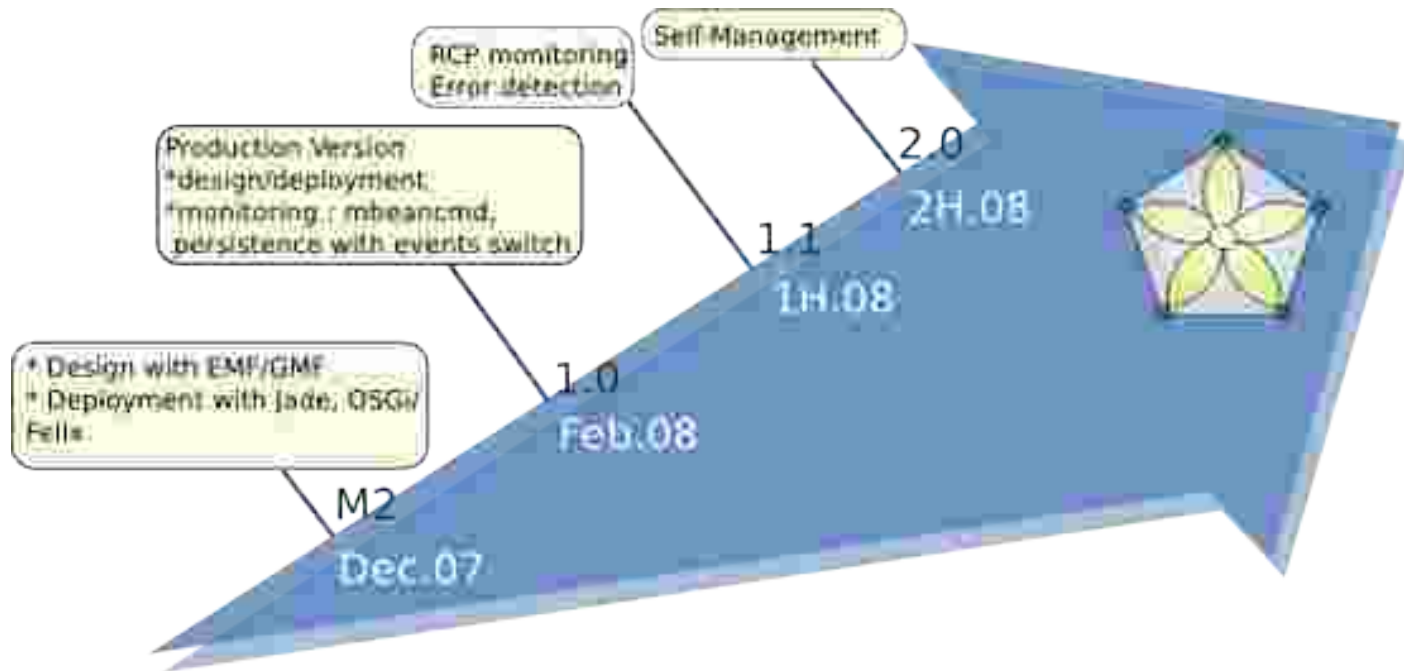
**Java EE Modules**

**SOA Modules**

Service Bus

**BULL**

# Roadmap



# Agenda

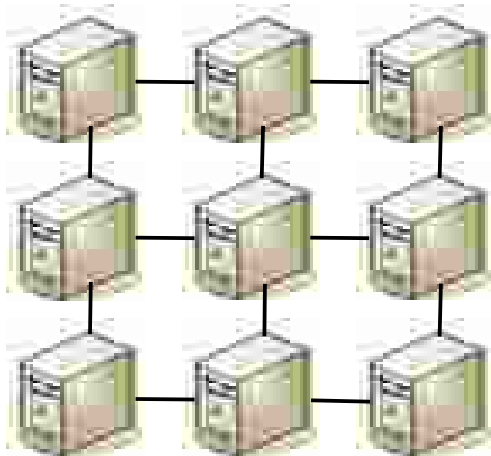
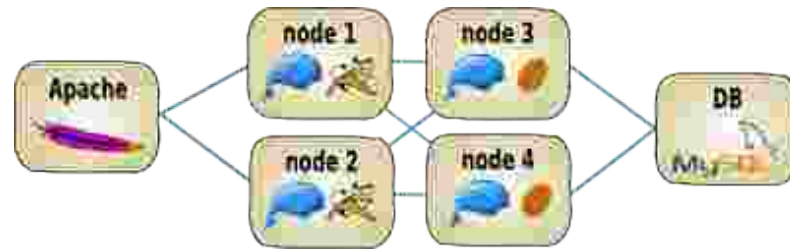
- Rationale
- Towards autonomic system
- JASMINe Design/Deploy
- JASMINe Monitoring
- JASMINe Self-management
- To conclude



# Principles

- Eclipse EMF/GMF for describing the cluster configuration (RCP GUI)
- Jade/OSGi for deploying the configuration across the infrastructure

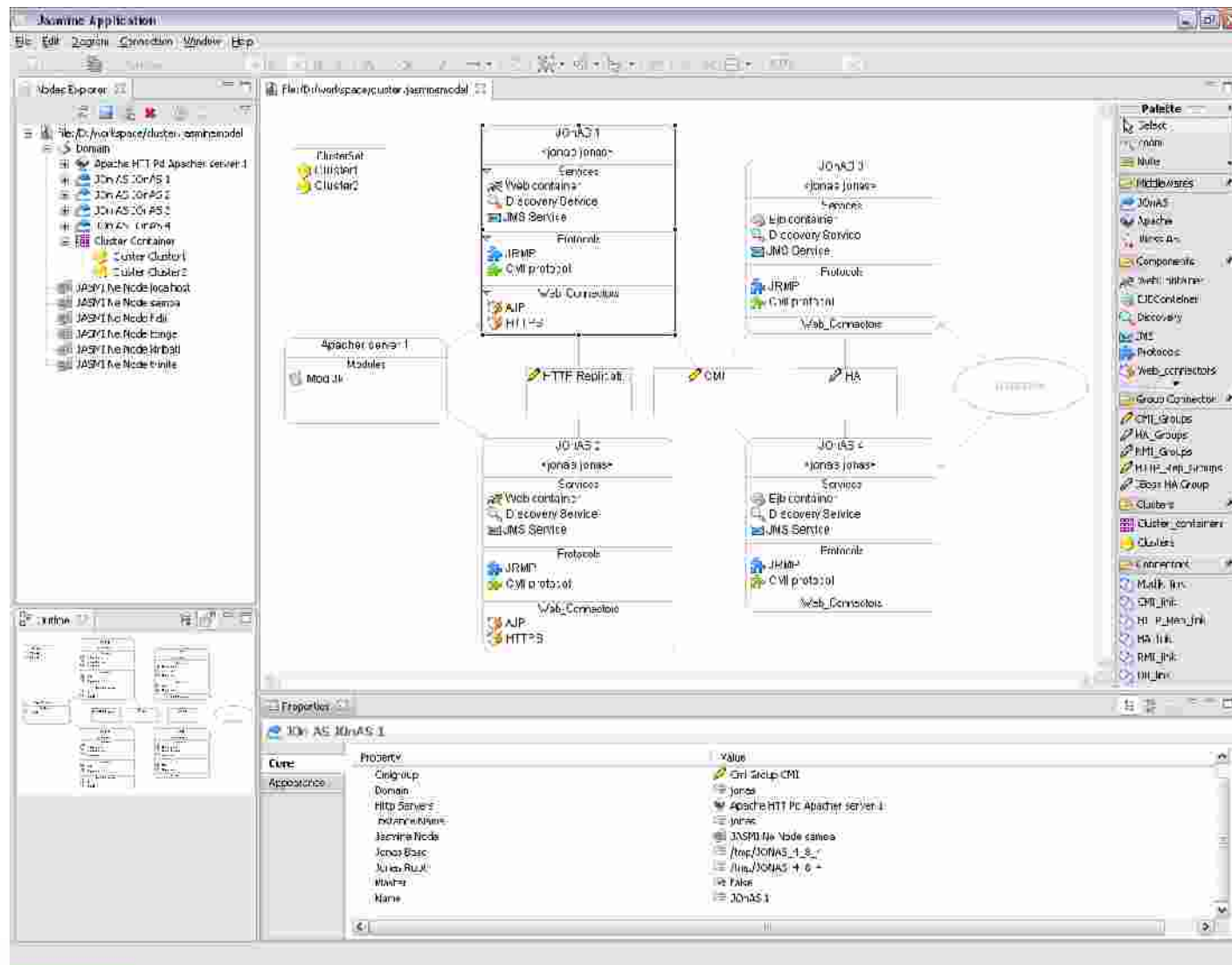
**1. Describe the middleware configuration**



**2. Deploy the middleware configuration**

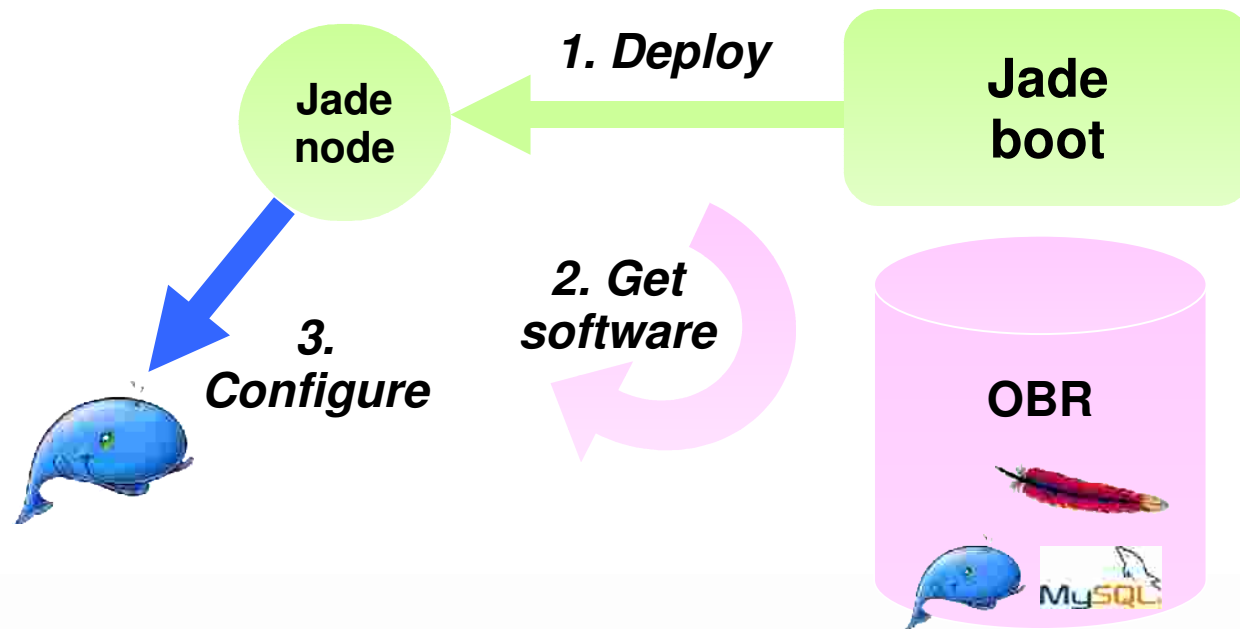


# Design Panel with Eclipse GMF/EMF



# Deployment with Jade/OSGi

- Fractal component model
- OSGi for deploying the configuration across the infrastructure



# Agenda

- Rationale
- Towards autonomic system
- JASMINe Design/Deploy
- JASMINe Monitoring
- JASMINe Self-management
- To conclude



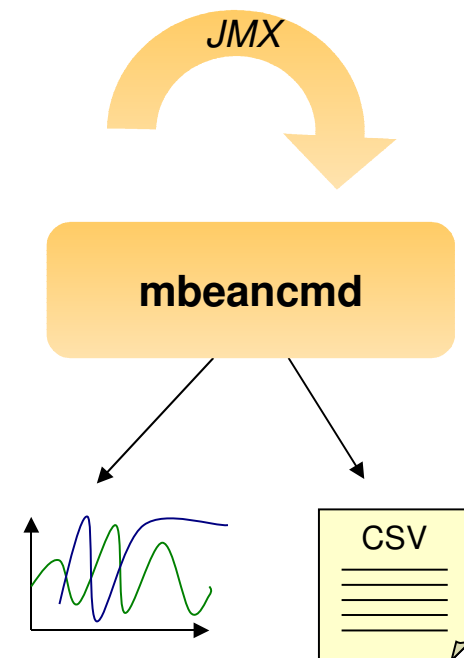
# Monitoring activities

- **Realtime monitoring**
  - Check the system health, performance analysis, error detection
  - High reactivity
  
- **Tactic monitoring**
  - Check the system behavior over a long period of time
    - Leaks (memory, threads, connections, ...)
    - Bottlenecks
  - Low reactivity (a few days/weeks)
  
- **Strategic monitoring**
  - Capacity planning
  - Very low reactivity (a few months/years)



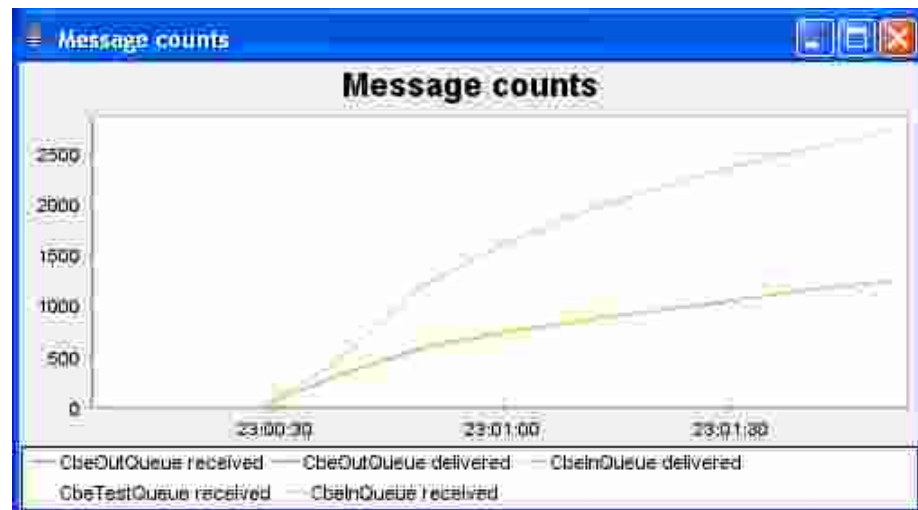
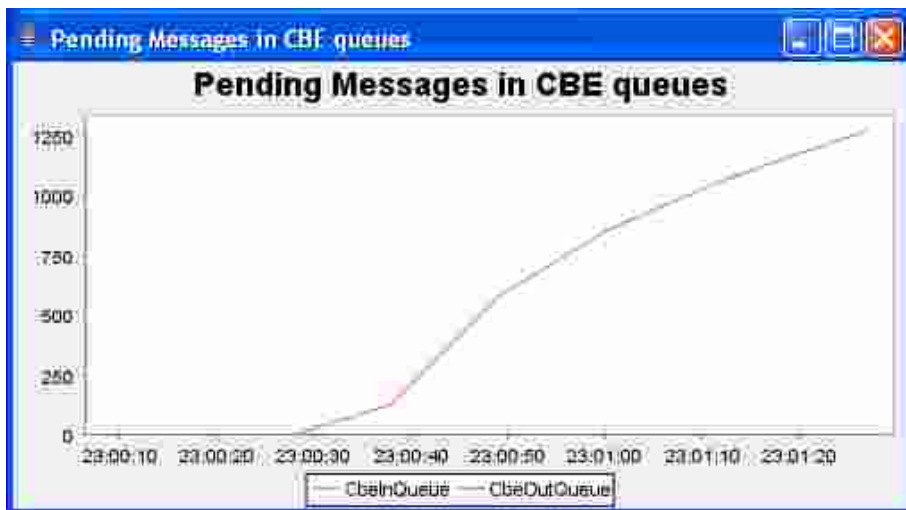
# MBeancmd for monitoring the JOnAS MBeans

- Available in JASMINe 1.0
- Java command (mbean.jar), enables scripting
- Relies on the JMX Remote interface
- Capability to :
  - get and set MBean attributes
  - invoke MBean methods
  - poll any MBean
- Shortcuts for probing the most relevant JOnAS indicators
  - Tx,datasource, http connectors, threads pool, jms statistics, ...
- File storage (CSV) or database
- Replay mode
- Graphic console



# MBeancmd, example (1)

```
java -jar mbean.jar stat -name "joramClient:type=queue,*" \  
-a AdminName PendingMessages PendingRequests Statistic \  
-graph graphQueues.xml -f queues.log
```



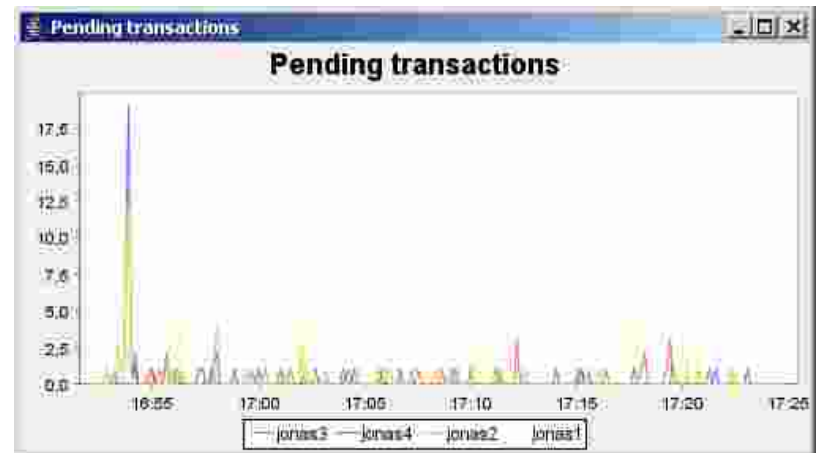
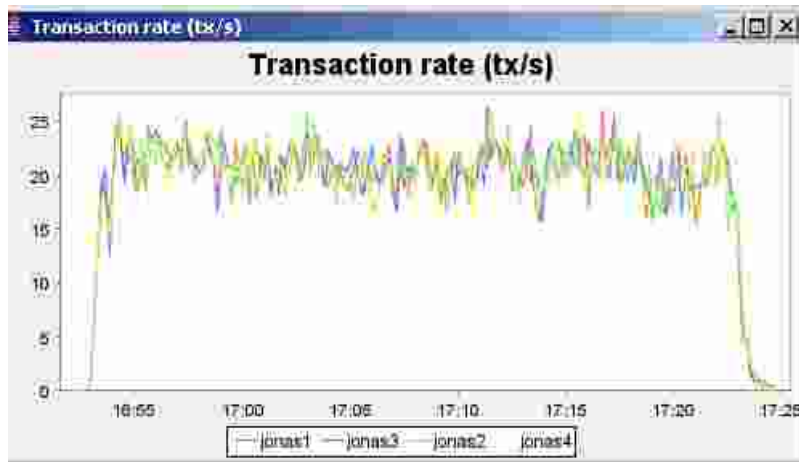
```
java -jar mbean.jar replay -i queues.log -graph graphQueues.xml
```



# MBeancmd, example (2)

Measure transaction throughput balanced over 4 JOnAS instances

```
java -jar mbean.jar poll -tx -target jonas1 jonas2 jonas3 jonas4 \  
-graph graphTx.xml -f tx.log
```



```
java -jar mbean.jar replay -i tx.log -graph graphTx.xml
```



# MBeancmd, example (3)

## ■ Change transaction timeout

```
java -jar mbean.jar mbean \  
-name "dom:j2eeType=JTAResource,name=JTAResource,J2EEServer=jonas1" \  
-set timeOut java.lang.Integer 120 ,
```

## ■ Resize DataSource connection pool

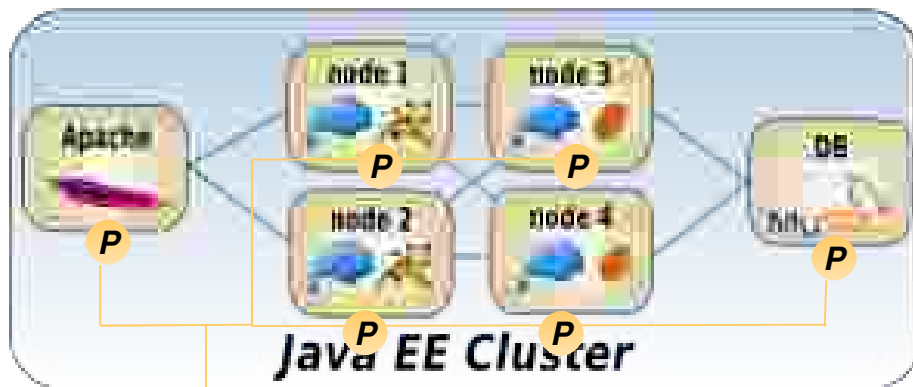
```
java -jar mbean.jar mbean -name \  
"dom:j2eeType=JDBCDataSource,name=ORA1,JDBCResource=JDBCResource,J2EEServer=j1" \  
-set jdbcMaxConnPool java.lang.Integer 50 ,
```

## ■ Other actions

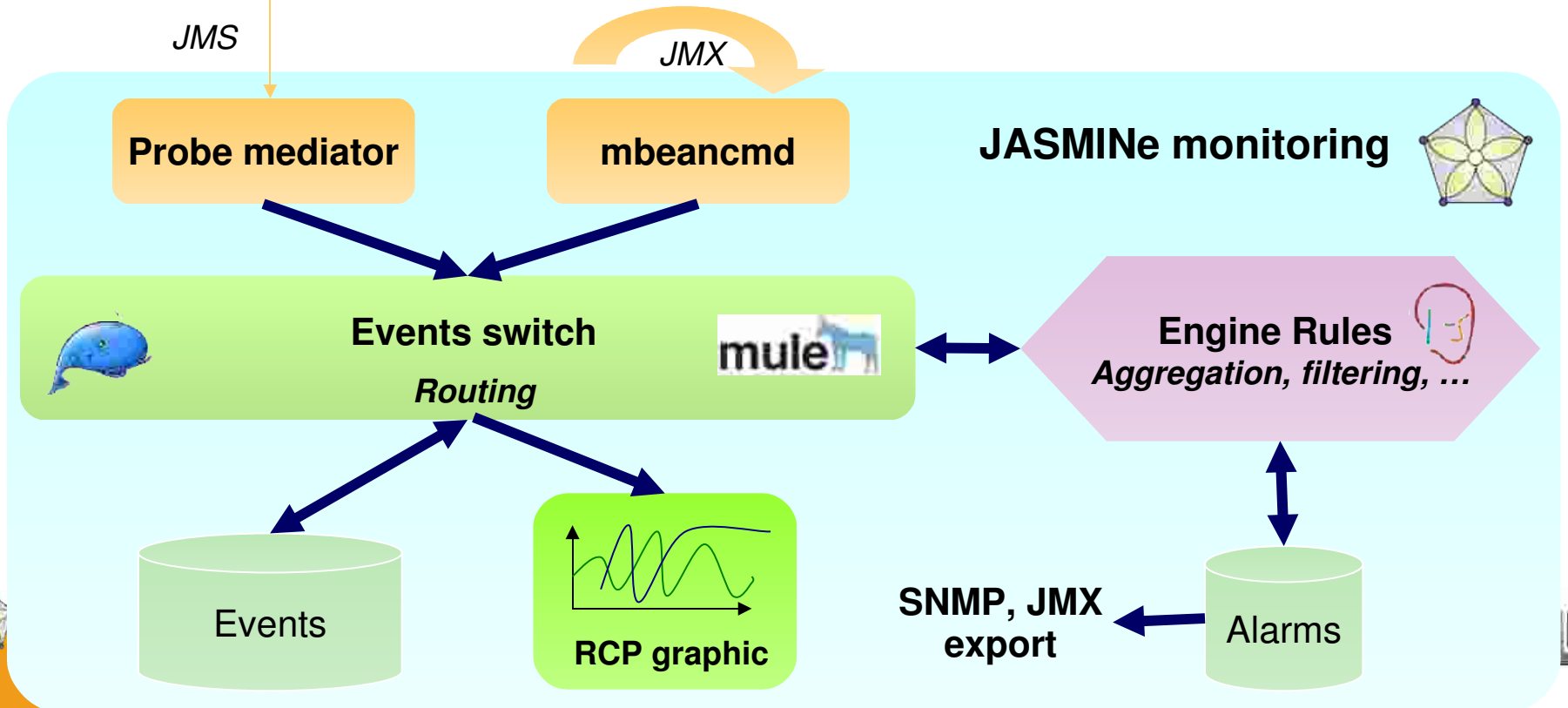
- Invalidate an Http session
- Retrieve the principal of a Http session
- Etc ...



# Monitoring architecture : events processing

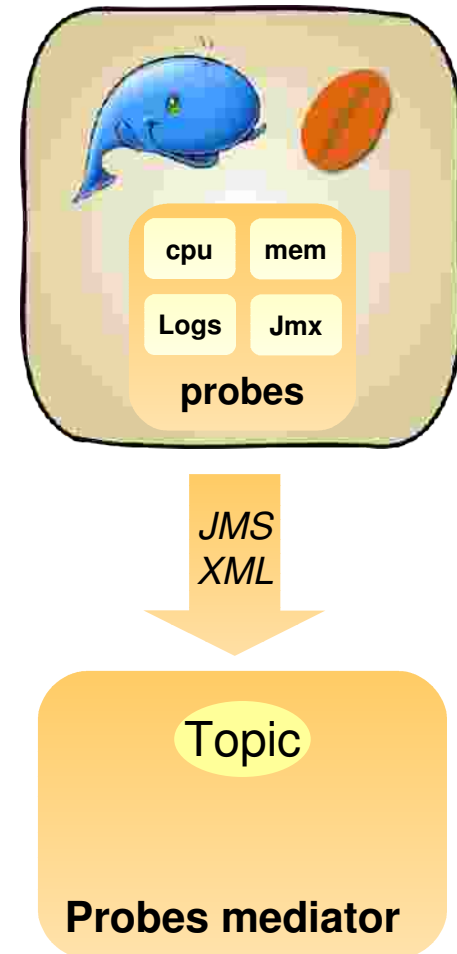


- Events switch & events db available in 1.0
- Alpha version Engine Rules, probes, alarms export available in 1.0
- RCP graphs in 1.1



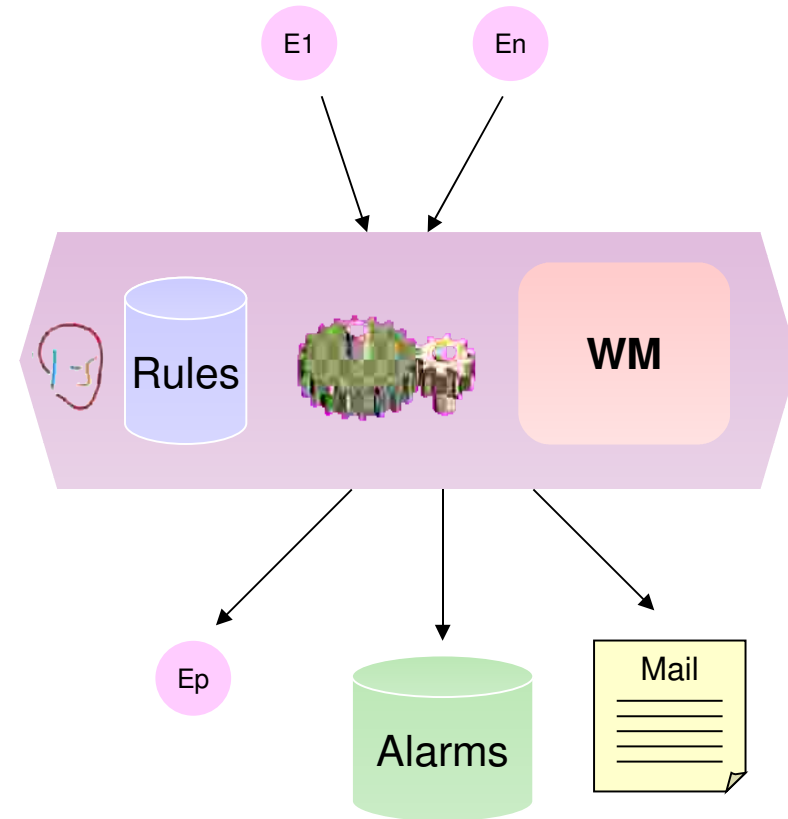
# Probes infrastructure

- Different levels
  - System (CPU, Memory, ...)
  - JVM (Mem Heap, CPU, ...)
  - J2EE (error log, ejb instances, datasource,tx,...)
  - Other (MySQL, Apache, ...)
- JMS interface
  - Through JORAM (TCP,SOAP,SSL)
  - Aggregation in XML buffer



# Error detection

- Relies on the engine rules Drools
- Enables to implement the user's management policy
- A few rules examples :
  - Error logs aggregation
  - Counter aggregation
  - Cpu overload detection over a significant period (ignore peak load)
  - Memory saturation is close
  - Datasource bottleneck
  - Alarms burst filtering
  - Multiple error events correlation for determining the in-fault component
- Actions : alarm generation, mail sending, ... may be extended



# Agenda

- Rationale
- Towards autonomic system
- JASMINe Design/Deploy
- JASMINe Monitoring
- JASMINe Self-management
- To conclude

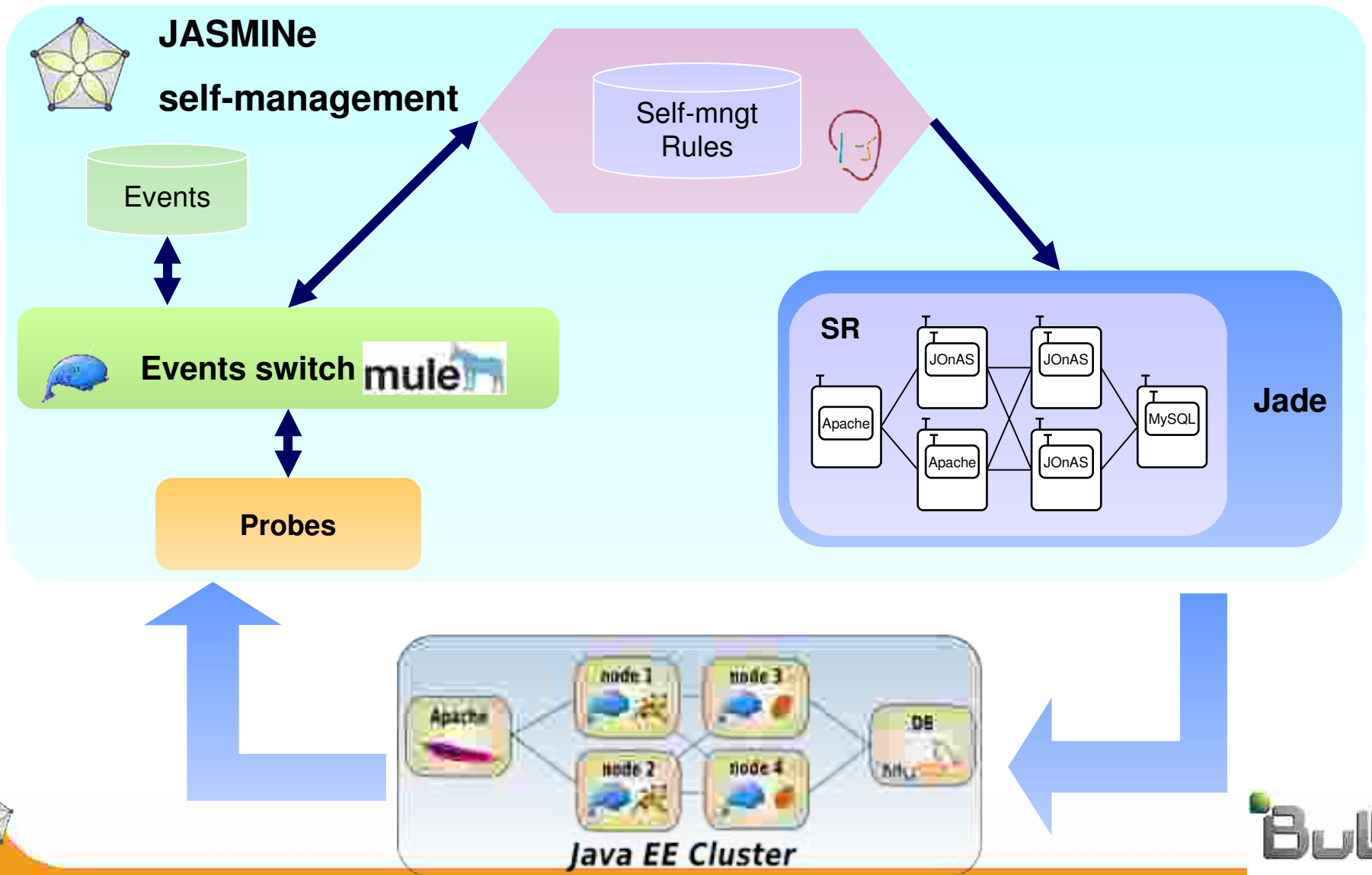


# Principles

- Add some autonomic behaviour for managing the system
- Relies on the Jade framework
  - Architecture-Based management approach
    - High level abstraction to model and control the managed system (legacy) and the management system
  - Component model (Fractal - Julia)
    - Wrap legacy managed resource (software/hardware)
    - An architectural view of a legacy system
    - Deployment, Introspection and Reconfiguration mechanisms
    - A naming system, A typed system, Ensure distributed configuration consistency for legacy complex system
  - **System representation causally connected**
- Relies on JASMINe deploy for the actuators
- Relies on JASMINe monitoring for the sensor

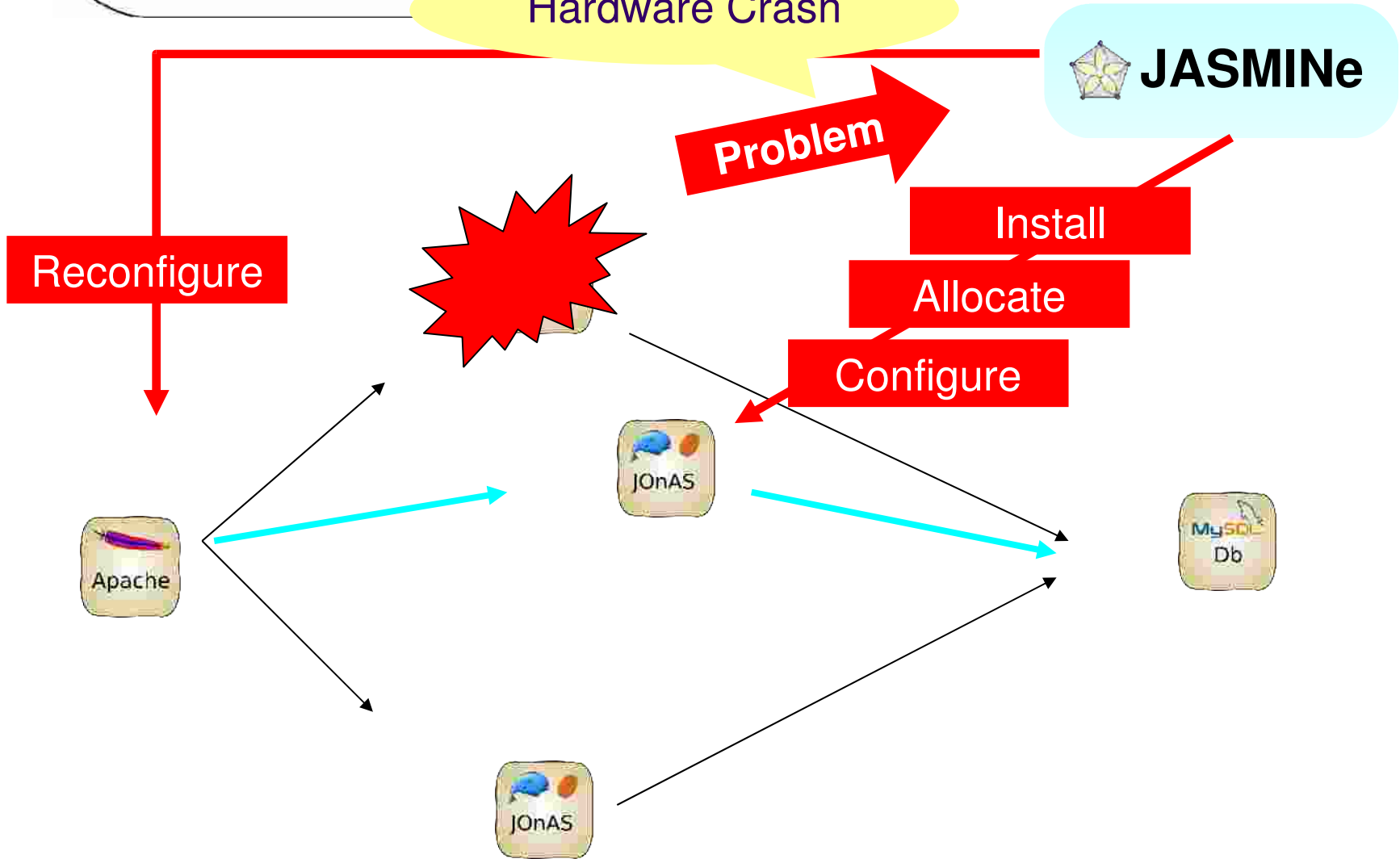


# Architecture

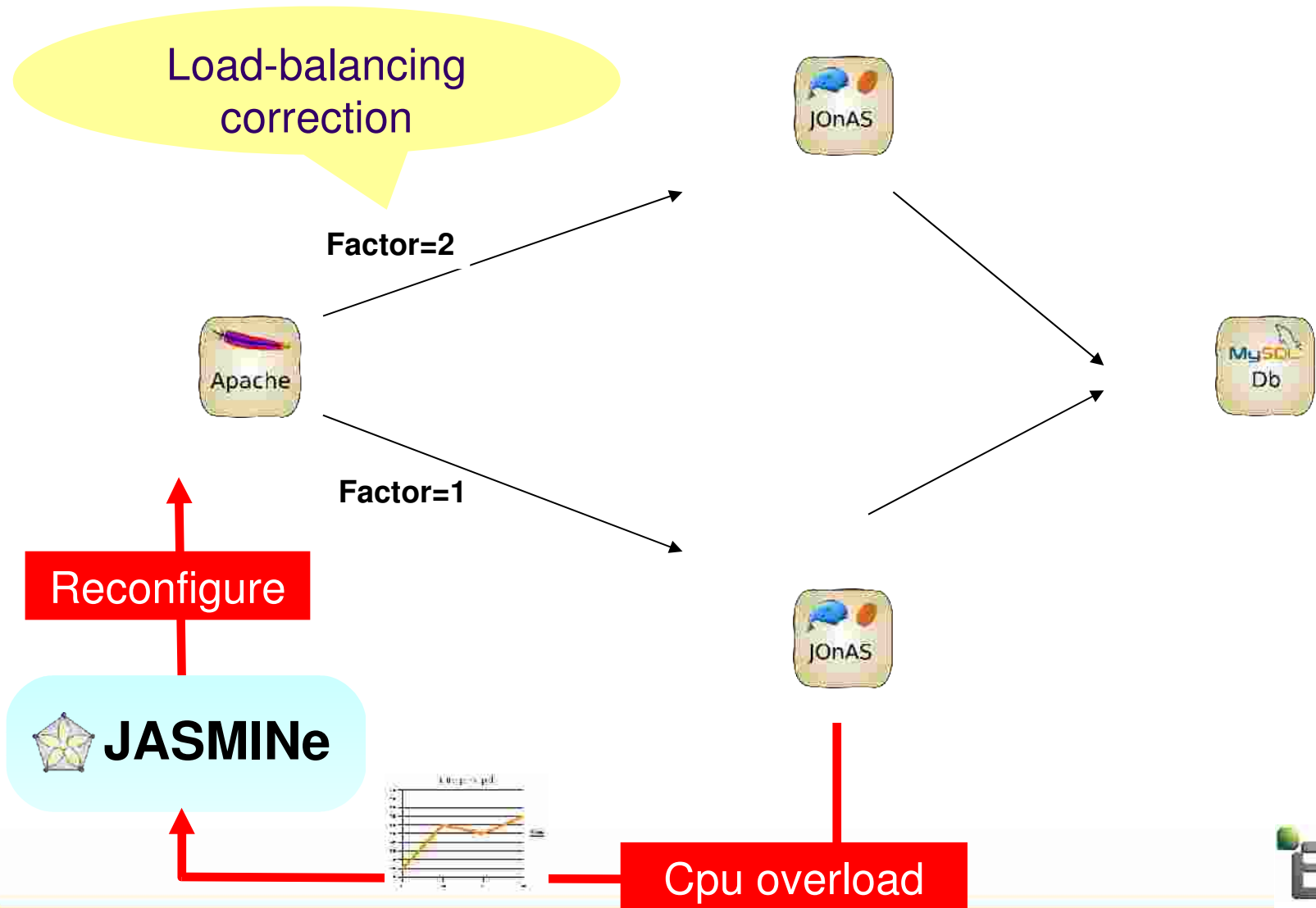


# Use case– self repair

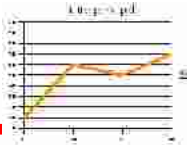
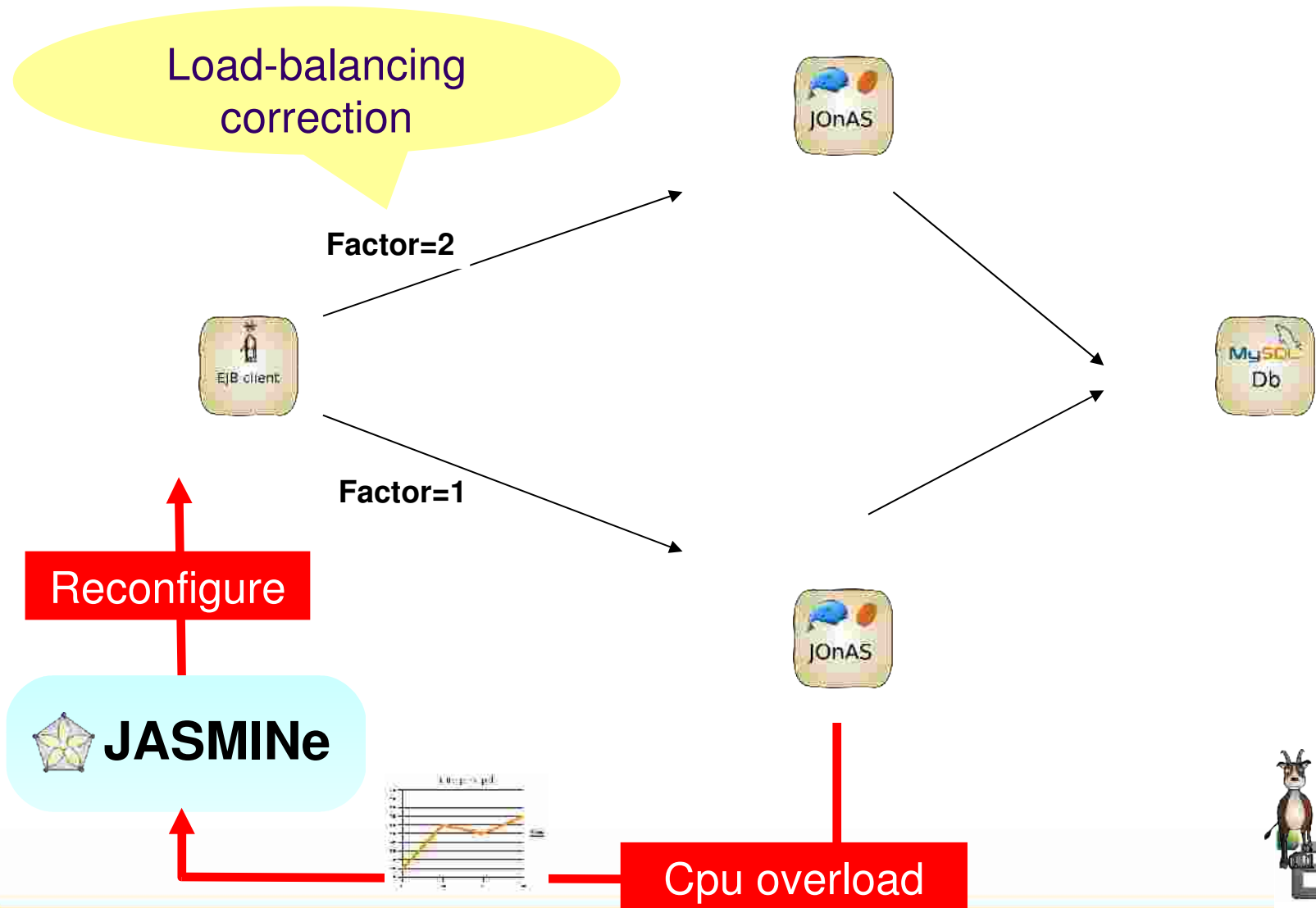
Hardware Crash



# Use case– LB self optimization (jk)



# Use case– LB self optimization (cmiv2)



# Agenda

- Rationale
- Towards autonomic system
- JASMINe Design/Deploy
- JASMINe Monitoring
- JASMINe Self-management
- To conclude



# The next steps

- Project
  - Mavenization
- External interfaces
  - Add WSDM
- Probes & monitoring
  - RCP graphs & ER
  - Collaboration with the CLIF project
- Self-management
  - SR integration (MIB Jade)
  - Rules writing (+workflow)
- Decision support
  - spagoBI integration
- Add others SOA modules
  - ESB
    - Collaboration with spagic project (engineering)
    - Collaboration with Petals project (EBM Websourcing)
  - And then BPEL (Orchestra), Workflow (Bonita), Axis2, ER, Portal, ...
- Others SOA stacks support
  - *JBoss*, *Glassfish*, *Geronimo*

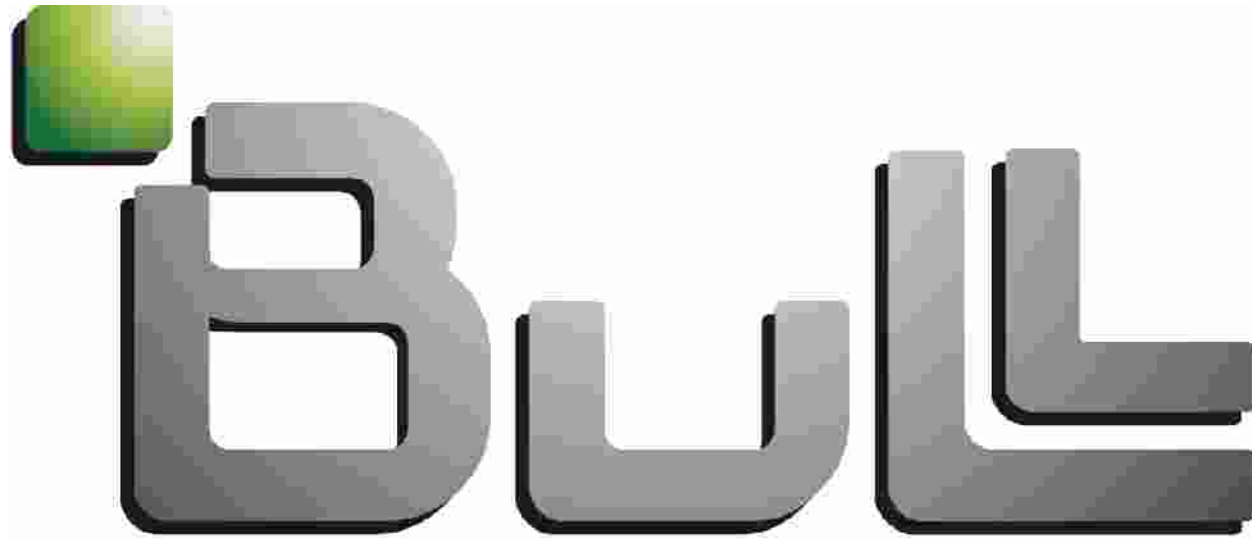
stay tuned !!!



Visit the web site

<http://jasmine.ow2.org>



The logo for BULL features the word "BULL" in a bold, grey, sans-serif font with a black outline. The letter "B" is significantly larger than the other letters. A small, square icon with a green-to-yellow gradient is positioned above the top left corner of the "B".

**BULL**

Architect of an Open World™