

# Grid Component Model and



# Bridging Distributed and Multi-Core Computing

Denis Caromel, et al.

<http://ProActive.ObjectWeb.org>

OASIS Team

INRIA -- CNRS - I3S -- Univ. of Nice Sophia-Antipolis, IUF

October 14<sup>th</sup>, CGW, Kraków, Poland

1. Background: INRIA, OASIS, ActiveEon
2. ProActive Parallel Suite & Active Objects
3. Components and Standardization (GCM)
4. Applications and Perspectives: SOA+GRID



# 1. Background



# INRIA and OASIS Team



Computer Science and Control

- 8 Centers all over France
- Workforce: 3 800
- 186 Million Euro annual budget
- Strong in standardization committees:
  - IETF, W3C, ETSI, ...
- Strong Industrial Partnerships
  
- Foster company foundation:  
90 startups so far
  - Ilog (Nasdaq, Euronext)
  - ...
  - ActiveEon



A joint team between:

- INRIA, University of Nice- CNRS
- Created in 1999
- Started the *ProActive Parallel Suite*
- Over 40 persons
- Distributed and Parallel:  
From Multi-cores to Enterprise GRIDs





ACTIVEeon  
SCALE BEYOND LIMITS



## Startup Company Born of INRIA

Co-developing, Providing support for Open Source [ProActive Parallel Suite](#)

Winner 80/1000 applications (Minister of Research Contest)

Several Customers (Worldwide: Boston USA, etc.)



# 2. ProActive Parallel Suite & Active Objects



# ProActive

## Parallel Suite



- Written in Java
- Features:
  1. Eclipse GUI
  2. Parallel+Dist. Progr.
  3. Scheduling & Grids

Used in  
production by  
industry



# ProActive Parallel Suite

## PROGRAMMING

### Java Parallel Frameworks

for HPC, Multi-Cores, Distribution, Enterprise Grids and Clouds.

Featuring: Async. comms, Master-Worker, Monte-Carlo, SPMD, components and legacy code wrapping.

## OPTIMIZING

### Eclipse GUI (IC2D)

for Developing, Debugging, Optimizing your parallel applications.

Featuring: graphical monitoring and benchmarking with report generation.

## SCHEDULING

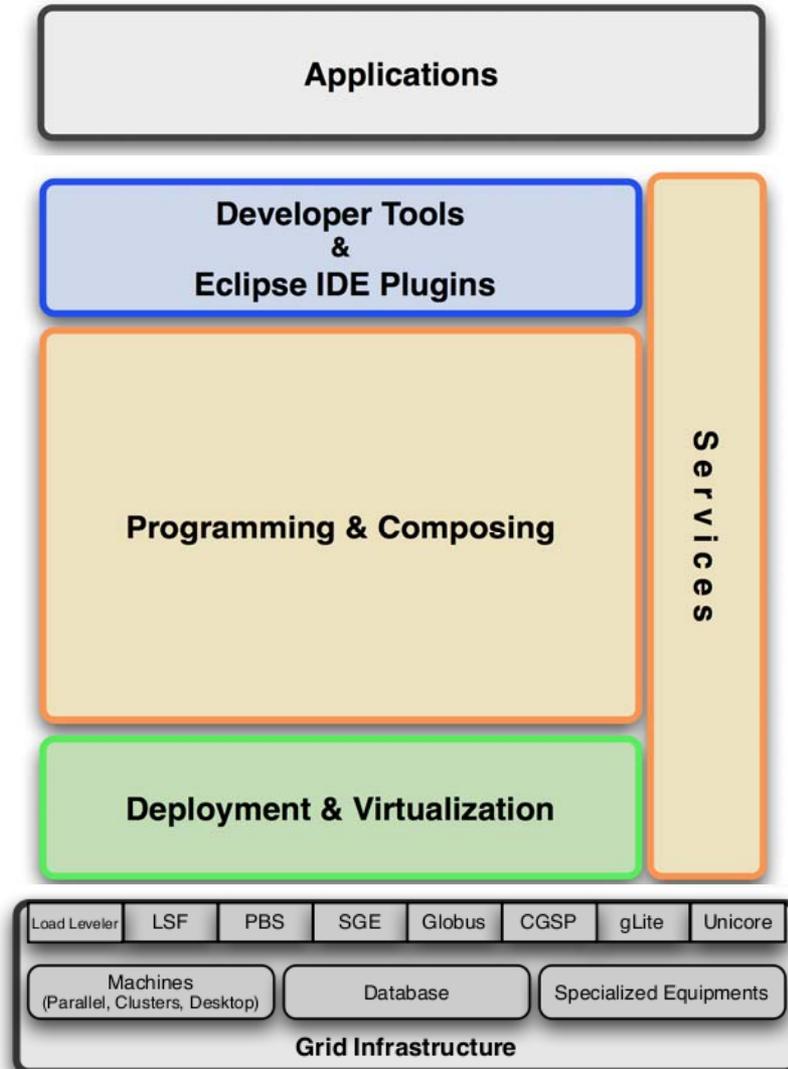
### Multi-Language Scheduler

for Workflows made of C, C++, Java, Scripts, Matlab, Scilab tasks.

Featuring: graphical user interface, resource acquisition and virtualization.



# ProActive Parallel Suite (1)



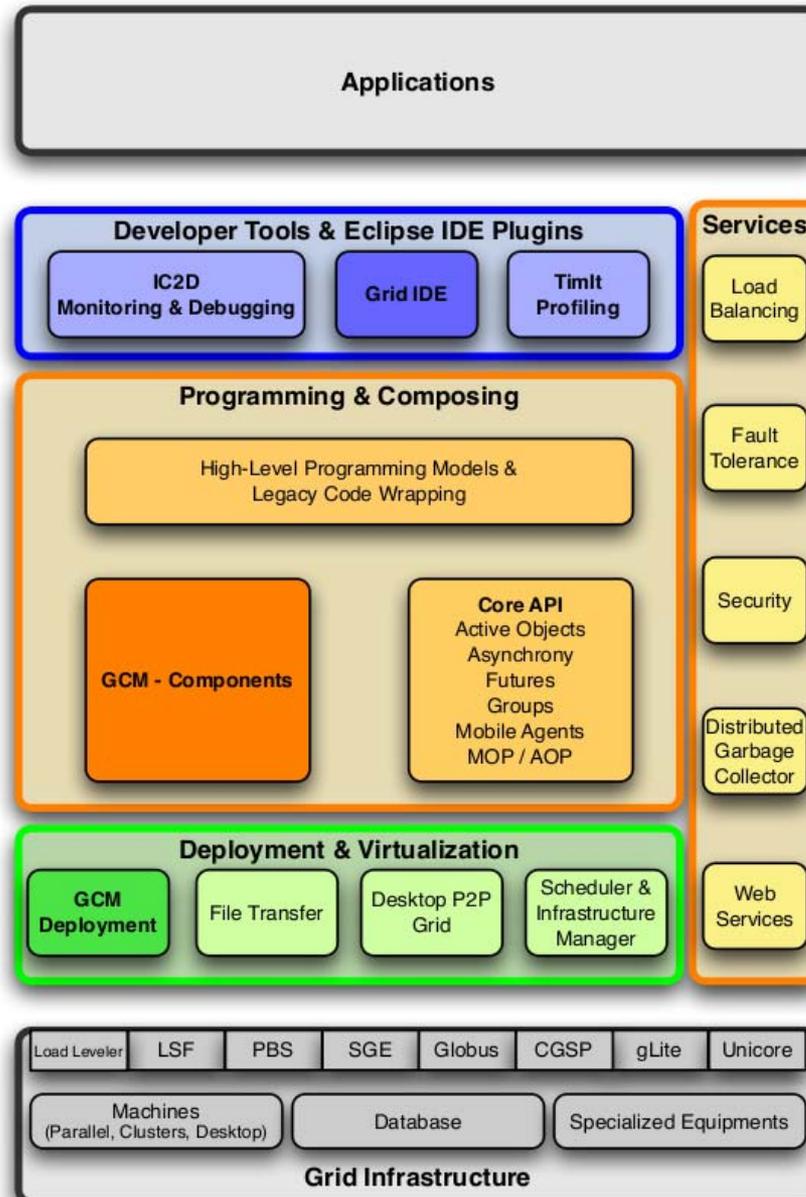
**Open  
Source**

**+**

**PROFESSIONAL  
SUPPORT**



# ProActive Parallel Suite: GUI



# ProActive Parallel Suite: GUI



## Monitoring View

## Job Monitoring View

The screenshot displays the Eclipse IDE's Monitoring View, which is split into two panes: 'Monitoring' and 'Job Monitoring'.

**Monitoring View (Left Pane):**

- Virtual nodes:** Includes checkboxes for 'Renderm', 'DefaultVN', 'Dispatcher', and 'User'. 'Renderm' is checked.
- Topology:** A hierarchical diagram showing a 'Node Node60562498...' containing 'DinnerLayout#2', 'Table#3', and 'Philosopher#4' through '#8'. These are connected to various 'Node' objects (e.g., 'Node Renderm-127...', 'Node Dispatcher-5...', 'Node User16026446...', 'Node Renderm1307...') which are further connected to 'C3D' objects (e.g., 'C3DRendering...', 'C3DDispatche...', 'C3DUser#13', 'C3DRendering...').
- Bottom Controls:** Includes radio buttons for 'Display topology', 'Proportional', 'Ratio', and 'Filaire' (selected). There is a 'Reset Topology' button and a 'Monitoring enable' checkbox.
- Console:** Shows a log entry: '15:09:15 => NodeObject id=Node-455186381 already monitored, ckeck for new active objects'.

**Job Monitoring View (Right Pane):**

- Legend:** A tree view showing the hierarchy of jobs and nodes. It includes 'DefaultVN (JOB-1357457629)', 'bebita.inria.fr:1099:OS un', 'PA\_JVM1357457629\_', 'Node Node60562498...', 'DinnerLayout#2', 'Table#3 (JOB-13', 'Philosopher#4 (J', 'Philosopher#5 (J', 'Philosopher#6 (J', 'Philosopher#7 (J', 'Philosopher#8 (J', 'sidonie.inria.fr:1099:OS u', 'Dispatcher (JOB--167207649', 'User (JOB--294719007)', 'bebita.inria.fr:1099:OS un', 'PA\_JVM-294719007\_J', 'Node User1602644', 'C3DUser#13 (J', 'Renderer (JOB--1672076495', 'bebita.inria.fr:1099:OS un', 'PA\_JVM-1631909824\_'. A red box highlights 'Node Node-455186381'.



# IC2D

The screenshot displays the IC2D monitoring application interface, which is divided into several main sections:

- Monitoring#1 (Virtual nodes):** A central diagram showing a network topology. It includes nodes like PA\_JVM1820960857 (Node Node-632703901), Node\_matrixNode15..., OctTree#2, Domain#3, Domain#4, Domain#5, Domain#6, Maestro#7, and BioMaestro#8. There are also nodes for PA\_JVM1949849146 (Node Node-2003411204, Displayer#1) and PA\_JVM1370729570 (Node Node-557274178).
- Legend:** A panel on the right side defining the visual encoding for active objects (e.g., Active by itself, Serving request, Waiting for request), pending requests (1, 5, 50), nodes (RMI, HTTP, RMI/SSH), and JVMs (Standard, started with Globus).
- Timer Tree View:** A table showing performance metrics for different domains.
 

Name	Time [ms]	Total [%]	Invocations	Parent [%]
Domain#5				
Total	142212.28	100.00	1	0.00
WaitForRequest	21627.76	15.21	2056	15.21
Serve	120543.91	84.76	5352	84.76
SendReply	0.00	0.00	0	0.00
WaitByNecessity	17050.55	11.99	5340	14.14
SendRequest	101773.58	71.56	16054	84.43
Domain#4				
Total	142228.27	100.00	1	0.00
WaitForRequest	21249.88	14.94	2114	14.94
Serve	120936.36	85.03	5353	85.03
SendReply	0.00	0.00	0	0.00
GroupOneWayCall	0.00	0.00	0	0.00
GroupAsyncCall	0.00	0.00	0	0.00
WaitByNecessity	16765.29	11.79	5348	13.86
SendRequest	102320.24	71.94	16057	84.61
Serialization	1101.89	0.77	5352	1.08
LocalCopy	2471.16	1.74	10705	2.42
BeforeSerializati	20631.26	14.51	5352	20.16
- Timt View:** A horizontal bar chart for Domain#4 showing the distribution of time across various operations. The total time is 2.37m. The chart is highlighted with a red border.
 

Operation	Time
GroupAsyncCall	1.28m
GroupOneWayCall	1.10s
AfterSerialization	20.63s
BeforeSerialization	2.47s
LocalCopy	21.24s
WaitForRequest	16.76s
WaitByNecessity	1.70m
SendReply	2.01m
SendRequest	2.37m
Serve	2.37m
Total	2.37m
- Time Line View:** A console window showing a detailed timeline for various components: BigMaestro#8, Maestro#7, Domain#6, Domain#5, Domain#4, Domain#3, and OctTree#2. The x-axis represents time in milliseconds, ranging from 0ms to 746.4ms.



# ChartIt

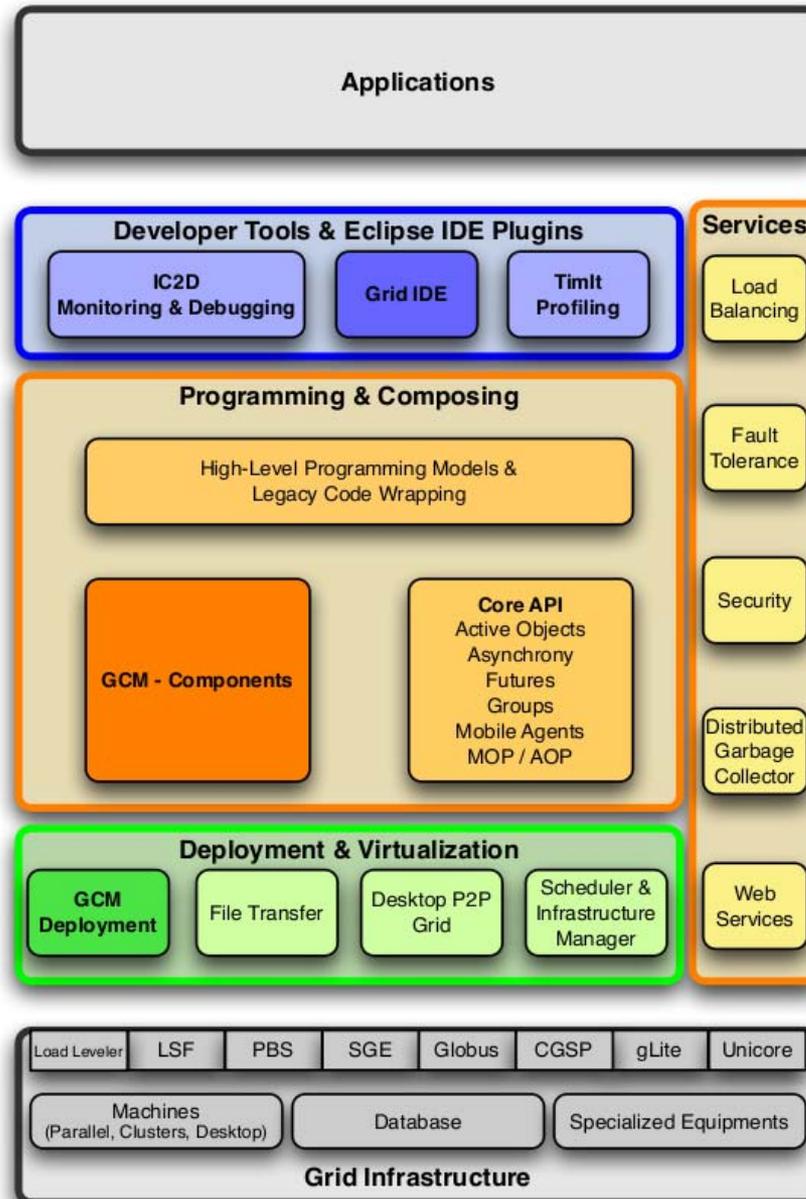


# Video 1: IC2D

## Monitoring, Debugging, Optimizing



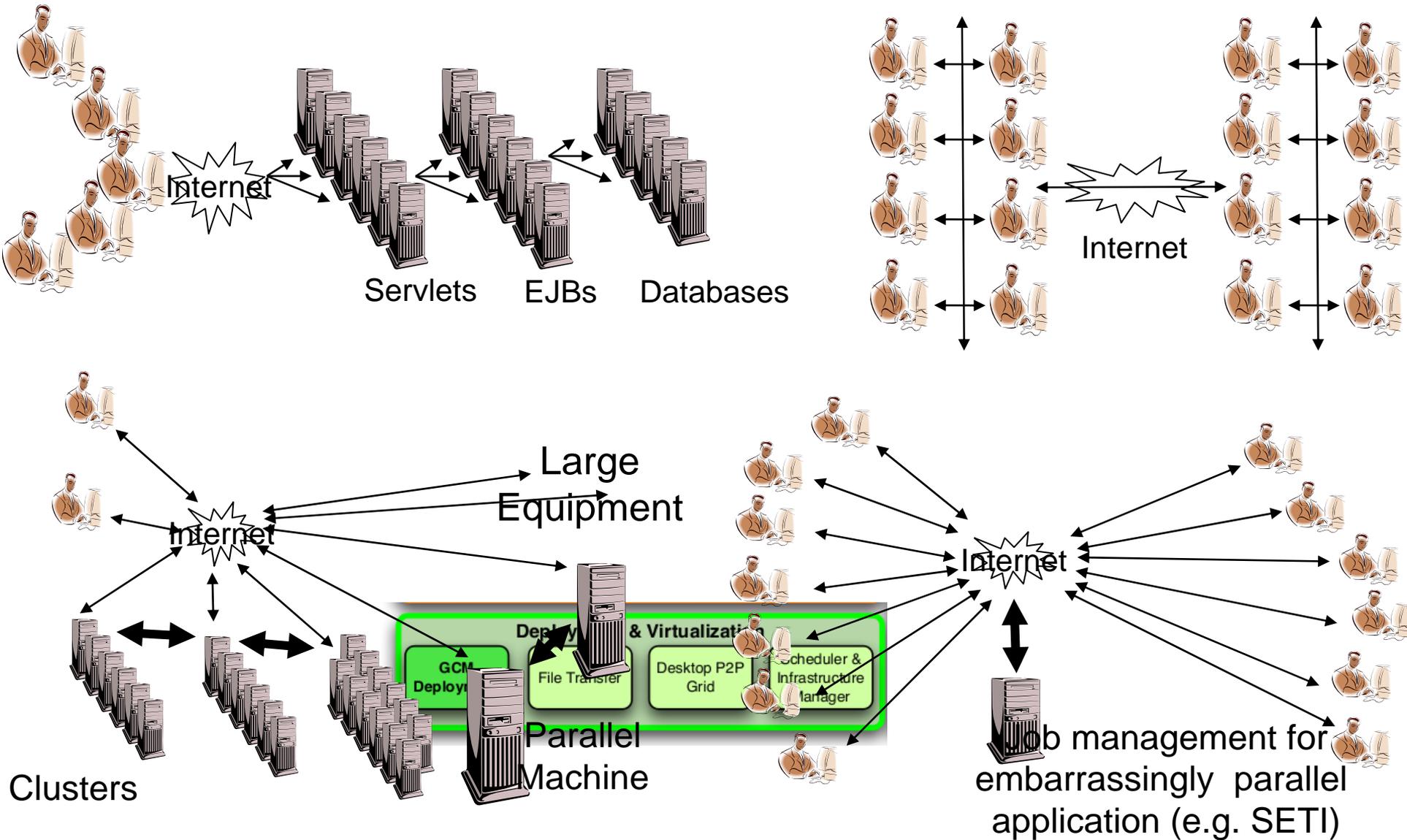
# ProActive Parallel Suite: Deploy



# ProActive Parallel Suite: Deploy



# Deploy on Various Kinds of Infrastructures



# Scheduler and Resource Manager: User Interface



# Scheduler: User Interface

The screenshot displays the ProActive Scheduler interface with three main panels: Pending (8), Running (13), and Finished (11). The Running panel is selected, showing a progress bar and details for job 55. Below the main panels is a 'STARTED' button. At the bottom, there are two sub-panels: 'Console' and 'Job Info'. The 'Console' panel shows a table of tasks for job 55, and the 'Job Info' panel shows the job's properties.

**Pending (8)**

Id	State	User	Priority	Name
172	Pending	user1	Low	job_2_tasks
173	Pending	user1	Low	job_2_tasks
174	Pending	user1	Low	job_2_tasks
176	Pending	user1	Low	job_2_tasks
177	Pending	user1	Low	job_2_tasks
178	Pending	user1	Low	job_2_tasks
179	Pending	user1	Low	job_2_tasks
180	Pending	user1	Low	job_2_tasks

**Running (13)**

Id	State	Progress	# Finishes	User	Priority	Name
54	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
55	Running	<div style="width: 50%;"></div>	0/2	user1	Low	job_2_t
56	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
160	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
161	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
162	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
163	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
164	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
165	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
166	Running	<div style="width: 50%;"></div>	1/2	user1	Low	job_2_t
168	Running	<div style="width: 50%;"></div>	0/2	user1	Low	job_2_t
169	Running	<div style="width: 50%;"></div>	0/2	user1	Low	job_2_t
170	Running	<div style="width: 50%;"></div>	0/2	user1	Low	job_2_t

**Finished (11)**

Id	State	User	Priority	Name
152	Finished	user1	Low	job_2_tasks
167	Finished	user1	Normal	job_2_tasks
171	Finished	user1	Normal	job_2_tasks
153	Finished	user1	Low	job_2_tasks
175	Finished	user1	Normal	job_2_tasks
154	Finished	user1	Low	job_2_tasks
155	Finished	user1	Low	job_2_tasks
156	Finished	user1	Low	job_2_tasks
157	Finished	user1	Low	job_2_tasks
158	Finished	user1	Low	job_2_tasks
159	Finished	user1	Low	job_2_tasks

**STARTED**

**Console**

Job 55 has 2 tasks

Id	State	Name	Host name	Start time	Finished time	Re-rur	Description
55000	Running	task1	eon8.inria.fr	16:09:28 08/27/08	Not yet	0/3	task WaitAndPrint - will sleep for 3s
55000	Running	task2	eon8.inria.fr	16:09:28 08/27/08	Not yet	0/1	task WaitAndPrint - will sleep for 20s

**Job Info**

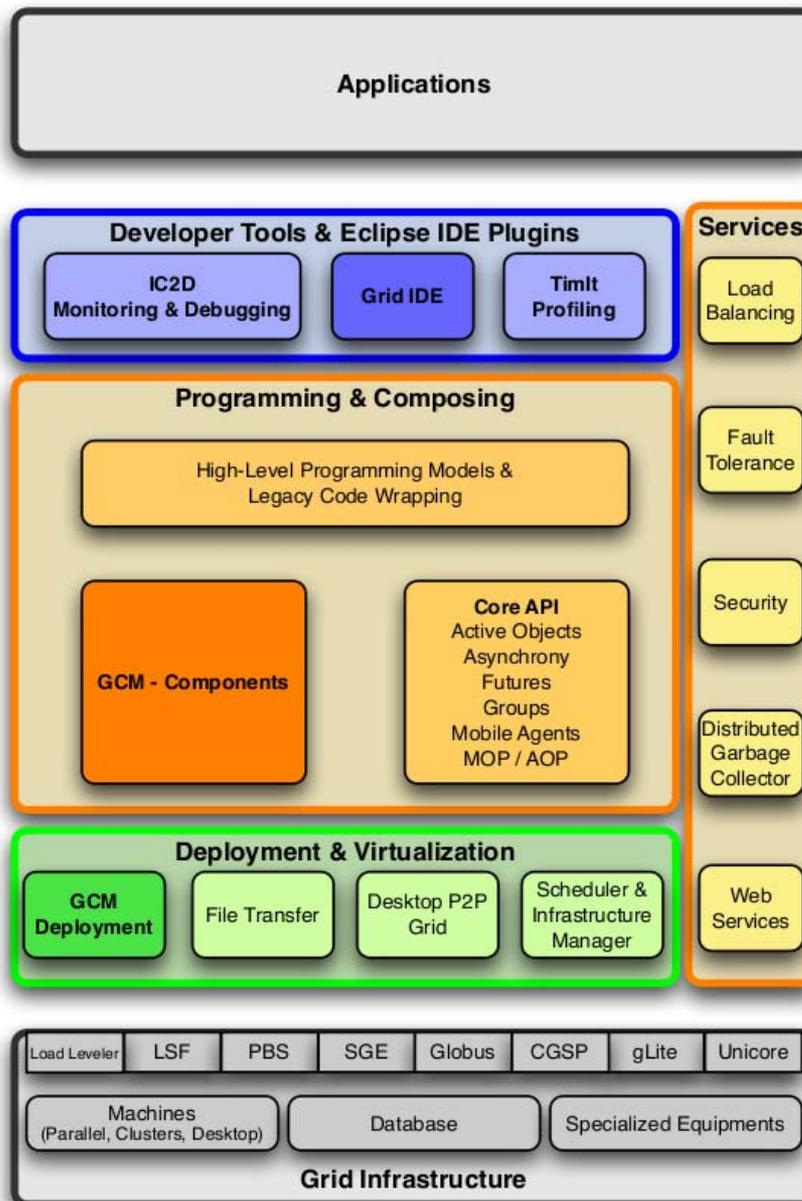
Property	Value
Id	55
State	Running
Name	job_2_tasks
Priority	Low
Pending tasks number	0
Running tasks number	2
Finished tasks number	0
Total tasks number	2
Submitted time	16:09:28 08/27/08



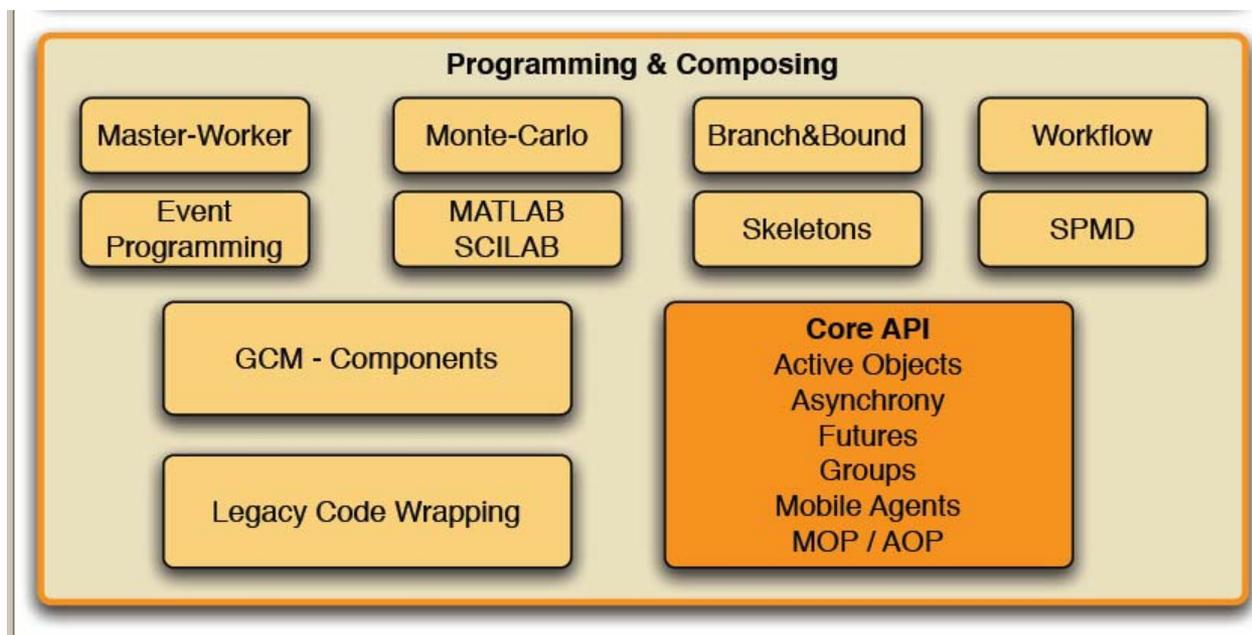
# Video 2: Scheduler, Resource Manager



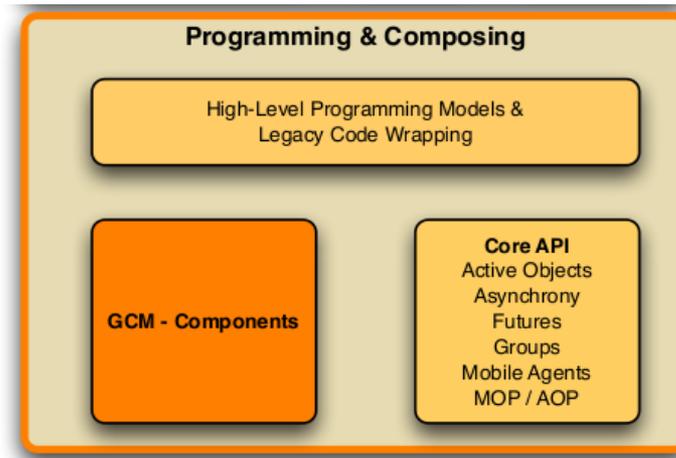
# ProActive Parallel Suite: Program



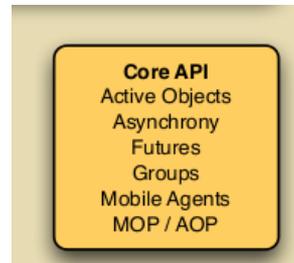
# ProActive Parallel Suite



# ProActive Parallel Suite: Program



# ProActive Parallel Suite: Program



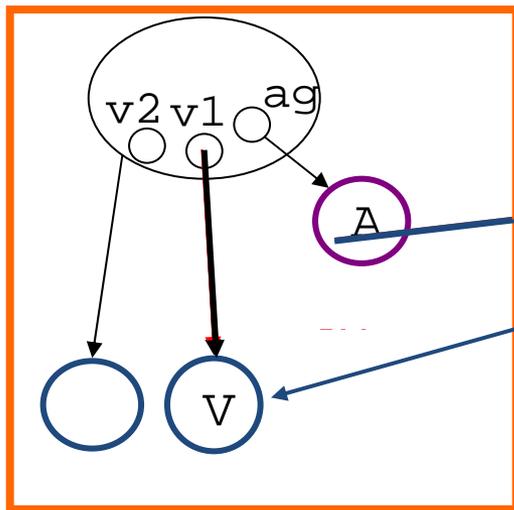
# Distributed and Parallel Active Objects



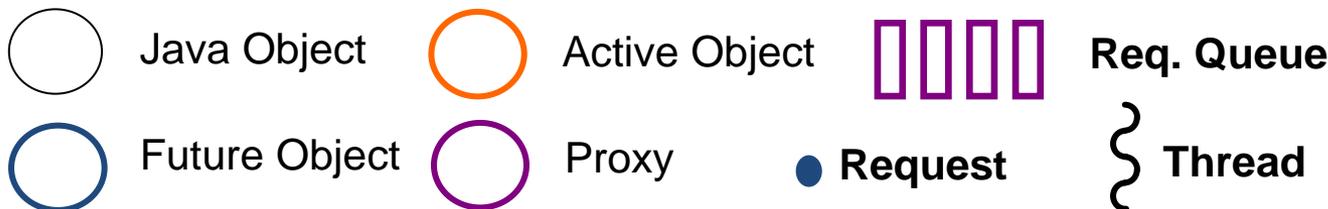
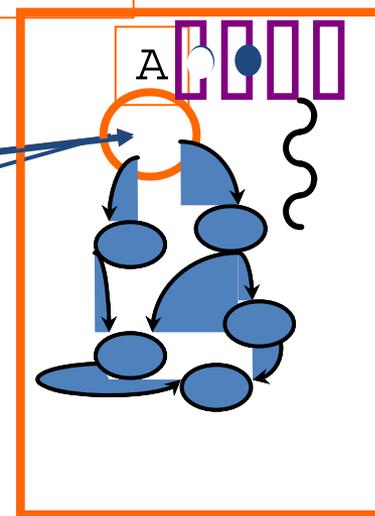
# ProActive : Active objects

- A ag = `newActive` ("A", [...], VirtualNode)
- V v1 = ag.foo (param);
- V v2 = ag.bar (param);
- ...
- v1.bar(); //Wait-By-Necessity

JVM

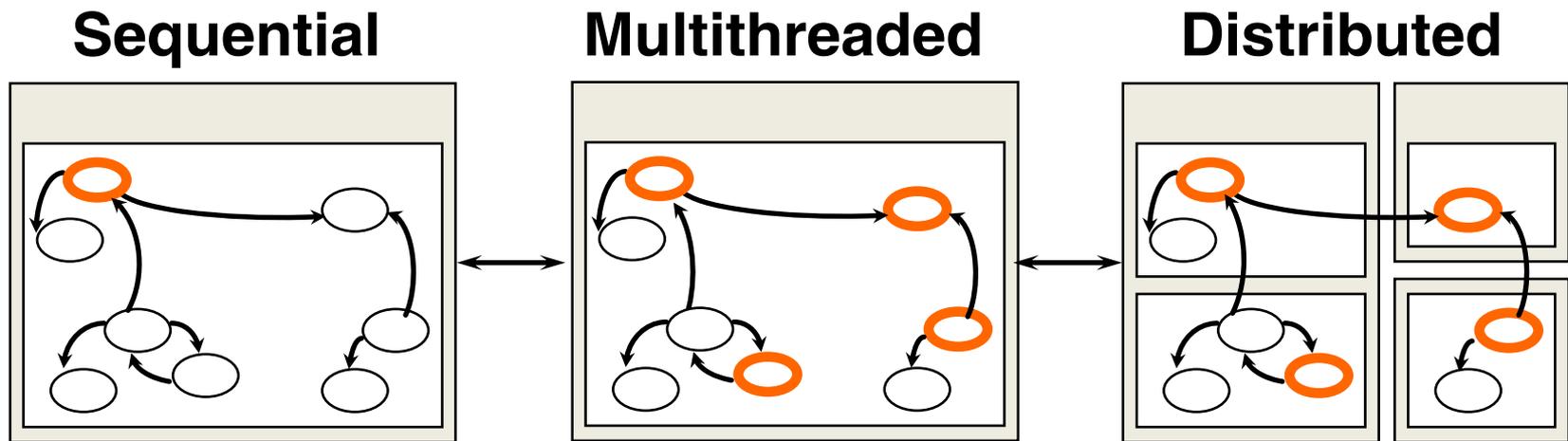


JVM



**Wait-By-Necessity**  
is a  
**Dataflow**  
**Synchronization**

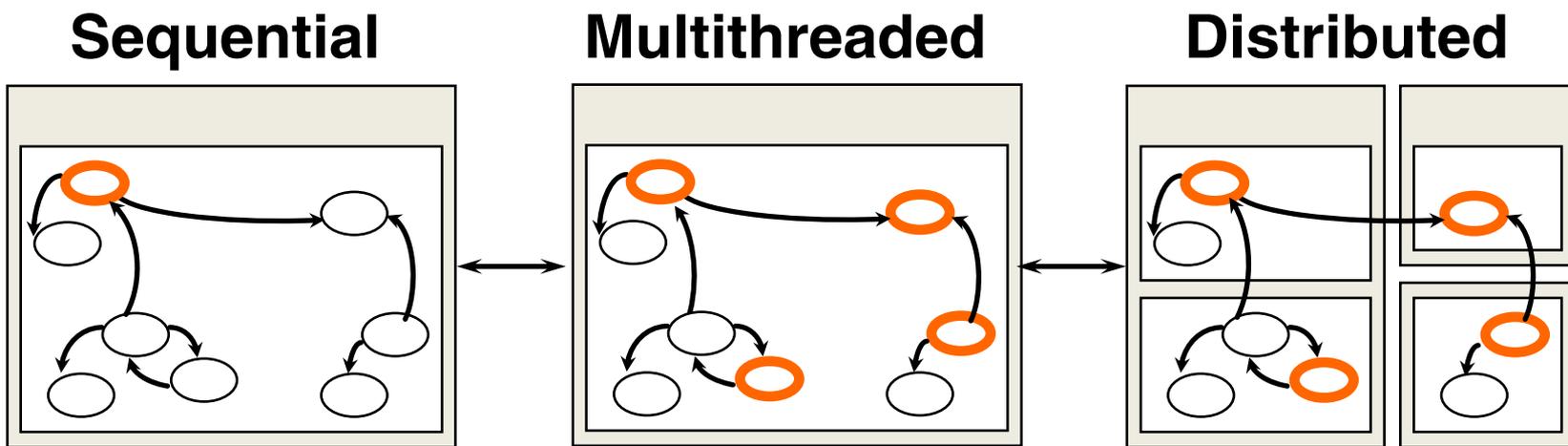
# ProActive: Inter- to Intra- Synchronization



Synchronizations, Behavior: not dependent upon  
the physical location (mapping of activities)



# ProActive: First-Class Futures

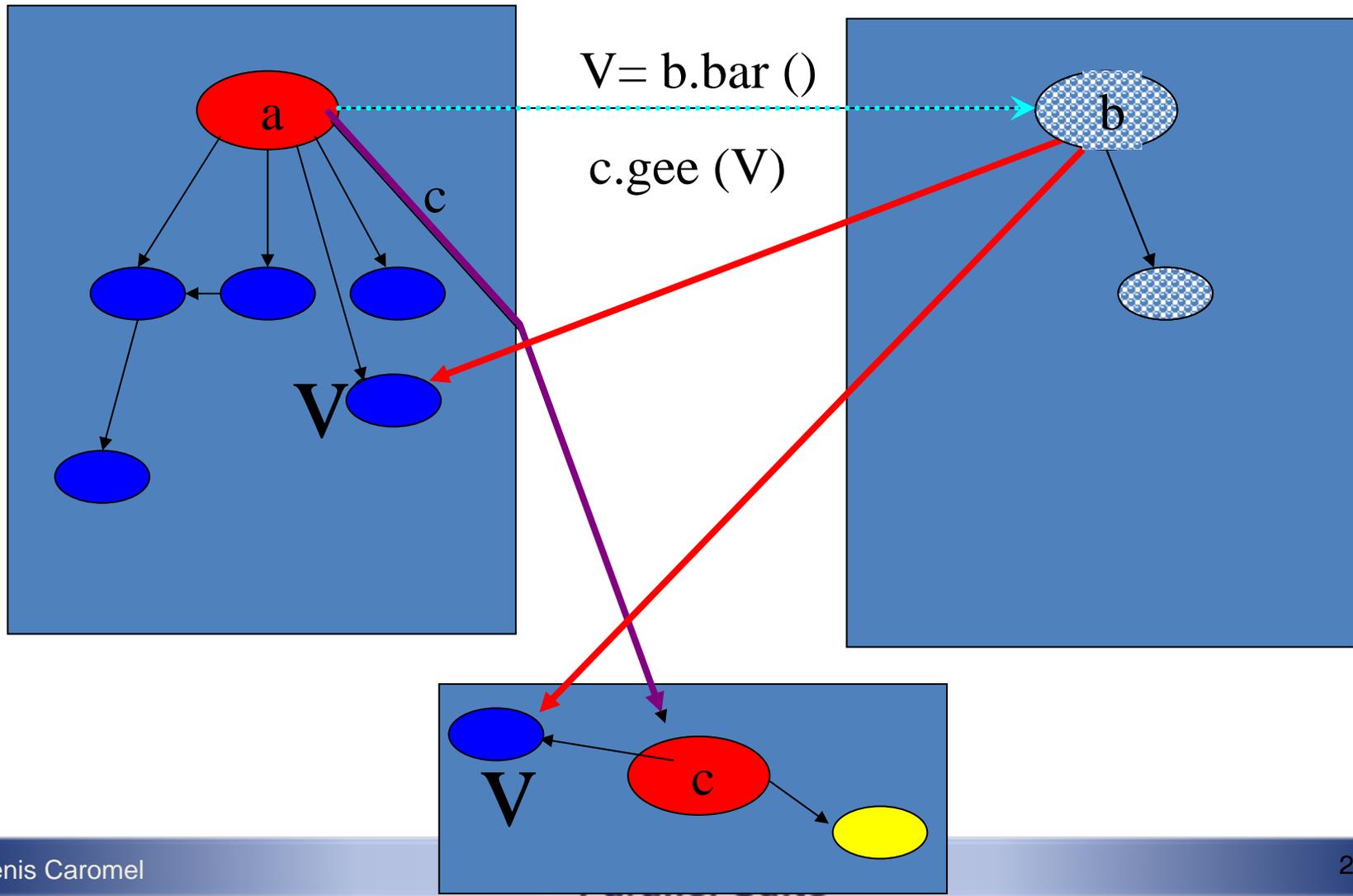


Synchronizations, Behavior: not dependent upon  
the physical location (mapping of activities)



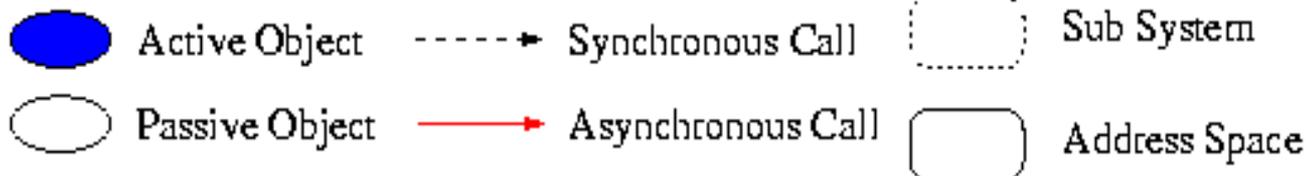
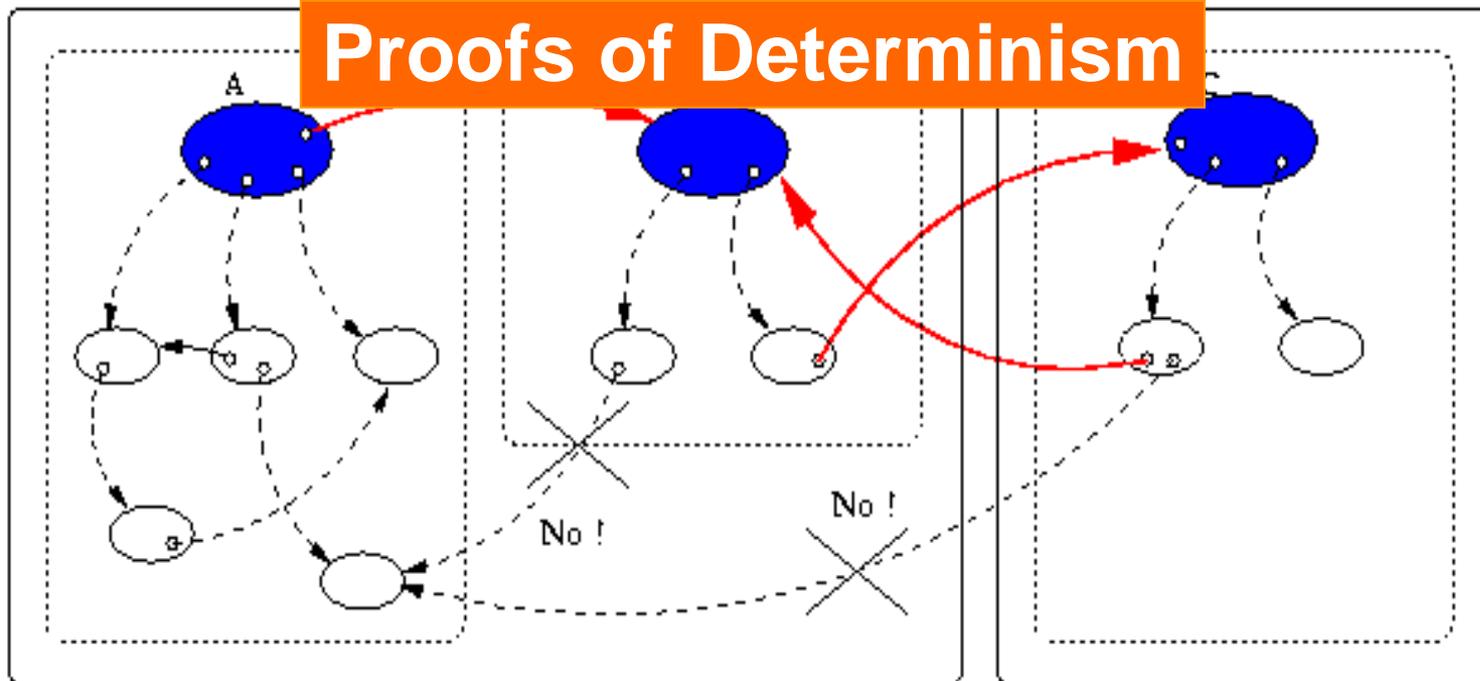
# Wait-By-Necessity: First Class Futures

Futures are Global Single-Assignment Variables



# Standard system at Runtime: No Sharing NoC: Network On Chip

## Proofs of Determinism



# Calculus

## ASP: Asynchronous Sequential Processes



# Proofs in GREEK

$$\frac{(a, \sigma) \rightarrow_S (a', \sigma')}{\alpha[a; \sigma; \iota; F; R; f] \parallel P \longrightarrow \alpha[a'; \sigma'; \iota; F; R; f] \parallel P} \text{ (LOCAL)}$$

$$\frac{\begin{array}{l} \gamma \text{ fresh activity} \quad \iota' \notin \text{dom}(\sigma) \quad \sigma' = \{\iota' \mapsto AO(\gamma)\} :: \sigma \\ \sigma_\gamma = \text{copy}(\iota'', \sigma) \quad \text{Service} = (\text{if } m_j = \emptyset \text{ then } \text{FifoService} \text{ else } \iota''.m_j()) \end{array}}{\alpha[\mathcal{R}[\text{Active}(\iota'', m_j)]; \sigma; \iota; F; R; f] \parallel P \longrightarrow \alpha[\mathcal{R}[\iota']; \sigma'; \iota; F; R; f] \parallel \gamma[\text{Service}; \sigma_\gamma; \iota''; \emptyset; \emptyset; \emptyset] \parallel P} \text{ (N)}$$

$$\frac{\begin{array}{l} \sigma_\alpha(\iota) = AO(\beta) \quad \iota'' \notin \text{dom}(\sigma_\beta) \quad f_i^{\alpha \rightarrow \beta} \text{ new future} \quad \iota_f \notin \text{dom}(\sigma_\alpha) \\ \sigma'_\beta = \text{Copy\&Merge}(\sigma_\alpha, \iota'; \sigma_\beta, \iota'') \quad \sigma'_\alpha = \{\iota_f \mapsto \text{fut}(f_i^{\alpha \rightarrow \beta})\} :: \sigma_\alpha \end{array}}{\alpha[\mathcal{R}[\iota.m_j(\iota')]; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \longrightarrow \alpha[\mathcal{R}[\iota_f]; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma'_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] :: [m_j; \iota''; f_i^{\alpha \rightarrow \beta}]; f_\beta] \parallel P} \text{ (RE)}$$

$$\frac{R = R' :: [m_j; \iota_r; f'] :: R'' \quad m_j \in M \quad \forall m \in M, m \notin R'}{\alpha[\mathcal{R}[\text{Serve}(M)]; \sigma; \iota; F; R; f] \parallel P \longrightarrow \alpha[\iota.m_j(\iota_r) \uparrow f, \mathcal{R}[\Box]; \sigma; \iota; F; R' :: R''; f'] \parallel P}$$

$$\frac{\iota' \notin \text{dom}(\sigma) \quad F' = F :: \{f \mapsto \iota'\} \quad \sigma' = \text{Copy\&Merge}(\sigma, \iota; \sigma, \iota')}{\alpha[\iota \uparrow (f', a); \sigma; \iota; F; R; f] \parallel P \longrightarrow \alpha[a; \sigma'; \iota; F'; R; f'] \parallel P} \text{ (ENDS)}$$

$$\frac{\sigma_\alpha(\iota) = \text{fut}(f_i^{\gamma \rightarrow \beta}) \quad F_\beta(f_i^{\gamma \rightarrow \beta}) = \iota_f \quad \sigma'_\alpha = \text{Copy\&Merge}(\sigma_\beta, \iota_f; \sigma_\alpha, \iota)}{\alpha[a_\alpha; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \longrightarrow \alpha[a_\alpha; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P} \text{ (C)}$$

ASP  $\Rightarrow$  Confluence and Determinacy

**Future updates can occur at any time, Mobility does not change behavior**



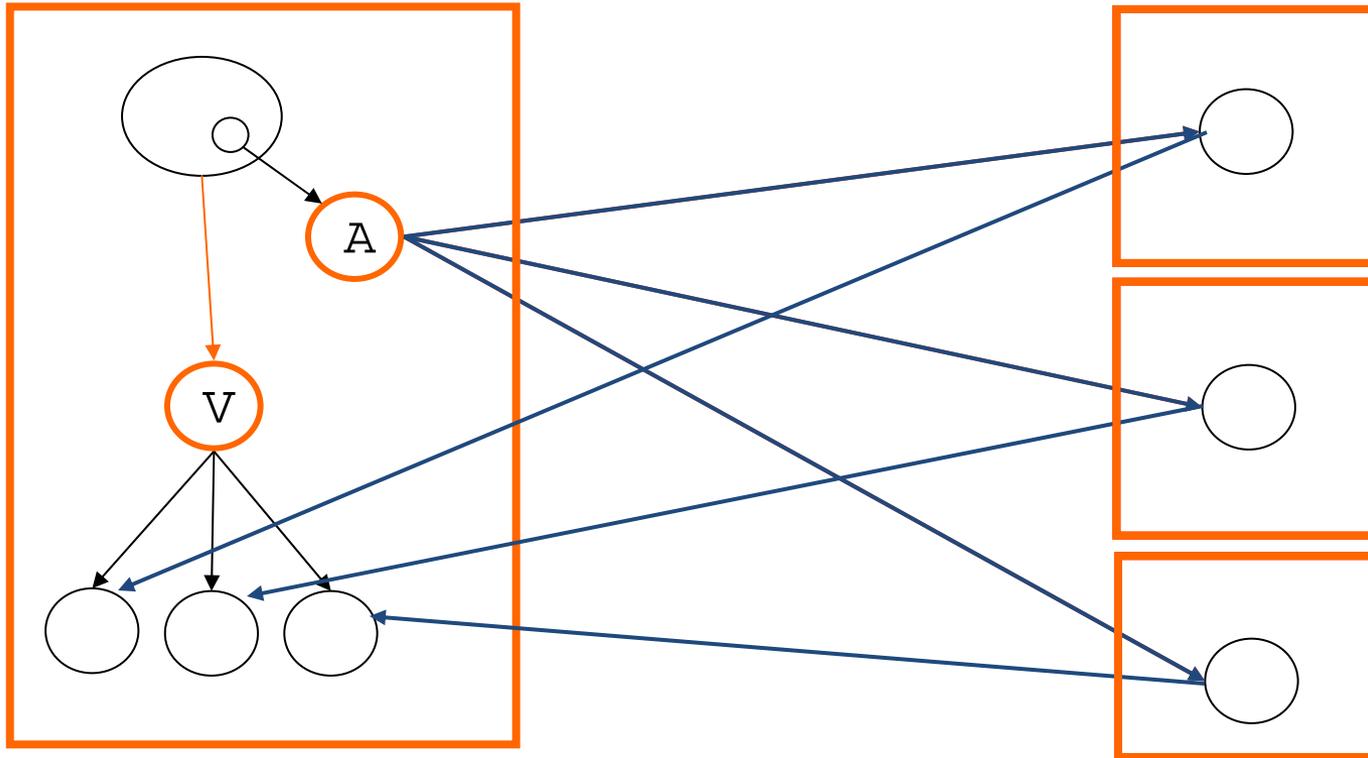
# TYPED ASYNCHRONOUS GROUPS



# Creating AO and Groups

- A ag = `newActiveGroup` ("A", [...], VirtualNode)
- V v = ag.foo(param);
- ● ● ●
- v.bar(); //Wait-by-necessity

JVM



○ Typed Group ○ Java or Active Object

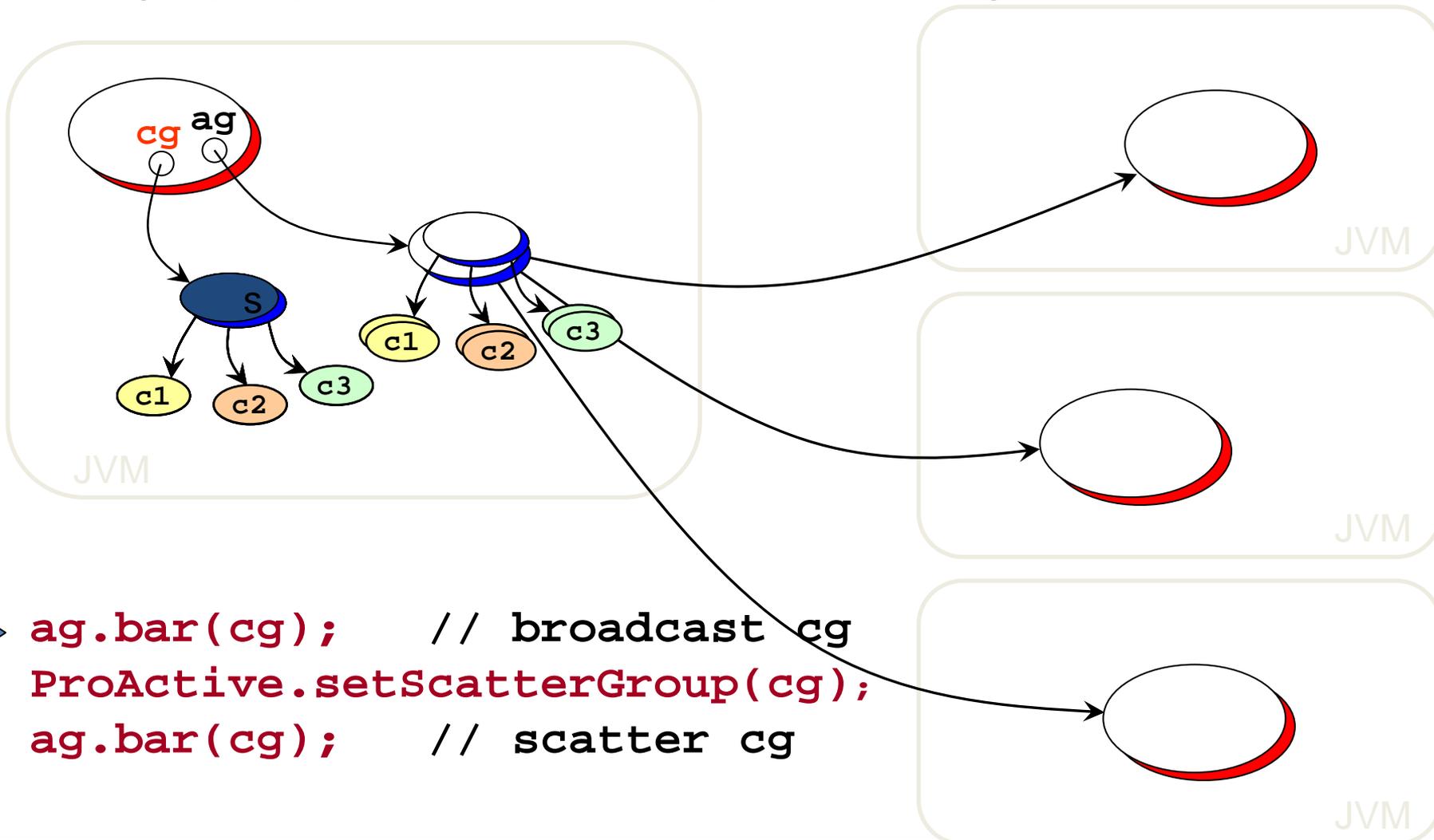
Group, Type, and Asynchrony  
are crucial for Cpt. and GRID



# Broadcast and Scatter

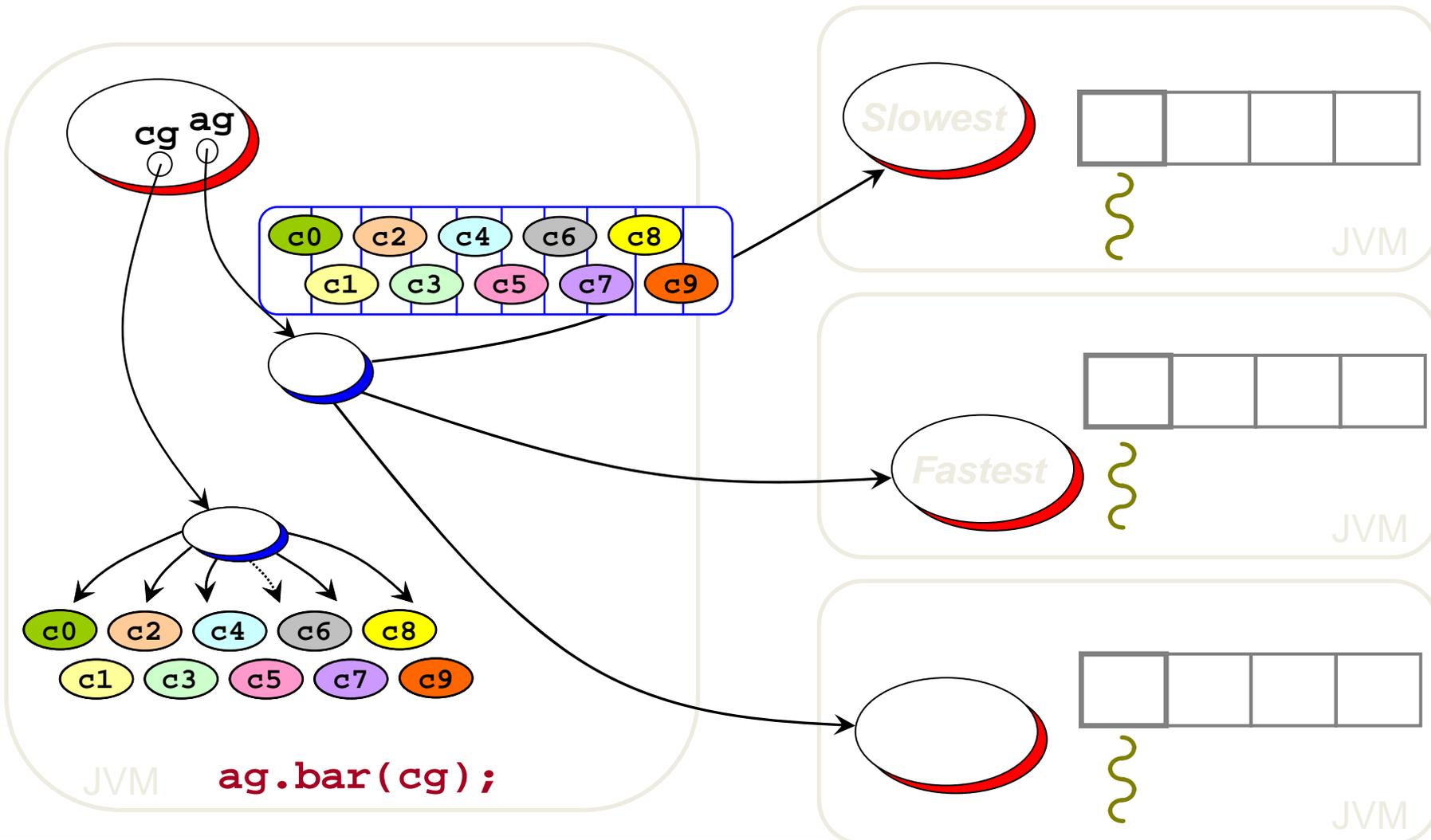
Broadcast is the default behavior

Use a group as parameter, Scattered depends on rankings



➔ `ag.bar(cg); // broadcast cg`  
`ProActive.setScatterGroup(cg);`  
`ag.bar(cg); // scatter cg`

# Dynamic Dispatch Group

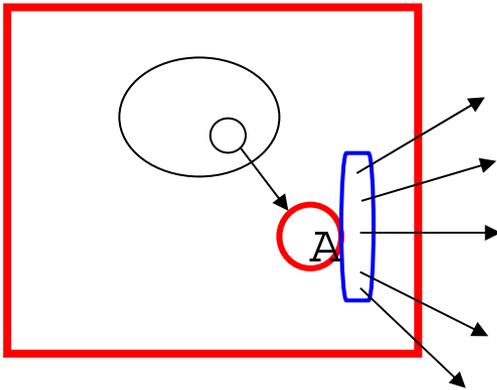


# OO SPMD

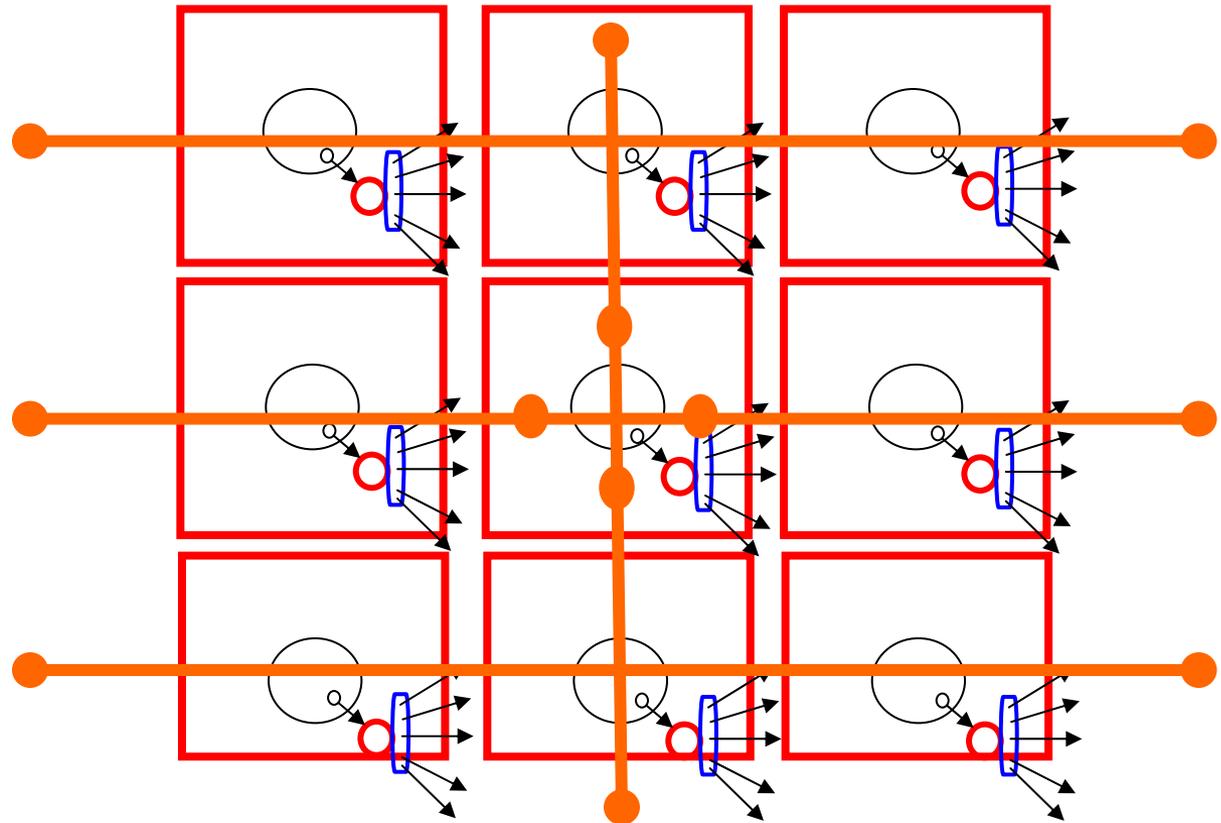
● A ag = `newSPMDGroup` ("A", [...], VirtualNode)

// In each member

- `myGroup.barrier ("2D"); // Global Barrier`
- `myGroup.barrier ("vertical"); // Any Barrier`
- `myGroup.barrier ("north","south","east","west");`



Still,  
not based on raw  
messages, but  
Typed Method Calls  
==> Components



Parallel, Distributed, Hierarchical

# 3. Components and Standardization (GCM)







## GCM: Grid Component Model

GCM Being defined in the NoE CoreGRID

(42 institutions)

Open Source *ObjectWeb ProActive*

implements a preliminary version of GCM

Service Oriented: NESSI relation

**The vision: GCM to be the IT Service GSM**

## GridCOMP takes:

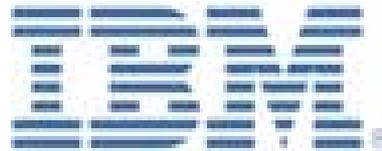
GCM as a first specification,

*ProActive* as a starting point, and

Open Source reference implementation.



# GridCOMP Partners





Jem3D

Steering and Visualisation GUI

invocation parameters

Scalability

Grid

No

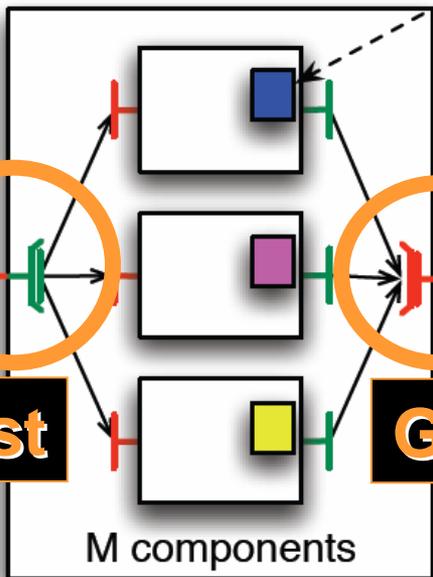
Innovative

**MultiCast**

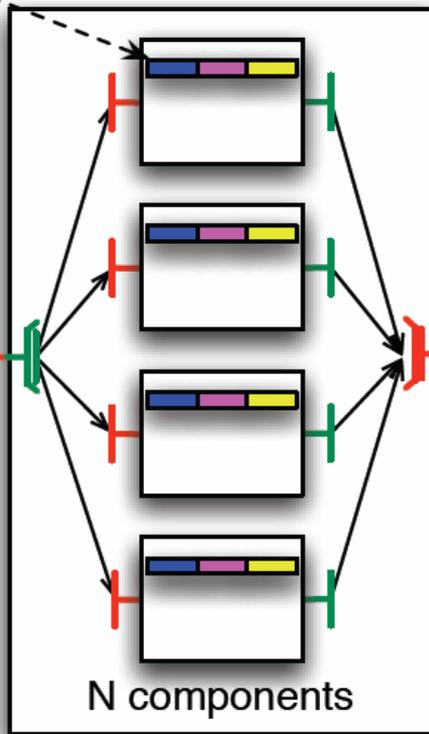
**GatherCast**

Complex

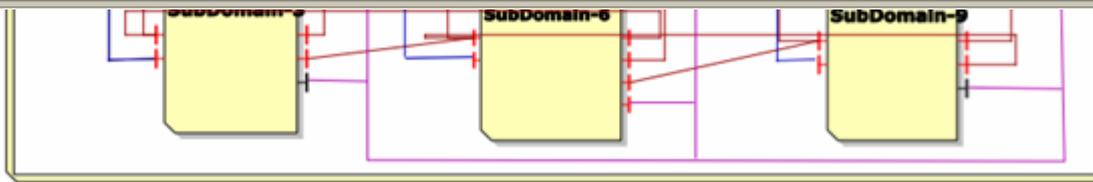
Multiscale



M components



N components



# Standardization



# GCM ETSI STANDARDIZATION

ETSI TC GRID Standardization Group :  
one meeting every 3 or 4 months since Oct. 2006

On 12 June 2008, at the #8 ETSI TC Grid meeting, the two standards:  
GCM Interoperability Deployment  
GCM Interoperability Application Description  
**have been officially approved!**

Overall, the standardization is supported by industrials:  
**BT, FT-Orange, Nokia-Siemens, Telefonica,  
NEC, Alcatel-Lucent, Huawei ...**



# ETSI GCM TC Grid Standard

## Official Standard No 1

GCM Interoperability Deployment

## Official Standard No 2

GCM Application Description

## Work Item No 3

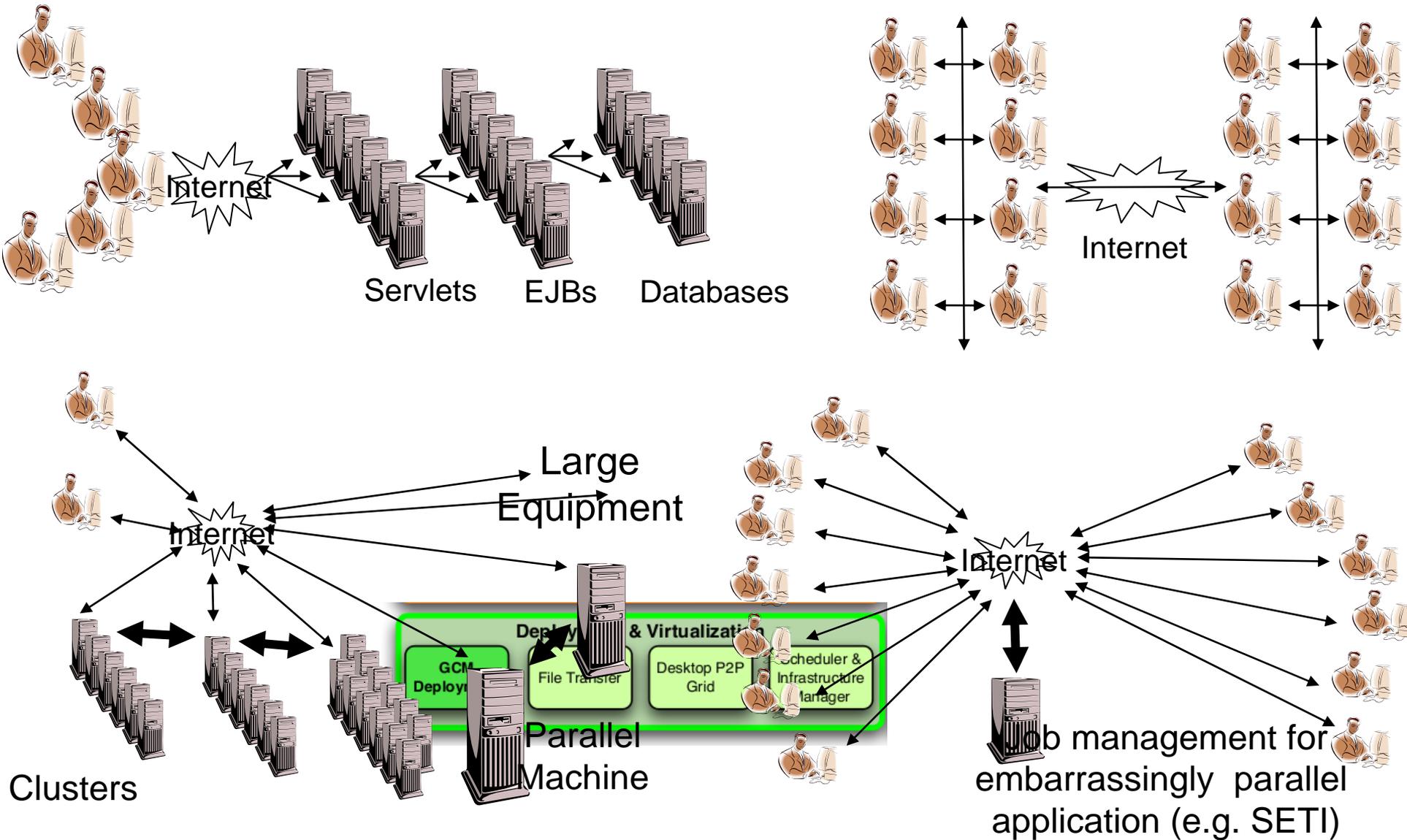
GCM Fractal ADL  
(Architecture Description Language)

## Work Item No 4

GCM Management (Java, C, WSDL API)



# Deploy on Various Kinds of Infrastructures



# Protocols and Scheduler in GCM Deployment

## Protocols:

rsh

ssh

Oarsh

Gsissh

## Scheduler, and Grids:

GroupSSH, GroupRSH, GroupOARSH

ARC (NorduGrid), CGSP China Grid, EGE gLITE,

Fura/InnerGrid (GridSystem Inc.)

GLOBUS

GridBus

IBM Load Leveler, LSF, Microsoft CCS (WHPC 2008)

Sun Grid Engine, OAR, PBS / Torque, PRUN



# GridCOMP / GCM ProActive Usage

Used in Production by Companies:

E.g. Amadeus (Air France, Lufthansa)

At least 4 on going PhD. thesis:

Component reconfiguration (Marcela Rivera),

GCM extensions for autonomic applications (Paul Naoumenko),

Specification Languages and Model-Checking (Antonio Cansado)

Autonomic Service Management of Enterprise Grid Services (Cristian Ruz)

Used in other projects:

EU PROJECTS: SOA4ALL, QosCosGrid, Prospect: RESERVOIR

INRIA ADT Galaxy, Pole Comp. AGOS (HP, Oracle)

Used in:

Barcelona

Krakow (SemMon: Semantic Monitoring)

...



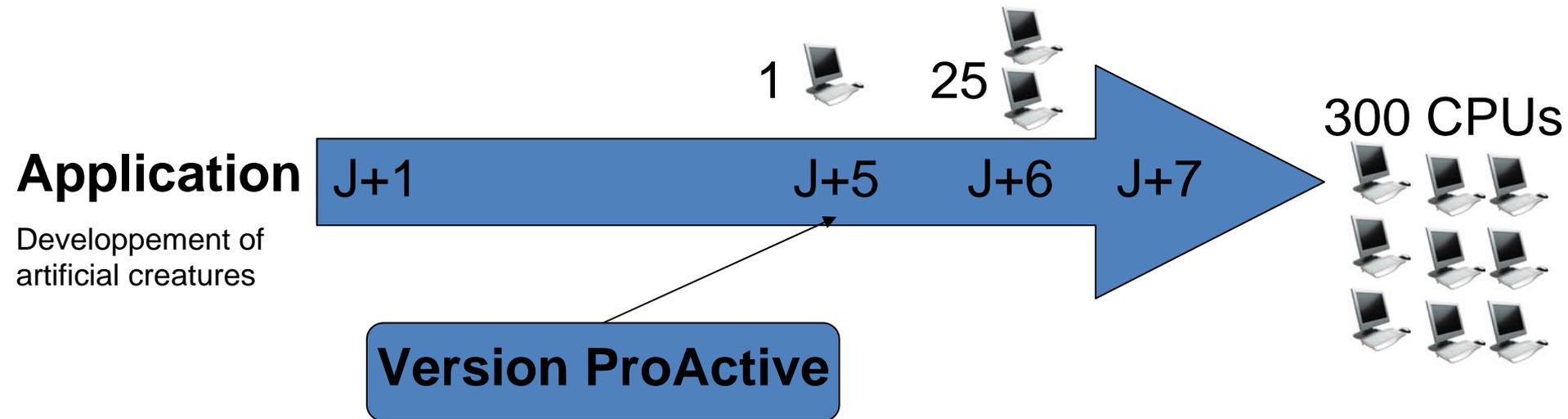
**4.**

# Applications and Perspectives: SOA+GRID



# Artificial Life Generation

Sylvain Cussat-Blanc, Yves Duthen – IRIT TOULOUSE

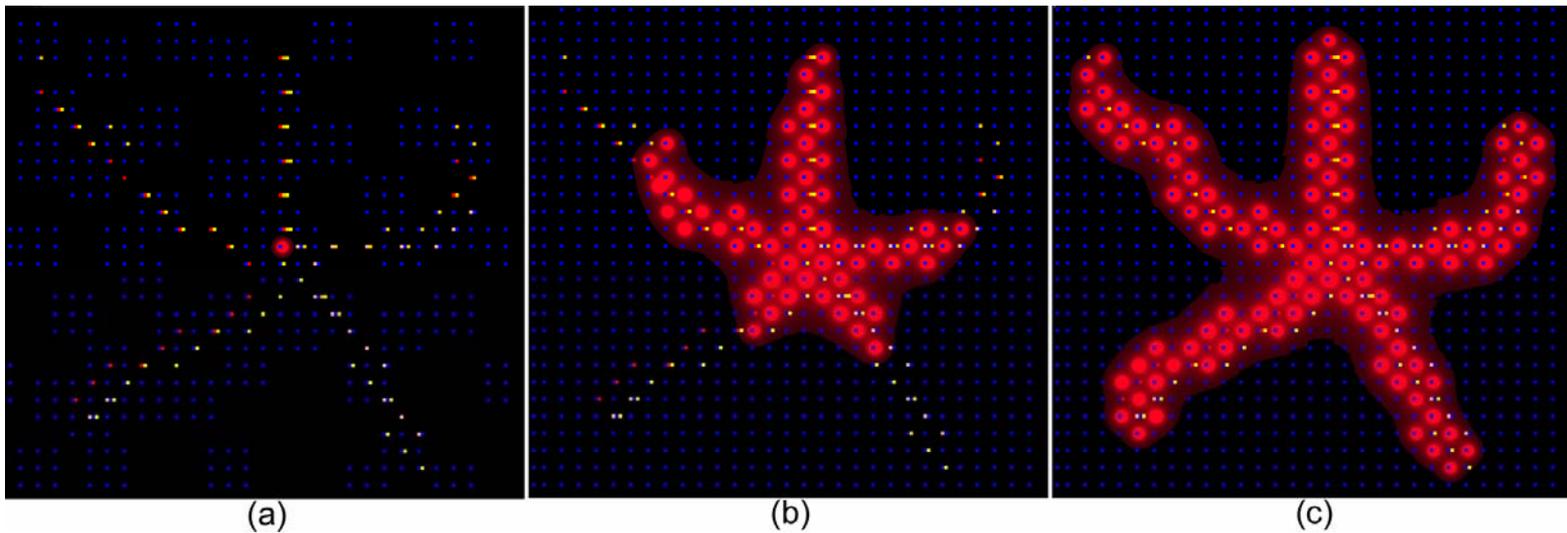


<b>Initial Application</b>	<b>1 PC</b>	<b>56h52 =&gt; Crash!</b>
<b>ProActive Version</b>	<b>300 CPUs</b>	<b>19 minutes</b>



# Artificial Life Generation

Sylvain Cussat-Blanc, Yves Duthen – IRIT TOULOUSE



(a)

(b)

(c)



# JECS : 3D Electromagnetism Radar Reflection on Planes

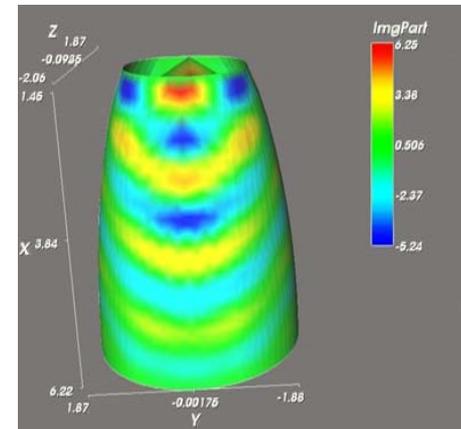
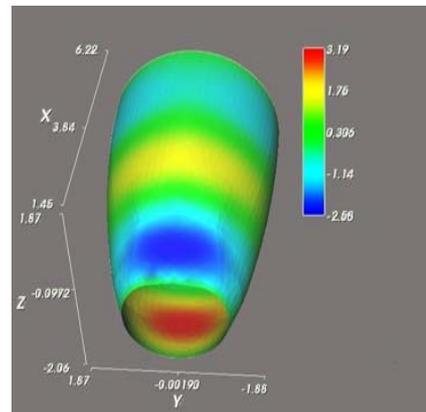
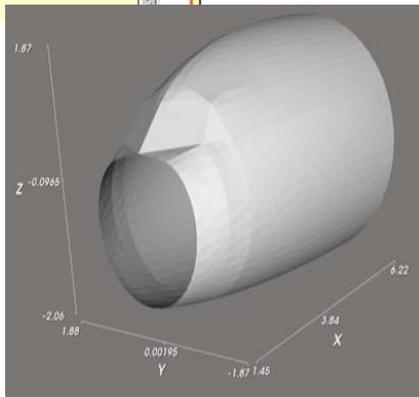
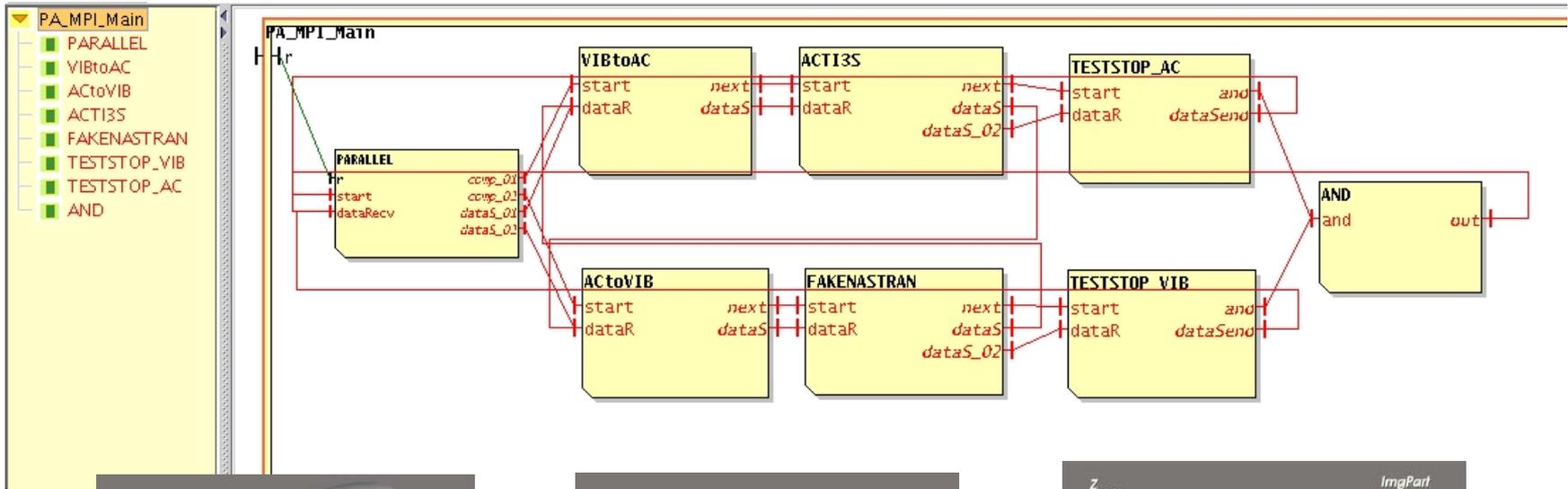
The screenshot displays the JECS software interface. On the left, there are two dialog boxes: 'Display Preferences' and 'Customize scalar bar'. The 'Display Preferences' dialog shows settings for color (scalars), display style (Geometry, Surface), point size (1.0), line width (1.0), decimate factor (0%), visibility (checked), anti-aliasing (unchecked), outline when moving (unchecked), and show fps (unchecked). The 'Customize scalar bar' dialog shows settings for title (H), number of labels (15), visible (checked), height and width sliders, X and Y position sliders, and orientation (Vertical selected).

In the center, there is a 'JECS Tutorial' window with a tree view on the left and a text area on the right. The text area contains instructions on how to install and run JECS. Below the tutorial is the 'Java Environment for Computational Steering' window, which has a menu bar (File, Visualize, Edit, Deployment, Steering, Collaboration, Look & Feel, Help) and a toolbar. The main window shows a 3D view of a jet airplane with a color-coded radar reflection field. A color scale on the right indicates the intensity of the reflection, ranging from 0.326 (red) to 1.44 (blue).

The taskbar at the bottom shows several open applications: Mozilla, Netscape [3], and Caiman.jecs.JECS [2]. The system clock shows the date 27/08/2004.

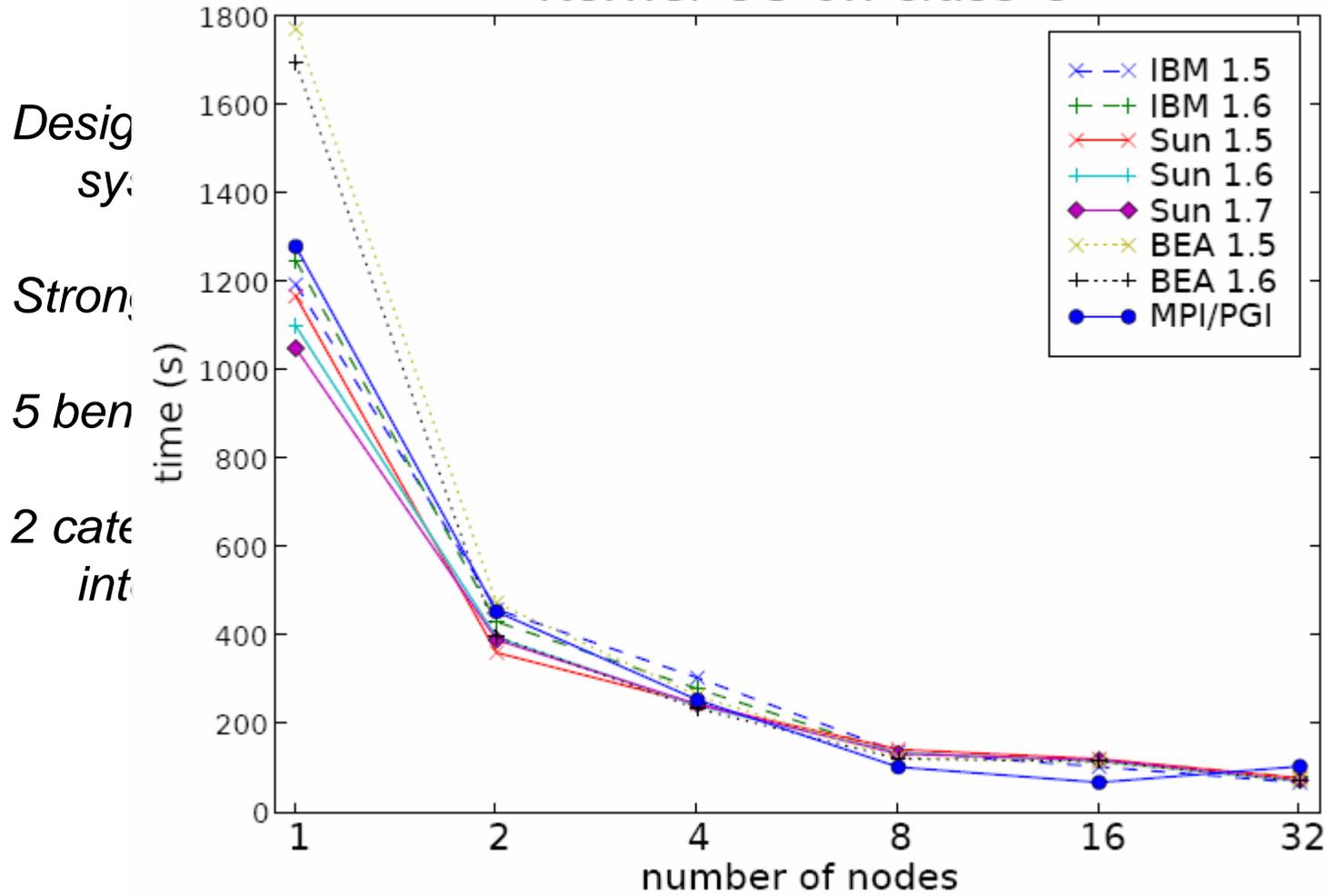


# Code Coupling : Vibro Acoustic (courtesy of EADS)



# NAS Parallel Benchmarks

## Kernel CG on class C



# Enterprise IT: Software Tests

**aMADEUS**

Your technology partner

Amadeus (Opodo, Air France, KLM, Lufthansa):  
500 programmers → 20 machines with

ProActive to execute  
Dist. Regression Tests  
in the production chain



# Parallel BLAST with ProActive (1)

## together with Mario Leyton

**B**asic **L**ocal **A**lignment **S**earch **T**ool for rapid sequence comparison  
BLAST developed by NCBI (**N**ational **C**enter for **B**iotechnology **I**nformation)

Standard native code package, no source modification!

With PPS Skeletons parallelization and distribution *added to the application*

A seamless deployment on all Grid platforms is obtained:

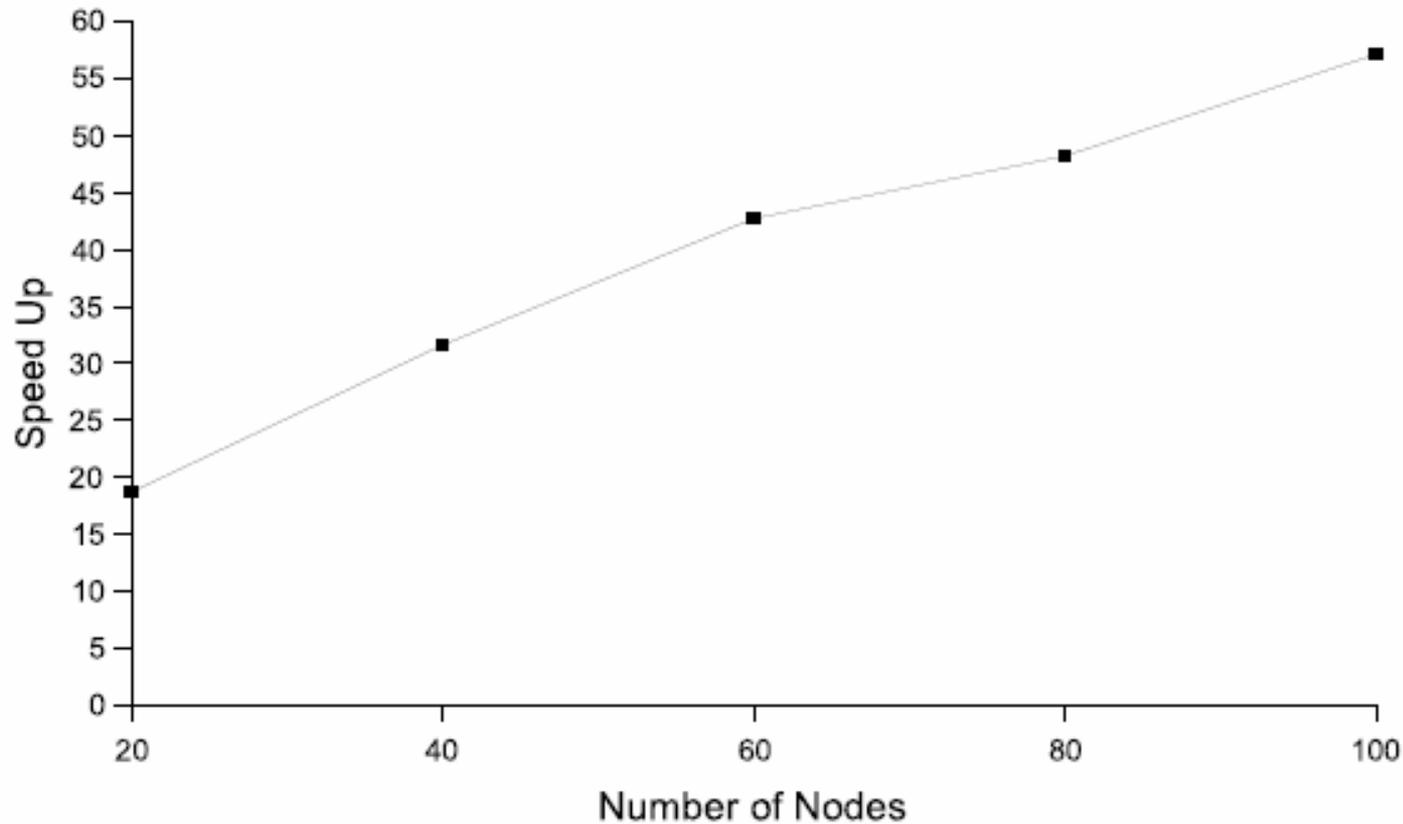
- Input Files are automatically copied to computational nodes at Job submission
- Result Files will be copied on client host

BLAST Skeleton program using the Divide and Conquer skeleton:

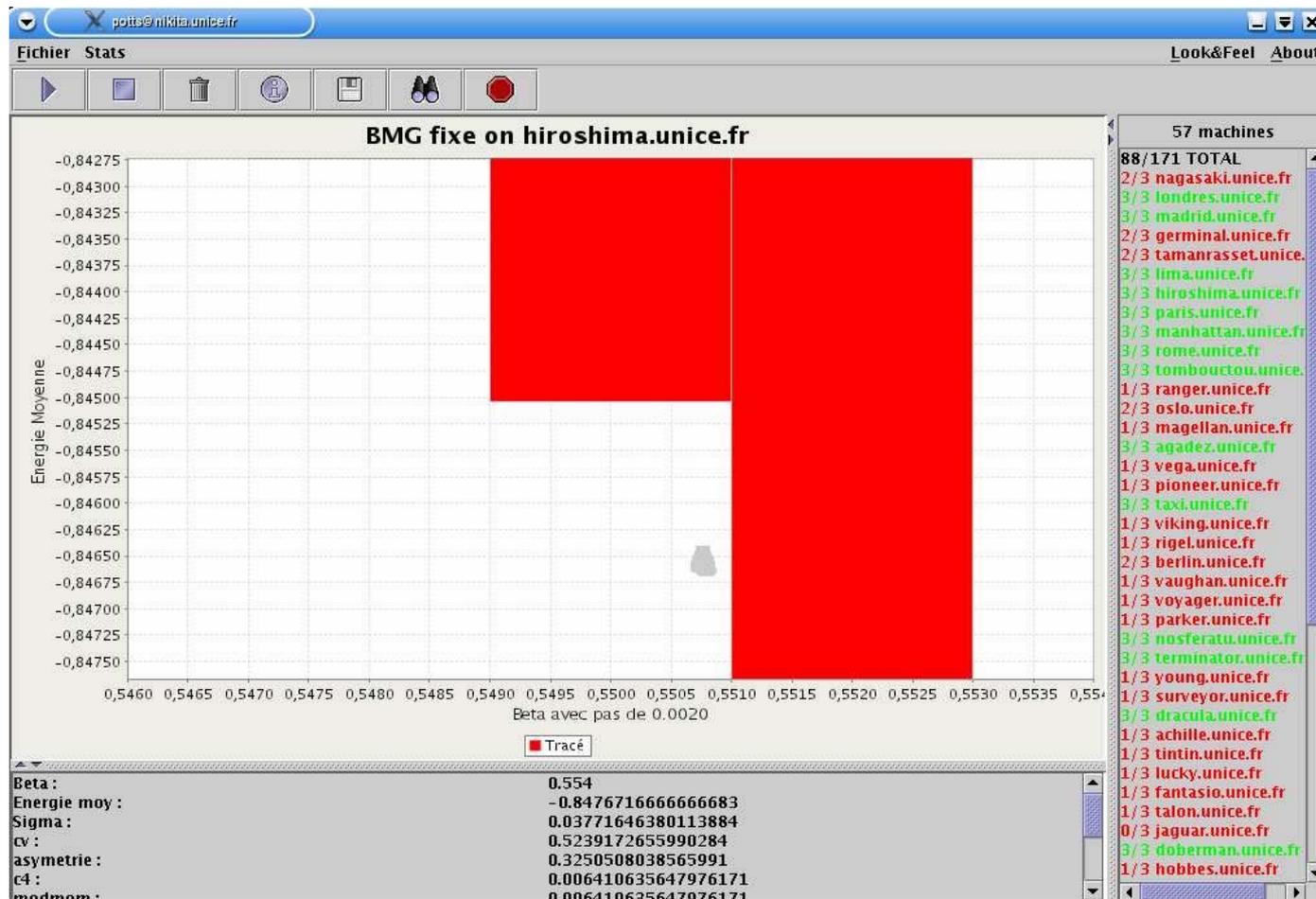
- Division of Database based on conditions (Nb. Nodes, Size, etc.)



# Speedup of Distributed BLAST on Grid5000



# Monte Carlo Simulations, Non-Linear Physics, INLN



# Matlab and Scilab Grid Interface

The screenshot displays the 'Grid Scilab ToolBox' interface, which is used for managing tasks on a grid. The interface is divided into several sections:

- Command:** A tree view on the left shows 'Scilab Engines' with sub-items for Engine0 through Engine5.
- Pending Tasks:** A table listing tasks that are waiting to be executed. The columns are Id Task, Script, Priority, Awaited Time(ms), and State. Tasks Task22, Task23, and Task24 are shown.
- Executing Tasks:** A table listing tasks currently being processed. The columns are Id Task, Script, Id Engine, Global Time(ms), and State. Tasks Task14 through Task18 are shown.
- Terminated Tasks:** A table listing tasks that have completed or failed. The columns are Id Task, Script, Execution Time(ms), Global Time(ms), and State. Tasks Task6 through Task12 are shown.
- Operations:** A log window at the bottom showing the sequence of actions performed, such as deployment, adding tasks, and executing them.
- ToolBox Legend:** A separate window on the right that defines the state icons: a green circle with a dot for 'Pending', a red square with an 'X' for 'Cancelled', a green circle with a dot for 'Executing', a red square with an 'X' for 'Killed', a green checkmark for 'Succeeded', and a red 'X' for 'Aborted'.

At the bottom of the interface, the 'ProActive' and 'Scilab' logos are visible.



# Mikros Image: Post Production

## Frames Making of Nissan



Sébastien Crème // Mikros Image // 20



# New Developments:

## Grid & SOA



# AGOS

## Grid Architecture for SOA

### Building a Platform for Agile SOA with Grid

#### Partners and Solutions



#### Use Cases



# AGOS: What for ?

## AGOS Objectives:

- Create an architecture and environment for integration of
  - SOA business management with
  - GRID IT management
- Well fitted for data intensive and computational intensive applications:
  - Enact sub-parts of a BPEL workflow on dynamically allocated resource  
E.g.: Financial Simulations, Insurance, Revenue Management, BIO, HPC
- Full dynamic scheduling of Services on GRIDs in the future
- Integrated Management of SLO, SLA, QoS:
  - Bottom to top
  - Dynamic enforcement: Adaptive behavior



# Summary



*Concurrency + Parallelism*  
*Multi-Cores + Distribution*



# Conclusion: Why does it scale?

Thanks to a few key features:

Connection-less, RMI+JMS unified

Messages rather than long-living interactions

ACTIVE OBJECTS --- GROUPS --- COMPONENTS



# Conclusion: Why does it Compose?

Thanks to a few key features:

Because it **Scales**: asynchrony !

Because it is Typed: RMI with **interfaces** !

First-Class Futures: **No unstructured** Call Backs and Ports

**ACTIVE OBJECTS --- GROUPS --- COMPONENTS**

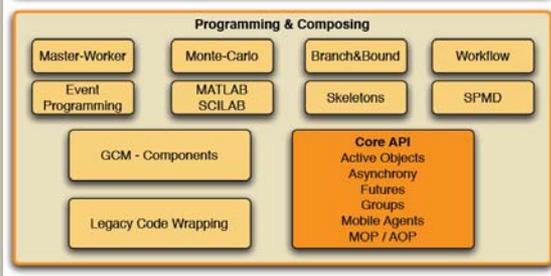


# Conclusion:



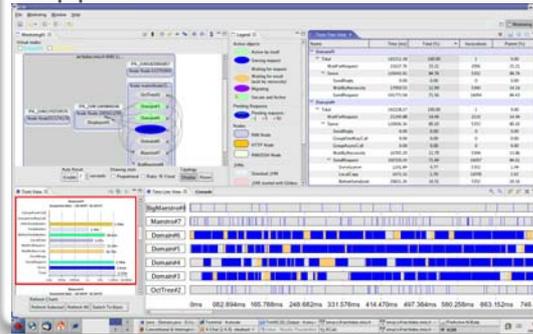
## PROGRAMMING

**Java Parallel Frameworks**  
for HPC, Multi-Cores,  
Distribution, Enterprise  
Grids and Clouds.



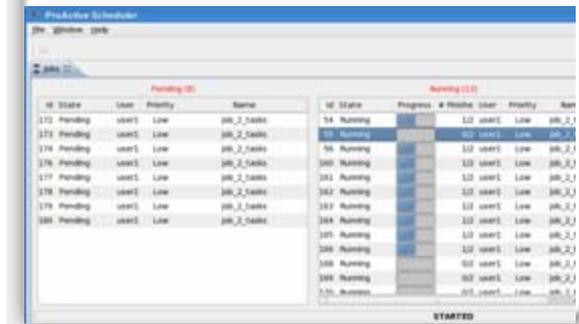
## OPTIMIZING

**Eclipse GUI (IC2D)**  
for Developing, Debugging,  
Optimizing your parallel  
applications.



## SCHEDULING

**Multi-Language Scheduler**  
for Workflows made of C,  
C++, Java, Scripts, Matlab,  
Scilab tasks.



# A Toolkit for Acceleration: Multi-Core & Distributed



**ProActive/  
GCM  
Specifications**

**for**

**Components  
Services  
SLA  
QoS**

**Open the way  
to Soft.+Serv.  
EU Industry  
with  
Clouds &  
Utilities,  
DAAS**

**PLUGTESTS**  
THE INTEROPERABILITY SERVICE

**COME & test**

**GRIDS  
@WORK  
VGRID PLUGTESTS  
2008**

WITH THE SUPPORT OF:

EUROPEAN PARTNERS:

CO-ORGANIZERS

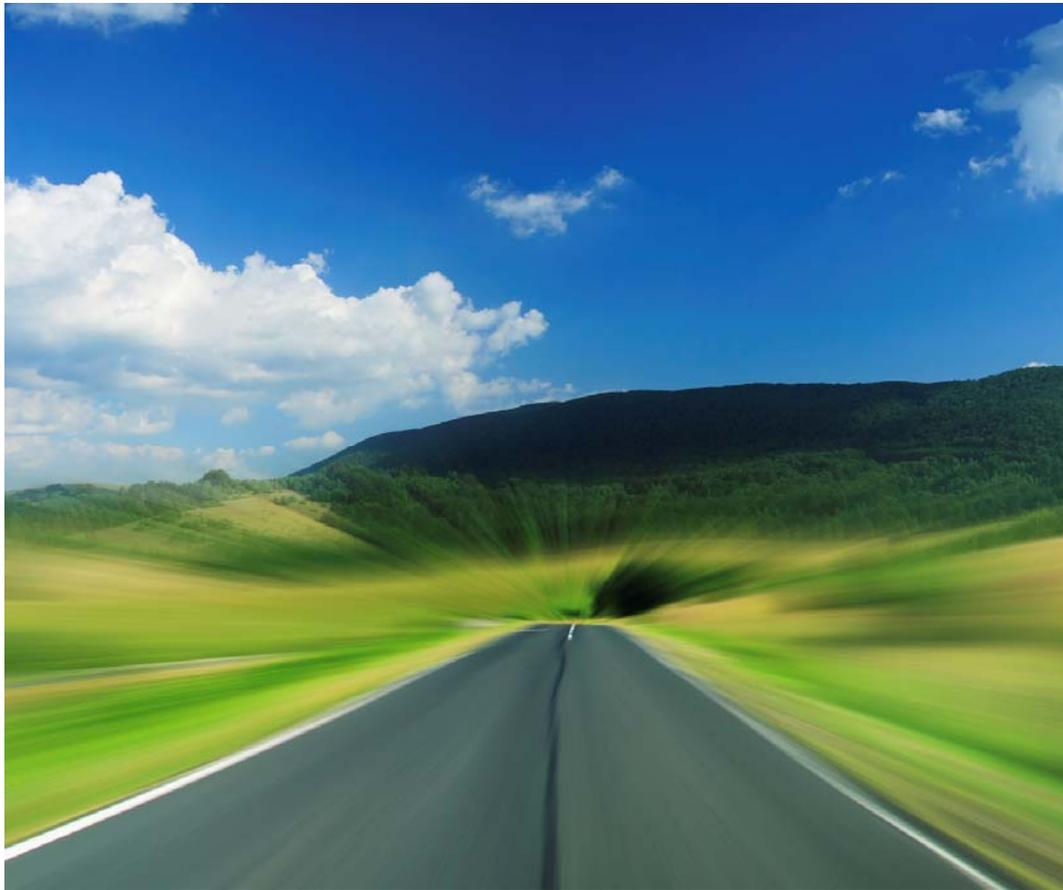
INDUSTRIAL SPONSORS

**FRENCH RIVIERA  
INRIA / SOPHIA ANTIPOLIS  
OCT. 20 - 24**

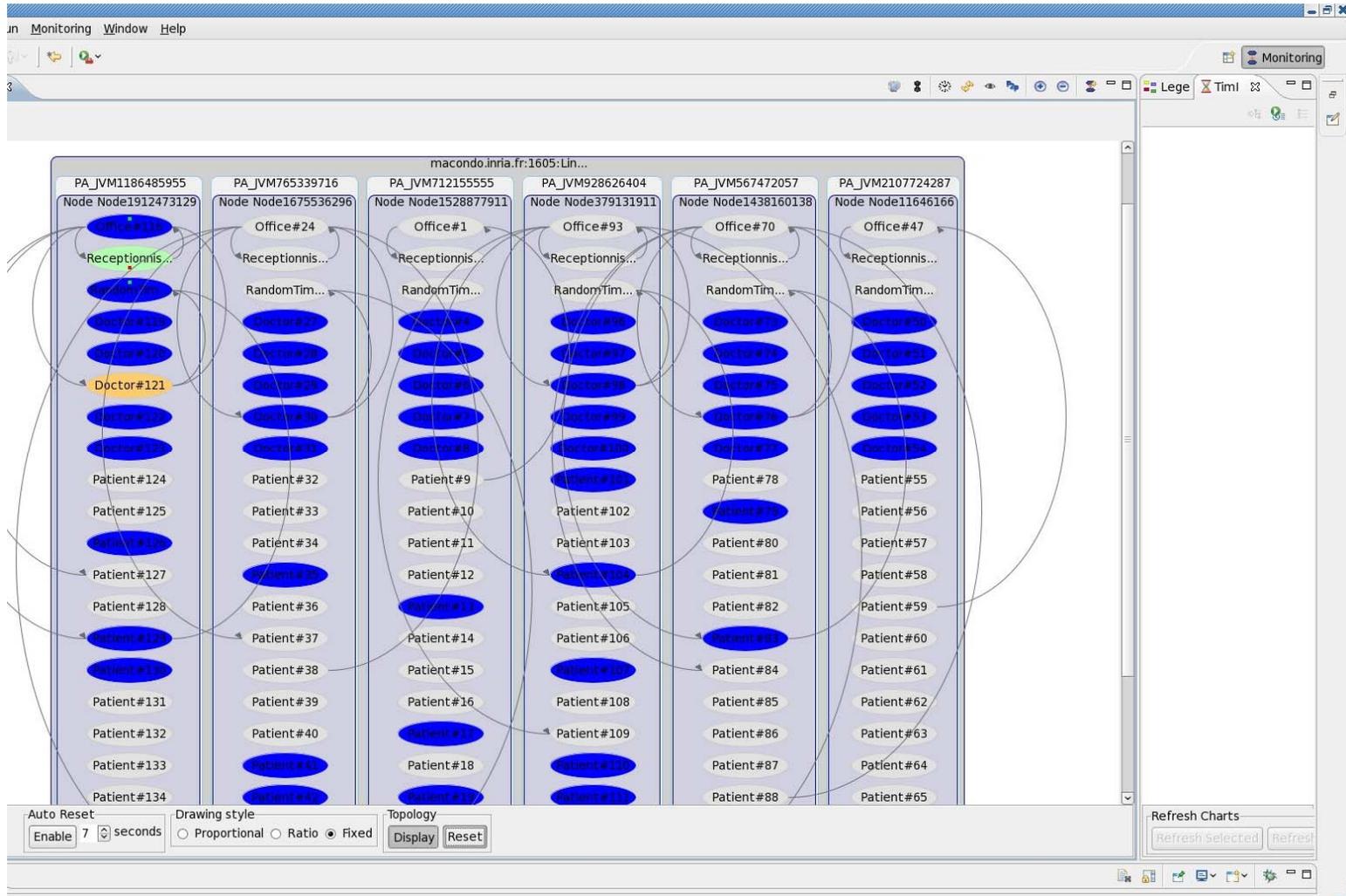
**ETSI**  
World Class Standards

**GRIDs for Finance & Telecommunications**





# Multi-Active Object in 1 Address Space for Multi-cores



## Trainings

### ProActive Overview and Application to Finance

Location: French Riviera  
Schedule:  
- 22, 23rd October 2008

## News

NEW TECHNICAL PAPER:  
**Current State of Java for HPC**

### 5th Grid Plugtests (Grids@Work)

20-24 October 2008

Monte-Carlo Finance Contest and ProActive User Group

Invited Keynote at CGW'08, Kraków, Poland  
October, 13-15, 2008

**Grid Component Model and ProActive Parallel Suite for Science**

## Careers

Engineer positions available, click [here](#) to learn more.

## Latest releases

Latest:  
**ProActive 4.0.1, Sep. 2008**  
[\[Download ProActive\]](#)

Welcome to the Open Source World of ProActive !

**New ! ProActive 4.0.1 (15/09/2008)**

*ProActive Parallel Suite* is an **Open Source** middleware (OW2 consortium) for **parallel, distributed, multi-core** computing.

*ProActive* tools **simplify** the programming and execution of parallel applications on: Multi-core processors, Distributed Local Area Network (LAN), Clusters and Data Center Servers, GRIDs, and on OS such as Linux, Windows, Mac.

*ProActive Parallel Suite* features:

### PROGRAMMING

**Java Parallel Frameworks**  
for HPC, Multi-Cores, Distribution, Enterprise Grids and Clouds.

Featuring: Async. comms, Master-Worker, Monte-Carlo, SPMD, components and legacy code wrapping.

### OPTIMIZING

**Eclipse GUI (IC2D)**  
for Developing, Debugging, Optimizing your parallel applications.

Featuring: graphical monitoring and benchmarking with report generation.

### SCHEDULING

**Multi-Language Scheduler**  
for Workflows made of C, C++, Java, Scripts, Matlab, Scilab tasks.

Featuring: graphical user interface, resource acquisition and virtualization.

Featuring **fault-tolerance, load-balancing, mobility, and security**, ProActive brings ease-of-use in the world of parallel computing. Programmers write standard Java code (POJO) for their parallel and Grid applications.  
***The Grid Application Server for the Enterprise!***

Developer Tools & Eclipse IDE Plugins

5 Minutes

Programming &

Professional

ProActive



## PROGRAMMING

el  
s  
ti-Cores,  
Enterprise  
ouds.

sync. comms,  
er, Monte-  
, components  
ode wrapping.

## OPTIMIZING

**Eclipse GUI (IC2D)**  
for Developing, Debugging,  
Optimizing your parallel  
applications.

Featuring: graphical  
monitoring and  
benchmarking with report  
generation.

## SCHEDULING

**Multi-Language S**  
for Workflows made  
C++, Java, Scripts, M  
Scilab tasks.

Featuring: graphical  
interface, resource a  
and virtualization.



# Object-Oriented SPMD

## Single Program Multiple Data

### Motivation

Cluster / GRID computing

SPMD programming for many numerical simulations

**Use enterprise technology (Java, Eclipse, etc.) for Parallel Computing**

Able to express most of MPI's Collective Communications:

**broadcast**

**reduce**

**scatter**

**allscatter**

**gather**

**allgather**

and Barriers, Topologies.



# GCM Deployment (2/2)

Grid description: clear concepts

Bridges (1 -> 1)

Groups (1 -> N)

Hosts

Acquisition (lookup, p2p)

Application description:

Split Grid / Application Description

Allows reuse of grid descriptors for any application type,

ProActive, using Virtual nodes



# Deployment descriptor : example

```
<resources>
  <bridge refid="bSchubby">
    <host refid="hSchubby" />
  </bridge>
</resources>

<acquisition>
  <lookup type="RMI" port="6666" hostList="host[0-9].grid.fr"></lookup>
  <p2p nodesAsked="50">
    <localClient protocol="RMI" port="2410" />
    <peerSet>
      <peer>rmi://schubby.inria.fr</peer>
      <peer>http://gaudi.inria.fr</peer>
    </peerSet>
  </p2p>
</acquisition>

<infrastructure>
  <hosts>
    <host id="hSchubby" os="unix" hostCapacity="1" vmCapacity="1">
      <homeDirectory base="root" relpath="/user/cmathieu/home" />
    </host>
  </hosts>

  <bridges>
    <sshBridge commandPath="/usr/bin/ssh" hostname="schubby.inria.fr" id="bSchubby" username="cmathieu" />

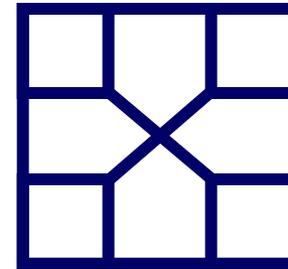
    <rshBridge hostname="schubby.inria.fr" id="brSchubby" username="cmathieu" />

  </bridges>
</infrastructure>
```

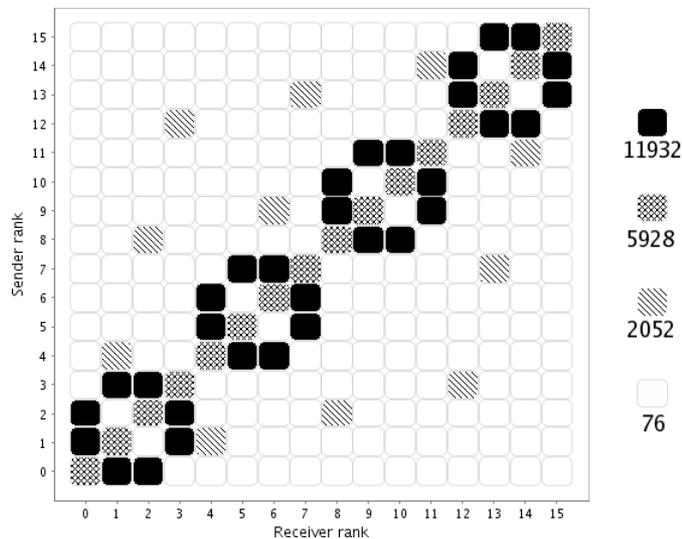


# Communication Intensive CG Kernel (Conjugate Gradient)

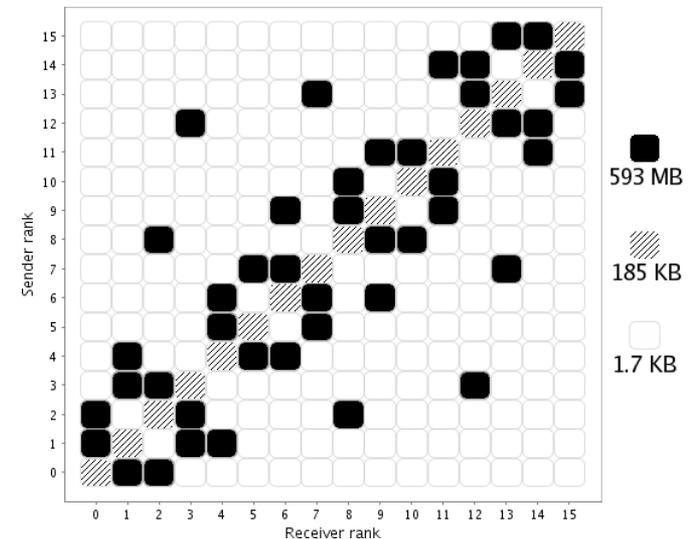
Floating point operations  
Eigen value computation  
High number of unstructured communications



- 12000 calls
- 570 MB sent
- 1 min 32
- 65 % comms



Data density distribution

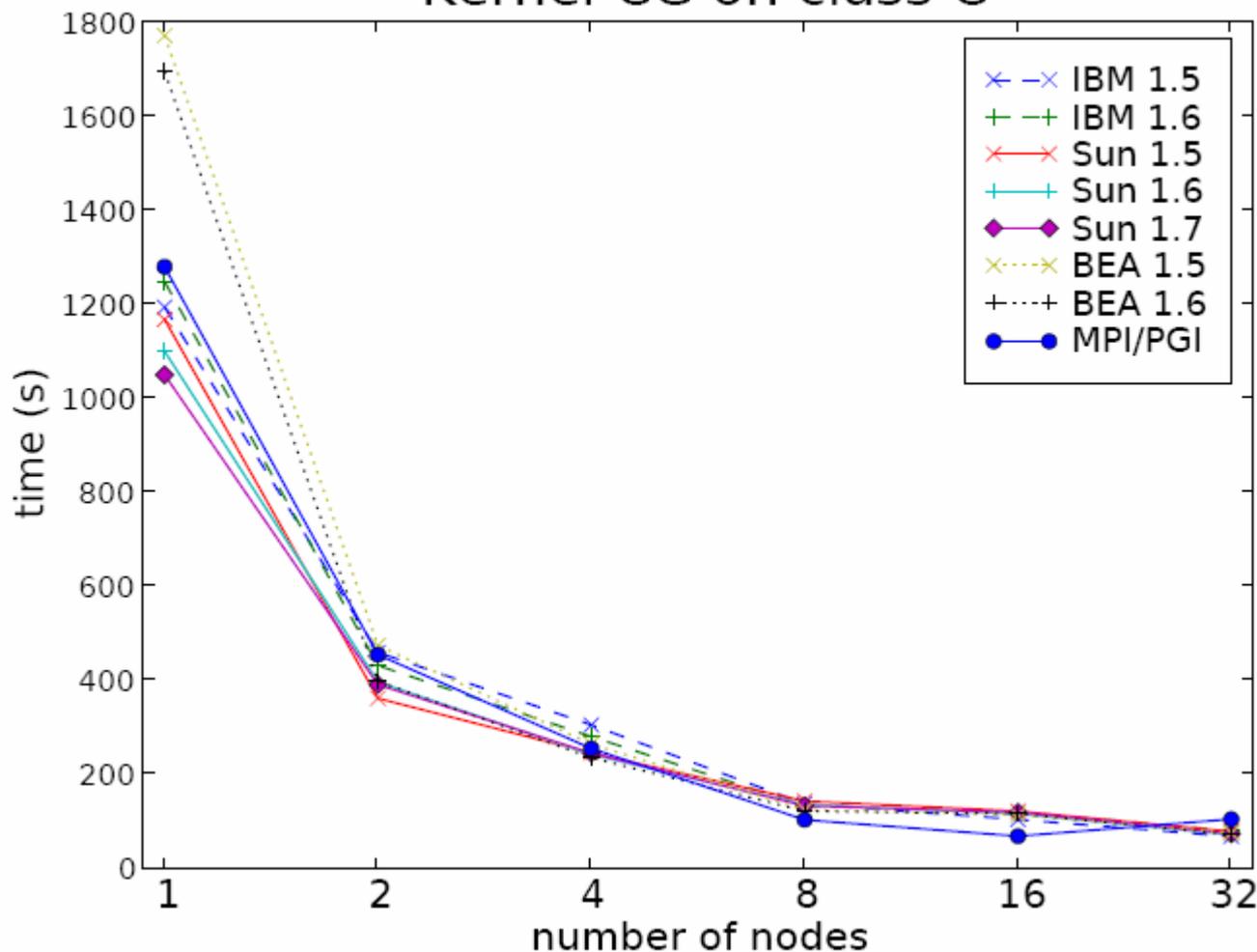


Message density distribution

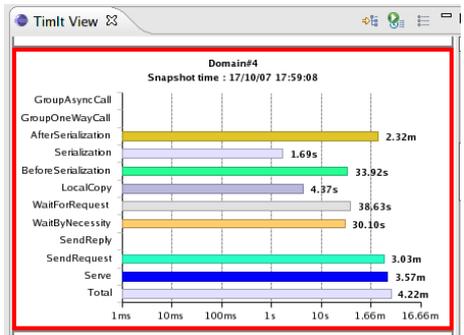
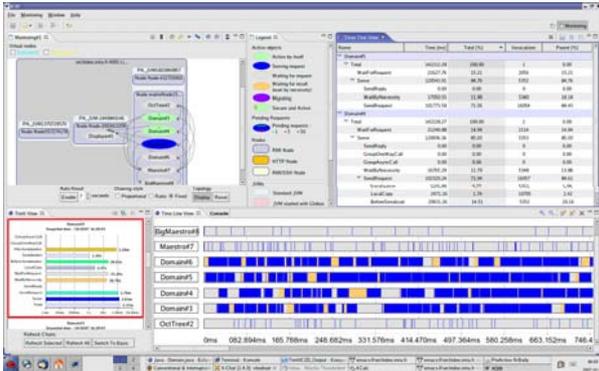


# Communication Intensive CG Kernel (Conjugate Gradient)

Kernel CG on class C

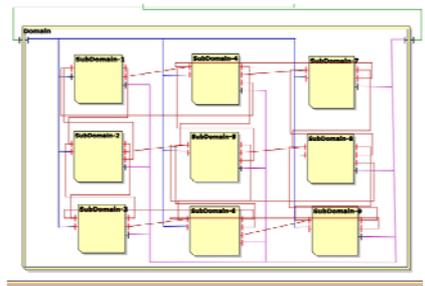


# Summary-Perspective: Comprehensive Toolkit



Programming:  
Models &  
Tools

Parallel:  
Multi-Core &  
Distributed



ID	Name	User	Priority	Status
113	Running	user1	LOW	Running
114	Running	user1	LOW	Running
115	Running	user1	LOW	Running
116	Running	user1	LOW	Running
117	Running	user1	LOW	Running
118	Running	user1	LOW	Running
119	Running	user1	LOW	Running
120	Running	user1	LOW	Running

Applications

Developer Tools & Eclipse IDE Plugins

- IC2D Monitoring & Debugging
- Grid IDE
- Timt Profiling

Services

- Load Balancing

Programming & Composing

- High-Level Programming Models & Legacy Code Wrapping
- GCM - Components
- Core API: Active Objects, Asynchrony, Futures, Groups, Mobile Agents, MOP / AOP

- Fault Tolerance
- Security
- Distributed Garbage Collector

Deployment & Virtualization

- GCM Deployment
- File Transfer
- Desktop P2P Grid
- Scheduler & Infrastructure Manager

- Web Services

Grid Infrastructure

- Load Leveler, LSF, PBS, SGE, Globus, CGSP, gLite, Unicore
- Machines (Parallel, Clusters, Desktop)
- Database
- Specialized Equipments

