

Bridging

Multi-Core and Distributed Computing: all the way up to the Clouds

D. Caromel, et al.

Agenda

1. Background: OASIS, ActiveEon
2. ProActive Overview :
Programming, Scheduling, Resourcing
3. Use Case: Genomics
4. Cloud Seeding



**Parallelism+Distribution
with Strong Model: Speed & Safety**

Key Objectives

- ❑ **Parallel Programming Model and Tools**
 - **Badly needed**
 - **for the masses**
 - **for new architectures: Multi-Cores & Clouds**
- ❑ **As Effective as possible:**
 - **Efficient**
 - **However Programmer Productivity is first KSF**
- ❑ **For both Multi-cores and Distributed**
 - **Actually the way around**
- ❑ **Handling of “Large-scale”: up to 4 000 so far**



1. Background

- A joint team, Now about 35 persons
- 2004: First ProActive User Group
- 2009, April: ProActive 4.1, Distributed & Parallel:
From Multi-cores to Enterprise GRIDs



OASIS Team Composition (35)

□ Researchers (5):

- D. Caromel (UNSA, Det. INRIA)
- E. Madelaine (INRIA)
- F. Baude (UNSA)
- F. Huet (UNSA)
- L. Henrio (CNRS)

□ PhDs (11):

- Antonio Cansado (INRIA, Conic)
- Brian Amedro (SCS-Agos)
- Cristian Ruz (INRIA, Conicyt)
- Elton Mathias (INRIA-Cordi)
- Imen Filali (SCS-Agos / FP7 SC)
- Marcela Rivera (INRIA, Conicyt)
- Muhammad Khan (STIC-Asia)
- Paul Naoumenko (INRIA/Région)
- Viet Dung Doan (FP6 Bionets)
- Virginie Contes (SOA4ALL)
- Guilherme Pezzi (AGOS, CIFR)

□ + Visitors + Interns



Located in Sophia Antipolis, between
Nice and Cannes,
Visitors and Students Welcome!

Startup Company Born of INRIA



Some Partners:

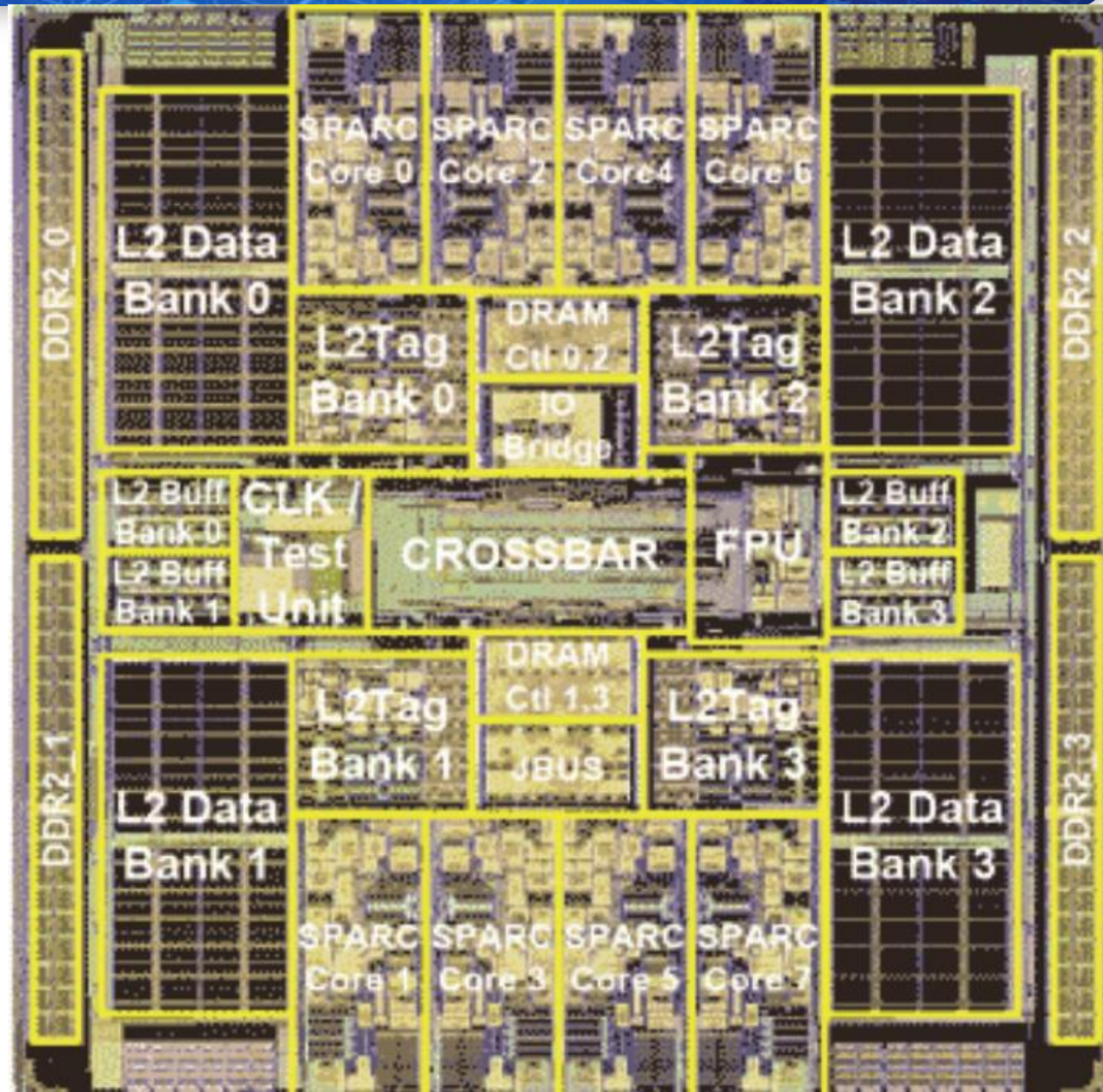


- ❑ Co-developing, Support for [ProActive Parallel Suite](#)
- ❑ Worldwide Customers: Fr, UK, Boston USA

Multi-Cores

Symmetrical Multi-Core: 8-ways Niagara II

- ❑ 8 cores
- ❑ 4 Native threads per core
- ❑ Linux see 32 cores!



Multi-Cores

A Few Key Points

- ❑ **Not Shared Memory (NUMA)**
- ❑ **Moore's Law rephrased:**
 - Nb. of Cores double every 18 to 24 months**
- ❑ **Key expected Milestones: Cores per Chips (OTS)**
 - **2010: 32 to 64**
 - **2012: 64 to 128**
 - **2014: 128 to 256**
- 1 Million Cores Parallel Machines in 2012**
- 100 M cores coming in 2020**
- ❑ **Multi-Cores are NUMA, and turning Heterogeneous (GPU)**
They are turning into SoC with NoC: NOT SMP!

2. Overview

ProActive Parallel Suite

ProActive Parallel Suite



Parallel Acceleration Toolkit in Java:

- Java Parallel Programming
+ Legacy-Code + Wrapping and Control
- Taskflow Scheduling
- Resource Manager

Multi-Core + Distributed

Open Source Used in production by industry



OW2: Object Web + Orient Ware



OW2
Consortium

Leading Open Source Middleware

OW2: Object Web + Orient Ware

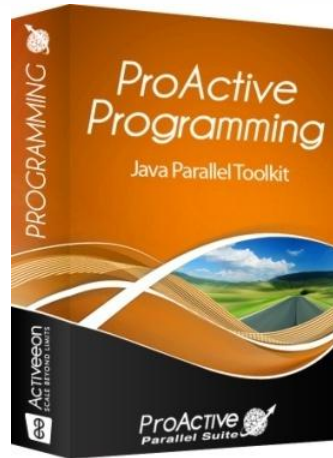


LIU Jiangning (CVIC SE), Prof. MA Dianfu (BEIHANG UNIVERSITY), Prof. WEI Jun (ISCAS), Prof. JIA Yan (NUDT), Prof. WANG Huaiming (NUDT), Mr. YUCHI Jan (MOST), Jean-Pierre Laisné (BULL), Prof. HUI Jinpeng (BEIHANG UNIVERSITY), Julie Marguerite (INRIA), ZHOU Minghui (PEKING UNIVERSITY), Stephane Grumbach (French Embassy), Hongbo XU (GMRC), ZHOU Bin (NUDT), Than Ha Ngo (French Embassy).

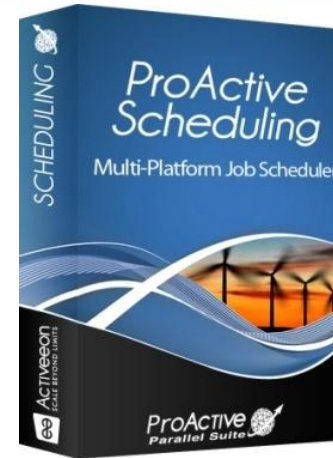
OW2
Consortium

Leading Open Source Middleware

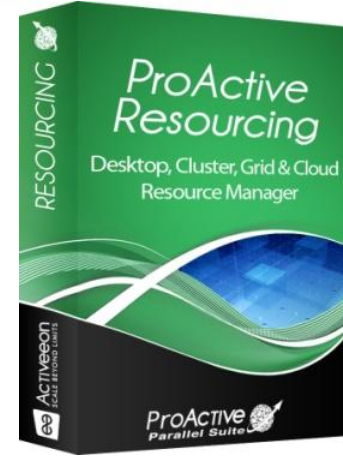
Product: ProActive Parallel Suite



Java Parallel Toolkit



Multi-Platform Job Scheduler



Resource Manager

amaDEUS
Your technology partner

Used in Production Today:
50 Cores → 300 Cores early 2010

Strong Differentiation:

- Java Parallel Programming + Integration +
- Portability: Linux, Windows, Mac +
- Versatility: Desktops, Cluster, Grid, Clouds = Perfect Flexibility

ProActive Parallel Suite

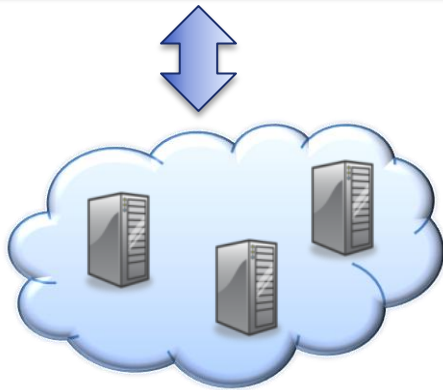
□ Three fully compatible modules:



Programming

Scheduling

Resource Management



Pending (674)				Running (90)				Finished (13)						
Id	State	User	Priority	Name	Id	State	User	Priority	Name	Id	State	User	Priority	Name
1996	Pending	j	Normal	job_with_dep	1323	Running	7/8	admin	Normal	101	Finished	j	Low	job_proactive
1997	Pending	j	Normal	job_with_dep	1324	Running	4/8	user1	Normal	102	Finished	j	Low	job_proactive
1998	Pending	j	Normal	job_with_dep	1325	Running	7/8	admin	Normal	103	Finished	j	Low	job_proactive
1999	Pending	j	Normal	job_with_dep	1326	Running	4/8	user1	Normal	104	Finished	j	Low	job_proactive
2000	Pending	j	Normal	job_with_dep	1327	Running	4/8	user1	Normal	105	Finished	j	Low	job_proactive
2001	Pending	j	Normal	job_with_dep	1328	Running	4/8	user1	Normal	106	Finished	j	Low	job_proactive
2002	Pending	j	Normal	job_with_dep	1329	Running	7/8	admin	Normal	107	Finished	j	Low	job_proactive
2003	Pending	j	Normal	job_with_dep	1330	Running	7/8	admin	Normal	108	Finished	j	Low	job_proactive
2004	Pending	j	Normal	job_with_dep	1331	Running	4/8	user1	Normal	109	Finished	j	Low	job_proactive
2005	Pending	j	Normal	job_with_dep	1332	Running	7/8	admin	Normal	110	Finished	j	Low	job_proactive
2006	Pending	j	Normal	job_with_dep	1333	Running	4/8	user1	Normal	111	Finished	j	Low	job_proactive
2007	Pending	j	Normal	job_with_dep	1334	Running	7/8	admin	Normal	112	Finished	user1	Normal	job_with_dep
2008	Pending	j	Normal	job_with_dep	1335	Running	2/8	user1	Normal	113	Finished	admin	Normal	job_with_dep
2009	Pending	j	Normal	job_with_dep	1336	Running	2/8	user1	Normal	114	Finished	admin	Normal	job_with_dep
2010	Pending	j	Normal	job_with_dep	1337	Running	7/8	user1	Normal	115	Finished	admin	Normal	job_with_dep

ProActive Contributors

Abhijeet Gaikwad
(Option Pricing)
Abhishek-Rajeev Gupta
Antonio Cansado
Baptiste De Stefano
Bastien Sauvan
Brian Amedro (SPMD)
Cédric Dalmaso (Component)
Clement Mathieu (Core,
GCM Deployment)
Elaine Isnard
Elton Mathias
Eric Madelaine
Etienne Vallette-De-Osia
Fabien Viale (Matlab, Scilab)
Fabrice Huet (Mobility, P2P)
Florin Bratu
Franca Perrina
Francoise Baude
Germain Sigety (Scheduling)
Guillaume Laurent
Guilherme Perretti Pezzi
Imen Filiali
Jean-Luc Scheefer (Scheduling)

Jean-Michael Guillamume
Johann Fradj (Scheduling)
Jonathan Martin
Julian Krzeminski
Kamran Qadir
Khan Muhammad
Laurent Vanni
Ludovic Henrio
Marcela Rivera
Mario Leyton (Skeleton)
Maxime Menant
Nicolas Dodelin
Olivier Helin
Paul Naoumenko
Regis Gascon
Tomasz Dobek
Vasile Jureschi (Technical Writer)
Viet Dong Doan
Vincent Cave (Legacy Wrapping)
Virginie Contes (OSGi, WS)
Yu Feng
Yulai Yuan
Zhihui Dai

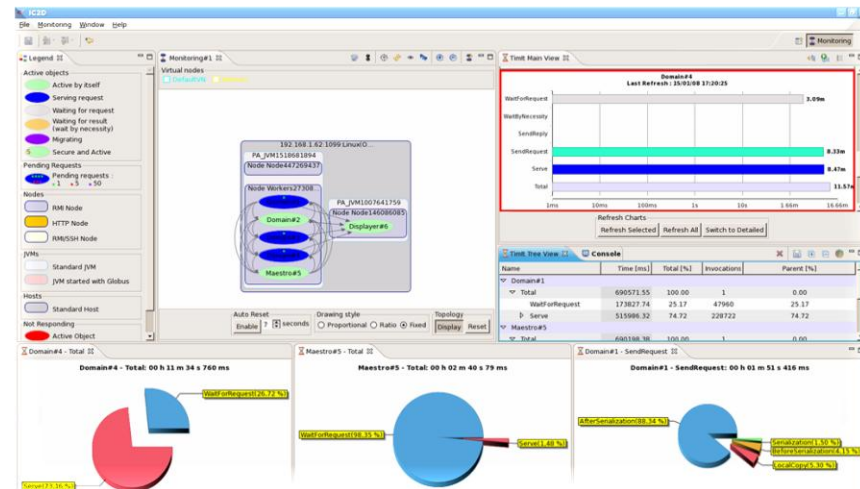
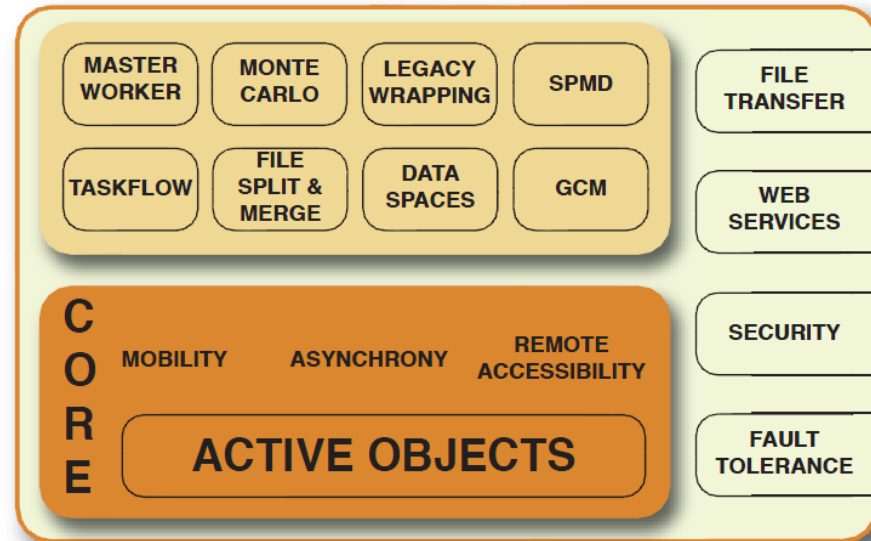
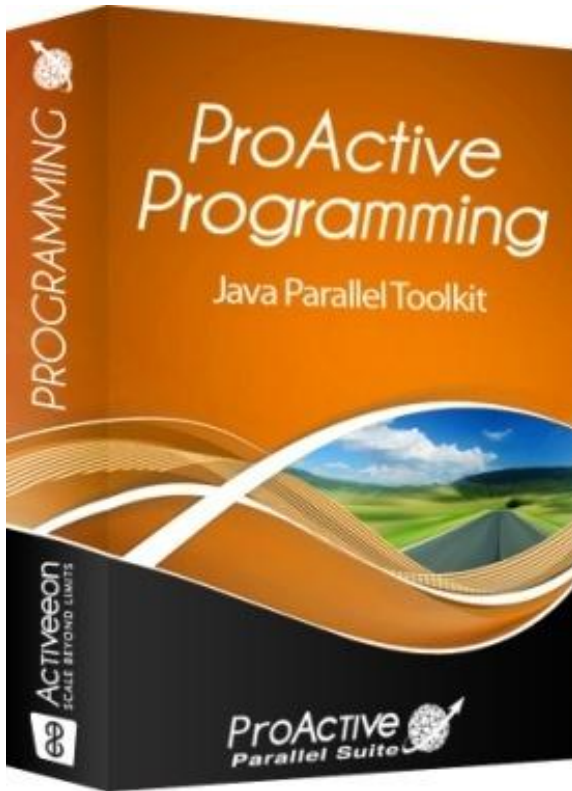
Arnaud Contes (Core, Security, Debugging)
Christian Delbé (FT, Scheduling)
Emil Salageanu (IC2D, Agent)
Vladimir Bodnartchouk (IC2D, TimIt)

Alexandre di
Costanzo (P2P, B&B)
Boutheina Bennour
Guillaume Chazarain (DGC)
Julien Vayssiere
(MOP, Active Objects)

Lionel Mestre
Laurent Baduel (Group
Communications)
Matthieu Morel (Initial
Component Work)
Nadia Rinaldo
(Core, Deployment)
Romain Quilici

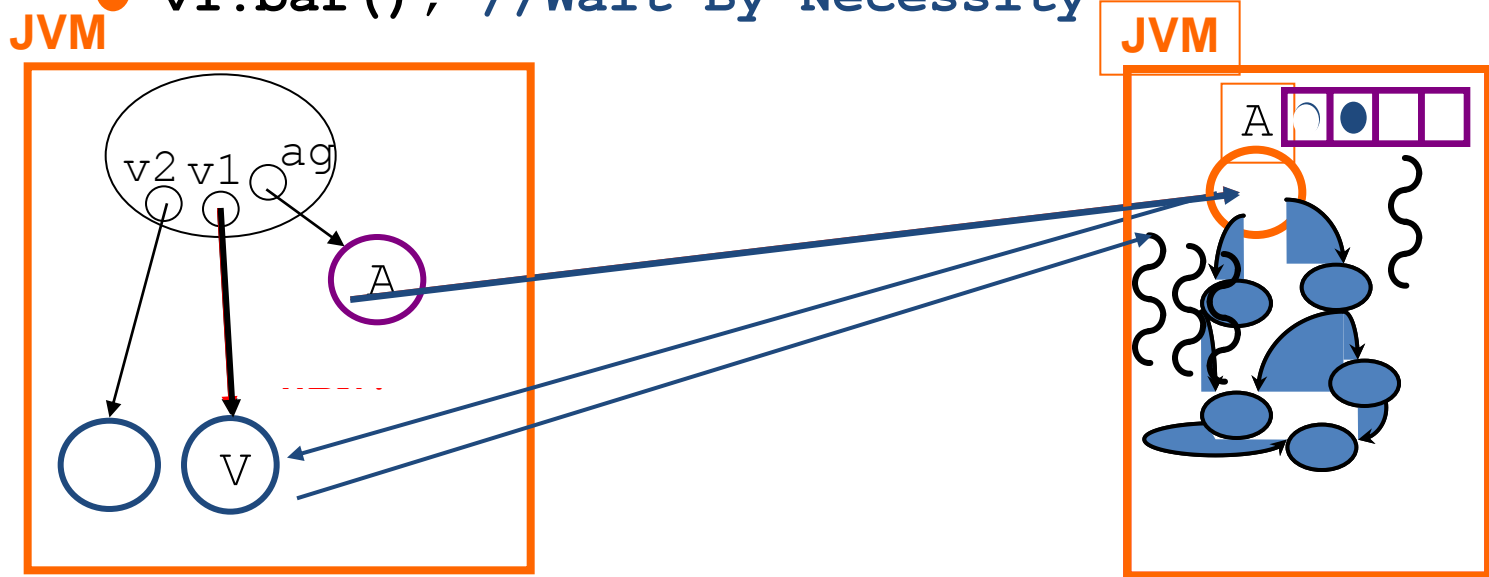
ProActive Programming: Active Objects

ProActive Programming

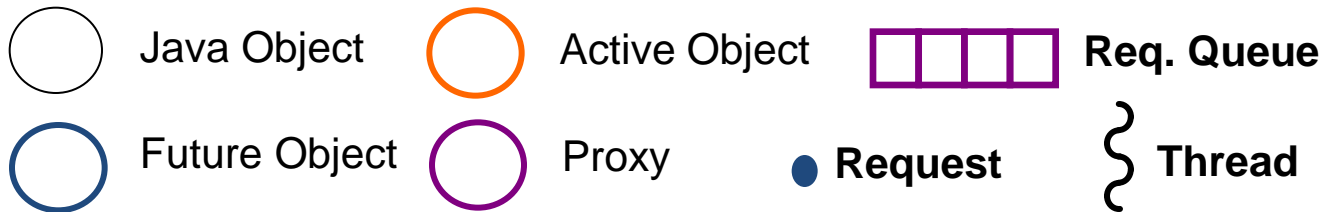


ProActive : Active objects

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



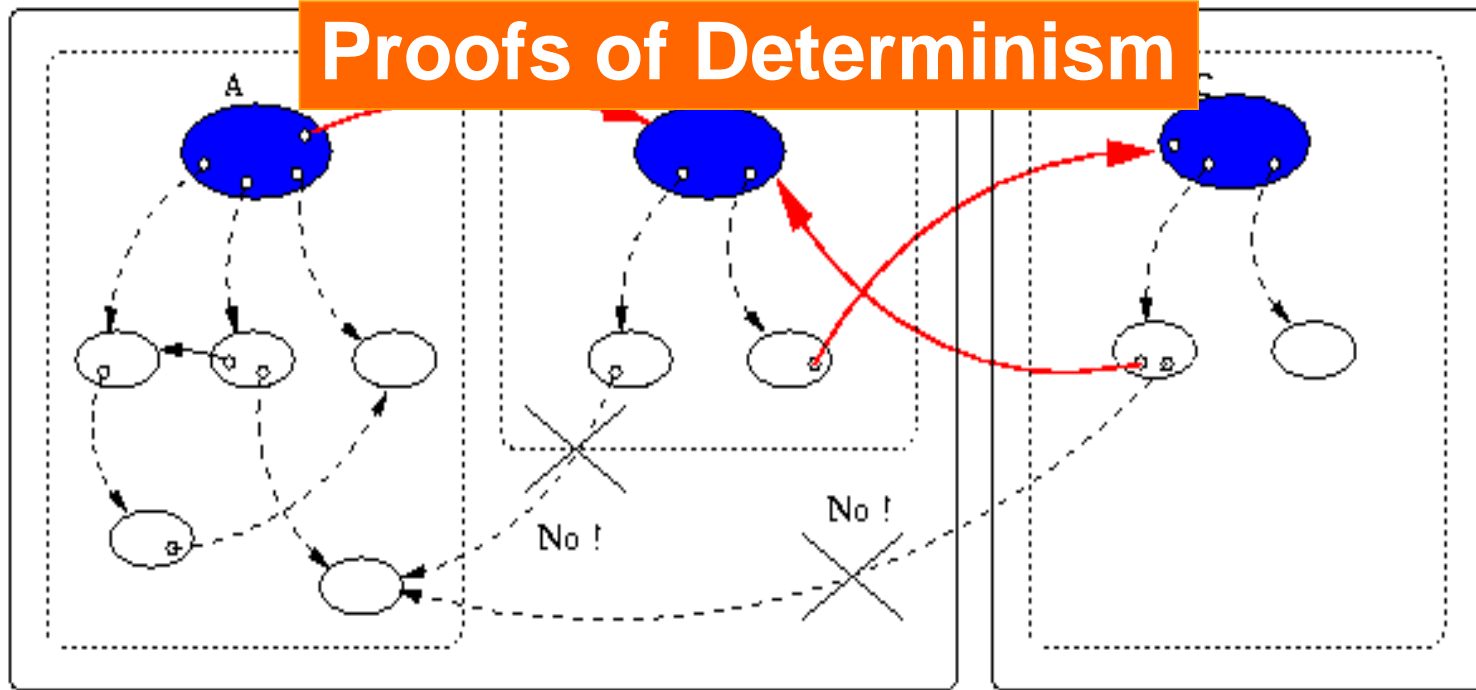
Wait-By-Necessity
is a
Dataflow
Synchronization



Standard system at Runtime: No Sharing

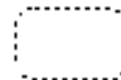
NoC: Network On Chip

Proofs of Determinism



Active Object

-----> Synchronous Call



Sub System



Passive Object

-----> Asynchronous Call



Address Space

(2) ASP: Asynchronous Sequential Processes

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

Local

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

Creating an Activity

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

Sending a Request

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

Service

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

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

Sending a Reply



Key Point:
Locality will more than ever be
Fundamental

- ❑ Let the programmer control it
- ❑ No global shared memory

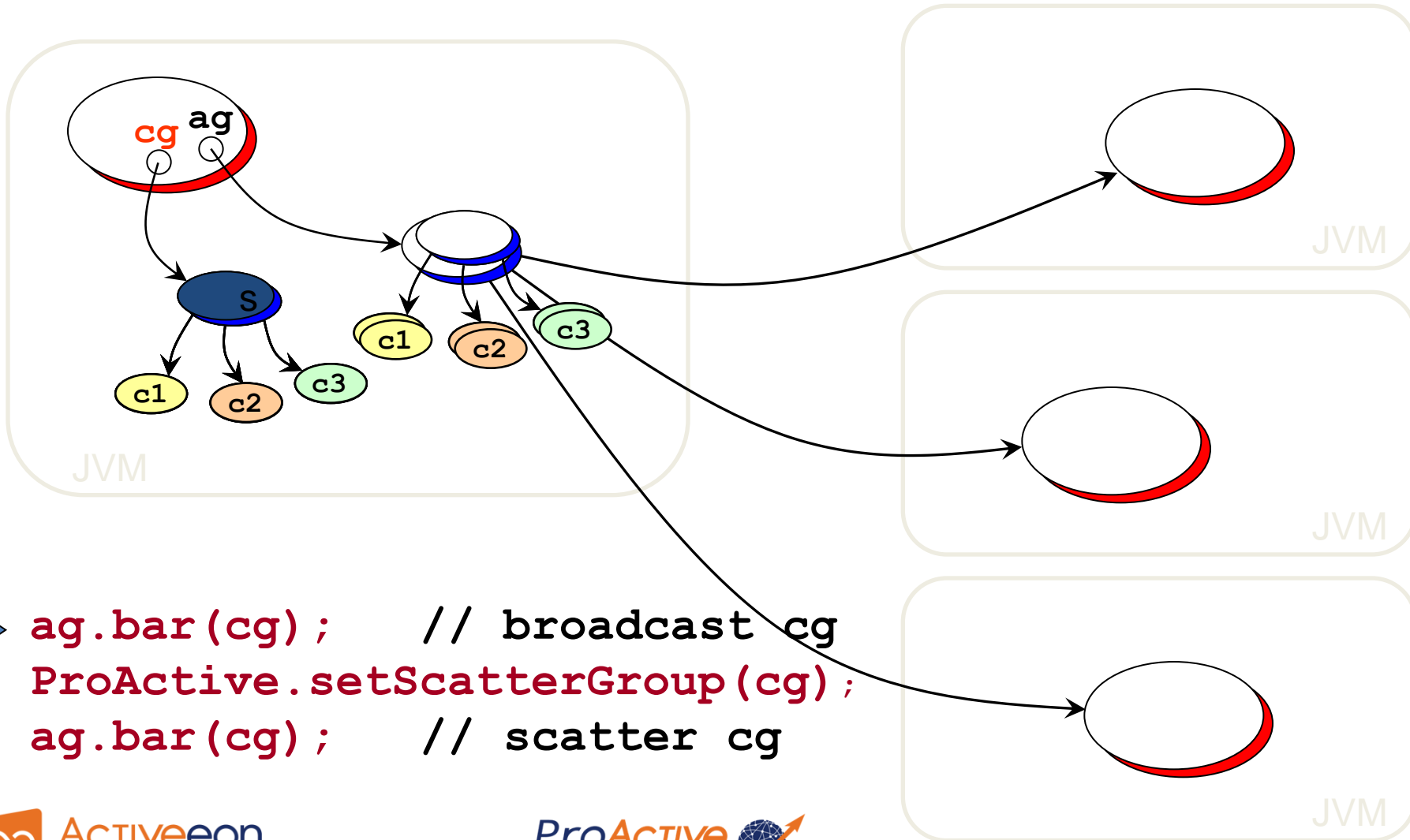
*At user choice **PGAS**: Partitioned Global Address Space*

TYPED ASYNCHRONOUS GROUPS

Broadcast and Scatter

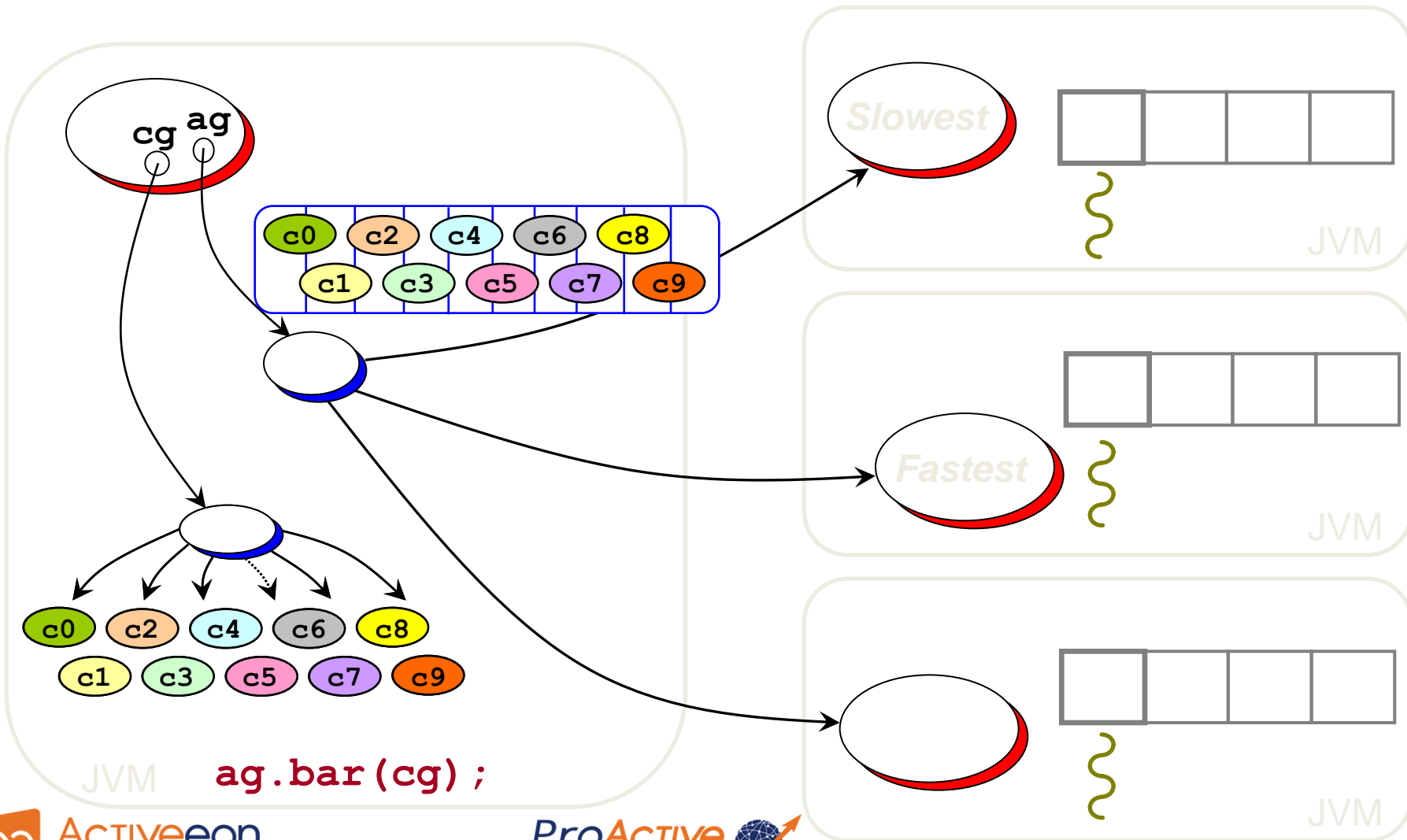
Broadcast is the default behavior

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

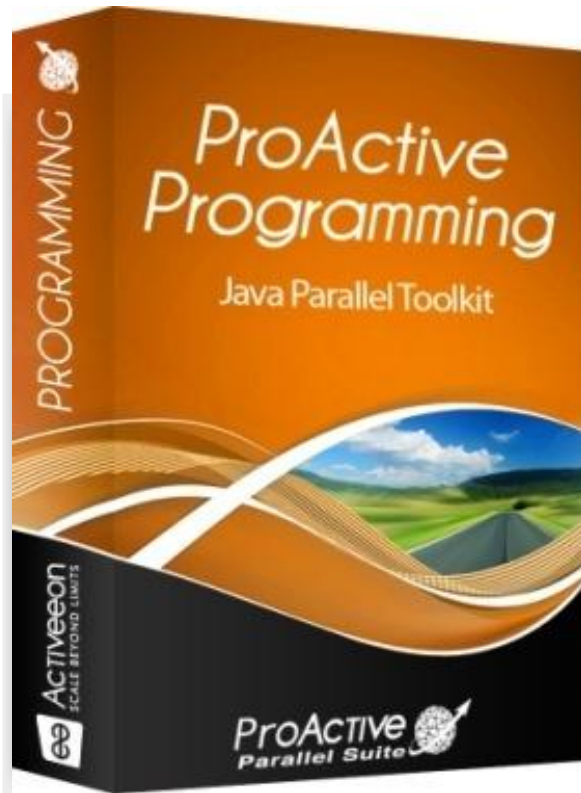


JVM **ag.bar (cg) ;**

Abstractions for Parallelism

The right Tool to do the Task right

ProActive Parallel Suite



- Workflows in Java
- Master/Workers
- SPMD
- Components
- ...

Core API
Active Objects
Asynchrony
Futures
Groups
Mobile Agents
MOP / AOP

Components: GCM Standard

GridCOMP Partners

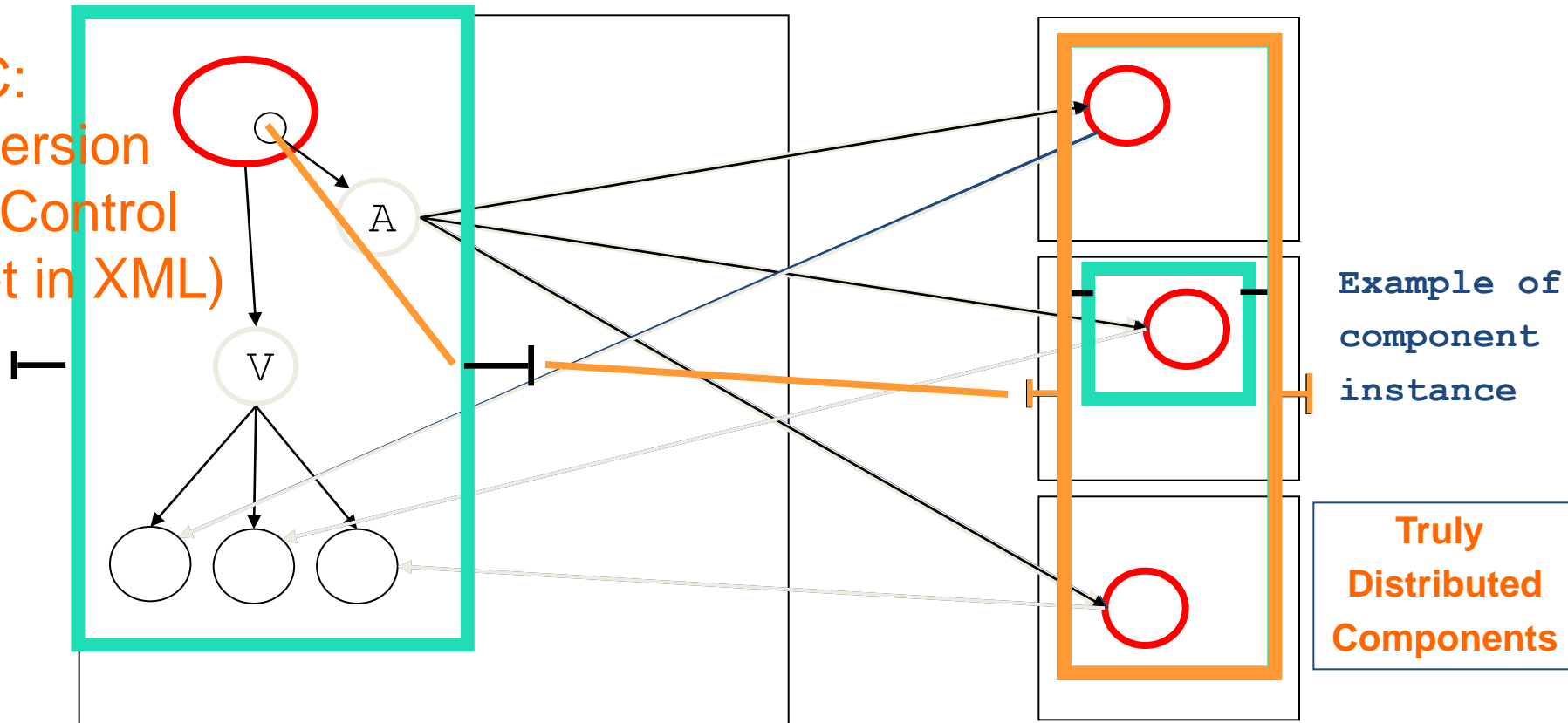


THE UNIVERSITY OF
MELBOURNE



Objects to Distributed Components

IoC:
Inversion
Of Control
(set in XML)



○ Typed Group

○ Java or Active Object



JVM

GRIDS for Finance
& Telecommunications

V

GRIDS

20-24 Oct
INRIA – Sophia



Information & Registration: www.etsi.org/plugtests/GRID2008/GRID.htm

With the support of



European partners

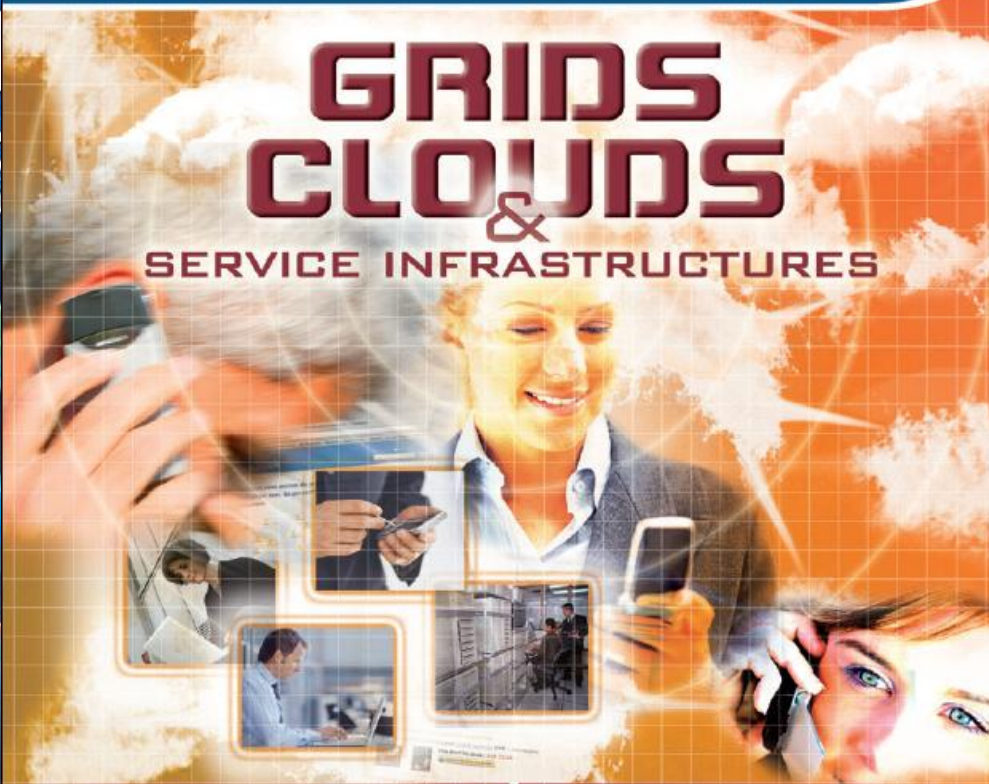


Industrial sponsors



GRIDS & CLOUDS

SERVICE INFRASTRUCTURES



PLUGTESTS
30 NOV. - 02 DEC.

WORKSHOP
02 - 03 DEC.

Information & registration at <http://www.etsi.org/plugtests/GRID09/GRID.htm>

30 NOV. - 03 DEC. 2009
ETSI - FRANCE

From 2004 to 2008:

□ 2004 Grid Plugtests:

Winner: Univ CHILE

Deployed 560 Workers all over the world
on a very heterogeneous infrastructure (no VO)

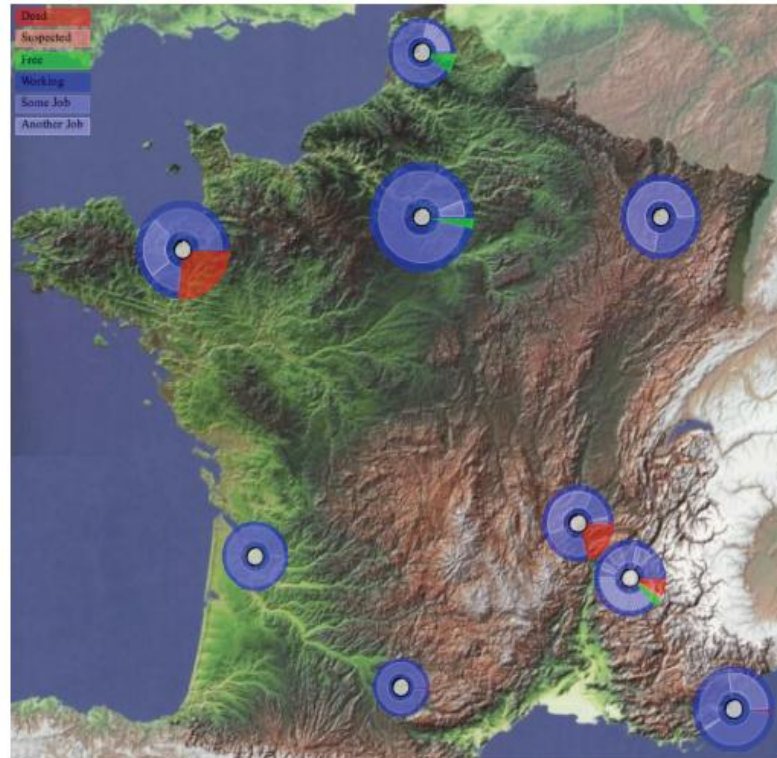
□ 2008 Grid Plugtests:

KA-API, MOAIS Grenoble: 3609 Nodes

ACT, China: Beihang University, Beijing, China:
4329 Nodes



Grid'5000*



Lille:
500 (198)
Orsay
1000 (684)

Nancy:
500 (334)
Lyon
500 (252)

Grenoble
500 (270)
Rennes
522 (522)

Toulouse
500 (116)

Bordeaux
500 (198)

Sophia Antipoli
500 (434)



INRIA

Renater



CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

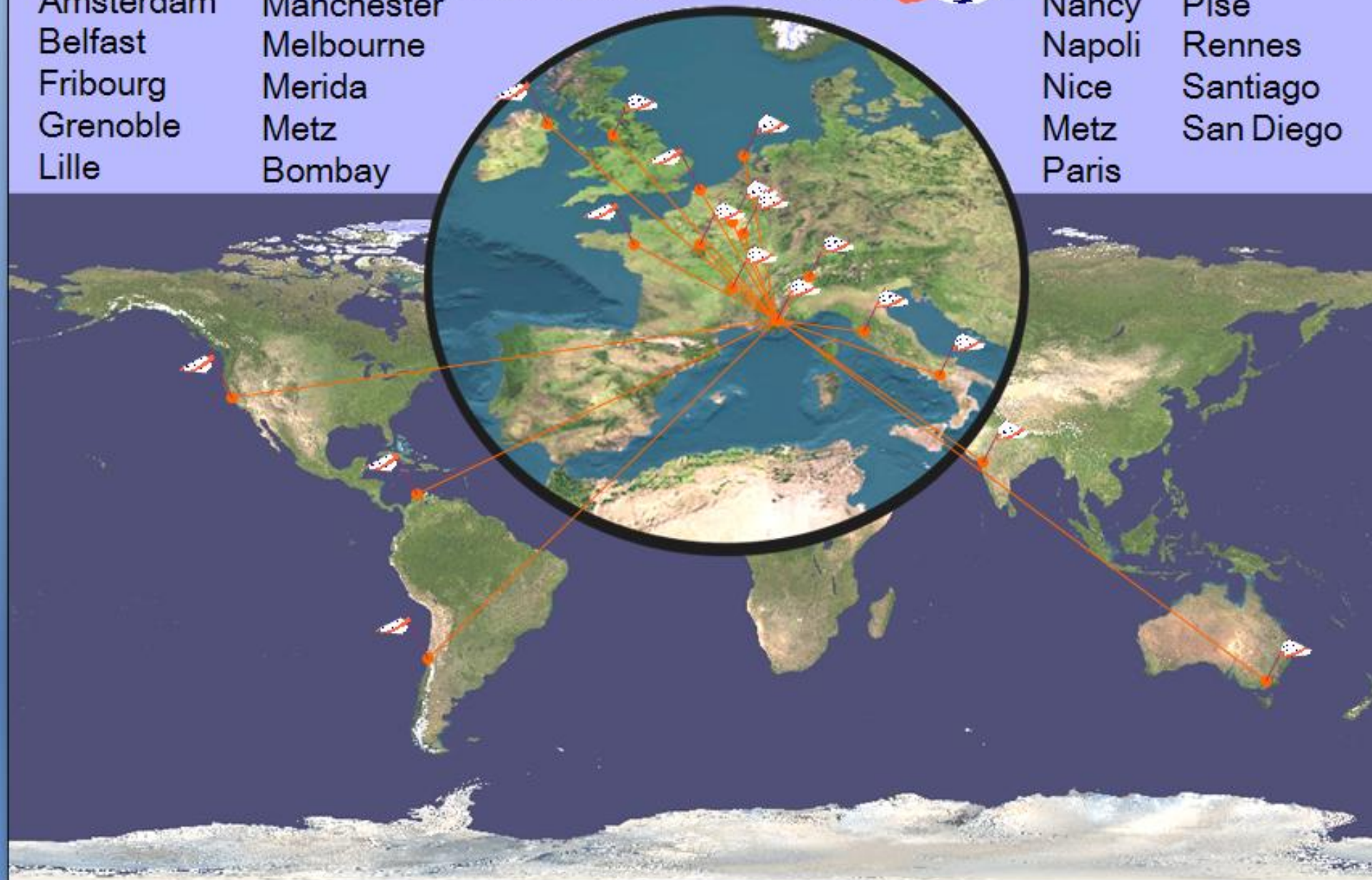


Amsterdam
Belfast
Fribourg
Grenoble
Lille

Manchester
Melbourne
Merida
Metz
Bombay

Nancy
Napoli
Nice
Metz
Paris

Pise
Rennes
Santiago
San Diego



Chinese Collaborations on Grid PlugTests

- ❑ Professor Chi
- ❑ Prof. Baoping Yan
- ❑ Hosted the IV Grid Plugtests [Grid@works](#) 2007
- ❑ CNIC: Computer and Network Information Center
- ❑ SCC AS: Super Computing Center of AS



- ❑ Prof. Ji Wang
- ❑ In EchoGrid, Chinese Leader of OW2
- ❑ NUDT: National Univ. of Defense Technology
- ❑ PDL: Laboratory of Parallel & Distributed Processing



Infrastructure tested in Plugtests and in GCM Deployment Standard

□ 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 (Windows HPC Server 2008)
- Sun Grid Engine, OAR, PBS / Torque, PRUN

□ Clouds:

- Amazon EC2

GCM Official Standardization

Grid Component Model



Overall, the standardization is supported by industrials:

BT, FT-Orange, Nokia-Siemens, NEC, Telefonica, Alcatel-Lucent, Huawei ...

Infrastructure tested in Plugtests and in GCM Deployment Standard

Interoperability:

Cloud will start with existing IT infrastructure,
Build Non Intrusive Cloud with ProActive

❑ 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 (Windows HPC Server 2008)
- Sun Grid Engine, OAR, PBS / Torque, PRUN

❑ Clouds:

- Amazon EC2

IC2D: Optimizing

The screenshot displays the Eclipse IDE interface with two main views: **Monitoring View** and **Job Monitoring View**.

Monitoring View: Shows a hierarchical tree of virtual nodes. The selected node is **PA_JVM1357457629_...** under **DefaultVN (JOB-1357457629)**. The tree includes:

- Virtual nodes:** **Renderer**, **DefaultVN**, **Dispatcher**, **User**
- Node Node60562498...**
 - DinnerLayout#2
 - Table#3
 - Philosopher#4
 - Philosopher#5
 - Philosopher#6
 - Philosopher#7
 - Philosopher#8
- Node Renderer-127...** (C3DRendering...)
- Node Dispatcher-5...** (C3DDispatcher...)
- Node User16026446...** (C3DUser#13)
- Node Renderer1307...** (C3DRendering...)

Job Monitoring View: Shows a detailed view of the job structure. The selected job is **DefaultVN (JOB-1357457629)**. The tree includes:

- DefaultVN (JOB-1357457629)**
 - bebita.inria.fr:1099:OS un**
 - PA_JVM1357457629_**
 - Node Node60562498...**
 - DinnerLayout#2
 - Table#3 (JOB-1357457629)
 - Philosopher#4 (JOB-1357457629)
 - Philosopher#5 (JOB-1357457629)
 - Philosopher#6 (JOB-1357457629)
 - Philosopher#7 (JOB-1357457629)
 - Philosopher#8 (JOB-1357457629)
 - sidonie.inria.fr:1099:OS u**
 - Dispatcher (JOB--1672076495)**
 - User (JOB--294719007)**
 - bebita.inria.fr:1099:OS un**
 - PA_JVM-294719007_I**
 - Node User1602644**
 - C3DUser#13 (JOB-1357457629)
 - Renderer (JOB--1672076495)**
 - bebita.inria.fr:1099:OS un**
 - PA_JVM-1631909824_**
 - Node Node-455186381...**

Monitoring#1

Virtual nodes

DefaultVN matrixNode

orchidee.inria.fr:4001:Li...

PA_JVM1820960857
Node Node-632703901

Node matrixNode15...

OctTree#2

Domain#3

Domain#4

Domain#5

Domain#6

Maestro#7

BioMaestro#8

PA_JVM1370729570
Node Node-2003411204
Displayer#1

PA_JVM1949849146
Node Node-2003411204

Legend

Active objects

- Active by itself
- Serving request
- Waiting for request
- Waiting for result (wait by necessity)
- Migrating
- Secure and Active

Pending Requests

Pending requests : 1 5 50

Nodes

- RMI Node
- HTTP Node
- RMI/SSH Node

JVMs

- Standard JVM
- JVM started with Globus

Timer Tree View

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

Domain#4

Snapshot time : 18/10/07 16:20:45

GroupAsyncCall

GroupOneWayCall

AfterSerialization

Serialization

BeforeSerialization

LocalCopy

WaitForRequest

WaitByNecessity

SendReply

SendRequest

Serve

Total

1ms 10ms 100ms 1s 10s 1.66m 16.66m

Time Line View Console

BigMaestro#8

Maestro#7

Domain#6

Domain#5

Domain#4

Domain#3

OctTree#2

0ms 082.894ms 165.788ms 248.682ms 331.576ms 414.470ms 497.364ms 580.258ms 663.152ms 746.4

Refresh Charts

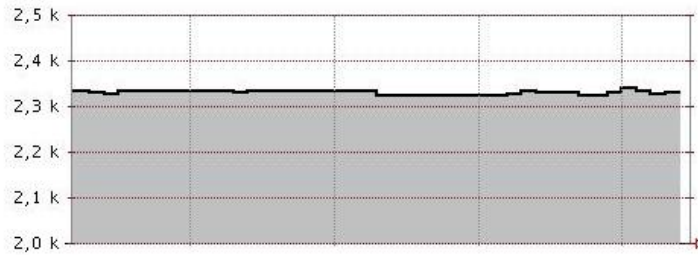
Refresh Selected Refresh All Switch To Basic

ChartIt

PA_JVM251111462

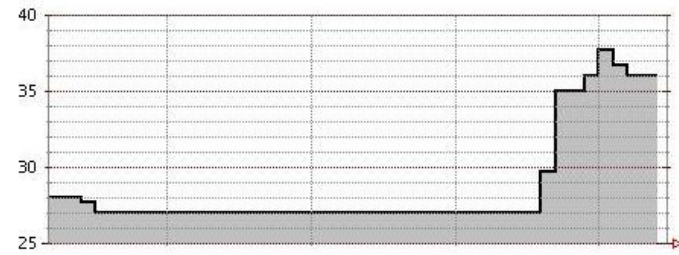
Charts

Chart#1 [LoadedClassCount]



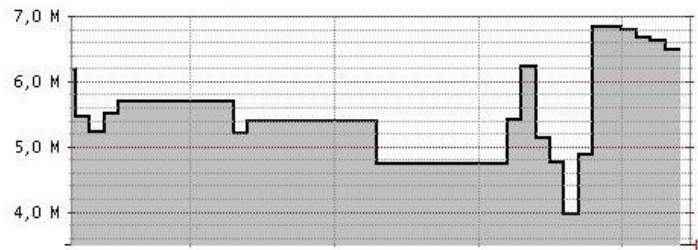
LoadedClassCount

Chart#3 [ThreadCount]



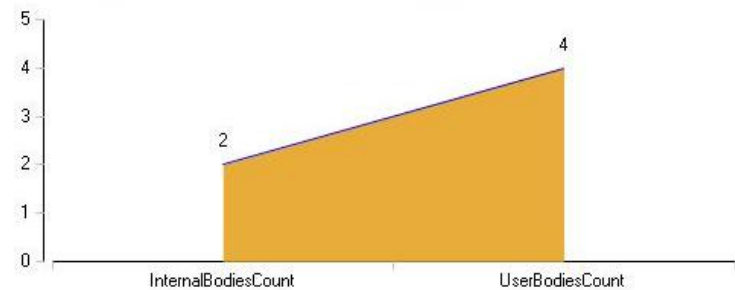
ThreadCount

Chart#2 [UsedHeapMemory]

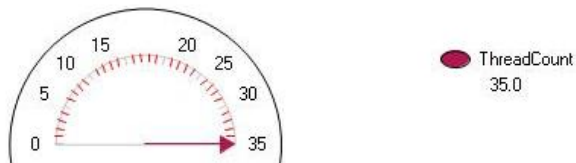


UsedHeapMemory

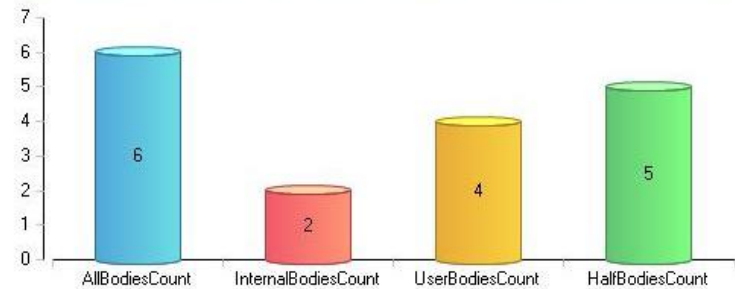
Chart#4 [InternalBodiesCount, UserBodiesCount]



Chart#5 [ThreadCount] Meter



Chart#6 [AllBodiesCount, InternalBodiesCount, UserBodiesCount, HalfBodiesCount]



Pies for Analysis and Optimization

IC2D Monitoring Window Help

Monitoring#1

Virtual nodes: DefaultTVN Workers

192.168.1.62:1099:Linux(O...)

PA_JVM1518681894
Node Node447269437

Node Workers27308...

PA_JVM1007641759
Node Node146086085

Displayer#6

Domain#1

Domain#2

Domain#3

Domain#4

Maestro#5

Auto Reset: Enable 7 seconds

Drawing style: Proportional Ratio Fixed

Topology:

Domain#4 - Total: 00 h 11 m 34 s 760 ms

Maestro#5 - Total: 00 h 02 m 40 s 79 ms

Domain#1 - SendRequest: 00 h 01 m 51 s 416 ms

Domain#4 - Performance Metrics (Last Refresh: 15/01/08 17:20:25)

Metric	Value
WaitForRequest	3.09m
WaitByNecessity	-
SendReply	-
SendRequest	8.33m
Serve	8.47m
Total	11.57m

Refresh Charts: Refresh Selected Refresh All Switch to Detailed

Timit Tree View Console

Name	Time [ms]	Total [%]	Invocations	Parent [%]
Domain#1				
Total	690571.55	100.00	1	0.00
WaitForRequest	173827.74	25.17	47960	25.17
Serve	515986.32	74.72	228722	74.72
Maestro#5				
Total	690198.38	100.00	1	0.00

Domain#4 - Total: 00 h 11 m 34 s 760 ms

Serve(73.16%)

WaitForRequest(26.72%)

Maestro#5 - Total: 00 h 02 m 40 s 79 ms

WaitForRequest(98.35%)

Serve(1.48%)

Domain#1 - SendRequest: 00 h 01 m 51 s 416 ms

AfterSerialization(88.34%)

LocalCopy(5.30%)

BeforeSerialization(4.15%)

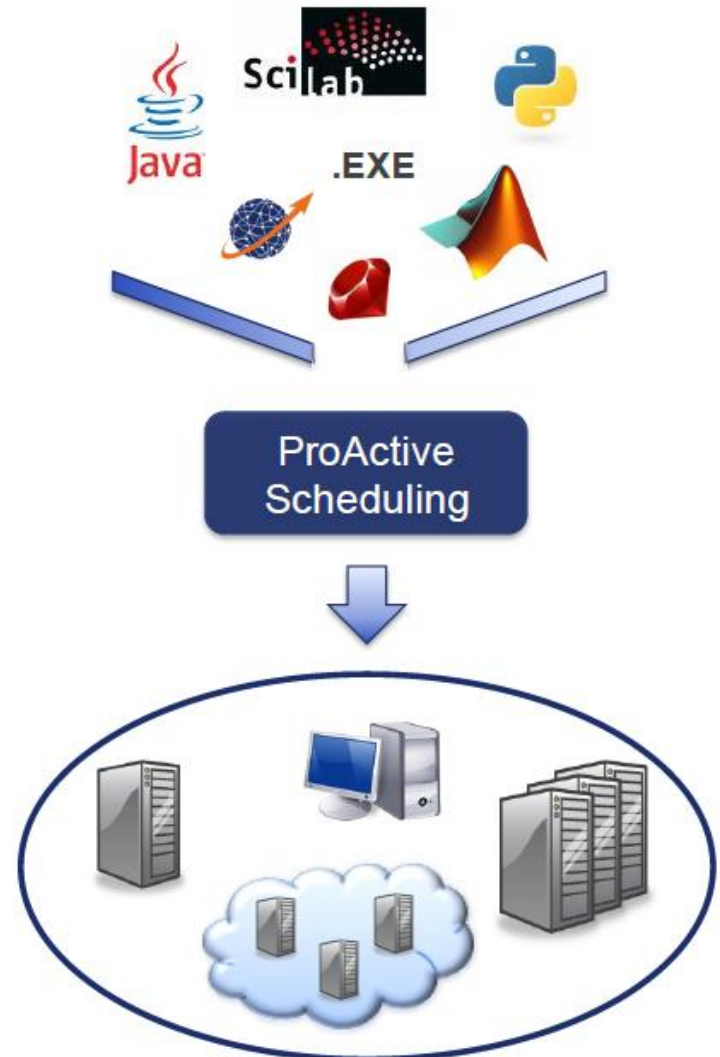
Serialization(1.50%)

Video 1: IC2D Optimizing Monitoring, Debugging, Optimizing



Scheduling & Resourcing

ProActive Scheduling



ProActive Scheduling Big Picture

ProActive Scheduler

File Window Help

Scheduler

Jobs

Pending (674) Running (60) Finished (31)

Id	State	User	Priority	Name
1996	Pending	jl	Normal	job_with_dep
1997	Pending	jl	Normal	job_with_dep
1998	Pending	jl	Normal	job_with_dep
1999	Pending	jl	Normal	job_with_dep
2000	Pending	jl	Normal	job_with_dep
2001	Pending	jl	Normal	job_with_dep
2002	Pending	jl	Normal	job_with_dep
2003	Pending	jl	Normal	job_with_dep
2004	Pending	jl	Normal	job_with_dep
2005	Pending	jl	Normal	job_with_dep
2006	Pending	jl	Normal	job_with_dep
2007	Pending	jl	Normal	job_with_dep
2008	Pending	jl	Normal	job_with_dep
2009	Pending	jl	Normal	job_with_dep
2010	Pending	jl	Normal	job_with_dep

Id	State	Progress	# Finished	User	Priority
1313	Running	<div style="width: 25%;"></div>	4/8	user1	Normal
1314	Running	<div style="width: 25%;"></div>	4/8	user1	Normal
1315	Running	<div style="width: 25%;"></div>	7/8	admin	Normal
1316	Running	<div style="width: 25%;"></div>	4/8	user1	Normal
1317	Running	<div style="width: 25%;"></div>	7/8	admin	Normal
1318	Running	<div style="width: 25%;"></div>	4/8	user1	Normal
1319	Running	<div style="width: 25%;"></div>	7/8	admin	Normal
1320	Running	<div style="width: 25%;"></div>	3/8	user1	Normal
1321	Running	<div style="width: 25%;"></div>	7/8	admin	Normal
1322	Running	<div style="width: 25%;"></div>	3/8	user1	Normal
1323	Running	<div style="width: 25%;"></div>	7/8	admin	Normal
1324	Running	<div style="width: 25%;"></div>	2/8	user1	Normal
1325	Running	<div style="width: 25%;"></div>	2/8	user1	Normal
1326	Running	<div style="width: 25%;"></div>	2/8	user1	Normal
1327	Running	<div style="width: 25%;"></div>	2/8	user1	Normal

Id	State	User	Priority	Name
010	Finished	jl	Low	job_proActive
008	Finished	jl	Low	job_proActive
005	Finished	jl	Low	job_proActive
001	Finished	jl	Low	job_proActive
006	Finished	jl	Low	job_proActive
004	Finished	jl	Low	job_proActive
003	Finished	jl	Low	job_proActive
009	Finished	jl	Low	job_proActive
007	Finished	jl	Low	job_proActive
002	Finished	jl	Low	job_proActive
245	Finished	user1	Normal	job_with_dep
246	Finished	user1	Normal	job_with_dep
247	Finished	user1	Normal	job_with_dep
252	Finished	admin	Normal	job_with_dep
253	Finished	admin	Normal	job_with_dep

RESUMED

Console Tasks

Job Info Result Preview

Job 2008 has 8 tasks

Id	State	Name	Host name	Start time	Finished time	Re-run	Description
200800	Submitted	task4	n/a	Not yet	Not yet	0/2	This task will sleep 5s
200800	Submitted	task2	n/a	Not yet	Not yet	0/1	This task will sleep 10s
200800	Submitted	task6	n/a	Not yet	Not yet	0/1	This task will sleep 8s
200800	Submitted	task1	n/a	Not yet	Not yet	0/2	This task will sleep 6s
200800	Submitted	task5	n/a	Not yet	Not yet	0/1	This task will sleep 2s
200800	Submitted	task7	n/a	Not yet	Not yet	0/2	This task will sleep 6s
200800	Submitted	task3	n/a	Not yet	Not yet	0/1	This task will sleep 4s
200800	Submitted	task8	n/a	Not yet	Not yet	0/1	This task will sleep 6s

Property	Value
Id	2008
State	Pending
Name	job_with_dep
Priority	Normal
Pending tasks number	0
Running tasks number	0
Finished tasks number	0
Total tasks number	8
Submitted time	09:40:06 03/12/08
Started time	Not yet
Finished time	Not yet

down nodes 0



RES

Scheduler: User Interface

ProActive Scheduler

File Window Help

Scheduler

Jobs

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 Tasks Users

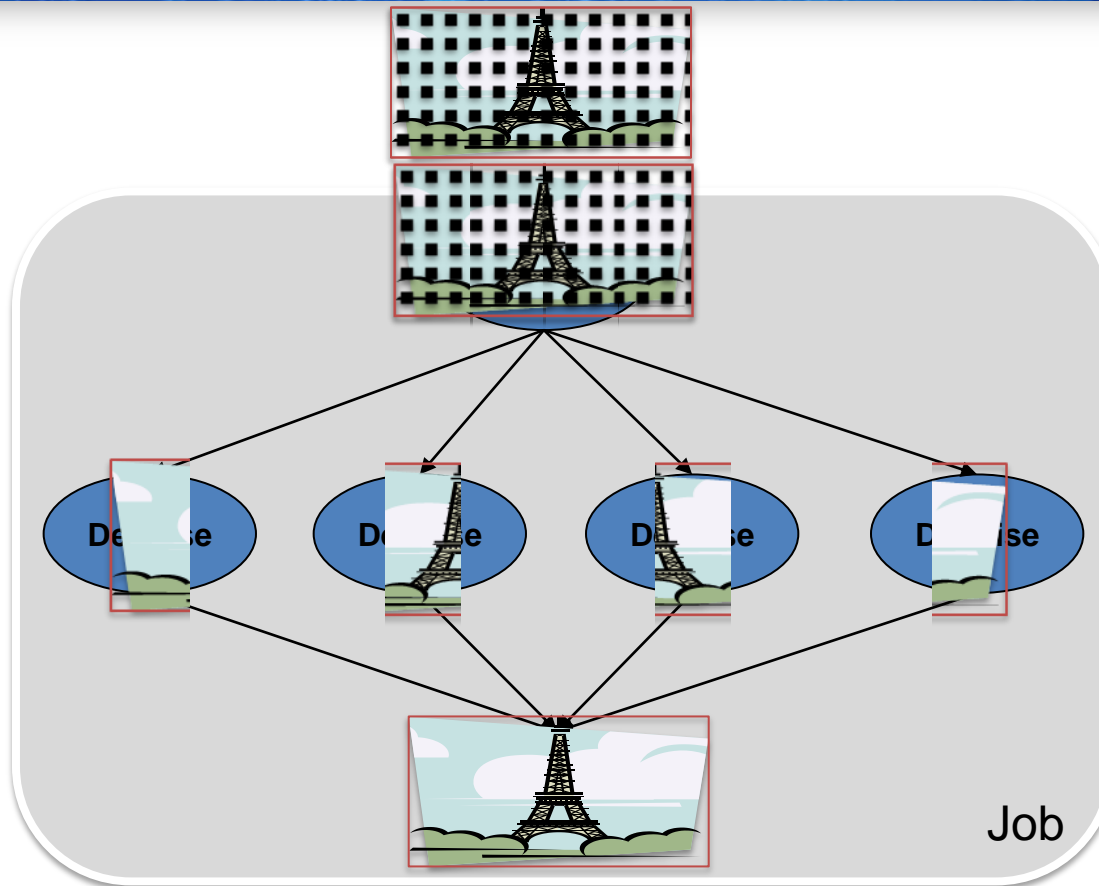
Job 55 has 2 tasks

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 Result Preview

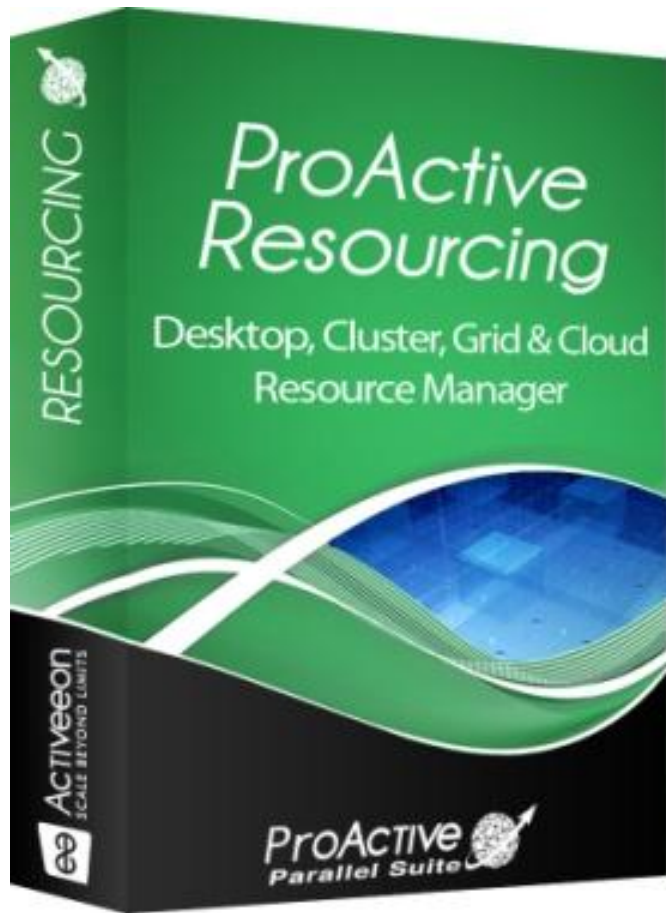
Property	Value
Id	55
State	Running
Name	job_2_tasks
Priority	Low
Pending tasks number	0
Running tasks number	2
Finished tasks number	0
Total tasks number	2
Submitted time	16:09:28 08/27/08

Another Example : Picture Denoising



- with **selection** on native executable availability (ImageMagik, GREYstoration)
 - Multi-platform selection and command generation
- with **file transfer** in pre/post scripts

ProActive Resourcing



RESOURCING User Interface

The screenshot displays the ProActive Resource Manager (PA) User Interface. The main window is titled "ProActive Resource Manager" and includes a menu bar (File, Connection, Actions, Help, Window) and a toolbar with a "Shutdown" button. The interface is divided into several panes:

- Tab Explorer / Tree Explorer:** Shows a hierarchical tree of resources. The tree is expanded to show three main resource groups:
 - PA_JVM2114960478
 - rmi://eon14.inria.fr:1099/PA_JVM2114960478_GCMNode-0
 - rmi://eon14.inria.fr:1099/PA_JVM2114960478_GCMNode-1
 - rmi://eon14.inria.fr:1099/PA_JVM2114960478_GCMNode-2
 - PA_JVM477486534
 - PA_JVM2003420561
 - rmi://eon14.inria.fr:1099/PA_JVM2003420561_GCMNode-0
 - rmi://eon14.inria.fr:1099/PA_JVM2003420561_GCMNode-1
 - rmi://eon14.inria.fr:1099/PA_JVM2003420561_GCMNode-2
- Compact View:** A grid of colored circles representing the state of individual nodes. The colors range from green (free) to yellow (busy) to red (down).
- JMX Monitoring:** Contains three charts:
 - Activity History:** A line graph showing activity levels over time, with a peak around 16:50.
 - Node States Peaks:** A bar chart showing the maximum number of nodes in different states: Max Free (330), Max Busy (80), Max ToBeReleased (0), and Max Down (0).
 - Free Nodes History:** A line graph showing the number of free nodes over time, fluctuating between approximately 250 and 350.
- Overview / Charts:** A section with tabs for "Statistics" and "Info". The "Statistics" tab is active, showing a table of resource counts:

state	aggregate
# free nodes	272
# busy nodes	52
# down nodes	6

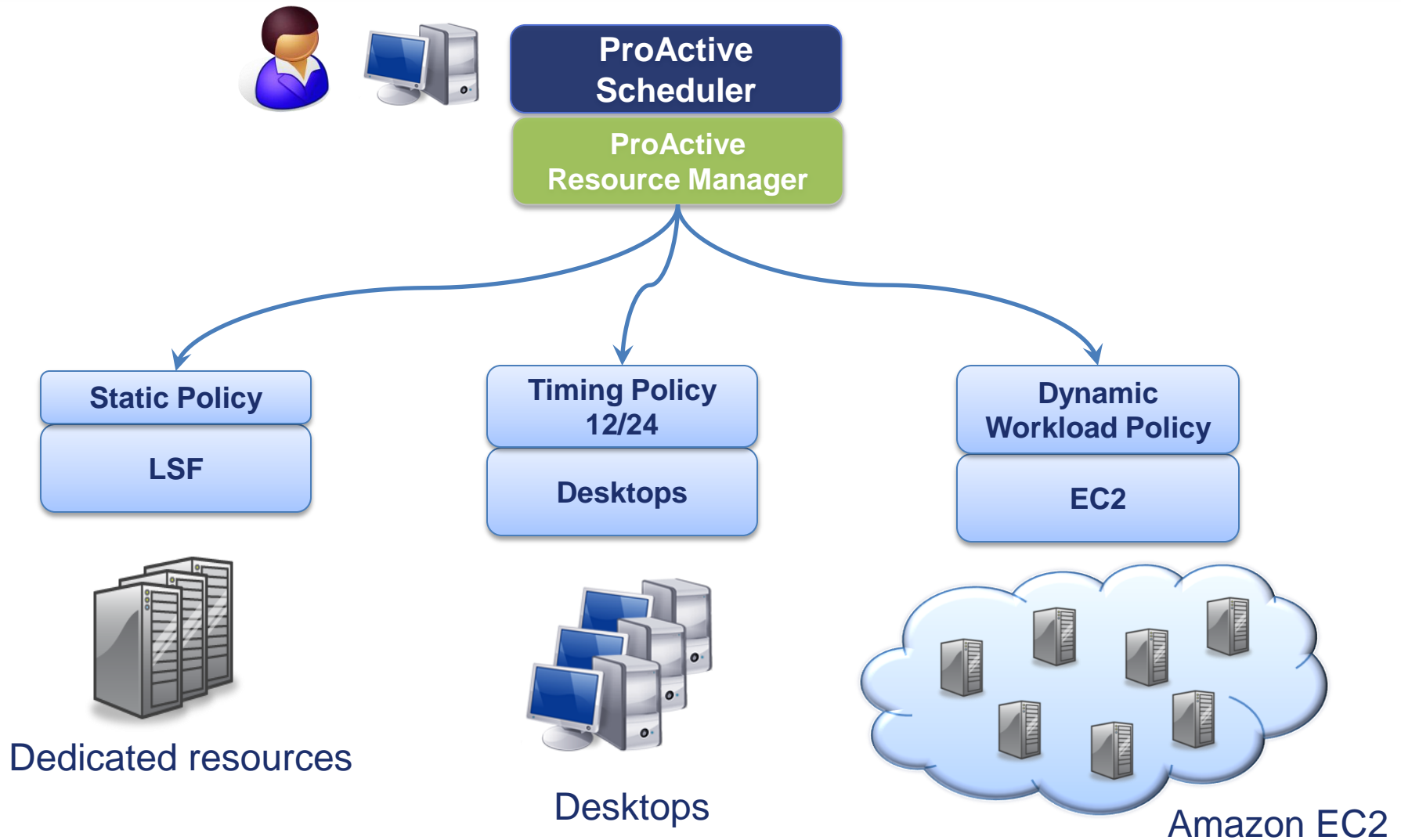
The status bar at the bottom right indicates "connected".

Video 2: Scheduler, Resource Manager



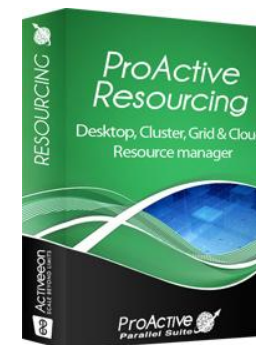
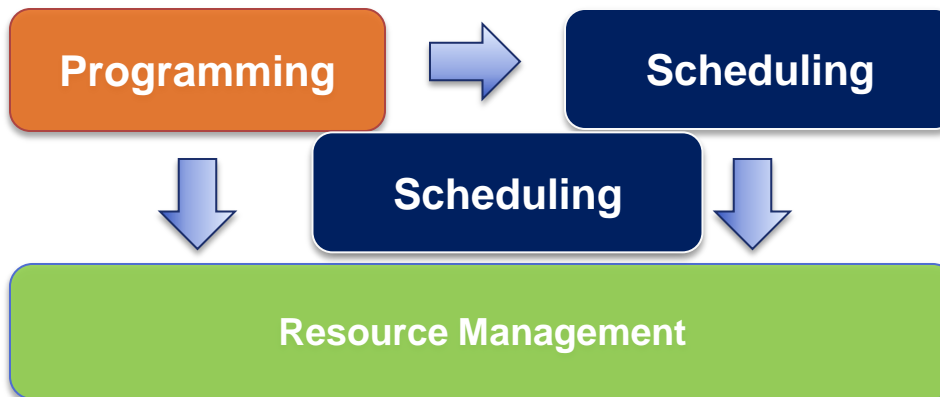
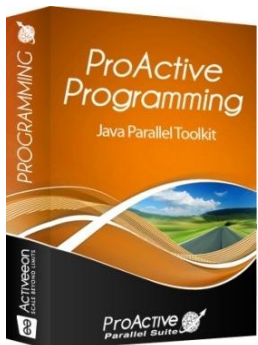
Clusters to Grids to Clouds ***e.g. on Amazon EC2***

Node source Usecase : Configuration for external cloud with EC2

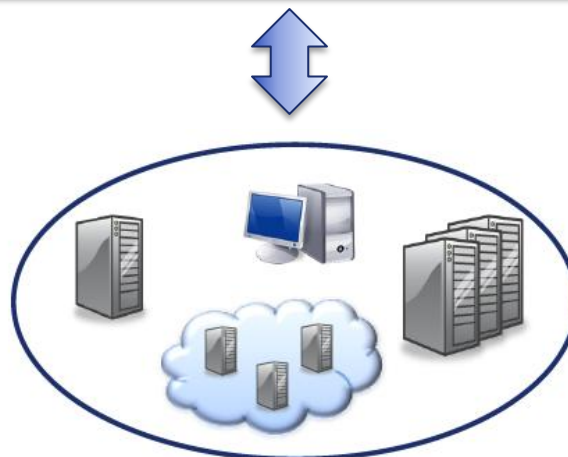


ProActive Parallel Suite

Three fully compatible modules



Clutch Power:
Solid Building Blocks
for Flexible Solutions



3. Use Case: Genomics

SOLiD and ProActive

- ❑ SOLiD Transcriptom Pipeline:
 - Genomic Sequencing Solution
 - Including Multi-language tools, partially ported on Windows
 - Pipelined with Java wrappers

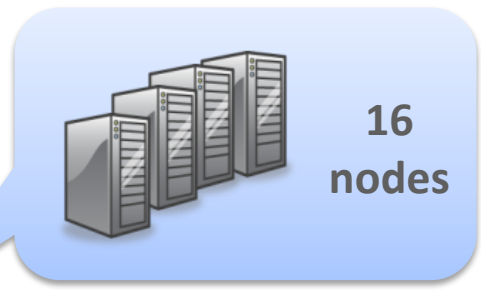
- ❑ SOLiD Platform:
Hardware provided with preconfigured Linux solution
(based on Torque)

- ❑ Up to 20 days Long Computation !
 ➔ *Need for extra computational power to reduce
 computation time*

- ❑ Many Windows Desktops are Available
 ➔ *Need for a dynamic and multi-OS solution*

Resources set up

SOLID
machine from
Applied Biosystems



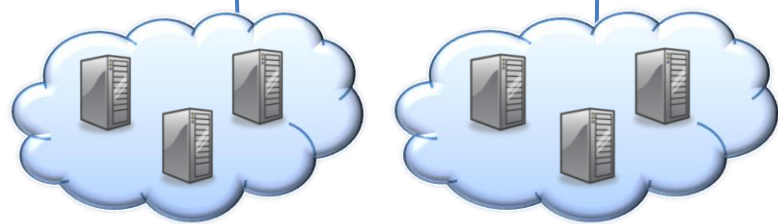
PBS

Cluster



Desktops

Nodes can be dynamically added!

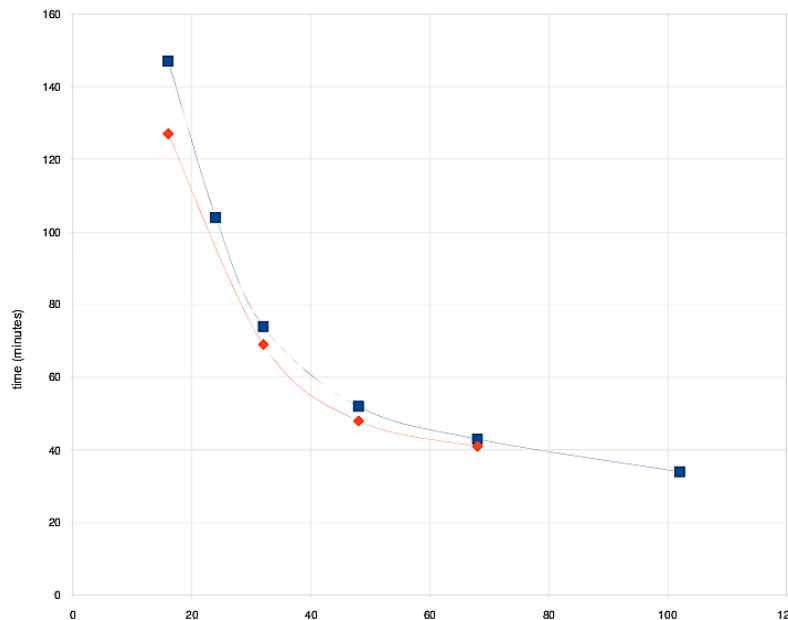


EC2

Clouds

First Benchmarks

- ❑ The distributed version with ProActive of Mapreads has been tested on the INRIA cluster with two settings: the Reads file is split in either 30 or 10 slices
- ❑ Use Case: Matching 31 millions Sequences with the Human Genome (M=2, L=25)



4 Time FASTER from 20 to 100
Speed Up of 80 / Th.
Sequential : 50 h → 35 mn

EC2 only test: nearly the same
performances as the local
SOLiD cluster (+10%)

**For only \$3,2/hour, EC2 has nearly the same perf. as
the local SOLiD cluster (16 cores, for 2H30)**

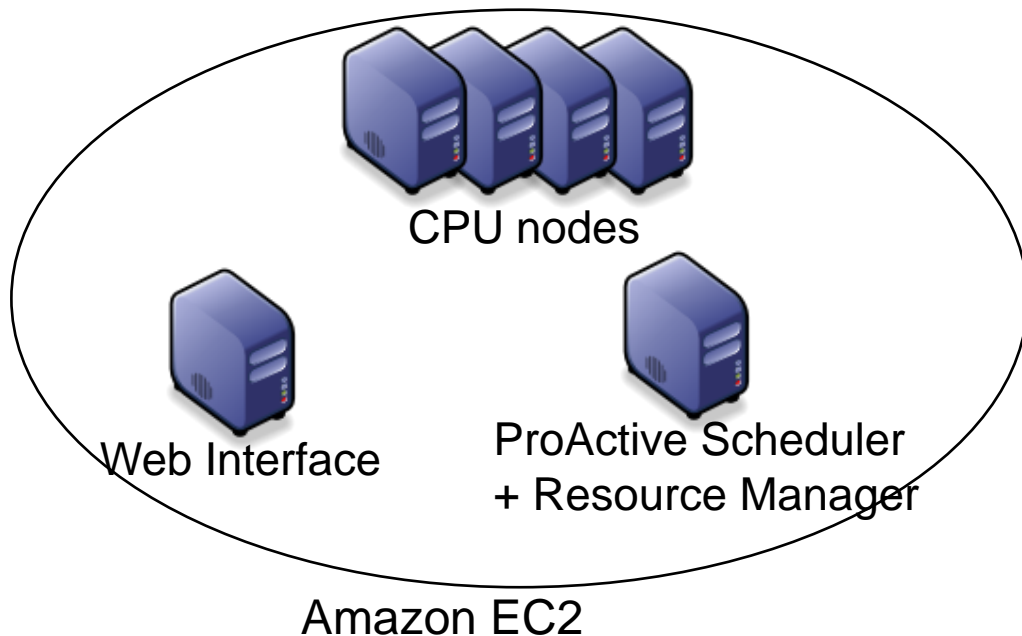
4. *Cloud Seeding*



Cloud Seeding with ProActive

- ❑ Amazon EC2 Execution
- ❑ *Cloud Seeding* strategy to mix heterogeneous computing resources :
 - External GPU resources

Cloud Seeding with ProActive



User

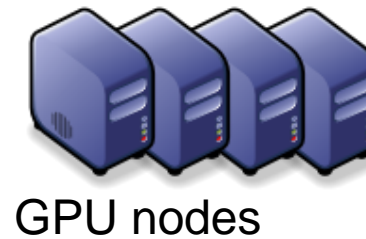
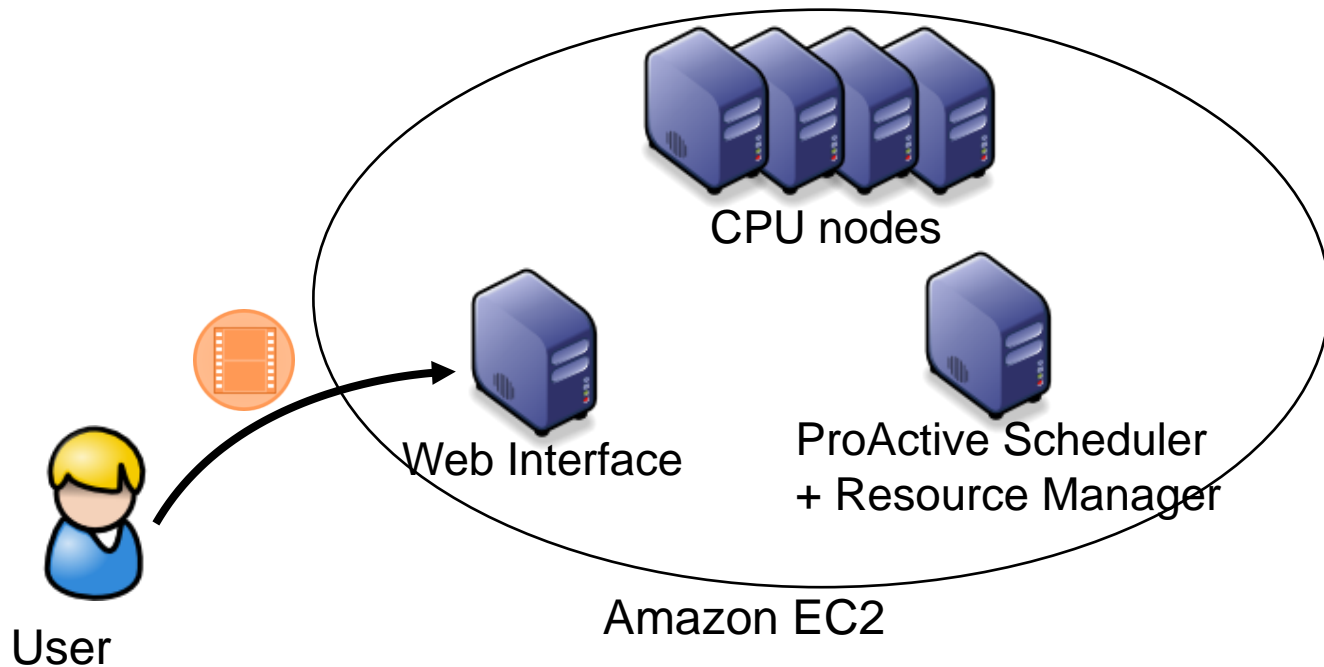


Noised video file



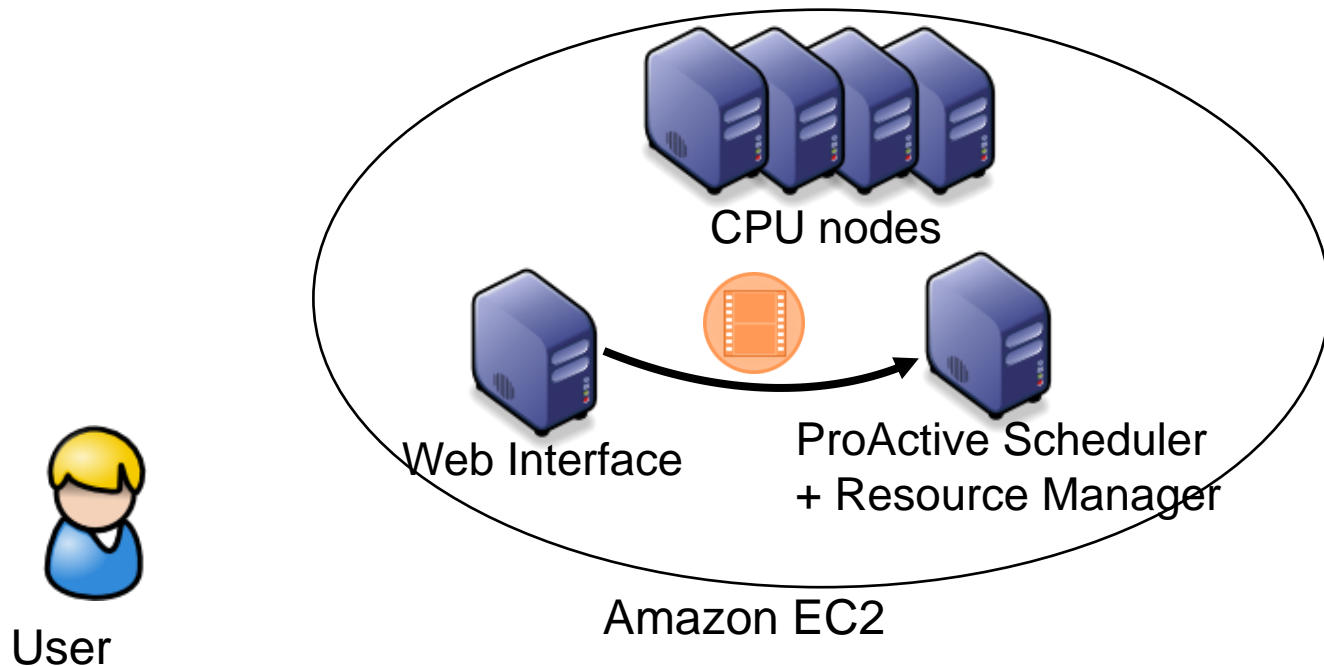
GPU nodes

Cloud Seeding with ProActive

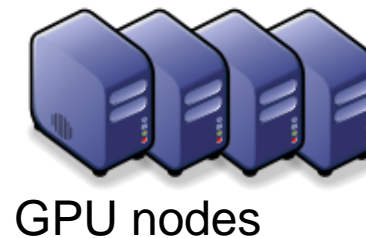


User submit its noised video to the web interface

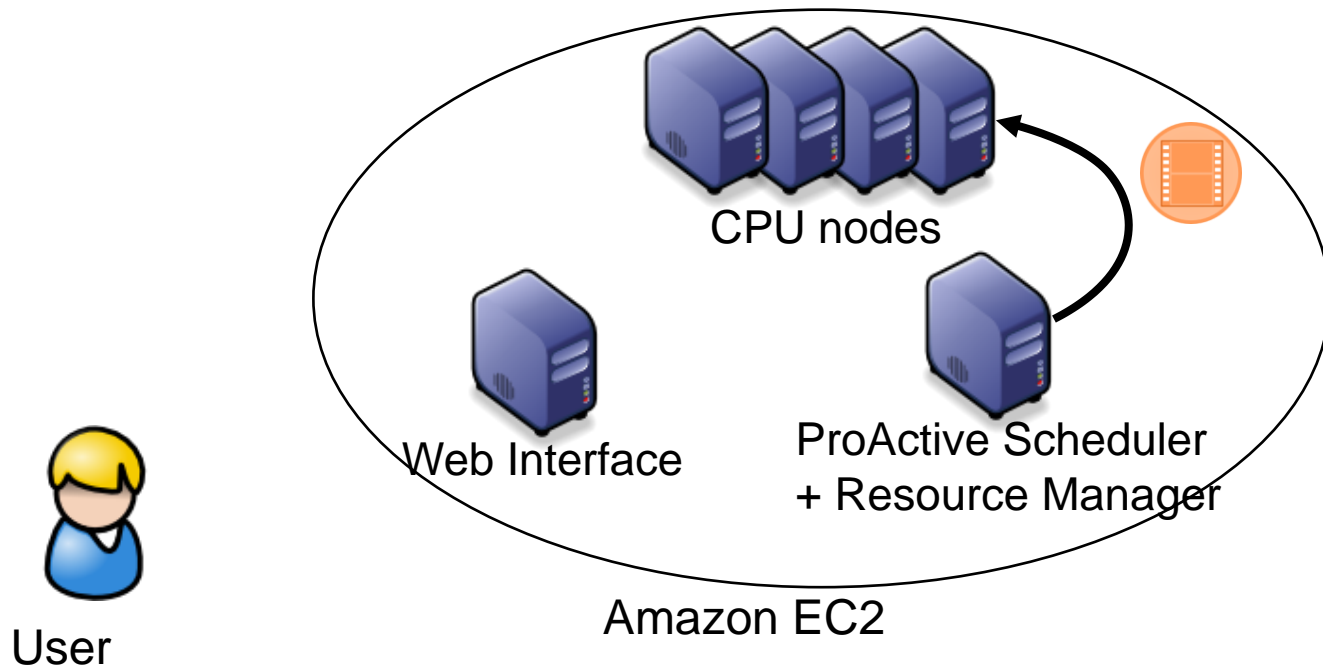
Cloud Seeding with ProActive



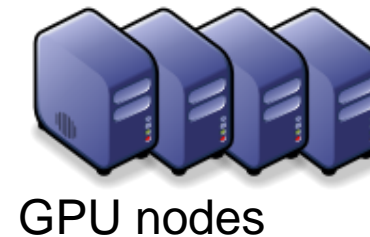
Web Server submit a denoising job the ProActive Scheduler



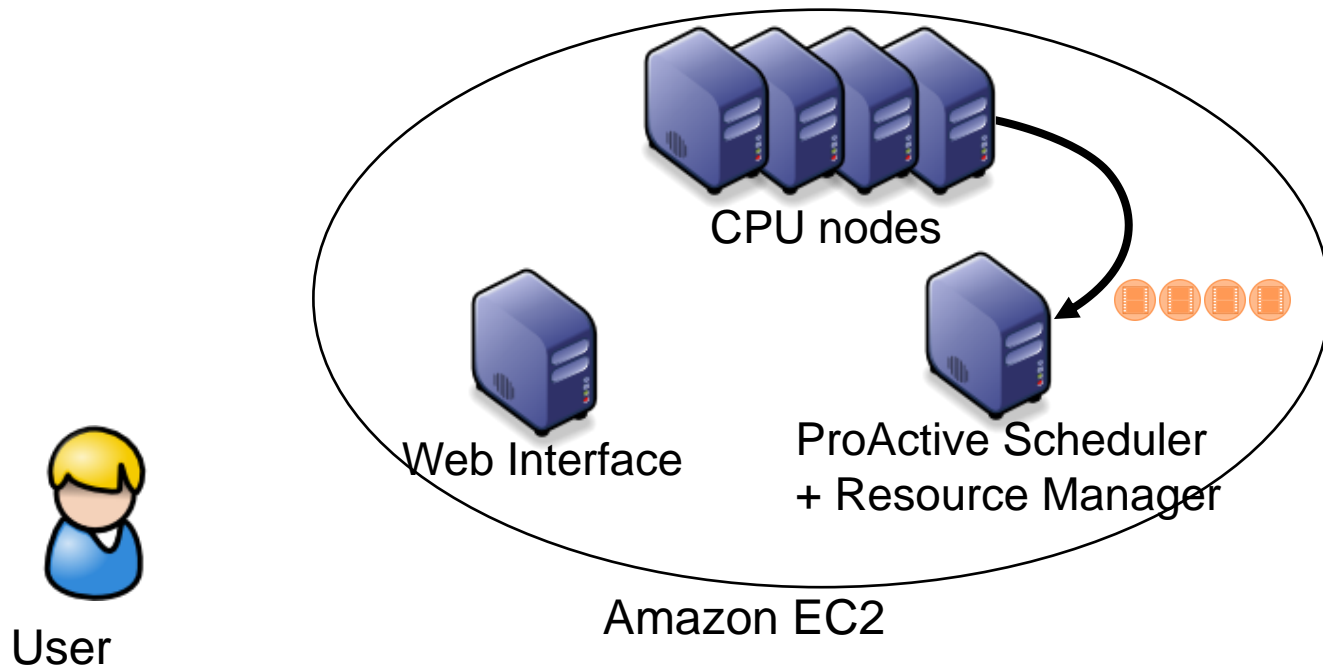
Cloud Seeding with ProActive



CPU nodes are used to split the video into smaller ones

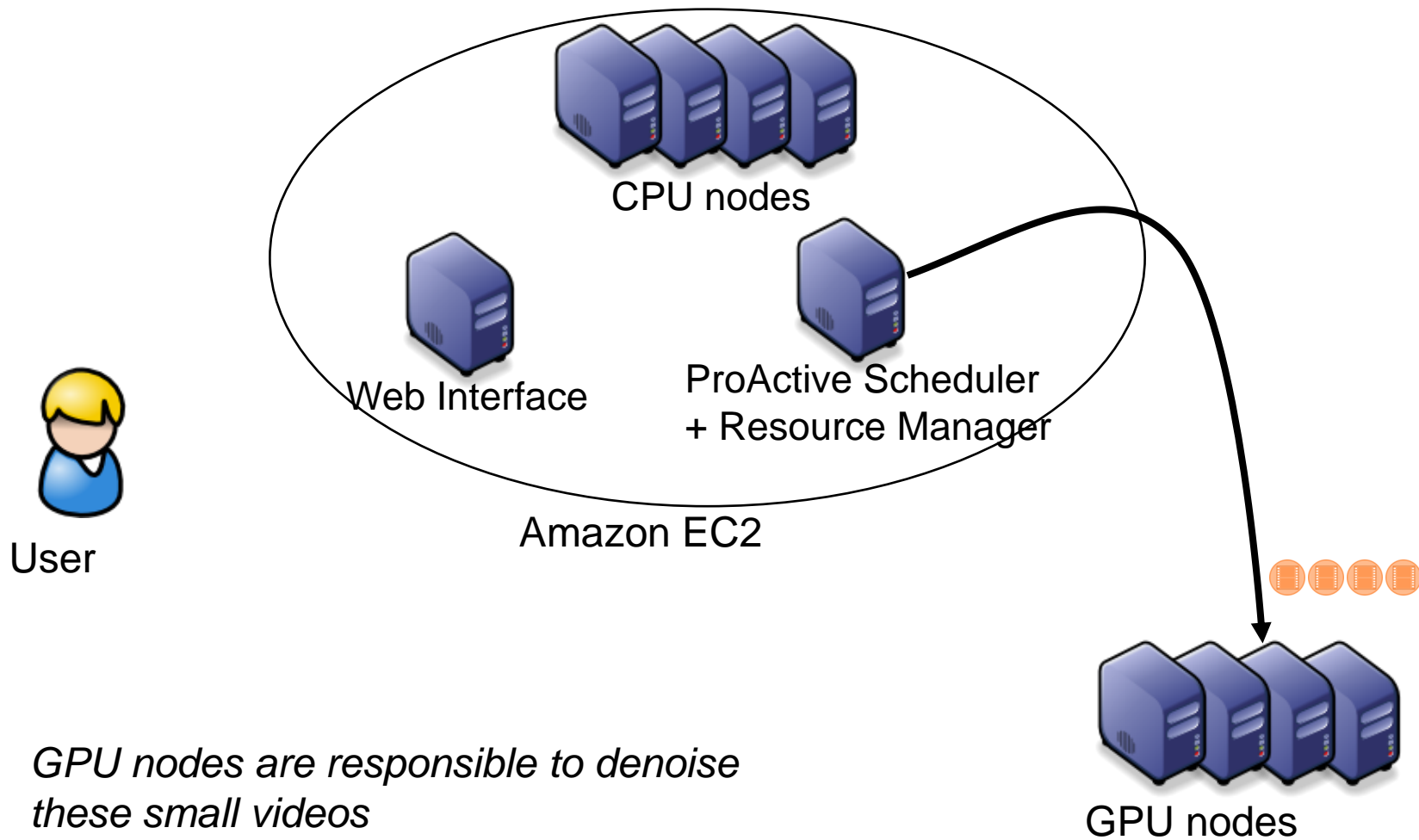


Cloud Seeding with ProActive

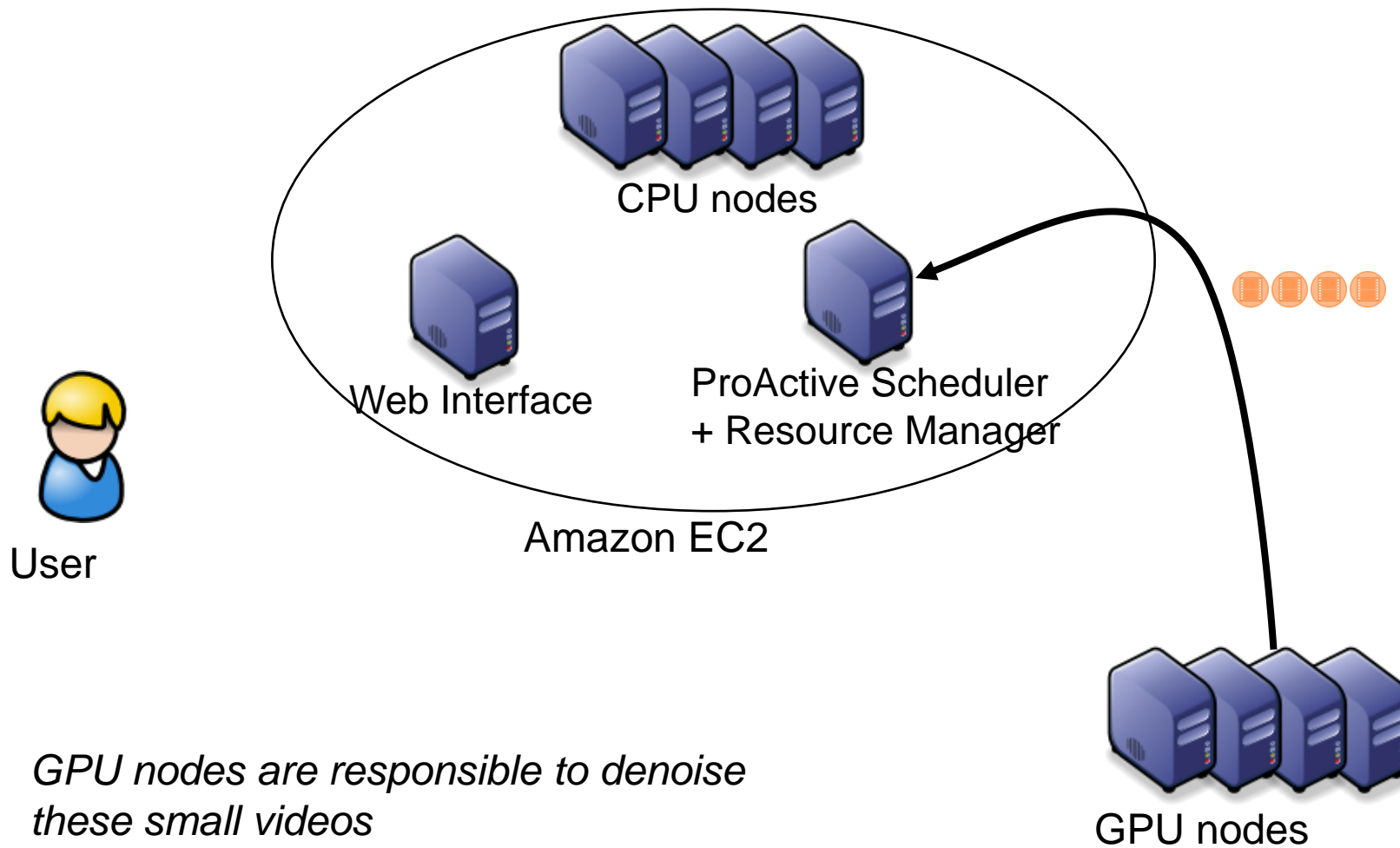


CPU nodes are used to split the video into smaller ones

Cloud Seeding with ProActive

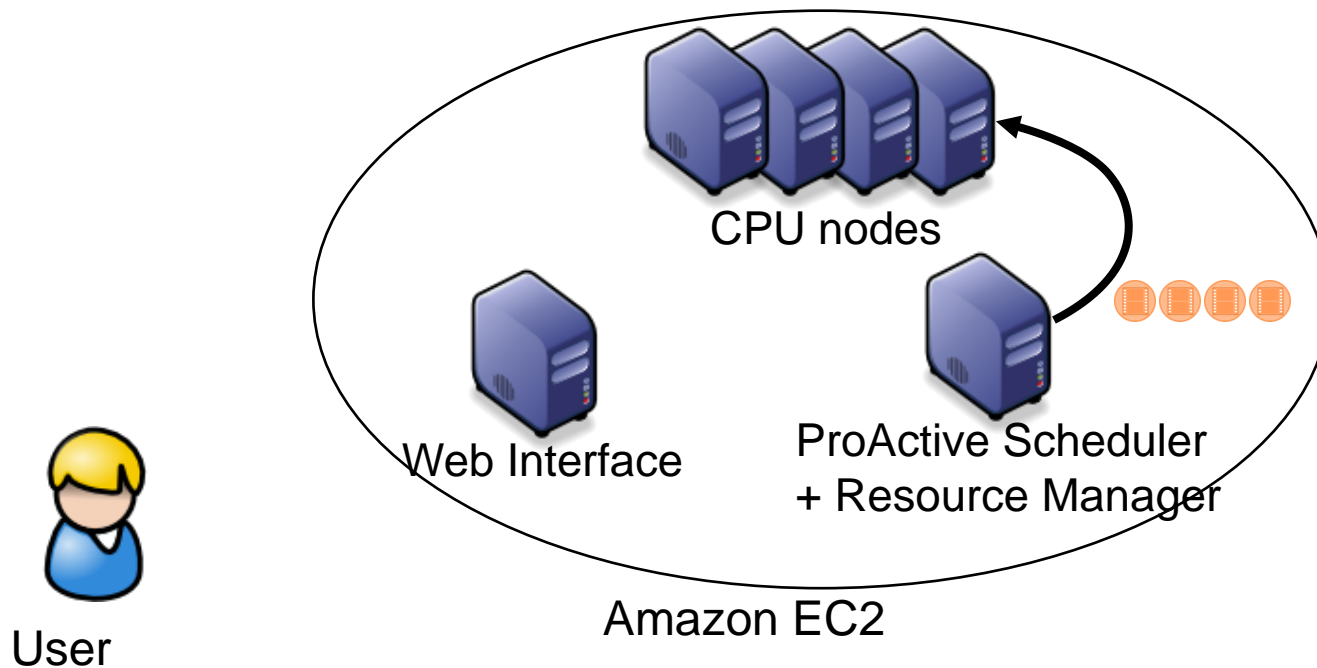


Cloud Seeding with ProActive

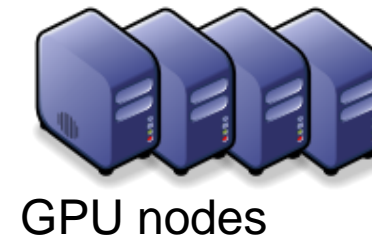


GPU nodes are responsible to denoise these small videos

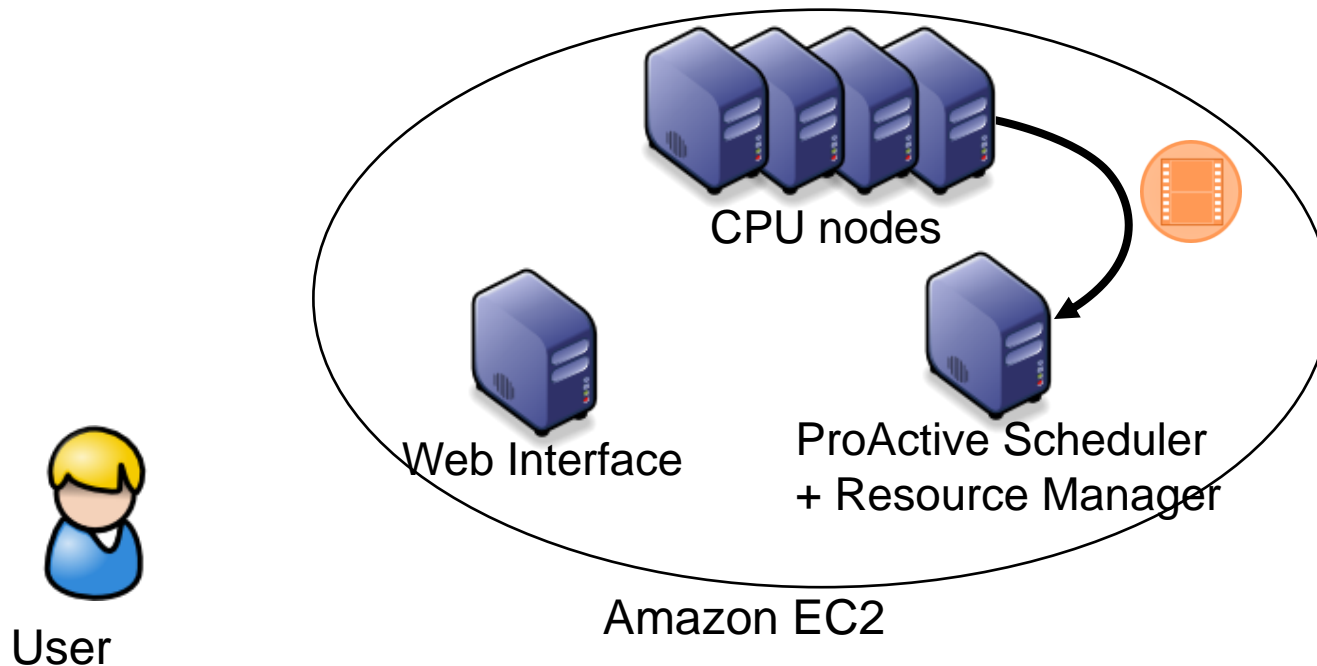
Cloud Seeding with ProActive



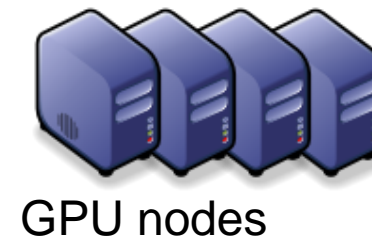
CPU nodes merge the denoised video parts



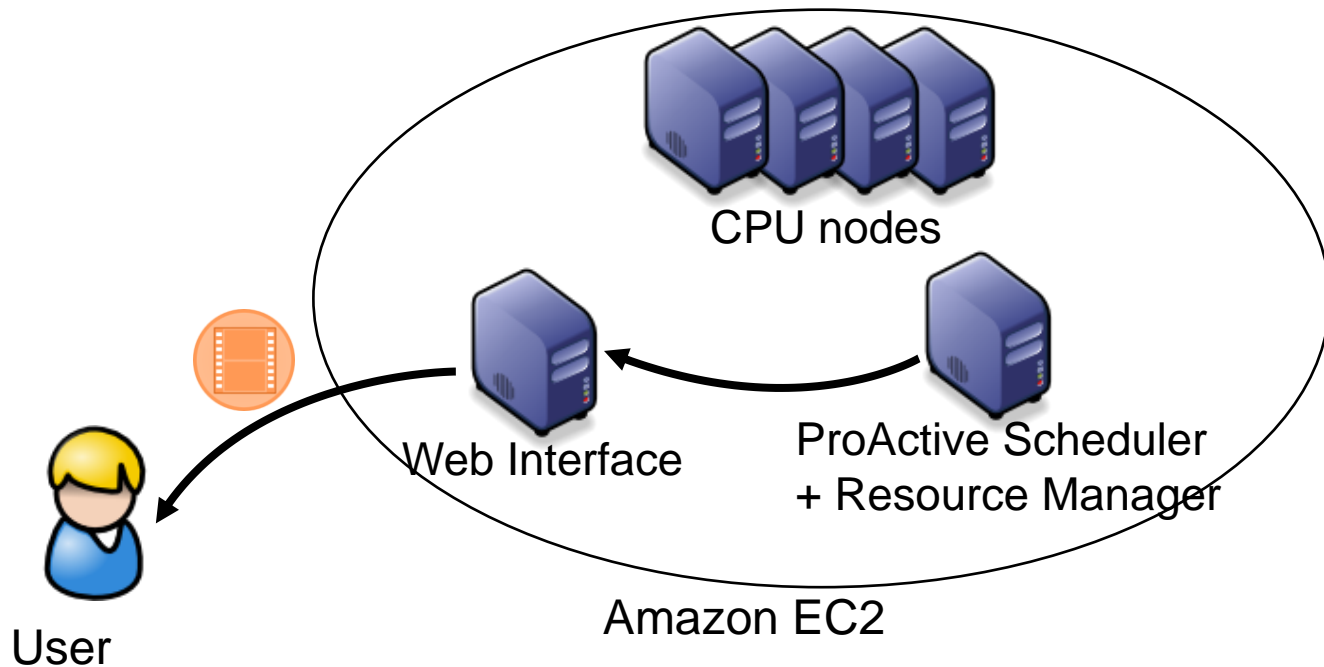
Cloud Seeding with ProActive



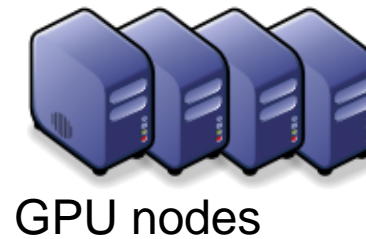
CPU nodes merge the denoised video parts



Cloud Seeding with ProActive



The final denoised video is sent back to the user



Conclusion

Conclusion



Flexibility

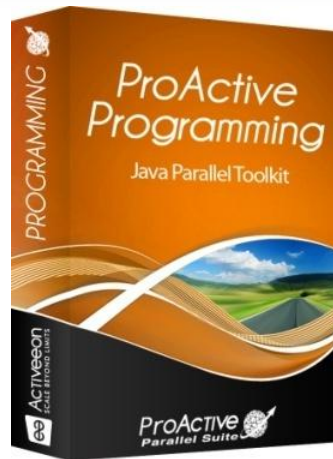
Clutch Power

Portability:

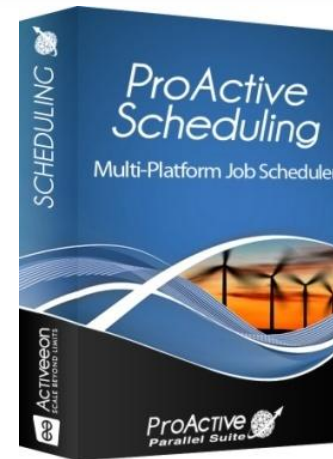
Windows, Linux, Mac

Versatility:

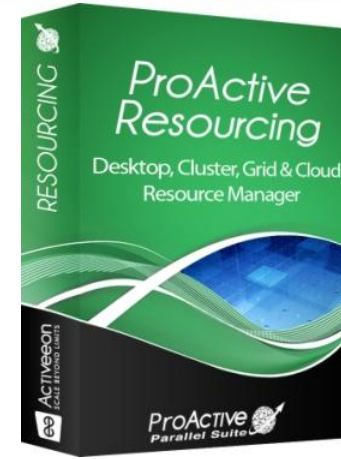
Desktops, Grids, Clouds



Java Parallel Toolkit



Multi-Platform Job Scheduler



Resource Manager

Free Professional
Open Source Software



ProActive.inria.fr

Multi-Core: No sharing Parallel Programming Model

Cloud: Smooth transition needed (Interop)

We removed VO, but we Hype the same dreams!!

Danger: same KO than experienced with Grid

Lets be pragmatic!



AGOS: Grid Architecture for SOA

Building a Platform for Agile SOA with Grid

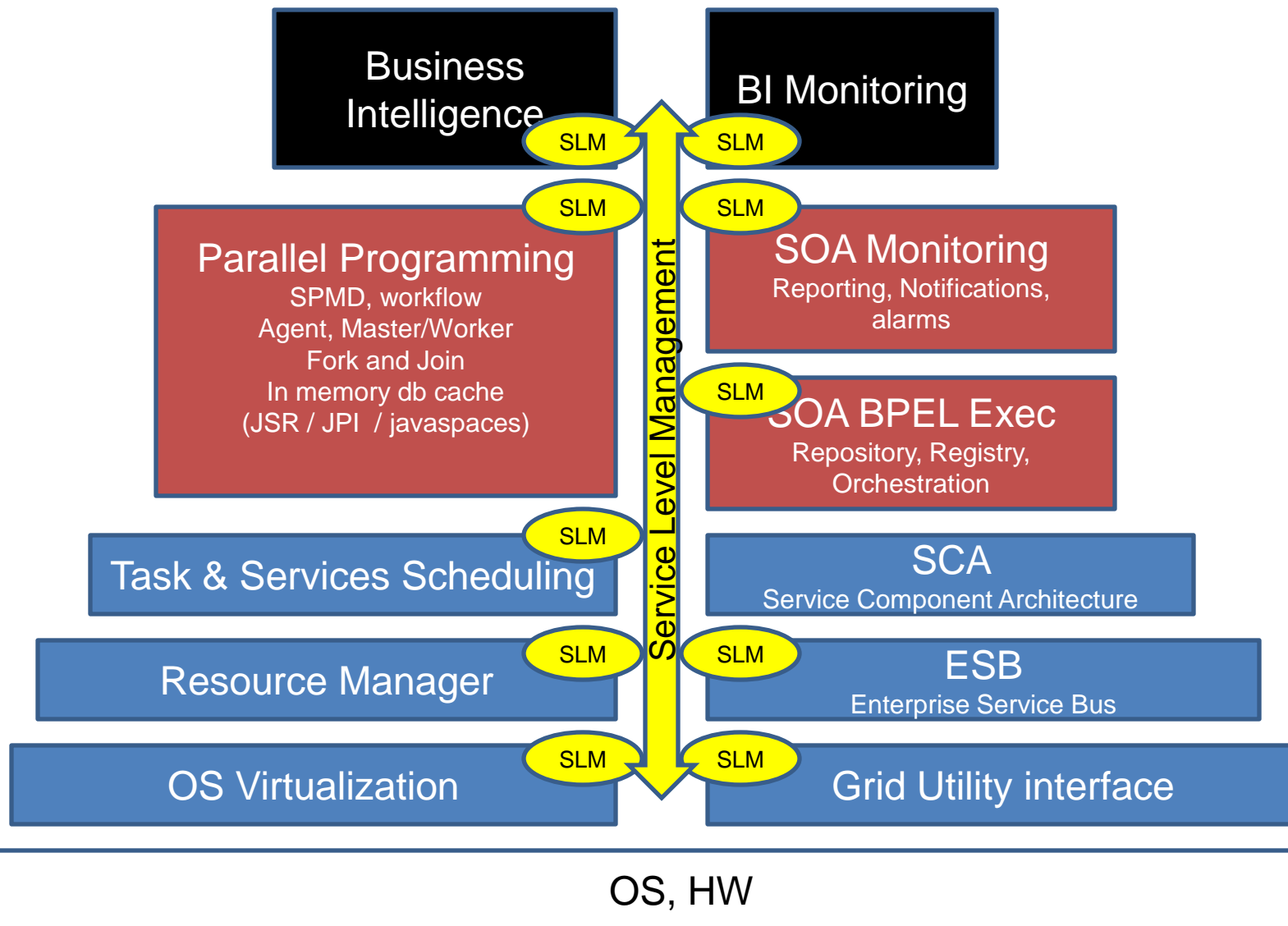
□ AGOS Solutions



In Open Source with Professional Support



AGOS Generic Architecture for Autonomic SOA with GRIDs & Clouds



Key Point: Software Evolution

- ❑ **Distributed To Multicores**

- ❑ **Multi-Cores: 32 (2010) to 64 to 128 to 256 (2014)**

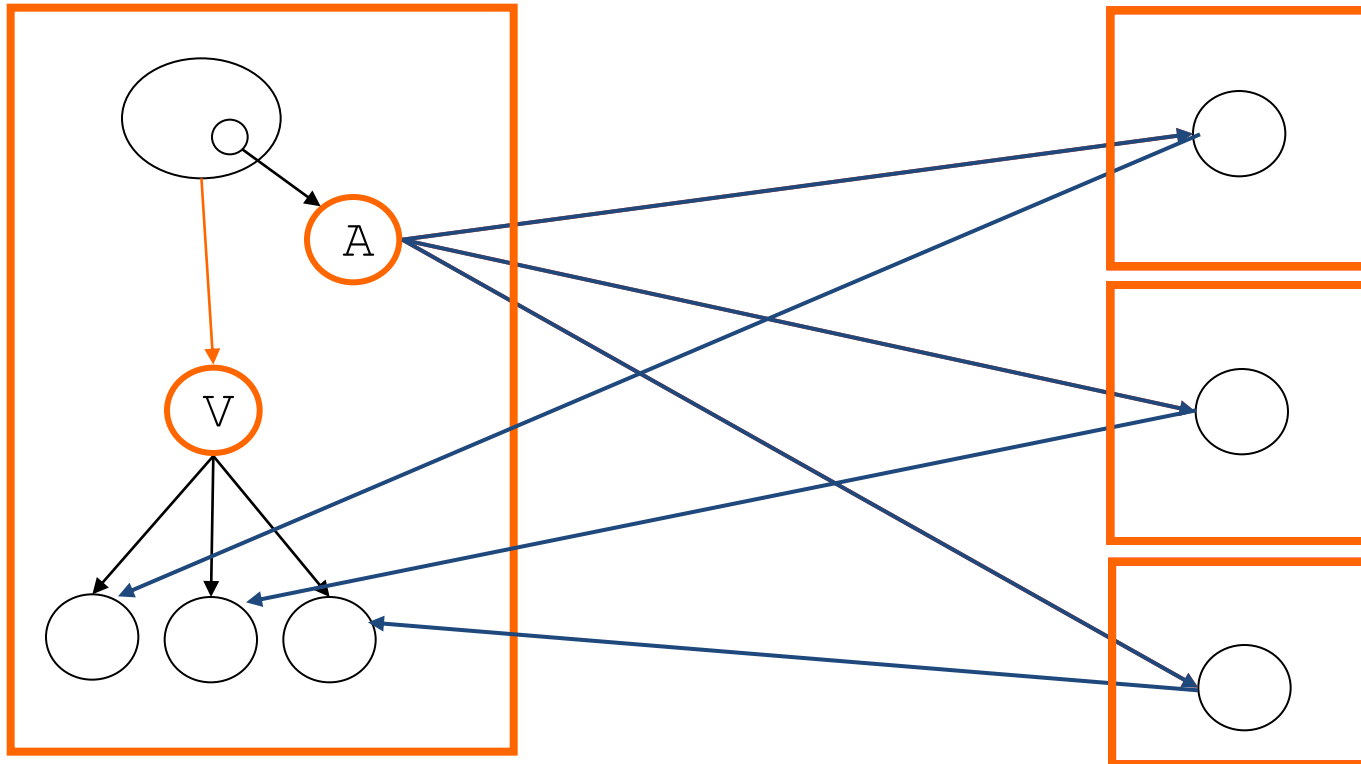
Shift the execution from several multi-cores executing the same application simultaneously to a single, larger multi-core chip.

An application requiring 128 cores to correctly execute, can be executed in 2012 on four 32 cores, and seamlessly executed in 2016 on a single 128-core chips

**→ Smooth evolutivity of applications:
Distributed and Multi-core Platforms**

Creating AO and Groups

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



○ Typed Group ○ Java or Active Object

Group, Type, and Asynchrony
are crucial for Composition

GCM Standardization

Fractal Based Grid Component Model



4 Standards:

1. GCM Interoperability Deployment
2. GCM Application Description
3. GCM Fractal ADL
4. GCM Management API

Key Points about Parallel Components

- ❑ Parallelism is captured at the Module interface
Identical to Typing for functional aspects
- ❑ Composition, parallel word, becomes possible
- ❑ Configuration of the Parallel *aspects*