ProActive

Programming, Composing, Deploying on the Grid

Hierarchical Components for the GRID

Denis Caromel, et al.

ProActive.ObjectWeb.org

OASIS Team

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

Santiago, Nov. 9 2004

0. GRIDs
1. *ProActive*: Asynchronous Distributed Objects
2. Groups
3. Components

Université Nice Sophia Antipolis — CNRS Centre National de la Recherche Scientifique — INRIA — ObjectWeb Open Source Middleware

ProActive:
A Java API + Tools for Parallel, Distributed Computing

- A uniform framework: **An Active Object pattern**
- A formal model behind: **Prop. Determinism, insensitivity to deploy.**

Programming Model:

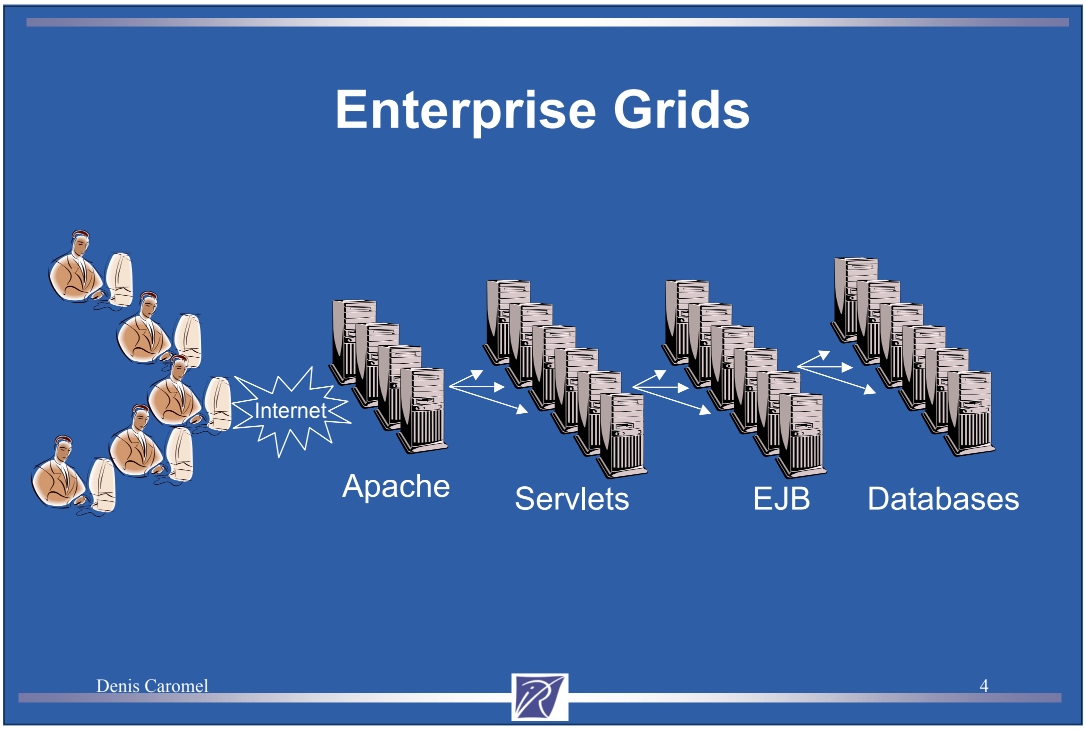
- **Remote Objects (Classes, not only Interfaces, Dynamic)**
- **Asynchronous Communications, Futures, Wait-By-Necessity**
- **Groups, Mobility, Components, Security, Fault Tolerance: Checkpoints**

Environment:

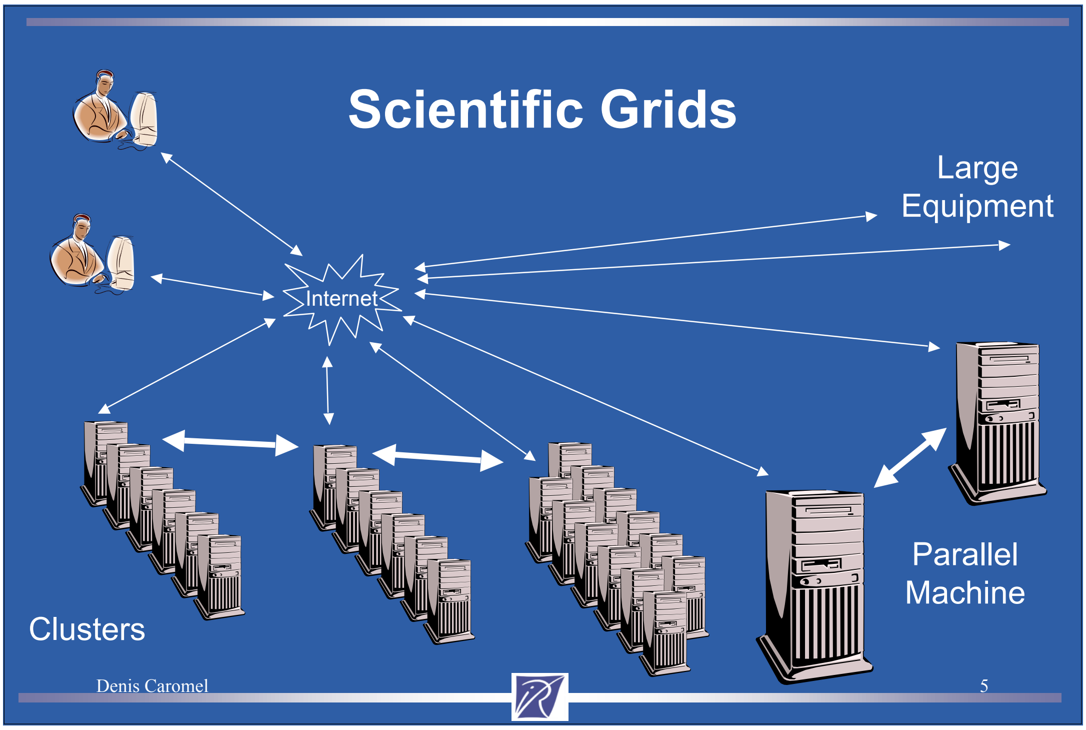
- **XML Deployment Descriptors,** `Web Service Export., HTTP, ssh Tunneling`
- **Various protocols:** `rsh,ssh,LSF,PBS,`**`Globus`**`,Sun Grid Engine, sshGSI`
- **Visualization and monitoring: IC2D**

In the www. ObjectWeb .org Consortium (Open Source middleware) since April 2002 (**LGPL license**)

GRIDs



Enterprise Grids



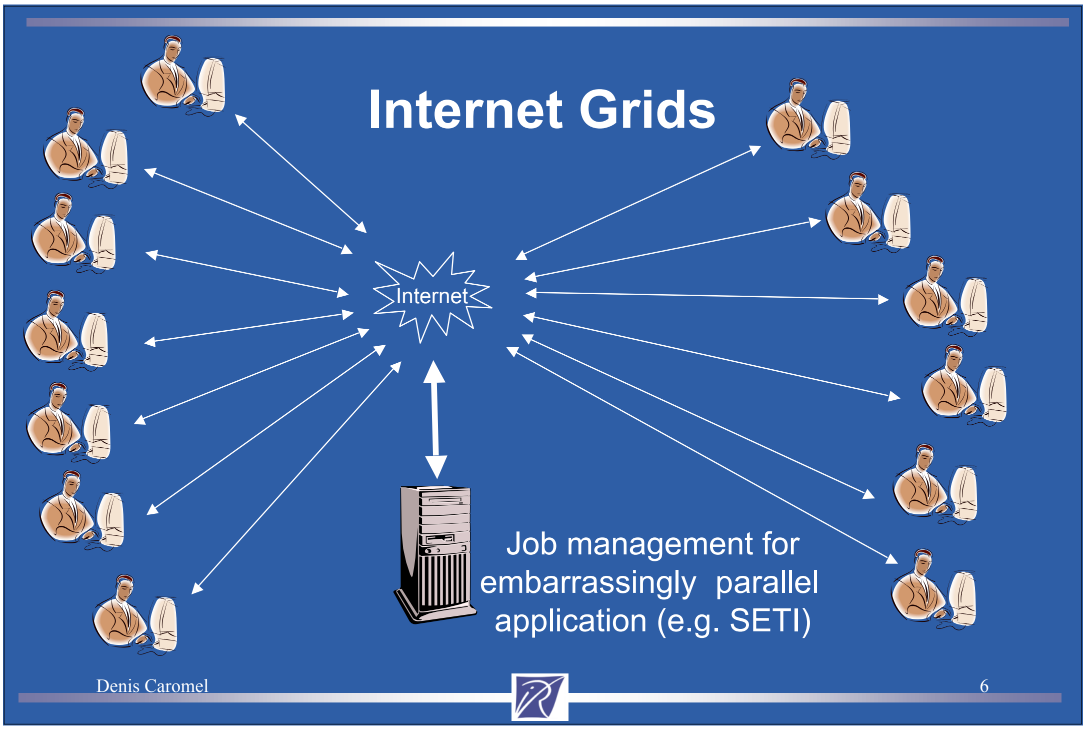
Scientific Grids

Internet

Large Equipment

Parallel Machine

Clusters

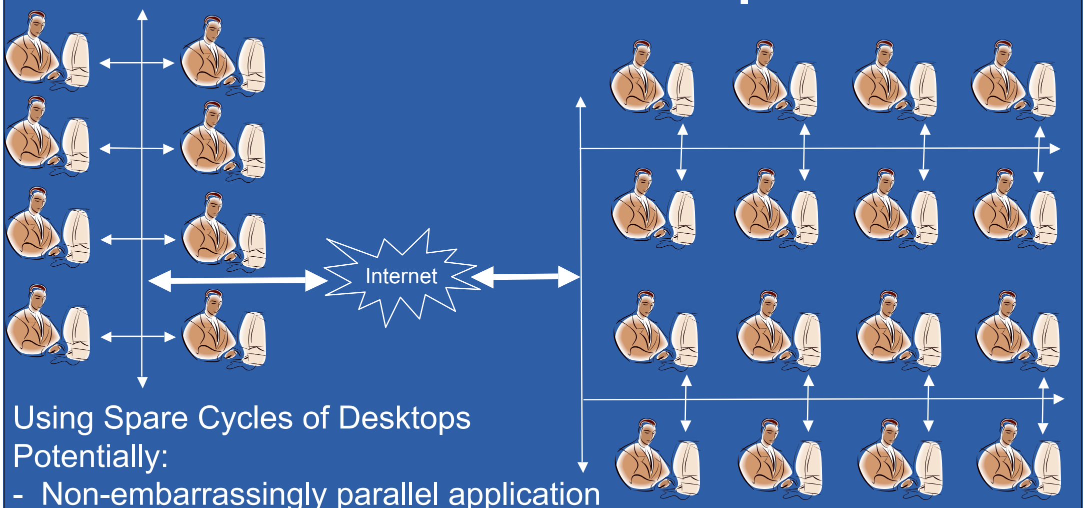


Internet Grids

Internet

Job management for embarrassingly parallel application (e.g. SETI)

Intranet Grids - Desktop Grids



Using Spare Cycles of Desktops
Potentially:
- Non-embarrassingly parallel application
- Across several sites

The multiple GRIDs

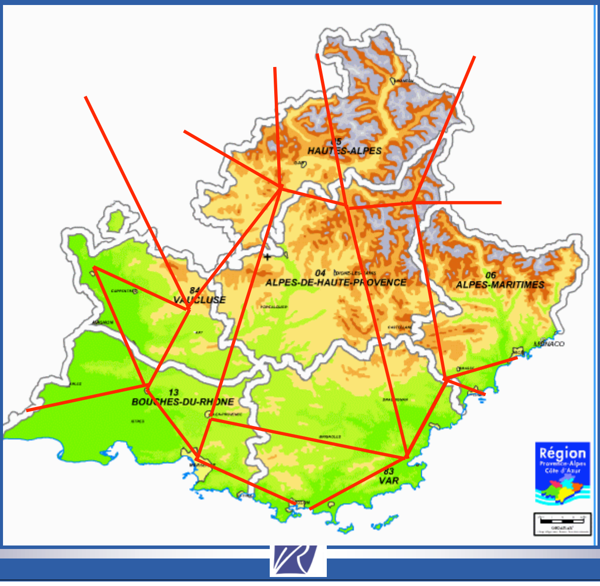
- Scientific Grids
- Enterprise Grids
- Internet Grids
- Intranet Desktop Grids

Strong convergence in process!

At least at the infrastructure level, i.e. WS

Grid: from enterprise ... to regional

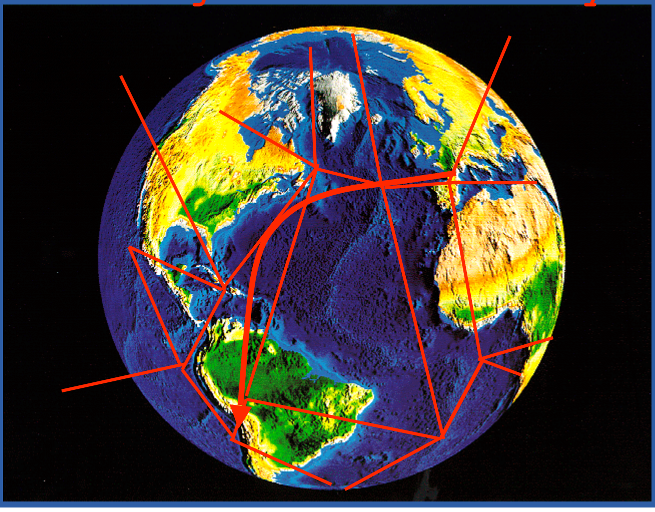
Very hard deployment problems … right from the beginning



Grid: from regional ... to worldwide

Communication Nice - Santiago: 70 ms Light Speed

Challenge: Hide the latency !



Define adequate programming model

ProActive

Programming, Composing, Deploying on the Grid

Programming

Wrapping

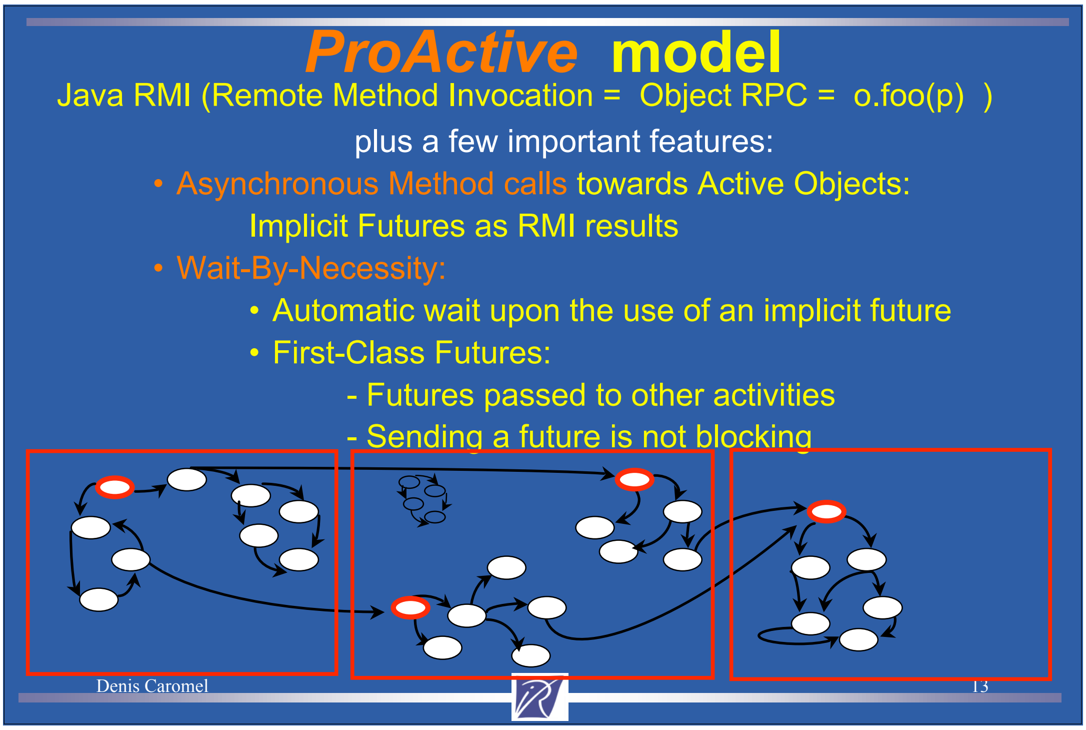
Composing

Deploying

Figures: Web Page Hits: ~ 3 000 / month,
Downloads: 150-300 / month, Users: ?? .us, .mx, .br, .cl, .ch, .it, .fr, ...

Distributed Objects

ProActive Programming

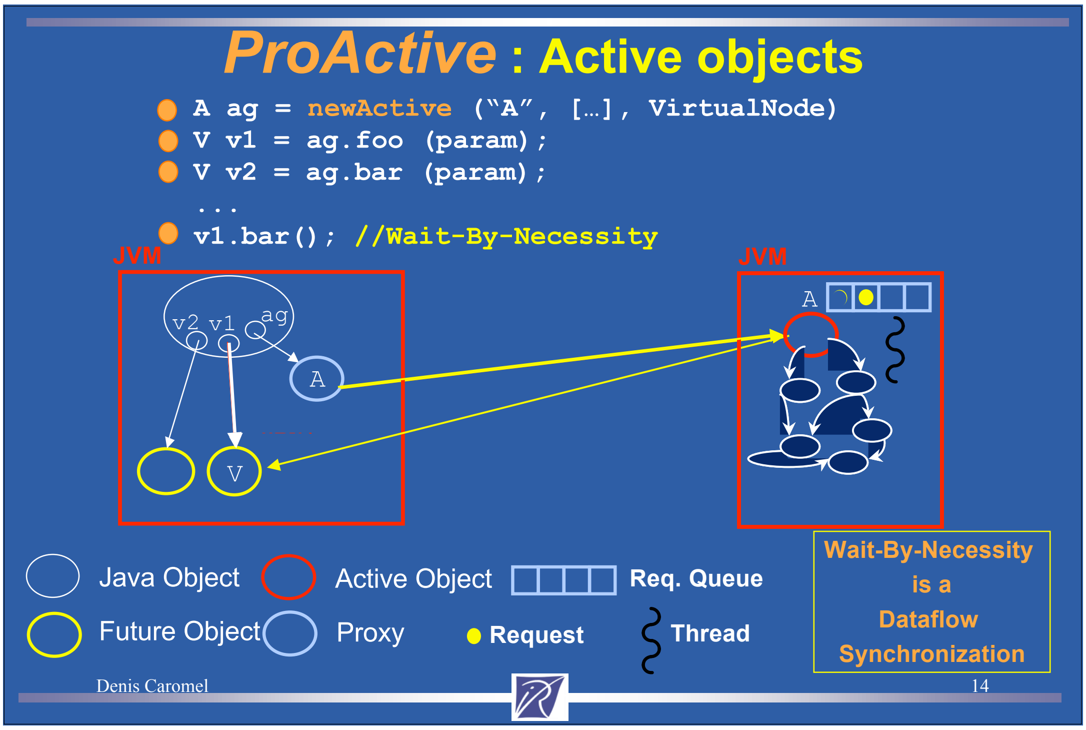


ProActive model

Java RMI (Remote Method Invocation = Object RPC = o.foo(p))

plus a few important features:

- Asynchronous Method calls towards Active Objects:
  Implicit Futures as RMI results
- Wait-By-Necessity:
  - Automatic wait upon the use of an implicit future
  - First-Class Futures:
    - Futures passed to other activities
    - Sending a future is not blocking



ProActive : Active objects

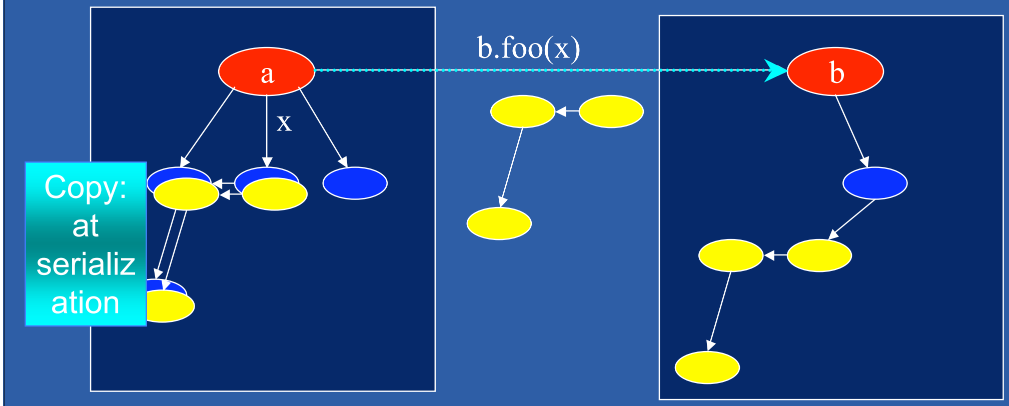
```
A ag = newActive ("A", […], VirtualNode)
V v1 = ag.foo (param);
V v2 = ag.bar (param);
...
v1.bar(); //Wait-By-Necessity
```

○ Java Object ○ Active Object □□□□ Req. Queue

○ Future Object ○ Proxy ● Request ∫ Thread

Wait-By-Necessity is a Dataflow Synchronization

Call between Objects: Parameter passing: Copy of Java Objects

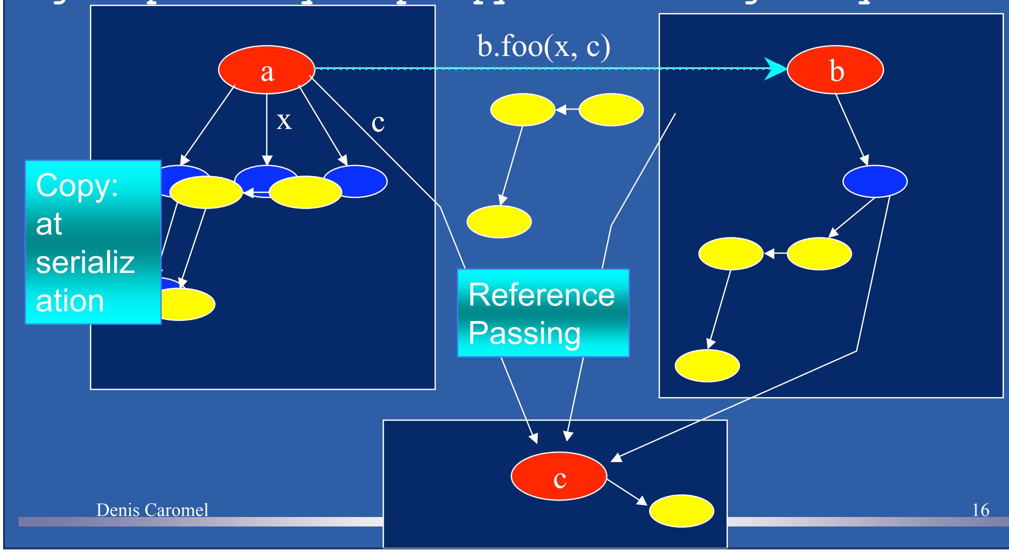


Copy: at serializ ation

(Deep) Copies evolve independently -- No consistency

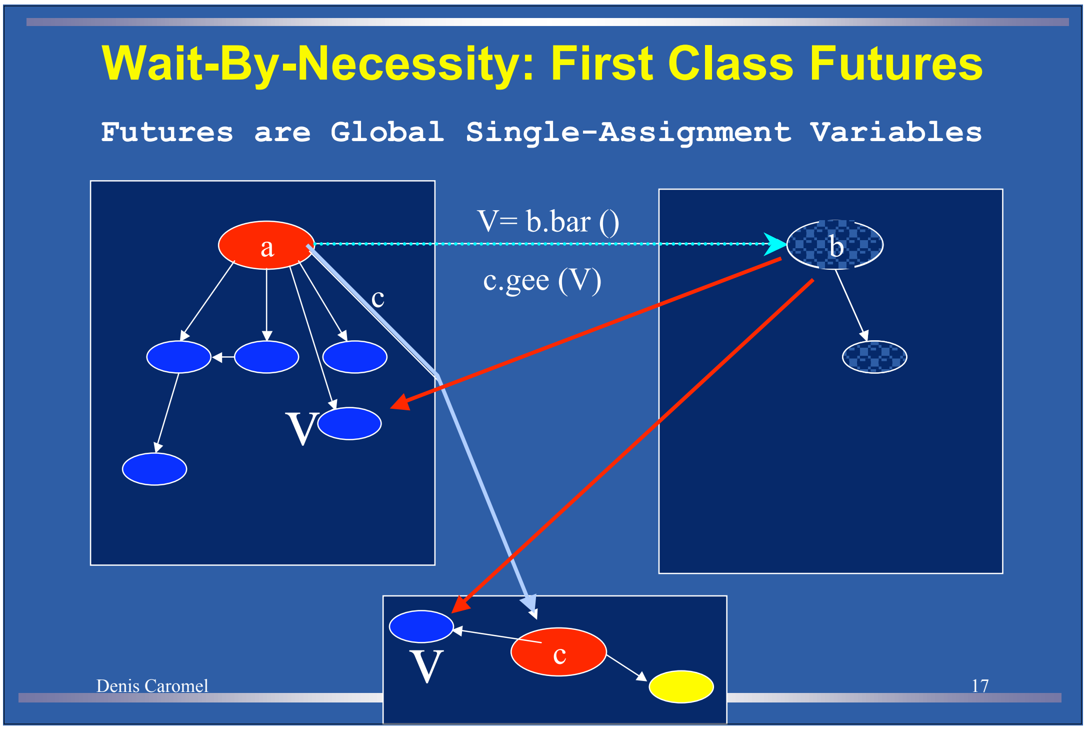
Call between Objects: Parameter Passing: Active Objects

Object passed by Deep Copy - Active Object by Reference



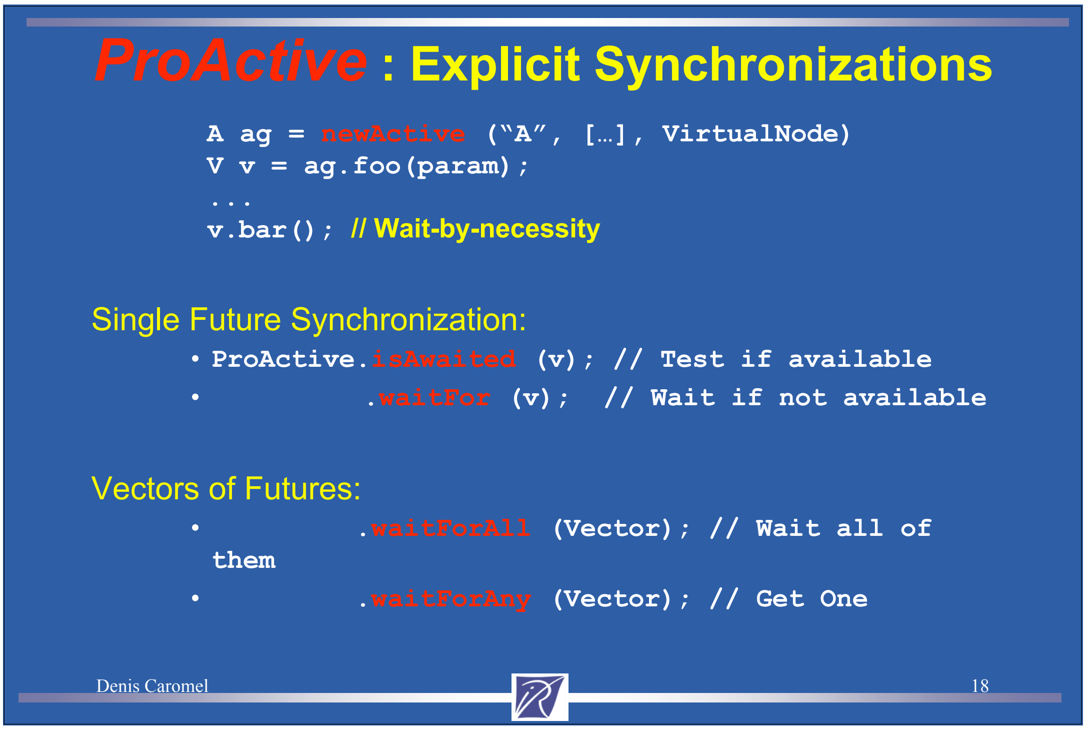
Copy: at serializ ation

Reference Passing



Wait-By-Necessity: First Class Futures

Futures are Global Single-Assignment Variables



ProActive : Explicit Synchronizations

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

Single Future Synchronization:

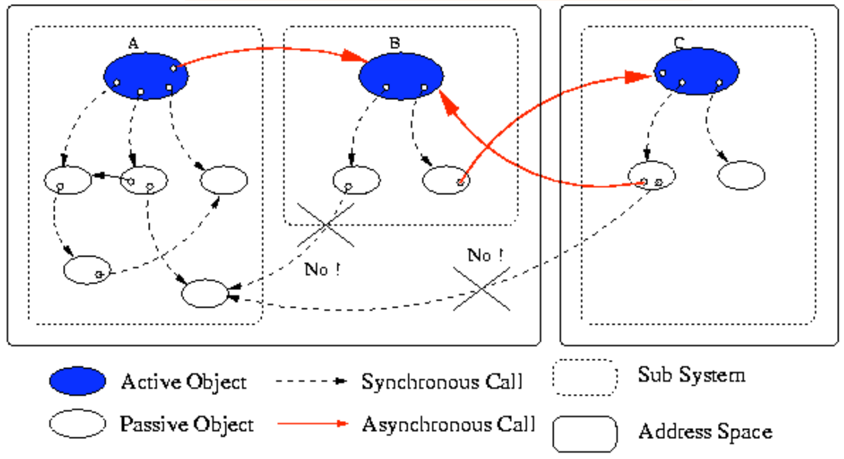
- `ProActive.isAwaited (v); // Test if available`
- `          .waitFor (v);  // Wait if not available`

Vectors of Futures:

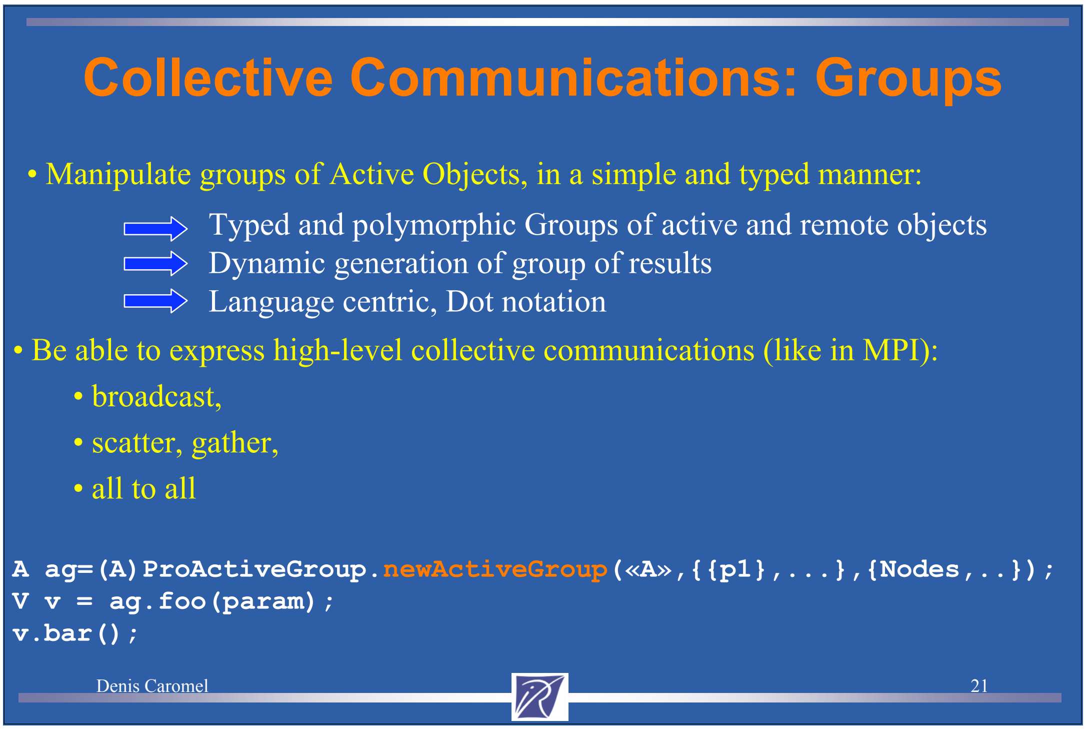
- `          .waitForAll (Vector); // Wait all of them`
- `          .waitForAny (Vector); // Get One`

Standard system at Runtime: Asynchrony, WbN, ... but no sharing

Proofs of Determinism



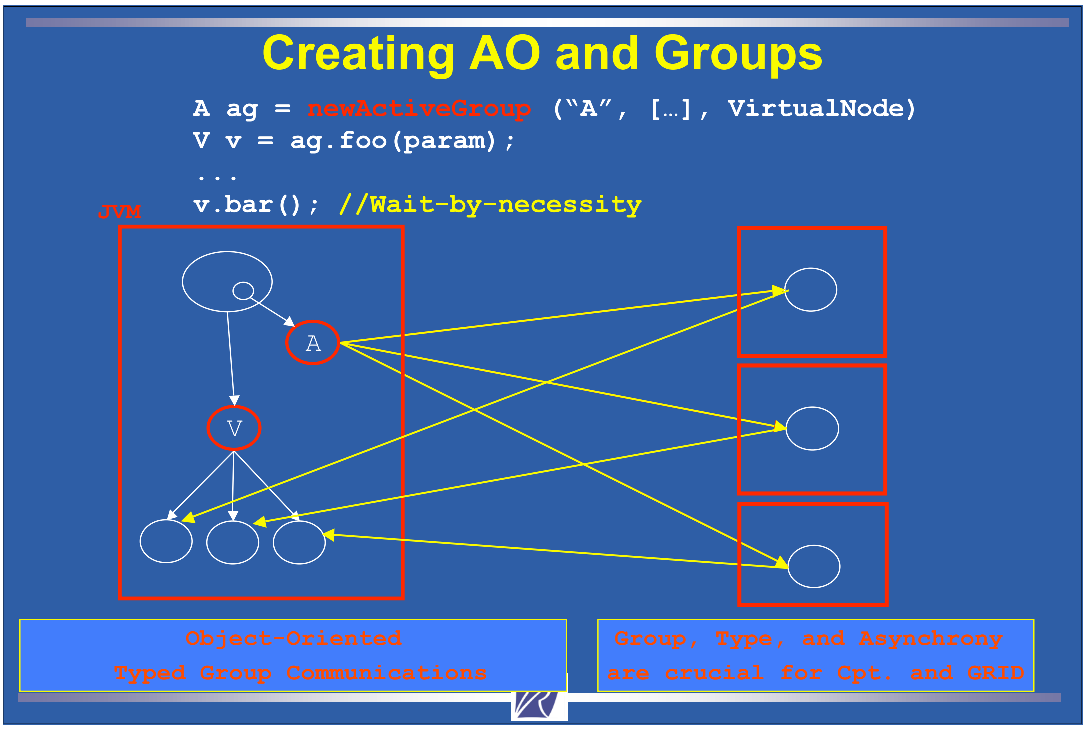
Groups



Collective Communications: Groups

- Manipulate groups of Active Objects, in a simple and typed manner:
  - ⇨ Typed and polymorphic Groups of active and remote objects
  - ⇨ Dynamic generation of group of results
  - ⇨ Language centric, Dot notation
- Be able to express high-level collective communications (like in MPI):
  - broadcast,
  - scatter, gather,
  - all to all

```
A ag=(A)ProActiveGroup.newActiveGroup(«A»,{{p1},...},{Nodes,..});
V v = ag.foo(param);
v.bar();
```



Creating AO and Groups

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

Object-Oriented
Typed Group Communications

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

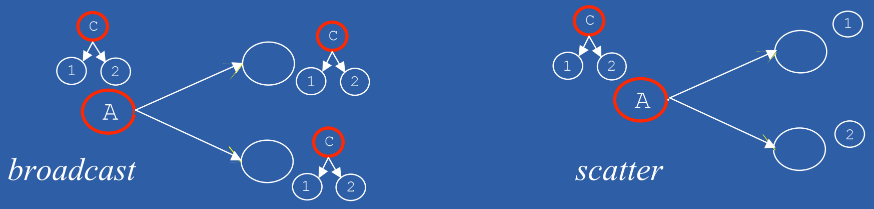
Broadcast or Scatter

Broadcast is the default behavior

Scatter is also possible

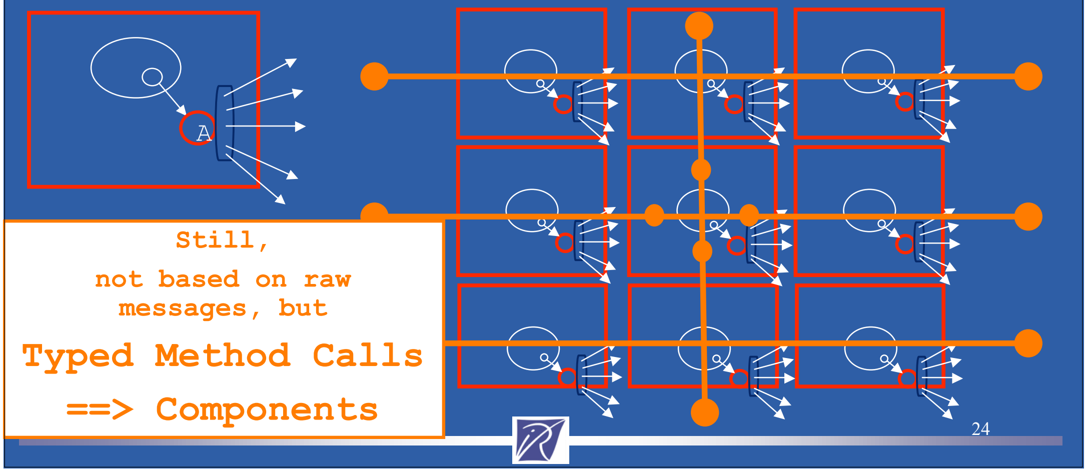
- use a group as parameter
- Scattered depends on rankings

```
  gA.bar(gC);    // broadcast gC
  ProActive.setScatterGroup(gC);
➔ ga.bar(gC);    // scatter gC
```



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

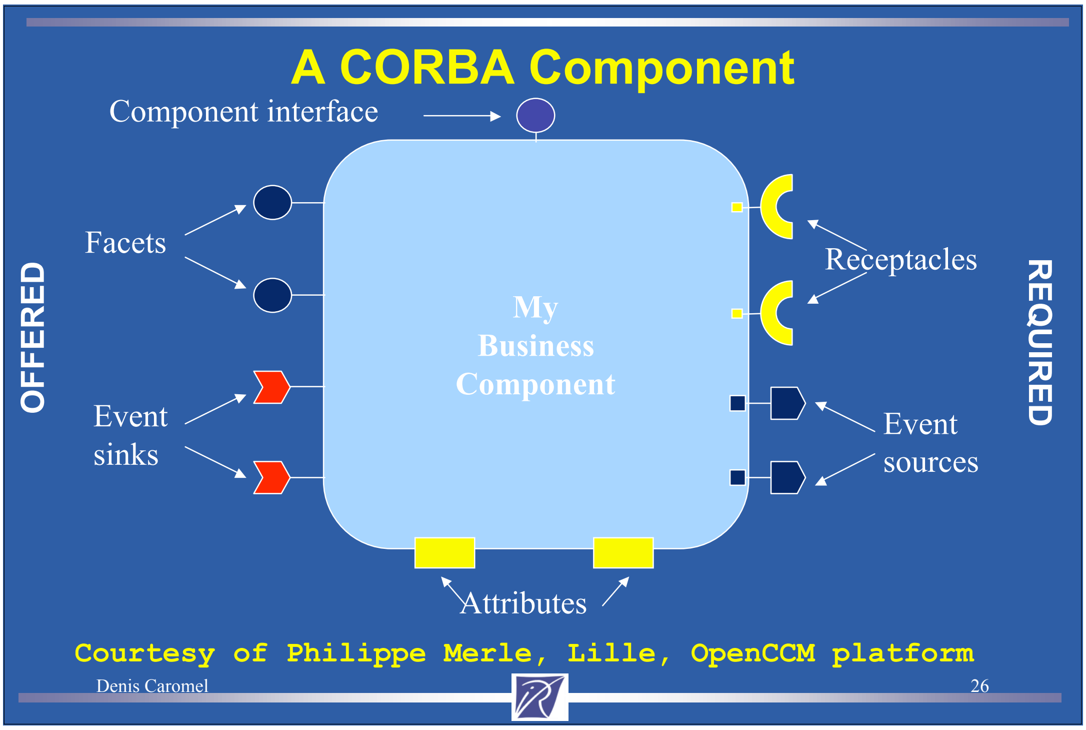
==> Components

Parallel, Distributed, Hierarchical

Components

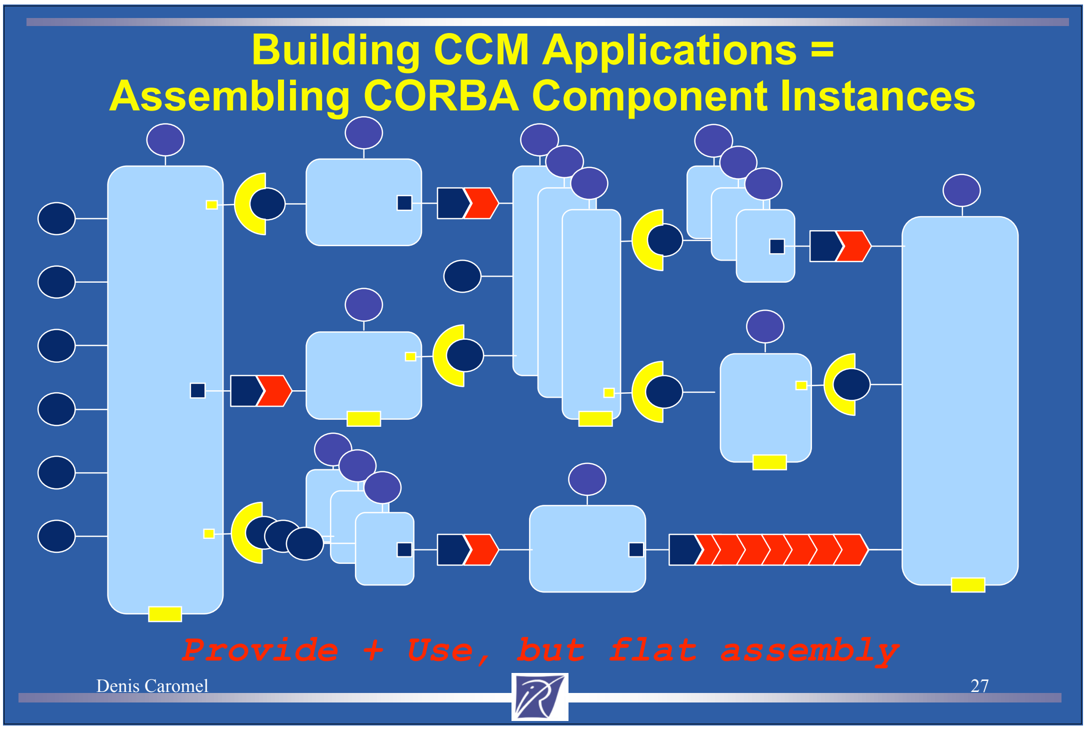
for the Grid

Composing



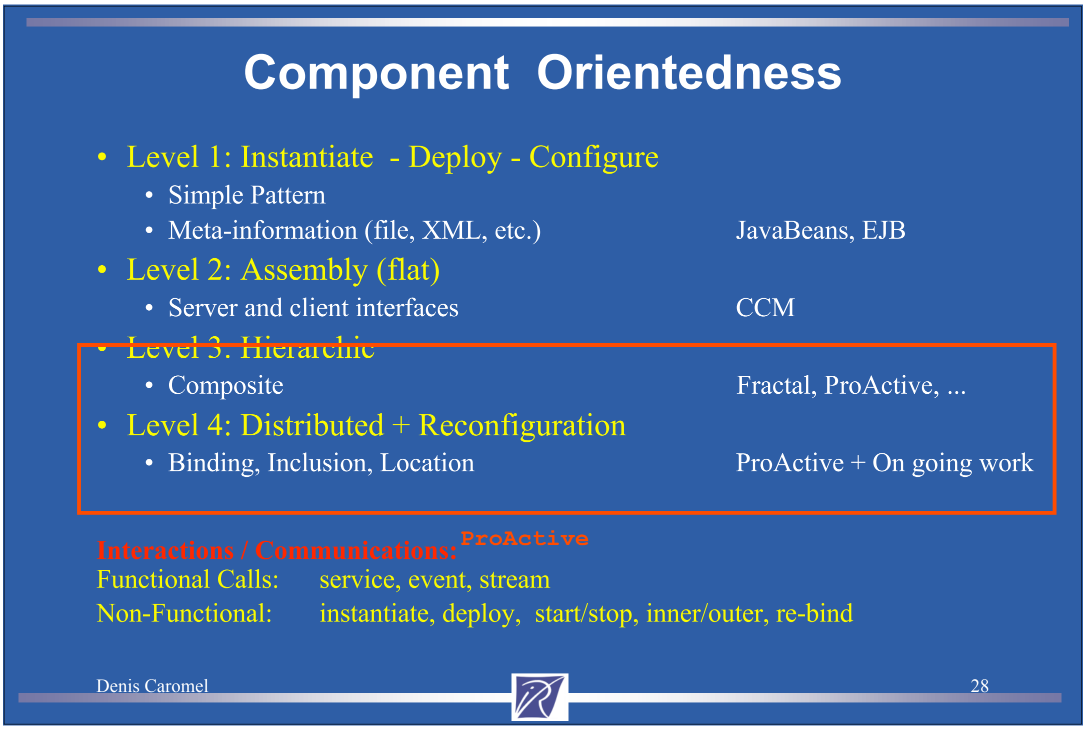
A CORBA Component

Courtesy of Philippe Merle, Lille, OpenCCM platform



Building CCM Applications = Assembling CORBA Component Instances

Provide + Use, but flat assembly



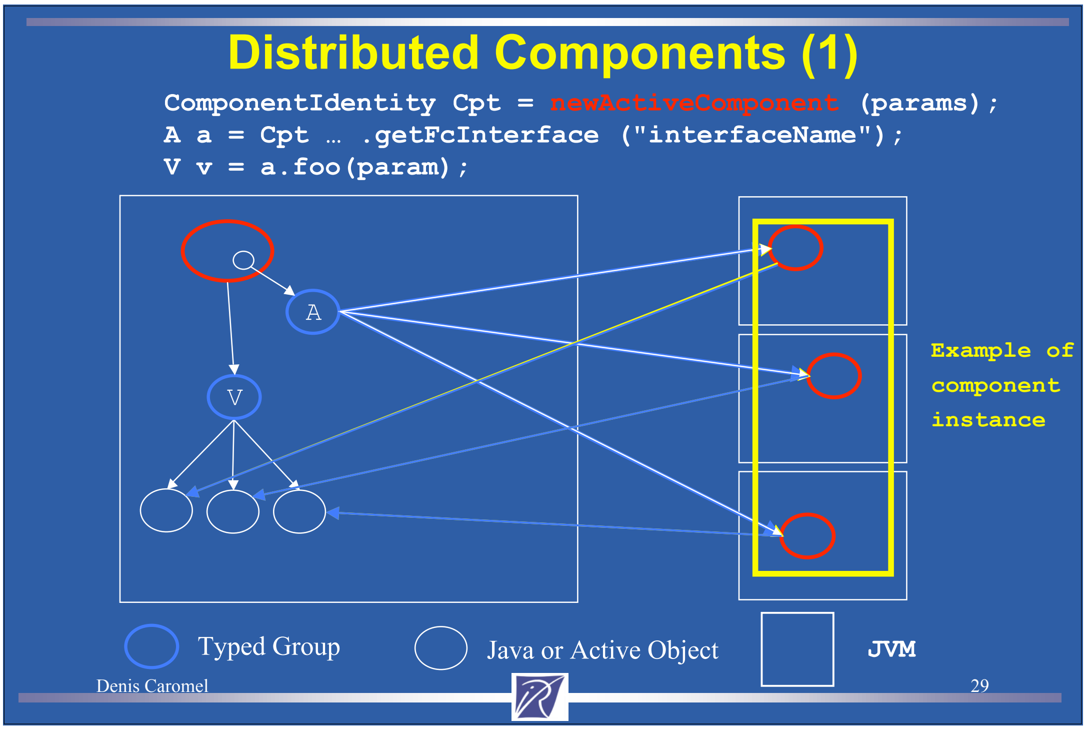
Component Orientedness

- Level 1: Instantiate - Deploy - Configure
  - Simple Pattern
  - Meta-information (file, XML, etc.) JavaBeans, EJB
- Level 2: Assembly (flat)
  - Server and client interfaces CCM
- Level 3: Hierarchic
  - Composite Fractal, ProActive, ...
- Level 4: Distributed + Reconfiguration
  - Binding, Inclusion, Location ProActive + On going work

Interactions / Communications: ProActive

Functional Calls: service, event, stream

Non-Functional: instantiate, deploy, start/stop, inner/outer, re-bind

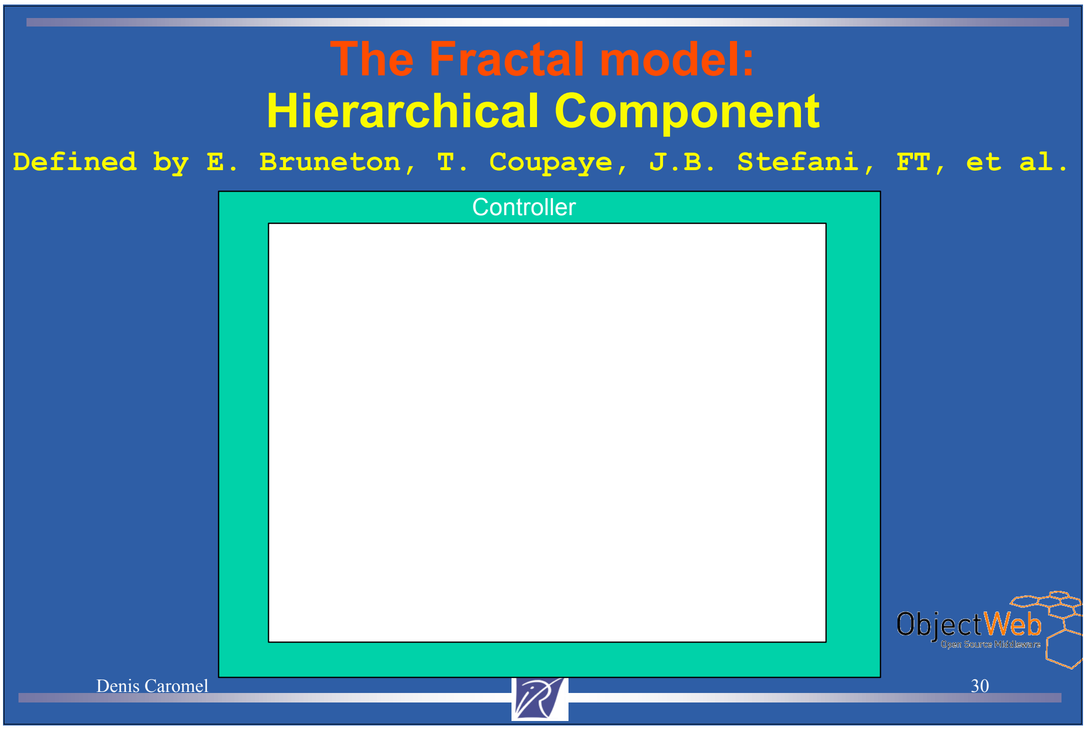


Distributed Components (1)

```
ComponentIdentity Cpt = newActiveComponent (params);
A a = Cpt … .getFcInterface ("interfaceName");
V v = a.foo(param);
```

Example of component instance

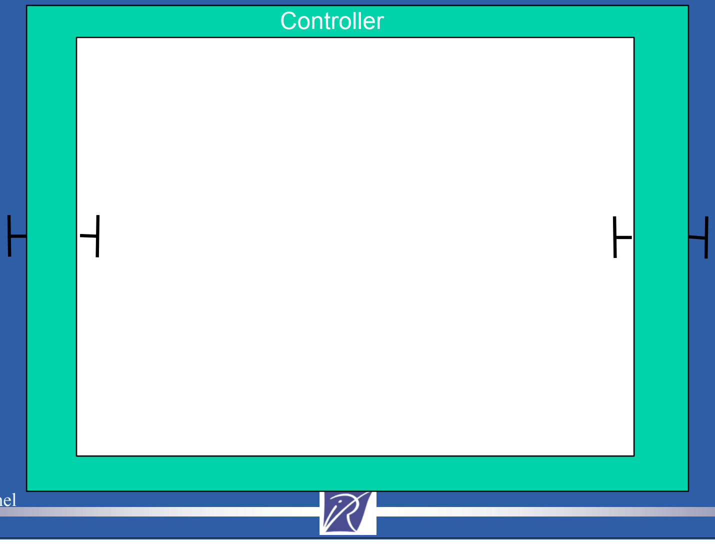
○ Typed Group ○ Java or Active Object □ JVM



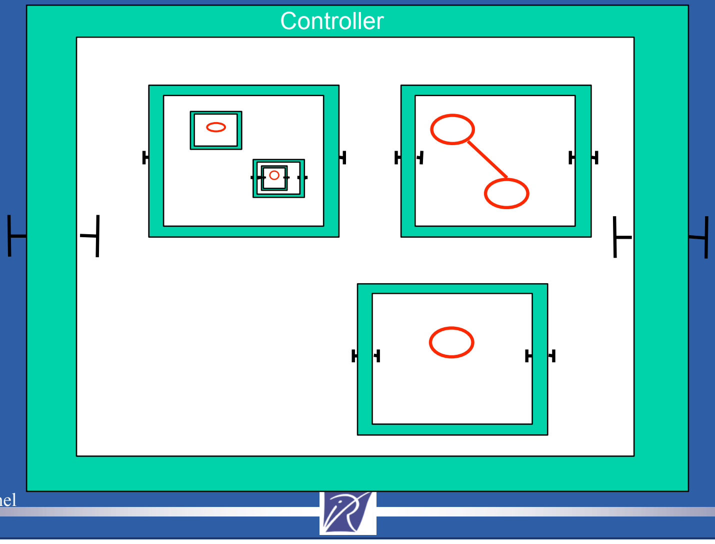
The Fractal model: Hierarchical Component

Defined by E. Bruneton, T. Coupaye, J.B. Stefani, FT, et al.

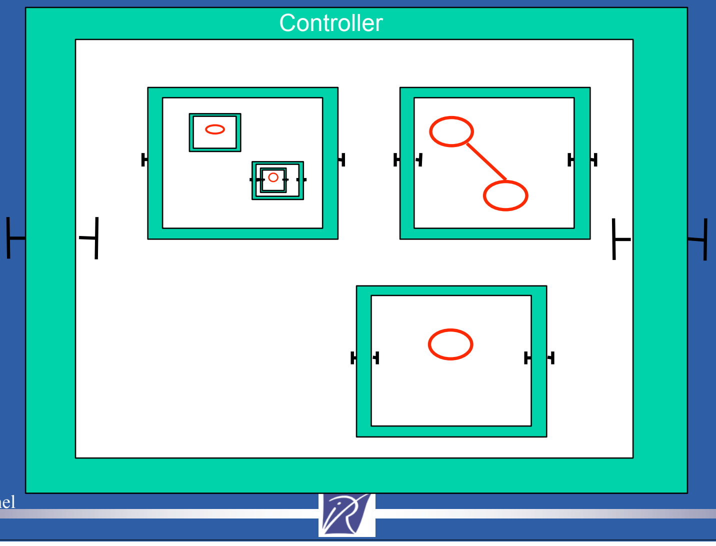
Interface = access point



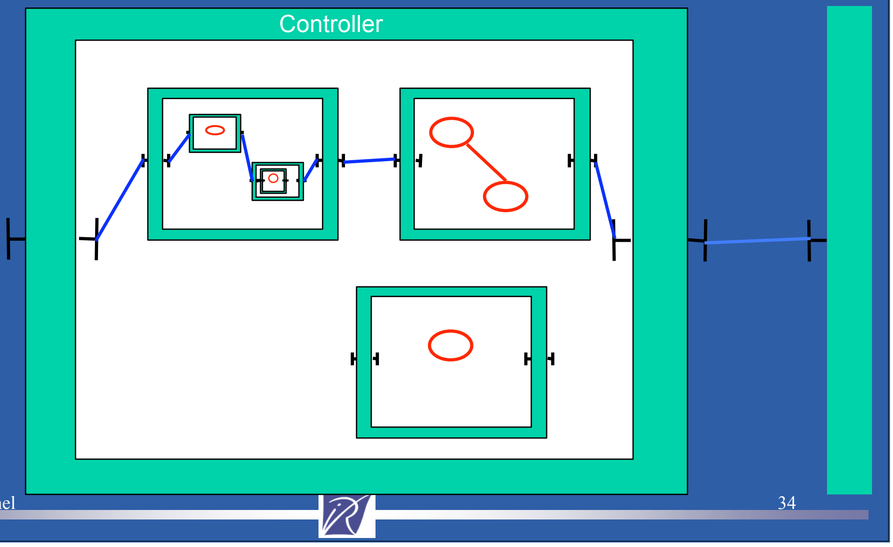
Hierarchical model : composites encapsulate primitives encapsulate Java code

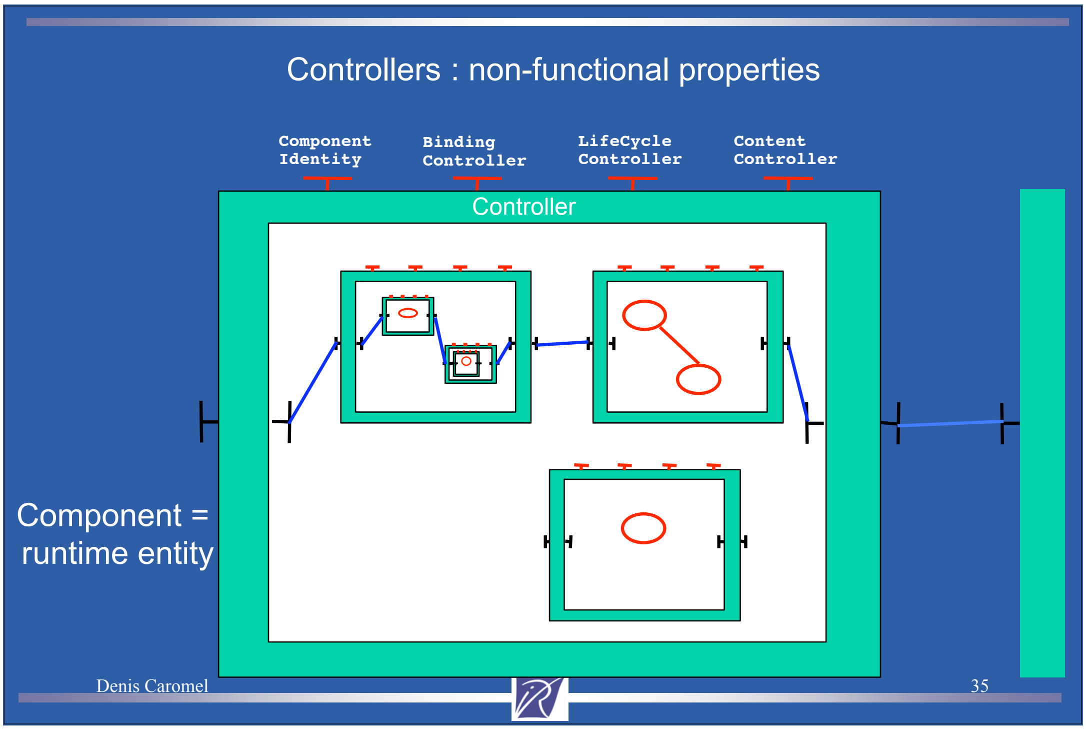


Binding = interaction



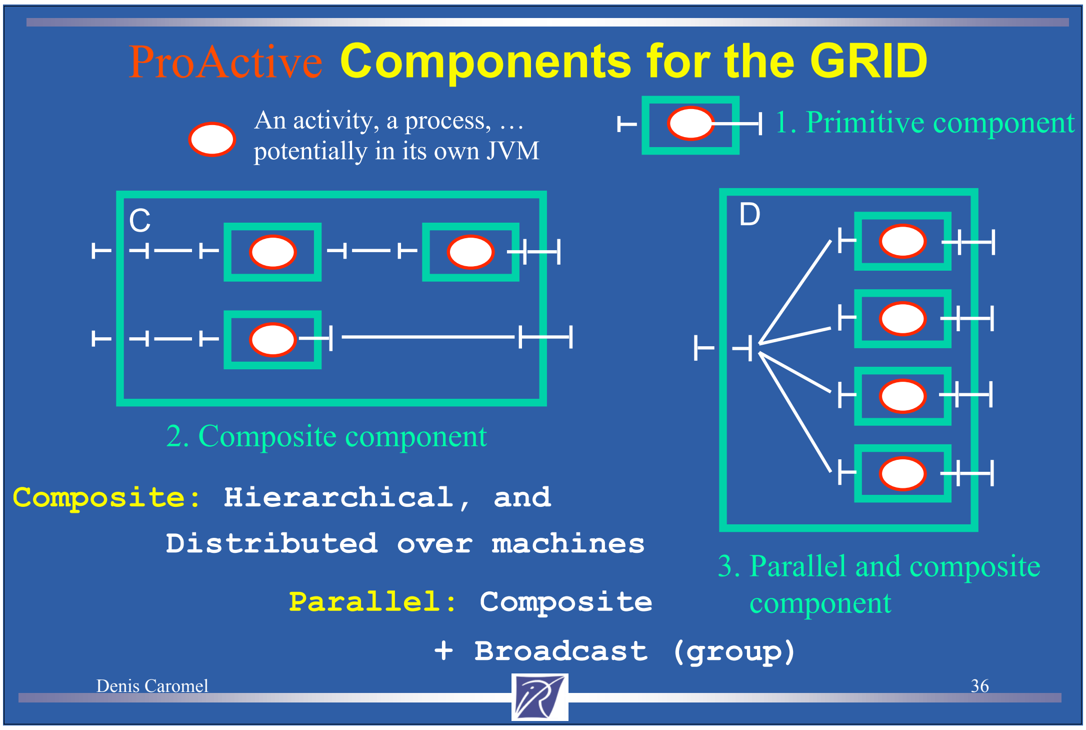
Binding = interaction





Controllers : non-functional properties

Component = runtime entity



ProActive Components for the GRID

An activity, a process, …
potentially in its own JVM

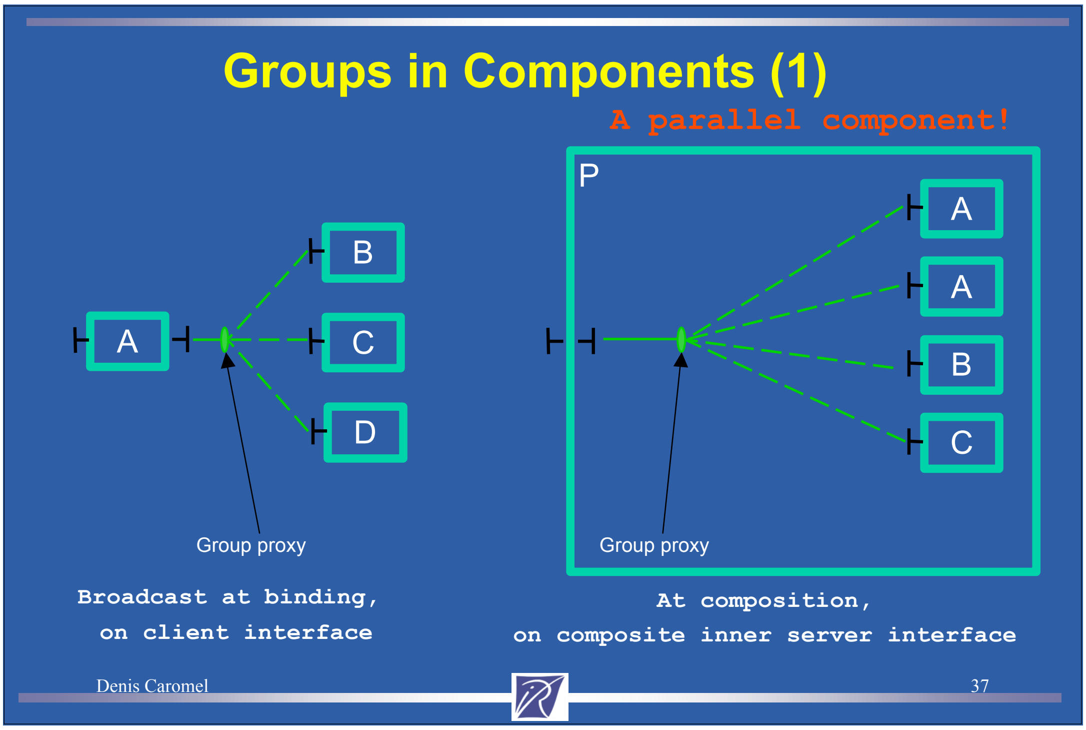
1. Primitive component

2. Composite component

3. Parallel and composite component

Composite: Hierarchical, and
Distributed over machines

Parallel: Composite
+ Broadcast (group)

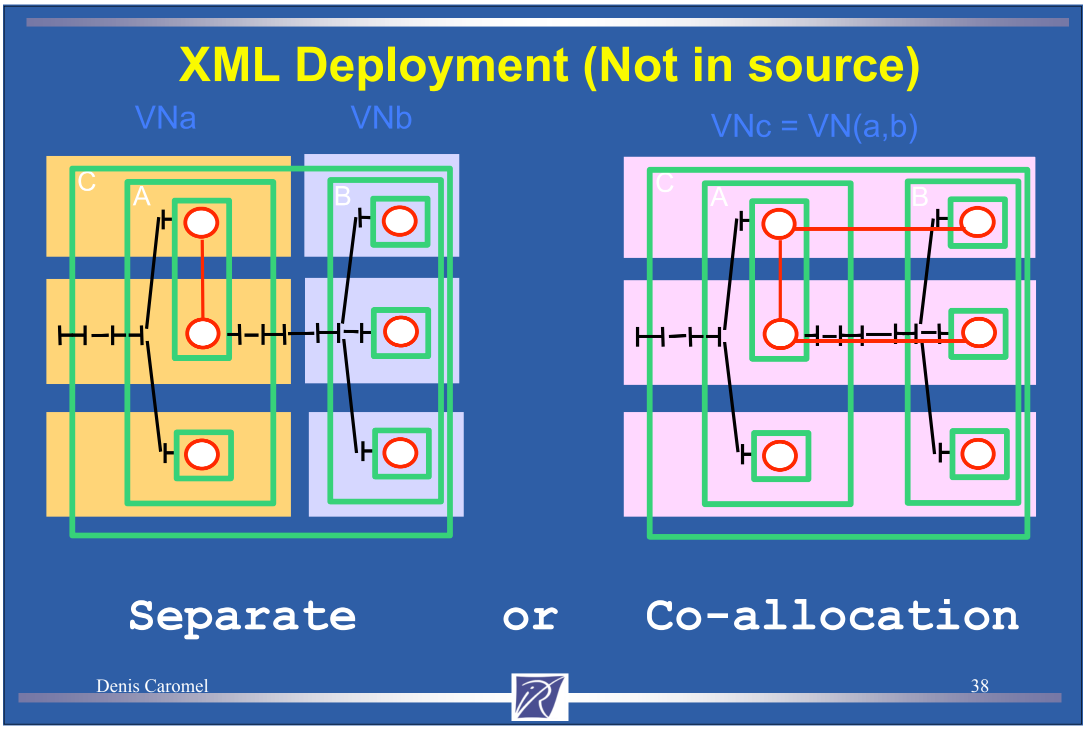


Groups in Components (1)

A parallel component!

Broadcast at binding,
on client interface

At composition,
on composite inner server interface



XML Deployment (Not in source)

Separate or Co-allocation

ProActive Component Definition

A component is:

- Formed from one (or several) Active Object
- Executing on one (or several) JVM
- Provides a set of server ports: Java Interfaces
- Uses a set of client ports: Java Attributes
- Point-to-point or Group communication between components

XML Example

Hierarchical:

- Primitive component: define with Java code and a descriptor
- Composite component: composition of primitive + composite
- Parallel component: multicast of calls in composites

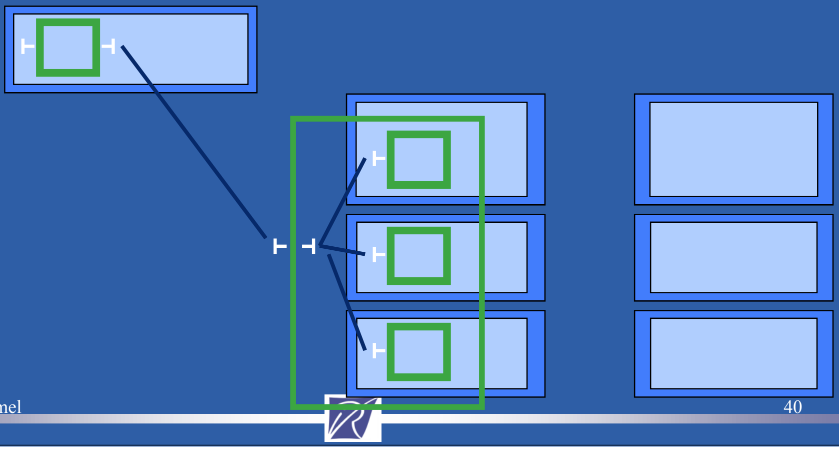
Descriptor:

- XML definition of primitive and composite (ADL)
- Virtual nodes capture the deployment capacities and needs

Virtual Node is a very important abstraction for GRID components

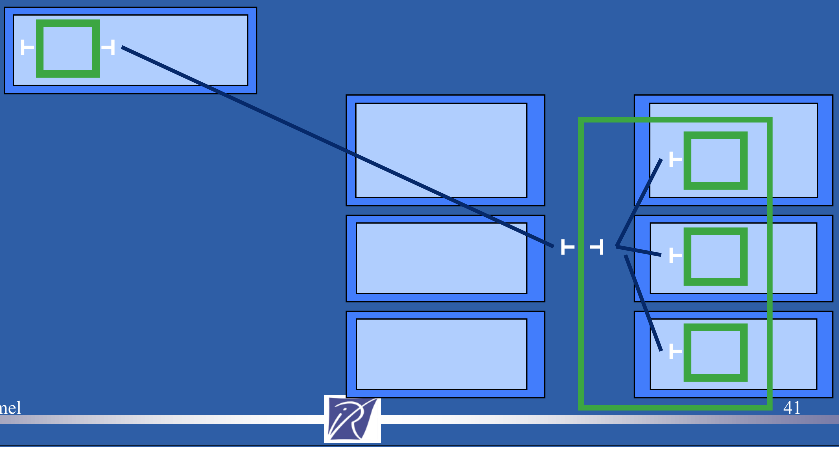
Migration Capability of composites

Migrate sets of components, including composites



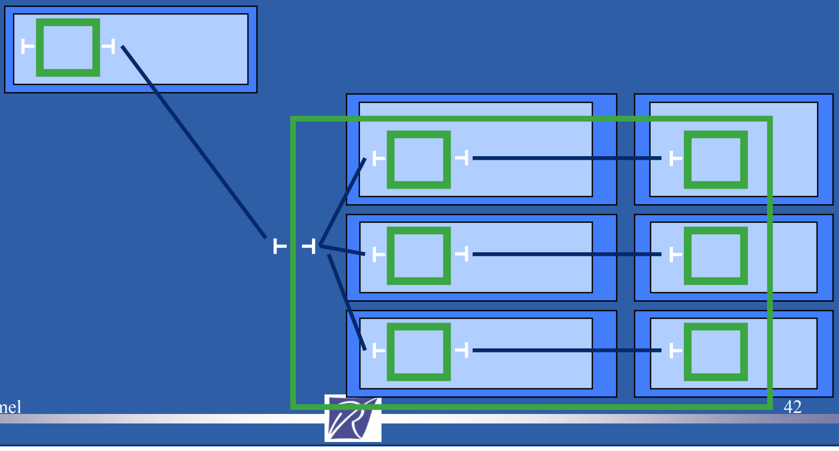
Migration Capability of composites

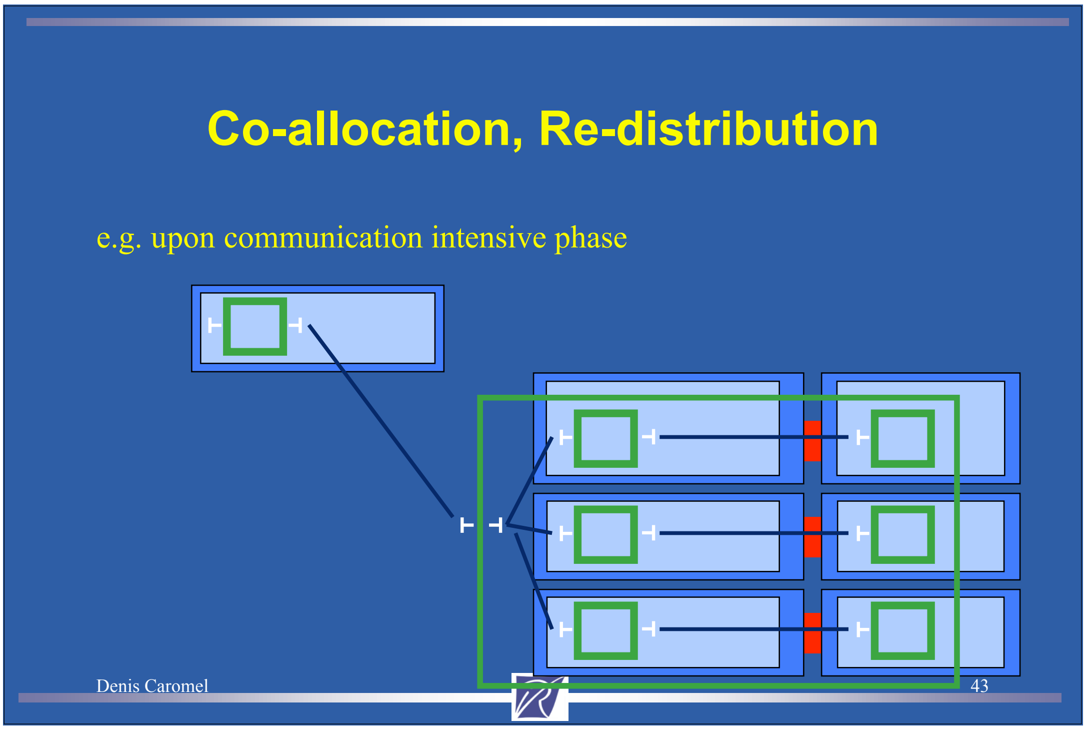
Migrate sets of components, including composites



Co-allocation, Re-distribution

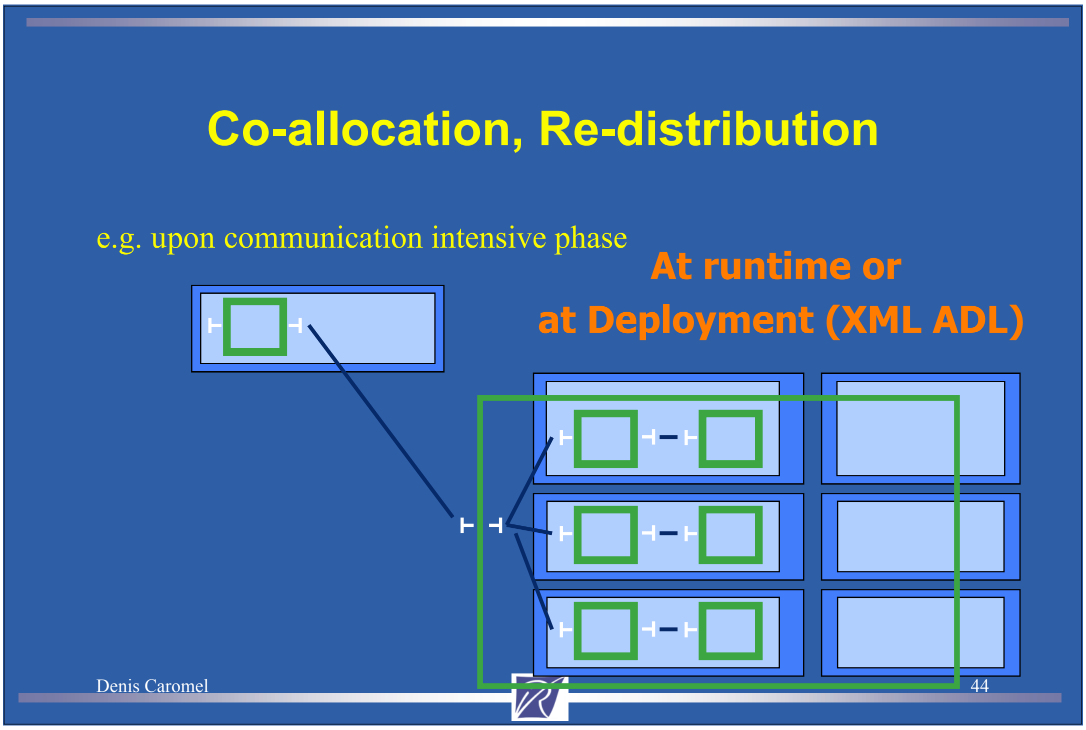
e.g. upon communication intensive phase





Co-allocation, Re-distribution

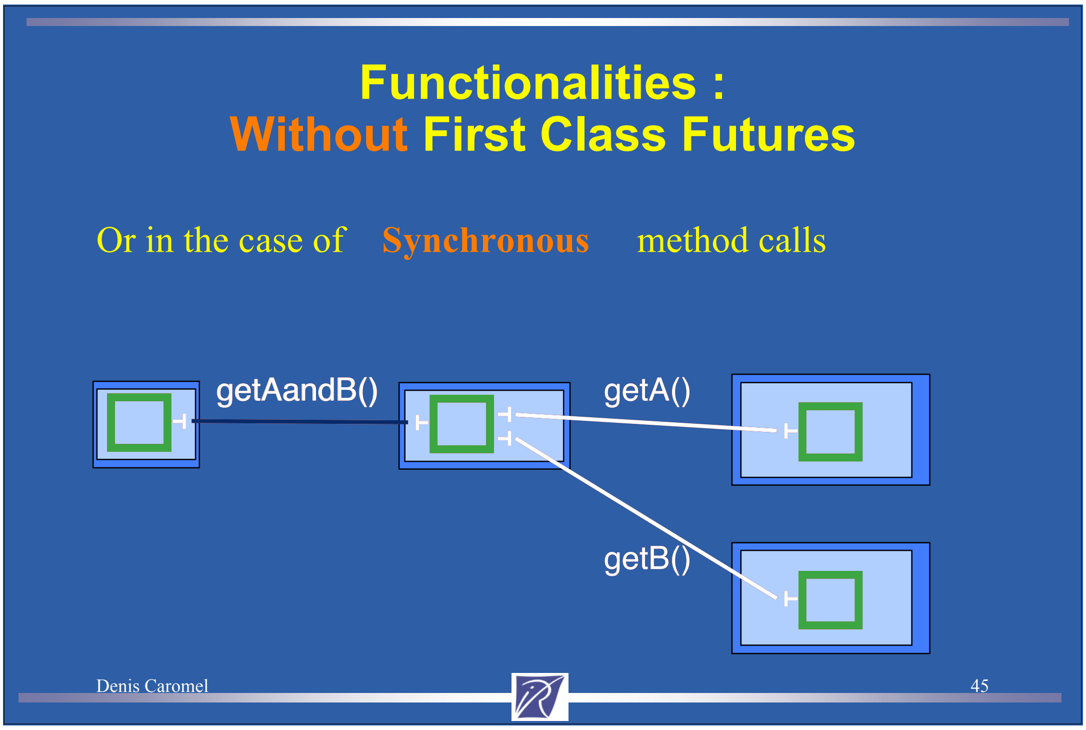
e.g. upon communication intensive phase



Co-allocation, Re-distribution

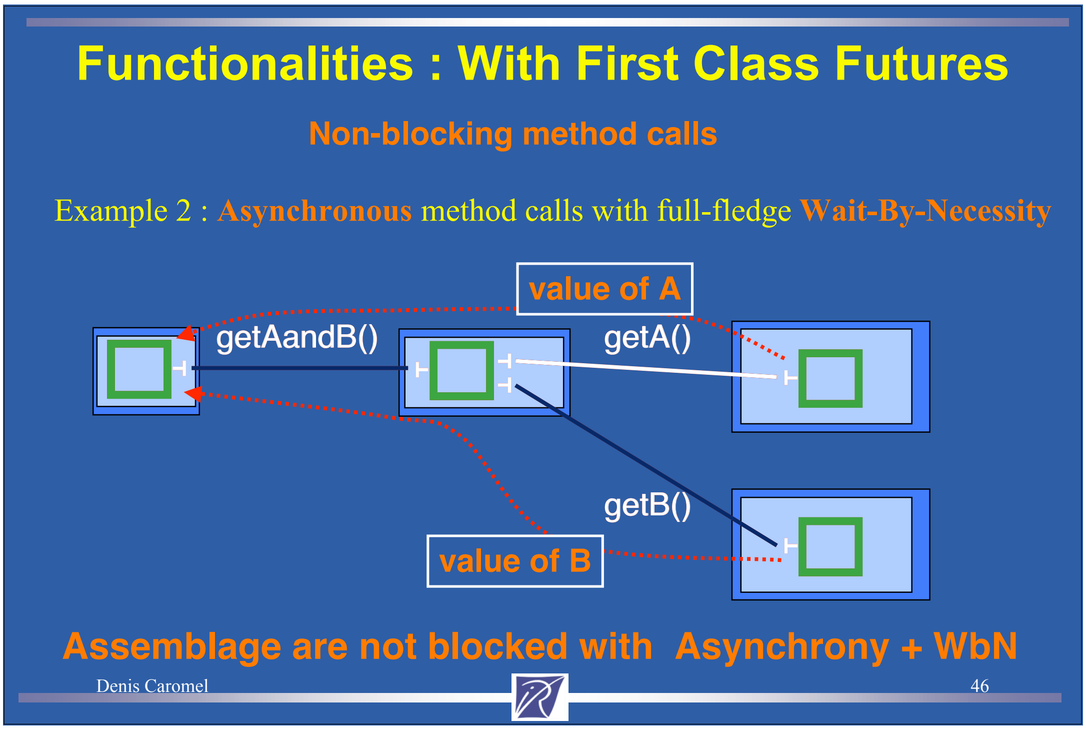
e.g. upon communication intensive phase

At runtime or
at Deployment (XML ADL)



Functionalities : Without First Class Futures

Or in the case of **Synchronous** method calls



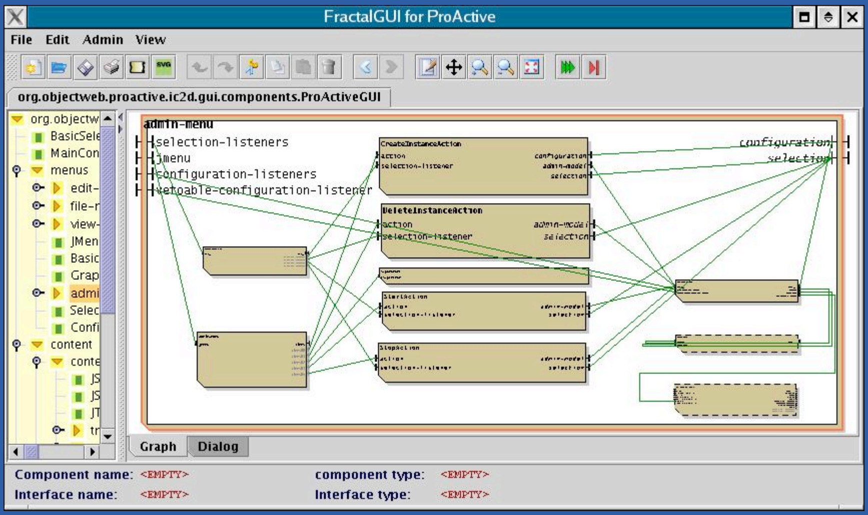
Functionalities : With First Class Futures

Non-blocking method calls

Example 2 : **Asynchronous** method calls with full-fledge **Wait-By-Necessity**

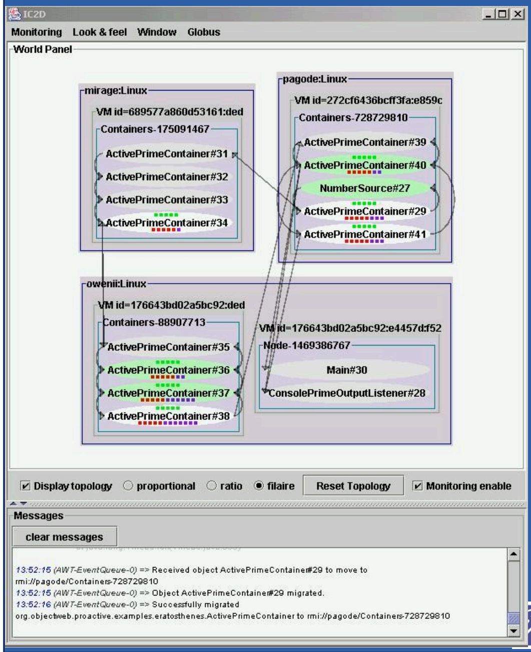
Assemblage are not blocked with Asynchrony + WbN

On-going work : GUI

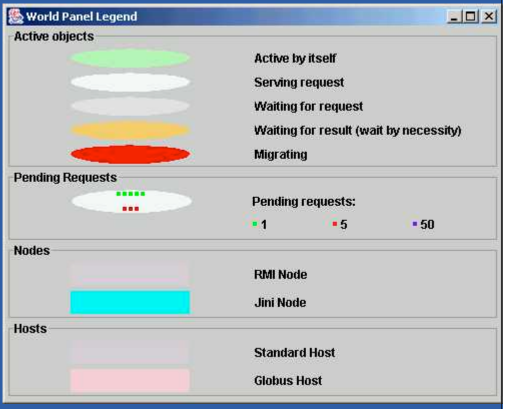


Component name: <EMPTY>
component type: <EMPTY>
Interface name: <EMPTY>
Interface type: <EMPTY>

IC2D: Interactive Control and Debugging of Distribution



```
13:52:15 (AWT-EventQueue-0) => Received object ActivePrimeContainer#29 to move to
rmi://pagode/Containers-728729810
13:52:15 (AWT-EventQueue-0) => Object ActivePrimeContainer#29 migrated.
13:52:16 (AWT-EventQueue-0) => Successfully migrated
org.objectweb.proactive.examples.eratosthenes.ActivePrimeContainer to rmi://pagode/Containers-728729810
```



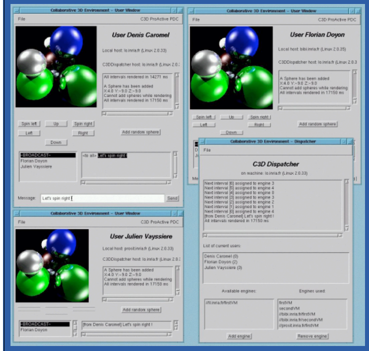
With any ProActive application

Features:

Graphical and Textual visualization

Monitoring and Control

C3D Monitoring: graphical and textual com.



User Denis Caromel

Local host: lo.inria.fr (Linux 2.0.33)

C3DDispatcher host: lo.inria.fr (Linux 2.0.3

All intervals rendered in 14271 ms
A Sphere has been added
X:4.0 Y:-9.0 Z:-9.0
Cannot add spheres while rendering
All intervals rendered in 17150 ms

~BROADCAST~
Florian Doyon
Julien Vayssiere

<to all> Let's spin right !

Message: Let's spin right !

User Florian Doyon

Local host: bibi.inria.fr (Linux 2.0.35)

C3DDispatcher host: lo.inria.fr (Linux 2.0.3

A Sphere has been added
X:4.0 Y:-9.0 Z:-9.0
Cannot add spheres while rendering
All intervals rendered in 17150 ms

User Julien Vayssiere

Local host: prosit.inria.fr (Linux 2.0.33)

C3DDispatcher host: lo.inria.fr (Linux 2.0.3

A Sphere has been added
X:4.0 Y:-9.0 Z:-9.0
Cannot add spheres while rendering
All intervals rendered in 17150 ms

~BROADCAST~
Florian Doyon

[from Denis Caromel] Let's spin right !

C3D Dispatcher

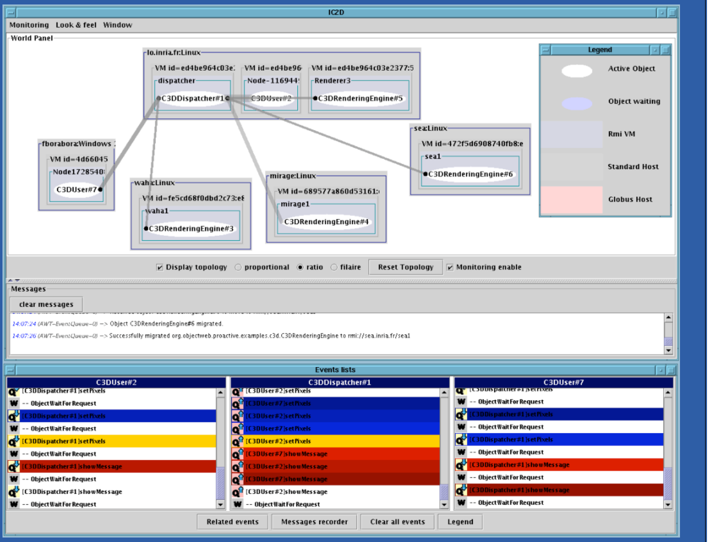
on machine: lo.inria.fr (Linux 2.0.33)

Next interval [6] assigned to engine 3
Next interval [5] assigned to engine 4
Next interval [4] assigned to engine 0
Next interval [3] assigned to engine 3
Next interval [2] assigned to engine 2
Next interval [1] assigned to engine 1
Next interval [0] assigned to engine 4
[from Denis Caromel] Let's spin right !
All intervals rendered in 17150 ms

List of current users:
Denis Caromel (0)
Florian Doyon (2)
Julien Vayssiere (3)

Available engines:
//lo.inria.fr/firstVM

Engines used:
firstVM
secondVM
//bibi.inria.fr/firstVM
//bibi.inria.fr/secondVM
//prosit.inria.fr/firstVM



☑ Display topology ○ proportional ● ratio ○ filaire ☑ Monitoring enable

14:07:24 (AWT-EventQueue-0) -> Object C3DRenderingEngine#6 migrated.
14:07:26 (AWT-EventQueue-0) -> Successfully migrated org.objectweb.proactive.examples.c3d.C3DRenderingEngine to rmi://sea.inria.fr/sea1

C3DUser#2

[C3DDispatcher#1]setPixels
-- ObjectWaitForRequest
[C3DDispatcher#1]setPixels
-- ObjectWaitForRequest
[C3DDispatcher#1]setPixels
-- ObjectWaitForRequest
[C3DDispatcher#1]showMessage
-- ObjectWaitForRequest
[C3DDispatcher#1]showMessage
-- ObjectWaitForRequest

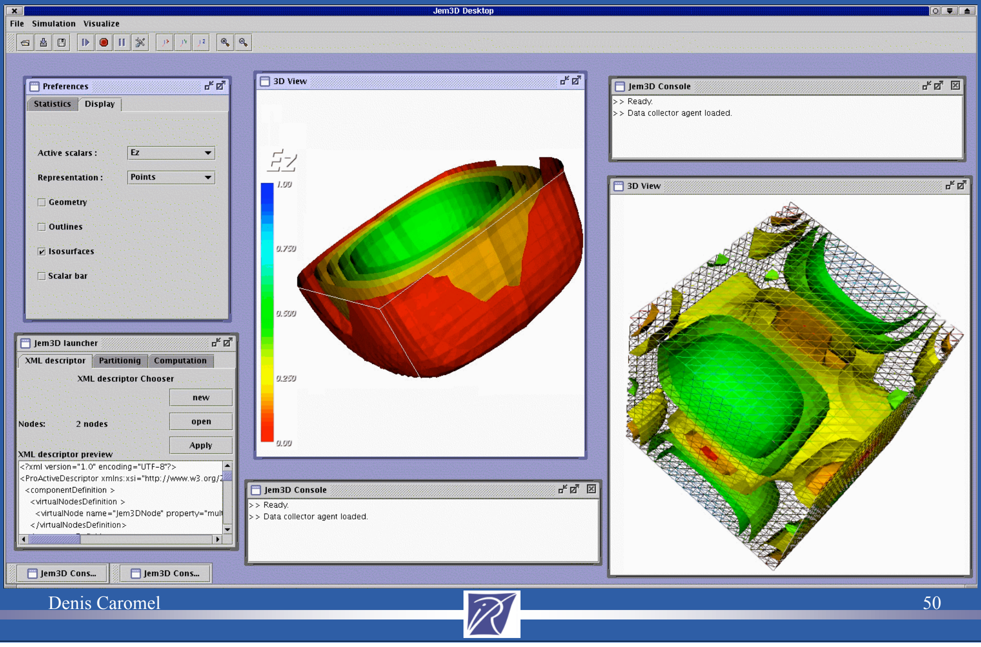
C3DDispatcher#1

[C3DUser#2]setPixels
[C3DUser#7]setPixels
[C3DUser#2]setPixels
[C3DUser#7]setPixels
[C3DUser#2]setPixels
[C3DUser#7]showMessage
[C3DUser#2]showMessage
[C3DUser#7]showMessage
[C3DUser#2]showMessage
-- ObjectWaitForRequest

C3DUser#7

[C3DDispatcher#1]setPixels
-- ObjectWaitForRequest
[C3DDispatcher#1]setPixels
-- ObjectWaitForRequest
[C3DDispatcher#1]setPixels
-- ObjectWaitForRequest
[C3DDispatcher#1]showMessage
-- ObjectWaitForRequest
[C3DDispatcher#1]showMessage
-- ObjectWaitForRequest

Jem3D



Active scalars : Ez

Representation : Points

☐ Geometry

☐ Outlines

☑ Isosurfaces

☐ Scalar bar

Nodes: 2 nodes

```
<?xml version="1.0" encoding="UTF-8"?>
<ProActiveDescriptor xmlns:xsi="http://www.w3.org/2
  <componentDefinition >
    <virtualNodesDefinition >
      <virtualNode name="Jem3DNode" property="mult
    </virtualNodesDefinition>
```

```
>> Ready.
>> Data collector agent loaded.
```

```
>> Ready.
>> Data collector agent loaded.
```

JEM 3D : Java 3D Electromagnetism

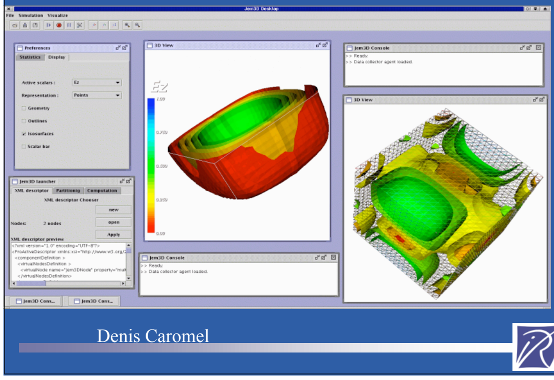
together with Said El Kasmi, Stéphane Lanteri (caiman)

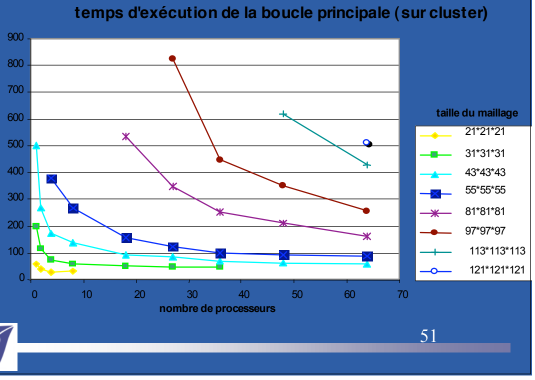
Maxwell 3D equation solver, Finite Volume Method (FVM)

Pre-existing Fortran MPI version: EM3D (CAIMAN team @ INRIA)

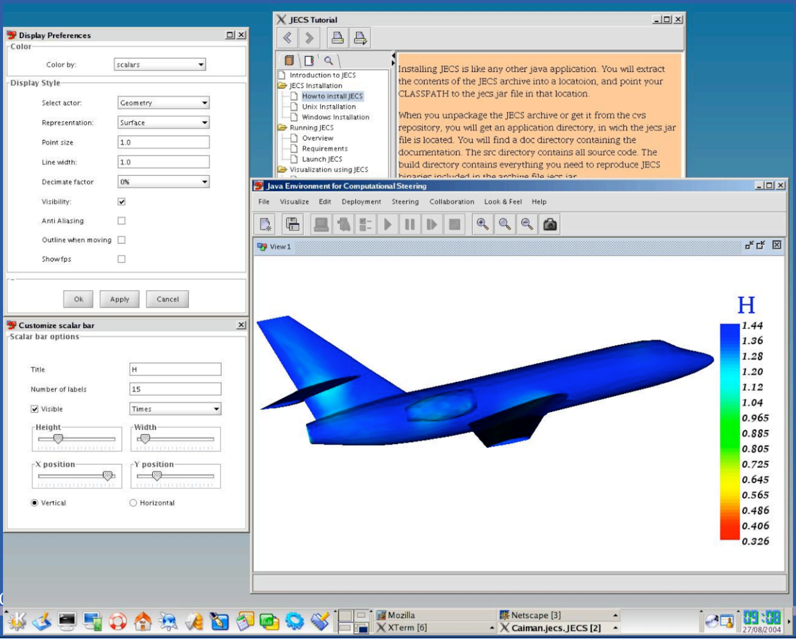
Up to 294 machines at the same time (Intranet and cluster)

Large data sets: 150x150x150 (100 million facets)





JECS : A Generic Version of Jem3D



Color by: scalars

Select actor: Geometry

Representation: Surface

Point size 1.0

Line width: 1.0

Decimate factor 0%

Visibility: ☑

Anti Aliasing ☐

Outline when moving ☐

Showfps ☐

Title H

Number of labels 15

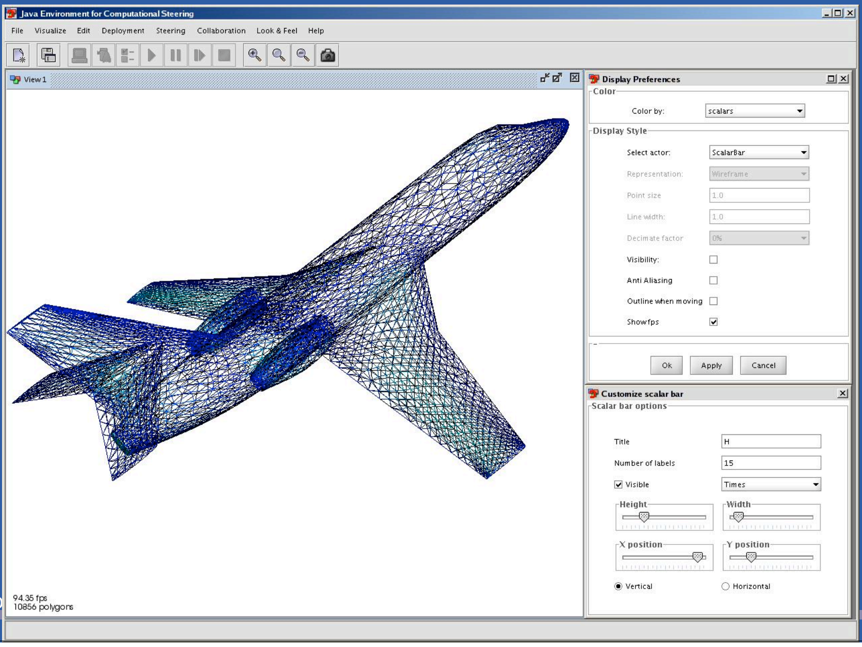
☑ Visible Times

● Vertical ○ Horizontal

Installing JECS is like any other java application. You will extract the contents of the JECS archive into a locatoion, and point your CLASSPATH to the jecs.jar file in that location.

When you unpackage the JECS archive or get it from the cvs repository, you will get an application directory, in wich the jecs.jar file is located. You will find a doc directory containing the documentation. The src directory contains all source code. The build directory contains everything you need to reproduce JECS binaries included in the archive file jecs.jar

JECS : A Generic Version of Jem3D



94.35 fps
10856 polygons

Color

Color by: scalars

Display Style

Select actor: ScalarBar

Representation: Wireframe

Point size 1.0

Line width: 1.0

Decimate factor 0%

Visibility: □

Anti Aliasing □

Outline when moving □

Showfps ☑

Scalar bar options

Title H

Number of labels 15

☑ Visible Times

Height

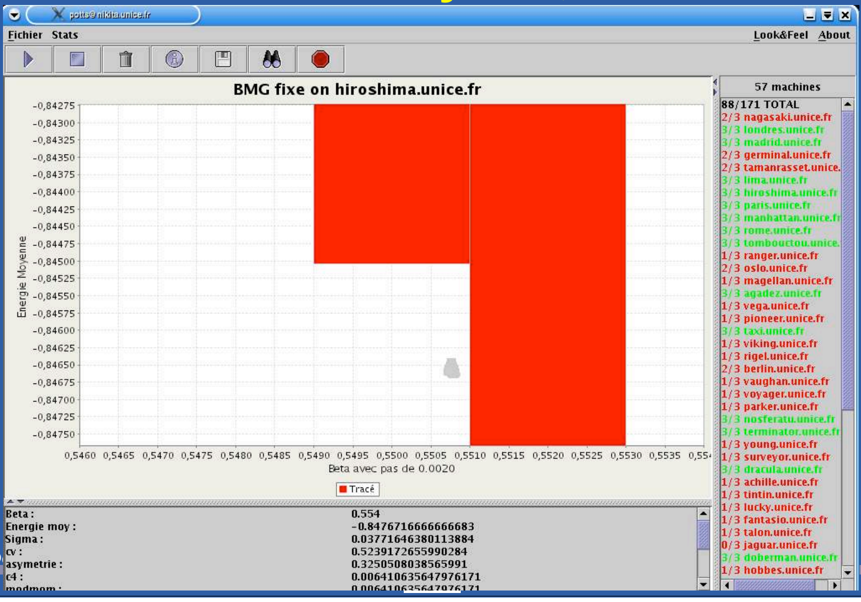
Width

X position

Y position

● Vertical ○ Horizontal

Monte Carlo Simulations, Non-Linear Physics, INLN



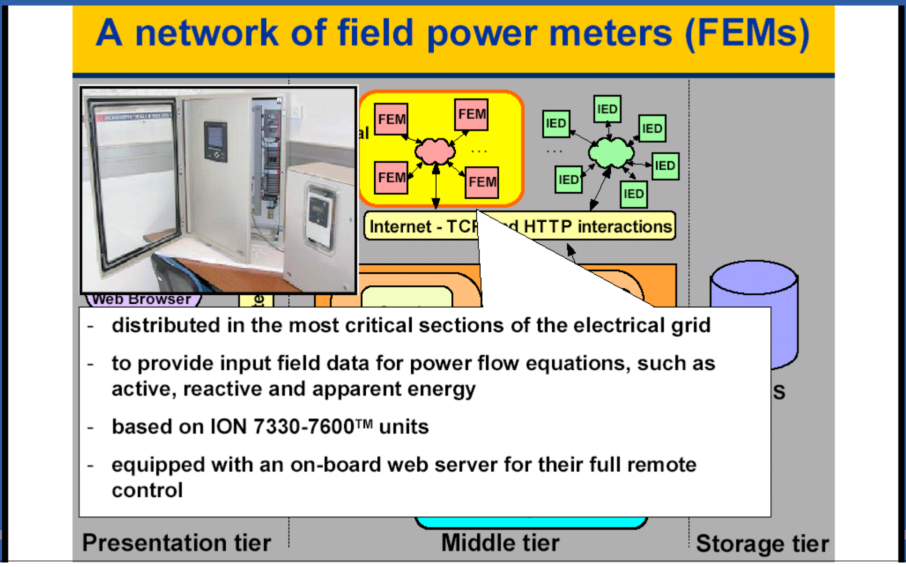
```
Beta :                 0.554
Energie moy :          -0.8476716666666683
Sigma :                0.03771646380113884
cv :                   0.5239172655990284
asymetrie :            0.3250508038565991
c4 :                   0.006410635647976171
modmom :               0.006410635647976171
```

```
57 machines
88/171 TOTAL
2/3 nagasaki.unice.fr
3/3 londres.unice.fr
3/3 madrid.unice.fr
2/3 germinal.unice.fr
2/3 tamanrasset.unice.
3/3 lima.unice.fr
3/3 hiroshima.unice.fr
3/3 paris.unice.fr
3/3 manhattan.unice.fr
3/3 rome.unice.fr
3/3 tombouctou.unice.
1/3 ranger.unice.fr
2/3 oslo.unice.fr
1/3 magellan.unice.fr
3/3 agadez.unice.fr
1/3 vega.unice.fr
1/3 pioneer.unice.fr
3/3 taxi.unice.fr
1/3 viking.unice.fr
1/3 rigel.unice.fr
2/3 berlin.unice.fr
1/3 vaughan.unice.fr
1/3 voyager.unice.fr
1/3 parker.unice.fr
3/3 nosferatu.unice.fr
3/3 terminator.unice.fr
1/3 young.unice.fr
1/3 surveyor.unice.fr
3/3 dracula.unice.fr
1/3 achille.unice.fr
1/3 tintin.unice.fr
1/3 lucky.unice.fr
1/3 fantasio.unice.fr
1/3 talon.unice.fr
0/3 jaguar.unice.fr
3/3 doberman.unice.fr
1/3 hobbes.unice.fr
```

Electric Network Planning, E. Zimeo et al., Benevento (Naples), Italy

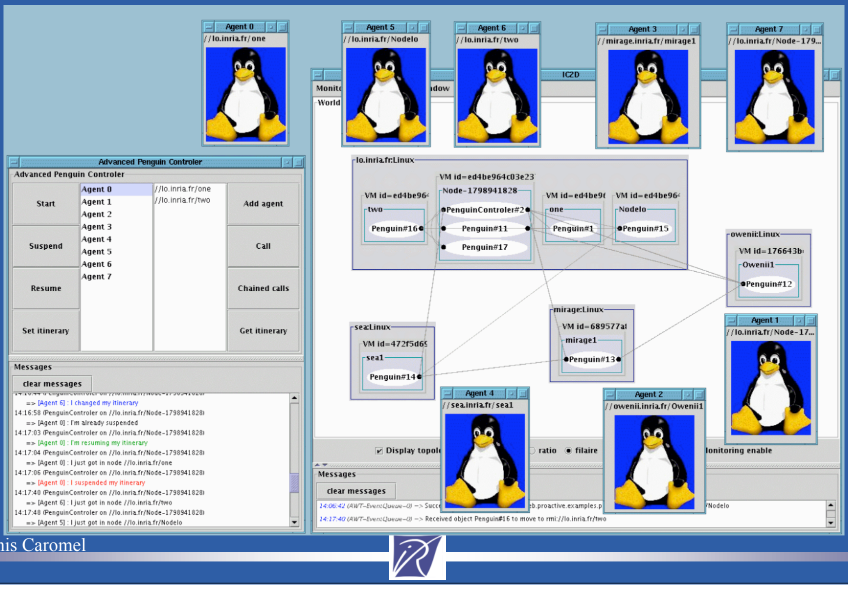
On-line Power Systems Security Analysis (OPSSA)

A network of field power meters (FEMs)



- distributed in the most critical sections of the electrical grid
- to provide input field data for power flow equations, such as active, reactive and apparent energy
- based on ION 7330-7600™ units
- equipped with an on-board web server for their full remote control

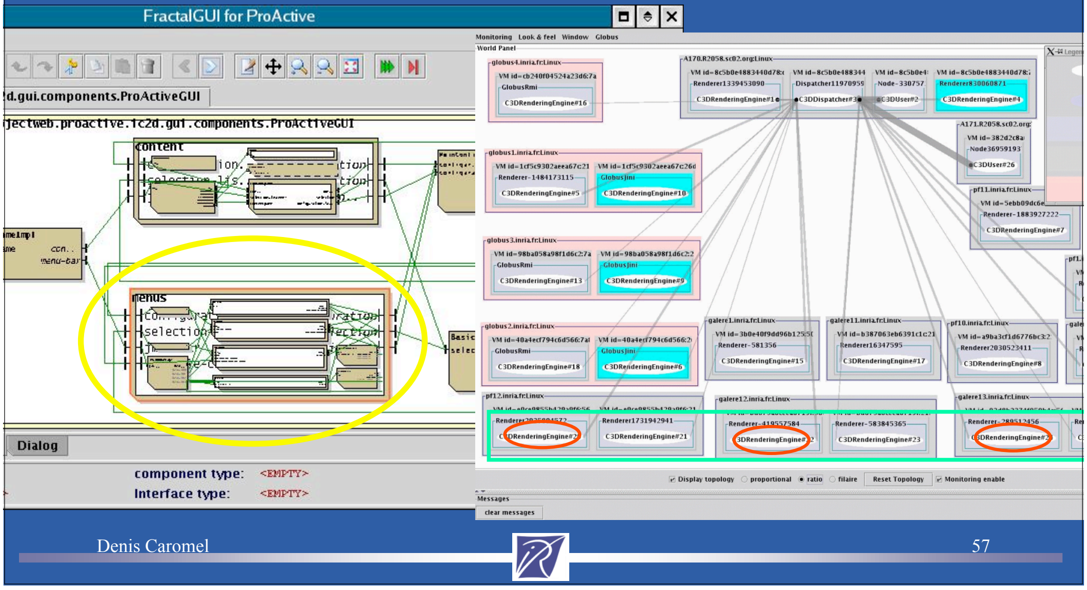
Mobile Application executing on 7 JVMs



```
   => [Agent 6] : I changed my itinerary
14:16:58 (PenguinControler on //lo.inria.fr/Node-1798941828)
   => [Agent 0] : I'm already suspended
14:17:03 (PenguinControler on //lo.inria.fr/Node-1798941828)
   => [Agent 0] : I'm resuming my itinerary
14:17:04 (PenguinControler on //lo.inria.fr/Node-1798941828)
   => [Agent 0] : I just got in node //lo.inria.fr/one
14:17:06 (PenguinControler on //lo.inria.fr/Node-1798941828)
   => [Agent 0] : I suspended my itinerary
14:17:40 (PenguinControler on //lo.inria.fr/Node-1798941828)
   => [Agent 6] : I just got in node //lo.inria.fr/two
14:17:48 (PenguinControler on //lo.inria.fr/Node-1798941828)
   => [Agent 5] : I just got in node //lo.inria.fr/Nodelo
```

```
14:06:42 (AWT-EventQueue-0) -> Succe              eb.proactive.examples.p              /Nodelo
14:17:40 (AWT-EventQueue-0) -> Received object Penguin#16 to move to rmi://lo.inria.fr/two
```

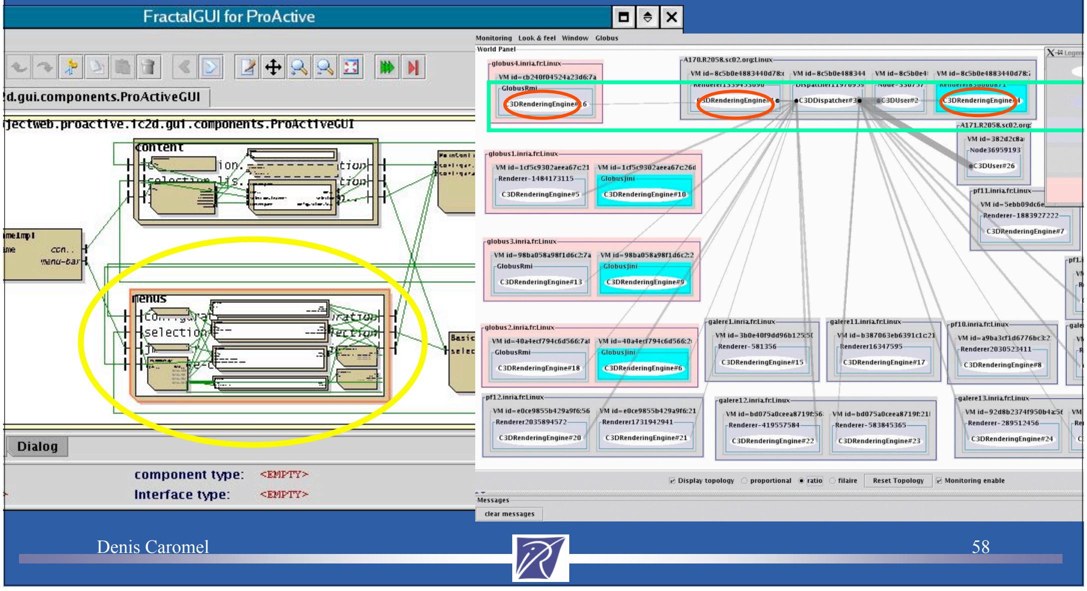
Perspective for Components - PSE
Graphical Composition, Monitoring, Migration


FractalGUI for ProActive
Monitoring Look & feel Window Globus
World Panel
content
menus
Dialog
☑ Display topology ○ proportional ● ratio ○ filaire Reset Topology ☑ Monitoring enable
Messages
clear messages

component type: <EMPTY>

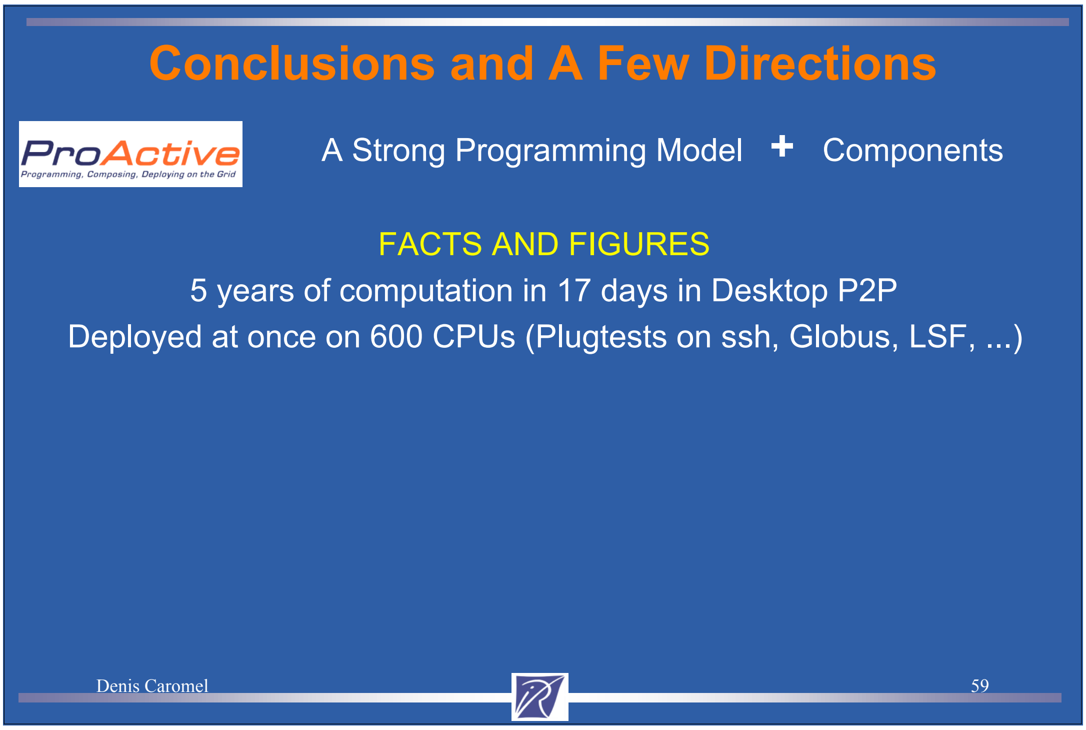
Interface type: <EMPTY>

Perspective for Components - PSE
Graphical Composition, Monitoring, Migration


FractalGUI for ProActive
Monitoring Look & feel Window Globus
World Panel
content
menus
Dialog
☑ Display topology ○ proportional ● ratio ○ filaire Reset Topology ☑ Monitoring enable
Messages
clear messages

component type: <EMPTY>

Interface type: <EMPTY>



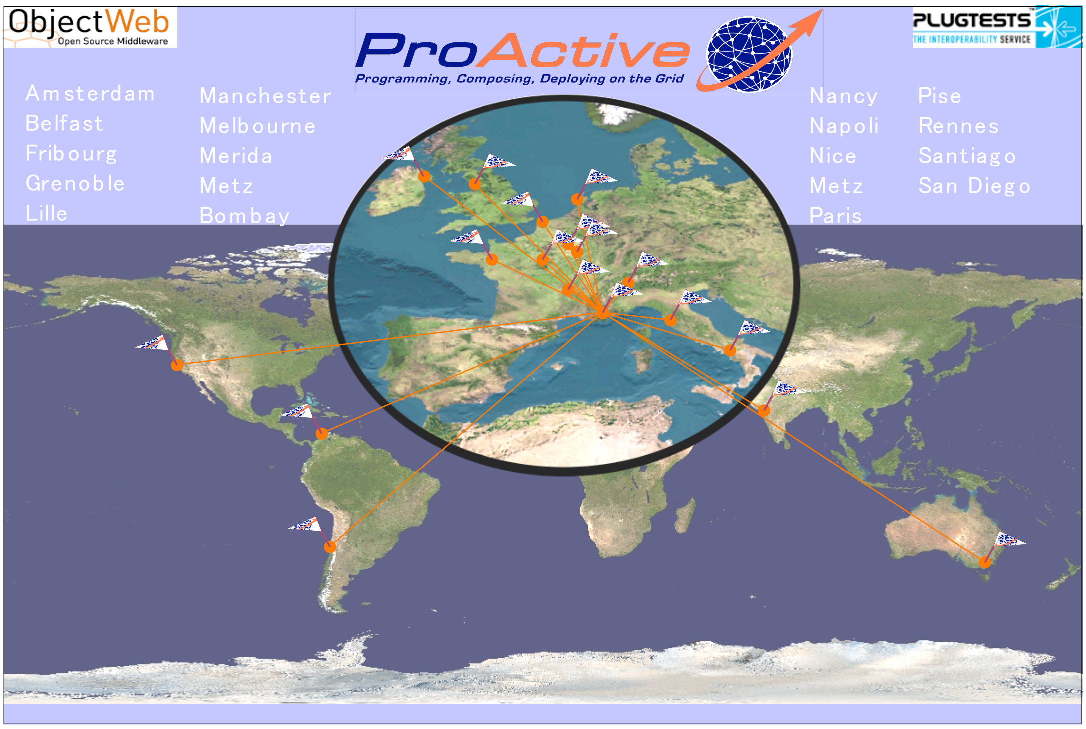
Conclusions and A Few Directions

A Strong Programming Model + Components

FACTS AND FIGURES

5 years of computation in 17 days in Desktop P2P

Deployed at once on 600 CPUs (Plugtests on ssh, Globus, LSF, ...)



Amsterdam
Belfast
Fribourg
Grenoble
Lille

Manchester
Melbourne
Merida
Metz
Bombay

Nancy
Napoli
Nice
Metz
Paris

Pise
Rennes
Santiago
San Diego

Conclusions and A Few Directions

ProActive
Programming, Composing, Deploying on the Grid

A Strong Programming Model + Components

FACTS AND FIGURES

5 years of computation in 17 days in Desktop P2P

Deployed at once on 600 CPUs (Plugtests on ssh, Globus, LSF, ...)
(Close to) Beating Fortran on an Electromagnetic Application

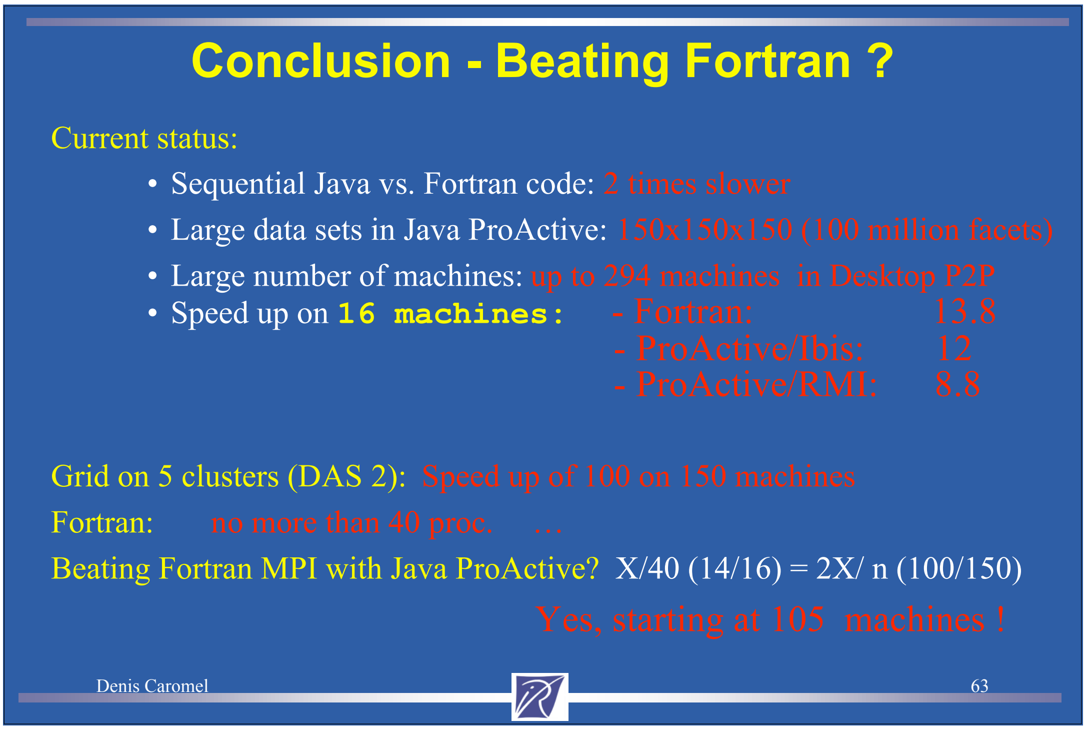
PERSPECTIVES FOR COMPONENTS

Safe Reconfiguration

How to specify for components: QoS, Ranking, etc. ?

A great alchemy for the Grid:

Asynchrony + Wait By Necessity + Groups + Components



Conclusion - Beating Fortran ?

Current status:

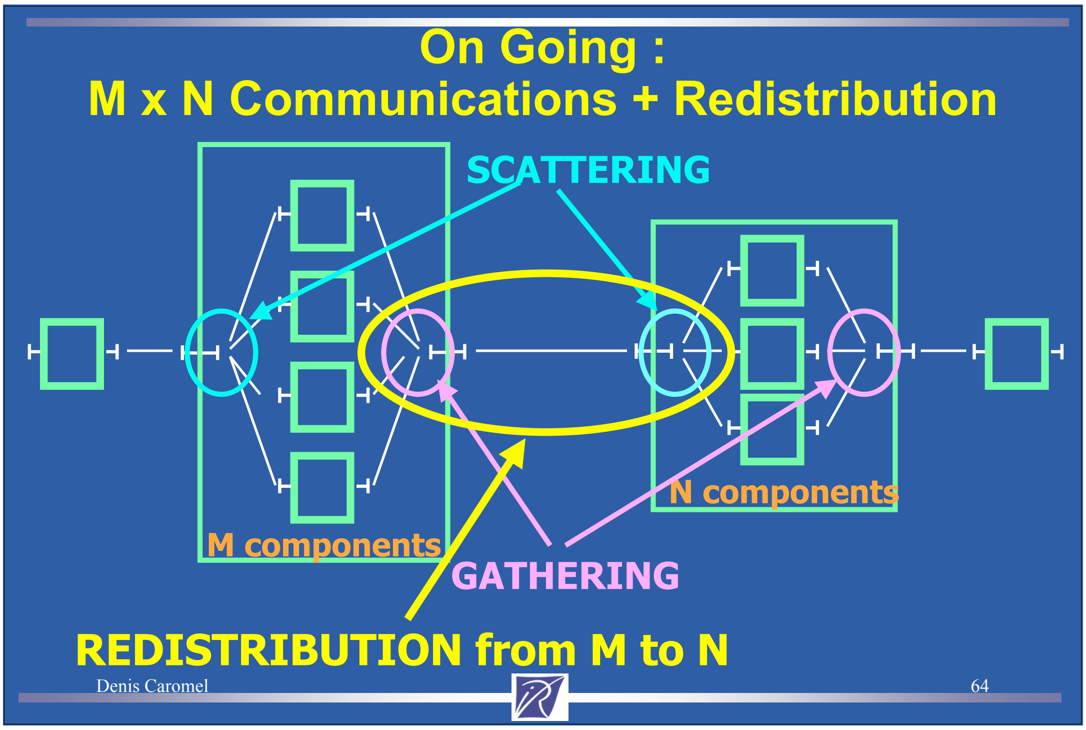
- Sequential Java vs. Fortran code: 2 times slower
- Large data sets in Java ProActive: 150x150x150 (100 million facets)
- Large number of machines: up to 294 machines in Desktop P2P
- Speed up on **16 machines:** - Fortran: 13.8
- ProActive/Ibis: 12
- ProActive/RMI: 8.8

Grid on 5 clusters (DAS 2): Speed up of 100 on 150 machines

Fortran: no more than 40 proc. …

Beating Fortran MPI with Java ProActive? X/40 (14/16) = 2X/ n (100/150)

Yes, starting at 105 machines !



On Going : M x N Communications + Redistribution

SCATTERING

M components

N components

GATHERING

REDISTRIBUTION from M to N

Adaptive Feature: Multi-transports layer RMI, RMI-ssh, …, Ibis, HTTP XML, …

Adaptive choice of transport layer between:

- RMI
- ssh/RMI

Also available with static configuration:

- Ibis (TCP, Myrinet, etc.)
- HTTP
- … ssh/HTTP

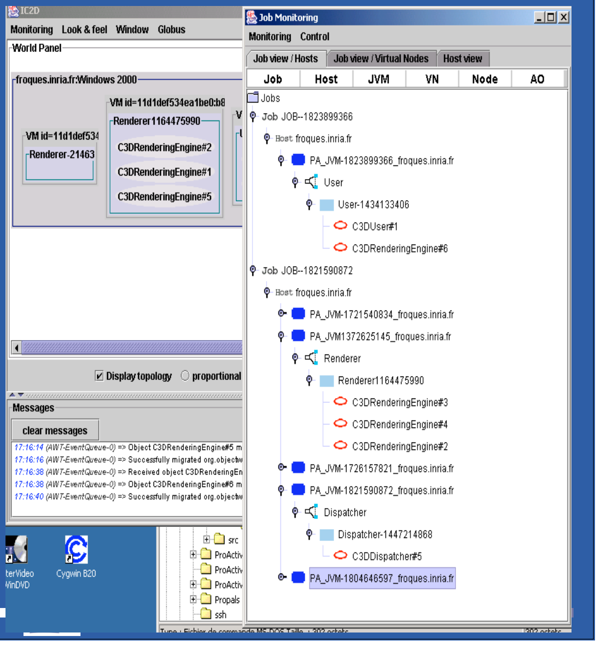
Short Term Perspective:

Fully Adaptive Choice between all transports

IC2D: Basic features cont.

Job Management:

- JVM, AO per Job ID
- Textual visualisation,
- control (kill all, etc.)



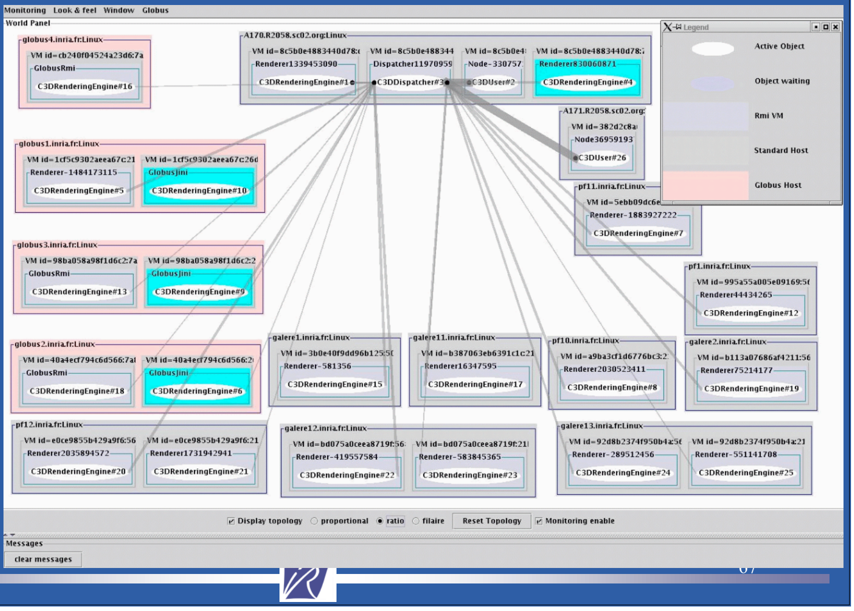
```
17:16:14 (AWT-EventQueue-0) => Object C3DRenderingEngine#5 m
17:16:16 (AWT-EventQueue-0) => Successfully migrated org.objectw
17:16:38 (AWT-EventQueue-0) => Received object C3DRenderingEn
17:16:38 (AWT-EventQueue-0) => Object C3DRenderingEngine#6 m
17:16:40 (AWT-EventQueue-0) => Successfully migrated org.objectw
```

```
Jobs
Job JOB--1823899366
  Host froques.inria.fr
    PA_JVM-1823899366_froques.inria.fr
      User
        User-1434133406
          C3DUser#1
          C3DRenderingEngine#6
Job JOB--1821590872
  Host froques.inria.fr
    PA_JVM-1721540834_froques.inria.fr
    PA_JVM1372625145_froques.inria.fr
      Renderer
        Renderer1164475990
          C3DRenderingEngine#3
          C3DRenderingEngine#4
          C3DRenderingEngine#2
    PA_JVM-1726157821_froques.inria.fr
    PA_JVM-1821590872_froques.inria.fr
      Dispatcher
        Dispatcher-1447214868
          C3DDispatcher#5
    PA_JVM-1804646597_froques.inria.fr
```

Monitoring of RMI, Globus, Jini, LSF cluster Nice -- Baltimore

ProActive IC2D:

Width of links proportional to the number of communications



☑ Display topology ○ proportional ● ratio ○ filaire ☑ Monitoring enable

ProActive:
A Java API + Tools for Parallel, Distributed Computing

- A uniform framework: An Active Object pattern
- A formal model behind: Determinism, Insensitivity to deployment

Programming Model:

- **Remote Objects (Classes, not only Interfaces, Dynamic)**
- **Asynchronous Communications, Automatic dataflow synchro: Futures**
- **Groups, Mobility, Components, Security**

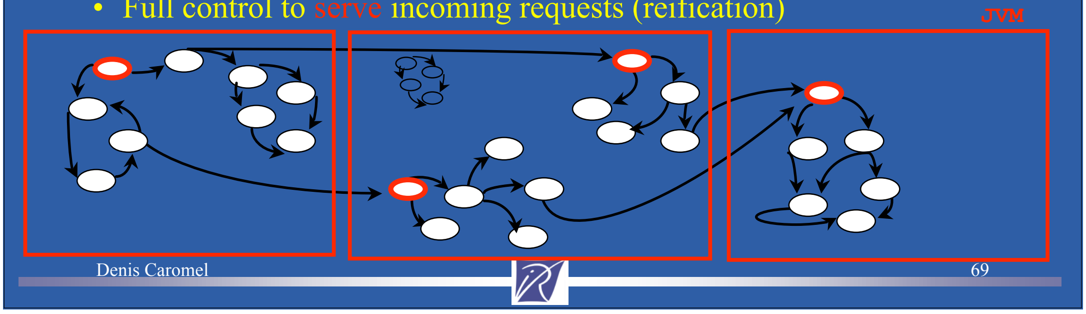
Environment:

- **XML Deployment Descriptors**
- **Interfaced with various protocols:** `rsh,ssh,LSF,Globus,Jini,RMIregistry`
- **Visualization and monitoring: IC2D**

In the www. ObjectWeb .org Consortium (Open Source middleware) since April 2002 (LGPL license)

ProActive : model

- Active objects : coarse-grained structuring entities (subsystems)
- Each active object: - possibly owns many passive objects
  - has exactly one thread.
- No shared passive objects -- Parameters are passed by deep-copy
- Asynchronous Communication between active objects
- Future objects and wait-by-necessity.
- Full control to serve incoming requests (reification)



ProActive model (2)

Java RMI (Remote Method Invocation = Object RPC = o.foo(p))

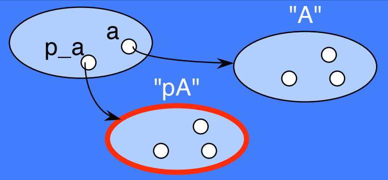
plus a few important features:

- Sequential Object: a single thread with FIFO service
- Asynchronous Method calls towards Active Objects:
  Implicit Futures as method results
- Wait-By-Necessity:
  - Automatic wait upon a strict operation on an unknown future
  - First-Class Futures:
    - Futures can be passed to other activities
    - Sending a future to another machines is not blocking

ProActive : Reuse and seamless

Two key features:

- Polymorphism between standard and active objects
  - Type compatibility for classes (and not only interfaces)
  - Needed and done for the future objects also
  - Dynamic mechanism (dynamically achieved if needed)



```
foo (A a)
{
   a.g (...);
   v = a.f (...);
   ...
   v.bar (...);
}
```

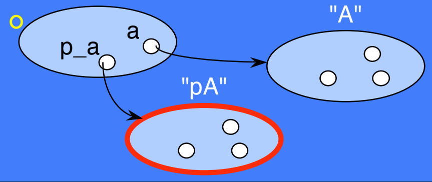
- Wait-by-necessity: inter-object synchronization
  - Systematic, implicit and transparent futures

    Ease the programming of synchronizations, and the reuse of routines

ProActive : Reuse and seamless

Two key features:

- Polymorphism between standard and active objects
  - Type compatibility for classes (and not only interfaces)
  - Needed and done for the future objects also
  - Dynamic mechanism (dynamically achieved if needed)



```
foo (A a)
{
   a.g (...);
   v = a.f (...);
   ...
   v.bar (...);
}
```

```
O.foo(a) : a.g()
and a.f() are
« local »
O.foo(p_a):
a.g() and
a.f()are «remote
+ Async.»
```

- Wait-by-necessity: inter-object synchronization
  - Systematic, implicit and transparent futures (“value to come”)

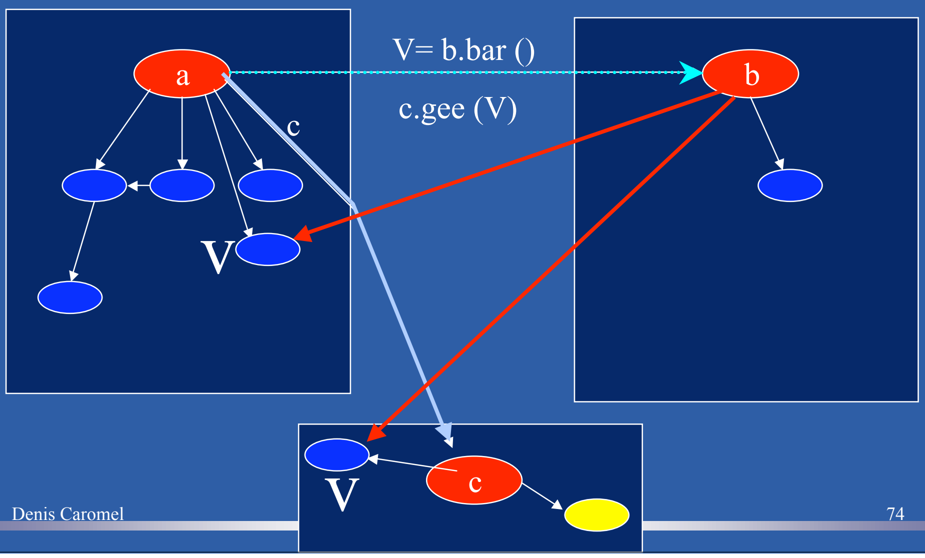
    Ease the programming of synchronizations, and the reuse of routines

First-Class Futures

Update

Wait-By-Necessity: First Class Futures

Futures are Global Single-Assignment Variables



Future update strategies

No partial replies and requests:

- No passing of futures between activities, more deadlocks

Eager strategies: as soon as a future is computed

- Forward-based:
  - Each activity is responsible for updating the values of futures it has forwarded
- Message-based:
  - Each forwarding of future generates a message sent to the computing activity
  - The computing activity is responsible for sending the value to all

Mixed strategy:

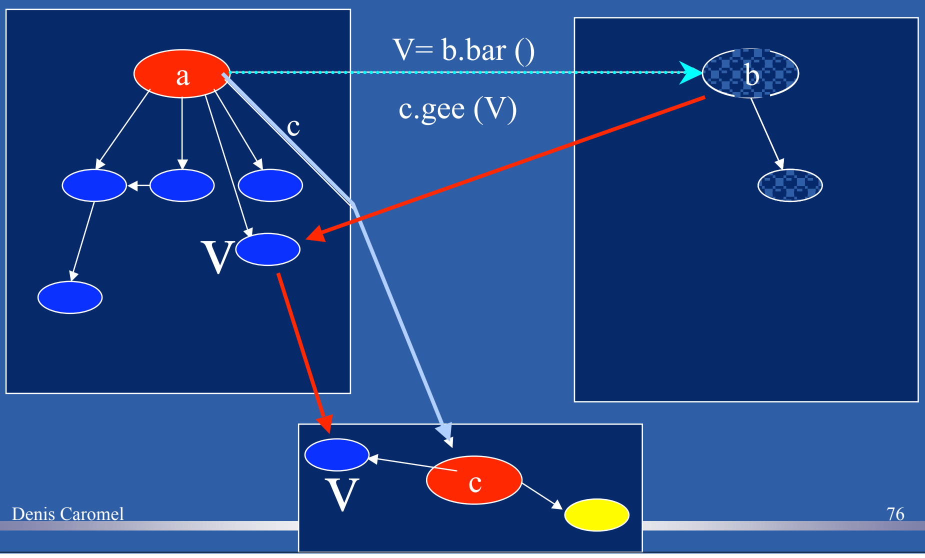
- Futures update any time between future computation and WbN

Lazy strategy:

- On demand, only when the value of the future is needed (WbN on it)

Wait-By-Necessity: Eager Forward Based

`AO forwarding a future: will have to forward its value`

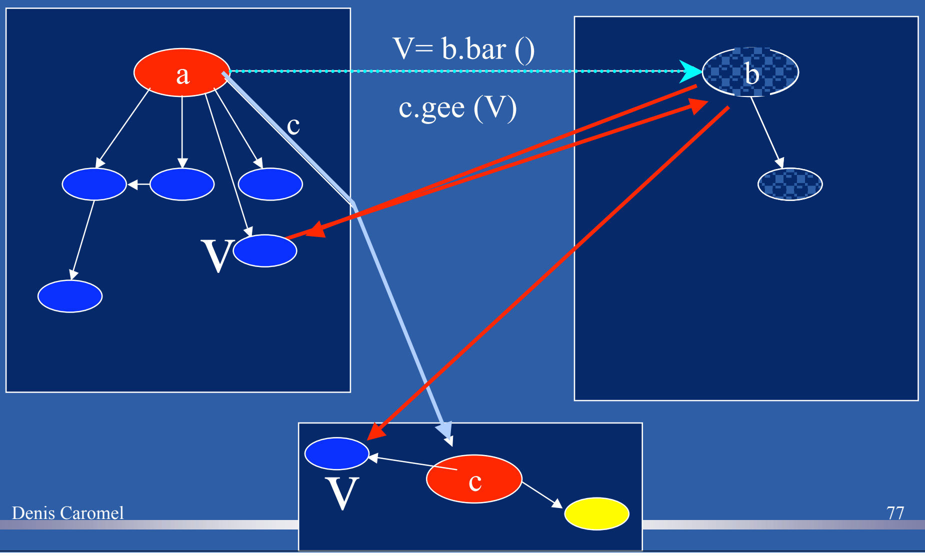


V= b.bar ()

c.gee (V)

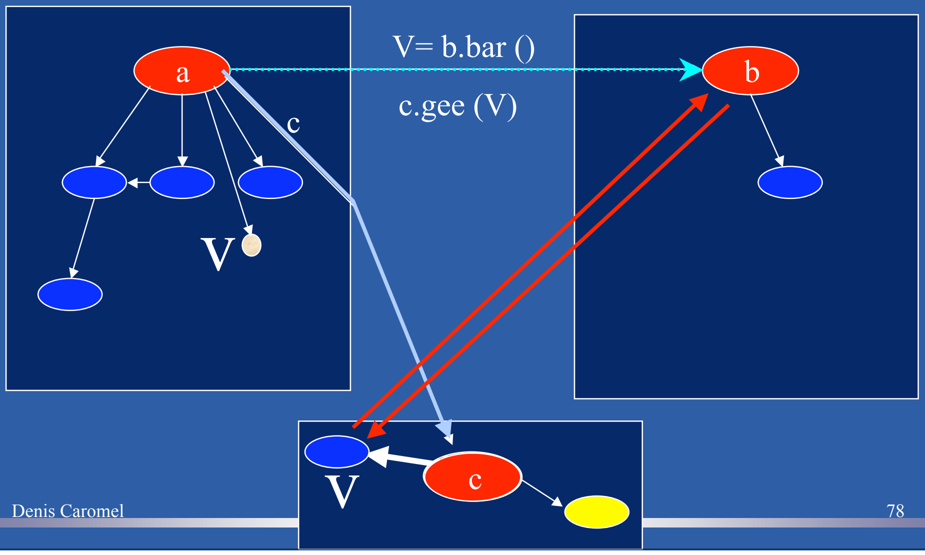
Wait-By-Necessity: Eager Message Based

AO forwarding a future: send a message



Wait-By-Necessity: Lazy Strategy

An Active Object requests a Future Value when needed



V= b.bar ()
c.gee (V)
a
c
b
V
V
c