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 14th, 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
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
• Written in Java
• Features:
  1. Eclipse GUI
  3. Scheduling & Grids

Used in production by industry
**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

Featuring: graphical user interface, resource acquisition and virtualization.
ProActive Parallel Suite (1)
ProActive Parallel Suite: GUI
ProActive Parallel Suite: GUI
### IC2D

#### Domain#5
- **Total**: 142212.28 ms, 100.00%
- **SendReply**: 0.00 ms, 0.00%
- **GroupOneWayCall**: 0.00 ms, 0.00%
- **WaitForNecessity**: 17858.33 ms, 12.50%
- **SendRequest**: 101773.58 ms, 71.56%

**Invocation Counts**
- **SendReply**: 1
- **GroupOneWayCall**: 0
- **GroupAsyncCall**: 0
- **WaitForNecessity**: 1
- **SendRequest**: 1

#### Domain#4
- **Total**: 142228.17 ms, 100.00%
- **SendReply**: 0.00 ms, 0.00%
- **GroupOneWayCall**: 0.00 ms, 0.00%
- **GroupAsyncCall**: 0.00 ms, 0.00%
- **WaitForNecessity**: 16765.29 ms, 11.79%
- **SendRequest**: 103230.24 ms, 71.94%

**Invocation Counts**
- **SendReply**: 1
- **GroupOneWayCall**: 0
- **GroupAsyncCall**: 0
- **WaitForNecessity**: 0
- **SendRequest**: 1

#### Domain#3
- **Total**: 142228.17 ms, 100.00%
- **SendReply**: 0.00 ms, 0.00%
- **GroupOneWayCall**: 0.00 ms, 0.00%
- **GroupAsyncCall**: 0.00 ms, 0.00%
- **WaitForNecessity**: 14847.59 ms, 10.43%
- **SendRequest**: 103829.01 ms, 72.41%

**Invocation Counts**
- **SendReply**: 1
- **GroupOneWayCall**: 0
- **GroupAsyncCall**: 0
- **WaitForNecessity**: 0
- **SendRequest**: 1

---

**ProActive Parallel Suite**

12  Denis Caromel
Video 1: IC2D
Monitoring, Debugging, Optimizing
ProActive Parallel Suite: Deploy
ProActive Parallel Suite: Deploy
Deploy on Various Kinds of Infrastructures

- Internet
- Servlets
- EJBs
- Databases

- Large Equipment
- Clusters
- Parallel Machine

Job management for embarrassingly parallel application (e.g. SETI)
Scheduler and Resource Manager: User Interface
Scheduler: User Interface
Video 2: Scheduler, Resource Manager
ProActive Parallel Suite: Program
ProActive Parallel Suite
ProActive Parallel Suite: Program
ProActive Parallel Suite: Program

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

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

[Diagram of JVM and Active Objects]

- Java Object
- Active Object
- Future Object
- Proxy
- Req. Queue
- Thread
- Wait-By-Necessity
ProActive: Inter- to Intra- Synchronization

Sequential  Multithreaded  Distributed

Synchronizations, Behavior: not dependent upon the physical location (mapping of activities)
ProActive: First-Class Futures

Sequential

Multithreaded

Distributed

Synchronizations, Behavior: not dependent upon the physical location (mapping of activities)
Wait-By-Necessity: First Class Futures

Futures are Global Single-Assignment Variables

\[
V = b.\overline{\text{bar}}()
\]

\[
c.\text{gee}(V)
\]
Standard system at Runtime:
No Sharing

NoC: Network On Chip

Proofs of Determinism
Calculus

ASP:
Asynchronous Sequential Processes
Proofs in GREEK

\[(a, \sigma) \rightarrow_\Sigma (a', \sigma')\]
\[\alpha[a; \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[a'; \sigma'; \iota; F; R; f] \parallel P \] (LOCAL)

\[\text{γ fresh activity } \iota' \notin \text{dom}(\sigma) \quad \sigma' = \{\iota' \mapsto AO(\gamma)\} :: \sigma \]
\[\sigma_\gamma = \text{copy}(\iota'', \sigma) \quad \text{Service} = (\text{if } m_j = \emptyset \text{ then } \text{FifoService} \text{ else } \iota''.m_j())\]

\[\alpha[\mathcal{R}[[\text{Active}(\iota'', m_j)]; \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[\mathcal{R}[\iota']; \sigma'; \iota; F; R; f] \parallel \gamma[\text{Service}; \sigma_\gamma; \iota''; \emptyset; \emptyset; \emptyset] \parallel P \]

\[\sigma_\alpha(\iota) = AO(\beta) \quad \iota'' \notin \text{dom}(\sigma_\beta) \quad F_i^{\alpha \rightarrow \beta} \text{ new future } \iota_f \notin \text{dom}(\sigma_\alpha) \]
\[\sigma'_\beta = \text{CopyMerge}(\sigma_\alpha; \iota' ; \sigma_\beta; \iota'') \quad \sigma'_\alpha = \{\iota_f \mapsto \text{ful}(\iota_i^{\alpha \rightarrow \beta})\} :: \sigma_\alpha \quad \text{(REGEN)}\]

\[\alpha[\mathcal{R}[\iota.m_j(\iota')]; \sigma; \iota; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow \alpha[\mathcal{R}[\iota_f]; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma'_\beta; \iota_\beta; F_\beta; R_\beta :: [m_j; \iota''; F_i^{\alpha \rightarrow \beta}; f_\beta]] \parallel P \]

\[R = R' :: [m_j; \iota_r; f'] :: R'' \quad m_j \in M \quad \forall m \in M, m \notin R' \]
\[\alpha[\mathcal{R}[[\text{Server}(M)]; \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[\mathcal{E}[m_j(\iota_r) \uparrow f, \mathcal{R}[]]; \sigma; \iota; F; R' :: R''; f'] \parallel P \] (ENDS)

\[\iota' \notin \text{dom}(\sigma) \quad \text{F} = F :: \{f \mapsto \iota'\} \quad \sigma' = \text{CopyMerge}(\sigma; \iota ; \sigma', \iota') \]
\[\alpha[\iota \uparrow (f', a); \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[a; \sigma'; \iota; F'; R; f'] \parallel P \] (ENDS)

\[\sigma_\alpha(\iota) = \text{ful}(\iota_i^{\alpha \rightarrow \beta}) \quad \sigma'_\beta(f_i^{\alpha \rightarrow \beta}) = \iota_f \quad \sigma'_\alpha = \text{CopyMerge}(\sigma_\beta; \iota_f ; \sigma_\alpha; \iota) \quad \text{(ENDS)}\]

\[\alpha[a_\alpha; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow \alpha[a_\alpha; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \]

ASP ⇒ Confluence and Determinacy

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

Preface by Luca Cardelli

Denis Caromel

Ludovic Henrio
TYPED

ASYNCRONOUS

GROUPS
Creating AO and Groups

- A \( ag = \text{newActiveGroup} \) ("A", [...], VirtualNode)
- V v = ag.foo(param);
- ...

JVM v.bar(); //Wait-by-necessity

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

Typed Group Java or Active Object
Broadcast and Scatter

Broadcast is the default behavior
Use a group as parameter, Scattered depends on rankings

```java
ag.bar(cg);  // broadcast cg
ProActive.setScatterGroup(cg);
ag.bar(cg);  // scatter cg
```
Dynamic Dispatch Group

```plaintext
ag.bar(cg);
```

![Diagram showing dynamic dispatch group with JVMs and classes](image-url)
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)
Objects to Distributed Components (1)

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

IoC: Inversion Of Control (set in XML)

Example of component instance

Truly Distributed Components

Typed Group

Java or Active Object

JVM
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
Scopes and Objectives:

Grid Codes that Compose and Deploy

No programming, No Scripting, …

Innovation:

Abstract Deployment

Composite Components

Multicast and GatherCast

invocation parameters

M components

N components

MultiCast

GatherCast

ProActive Parallel Suite
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

Internet

Clusters

Servlets EJBs Databases

Large Equipment

Parallel Machine

Job management for embarrassingly parallel application (e.g. SETI)
Protocols and Scheduler in GCM Deployment

**Protocols:**
- rsh
- ssh
- Oarsh
- Gsissh

**Scheduler, and Grids:**
- GroupSSH, GroupRSH, GroupOARSH
- ARC (NorduGrid), CGSP China Grid, EEGE 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 Naouumenko),
   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

Application

Developpement of artificial creatures

Version ProActive

<table>
<thead>
<tr>
<th>Initial Application</th>
<th>1 PC</th>
<th>56h52 =&gt; Crash!</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProActive Version</td>
<td>300 CPUs</td>
<td>19 minutes</td>
</tr>
</tbody>
</table>
Artificial Life Generation

Sylvain Cussat-Blanc, Yves Duthen – IRIT TOULOUSE
JECS : 3D Electromagnetism
Radar Reflection on Planes
Code Coupling:
Vibro Acoustic (courtesy of EADS)
NAS Parallel Benchmarks

Designed by NASA to evaluate benefits of high performance systems

Strongly based on CFD

5 benchmarks (kernels) to test different aspects of a system

2 categories: communication intensive and computation intensive

Kernel CG on class C

Desig sys.

Strong

5 ben

2 cate int

number of nodes
Enterprise IT: Software Tests

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

Basic Local Alignment Search Tool for rapid sequence comparison
BLAST developed by NCBI (National Center for Biotechnology Information)

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

![Graph showing speedup vs number of nodes]
Monte Carlo Simulations, Non-Linear Physics, INLN
Matlab and Scilab Grid Interface
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
SCALE BEYOND LIMITS
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:

A Toolkit for Acceleration:

Multi-Core & Distributed
Open the way to Soft.+Serv. EU Industry with Clouds & Utilities, DAAS
Multi-Active Object in 1 Address Space for Multi-cores
Welcome to the Open Source World of ProActive!

**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. somms, 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**

  - Featuring: graphical user interface, resource acquisition and virtualization.


The **Grid Application Server for the Enterprise**!
RAMMING

Multi-Core, Enterprise clouds.
sync. comms, cluster, Monte-Carlo, components code wrapping.

OPTIMIZING

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

Featuring: graphical monitoring and benchmarking with report generation.

SCHEDULING


Featuring: graphical interface, resource allocation 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)
Summary-Perspective: Comprehensive Toolkit

Programming: Models & Tools

Parallel: Multi-Core & Distributed