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 4000 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
Located in Sophia Antipolis, between Nice and Cannes, Visitors and Students Welcome!
Startup Company Born of INRIA

Co-developing, Support for ProActive Parallel Suite

Worldwide Customers: Fr, UK, Boston USA
Multi-Cores
Symetrical 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
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

Leading Open Source Middleware
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. HUAI 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).
Product: ProActive Parallel Suite

Java Parallel Toolkit
Multi-Platform Job Scheduler
Resource Manager

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
ProActive Contributors

Abhijeet Gaikwad (Option Pricing)
Abhishek-Rajeev Gupta
Antonio Cansado
Baptiste De Stefano
Bastien Sauvan
Brian Amedro (SPMD)
Cédric Dalmasso (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 Bodnarchouk (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 Ranaldo (Core, Deployment)
Romain Quilici
ProActive Programming

ProActive Programming
Java Parallel Toolkit

CORE
- Mobility
- Asynchrony
- Remote Accessibility

ACTIVE OBJECTS

FILE TRANSFER
- Web Services
- Security
- Fault Tolerance

MASTER WORKER
- Monte Carlo
- Legacy Wrapping
- SPMD

TASKFLOW
- File Split & Merge
- Data Spaces
- GCM

Activeeon
Scale Beyond Limits

ProActive
Parallel Suite

OW2 Consortium
ProActive : Active objects

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

... v1.bar();  //Wait-By-Necessity

JVM

Wait-By-Necessity is a Dataflow Synchronization

Java Object  Active Object  Req. Queue
Future Object  Proxy  Thread

ProActive
Parallel Suite

Activeeon
SCALE BEYOND LIMITS

OW2 Consortium
Standard system at Runtime: No Sharing

NoC: Network On Chip

Proofs of Determinism
ASP: Asynchronous Sequential Processes

\[ (a, \sigma) \rightarrow_S (a', \sigma') \]

\[ \alpha[a; \sigma; \nu; F; R; f] \parallel P \longrightarrow \alpha[a'; \sigma'; \nu; F; R; f] \parallel P \]

\[ (\text{LOCAL}) \]

\[ \gamma \text{ fresh activity } \nu' \notin \text{dom}(\sigma) \quad \sigma' = \{ \nu' \mapsto AO(\gamma) \} \cdot \sigma \]

\[ \sigma_{\gamma} = \text{copy}(\nu'', \sigma) \quad \text{Service} = (\text{if } m_j = \emptyset \text{ then } \text{FifoService} \text{ else } \nu''.m_j()) \]

\[ (\text{NEWACT}) \]

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

\[ \sigma_{\nu}(\nu) = AO(\beta) \quad \nu'' \notin \text{dom}(\sigma_{\beta}) \quad f_{i}^{\alpha \rightarrow \beta} \text{ new future } \quad \nu_f \notin \text{dom}(\sigma_{\alpha}) \]

\[ \sigma'_{\beta} = \text{CopyMerge}(\sigma_{\alpha}, \nu'; \sigma_{\beta}, \nu'') \quad \sigma'_{\alpha} = \{ \nu_f \mapsto \text{fut}(f_{i}^{\alpha \rightarrow \beta}) \} \cdot \sigma_{\alpha} \]

\[ (\text{REQUEST}) \]

\[ \alpha[\mathcal{R}[\nu.m_j(\nu')]; \sigma_{\alpha}; \nu; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta[a_{\beta}; \sigma_{\beta}; \nu_{\beta}; F_{\beta}; R_{\beta}; f_{\beta}] \parallel P \rightarrow \]

\[ \alpha[\mathcal{R}[\nu_f]; \sigma'_{\alpha}; \nu; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta[a_{\beta}; \sigma'_{\beta}; \nu_{\beta}; F_{\beta}; R_{\beta}; m_j; \nu''; f_{i}^{\alpha \rightarrow \beta}; f_{\beta}] \parallel P \]

\[ (\text{REQUEST}) \]

\[ R = R' :: [m_j; \nu_f; f_f] :: R'' \quad m_j \in M \quad \forall m \in M, m \notin R' \]

\[ \alpha[\mathcal{R}[\text{Serve}(M)]; \sigma; \nu; F; R; f] \parallel P \longrightarrow \alpha[\nu.m_j(\nu_f) \mapsto f, \mathcal{R}([]); \sigma; \nu; F; R' \mapsto R''; f'] \parallel P \]

\[ (\text{SERVE}) \]

\[ \nu' \notin \text{dom}(\sigma) \quad F'' = F :: \{ \nu \mapsto \nu' \} \quad \sigma' = \text{CopyMerge}(\sigma; \nu; \sigma; \nu') \]

\[ (\text{ENDSERVICE}) \]

\[ \alpha[\nu \mapsto (f', \alpha); \sigma; \nu; F; R; f] \parallel P \longrightarrow \alpha[\alpha[a_{\alpha}; \sigma; \nu; F; R; f'] \parallel P \]

\[ \sigma_{\alpha}(\nu) = \text{fut}(f_{i}^{\gamma \rightarrow \beta}) \quad F_{\beta}(f_{i}^{\gamma \rightarrow \beta}) = \nu_f \quad \sigma'_{\alpha} = \text{CopyMerge}(\sigma_{\beta}, \nu_f; \sigma_{\alpha}, \nu) \]

\[ (\text{REPLY}) \]

\[ \alpha[a_{\alpha}; \sigma_{\alpha}; \nu; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta[a_{\beta}; \sigma_{\beta}; \nu_{\beta}; F_{\beta}; R_{\beta}; f_{\beta}] \parallel P \rightarrow \]

\[ \alpha[a_{\alpha}; \sigma'_{\alpha}; \nu; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta[a_{\beta}; \sigma_{\beta}; \nu_{\beta}; F_{\beta}; R_{\beta}; f_{\beta}] \parallel P \]
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

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

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
Objects to Distributed Components

IoC: Inversion Of Control (set in XML)

Typed Group

Java or Active Object

Truly Distributed Components

Example of component instance

JVM
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:**
  
  KAAPI, 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 Antipolis: 500 (434)
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, EEGE 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
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

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

- **Clouds:**
  - Amazon EC2

Interoperability:
Cloud will start with existing IT infrastructure,
Build Non Intrusive Cloud with ProActive
IC2D: Optimizing
Video 1: IC2D Optimizing Monitoring, Debugging, Optimizing
Scheduling & Resourcing
ProActive Scheduling

Multi-Platform Job Scheduler

ProActive Scheduling

ProActive Scheduling

Activeeon
SCALE BEYOND LIMITS

ProActive Parallel Suite

OW2 Consortium
ProActive Scheduling Big Picture

RESOURCES

- Multi-platform Graphical Client (RCP)
- File-based or LDAP authentication
- Static Workflow Job Scheduling, Native and Java tasks, Retry on Error, Priority Policy, Configuration Scripts,…
- Dynamic and Static node sources, Resource Selection by script, Monitoring and Control GUI,…
- ProActive Deployment capabilities: Desktops, Clusters, Clouds,…

ProActive Scheduler

ProActive Resource Manager
Another Example: Picture Denoising

• with selection on native executable availability (ImageMagick, GREYstoration)
  • Multi-platform selection and command generation
• with file transfer in pre/post scripts
RESOURCING User Interface

ProActive Resource Manager

Tab Explorer

Compact View

Activity History

Node States Peaks

Free Nodes History

Overview

Charts

Statistics

Info

State: aggregate

- # free nodes: 272
- # busy nodes: 52
- # down nodes: 6

connected
Video 2: Scheduler, Resource Manager
Clusters to Grids to Clouds:
e.g. on Amazon EC2
Node source Use case: Configuration for external cloud with EC2

- **ProActive Scheduler**
- **ProActive Resource Manager**
  - **Static Policy**
    - LSF
  - **Timing Policy 12/24**
    - Desktops
  - **Dynamic Workload Policy**
    - EC2

- Dedicated resources
- Desktops
- Amazon EC2
ProActive Parallel Suite

Three fully compatible modules

Programming → Scheduling

Scheduling

Resource Management

Clutch Power:
Solid Building Blocks for Flexible Solutions

Activeeon
SCALE BEYOND LIMITS
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

Nodes can be dynamically added!

ProActive Parallel Suite

Cluster

PBS

Desksops

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

Web Interface

ProActive Scheduler + Resource Manager

CPU nodes

Amazon EC2

Noised video file

GPU nodes
Cloud Seeding with ProActive

User submit its noised video to the web interface
Cloud Seeding with ProActive

User

Web Interface

ProActive Scheduler + Resource Manager

Amazon EC2

CPU nodes

GPU nodes

Web Server submit a denoising job to the ProActive Scheduler
Cloud Seeding with ProActive

User

Web Interface

CPU nodes

ProActive Scheduler + Resource Manager

Amazon EC2

GPU nodes

CPU nodes are used to split the video into smaller ones
CPU nodes are used to split the video into smaller ones.
Cloud Seeding with ProActive

GPU nodes are responsible to denoise these small videos
Cloud Seeding with ProActive

GPU nodes are responsible to denoise these small videos

User

Web Interface

ProActive Scheduler + Resource Manager

Amazon EC2

CPU nodes

GPU nodes
Cloud Seeding with ProActive

User

Web Interface

ProActive Scheduler
+ Resource Manager

CPU nodes

Amazon EC2

GPU nodes

CPU nodes merge the denoised video parts
Cloud Seeding with ProActive

- User
- Web Interface
- ProActive Scheduler + Resource Manager
- Amazon EC2
- CPU nodes
- GPU nodes

CPU nodes merge the denoised video parts
Cloud Seeding with ProActive

User → Web Interface → ProActive Scheduler + Resource Manager → Amazon EC2 → CPU nodes

The final denoised video is sent back to the user

GPU nodes
Conclusion
Conclusion

- **Flexibility**
  - Clutch Power

- **Portability:**
  - Windows, Linux, Mac

- **Versatility:**
  - Desktops, Grids, Clouds

---

**Free Professional Open Source Software**

---

**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

Let's 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

- Business Intelligence
- BI Monitoring
- Parallel Programming
  - SPMD, workflow
  - Agent, Master/Worker
  - Fork and Join
  - In memory db cache
    (JSR / JPI / javaspaces)
- Task & Services Scheduling
- Resource Manager
- OS Virtualization
- OS, HW

Service Level Management

SLM

SOA Monitoring
  - Reporting, Notifications, alarms

SOA BPEL Exec
  - Repository, Registry, Orchestration

SCA
  - Service Component Architecture

ESB
  - Enterprise Service Bus

Grid Utility interface

OS, HW
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

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