ProActive Hybrid Workflows with CPUs and GPUs and various Use Cases

D. Caromel, et al.

Agenda

1. Background: INRIA, ActiveEon
2. Multi-Core and Virtualization
3. ProActive Parallel Suite
   Programming, Scheduling, Resourcing
4. Use Cases & Demos (PACA Grid)
5. Conclusion

Cloud Computing Revolution ?
Workflow Execution
Studio Editor and Visualization
Parallel Programming in Java

Portal, Multi-Application & Multi-Tenant
Enterprise Orchestration

Physical and Virtual Machines Management

GPU nodes
INRIA 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, Conicyt)
- Brian Amedro (SCS-Agos)
- Cristian Ruz (INRIA, Conicyt)
- Elton Mathias (INRIA-Cordi)
- Imen Filali (SCS-Agos / FP7 SOA4All)
- Marcela Rivera (INRIA, Conicyt)
- Muhammad Khan (STIC-Asia)
- Paul Naoumenko (INRIA/Région PACA)
- Viet Dung Doan (FP6 Bionets)
- Virginie Contes (SOA4ALL)
- Guilherme Pezzi (AGOS, CIFRE SCP)

Visitors + Interns
- PostDoc (1):
  - Regis Gascon (INRIA)
- Engineers (10):
  - Elaine Isnard (AGOS)
  - Fabien Viale (ANR OMD2, Renault)
  - Franca Perrina (AGOS)
  - Germain Sigety (INRIA)
  - Yu Feng (ETSI, FP6 EchoGrid)
  - Bastien Sauvan (ADT Galaxy)
  - Florin Alexandru Bratu (INRIA CPER)
  - Igor Smirnov (Microsoft)
  - Fabrice Fontenoy (AGOS)
  - Open position (Thales)
- Trainee (2):
  - Etienne Vallette d'Osia (Master 2 ISI)
  - Laurent Vanni (Master 2 ISI)
- Assistants (2):
  - Patricia Maleyran (INRIA)
  - Sandra Devauchelle (I3S)

Located in Sophia Antipolis, between Nice and Cannes, Visitors Welcome!
OASIS Team & INRIA

- A joint team, about 35 persons
- 2004: First ProActive User Group
- 2011, July: ProActive 5.1, Distributed & Parallel: From Multi-cores to Enterprise GRIDs & Clouds
- Co-developing, Support for ProActive Parallel Suite
- Worldwide Customers: Fr, UK, Boston USA
ActiveEon Overview


- Co developing with INRIA **ProActive Parallel Suite®**, a Professional Open Source middleware for parallel, distributed, multi-core computing. 30 peoples in total.

- Core mission: Scale Beyond Limits.

- Providing a full range of services for ProActive Parallel Suite.

- Worldwide customers and production users:
Multi-Core Push
Symmetrical Multi-Core: 8-ways Niagara II

- 8 cores
- 4 Native threads per core
- Linux see 32 cores!
Off The Shelf Multi-Cores, 3 GHz

Intel Xeon 5670, 6 cores, 10, … 12

AMD's Opteron 6174, “Magny-Cours”, 12 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)
  - 2012: 32 to 64
  - 2013: 64 to 128
  - 2015: 128 to 256

  1 Million Cores Parallel Machines in 2014
  100 M cores coming in 2020

- Multi-Cores are NUMA, and turning Heterogeneous (GPU)
  They are turning into SoC with NoC: NOT SMP!
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
Virtualization
Virtualization

Source: http://www.apac.redhat.com
Virtualization

Sun, Blog Marc Hamilton
Virtualization
What we Used to do as Syst. Admin.
With Virtualization + Software Appliance
ProActive Parallel Suite
ProActive Parallel Suite

- Professional Open Source middleware for parallel, distributed, multi-core, Grid and Cloud computing

- Composed of three modules:
  - Java Programming Library
  - Multiplatform Job scheduler
  - Global resource manager
ProActive Parallel Suite

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

Core API
- Active Objects
- Asynchrony
- Futures
- Groups
- Mobile Agents
- MOP / AOP
ProActive Programming:
Active Objects
ProActive Programming View
ProActive Programming View

GPU nodes
A ag = newActive ("A", [...], VirtualNode)

V v1 = ag.foo (param);
V v2 = ag.bar (param);
...

v1.bar(); //Wait-By-Necessity
NoC: Network On Chip

Proofs of Determinism
ASP: Asynchronous Sequential Processes

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

\[\alpha[a; \sigma; \mathfrak{f}; F; R; f] || P \rightarrow \alpha[a'; \sigma'; \mathfrak{f}; F; R; f] || P \quad \text{LOCAL}\]

\[\gamma \text{ fresh activity} \quad \mathfrak{f} \not\in \text{dom}(\sigma) \quad \sigma' = \{ \mathfrak{f} \mapsto \text{AO}(\gamma) \} :: \sigma \]

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

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

\[\sigma_{\alpha}(\mathfrak{f}) = \text{AO}(\beta) \quad \mathfrak{f}'' \not\in \text{dom}(\sigma_{\beta}) \quad f_{i}^{\alpha} \mapsto \beta \text{ new future} \quad \mathfrak{f}_f \not\in \text{dom}(\sigma_{\alpha}) \]

\[\sigma'_{\beta} = \text{Copy}&\text{Merge}(\sigma_{\alpha}, \mathfrak{f}''; \sigma_{\beta}, \mathfrak{f}_f) \quad \sigma'_{\alpha} = \{ \mathfrak{f}_f \mapsto \text{fut}(f_{i}^{\alpha} \mapsto \beta) \} :: \sigma_{\alpha} \quad \text{REQUEST}\]

\[\alpha[\mathcal{R}[\mathfrak{f}.m_j(\mathfrak{f}'')]; \sigma_{\beta}; \mathfrak{f}; F_{\alpha}; R_{\alpha}; f_{\alpha}] || \beta[a_{\beta}; \sigma_{\beta}; \mathfrak{f}_f; F_{\beta}; R_{\beta}; f_{\beta}] || P \rightarrow \alpha[\mathcal{R}[\mathfrak{f}'']; \sigma'_{\alpha}; \mathfrak{f}; F_{\alpha}; R_{\alpha}; f_{\alpha}] || \beta[a_{\beta}; \sigma'_{\beta}; \mathfrak{f}_f; F_{\beta}; R_{\beta}; \{m_j; \mathfrak{f}''; f_{i}^{\alpha} \mapsto \beta\}; f_{\beta}] || P\]

\[\text{R} = R' :: [m_j; \mathfrak{f}; f'] :: R'' \quad m_j \in M \quad \forall m \in M, m \not\in R' \quad \text{SERVE}\]

\[\alpha[\mathcal{R}[\text{Serve}(M)]; \sigma; \mathfrak{f}; F; R; f] || P \rightarrow \alpha[\mathfrak{f}.m_j(\mathfrak{f}_r) \uparrow f, \mathcal{R}[]]; \sigma; \mathfrak{f}; F; R'; :: R''; f''] || P\]

\[\mathfrak{f}' \not\in \text{dom}(\sigma) \quad F'' = F :: \{ f \mapsto \mathfrak{f}' \} \quad \sigma' = \text{Copy}&\text{Merge}(\sigma, \mathfrak{f}''; \sigma, \mathfrak{f}') \quad \text{ENDSERVICE}\]

\[\alpha[\mathfrak{f} \uparrow (f'', \alpha); \sigma; \mathfrak{f}; F; R; f] || P \rightarrow \alpha[a; \sigma'; \mathfrak{f}; F''; R; f'] || P\]

\[\sigma_{\alpha}(\mathfrak{f}) = \text{fut}(f_{i}^{\gamma} \mapsto \beta) \quad F_{\beta}(f_{i}^{\gamma} \mapsto \beta) = \mathfrak{f}_f \quad \sigma'_{\alpha} = \text{Copy}&\text{Merge}(\sigma_{\beta}, \mathfrak{f}_f; \sigma_{\alpha}, \mathfrak{f}) \quad \text{REPLY}\]

\[\alpha[a_{\alpha}; \sigma_{\alpha}; \mathfrak{f}; F_{\alpha}; R_{\alpha}; f_{\alpha}] || \beta[a_{\beta}; \sigma_{\beta}; \mathfrak{f}_f; F_{\beta}; R_{\beta}; f_{\beta}] || P \rightarrow \alpha[a_{\alpha}; \sigma'_{\alpha}; \mathfrak{f}; F_{\alpha}; R_{\alpha}; f_{\alpha}] || \beta[a_{\beta}; \sigma_{\beta}; \mathfrak{f}_f; F_{\beta}; R_{\beta}; f_{\beta}] || P\]
Distributed Objects On Chip
Distributed Objects On Chip, Boards, Clouds
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

```
g ag

c0 c2 c4 c6 c8

c1 c3 c5 c7 c9

JVM

ag.bar(cg);`
Object-Oriented SPMD
“MPI and programming languages from the 60’s will not make it”

Jack Dongarra, 2/13/2009,
Wake Forest University talk

“It is time to get ride of MPI”

Alan Edelman, MIT, 06/16/2010,
ScilabTec’10 Users’ Day
A ag = newSPMDGroup ("A", [...], VirtualNode)
// In each member
  myGroup.barrier ("2D"); // Global Barrier
  myGroup.barrier ("vertical"); // Any Barrier
  myGroup.barrier ("north","south","east","west");
NAS Parallel Benchmarks

- Experimented on 3D ElectroMagnetism, and Nasa 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 or focus variations:
  - communication intensive and computation intensive
Communication Intensive
CG Kernel (Conjugate Gradient)

- Floating point operations
- Eigen value computation
- High number of unstructured communications

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

Message density distribution
Data density distribution
Communication Intensive
CG Kernel (Conjugate Gradient)

Kernel CG on class C

Comparable Performances
Scheduling and Workflow Orchestration
ProActive Scheduling
Job Scheduling

Optimize:
- workload distribution
- computing resources usage
- software licences

Straightforward batch, jobs and workflow construction

Any kind of jobs supported:
- Java, Applications and Scripts

Data management with automatic transfer
ProActive Scheduling Big Picture

- 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,…

### Jobs

<table>
<thead>
<tr>
<th>Id</th>
<th>State</th>
<th>User</th>
<th>Priority</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>1997</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>1998</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>1999</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2000</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2001</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2002</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2003</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2004</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2005</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2006</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2007</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>2008</td>
<td>Pending</td>
<td>j1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
</tbody>
</table>

### Running

<table>
<thead>
<tr>
<th>Id</th>
<th>State</th>
<th>Progress</th>
<th># Finished</th>
<th>User</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1313</td>
<td>Running</td>
<td></td>
<td>4/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1314</td>
<td>Running</td>
<td></td>
<td>4/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1315</td>
<td>Running</td>
<td></td>
<td>7/8</td>
<td>admin</td>
<td>Normal</td>
</tr>
<tr>
<td>1316</td>
<td>Running</td>
<td></td>
<td>4/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1317</td>
<td>Running</td>
<td></td>
<td>7/8</td>
<td>admin</td>
<td>Normal</td>
</tr>
<tr>
<td>1318</td>
<td>Running</td>
<td></td>
<td>4/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1319</td>
<td>Running</td>
<td></td>
<td>7/8</td>
<td>admin</td>
<td>Normal</td>
</tr>
<tr>
<td>1320</td>
<td>Running</td>
<td></td>
<td>3/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1321</td>
<td>Running</td>
<td></td>
<td>7/8</td>
<td>admin</td>
<td>Normal</td>
</tr>
<tr>
<td>1322</td>
<td>Running</td>
<td></td>
<td>3/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1323</td>
<td>Running</td>
<td></td>
<td>7/8</td>
<td>admin</td>
<td>Normal</td>
</tr>
<tr>
<td>1324</td>
<td>Running</td>
<td></td>
<td>2/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1325</td>
<td>Running</td>
<td></td>
<td>2/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1326</td>
<td>Running</td>
<td></td>
<td>2/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
<tr>
<td>1327</td>
<td>Running</td>
<td></td>
<td>2/8</td>
<td>user1</td>
<td>Normal</td>
</tr>
</tbody>
</table>

### Finished

<table>
<thead>
<tr>
<th>Id</th>
<th>State</th>
<th>User</th>
<th>Priority</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>008</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>005</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>001</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>006</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>004</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>003</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>009</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>007</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>002</td>
<td>Finished</td>
<td>j1</td>
<td>Low</td>
<td>job_proActive</td>
</tr>
<tr>
<td>245</td>
<td>Finished</td>
<td>user1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>246</td>
<td>Finished</td>
<td>user1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>247</td>
<td>Finished</td>
<td>user1</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>252</td>
<td>Finished</td>
<td>adm1r</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
<tr>
<td>253</td>
<td>Finished</td>
<td>adm1r</td>
<td>Normal</td>
<td>job_with_dep</td>
</tr>
</tbody>
</table>

---

**ProActive Scheduler**

**ProActive Resource Manager**

---

**Activeeon**

SCALE BEYOND LIMITS

---

**ProActive Parallel Suite**
- Provides **highly configurable scheduling policies**
- **Self-healing** with automatic restart from latest valid point
- **Hardware and software fault tolerant mechanism** for task execution
- Graphical interface and command line client
- **Seamless integration** with third-party application
  - Java, Web Service and C/C++
- Accounting per user
Workflow Example: Picture Denoising

- with **selection** on native executable availability (ImageMagik, GREYstoration)
  - Multi-platform selection and command generation
- with **file transfer** in pre/post scripts
Studio Demonstration
Heterogeneous Resource Management
ProActive Resourcing

ProActive Resource Manager

ProActive Resourcing
Desktop, Cluster, Grid & Cloud Resource Manager

ProActive Parallel Suite
ProActive Resourcing

- **Virtualizes** organizations’ existing infrastructure for a heightened computing power

- Enables **business driven** computing resources acquisitions
  - Elastic computing platform

- Reaps the benefits from **Clouds**, e.g. Amazon EC2, and latest hardware architecture

- **Aggregates** and leverages any type of resources

- Manage your **virtual machines**
  - VMWare, Xen, KVM, Hyper-V, ...

- Accounting per resource provider
Resource Manager GUI
Topology with the ProActive Resourcing
Private, Public & Hybrid Clouds

Enterprise Private Hybrid Cloud

Static Policy

LSF

Timing Policy 12/24

Desktops

Dynamic Workload Policy

On Burst

Application and Workflow Acceleration

Dedicated resources

On premise

External Data Center, Amazon EC2, Azure
ProActive: Security for your Enterprise Cloud platform

- User Authentication (LDAP or Files)
- Authentication and encryptions of network communications using SSH, SSL and PKI
- Secure communication routing through Firewall and NAT configuration of LAN network
Use Cases and Demonstration on a Production Platform
The ProActive PACA Grid Platform (4)

Total:

- 1,368 Cores
- 480 CUDA Cores
- 30TB Storage

Publically Available Today for Production
Use Case 1:
CPU + GPU Workflow
Workflow ProActive for CPU and GPU
Live Demo
CPU + GPU ProActive Workflows

- Resource selection for each Task of a ProActive Workflow
- Selection of Host with GPU capacity
- Data Transfer to the GPU Host
- Configuration of GPU Capacity at the level of Admin (Number of GPU Nodes, size)
- Freedom to request one or several GPU capacities for one GPU program
- Global Scheduling (Multi-Tenant, Multi-Application) of GPU Tasks
Use Case 2: OMD2
Distributed Multi-Disciplinary Optimizations with Remote Visualization
Open Source Interfaces
For
Distributed Multi-Disciplinary Optimisations
OMD2: Open Source Interfaces For Distributed Multi-Disciplinaires Optimisations
Distributed Workflow

Acquisition des maillages, etc

Stratégie de résolution

Calculs f(x)

Mailleur

Solver

PO

WS

BD

Scheduler

Catia

Soumission du PO

Boucle optimisation

Ressources

N1

N2

Ni1

Nik

Visualisation stats, etc
Coupling Mechanics, Aerodynamics …

3D Air Conditionning

10min CPU

<1min CPU

2D Air Conditionning

100h CPU

1000h CPU

Cylinder Head

External Aerodynamic
Engineering Optimizations: Renault UC

PRE
- Resource Selection
- Dataspace Input
- Native Executable
- Dataspace Output

GEOM_AND_MESH
- Resource Selection
- Dataspace Input
- Native Executable
- Dataspace Output

SOLVE
- Resource Selection
- Dataspace Input
- Native Executable
- Dataspace Output

POST
- Resource Selection
- Dataspace Input
- Native Executable
- Dataspace Output

OpenFOAM
OpenCFD
Use Case 3: Map Reduce
ProActive MapReduce (CO, SP2, Task 2.1)

- Same APIs as Hadoop
  (Easy switch from Hadoop to ProActive)

- Does not require an HDFS File System

- Runs on general purpose, Multi-tenant, Multi-Applications Grids and Clouds

- Available as PaaS in Java
Workflow ProActive MapReduce
Map Reduce Demo
ProActive MapReduce vs. Hadoop+HDFS

<table>
<thead>
<tr>
<th>File Size</th>
<th>Sequential</th>
<th>Hadoop</th>
<th>PA MapReduce</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7 GB</td>
<td>5m 04s</td>
<td>1m 17s</td>
<td>1m 05s</td>
<td>4.6</td>
</tr>
<tr>
<td>4.3 GB</td>
<td>25m 31s</td>
<td>2m 30s</td>
<td>2m 20s</td>
<td>10.9</td>
</tr>
<tr>
<td>7.3 GB</td>
<td>46m 00s</td>
<td>3m 31s</td>
<td>3m 30s</td>
<td>13.1</td>
</tr>
<tr>
<td>20 GB</td>
<td>2h 07m 00s</td>
<td>8m 30s</td>
<td>7m 09s</td>
<td>17.8</td>
</tr>
<tr>
<td>50 GB</td>
<td>5h 19m 00s</td>
<td>21m 05s</td>
<td>25m 11s</td>
<td>12.7</td>
</tr>
<tr>
<td>100 GB</td>
<td>10h 38m 00s</td>
<td>43m 23s</td>
<td>58m 42s</td>
<td>10.9</td>
</tr>
</tbody>
</table>

- Data available in a NAS (General purpose storage)
- Transfer to HDFS for Hadoop
- Used directly without copy for ProActive
- Use Case of Map/Reduce on fresh data
- Different ProActive Map/Reduce configuration for recurrent MR on in place Data (e.g. ProActive HDFS interface)
Use Case 4: BioTechs
IPMC Use Case and Collaboration

SOLID machine from Applied Biosystems

Nodes can be dynamically added!

Cluster

ProActive Parallel Suite

Desksops

Clouds

ProActive Parallel Suite

Nodes: 16 nodes

EC2

PBS

Amazon Web Services™
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)
Use Case 5: Hydrodynamic with K-Epsilon and FineMarine
Hydrodynamic Optimization: Workflow generated from a GUI
Hydrodynamic Optimization: Execution
Hydrodynamic: Remote Steering during execution
UC 6: Acceleration of Financial Valuations

C++ library developed by Pricing Partners
Pricing solution dedicated to highly complex derivatives, Greek computation
How Does it Work?
Price-it Computing Distribution

- ProActive Scheduler
  - Regular Price-it Excel Interface
  - Automatic execution via job scheduler
  - Pool of shared resources

- Price-it Excel

- Price-it Excel

- Price-it Excel

- Price-it Excel
Accelerated Price-it Performances

Increased Productivity: Reduces Price-it Execution Time by 6 or more!

Use Case: Bermuda Vanilla, Model American MC

Test conditions:
- One computation is split in 130 tasks that are distributed
- Each task uses 300ko

![Graph showing execution time improvements with increasing nodes from 4 to 9 nodes. The graph shows more than 3 times faster with only 4 nodes and even 6 times faster with 9 nodes.](Image)
UC 7: IT
SOA Analysis of Web Server Logs
Parallel Services

- Separation: BPEL – Parallel Serv. – Task Flow
- Standards et Portable
- Flexibility

Diagram:

- High level Business Process
- Domain specific Service
- Other Basic Service
- Other Operational Service
- Operational Services

- Job Scheduling
- Parameter Sweeping
- Divide & Conquer

Scheduling of Taskflow Jobs
AGOS Platform Management

HP- Business Availability Center (HP-BAC)

- Monitoring of entire platform
- Cover all layers in the scope
- Monitoring dashboard and reports

Tasks Scheduler & Resources manager

- Integration with grid
- Indicator on running jobs
- Hypervisor & VM management
Integration with Scilab and Matlab
Integration with Scilab and Matlab

- Static Policy
  - LSF

- Timing Policy
  - 12/24
  - Desktops

- Dynamic Workload Policy
  - EC2
  - Amazon EC2

Dedicated resources

Desktops

ProActive Parallel Suite
Interface ProActive ⇔ Matlab
Integration with Scilab and Matlab

Static Policy
- LSF

Timing Policy 12/24
- Desktops

Dynamic Workload Policy
- EC2

Dedicated resources
- Desktops

Amazon EC2

ProActive Parallel Suite
Integration with Applications

Using your ProActive Cloud

Smooth integration of your codes!
Conclusion
Conclusion

- **Flexibility**
  - Clutch Power

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

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

---

Free Professional Open Source Software

[Java Parallel Toolkit](#)
[Multi-Platform Job Scheduler](#)
[Resource Manager](#)

---

**Multi-Core:** No sharing Parallel Programming Model

**Cloud:** Smooth transition needed: Consolidation + Interoperability
Physical and Virtual Machines Management
Portal, Multi-Application & Multi-Tenant
Enterprise Orchestration
Workflow Execution
Studio Editor and Visualization
Parallel Programming in Java
Conclusion

- **Portability:** Windows, Linux, Mac
- **Versatility:** Desktops, Grids, Clouds

- **Infrastructure Management:** Dynamic (Local, Remote, Public Cloud)
- **Application Acceleration:** APIs: Java, C++, Script, WS REST full

Free Professional Open Source Software
### Industrial (1750) & Cloud Revolution Compared

<table>
<thead>
<tr>
<th>Concept</th>
<th>Industrial Revolution</th>
<th>Cloud Revolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept</td>
<td>Mechanization and centralization of manufacturing activities</td>
<td>Computing as a Utility Centralization of Data Center, Automation</td>
</tr>
<tr>
<td>Technology</td>
<td>Supporting new technos (Mechanic, Tool Machines, etc.)</td>
<td>Distributed Computing Virtualization Multi-Cores Network</td>
</tr>
<tr>
<td>Socio Economical Factors</td>
<td>Large new demand was ready to use the new offer. (A change in business attitude &amp; organization)</td>
<td>IT Cost Reduction Pressure CIO Nightmare CEO Out-of-DataCenter CapEx</td>
</tr>
</tbody>
</table>

⇒ All elements converge for a strong Cloud Revolution

Sources & Inspiration: Simon Wardley (CSC) Scott Stewart
http://proactive.inria.fr
Thank you for your attention!