

ProActive Programming

ProActive Programming provides a powerful framework to easily develop **parallel and distributed** applications with **Java**. Focus on your application, ProActive will care about all parallel programming aspects!

It indeed hides parallel programming complexity to make it finally within everyone's reach!

Highlights

- Scales up your most demanding applications
- Simplifies the development of parallel and distributed applications
- Accelerates your business critical processes
- Provides an innovative toolkit meeting all your requirements

PROACTIVE PROGRAMMING LIBRARY

General features

Portability	ProActive Programming is a Java library. This confers to ProActive a very high portability. Java 1.5 and upper are supported
Fault-tolerance	Application can be configured to be self-healing and restart automatically from latest valid checkpoint: application completeness is ensured. Fault-tolerance properties are configurable
Security	Communications can be encrypted using a private/public key mechanism to ensure fully secure data transmissions. If needed, connections to all ProActive Programming entities can require authentication
Web Services	ProActive Programming Active Objects and GCM components can be exposed as web services without any additional development
OSGI	ProActive Programming can be used on top of the standardized OSGI framework (Open Services Gateway initiative) via the ProActive bundle. Thanks to a local ProActive active object, the framework becomes accessible from any Java application or OSGI framework
File transfer	Files can be transmitted from or to a remote node at different stage of the application lifecycle
Message Routing	ProActive Message Routing (PAMR) protocol enables communications with remote resources sitting behind a firewall or a NAT

Programming toolkit

High-Level and Domain-Specific APIs

These APIs enables you to quickly and easily integrate parallel and distributed computing in your applications

Master-Worker	This pattern suits for a wide range of parallel computing cases, especially embarrassingly parallel problems. It is made of one master that initiates the computation by creating a set of tasks and one or several workers which execute these tasks
Taskflow	Applications can be programmed as a flow of tasks triggered by data or temporal dependencies, and executed by <i>ProActive Scheduling</i>
OO SPMD	SPMD technique is ideal for solving tightly-coupled parallel problems requiring intensive communications (CFD, Maxwell,...). This is an alternative to MPI in Java
Monte-Carlo	Easy way to accelerate monte-carlo simulations in a distributed environment
Skeletons framework	This API provides a high level framework to successfully program structured parallel and distributed applications. It provides a comprehensive set of efficient structured patterns, the skeletons, which can be nested to create more complex patterns. Nine patterns are available to fulfill all requirements: farm, pipe, divide and conquer, map, fork, seq, for, while and if
Legacy Code Wrapping	Efficient methods to wrap, couple, and gridify legacy code with ProActive, including MPI or other native codes

Active Object API

This powerful API allows you to have a very precise and configurable integration of parallel and distributed aspects in your applications. An Active Object has its own thread and execution queue and can be manipulated like any standard Java object although they can be located on the same JVM or on a remote one. They feature the following functionalities:

Remote accessibility	Methods defined by active objects can be called from any point of the application. The underlying mechanism to process this call is transparent to the user
Asynchrony	Calls on Active Objects are asynchronous: an answer, a future, is provided without waiting for the end of the result processing. If needed, calls can be synchronous as well; execution is then blocked until the result is available. Although requests are served asynchronously, determinism characterizes the entire application
Mobility	Active Objects have the ability to migrate between nodes while the application is running. The application continues to run normally. This ensures a high flexibility in the hardware resource utilization
Object lifecycle control	Active Objects initialization, activity and end can be easily customized
Immediate service	Methods specified as immediate service are processed immediately and not queued like other requests: this allows having an immediate response to a method call
Behaviour integrity	Application functional behaviour remains the same wherever the processing is done: sending a call to a local Active Object is equivalent to sending one to a remote Active Object

Grid Component Model

Grid Component Model (GCM) API is based on and extends the Fractal model and provides a leading-edge modular approach for the development of efficient grid applications and a comprehensive framework to specify at the design stage the parallel and distributed parts of the applications. Distributed programming and collective communication are embedded within the GCM: developers just need to concentrate on implementing the business code.

Component model	Components are software modules, offering a range of services, that can be connected to each other and also nested. Connections can be defined either in the source code or outside to ease code reuse
Component structure	Components are structured in two parts: the <i>content</i> , providing the component behaviour, and the <i>membrane</i> , made of <i>controllers</i> and <i>interceptors</i> which manage all non-functional aspects. This structure allows a clear separation of the business code
Singleton interface	Point-to-point connections between components are easily described using the interfaces: the client interface, to emit operation requests, and the server interface to receive them. Interfaces can be introspected using standardized APIs
Collective interface	Multiple connections between one component and a set of other components can be easily specified. Multicast and gathercast cardinality are available. Message content distribution can be configured (broadcast, scatter, round robin,...) and customized
ADL support	The Architecture Description Language (ADL) provides an intuitive way to statically configure and instantiate the components and the bindings

PROACTIVE DEPLOYMENT FRAMEWORK

Entire application deployment

Deployment functionalities are independent from the source code to allow a quick and easy update when moving to another infrastructure.

Standardized descriptor	Specify in ETSI standardized XML files the resources that will be aggregated and the way to connect to the resources, the applications that will be used. Connection to the resources can be done by: <ul style="list-style-type: none">• Using RSH, SSH, SSHGSI protocols• Requesting resources to third-party job schedulers like LSF or PBS. See the complete list of supported job schedulers in the below IT ENVIRONMENT paragraph
ProActive Scheduling	ProActive Programming applications can be submitted as jobs to ProActive Scheduling

Programming-based deployment

ProActive programming is fully compatible and integrated with ProActive Agent and ProActive Resource Manager:

Resource Manager	From ProActive programming, resources can be acquired via the ProActive Resource Manager and used to deploy whole or parts of the application
ProActive Agent	From ProActive programming, nodes managed by ProActive Agent can be acquired and used to deploy whole or parts of the application

PROACTIVE DEBUGGING AND OPTIMIZING TOOLS

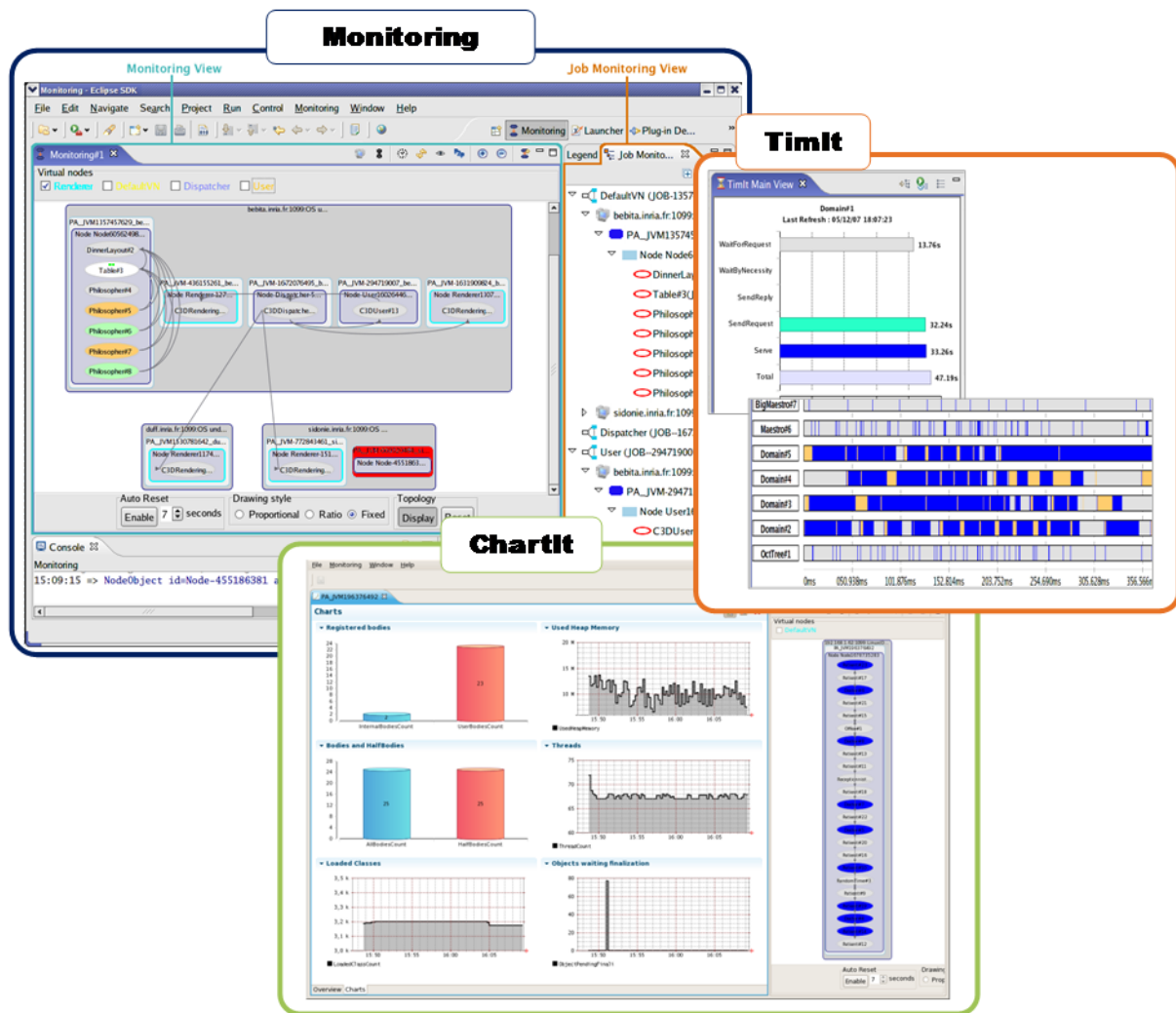
Embedded debugging functionalities

Zero deployment model	Simplify application development: allow testing the application easily in conditions equivalent to real ones while keeping testing local
Compile time annotation	Ensures source code compliance with ProActive rules. Check is performed at compile time

IC2D, standalone debugging tool

IC2D is ProActive graphical monitoring and control framework based on Eclipse Rich Client Platform (RCP).

Monitoring plug-in	Displays virtual nodes available in the applications as well as the deployed runtimes and the Active Objects instantiated with their state and their communication. These elements can be displayed in a diagram or tree view. The associated console displays textual information about the system activity
ChartIt	The ChartIt plug-in allows users to chart in real time any numerical values provided by ProActive resources such as runtimes (JVMs), nodes and active objects involved in the distributed application
TimIt	The TimIt plug-in allows profiling a distributed application in real time and this way underlines activity bottlenecks in order to further improve the application performance. It provides statistics and timeline view of the application activity. Special events within the application can be monitored as well



IC2D GUI

IT ENVIRONMENT

The following IT environments are supported by ProActive Programming:

OS	Linux, Windows, Solaris, MacOS
JVMs	Sun, SGI, BEA, JRockit
Communication protocols	RMI, RMIssh, HTTP, IBIS, PAMR (ProActive Message Routing, to handle firewalls and NAT)
Job schedulers	PBS, LSF, Sun Grid Engine, Oar, Prun, EGEE gLite, Globus, IBM Load Leveler.

COMING SOON

File Split & Merge	Inputs data files is easily split into relevant chunks using ProActive Scheduling. Results merge is done the same way
Run time Annotation	Will enable a more transparent integration of parallel aspects
Debugging	A Java debugger will be integrated to IC2D tool as well as the display of DSI-DSF request tagging



2004, route des Lucioles
BP 93
06902 Sophia Antipolis Cédex
France

Tel: +33 (0) 4 92 38 50 00
info@activeeon.com

**For more information, visit:
www.activeeon.com**