JppfProposal

JPPF : a parallel processing framework for Java

Abstract

JPPF enables applications with large processing power requirements to be run on any number of computers, in order to dramatically reduce their processing time. This is done by splitting an application into smaller parts that can be executed simultaneously on different machines.

Proposal

JPPF aims at facilitating the parallelization of computation-intensive applications, with a focus on ease of use, performance and reliability.

To achieve these goals, the framework comes with a number of outstanding features:

- Comprehensive, easy to use APIs: passing from a single-threaded application model to a grid-based parallel model can be a daunting task. JPPF facilitates this work by providing developers with a set of APIs that are simple, can be learned quickly and require a minimal or no modification to the existing code.
- No configuration usage: in most environments, JPPF can be deployed without any additional configuration burden. Nodes and application clients will automatically discover the servers on the network. The server will automatically adapt to workload changes and optimize the throughput. Required code and libraries will be automatically deployed where they are needed.
- Dynamic grid scaling and self-repair: the JPPF grid is fault-tolerant, meaning that the failure of a node, or even a server, does not compromise the jobs currently executing or scheduled. In most cases, the performance degradation will be barely noticeable, as JPPF automatically adapts to topology and workload changes. Furthermore, nodes and servers can be dynamically started and will be automatically recognized, allowing JPPF to function in "crunch mode". In addition to this, JPPF components benefit from automatic recovery functionalities.
- Job-level SLA: each job submitted to the JPPF grid runs within limits defined by its own SLA (service level agreement). This allows to specify the characteristics (i.e. available memory, processors, disk space, operating systems, etc.) of the nodes a job can run on, as well as how many nodes it can run on. As many functionalities in JPPF, this one can be dynamically adjusted, manually or automatically.
- Advanced Management and monitoring: full-fledged management and monitoring features are provided out of the box: server and nodes status monitoring, detailed statistics and events, remote administration, job-level real-time monitoring and management, charts, cpu utilization (for billing). These functionalities are available via a graphical user interface as well as from the JPPF APIs.
- Integration with leading application and web servers: by complying with the Java Connector Architecture 1.5 specification, JPPF integrates seamlessly with and completes the offering of leading J2EE application servers: Apache Geronimo, JBoss, Glassfish, IBM Websphere, Oracle Weblogic, Oracle OC4J. JPPF also integrates with GigaSpaces eXtreme Application Platform and Apache Tomcat web server
- Extensibility: JPPF offers a number of hooks and extension points that allow users to extend the framework and adapt it to their own requirements and needs. Such extension mechanisms are available for custom management and monitoring MBeans, startup classes for grid components, network data transformation/encryption, additional load-balancing algorithms, alternate object serialization mechanisms.

Background

JPPF is a project being actively developed at SourceForge. It was created to address a class of problems called "embarrassingly parallel", which groups computational problems that can be decomposed into many smaller sub-problems, that are independant from each other and that can thus be executed in parallel.

Rationale

Given these last years' emergence of technologies that make commodity hardware, virtualization and cloud computing available to a fast-growing computing ecosystem, the project answers the need to execute applications ever faster, with a low entry cost, while at the same time preserving historical technological investments.

Current Status

Meritocracy

We acknowledge that a meritocratic governance is the only way for the project to grow and expand, in the spirit of open source and the ASF. It will benefit the project, its communities, the ASF, as well as the outer ecosystems.

Community

JPPF already has a relatively small, but steadily growing community of users. Given the applicability of the project to numerous industries and technological areas such as scientific research, finance, graphical/video rendering, telecoms, data mining, etc.., we are confident that there is a very large growth potential for developers and users communities around JPPF.

Core Developers

JPPF was founded in April 2005 by Laurent Cohen (laurent.cohen at jppf.org), who is currently the only active developer and code committer. The other active contributor is John Channing (john.channing at gmail.com) who provides advice and peer review on the project's architecture, design, requirements and promotion. Other project members and former project contributors include:

- Domingos Creado (dcreado at users.sourceforge.net)
- Mahendra Kutare (mahendra.kutare at gmail.com)
- Guy Korland (Guy at gigaspaces.com)
- Fredreric Barachant (pepe-barachant at users.sourceforge.net)
- Peter Becker (nc-heuelhe at netcologne.de)
- Wolfgang Wagner (wolfgang.wagner at iname.com)
- Jay Yusko (jay.yusko at gensolco.com)

Alignment

The project is currently undergoing a refactoring of its network communication infrastructure, relying essentially on Mina, in order to provide a greater maintainability of the code, as well as extend its scalability and provide new functionalities such as truly secure communications and integration of other distributed computing models (e.g. P2P, map/reduce). JPPF also offers connectors for Geronimo and Tomcat.

Known Risks

Orphaned products

This is the main and most obvious risk of this project. The knowledge of the code is owned by a single developer, Laurent Cohen, who has committed about 98% of the existing code in the current repository. We are well aware of this problem, and it is our hope and challenge that integrating the Apache community will inspire the growth of a strong developers community around JPPF.

Inexperience with Open Source

JPPF was created as an Open Source project in April 2005, and has remained so since.

Homogenous Developers

A single active developer implies homogeneity.

Reliance on Salaried Developers

No one is currently paid to work on JPPF.

Relationships with Other Apache Products

- JPPF is currently refactoring its network communication infrastructure with the help of Mina (which is how we came to find a sponsor).
- The project's logging relies entirely on commons-logging and Log4j.
- The project also provides connectors for Geronimo and Tomcat.
- Builds are done with Ant, and we are currently looking at switching to Maven-based builds.
- We foresee that JPPF could be used by many other Apache projects to speed-up the execution of unit tests by running them in parallel, and to serve as a basis for running distributed charge and load tests.
- One of our samples uses Lucene to demonstrate a simple use of a distributed framework for web search, crawling and indexing

A Excessive Fascination with the Apache Brand

Our main hope in joining the Apache community is that it will help build a strong developers and committers community around the project, and remove its reliance on a single developer. We aim at achieving this while attracting new users, in conformance with the Apache spirit and policies.

Documentation

Information on JPPF can be found at:

http://www.jppf.org (main site, documentation and user forums)

http://sourceforge.net/projects/jppf-project (code repository, bug tracker, features tracker)

Initial Source

The entire code for JPPF is held in a SourceForge CVS repository, and has been so since the project's inception in April 2005. This entire code base will be donated to Apache. The source code has been licensed under the ASL 2.0 since August 2007. Before that it was LGPL. All artifacts in the trunk and existing branches (1 branch) are now licensed under the ASL 2.0, with corresponding header when applicable.

External Dependencies

Other than the Apache products used in JPPF, the project currently depends on the following ASL 2.0 compatible products/licenses:

- Rhino (MPL)
- JGoodies Looks (BSD)
- Groovy (ASL 2.0)
- Hazelcast (ASL 2.0)
- JUnit (CPL)
- MigLayout (BSD)
- Smart and Simple Web Crawler (ASL 2.0)

Dependencies on libraries with non AL-compatible licenses:

- JFreeChart (LGPL) - runtime dependency only
- SaverBeans SDK (LGPL) - in the process of being removed
- JAligner (GPL)
- Izpack (GPL)
- NSIS (zlib/libpng, bzip2, CPL) - in the process of being removed

Cryptography

JPPF does not have any cryptographic component. However, the distribution includes a sample that shows encryption/decryption of data as a demonstration of one of its features. The sample is delivered with full source code and can be found at:

http://www.jppf.org/wiki/index.php?title=Extending_and_Customizing_JPPF#Transforming_and_encrypting_networked_data

Required Resources

Mailing lists

- jppf-private (with moderated subscriptions)
- jppf-dev
- jppf-commits
- jppf-user

Subversion Repository

- https://svn.apache.org/repos/asf/incubator/jppf

Issue Tracking

- JIRA (JPPF)

Others

- Web site: Confluence (JPPF)

Initial Committers

- Laurent Cohen (laurent.cohen at jppf.org)
- John Channing (john.channing at gmail.com)

Affiliations

None of the initial committers are paid by their employer, nor do they represent their employer in any activity related to JPPF.

Sponsors

Champion

- Emmanuel Lecharny (elecharny at apache dot org)

Nominated Mentors

We are currently looking for mentors within the community.

Sponsoring Entity

- Apache Incubator