

Comparação entre Plataformas *Peer-to-Peer* para Desenvolvimento de Aplicações de Computação Distribuída

Marcelo Veiga Neves, Tiago Scheid, Elton Nicoletti Mathias,
Edmar Pessoa Araújo Neto, Andrea Schwertner Charão

¹Curso de Ciência da Computação
Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brasil

{veiga, scheid, emathias, araujo, andrea}@inf.ufsm.br

Resumo. Este artigo apresenta uma comparação entre plataformas Peer-to-Peer para desenvolvimento de aplicações de computação distribuída. Acredita-se que esta comparação possa fornecer subsídios para construção de novas aplicações distribuídas através da identificação das vantagens e desvantagens de algumas plataformas, em relação as necessidades mais freqüentes e dos recursos freqüentemente disponíveis.

1. Introdução

O processamento paralelo e distribuído é freqüentemente utilizado para resolver problemas que demandam um grande poder computacional. Existem diferentes abordagens para a construção de aplicações paralelas e distribuídas de alto desempenho, sendo que os sistemas *peer-to-peer* (P2P) estão entre as alternativas mais discutidas atualmente.

Basicamente, um sistema P2P caracteriza-se pela forma essencialmente descentralizada de empregar recursos distribuídos - como poder computacional, capacidade de armazenamento ou largura de banda - para realizar uma operação crítica [Milojicic et al. 2002]. Um exemplo de utilização desta abordagem é o projeto SETI@home [Korpela et al. 2001], que emprega milhões de computadores conectados à Internet para processar grandes conjuntos de dados.

Nos últimos anos, cresceu o número de plataformas destinadas a facilitar o desenvolvimento e execução de aplicações P2P. Com isso, antes de optar-se por uma ou outra ferramenta, torna-se necessário conhecer suas características e funcionalidades, levando-se igualmente em conta os requisitos das aplicações a serem desenvolvidas.

No intuito de fornecer subsídios para a construção de novas aplicações de computação distribuída, este trabalho apresenta uma comparação entre algumas plataformas P2P disponíveis atualmente. O texto está organizado da seguinte maneira: inicialmente (seção 2) discute-se algumas características desejáveis em plataformas para desenvolvimento e execução de aplicações P2P. Em seguida (seção 3), apresenta-se as plataformas escolhidas para análise. Por fim (seção 4), traça-se um comparativo entre estas plataformas, ressaltando-se suas principais vantagens e desvantagens.

2. Características Desejáveis em Plataformas P2P

A abordagem P2P é atraente para aplicações onde uma tarefa computacionalmente onerosa possa ser dividida em múltiplas sub-tarefas praticamente independentes, que possam ser executadas por diferentes computadores (*peers*) interligados.

Quando se deseja desenvolver uma aplicação P2P para processamento de alto desempenho, a escolha da plataforma deve levar em consideração algumas necessidades freqüentes:

- Para algumas aplicações, existe a necessidade de um elemento central, o qual se responsabiliza por armazenar as informações necessárias para a execução das tarefas. Para resolver esse problema, alguns sistemas adotam uma abordagem P2P híbrida, que alia as características de computação P2P ao modelo cliente-servidor.
- Há aplicações que necessitam resolver dependências de dados em algum ponto da computação, devendo assim haver comunicação entre *peers*. Para isso é desejável que os *peers* possam trocar mensagens entre si.
- Quando se utiliza a plataforma para computação colaborativa, isto é, para permitir que computadores voluntários colaborem na resolução de um problema, é importante prover segurança aos participantes.
- É possível que os usuários dessas plataformas não sejam especialistas em computação e sim cientistas, como físicos, matemáticos, biólogos, entre outros. Portanto, é importante que o sistema seja fácil de instalar e de utilizar.

Além destas necessidades, é importante levar em consideração os recursos freqüentemente disponíveis:

- Nem sempre há a disponibilidade de um ambiente computacional dedicado para computação paralela e distribuída. Neste caso, pode-se fazer uso compartilhado de redes de computadores. Por isso, a plataforma deve poder trabalhar com a dinamicidade de ambientes como esse.
- Também em ambientes compartilhados, pode ser necessário trabalhar apenas com períodos de ociosidade computacional, de modo a não concorrer por recursos, com as aplicações de usuários.
- É comum a utilização de ferramentas para o gerenciamento de recursos, como por exemplo Condor [Lodygensky et al. 2003]. Portanto, é geralmente desejável que o sistema de computação P2P seja capaz de interoperar com estas e outras ferramentas que eventualmente possam estar em funcionamento no ambiente utilizado.
- O ambiente computacional existente pode possuir diferentes configurações de *hardware* e *software*. Portanto, o sistema para computação P2P também deve ser capaz de operar em sistemas heterogêneos.

3. Apresentação das Plataformas Analisadas

Existem várias plataformas que fornecem uma infra-estrutura para o desenvolvimento e execução de aplicações distribuídas P2P. Neste trabalho, optou-se por analisar as plataformas XtremWeb [Fedak et al. 2000], BOINC [Anderson 2002], JXTA [Li 2001] e Pro-Active P2P [Caromel et al. 1998].

Estas plataformas foram escolhidas, principalmente, por serem *open source* e fazerem parte de projetos ativos na área de computação P2P.

3.1. XtremWeb

XtremWeb é uma plataforma para computação P2P desenvolvida na Universidade de Paris-Sud, França [Fedak et al. 2000]. Este sistema foi originalmente projetado para o

estudo de modelos de execução para a Computação Global, mas tornou-se uma plataforma completa para execução de aplicações distribuídas P2P.

XtremWeb possui uma arquitetura que segue um modelo coordenador-trabalhador. Um nó coordenador gerencia um conjunto de tarefas (*bag of tasks*) e coordena o escalonamento destas sobre um conjunto de máquinas voluntárias (trabalhadores). Normalmente, uma ou mais aplicações são instaladas nos trabalhadores e clientes podem submeter tarefas ao coordenador. Nesse caso, as tarefas submetidas são registradas no coordenador para serem então escalonadas para os trabalhadores. Para ambientes de uso compartilhado, esse escalonamento pode utilizar apenas períodos de ociosidade dos trabalhadores.

Todos os trabalhadores são nós voluntários. Dessa forma, o coordenador não tem controle sobre os nós disponíveis, pois todas as ações e conexões são iniciadas pelos trabalhadores. Por isso, é possível trabalhar com ambientes bastante dinâmicos. Esse modelo é conhecido com *pull* e implica na independência de todos os componentes, não sendo permitida a comunicação entre *peers*.

Esta arquitetura não provê nenhum mecanismo de segurança para proteger os nós voluntários contra aplicações de usuários não confiáveis. No entanto, está sendo desenvolvido um mecanismo de *sandboxing* para proteger os trabalhadores de códigos maliciosos que podem estar contidos em aplicações de usuários. Além disso, todas as comunicações entre clientes e o coordenador são criptografadas utilizando SSL (*Secure Sockets Layer*).

Atualmente, XtremWeb possui implementações para Linux, Windows e MacOS-X e consegue interoperar com o gerenciador de recursos Condor. XtremWeb está disponível em <http://www.xtremweb.org/>.

3.2. BOINC

BOINC (*Berkeley Open Infrastructure for Network Computing*) [Anderson 2002] é uma plataforma para computação distribuída que permite utilizar recursos públicos. BOINC foi desenvolvido pelo Laboratório de Ciências Espaciais de Berkeley, mesmo grupo desenvolveu e continua a operar o projeto SETI@home. BOINC tem como principal objetivo avançar o paradigma de computação que utiliza recursos públicos, e encorajar uma boa parte dos usuários de computador do mundo a disponibilizar o tempo ocioso de seus computadores para um ou mais projetos de pesquisa.

BOINC possui uma arquitetura cliente-servidor. O servidor é centrado em um banco de dados relacional, tipicamente MySQL, o qual armazena a descrição das aplicações e todas as informações necessárias para o seu funcionamento. As funções do servidor são executadas por um conjunto de *web services* e processos *daemons*, que juntos realizam o escalonamento de tarefas, atendimento aos clientes via RPC, carga de arquivos e recebimento dos resultados final da computação. Já os clientes são responsáveis por executar a aplicação no computador voluntário, quando o mesmo estiver ocioso.

Para garantir a validade dos resultados, normalmente o servidor escalona várias unidades de trabalho e depois compara as várias respostas considerando a moda como resultado final. Com a resposta escolhida, o servidor remove do banco de dados a unidade de trabalho terminada, eliminando também as redundâncias de dados. A geração de unidades de trabalho fica sob responsabilidade do desenvolvedor de aplicações. Para isso, BOINC oferece uma API para criação de unidades de trabalho no banco de dados.

É importante mencionar que BOINC não possui suporte para comunicação entre clientes.

Atualmente, BOINC suporta os sistemas operacionais Solaris, Linux, Mac OS X e Microsoft Windows. Além disso, é capaz de operar em conjunto com LHC Computing Grid [CERN 2005] através de uma ponte entre os dois sistemas. BOINC também possui uma API em C++ para implementação da aplicação que é executada pelo cliente e algumas funções que são utilizadas pelo servidor. Também possui uma interface para algumas funções em C e FORTRAN. BOINC está disponível em <http://boinc.berkeley.edu/>.

3.3. JXTA

JXTA é uma plataforma para desenvolvimento de aplicações distribuídas projetada pela Sun Microsystems e desenvolvida com o auxílio de alguns especialistas de instituições acadêmicas e industriais. Esta plataforma tem como objetivo resolver problemas de computação P2P e, ao mesmo tempo, alcançar características como interoperabilidade, portabilidade e a capacidade de ser agregado a todo tipo de dispositivos digitais que suportem aplicações distribuídas.

JXTA consiste em um conjunto de protocolos onde cada um deles é definido pela troca de algumas mensagens entre os participantes, sendo que cada mensagem tem um formato pré-definido. Estes protocolos definem, por exemplo, a busca por novos nós, busca por recursos, busca por informações sobre os *peers* e seus estados, criação de grupos de *peers* e etc.

Os protocolos do JXTA criam um rede virtual de *peers* que abstrai a rede física. Eles possibilitam que os *peers* possam trocar mensagens independente da sua localização real, roteando as mensagens de forma transparente entre nós cercados por *firewalls* ou que utilizam protocolos de comunicação diferentes. JXTA também dá a capacidade aos *peers* de se comunicar sem conhecer ou precisar gerenciar mudanças físicas na topologia da rede, garantindo que *peers* móveis possam se locomover transparentemente. Estes protocolos padronizam a maneira em que os *peers* se encontram, se comunicam, formam grupos e disponibilizam recursos. Dessa forma os *peers* não dependem de um nó central que gerencia a dinamicidade dos recursos computacionais.

Atualmente, o projeto JXTA possui implementações disponíveis nas linguagens C e Java, com ampla documentação e vários guias de programação. A implementação Java garante a portabilidade para vários sistemas operacionais e arquiteturas. Para segurança, JXTA implementa os algoritmos RSA, SHA-1 e MD5. JXTA está disponível em <http://www.jxta.org/>.

3.4. ProActive P2P

ProActive é um *middleware* que busca oferecer um modelo de programação concorrente e distribuída com transparência. Esse *middleware* está sendo desenvolvido em um esforço conjunto entre o grupo de pesquisa Oasis, da Université de Nice Sophia Antipolis e o instituto francês INRIA (*Institute National de Recherche en Informatique et en Automatique*), sendo parte do consórcio europeu ObjectWeb. Este *middleware* oferece uma biblioteca para computação distribuída e paralela em Java, um ambiente gráfico de monitoração e lançamento de tarefas e uma infra-estrutura P2P.

O objetivo principal da infra-estrutura P2P de ProActive é a utilização transparente de ciclos de processadores disponíveis em estações de trabalho presentes em instituições, combinados com grades computacionais e aglomerados de computadores. Para tanto, utiliza-se de protocolos que permitem lançamento de processos e chamada remota de métodos através de protocolos bastante utilizados, como ssh, rsh, http, soap, prun, etc. Além desses protocolos, é possível interoperar com outras ferramentas de computação distribuída, como Globus [Foster and Kesselman 1997], Ibis [van Nieuwpoort et al. 2002], entre outras.

O funcionamento do ambiente P2P é guiado por arquivos de configuração onde se determinam os horários de funcionamento dos *peers*, os protocolos de comunicação utilizados e a lista de *peers* conhecidos. A localização de *peers*, para a formação da rede P2P, é feita através de um algoritmo de busca BFS (*Breadth-First Search*), semelhante ao algoritmo utilizado na rede Gnutella [Gnutella 2003], sendo que *peers* não acessíveis diretamente comunicam-se através de um mecanismo transparente de encaminhamento de mensagens.

Graças ao *middleware* ProActive, no qual este ambiente se insere, uma série de funcionalidades podem ser agregadas a aplicações que fazem uso desta estrutura. Entre essas funcionalidades podemos citar um mecanismo de tolerância a falhas e *checkpointing* distribuído, mecanismos de comunicação em grupo e chamadas de método criptografadas. Além disso, a ferramenta gráfica oferecida permite o controle do ambiente de execução, lançamento e migração de tarefas, e mecanismos que facilitam a depuração dessas aplicações.

Por ser implementado inteiramente em Java, a plataforma ProActive é portátil a vários sistemas operacionais e arquiteturas de computadores, dependendo apenas da existência de uma máquina virtual Java compatível com o sistema desejado. Os mecanismos de busca de *peers* e redirecionamento de mensagens permitem a implementação de aplicativos sem preocupações relativas a localização ou estrutura física na qual a rede P2P organiza-se. Entretanto, é obrigatória a criação de arquivos de configuração para cada *peer* que compõe a rede, o que pode ser uma tarefa complexa, especialmente em ambientes amplos.

ProActive está disponível em <http://proactive.objectweb.org/>.

4. Comparação

O estudo das plataformas apresentadas na seção anterior foi feito através da leitura da documentação existente, da instalação e configuração dos ambientes, seguida da experimentação e análise do seu funcionamento. A comparação das plataformas foi baseada nos seguintes critérios: arquitetura, suporte a comunicação, escalonamento, mecanismos de segurança, interoperabilidade e pré-requisitos de instalação.

A Tabela 1 mostra uma comparação das plataformas testadas à luz dos vários critérios considerados.

5. Conclusões

Neste trabalho apresentou-se uma comparação entre as plataformas P2P XtremWeb, BOINC, JXTA e ProActive P2P. Através desta, comparação pôde-se observar algumas

Tabela 1 - Comparação entre plataformas para computação P2P

	XtremWeb	BOINC	JXTA	ProActive P2P
Arquitetura	centralizada	centralizada	distribuída	distribuída
Suporte a comunicação entre os <i>peers</i>	não	não	sim	sim
Escalonamento	utiliza ciclos ociosos	utiliza ciclos ociosos	deve ser implementado	agendamento de períodos
Mecanismos de Segurança	SSL e <i>sandboxing</i>	falsificação de resultados	RSA, SHA-1 e MD5	assinaturas DSA
Interoperabilidade	Condor	LHC		Globus e Ibis
Pré-requisitos	Java e MySQL	Java e MySQL	Java	Java

particularidades relevantes, que podem ser úteis a desenvolvedores que estão a procura de uma plataforma para computação P2P. Em XtremWeb e BOINC, foi possível observar a existência de um servidor central, que gerencia a distribuição de tarefas, e a não possibilidade de comunicação entre os *peers*, o que inviabiliza a construção de aplicações que possuam dependências de dados. Em ProActive P2P, notou-se a existência de uma interface gráfica para o lançamento de tarefas e possibilidade de integração com outros sistemas bastante difundidos como Globus e Ibis, no entanto, observou-se a necessidade de programar com o modelo de objetos ativos, utilizado pelo *middleware* ProActive. Em JXTA, percebeu-se que o mesmo não implementa algoritmos de escalonamento e distribuição de tarefas, no entanto, oferece uma liberdade maior ao desenvolvedor da aplicação através de sua API.

Referências

- Anderson, D. (2002). BOINC, Berkeley Open Infrastructure for Network Computing.
- Caromel, D., Klauser, W., and Vayssière, J. (1998). Towards seamless computing and metacomputing in Java. *Concurrency: Practice and Experience*, 10(11–13):1043–1061.
- CERN (2005). The LHC Computing Grid Project. <http://lcg.web.cern.ch/LCG/>.
- Fedak, G., Germain, C., Neri, V., and Cappello, F. (2000). Xtremweb : a generic global computing platform – ccgrid’2001 special session global computing on personal devices. IEEE press.
- Foster, I. and Kesselman, C. (1997). Globus: A Metacomputing Infrastructure Toolkit. 11(2):115–128.
- Gnutella (2003). Gnutella. Web Page.
- Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Lebofsky, M. (2001). Seti@home: Massively Distributed Computing for SETI.
- Li, G. (2001). Project JXTA: A technology overview. Technical report, Sun Microsystems.

- Lodygensky, O., Fedak, G., Cappello, F., Neri, V., Thain, D., and Livny, M. (2003). Xtremweb and Condor : sharing resources between Internet Connected Condor Pools. In Press, I., editor, *Proceedings of the 3rd International Symposium on Cluster Computing and the Grid*. Tokyo Japan.
- Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., and Xu, Z. (2002). Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Labs.
- van Nieuwpoort, R. V., Maassen, J., Hofman, R., Kielmann, T., and Bal, H. E. (2002). Ibis: an Efficient Java-based Grid Programming Environment. In *Proc. ACM Java Grande - ISCOPE Conf.*, pages 18 – 27.