I Escola Regional de Alto Desempenho de SP
Sessão de Iniciação Científica

ERAD-SP 2010

30 e 31 de Julho de 2010

Mensagem do Coordenador da Sessão de iniciação Científica

É com grande prazer que apresento a Sessão de Iniciação Científica da I Escola Regional de Alto Desempenho de SP (ERAD-SP 2010). Esta escola foi planejada com o objetivo de colocar os alunos do Estado de São Paulo em contato com outros alunos e pesquisadores da área de computação de alto-desempenho, fornecendo uma excelente oportunidade de interação.

Para a Sessão de Iniciação Científica, queríamos um espaço informal onde os alunos pudessem apresentar e discutir seus trabalhos. Para tal, adotamos o formato de pôsteres, que é interessante por possibilitar um melhor contato entre os alunos e demais participantes. Foram selecionados 16 trabalhos produzidos por alunos e professores de 10 diferentes instituições do Estado de São Paulo. Esta diversidade certamente contribuirá para enriquecer as discussões.

Finalmente, gostaria de agradecer a todos os autores, membros do comitê e revisores externos, sem os quais não poderíamos já nesta primeira edição oferecer um evento de qualidade.

Raphael Y. de Camargo

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC (UFABC)

Coordenação Geral

Denise Stringhini (Mackenzie)

Coordenação de Programa

Alfredo Goldman (USP)

Coordenação do Fórum de Pós-Graduação

Hermes Senger (UFSCAR)
Rodolfo Jardim de Azevedo (UNICAMP)

Coordenação da Sessão de Iniciação Científica

Raphael Y. de Camargo (UFABC)

Coordenação do Open Contest

Augusto Gomes Jr. (USP)
Calebe de Paula Bianchini (Mackenzie)

Coordenação Financeira e de Patrocínio

Francisco Masseto (UFABC)

Comitê Organizador

Alfredo Goldman (USP)
Augusto Gomes Jr. (USP)
Calebe de Paula Bianchini (Mackenzie)
Denise Stringhini (Mackenzie)
Edson Midorikawa (USP)
Francisco Masseto (UFABC)
Hermes Senger (UFSCAR)
José Márcio Machado (UNESP)
Liria Matsumoto Sato (USP)
Luciano Silva (Mackenzie)
Nicolas Kassalyas (São Judas)
Raphael Y. de Camargo (UFABC)
Roberto Kenji (USP)
Rodolfo Jardim de Azevedo (UNICAMP)
Sérgio Takeo Kofuji (USP)

Comitê de Programa

Aleardo Manacero Jr. (IBILCE - UNESP)
Alfredo Goldman (IME - USP)
Augusto Gomes Jr (Anhembi Morumbi)
Calebe de Paula Bianchini (Mackenzie)
Celso Hirata (ITA)
Denise Stringhini (Mackenzie)
Edson Midorikawa (Poli - USP)
Francisco Massetto (UFABC)
Hélio Guardia (UFSCar)
Hermes Senger (UFSCar)
José Machado (UNESP)
Liria Sato (Poli - USP)
Marcos Cavenaghi (UNESP)
Marcos Santana (ICMC - USP)
Raphael Camargo (UFABC)
Regina Santana (ICMC - USP)
Renata Spolon Lobato (UNESP)
Roberta Spolon (UNESP)
Rodolfo Azevedo (UNICAMP)
Rodrigo Mello (ICMC - USP)
Sergio Kofuji (POLI - USP)
Siang Song (IME - USP)

Índice

Sessão 1

Redes de Sensores e Sistemas Embarcados

Tolerância a Falhas em Redes de Sensores

Everton de Carvalho Silva[1], Priscila Azar Gimenes[1], Delano Medeiros Beder[1]

[1]Escola de Artes Ciências e Humanidades (EACH) - Universidade de São Paulo (USP)

{everton.carvalho.silva, priscila.gimenes, dbeder}@usp.br

1. Introdução

A aplicação de técnicas para desenvolvimento de softwares tolerantes a falhas possui uma ampla utilização no que diz respeito à disponibilidade, segurança de informações e eficiência de um sistema, tornando sua utilização confiável e eficaz [1].

O foco desse projeto é a implementação de um sistema tolerante a falhas em uma rede de sensores [2], implantada para monitoramento da pressão das águas de certo trecho de um rio, o qual tem como objetivo prever a possibilidade de uma enchente, avisando o gerenciador da rede para que este possa tomar providências.

2. Conceitos Básicos

Tolerância a falhas [1, 3] tem como conceito manter sistemas operando corretamente mesmo que falhas aconteçam ou evitando as falhas, com o objetivo de conseguir confiabilidade e/ou disponibilidade para que o sistema não chegue a uma condição de inoperância.

Tolerância a falhas pode ser aplicada de acordo com o estado do sistema; em estados críticos, a rede de sensores poderia ser reconfigurada (em tempo de execução) de forma a proporcionar uma maior confiabilidade, permitindo assim que políticas de tolerância a falhas possam ser inseridas, sem que o sistema tenha que ser abortado e reinicializado [2].

O Sun SPOT (Sun Small Programmable Object Tecnology) [4] foi a plataforma de sensores usada para desenvolver o protótipo do sistema de monitoramento de enchentes, pois é uma plataforma que facilita o desenvolvimento de rede de sensores. Este sensor utiliza rádio frequência para comunicação com outros sensores da mesma natureza, e uma base que será conectada a um computador para tornar possível a verificação dos dados capitados em cada ponto do rio onde os sensores forem instalados.

3. Protótipo/Algoritmo

A aplicação desenvolvida para o monitoramento das enchentes, independe do número de terminais, desde que todos os endereços de comunicação sejam conhecidos. No entanto, para efeito de testes, fizemos a simulação com cinco terminais, os quais são conectados ponto-a-ponto, que é uma conexão que provê maior segurança de comunicação, com dois sensores atrás (mais longes da base), com os quais se comunica só para receber mensagens e dois sensores à frente (mais perto da base) para passar as mensagens recebidas ou percebidas.

Como é possível ocorrer perda da mensagem enviada durante esse percurso, alguns casos relacionados com tolerância a falhas foram pensados como acabar sua bateria ou parar de funcionar.

Para o primeiro problema, o Sun SPOT [4] fornece um mecanismo em que é possível medir o nível da bateria, sendo possível pensar que quando chegasse a um nível definido, o sensor mandaria uma mensagem à base dizendo que a bateria está acabando. Porém, para efeitos de teste, foi simulado com um evento de apertar um dos botões (o da direita), do Sun SPOT que ao ser apertado envia, assim, a mensagem "Bateria Fraca" e seu endereço para a base para que o usuário interessado recarregue sua bateria.

Já para o caso de o sensor parar de funcionar, foi criado um teste para a verificação de todos os sensores que estão em funcionamento. A cada período preestabelecido cada sensor manda uma mensagem dizendo que está em funcionamento para a base. A aplicação da base, por sua vez, a cada período preestabelecido faz uma verificação para certificar se todos enviaram a mensagem durante esse tempo. Caso perceba que algum não enviou a mensagem, tem como se saber qual é o sensor que parou de funcionar e o usuário interessado é informado para realizar as intervenções necessárias.

Outro fato pensado é o caso das mensagens se perderem durante o percurso até chegar à base. Para ter uma maior garantia que todas as mensagens cheguem ao seu destino final, foi pensado enviar a mensagem não só para um sensor, mas sim para dois. Assim, se falharem sensores alternados, a mensagem chegaria à base mesmo assim. Porém se dois sensores consecutivos falharem, a mensagem será perdida. Isso é assumido em nossos estudos, e no protótipo implementado, que dois sensores consecutivos não podem falhar.

4. Considerações finais

O projeto envolve um cenário crítico da aplicação de redes de sensores [2], exigindo, portanto, a implementação de tolerância a falhas [1, 3] para que o sistema seja contemplado com confiabilidade, agregando eficácia ao seu comportamento. Os próximos passos envolvem implantar essa aplicação em um ambiente real na cidade de São Paulo.

Referências

[1]DAMASCENO, Karla; CACHO, Nelio; GARCIA, Alessandro; ROMANOVSKY, Alexander; LUCENA, Carlos; *Context-Aware Exception Handling in Mobile Agent Systems: The MoCA Case*. Proceedings of the 2006 international workshop on Software Engineering for large-scale multi-agent systems, May 22-23, 2006, Shanghai, China.

[2]AMORIM, CLÁUDIO L. DE; CASTRO, Maria Clicia Stelling de; PEREIRA, Marluce R. *Tutorial sobre Redes de Sensores*. Disponível em: <http://magnum.ime.uerj.br/cadernos/cadinf/vol14/3-clicia.pdf>.

[3]WEBER, Taisy Silva. *Um roteiro para exploração dos conceitos básicos de tolerância a falhas*. Rio Grande do Sul, 2002. 60 p. Instituto de Informática, Universidade Federal do Rio Grande do Sul.

[4]Sun Small Programmable Object Technology (Sun SPOT). Disponível em: <www.sunspotworld.com>.

Proposta de uma Arquitetura de Middleware Usando Serviços Web para Redes de Sensores sem Fio

Richardson W. Scherer[1], João H. Kleinschmidt[2]

[1]Departamento de Informática

Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa, PR - Brasil

[2]Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas - CECS

Universidade Federal do ABC (UFABC) – Santo André, SP - Brasil

majindox@gmail.com, joao.kleinschmidt@ufabc.edu.br

1. Introdução

Este artigo propõe a criação de uma arquitetura para uma rede de sensores sem fio (RSSF) utilizando serviços web para a comunicação entre as aplicações cliente e servidor. As redes de sensores sem fio possuem os nós sensores responsáveis pela coleta, processamento e retransmissão de dados, e os nós coletores responsáveis pelo processamento e armazenamento dos dados (Loureiro et al, 2003). Os serviços web são aplicações baseadas em uma arquitetura orientada a serviços (SOA) capazes de disponibilizar coleções de operações que podem ser acessadas por outras aplicações (Delicato et al, 2004). Esta tecnologia permite a comunicação e troca de mensagens de modo independente da linguagem utilizada, já que os serviços web utilizam mensagens de forma padronizada compostas através da linguagem XML, fornecendo vantagens de interoperabilidade, comunicabilidade e acessibilidade (Abinader et al, 2006).

2. Serviços Web para Redes de Sensores

Após o estudo sobre as tecnologias e trabalhos relacionados, deu-se a concepção do modelo proposto da arquitetura, verificando-se quais requisitos deveriam ser cumpridos (Hadim 2006). Estabeleceu-se a idéia de que os nós sensores ficariam responsáveis pela coleta dos dados para então retransmiti-los para os nós coletores responsáveis por armazená-los em um banco de dados. Os serviços web fazem a consulta aos dados para disponibilizá-los a uma aplicação cliente, nesse caso uma página web. Como plataforma principal de desenvolvimento, foi escolhida a plataforma Java, capaz de atender todos os requisitos do projeto. A ferramenta principal utilizada foi o ambiente de desenvolvimento integrado NetBeans IDE versão 6.7.1.

Para o desenvolvimento do protótipo dos nós sensores foi escolhida a linguagem de programação Java 2 Micro Edition (J2ME), que por ser uma linguagem própria para dispositivos móveis e dispositivos compactos, favorece a aplicação quanto aos requisitos de processamento e armazenamento reduzido, proporcionando um melhor uso de energia na rede, tendo em mente que estes seriam princípios fundamentais a serem observados na aplicação de sensores reais. O protótipo dos nós coletores foi desenvolvido de maneira que funcionem como um servidor responsável por receber os dados enviados pelos nós sensores e armazená-los em um banco de dados, assim, foi utilizada a linguagem Java 2 Standard Edition (J2SE) proporcionando um

desenvolvimento mais amplo e próprio para aplicações Java comuns. Para o gerenciamento do banco de dados foi utilizado o MySQL.

Os serviços web foram criados no ambiente NetBeans, utilizando o pacote de ferramentas web do próprio ambiente, permitindo uma completa integração com as demais partes do projeto, em vista da maior facilidade de implementação dentro de uma mesma plataforma. Por fim, para a disponibilização das informações para os usuários foi criada uma página web utilizando a linguagem JSP (Java Server Page), escolhida por ser uma tecnologia totalmente baseada na Plataforma Java 2, tendo, portanto, acesso a todas as características e ferramentas da Plataforma Java, como os kits de desenvolvimento. Outra ferramenta utilizada no desenvolvimento das páginas web foi o JFreeChart. A aplicação cliente foi desenvolvida como páginas web utilizando JSP. Existem dois módulos principais: o módulo de usuário que é responsável por apresentar as informações ao usuário, e o módulo administrativo responsável pela parte de controle das funcionalidades dos sensores dentro do projeto. O módulo administrativo possui a opção de gerenciar algumas funcionalidades da RSSF. Através de um *login* feito na página inicial do projeto, o administrador é levado até a página inicial deste módulo, onde pode acessar algumas das outras páginas disponíveis, podendo realizar algumas tarefas gerenciais, como o monitoramento, alteração de estados e cadastro dos sensores.

3. Conclusão

Neste trabalho desenvolveu-se uma arquitetura para uma rede de sensores sem fio utilizando serviços web. O projeto foi feito considerando os principais requisitos a serem observados em uma RSSF, como a utilização de energia, processamento e autonomia dos dispositivos, e teve como objetivo criar uma arquitetura que fosse capaz de simular a coleta e processamento dos dados e então disponibilizá-los para o usuário. Os serviços web desenvolvidos permitiram que as informações armazenadas fossem requisitadas e utilizadas de modo eficiente e prático pela aplicação cliente. O desenvolvimento da arquitetura permitiu verificar o comportamento e funcionamento de uma rede de sensores sem fio e como os serviços web poderiam ser implementados nela de modo a facilitar o processo de disponibilização das informações para o usuário final. Todo o processo de coleta, tratamento, armazenamento e consulta das informações é realizado de modo a possibilitar a integração entre os sistemas e promover a usabilidade dos recursos da RSSF e dos serviços web.

4. Referências

Abinader Neto, Jorge Abílio; Lins, Rafael Duiere. “Web Services em Java”. Rio de Janeiro: Brasport, 2006.

Delicato, F. C.; Pirmez, L. Pires, P.; Rezende, J. F.; Lages, A.. “Middleware Orientado a Serviços para Redes de Sensores sem Fio”. In: 22º Simpósio Brasileiro de Redes de Computadores, Gramado, 2004.

Hadim, S. e Mohamed, N. “Middleware: Middleware challenges and approaches for wireless sensor networks”. *IEEE Distributed Systems Online*, vol. 7, no. 3, 2006.

Loureiro, Antonio A. F.; Nogueira, José Marcos S.; Ruiz, Linnyer B.; Freitas, Raquel Ap.; Nakamura, Eduardo F.; Figueiredo, Carlos M. S.. “Redes de Sensores sem Fio”. In: XXI Simpósio Brasileiro de Redes de Computadores, Natal, RN. XXI Simpósio Brasileiro de Redes de Computadores, 2003.

Estudo e Implementação de Segurança em Redes de Sensores usando Serviços Web

Fábio Stankievicz[1], João H. Kleinschmidt[2]

[1]Departamento de Informática
Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa, PR - Brasil

[2]Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas - CECS
Universidade Federal do ABC (UFABC) – Santo André, SP - Brasil

fabiostankievicz@hotmail.com, joao.kleinschmidt@ufabc.edu.br

1. Introdução

Serviços web são serviços lógicos e distribuídos que independem de linguagem de programação e sistema operacional, utilizando padrões abertos, como o *eXtensible Markup Language* (XML), WSDL (*Web Service Definition Language*) e o *Simple Object Access Protocol* (SOAP) para a troca de mensagens. Alguns serviços web estão disponíveis para uso público e não possuem nenhum esquema de segurança, porém a grande aplicação de serviços web em outras aplicações necessita de criptografia e autenticação (Abinader et AL, 2006). Redes de sensores sem fio (RSSFs) são redes compostas por milhares de dispositivos de baixo custo e tamanho reduzido com capacidade de realizar sensoriamento, processamento e transmissão de informações usando comunicação sem fio (Delicato et al, 2004) (Loureiro et al, 2003). Tendo em vista que algumas RSSFs podem estar localizadas em ambientes externos e necessitam de protocolos de transmissão de dados que necessitem de pouco processamento, os serviços web tornam-se uma alternativa para essas redes. Este trabalho traz um estudo de sistemas de segurança disponíveis para serviços web, tendo como estudo de caso a simulação de uma rede de sensores.

2. Serviços Web Seguros para Redes de Sensores

Entre as especificações de segurança adotadas estão o *XML Signature, XML Encryption* e XACML (Mello et AL, 2006). O *XML Encryption* é utilizado para garantir a confidencialidade das mensagens através de criptografia / decriptografia. Este processo cifra a mensagem do lado do emitente e decifra do lado do receptor utilizando tanto de algoritmos simétricos quanto assimétricos. *XML Signature* é utilizado para garantir a integridade das mensagens através de um sistema de assinaturas digitais. O processo de assinatura consiste na criptografia do resumo da mensagem com a utilização da chave privada, e do outro lado a decriptografia usando a chave pública e comparação desse resumo com um novo gerado. XACML é utilizado para garantir a autenticidade das mensagens. É baseado em pedidos e respostas entre solicitante e serviço, definindo assim permissões de acesso aos recursos desejados.

A aplicação consiste em uma simulação de uma rede de sensores com nós sensores e nós coletores que trocam mensagens através de serviços web que garantem

serviços seguros implementados em linguagem Java. Usuários externos (clientes) podem utilizar a RSSF para coletar informações. Foi utilizado o NetBeans e a biblioteca JAX-WS (*Java API for XML Web Services*).A aplicação do servidor (nós coletores da RSSF) possibilita a escolha de duas possibilidades de segurança para o cliente: criptografia e assinatura digital. A aplicação implementa envelope digital, através da encriptação da chave simétrica utilizada para encriptar o resultado da consulta, utilizando a chave pública fornecida pelo cliente, com isso garante confidencialidade e integridade dos dados trafegados. Além disso, o servidor utiliza assinatura digital quando o mesmo cria um novo par de chaves assimétricas, gera um resumo da mensagem criptografada, e cifra o mesmo com a nova chave privada gerada no servidor. A resposta enviada ao cliente contém a mensagem cifrada, a chave simétrica cifrada pela chave pública do cliente, resumo encriptado com a chave privada do servidor e a chave pública do servidor para decriptar o resumo. O cliente ao receber o pacote primeiramente terá que interpretá-lo e só depois disso realizar os passos reversos para ler a informação.

3. Conclusão

Este trabalho descreve várias especificações de segurança e uma comparação dos serviços de autenticidade, integridade e confidencialidade oferecido em algumas delas. Propõe também uma maneira alternativa de garantir segurança para a as trocas de mensagens através da linguagem Java, implementando alguns serviços fornecidos por algumas especificações de segurança. As características das RSSFs a tornam suscetíveis a diversos tipos de ataques (Wang 2006). Logo, em ambientes onde se utiliza RSSFs, é fundamental que as informações trafegadas na rede tenham tratamento para que os dados não sejam violados e estejam íntegros ao final da transmissão. O trabalho mostra que os serviços web são uma alternativa viável para a segurança de RSSFs.

4. Referências

Abinader Neto, Jorge Abílio; Lins, Rafael Duiere. “Web Services em Java”. Rio de Janeiro: Brasport, 2006.

Delicato, F. C.; Pirmez, L. Pires, P.; Rezende, J. F.; Lages, A.. “Middleware Orientado a Serviços para Redes de Sensores sem Fio”. In: 22º Simpósio Brasileiro de Redes de Computadores, Gramado, 2004.

Loureiro, Antonio A. F.; Nogueira, José Marcos S.; Ruiz, Linnyer B.; Freitas, Raquel Ap.; Nakamura, Eduardo F.; Figueiredo, Carlos M. S.. “Redes de Sensores sem Fio”. In: XXI Simpósio Brasileiro de Redes de Computadores, Natal, RN. XXI Simpósio Brasileiro de Redes de Computadores, 2003.

Mello, Emerson. R.; Wangham, Michelle. S.; Fraga, Joni. S.; Camargo, Edson. T. “Segurança em Serviços Web”. VI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg), p. 1-48. Santos, SP, setembro de 2006.

Wang, Y., Attebury, G. e Ramamurthy, B. “A survey of security issues in wireless sensor networks”. *IEEE Communications Surveys*, vol. 8, no. 2, pp. 02-23, 2006.

Aplicações Escaláveis e Realísticas em Sistemas Operacionais Embarcados

Bruno E. P. Costa, Cesar A. C. Marcondes

Departamento de Computação – Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 – 13565-905 – São Carlos – SP – Brasil

bruno_eustaquio@comp.ufscar.br, marcondes@dc.ufscar.br

Resumo. *Esse artigo apresenta um trabalho em andamento de Iniciação Cientifica, onde se têm desenvolvido aplicações de sistemas embarcados robóticos usando Lego Mindstorms e foram feitas decisões de linguagens e sistemas operacionais em função de simplicidade e escalabilidade. Como passo futuro, pretende-se realizar a integração das aplicações desenvolvidas com simuladores de alto desempenho para testar a escalabilidade das aplicações.*

1. Introdução

Este trabalho de Iniciação Cientifica visa o estudo da integração híbrida de aplicações embarcadas reais executando em robôs e/ou sensores com um ambiente de simulações de redes sem fio de larga escala. A idéia é facilitar testes de larga escala com sistemas embarcados críticos, como por exemplo, testar um novo software em milhares de nós, e simultaneamente permitindo que o mesmo código embarcado seja usado tanto em campo quanto no simulador, diminuindo o custo de recodificar um código específico para o simulador e um para o sistema embarcado real. A idéia chave da linha de pesquisa é permitir uma simulação híbrida e mais realística da eficiência de algoritmos voltados para sensores, bem como o consumo energético dos mesmos e integração de sistemas operacionais (SOs) para sensores em ambiente de simulação. Para desenvolver esse trabalho, foi feito um estudo em simulação de redes sem fio, no simulador proprietário de alto desempenho chamado Qualnet, em sua versão academica. No que tange a sistemas operacionais embarcados, foi realizada uma revisão geral dos principais representantes, entre eles TinyOS e LeJOS, que serão discutidos com mais detalhes.

2. Sistemas operacionais embarcados

Sistemas operacionais para sensores exibem várias limitações e restrições. Por exemplo, não podem ocupar excessivo espaço em disco, não podem exigir grande processamento, devem ter um gerenciamento otimizado de memória, devem ser de tempo real e econômico quanto ao consumo de energia. A seguir, faremos uma breve descrição dos S.Os que estudamos como boas opções a serem integrados a simuladores.

2.1 TinyOS

O TinyOS, ou Tiny Operational System, é um S.O. de tempo real, feito por Gay[2003], para ser embarcados em sensores (8 kb de memória de programa, 512 kb de memória RAM), contendo um escalonador de tarefas, um modelo de desenvolvimento e linguagem própria, o nesC. É baseado em eventos, permitindo concorrência com pouco

uso de memória. Nesse, existe a separação entre construção e composição: programas são construídos a partir de componentes, que são montados para compor o programa inteiro. Os componentes definem dois escopos, um para sua especificação, contendo os nomes das suas interfaces de instancia, e uma para sua implementação. A simultaneidade interna dos componentes é dada na forma de tasks. Threads de controle fazem a ligação de um componente até sua interface. Essas threads são tratadas nas tasks ou são baseadas em algum evento externo captado pelo hardware(sensores).

2.2 LeJOS

O S.O. LeJOS, Bagnall[2007], é constituído de um firmware que possui uma Máquina Virtual Java (MVJ), e modelo de programação independente de plataforma, facilitando a portabilidade. Com isso, o LeJOS fornece uma linguagem de programação orientada à objetos (Java), que possui recursos que podem ser aproveitados em sensores e na robótica, tais como threads de preferência, sincronização, suporte a fluxo de dados utilizando Bluetooth, vetores multi-dimensionais e etc. A MVJ permite que códigos Java sejam escritos de forma nativa para o robô programável Lego Mindstorms.

3. Aplicações embarcadas reais

Na parte prática do trabalho, foi utilizado um robô educacional Lego Mindstorms, que possui um hardware poderoso e baixo custo. Foram realizados testes com ambos os S.O., sendo que o escolhido foi LeJOS para experimentos mais sofisticados, por permitir a programação na linguagem Java, que é bastante difundida e tem mais recursos disponíveis. Em seguida, iniciou-se o desenvolvimento de aplicações que exploraram os recursos disponíveis, tais como sensores de toque, luz, distância, som e motores para locomoção. O resultado foi feito em contexto de aplicações eficientes pensadas para sistemas embarcados em ambientes críticos, tais como: locomoção à distância (através de Bluetooth), braço mecânico para pegar objetos, obter níveis de ruídos e claridade em ambientes, encontrar caminhos através de obstáculos entre outros.

4. Conclusão

O desenvolvimento de aplicações para redes de sensores tem uma curva de aprendizagem grande devido à complexidade de embarcar essas usando SOs existentes, pois cada sistema embarcado pode ter diferentes linguagens suportadas. Em contrapartida, usando a linguagem Java, é possui acessar um vasto campo de aplicações, seja na área comercial ou científica. Este trabalho em curso pretende fazer uso de Java em sensores de forma satisfatória, e trata a idéia futura de realizar a integração entre Java e simuladores de redes (como o Qualnet), incluindo sua API e sistema de eventos, de forma a estudar como se portariam aplicações Java envolvendo milhares de sensores. Se for comprovada sua eficiência dessa forma, será uma inovação na área de RSSF, talvez um incentivo para que mais pessoas da área se interessem pelo assunto.

Referências

Gay, D., Levis, P. Culler D. e Brewer, E. (2003), “nesC 1.1 Language Reference Manual”.

Bagnall, B. (2007), “Building Robots with Java Brains”, Variant Press, Canadá.

Sessão 2

Clusters e Grades Computacionais

ViCOS - Virtual Cluster Orchestration System

Henrique F. Baggio[1], **Raul Kist**[1], **Sandro Rigo**[1]

[1]Microsoft Innovation Center
State University of Campinas - Unicamp
Campinas, São Paulo, Brazil

{henrique.baggio, raul.kist}@students.ic.unicamp.br, sandro@ic.unicamp.br

Abstract. *ViCOS aims to provide a heterogeneous infrastructure to process high performance applications on demand, providing processing capabilities in both MPI and SOA applications in several platforms (Windows HPC Server, Suse Linux Enterprise Server, etc) using the same infrastructure. The main goal is to make an intelligent use of idle processing of the environment focusing both on performance increase and energy savings making use of cloud computing concepts to ensure a flexible, economic, and reliable environment for performing high performance demanding tasks.*

1. Introduction

While virtualization has gained significant momentum in the enterprise domain, for consolidation, cost and power reduction, ease of management, improved fault-tolerance and RAS properties, its adoption in the high performance domain remains lagging despite several works [Vallee et al. 2008, Huang et al. 2006, Mergen et al. 2006, Youseff et al. 2006] have proved that applications can achieve almost the same performance as those running in a native, non-virtualized environment.

ViCOS is an *environment aware load balancing virtual cluster system* that coordinates virtual machines that offer some service (either HPC, SOA, etc) in a Failover Cluster. In order to do that, it uses the available idle machines, occupying them with virtual machines, turning them on or off on demand and, at the same time, adapting to environment changes: if a host is to be used for other purposes, the system migrates the VMs to another host, so the task isn't interrupted and can be resumed elsewhere, managing the physical resources without interrupting the work load. It fits itself to the environment, expanding and contracting based on resource demand.

2. Solution Architecture

The ViCOS system is deployed over a special infrastructure built over virtualization and failover clustering with live migration capabilities. It was built using Microsoft Hyper-V Server and Windows Server 2008 Core.

ViCOS consists of a set of services that runs on each failover cluster's node and on each HPC cluster's headnode and monitors the physical and virtual environment. It is built using .NET framework and communicates with the system using Windows Management Instrumentation (WMI) and Powershell scripts.

On the failover cluster's nodes, the service can be a leader or a non-leader, the election is made using a slightly modified Bully Algorithm [Garcia-Molina 1982]. As a non-leader, the service monitors its own changes and informs the leader if any, if it cannot

reach a leader, it calls a new election, following the Bully Algorithm, so the system always has a leader when it needs. As a leader, the service monitors its own changes and listen for others requests.

On the virtual cluster's headnode, the service monitors HPC jobs requisitions and terminus and informs the leader about the necessity of more or less virtual nodes.

The leader collects environment data when informed about a change, provisioning the necessary resources, activating or suspending the virtual nodes. Also, it tries to get the best VM placement in virtualization hosts according with some determined scheduling schema, which can, for instance, be power saving or performance gain. When requested for more or less nodes by a virtual headnode, the number of active VMs changes and they can be moved among the hosts according to the system's profile using a less effort algorithm, since migration is a network load expensive operation.

3. Conclusions

ViCOS provided use of idle processing of the environment for high performance applications using a flexible infrastructure built over virtualization and failover clustering that provides high availability, save state capability, isolation and security to the virtual clusters. It also enables a multi-cluster environment: the same infrastructure can be used to manage several clusters with different architectures and operating systems at same time.

Moreover, the distributed implementation of the management layer allows the addition and the removal of VM hosts easily, while the system can adjust itself to such changes. As a self-managed system [Kramer and Magee 2007], it aims to be as *autonomic* as possible, quite minimising the degree of explicit management of the environment.

Some experiments using the HPL benchmark with the load balance schema showed a 25 to 45% processing time reduction with different values of problem and cluster size, as a result of performance improvement.

References

Garcia-Molina, H. (1982). Elections in a distributed computing system. *IEEE Trans. Comput.*, 31(1):48–59.

Huang, W., Liu, J., Abali, B., and Panda, D. K. (2006). A case for high performance computing with virtual machines. In *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*, pages 125–134, New York, NY, USA. ACM.

Kramer, J. and Magee, J. (2007). Self-managed systems: an architectural challenge. In *FOSE '07: 2007 Future of Software Engineering*, pages 259–268, Washington, DC, USA. IEEE Computer Society.

Mergen, M. F., Uhlig, V., Krieger, O., and Xenidis, J. (2006). Virtualization for high-performance computing. *SIGOPS Oper. Syst. Rev.*, 40:8–11.

Vallee, G., Naughton, T., Engelmann, C., Ong, H., and Scott, S. L. (2008). System-level virtualization for high performance computing. *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, 0:636–643.

Youseff, L., Wolski, R., Gorda, B., and Krintz, R. (2006). Paravirtualization for hpc systems. In *Proc. Workshop on Xen in High-Performance Cluster and Grid Computing*, pages 474–486. Springer.

Experimentos de Gerenciamento de Clusters Heterogêneos e Não Dedicados para Processamento Distribuído

Elaine N. Watanabe[1], Claudio D. Marins[2], Nancy M. Abe[2], Angelo Passaro[2]

[1]Universidade Braz Cubas (UBC)

[2] Laboratório de Engenharia Virtual
Instituto de Estudos Avançados (IEAv) –São José dos Campos, SP – Brasil
{elaine.n.watanabe, cdmbr85}@gmail.com, {nancy, angelo}@ieav.cta.br

Resumo. *Este trabalho apresenta o estágio atual de desenvolvimento de um sistema, denominado ROME, para o gerenciamento de processamento distribuído em clusters heterogêneos e não dedicados. O núcleo do sistema utiliza a linguagem de programação Java e a tecnologia de comunicação RMI. Ferramentas para monitoração gráfica do processamento distribuído estão em desenvolvimento, utilizando a linguagem C++ orientada a objetos e pacotes gráficos multiplataforma.*

1. Introdução

O sistema ROME foi desenvolvido com o objetivo inicial de gerenciar clusters heterogêneos e não dedicados para a realização de cálculos em paralelo para aplicações de grão grosso. Um dos principais focos do projeto foi permitir o balanceamento de carga e tolerância a falhas em tempo de execução, com um mínimo investimento de tempo de programação por parte do usuário. O sistema permite a utilização em clusters compostos por computadores com diferentes configurações de hardware, assim como diferentes sistemas operacionais e explora as várias unidades de processamento (UP) dos modernos processadores multinúcleo, Marins (2007), Abe (2009).

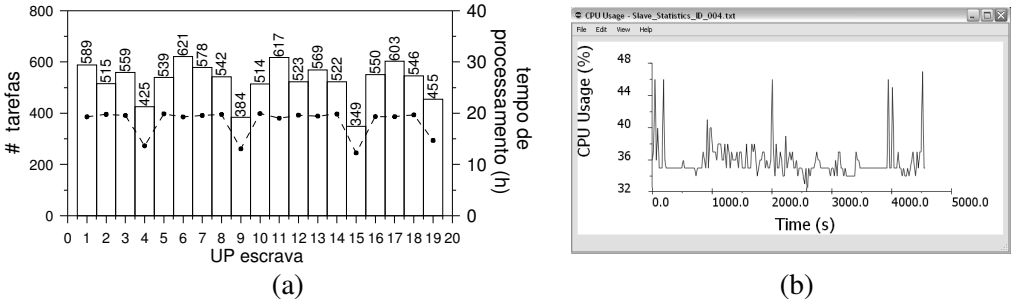
O sistema foi desenvolvido utilizando a linguagem Java e tecnologia RMI (Remote Method Invocation), a qual possibilita invocar remotamente um método de um objeto em outra JVM (Java Virtual Machine). O ROME é composto por mais de quarenta classes, responsáveis pelo controle da maioria dos processos que envolvem a interação entre as várias UP, tais como, seleção das UPs da rede que se encontram ociosas para composição de um cluster durante a execução da tarefa, identificação do sistema operacional de cada UP escrava, distribuição de tarefas e de respostas, incluindo arquivos de dados e códigos para execução, de acordo com o sistema operacional da UP escrava, estimativa de tempo de resposta das UPs escravas e verificação de resposta na faixa de tempo prevista. As tarefas podem ser iniciadas a partir de qualquer UP da rede na qual esteja sendo executado uma aplicação mestre. Qualquer UP que esteja executando uma aplicação escrava pode responder a uma requisição da aplicação mestre para compor o cluster e participar dos cálculos. As aplicações mestre e escrava são componentes do sistema ROME e podem ser executadas na mesma unidade de processamento. A tarefa do usuário para utilizar o ROME é derivar duas classes a partir de classes abstratas, disponíveis no sistema, que definem a sua própria aplicação (aplicação mestre do usuário e aplicação escrava do usuário).

2. Aplicações

O sistema ROME foi utilizado recentemente em aplicações de grão grosso : otimização computacional de moduladores eletroópticos e de dispositivos semicondutores nanoestruturados, utilizando algoritmos genéticos, Passaro (2010). No segundo caso, o tempo de execução de um indivíduo pode variar de alguns minutos até mais de uma hora. Na Figura 1a são apresentados a distribuição de tarefas e o tempo de processamento para as vinte UPs escravas utilizadas na otimização de uma nanoestrutura semicondutora.

3. Comentários Finais

No estágio atual, o ROME não permite a comunicação direta entre processos de uma mesma simulação, sendo mais adequado para processamento paralelo de grão grosso. Mecanismos de troca de mensagens entre processos distribuídos, visando a aplicação do ROME em simulações computacionais, por exemplo, na área de plasmas, serão foco dos próximos desenvolvimentos. Uma interface gráfica multiplataforma para entrada de dados e monitoração do andamento do processamento distribuído está em desenvolvimento (Figura 1b) utilizando a linguagem C++ com pacotes gráficos multiplataforma de uso livre. Esses recursos permitirão a monitoração do processamento paralelo em tempo de execução, o que não era possível nas versões anteriores, e serão particularmente úteis para monitoração da troca de mensagens diretas entre os processos executados nas UPs escravas de uma mesma simulação.



(a) (b)

Figura 1. Tarefas processadas e tempo de processamento por UP escrava (a) e exemplo da interface gráfica em desenvolvimento para monitoração em tempo de execução de uma UP escrava (b).

Referências

Marins, C.D. (2007). Proposta de um Sistema Robusto para Gerenciamento de Processamento Distribuído em Ambientes Heterogêneos. *Trabalho de Graduação*, Curso de Engenharia da Computação, Universidade Braz Cubas.

Abe, N.M., Marins, C.D. and Passaro, A. (2009) Distributed Processing Management using ROME. In: COMPUMAG 2009, Florianópolis. Proceedings of the 17th Conference on the Computation of Electromagnetic Fields. p. 1060-1061.

Passaro, A., Tanaka, R.Y., Muraro Jr, A., Vieira, G.S. and Abe, N.M. (2010). Self-consistent Optimization of Multi-Quantum Well Structures by a Genetic Algorithm. In *IEEE Transactions on Magnetics*. Aceito para publicação, Agosto, 2010.

Interpretador de Modelos Externos Para Simulador de Grades Computacionais[1]

Victor Aoqui, Aldo Ianelo Guerra, Marco Antonio Barros Alves Garcia, Paulo Henrique Maestrello Assad Oliveira, Renata Spolon Lobato, Aleardo Manacero Júnior

Departamento de Ciências de Computação e Estatística – Instituto de Biociências, Letras e Ciências Exatas - Universidade Estadual Paulista "Júlio de Mesquita Filho" (IBILCE/UNESP)
Caixa Postal 136 – 15.054-000 – São José do Rio Preto – SP – Brasil

{aoqui, aldoiguerra, oliveiraph, magarcia}@sjrp.unesp.br, {renata, aleardo}@ibilce.unesp.br

1. Introdução

A utilização de grades computacionais tem aumentado gradativamente ao longo dos anos, o que gera uma demanda por ferramentas adequadas que simulem e avaliem o desempenho destes sistemas [Coulouris *et al*. 2005] [Reis 2005]. Em geral, estas ferramentas disponíveis atualmente não possuem interface amigável para o usuário e geram um modelo que deve ser utilizado exclusivamente por sua própria ferramenta.

Visando sanar tais deficiências, o projeto, como um todo, foi subdividido em partes a fim de desenvolver uma plataforma de simulador de grades computacionais inovadora. Através desta plataforma, o usuário criará um modelo icônico da grade computacional, que será posteriormente convertida para um modelo simulável através de um interpretador. Este modelo, escrito na forma de *script*, será simulado para disponibilizar métricas de desempenho interessantes ao usuário. Além disso, esta ferramenta interpretará modelos externos de outros simuladores existentes (SimGrid, GridSim e OptorSim) e será de fácil utilização por possuir interface icônica.

2. Objetivo

Este resumo retrata a parte do projeto responsável pela especificação de uma gramática para o modelo simulável e outra para conversão de modelos externos. Por fim, foi especificado, implementado e testado um protótipo para o interpretador de modelos externos.

3. Metodologia

Para atingir os objetivos retratados, adotou-se uma metodologia dividida em três etapas, a saber:

- a primeira etapa consistiu em especificar as gramáticas regular e livre de contexto para o modelo simulável em notação BNF (*Backus-Naur Form*);
- a segunda etapa consistiu em especificar as gramáticas regular e livre de contexto para um modelo externo (SimGrid) e auxiliar na especificação das

[1] Este trabalho foi financiado pela FAPESP através dos processos 2009/00160-2, 2009/00502-0 2009/00182-6, 2009/00183-2 e 2008/09312-7.

gramáticas regular e livre de contexto para o modelo icônico;

- a última etapa consistiu em especificar, implementar e testar o protótipo para o interpretador de modelos externos que, além do próprio interpretador, consiste em analisadores léxico, sintático e semântico para as gramáticas do SimGrid implementados através da ferramenta JavaCC 5.0 [Delamaro 2004] e da linguagem Java [Deitel and Deitel 2006], utilizando o compilador JDK 1.6.0 Update 20. Salienta-se que todas estas ferramentas são gratuitas.

4. Resultados

A aplicação da metodologia apresentada permitiu a obtenção do interpretador de modelos externos, o qual será integrado ao restante do sistema, permitindo ao usuário realizar a simulação de grades computacionais por meio de uma interface icônica.

Para realizar a interpretação do modelo externo (SimGrid) para o modelo icônico, o analisador sintático realiza uma leitura do arquivo de especificação do modelo SimGrid, filtra os dados relevantes e estes são armazenados em objetos, que são instâncias das classes Server, Master, Network e Route.

Além disso, existe a classe SimGrid, que armazena todas as classes, possui métodos para manipular os arquivos de entrada e saída e define a main(), que recebe os arquivos application_file.xml e plataform_file.xml como entrada, chama o *parser* através do método modelo() e, após o processamento do *parser* (leitura dos arquivos de entrada e armazenamento dos dados nos objetos), gera o modelo de saída (modelo icônico), concluindo a interpretação do modelo externo.

5. Conclusões

A partir das gramáticas definidas, utilizou-se a ferramenta JavaCC para construir os analisadores léxico e sintático para as gramáticas do modelo SimGrid e implementar o protótipo do interpretador de modelos externos, que atualmente reconhece apenas modelos SimGrid.

Para trabalhos futuros, espera-se que este interpretador seja capaz de reconhecer modelos de outros simuladores além do SimGrid como, por exemplo, GridSim e OptorSim, e seja totalmente integrado à interface de forma a possibilitar ao usuário montar uma plataforma de simulação a partir dos ícones ou importar modelos de outros simuladores a fim de gerar automaticamente o modelo icônico correspondente.

Referências

Coulouris, G., Dollimore, J. and Kindberg, T. (2005). “Distributed Systems: Concepts and Design”. 4 ed. Harlow: Addison-Wesley.

Deitel, H. M. and Deitel, P. J. (2006). “Java how to program”. 7 ed. Upper Saddle River: Prentice Hall.

Delamaro, M. E. (2004). “Como Construir Um Compilador Utilizando Ferramentas Java”, São Paulo: Novatec.

Reis, V. Q. (2005). “Estudo em grids computacionais: estudo de caso”. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional). São Carlos: Instituto de Ciências Matemáticas e de Computação – USP.

Plataforma de Simulação de Grades Computacionais: Interface Icônica[1]

Aldo Ianelo Guerra, Paulo Henrique Maestrello Assad Oliveira, Marco Antonio Barros Alves Garcia, Victor Aoqui, Aleardo Manacero Junior, Renata Spolon Lobato

Departamento de Ciências de Computação e Estatística - Instituto de Biociências, Letras e Ciências Exatas - Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP)
São José do Rio Preto - SP - Brazil

{aldoiguerra,oliveiraph,magarcia,aoqui}@sjrp.unesp.br,
{aleardo,renata}@ibilce.unesp.br

1. Introdução

Com o aumento da necessidade de poder de processamento em aplicações, a procura por computação de alto desempenho tem crescido, e com isso surge a necessidade de avaliar os sistemas que realizam esse tipo de computação. Dentre as técnicas de avaliação existentes a simulação é a que possui o melhor custo benefício.

Levando-se em consideração os simuladores de grade computacional existentes, esses exigem do usuário um prévio conhecimento de linguagens de programação para a elaboração de um modelo que possa ser simulado. Este trabalho está inserido no contexto do projeto de criação de um simulador de grades computacionais e descreve a criação de uma interface baseada em ícones para elaboração de modelos de grades computacionais simuláveis, a fim de facilitar ao máximo esse processo para o usuário. A interface irá interagir com o simulador por meio de linguagens criadas para este fim.

2. Fundamentação Teórica

Para a elaboração deste trabalho foram estudados simuladores como, por exemplo, Simgrid, Bricks e OptorSim, que simulam grades computacionais mas exigem do usuário um prévio conhecimento em linguagens de programação, esses simuladores já haviam sendo estudados por membros do Grupo de Sistemas Paralelos e Distribuídos, grupo de pesquisa no qual o projeto está inserido [Oliveira 2008]. Com base nesses simuladores foram abstraídos os parâmetros necessários para realizar a construção de um modelo de grade computacional, e esses parâmetros foram inseridos na forma de ícones na interface do sistema para que o usuário possa criar um modelo de forma simples e fácil.

A interface com o usuário é um requisito de vital importância e o estudo da interação humano-computador auxilia o desenvolvimento da interface, de modo que os usuários sintam-se satisfeitos no uso do sistema. O estudo da interação humano-computador considera quatro elementos básicos: o sistema, os usuários, os desenvolvedores e o ambiente de uso [Dix *et al.* 2003].

[1] Este trabalho foi financiado pela FAPESP através dos processos 2009/00160-2, 2009/00502-0 2009/00182-6, 2009/00183-2 e 2008/09312-7.

Segundo [Rogers *et al.* 2007], algumas características como consistência, facilidade de aprendizagem e utilização, satisfação, manifestação de erros, suporte, familiaridade, simplicidade, incentivo, acessibilidade, versatilidade devem ser consideradas no projeto de interface.

3. Resultados e Discussões

No desenvolvimento da interface foram estudados todos os parâmetros necessários para efetuar a simulação, e de acordo com esses parâmetros foram criados quatro ícones que são essenciais para a elaboração de um modelo de grade computacional. O ícone *machine* representa um nó de processamento ou um nó de escalonamento dependendo de sua configuração, o ícone *network* representa as conexões de rede estabelecidas entre os nós do modelo, o ícone *cluster* representa um bloco fechado com um determinado número de maquinas simbolizando um grande poder de processamento e o ícone *internet* representa diversas interconexões entre os diversos nós do sistema que estão ligados a ele.

Cada ícone, ao ser adicionado no modelo elaborado pelo usuário, pode ser configurado com determinados parâmetros que serão utilizados pelo motor de simulação para realização do mesmo.

A interface possui uma área de desenho na qual o usuário pode posicionar o ícone onde desejar dentro desta área, e montar seu modelo de grade computacional da maneira mais fácil possível, e o usuário pode inserir nesta área uma grade para visualizar a posição do ícone que está inserindo. A configuração dos parâmetros pode ser feita a qualquer momento pressionando o botão do *mouse* duas vezes sobre o ícone, inserido no modelo, que deseja configurar. Os parâmetros das tarefas necessários para a simulação podem ser configurados através do botão localizado ao lado dos botões de ícones para facilitar o acesso do usuário.

4. Conclusões

A interface criada a partir de todo estudo realizado permite ao usuário criar, de maneira fácil e rápida, um modelo simulável. Essa facilidade e rapidez trazem a satisfação em utilizar a interface, além dos recursos incluídos para mostrar ao usuário respostas claras e objetivas como, por exemplo, janelas que informando os erros que o usuário cometer.

Referências

Dix, A., Finlay, J., Abowd, G. D., and Beale, R. (2003). “Human-Computer Interaction”. Prentice Hall.

Oliveira, L. (2008). “Comparação de ferramentas de simulação de grades computacionais”. Technical report, Grupo de Sistemas Parelelos e Distribuídos - UNESP, disponível em http://www.dcce.ibilce.unesp.br/spd/pubs/gridsimulation tools.pdf.

Rogers, Y., Sharp, H., and Preece, J. J. (2007). “Interaction Design: Beyond Human-Computer Interaction”. Wiley.

Escalabilidade de Aplicações Bag-of-Tasks em plataformas master-slave utilizando SimGrid

Luciano J. Miranda Filho, Hermes Senger
Departamento de Computação - Universidade Federal de São Carlos (UFSCar)
e-mail: lucianojmf@comp.ufscar.br , hermes@dc.ufscar.br

Resumo. *Este artigo tem como objetivo avaliar a escalabilidade de aplicações Bag-of-Tasks em grids computacionais master-slave com base na função de Isoeficiência como métrica. Os experimentos feitos com o simulador SimGrid mostram que a função de isoeficiência é $\Omega(P)$ para redes com broadcast e $\Omega(P^2)$ para redes de pacotes baseadas nos modelos TCP-LAN e TCP-WAN.*

1. Introdução

Grids são plataformas computacionais de alto poder de processamento e manipulação de grandes volumes de dados, e permitem o compartilhamento de uma grande variedade de recursos. Aplicações *Bag-of-Tasks* (ou BoT) são formadas por tarefas computacionais independentes que não se comunicam entre si e podem ser executadas em uma ordem arbitrária [1]. Além disso, as tarefas podem compartilhar arquivos de entrada e dependem da disponibilidade desses arquivos nos nós.

Diversas plataformas que executam aplicações BoT, como OurGrid [1] e Condor [5], adotam a arquitetura denominada *master-slave,* onde o mestre controla a execução e o disparo das tarefas nos demais nós da rede (escravos), que processam e devolvem arquivos de saída. Tais aplicações são frequentes em diversas áreas da ciência [2]. Este trabalho tem como objetivo avaliar a escalabilidade de aplicações BoT em grades computacionais dispostas em uma arquitetura *master-slave,* considerando um cenário onde todas as tarefas dependem dos mesmos arquivos (i.e, máximo compartilhamento).

2. Escalabilidade

Escalabilidade é a capacidade de aumentar o *speedup* de uma aplicação conforme aumenta-se o número de processadores [3]. Uma métrica muito utilizada na avaliação de escalabilidade é a função de isoeficiência $\boldsymbol{F(P)}$, que mede a quantidade de trabalho que deve ser acrescentada a uma aplicação para que o valor de eficiência **E** mantenha-se em um certo patamar à medida que aumenta o número de processadores no Grid. A eficiência computacional é definida pela expressão $\mathbf{E = (T_{seq}/Tp)/P}$, onde $\mathbf{T_{seq}}$ é o tempo de execução sequencial, e **Tp** é o tempo de execução em **P** processadores. Neste trabalho a função de isoeficiência será utilizada para a análise de escalabilidade.

3. Experimentos realizados

O SimGrid [4] é uma ferramenta que permite simular a execução de aplicações em ambientes heterogêneos distribuídos, tais como *grids, clusters* e *clouds*. Para a arquitetura *master-slave,* desenvolvemos um ambiente capaz de simular tarefas BoT em 4 diferentes modelos de comunicação: *oneport*, *broadcast,* TCP-LAN e TCP-WAN. No modelo *oneport***,** assume-se que as transferências de um arquivo entre mestre e escravos não sejam simultâneas. O mestre transmite uma única mensagem por vez,

podendo processar tarefas enquanto transmite. Embora conceitualmente simples, este modelo aproxima-se do comportamento de uma rede local [2]. Ao contrário do anterior, no modelo *broadcast* o mestre pode transferir arquivos para todos os escravos simultaneamente. Utilizamos o SimGrid para simular também o comportamento de enlaces TCP em ambientes de redes locais e de longa distância, com os modelos TCP-LAN e TCP-WAN**.** A relação entre o tempo de transferência de arquivos de uma tarefa e o tempo de execução, denomina-se **ccr** (*communication-to-computation ratio).* Nos experimentos utilizamos o valor de **ccr**=0.1(representado aplicações que gastam 10% do tempo comunicando), uma arquitetura de até 1000 processadores, e a eficiência **E=**0.5.

4. Resultados

Os resultados das simulações são ilustrados no gráfico da **Figura 1**.

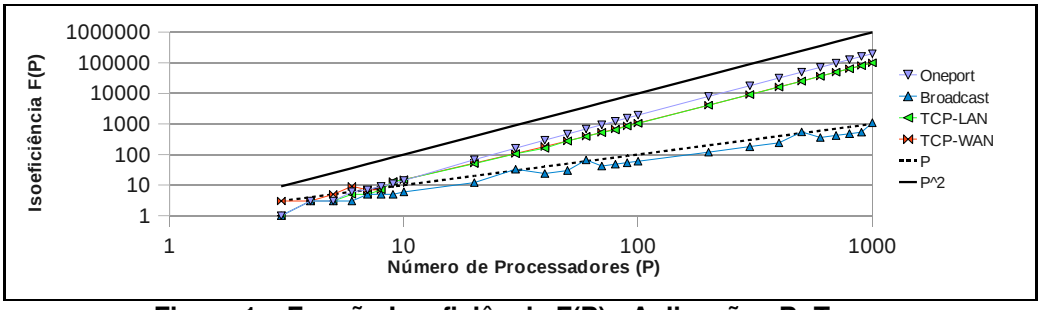


Figura 1 – Função Isoeficiência F(P) - Aplicações BoT.

5. Conclusão

No presente trabalho avaliamos a escalabidade de aplicações BoT em plataformas *master-salve*, supondo um cenário de máximo compartilhamento de arquivos entre as trarefas. Como contribuição, os experimentos mostram que no modelo *oneport* a função F(P) é $\Omega(P^2)$, confirmando os resultados demonstrados em [6]. Além disso, os resultados mostram que F(P) = $\Omega(P^2)$ também para redes *TCP-LAN* e *TCP-WAN. Já* para o modelo *broadcast*, a função de isoeficiência é $\Omega(P)$ (ou seja, segue uma tendência linear em P).

6. Referências

[1] Cirne, W., Paranhos, D., Costa L., Neto,E.S., Brasileiro, F.V., Sauvé, J., Silva,F.A.B., Barros, C.O., Silveira, C.. Running Bag-of-Tasks Applications on Computational Grids: The MyGrid Approach. In: Intl. Conf. Parallel Processing, 2003

[2] Beaumont, O., Carter, L., Ferrante, J., Legrand, A., Marchal, L., Robert, Y.: Scheduling multiple bags of tasks on heterogeneous master-worker platforms: centralized versus distributed solutions. Research Report No 2005-45, École Normale Supérieure de Lyon, 2005.

[3] Grama, A., Gupta, A., Kumar, V.: Isoefficiency: Measuring the Scalability of Parallel Algorithms and Architectures. IEEE Parallel and Dist. Tech., v.1, n. 3 (1993).

[4]http://simgrid.gforge.inria.fr/.

[5]M. Litzkow, M. Livny, M. Mutka. Condor–A Hunter of Idle Workstations. In Intl. Conference of Distributed Computing Systems, pp. 104-111, 1988.

[6] Silva, F.A.B., Senger H, Improving scalability of Bag-of-Tasks applications running on master-slave platforms. Parallel Computing, v. 35, p. 57-71, Elsevier, 2009.

O Motor de uma Plataforma de Simulação de Grades Computacionais[1]

Paulo Henrique Maestrello Assad Oliveira, Aldo Ianelo Guerra, Marco Antonio Barros Alves Garcia, Victor Aoqui, Renata Spolon Lobato, Aleardo Manacero Junior.

Departamento de Ciências de Computação e Estatística - Instituto de Biociências Letras e Ciências Exatas – Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP) - São José do Rio Preto – SP

{oliveiraph, aldoiguerra, magarcia, aoqui}@sjrp.unesp.br, {renata, aleardo}@ibilce.unesp.br.

Resumo. *Este resumo expandido aborda a motivação e os conceitos necessários para o desenvolvimento do motor de uma ferramenta de simulação de grades computacionais. Descreve também, sucintamente, os resultados obtidos com a primeira implementação deste motor, finalizando com a indicação de quais serão os próximos passos do trabalho.*

1. INTRODUÇÃO

O desenvolvimento de ferramentas para avaliar sistemas de computação de alto desempenho é imprescindível, pois atualmente há uma grande demanda por maior poder computacional, já que as aplicações computacionais estão cada vez mais complexas e manipulam um volume de dados crescente.

Entre as técnicas de construção dessas ferramentas está a simulação. Ela apresenta vantagens como: o sistema a ser simulado pode estar em desenvolvimento, os custos com simulação são menores e a credibilidade do método é satisfatória.

O projeto no qual este trabalho está inserido tem por objetivo o desenvolvimento de uma nova ferramenta de simulação de grades computacionais que procura sanar problemas - como a dificuldade de uso e a não geração de modelos gráficos - presentes nas ferramentas existentes. Enquanto isso, este trabalho em particular pretende levantar as funcionalidades desejáveis ao simulador, especificar, implementar e validar o motor de simulação.

2. FUNDAMENTAÇÃO TEÓRICA

Grades computacionais podem ser modeladas como sistemas discretos não determinísticos, uma vez que seu estado só é alterado com a ocorrência de certos eventos que, por sua vez, ocorrem de forma relativamente aleatória. Logo, uma boa técnica para modelar tais eventos é através de sistemas de filas.

Uma rede de filas é composta por centros de serviço, constituídos de um ou mais servidores, e um conjunto de usuários que recebem serviços nestes centros. Os usuários esperam em filas, que podem ser de diferentes tipos, pelo serviço requisitado. [BANKS, 2001].

[1]Este trabalho foi financiado pela FAPESP através dos processos 2009/00160-2, 2009/00502-0 2009/00182-6, 2009/00183-2 e 2008/09312-7.

O principal intuito da ferramenta é simular a infraestrutura das grades e como esse ambiente atende uma determinada carga de trabalho, exatamente como um servidor faz com uma tarefa, reforçando, assim, a escolha dessa técnica.

Para este caso concreto, a estruturação das filas é feita pela ferramenta, ou seja, de modo transparente para o usuário que é responsável apenas por entrar como os dados do ambiente físico da grade e da carga de trabalho que será executada nela e esperar os resultados da simulação. Desta maneira, soluciona-se a principal restrição da técnica que é a necessidade de um bom projetista.

3. RESULTADOS E DISCUSSÕES

Preliminarmente, foi realizado um estudo acerca dos sistemas distribuídos e de técnicas de simulação. A partir daí, foi necessário abstrair uma maneira para representar os sistemas distribuídos como redes de filas, sendo que os modelos "uma fila, um servidor" simulam as redes de comunicação entre os diversos nós das plataformas computacionais; os *clusters* são modelados pelas redes "uma fila, vários servidores". Já as redes "várias filas, vários servidores" modelam as grades computacionais.

Num passo subsequente, as distribuições de probabilidade que devem ser geradas para modelar as taxas de chegadas de clientes (tarefas e comunição) nas filas e de atendimento dos servidores (redes de comunicação e plataformas computacionais) foram determinadas. Isso foi feito com o auxílio de resultados anteriormente publicados, como o de [LUBLIN e FEITELSON, 2003]. Entre elas, é possível citar Poisson, exponencial e *two-stage uniform*.

Para se chegar à fase de implementação da ferramenta foi necessário determinar, através do estudo de outros simuladores, seus parâmetros de entrada, isto é, os parâmetros que serão manipulados pelo usuário. Alguns deles são: poder computacional (total ou individual), função (mestre ou escravo) do *host* na plataforma, política de escalonamento das tarefas, estrutura de conexão do sistema (ponto-a-ponto, nuvem), largura de banda e latência da rede de comunicação e tamanho das tarefas.

4. CONCLUSÕES E TRABALHOS FUTUROS

Considera-se que a primeira implementação do motor de simulação foi bastante satisfatória, pois além dos testes terem mostrado que ele é funcional, as decisões tomadas facilitam sua expansão na próxima fase do projeto.

Os próximos passos do trabalho consistem em implementar a expansão das redes de filas e validar o sistema.

5. REFERÊNCIAS BIBLIOGRÁFICAS

BANKS, J. CARSON, J. S., NICOL, D. M., NELSON B. L., Discrete-Event System Simulation, 3ª edição, Prentice-Hall, International Series In Industrial and Systems Engineering, 2001.

LUBLIN U., FEITELSON, D. G. The workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. Journal of Parallel and Distributed Computing, Volume 63, fascículo 11, páginas 1105-1122, Novembro de 2003.

Sessão 3

Análise de Desempenho, Otimização e Multi-Core

Modelo de Simulação e Desempenho de Filas em Java

Mariana G. V. Miano[1], Marcus V. Ap. Camargo[1], Renato H. Pires[1]

[1]Curso de Análise de Sistemas e Tecnologia da Informação – Centro Estadual de Educação Tecnológica "Paula Souza" – Americana, SP – Brasil

{mariana.miano,marcus.camargo,renato.pires}@fatec.sp.gov.br

Abstract. *This article discusses about the development of a simulation system that applies the Queue theory. The applied Queue models were M/M/1 and M/M/m. The simulation system was implemented in Java and can be handled by users who have no expertise on advanced simulation systems. Finally it presents a case study as validation.*

Resumo. *Este artigo versa sobre o desenvolvimento de um sistema de simulação aplicando-se a Teoria de Filas. Os modelos de Filas utilizados foram M/M/1 e M/M/m, implementados na linguagem Java e pode ser utilizado por usuários que não possuam conhecimentos técnicos avançados sobre simulação de sistemas. Por fim, apresenta como validação um estudo de caso.*

1. Introdução

A Teoria de Filas [Fogliatti e Mattos 2007] consiste na modelagem analítica de processos ou sistemas que resultam em espera e tem como objetivo determinar e avaliar quantidades, denominadas medidas de desempenho (número de elementos na fila, tempo de espera pelo atendimento e tempo ocioso dos prestadores de serviço, dentre outros), que expressam a produtividade/operacionalidade desses processos [Freitas 2008].

Tais medidas são importantes para auxiliar nas tomadas de decisão, no que diz respeito à modificação ou manutenção da operabilidade do sistema e no dimensionamento de recursos de infra-estrutura, pessoal e finanças [Souza, Cortez e Senger 2008].

Este artigo apresenta um sistema computacional, desenvolvido em linguagem Java, que ilustra um estudo de caso apresentado em [Fogliatti e Mattos 2007] e que também foi utilizado para validá-lo, pois é um sistema genérico. Os resultados gerados pelo sistema são fundamentais no processo de tomada de decisão gerencial.

2. Modelos de Filas

Há vários modelos de filas descritos em [Fogliatti e Mattos 2007]. Neste artigo, os modelos utilizados são o M/M/1 e o M/M/m. No modelo de fila M/M/1 os tempos entre chegadas sucessivas e os tempos de atendimento seguem distribuições exponenciais, sendo que as chegadas e atendimentos caracterizam um processo de nascimento e morte. A característica de maior destaque neste modelo de fila é a presença de apenas um atendente/servidor. O modelo M/M/m apresenta as mesmas características do modelo M/M/1, mas nesse caso, o número de atendentes/servidores é maior que um.

3. Simulação e Projeto Computacional

O projeto computacional desenvolvido recebeu o nome de SimulaCaixa 1.0, porque teve como base um estudo de caso sobre um Sistema de Pagamento de Mercadorias em uma Loja de Departamentos, apresentado em [Fogliatti e Mattos 2007], entretanto o sistema pode ser utilizado para simular qualquer outra aplicação de Desempenho de Filas.

A modelagem é analítica, com variáveis aleatórias e processos estocásticos, caracterizando um "sistema estocástico" que apresenta variações aleatórias no seu estado ao longo do tempo, sendo assim um sistema dinâmico em suas variáveis de estado.

O sistema foi desenvolvido em linguagem Java [Deitel e Deitel 2005], utilizando a API JFreeChart, para auxiliar na geração de gráficos. O sistema SimulaCaixa 1.0 pode ser utilizado por simples usuários ou especialistas. A tela inicial do sistema requisita os parâmetros de entrada para o processo de simulação (demanda, taxa de saída, número de atendentes/servidores) e o preenchimento da tabela de distribuição de frequência para posterior geração do gráfico de dispersão. De acordo com o número de atendentes/servidores, o sistema identifica o modelo de fila utilizado e direciona o fluxo de processamento para os métodos das classes responsáveis por efetuar os cálculos.

Os métodos implementados efetuam os seguintes cálculos: 1) Intensidade de tráfego; 2) Avaliação da condição de estabilidade do sistema; 3) Probabilidade de não haver clientes no sistema; 4) Probabilidade de enfileiramento (M/M/m); 5) Número médio de clientes no sistema; 6) Variância do número de clientes no sistema; 7) Número médio de clientes na fila; 8) Variância do número de clientes na fila; 9) Tempo médio no sistema (ou de resposta); 10) Variância do tempo no sistema; 11) Tempo médio de espera na fila; 12) Variância do tempo de espera na fila.

A cada nova simulação e/ou alteração da tabela de distribuição de frequência, os resultados são reapresentados, assim como os gráficos. A partir desses resultados, o analista/gerente de desempenho poderá avaliar as possíveis alterações que podem ser realizadas, visando uma melhor produtividade/operacionalidade do processo.

Com este desenvolvimento evidenciou-se a eficiência da simulação aliada à operacionalidade computacional, uma vez que a avaliação do comportamento dos modelos se faz com uma formulação matemática bastante complexa. O sistema mostrou-se amplamente utilizável para simular situações de fila M/M/1 e M/M/m em qualquer cenário (bancos, pedágios, lojas, supermercados).

Referências

Deitel, H. M. e Deitel, P. J. (2005) "Java: Como Programar", Editora Prentice-Hall, Brasil.

Fogliatti, M. C. e Mattos, N. M. C. (2007) "Teoria de Filas", Editora Interciência, Brasil.

Freitas, P. J. (2008) "Introdução à Modelagem e Simulação de Sistemas com Aplicações em Arena", Editora Visual Books, Brasil.

Souza, M. A., Cortez, O. A. C., Senger, L. J. (2008). Sistema de Monitoração para o Escalonamento de Processos: Estrutura e Métricas de desempenho. Em *RUTI (Revista Unieuro de Tecnologia da Informação)*, v.1, n.1.

Análise comparativa entre ambientes de programação para Multi-Cores e GPGPUs

Deivid Cunha Martins[1], Denise Stringhini[1]

[1]Faculdade de Computação e Informática– Universidade Presbiteriana Mackenzie
São Paulo – SP – Brasil

deividmartins@msn.com, dstring@mackenzie.br

Resumo. *É de conhecimento geral que a computação seqüencial atingiu seu limite de processamento, e que a programação paralela se tornou necessária para obter um bom desempenho. A programação por threads é comum e largamente utilizada. Hoje, tantos os multi-cores quanto as GPUs são boas alternativas para o processamento paralelo de alto desempenho. Este artigo apresenta uma análise comparativa entre ambientes de programação paralela.*

1. Introdução

Este trabalho tem como objetivo realizar uma análise comparativa entre os ambientes de programação paralela atuais. São eles: *Portable Operating System Interface* (POSIX) *Threads* (Pthreads) (BARNEY, 2009), *Open Multi-Processing* (OpenMP) (CHAPMAN, JOST, PAST, 2007) e *Compute Unified Device Architecture* (CUDA) (KIRK, 2008).

2. Descrição do algoritmo usado nos testes

Foi escolhido um algoritmo de ordenação, por se tratar de um problema clássico da computação e após seria feita uma análise comparando-os. O algoritmo escolhido foi o *Bitonic Mergesort*, por este ser o algoritmo mais usado para resolução de problemas de ordenação no paradigma paralelo (WILKINSON, ALLEN, 2005).

3. Metodologia e Resultados

O objetivo principal desse trabalho busca avaliar a facilidade de implementação e o poder computacional dos ambientes previamente descritos. Foi necessário definir uma metodologia para comparação dos mesmos. Para comparar esses três ambientes de programação paralela foram usados os seguintes parâmetros:

- Tempo de execução;
- Estrutura do programa.

Foram utilizados vetores com 1024, 32768 e 524288 elementos. Para 524288 elementos pode-se observar que o tempo de execução em CUDA é 17 vezes menor do que o tempo de execução em Pthreads e OpenMP. Pode-se observar que o tempo de execução em Pthreads e OpenMP são próximos.

Para 32768 elementos se pode notar que os melhores casos para Pthreads e OpenMP são distintos, sendo que o melhor caso em OpenMP teve tempo de execução

quase 2 vezes maior que o tempo de execução em Pthreads. Comparando o melhor caso de CUDA com o melhor de Pthreads, pode-se observar que o tempo de execução em CUDA é 5 vezes menor do que o melhor caso em Pthreads. Para 1024 elementos pode-se notar que os melhores casos para OpenMP e Pthreads são bem próximos. Comparando o melhor caso de CUDA com Pthreads e OpenMP, pode-se observar que o tempo de execução em CUDA é 2 vezes menor do que o melhor caso em OpenMP e Pthreads. A tabela 1 mostra os resultados nos melhores casos de cada ambiente.

Tabela 1. Tempo médio de execução nos melhores casos (em segundos)

Ambiente	Número de elementos		
	1024	32768	524288
CUDA	0.000301	0.001461	0.010769
Pthreads	0.000748	0.007873	0.188560
OpenMP	0.000754	0.015016	0.184431

Pode-se notar que o ganho obtido com o uso de CUDA para 1024 elementos não é tão grande quanto o ganho obtido no caso de 524288 elementos, mas se pode perceber que quanto mais paralelismo é pedido para plataforma CUDA, maiores são os ganhos de desempenho obtidos e que ao contrário do que acontece em OpenMP e Pthreads, em que passando de um limite de *threads*, o desempenho tende a cair, em CUDA pode-se notar que com o aumento do número de *threads* o desempenho sempre aumenta.

4. Conclusão

Pode-se concluir que em um cenário onde é necessário alto desempenho e que a curva de aprendizagem deve ser pequena, OpenMP é a melhor alternativa dos ambientes descritos neste trabalho. Porém em um cenário onde é necessário um controle minucioso das *threads* e a máquina em questão não possui uma GPU, Pthreads se torna a melhor alternativa.

Em um cenário onde é necessário obter alto desempenho e se tem disponível uma GPU, ou se tem um orçamento extra para adquirir um novo dispositivo de *hardware*, no caso uma GPU, CUDA é a melhor alternativa, pois com uma GPU pode-se obter um expressivo ganho de desempenho, com um custo inferior ao da compra de outra máquina. Conforme os experimentos deste trabalho mostram, quanto mais paralelismo é pedido para uma GPU, maiores são os ganhos de desempenho obtidos, o que torna o uso de uma GPU altamente viável.

Referências Bibliográficas

Barney, B. (2009) “POSIX Threads Programming” Lawrence Livermore National Laboratory, Livermore, USA.

Chapman, B., Jost, G. and Pas, R. (2007) “Using OpenMP: Portable shared memory parallel programming” Cambridge, Massachusetts, USA. MIT Press

Kirk, D. B. (2008) “NVIDIA CUDA Software and GPU Parallel Computing Architecture NVIDIA Corporation” NVIDIA Corporation

Wilkinson, B. and Allen, M. (2005) “Parallel Programming – Techniques and Applications using networked workstations and parallel computers” 2nd ed. University of North Carolina at Charlotte, Western Carolina University, Prentice Hall.

Análise e Implementação de uma Solução Distribuída para a Resolução do Problema da Cobertura de Conjuntos Utilizando o Algoritmo Ant System

Mário César M. de Araújo[1], Augusto Mendes Gomes Júnior[1]

[1]Escola de Engenharia e Tecnologia – Universidade Anhembi Morumbi
Caixa Postal 04546-001– São Paulo – SP – Brasil

mario@astronomiabrasil.com, augustomgjr@gmail.com

Resumo. *O problema de cobertura de conjuntos é um problema NP Completo. Com isso, a sua resolução ótima tem complexidade exponencial. Em virtude disso, este trabalho objetiva a implementação de uma solução distribuída, utilizando a heurística Ant System.*

1 Introdução

O problema de cobertura de conjuntos (SCP) é um problema pertencente à classe NP-Completo (CORMEN, 2002), ou seja, o tempo computacional gasto para a resolução do problema aumenta exponencialmente à medida que os dados de entrada aumentam.

No SCP nos é dada uma matriz $m \times n$ $A = [a_{ij}]$ em que todos os elementos da matriz são 0 ou 1. Além disso, a cada coluna é dado um custo não negativo b_j. Dizemos que uma coluna j abrange uma linha i se $a_{ij} = 1$. O objetivo do SCP é escolher um subconjunto das colunas de peso mínimo, que cobre todas as linhas (DORIGO, 2004).

Uma abordagem possível para a resolução de problemas que pertençam à classe NP-Completo é a utilização um algoritmo heurístico associado ao problema. Neste trabalho foi utilizada a meta-heurística *Ant System* (DORIGO, 2004).

A heurística se baseia na capacidade dos grupos de formigas apresentarem um comportamento coletivo muito mais inteligente do que uma formiga sozinha seria capaz de realizar. Inicialmente, um número de formigas percorre as proximidades do formigueiro em busca do alimento. Cada formiga, ao percorrer o seu caminho, deposita uma substância chamada feromônio. Posteriormente as formigas subseqüentes seguem o caminho com maior concentração de feromônio (MULATI, 2009).

2 Modelagem e Implementação Distribuída do SCP

A implementação da solução de *software* proposta neste trabalho é feita utilizando a linguagem de programação C e o padrão MPI (MPI2, 2003) para a comunicação entre processos. O número de formigas e de interações utilizados é igual a 100.

Cada escravo executa uma colônia de formigas, contudo o número total de iterações é dividido entre os escravos. O cálculo do número de iterações de cada colônia é o número total de iterações dividido pelo número de escravos. Essa é a única divisão feita entre os escravos.

Cada colônia trabalha no arquivo de entrada inteiro. Além disso, o primeiro componente adicionado a cada formiga é selecionado de forma randômica. Quanto à atualização do feromônio, ela é feita separadamente em cada escravo. Contudo, logo após este processo, cada escravo comunica aos demais onde e quanto feromônio ele depositou.

Desta forma, apesar do processamento ter sido distribuído, o contexto em que as soluções são criadas continua sendo um só.

Toda a comunicação entre os processos foi realizada com a interface de passagem de mensagem MPI. Através desta interface, o mestre faz a troca de mensagens com os escravos, sendo que toda a comunicação realizada foi com primitivas ponto-a-ponto. Ao final do processamento, os escravos enviam uma mensagem para o mestre com a melhor solução encontrada.

3 Conclusão

O algoritmo distribuído foi implementado e o seu desempenho ficou muito bom. Em pouco tempo de processamento, uma solução muito boa foi alcançada. Atualmente, algumas melhorias e aperfeiçoamento estão sendo realizados para que o algoritmo encontre soluções cada vez mais próximas da solução ótima.

O Gráfico 1 apresenta comparação entre os resultados dos tempos de execução obtidos através dos testes realizados com as versões sequencial e distribuída, com o processamento dos arquivos fonte adotados neste trabalho.

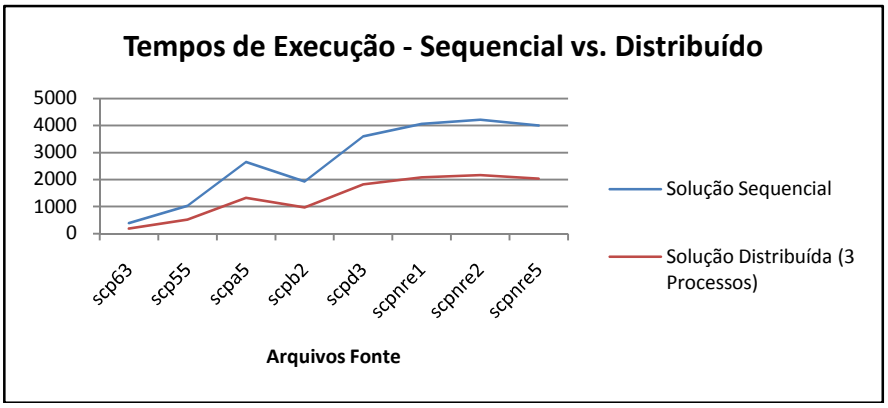


Gráfico 1: Tempos de Execução: Sequencial vs. Distribuído. Fonte: O Autor (2009).

Referências

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. Algoritmos: Teoria e Prática. 2° ed. Rio de Janeiro: Campus, 2002.

DORIGO M.; STÜTZLE T., Ant Colony Optimization. Cambridge: MIT Press, 2004.

MPI2, MPI-2 : The Message-Passing Interface. University of Tennessee, 2003.

MULATI, M. H. Investigação da meta-heurística de otimização por colônia de formigas artificiais aplicada ao problema de cobertura de conjunto. Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Maringá, Maringá. 2009.

SILVA, R. M. A. Otimização baseada em colônia de formigas aplicada ao Problema de cobertura de conjuntos, Tese de Doutorado. Universidade Federal de Pernambuco, 2003.

CARDOSO, A.; GUEDES, J.; SOUZA, P.; FARIA, R.; FERRO, W., Implementação De Uma Solução Paralela Para A Resolução Do Problema De Cobertura De Conjuntos Utilizando Algoritmo Genético. 2008. Monografia (Bacharelado em Ciência da Computação) - Universidade Anhembi Morumbi, São Paulo.

Implementação de um Suporte para Otimização de Acesso a Dados Distribuídos

Vinicius de Freitas Reis[1], **Rodrigo Fernandes de Mello**[1]

[1] Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
São Carlos – SP – Brasil

vreis@grad.icmc.usp.br, mello@icmc.usp.br

Resumo. *Aplicações executando em Data Grids têm seu custo fortemente influenciado pela latência de acesso a dados. A replicação de arquivos é uma técnica efetiva para reduzir tal custo, mas que requer algoritmos de otimização capazes de criar ou remover réplicas automaticamente. Esse artigo descreve a implementação de um suporte para otimização no sistema de arquivos distribuídos GFarm. Esse suporte é modular, permitindo que técnicas de otimização sejam implementadas e testadas por desenvolvedores.*

1. Introdução

Data Grids [Chervenak et al. 1999] são ambientes que provêm uma infraestrutura robusta, capaz de permitir o compartilhamento de grande quantidade de dados armazenados em localidades geograficamente distribuídas. Aplicações executando sobre esses ambientes fazem uso intensivo dos dados armazenados, de forma que seus custos de execução são fortemente influenciados pela latência de acesso aos dados.

A replicação de arquivos, ou seja, a criação de múltiplas cópias de arquivos em localidades distintas da grade, é uma técnica efetiva para reduzir esse custo. A grande quantidade de arquivos e a natureza dinâmica dos *Data Grids* impossibilita a disposição manual de réplicas, fazendo-se necessário um sistema de otimização capaz de detectar padrões de utilização de arquivos e atualizar a distribuição das réplicas de forma a minimizar o tempo de execução das tarefas.

Este trabalho explora soluções para o problema de acesso otimizado a dados em ambientes distribuídos. Para isso, foi implementado, sobre o sistema de arquivos distribuídos **GFarm** [Tatebe et al. 2002], um suporte modular, o qual permite adicionar políticas de otimização de acesso a dados. Esse suporte provê uma interface bem definida, permitindo que demais desenvolvedores projetem e testem novas políticas de otimização. Também foram implementadas algumas políticas de otimização a fim de validar o suporte e avaliar ganhos de desempenho de acesso a dados em ambientes distribuídos reais.

2. Abordagem

Este trabalho consistiu nas etapas de estudo, implementação do suporte e experimentos. Na etapa de estudo, foram testados vários sistemas de arquivos distribuídos para servirem de base ao trabalho, sendo escolhido o sistema **GFarm**, um software open-source, distribuído sob licença **GPL**, parte do projeto Asia Pacific Grid (**ApGrid**). Ele foi projetado para gerenciar o acesso a arquivos em ambientes dinâmicos, compostos por milhares de computadores e armazenando grandes volumes de dados. **GFarm** provê uma imagem

única do sistema de arquivos, e permite que aplicações cliente acessem os arquivos transparentemente por meio de uma interface **FUSE** (*Filesystem in Userspace*).

Na etapa de implementação, optou-se por iniciar otimização a partir da interceptação a chamadas de entrada/saída da aplicação cliente. Leituras e escritas realizadas em arquivos pertencentes ao **GFarm** são registradas, e quando um arquivo é freqüentemente acessado, um processo de otimização é iniciado em um servidor de dados. Esse servidor repassa as informações capturadas sobre os acessos (tempo/freqüência de leituras e escritas, réplicas disponíveis, tamanho dos arquivos etc) à técnica de otimização configurada. Devido à necessidade de modularização, cada técnica de otimização é um processo separado, que recebe, em sua entrada padrão, os dados de acesso aos arquivos e escreve, na saída padrão, as ações a serem tomadas (pode-se remover e/ou replicar arquivos).

Dessa maneira, pode-se facilmente desenvolver e testar políticas de otimização de acesso a dados utilizando qualquer linguagem de programação. Nesse trabalho, já foram implementadas políticas de otimização baseadas nos algoritmos clássicos de substituição de cache – **LRU** (*Least Recently Used*) e **LFU** (*Least Frequently Used*).

Foi montado um ambiente composto por múltiplas máquinas virtuais, utilizando **OpenVZ** [OpenVZ 2010], para realizar os experimentos do projeto. Ele permite que diversos servidores de dados sejam executados na mesma máquina, com *overhead* de virtualização desprezível [Soltesz et al. 2007]. Utilizando *bridges* do kernel Linux, em conjunto com o suporte a controle de tráfego (`tc`), várias interfaces de rede virtuais são interconectadas, permitindo a criação de redes heterogêneas. Pretende-se comparar os resultados experimentais com os obtidos no simulador OptorSim [Bell et al. 2002].

Este artigo foi apoiado por projeto CNPq-Universal, sob processo número 470739/2008-8 e FAPESP, sob processo número 2009/09455-5. Opiniões, descobertas e conclusões ou recomendações não necessariamente refletem as posições do CNPq e FAPESP.

Referências

Bell, W. H., Cameron, D. G., Capozza, L., Millar, A. P., Stockinger, K., and Zini, F. (2002). Simulation of dynamic grid replication strategies in optorsim. In *Journal of High Performance Computing Applications*, pages 46–57. Springer-Verlag.

Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. (1999). The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200.

OpenVZ (2010). http://openvz.org/ – acessado em 25/05/2010.

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. (2007). Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287.

Tatebe, O., Morita, Y., Matsuoka, S., Soda, N., and Sekiguchi, S. (2002). Grid datafarm architecture for petascale data intensive computing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, pages 102–110.

Monitoração de desempenho: uma abordagem para hardware multi-core

Narciso B. Benigno[1], Denise Stringhini[1]

[1]Faculdade de Computação e Informática – Universidade Presbiteriana Mackenzie
São Paulo – SP - Brazil

narciso.benigno@gmail.com, dstring@mackenzie.br

Resumo. *Um monitor de desempenho pode prover informações para programadores de aplicações paralelas, úteis para identificar pontos críticos no funcionamento destas aplicações. Este trabalho analisa o uso de uma biblioteca de instrumentação de software na extração dados de desempenho de um programa paralelo em OpenMP.*

1. Introdução

Este trabalho analisa uma ferramenta para monitorar o desempenho das aplicações para auxiliar aos desenvolvedores a usufruir ao máximo dos recursos de máquinas que dispõem de processadores *multi-core*. Para fazer a análise de desempenho foi utilizada a biblioteca PAPI (*Performance Application Programming Interface*). Esta biblioteca, usada para instrumentação de código, acessa os contadores de hardware que são responsáveis pela análise de desempenho.

A instrumentação consiste em analisar o software para detectar regiões de execução critica, esta possibilidade permite ao desenvolvedor identificar problemas no desempenho e aplicar melhorias em uma região identificada. Para fazer esta instrumentação foi utilizada duas versões de um programa de multiplicação de matrizes: uma sequencial e outra paralela. Na versão paralela, cada thread multiplica um pedaço da matriz, este pedaço foi chamado de *chunk*. O programa utiliza duas threads.

As informações sobre o hardware utilizado também podem ser obtidas pela biblioteca PAPI, através da função ***PAPI_get_hardware_info***. Para este teste, a seguinte configuração foi utilizada: um processador com 2 núcleos, modelo Intel Dual Core, velocidade de cada unidade de processamento: de 800MHz.

Para comparar o desempenho entre as implementações, foi utilizado o tempo de processamento, este é o tempo **efetivo** de utilização do recurso processador, informação que foi extraída pela função ***PAPI_flips*** da biblioteca. Esta função é utilizada para monitorar algumas informações, dentre elas está o tempo de utilização do processador.

Tabela 1: Tempos de execução extraídos da biblioteca PAPI.

Dimensão da Matriz	Tempo do Sequencial (s)	Tempo do Paralelo (s)
100	0,021396	0,033592
200	0,167572	0,131555
300	0,661631	0,633542
400	1,719048	1,260361
500	4,547075	3,340425

O teste varia a multiplicação das matrizes quadradas com um fator de 100 elementos de cada vez em cada dimensão. Na tabela 1, pode-se verificar que para uma matriz quadrada de dimensão 500 a diferença dos tempos de execução entre o programa

sequencial e o paralelo é considerável a favor da versão paralela, porém com a redução da dimensão da matriz a tendência se inverte.

Também foi testado o uso da diretiva *schedule* do *for* paralelo com os parâmetros *static* e *chunk*. Estes parâmetros informam que as iterações que ocorrerem dentro deste bloco terão no máximo tamanho *chunk*, até completar o tamanho total.

Na tabela 2 é possível perceber que com a redução de *chunk* o tempo total do uso do processador é reduzido significativamente, até um determinado limite onde esta tendência se inverte, quando há uma queda de desempenho na execução do algoritmo como pode se pode observar quando o *chunk* chega a 25.

Tabela 2: Tempo de execução com a redução do *chunk*.

Dimensão da Matriz	
Chunk	**Tempo (Em Segundos)**
400	2,526403
300	1,891673
200	1,253635
100	1,232373
50	1,245388
25	1,262814

4. Conclusão e Trabalhos Futuros

A biblioteca PAPI propiciou facilidade de acesso aos contadores de hardware, esta facilidade traz benefícios como a não necessidade de um conhecimento maior do hardware do processador. Além disto, possibilitou medições mais precisas, independentes de possíveis atrasos que possam ser gerados pelo sistema operacional. Um exemplo foi a contagem do tempo de processamento, pois é possível obter uma informação muito próxima da real extraída dos contadores do processador, o que dá exatamente o tempo de uso deste recurso. Uma maior exploração da variedade de contadores existentes, principalmente os que devolvem dados de uso da *cache*, é sugerido como trabalho futuro.

Bibliografia

Drongowski, Paul J. Instruction-Based Sampling: A New Performance Analysis Technique for AMD Family 10h Processors. Disponível em: <http://developer.amd.com/Assets/AMD_IBS_paper_EN.pdf>.

Ferreira, João Canas. Multiprocessadores: uma introdução breve. 2005. FEUP

Furber, Steve. ARM System-on-Chip Architecture. 2. ed. Addison-Wesley. 2000.

Site oficial PAPI. Disponível em: <http://icl.cs.utk.edu/projects/papi/wiki/Main_Page>. Acessado em 16/07/2009.

Wilkinson, Barry; Allen, Michael. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. 2. ed. 2005 Pearson Prentice Hall.

Zeichick, Alan. Surviving And Thriving In A Multi-Core Word An Ebook From AMD Developer Central. Disponível em: <http://developer.amd.com/Assets/ThrivingandSurvivinginaMulti-CoreWord.pdf>. 2006.

Um Sistema de Avaliação Paralelo de Algoritmos Genéticos aplicados à Otimização da Arquitetura de Redes Neurais Artificiais do tipo *Backpropagation*

Lourenço Alves P. Júnior[1], **Leandro Arakaki**[2], **Marcos Alberto de Carvalho**[2]

[1] Universidade de São Paulo – USP
Instituto de Ciências Matemáticas e de Computação – ICMC
Av. Trabalhador São-carlense, 400, 13560-000 – São Carlos – SP – Brazil

ljr@icmc.usp.br

[2]Universidade José do Rosário Vellano – UNIFENAS
Faculdade de Ciências de Computação – FCC
Alfenas – MG – Brazil

leandro@familiaarakaki.com.br, marcos.carvalho@unifenas.br

Resumo. *Este trabalho apresenta uma solução para a otimização da Arquitetura de Redes Neurais Artificiais do tipo Backpropagation, usando Algoritmos Genéticos para mapear uma Arquitetura. Devido à computação demandada pelo conjunto de Arquiteturas da população, cada indivíduo é instanciado em um nodo de um Cluster para treinamento. Resultados mostram um ganho de execução em torno de* 84% *quando são comparadas uma execução sequencial e outra paralela.*

Na implementação de uma Rede Neural Artificial (RNA) do tipo *Backpropagation*, podem ser identificadas pelo menos cinco fases, a saber: *concepção, modelagem, treinamento, aplicação e manutenção*. Durante a fase *concepção* é feito um estudo para analisar se o problema em questão é linear ou não. Caso a natureza seja não linear, inicia-se a *modelagem*, (1) propondo uma Arquitetura da RNA a ser modelada e (2) selecionando um conjunto de treinamento que possa representar os dados de entrada do sistema . Uma vez que a Arquitetura tenha sido definida, o *treinamento* com o conjunto de dados selecionado é iniciado para que seja feita uma avaliação sobre a adequação da Arquitetura escolhida. A decisão de aproveitar aquela escolha ou tentar uma outra se dá pelos resultados dos treinamentos preliminares, com participação direta do projetista, o qual observa e analisa os resultados obtidos. Uma vez que a fase de *treinamento* se encerre, ou seja, um Arquitetura aparentemente satisfatória tenha sido conseguida, a RNA obtida é transformada em um componente e integrada à *aplicação*. Caso o desempenho seja insatisfatório nas execuções do mundo real, deve ser feita uma *manutenção* a fim de melhorar a Arquitetura proposta ou realizar o treinamento com novos dados.

A escolha da melhor Arquitetura muitas vezes é feita na "tentativa e erro", baseada nas experiências do projetista [Whitley et al. 1990]. Uma RNA propõe-se a fazer um modelo matemático para o neurônio biológico, e, nesse modelo, muitos cálculos são feitos. Portanto, dependendo de como os neurônios tenham sido organizados na Arquitetura, a quantidade de processamento pode diminuir ou aumentar. Assim, uma solução que possa evoluir para uma Arquitetura ótima e também realizar essa tarefa de forma rápida pode se tornar uma ferramenta útil para os projetistas de tais sistemas.

A lacuna pesquisada neste trabalho foi a de propor uma solução que proporcionasse a automação da escolha de uma Arquitetura ótima para RNA projetada, transferindo a responsabilidade de sua escolha e treinamento ao sistema modelado. Para se chegar a uma Arquitetura ótima foram utilizados Algoritmos Genéticos (GAs), mapeando-se aquela proposta a um indivíduo da população e efetuando-se o treinamento em execuções paralelas através de um ambiente de Passagem de Mensagem baseado no padrão MPI.

Os Algoritmos Genéticos tem como objetivo encontrar soluções ótimas para problemas modelados a eles. Assim, com a associação de uma Arquitetura a um indivíduo da população a ser evoluida pelos GAs, a tendência é que haja um conversão para um população de Arquiteturas ótimas para aquela RNA modelada. No entanto, para que seja feita uma avaliação (*fitness*) dos indivíduos, deve haver o treinamento de cada um deles. Como o treinamento de cada Arquitetura exige bastante processamento, uma população de Arquiteturas demandará uma potência computacional maior ainda. Dessa forma, implementou-se os AGs em um *Cluster* modelando os processos de forma mestre/escravo com comunicação do tipo `scatter` e `gather` [Kumar 2002] . Portanto, na função de avaliação dos AGs, um processo para cada indivíduo da população é iniciado em um nodo do *Cluster*, onde é feito o treinamento com o conjunto de dados escolhidos. Ao final do processamento, os resultados são enviados de volta à função que instanciou aquele processo. Uma vez com o resultado de todos os indivíduos daquela população de Arquiteturas, os AGs evoluem, de forma sequencial, uma nova população. O processo termina quando existe uma estabilidade nas convergências obtidas ou quando um certo número de evoluções é atingido.

Resultados experimentais revelaram um *speedup* de 5.07 e eficiência de 84.49%, obtidos de experimentos executados em um *Cluster* com 6 nodos escravos. Para o processamento de uma nova população, há um ponto de sincronismo entre as tarefas, o que não torna a eficiência obtida maior. Um dos problemas é que, dependendo da Arquitetura, o tempo de execução pode ser maior ou menor, portanto aquelas que executam em um tempo menor liberam seus nodos, deixando-os ociosos até que uma nova população seja gerada, e novos treinamentos sejam solicitados a eles. Esses resultados consideram somente o custo computacional envolvido, no entanto, ainda existiria o esforço do projetista que deveria propor uma Arquitetura e avaliá-la, o qual torna-se transparente pelo sistema proposto.

Por fim, concluir-se que este trabalho contribui com o protótipo de uma ferramenta que apoia o desenvolvimento de RNAs e utiliza os recursos computacionais a uma taxa de eficiência em torno de 84.49%, proporcionando também a automação de duas fases do processo de implementação de RNAs. Como trabalhos futuros, espera-se: (1) explorar novas abordagens para diminuir o tempo ocioso existente entre o ponto de sincronismo; (2) implementar soluções que contemplem Computação em Grade e em Nuvens;

Referências

Kumar, V. (2002). *Introduction to Parallel Computing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Whitley, D., Starkweather, T., and Bogart, C. (1990). Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14(3):347–361.