Boletim Técnico da Escola Politécnica da USP
Departamento de Engenharia de Computação e
Sistemas Digitais

ISSN 1413-215X
BT/PCS/0220

BGLsim: Simulador de Sistema Completo para o Blue Gene/L

Luís Henrique de Barros Ceze
Wilson Vicente Ruggiero

São Paulo - 2002

O presente trabalho é um resumo da dissertação de mestrado apresentada por Luís Henrique de Barros Ceze, sob orientação do Prof. Dr. Wilson V. Ruggiero: "BGLsim: Simulador de Sistema Completo para o Blue Gene/L", defendida em 09/08/2002, na EPUSP.

A íntegra da dissertação encontra-se à disposição com o autor e na biblioteca de Engenharia de Eletricidade da Escola Politécnica da USP.

FICHA CATALOGRÁFICA

Ceze, Luís Henrique de Barros
BGLsim : simulador de sistema completo para o Blue Gene/L / L.H.B. Ceze, W.V. Ruggiero. – São Paulo : EPUSP, 2002.
12 p. – (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia de Computação e Sistemas Digitais, BT/PCS/0220)

1. Simulação de sistemas 2. Arquitetura e organização de computadores 3. ATM (Redes de computadores) I. Ruggiero, Wilson Vicente II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais III. Título IV. Série
ISSN 1413-215X CDD 003.3
004.22
004.6

BGLsim: Simulador de Sistema Completo para o Blue Gene/L

Luís Henrique de Barros Ceze
lceze@larc.usp.br

Wilson V. Ruggiero
wilson@larc.usp.br

julho de 2002

Resumo

Este artigo apresenta o ***BGLsim***, um simulador paralelo de sistema completo para o Blue Gene/L[1]. O ***BGLsim*** provê um correspondente no ambiente de simulação de praticamente todos os componentes da máquina real, sendo uma ferramenta bastante útil para o desenvolvimento e validação de software. Sua alta performance[2] possibilita a execução de imagens completas e sem modificações de sistemas operacionais e aplicações. O ***BGLsim*** é um simulador paralelo, onde cada uma das suas instâncias simula um nó da máquina real. Todas essas instâncias comunicam-se entre si através das interfaces de rede simuladas e juntas formam uma máquina Blue Gene/L virtual. O ambiente de simulação provido pelo ***BGLsim*** vai além dos nós e das redes de interconexão, pois também simula os componentes de controle que farão parte da máquina real. Este recurso possibilita a validação de sistemas de *bring-up* e controle, que são externos à máquina.

Abstract

This paper presents ***BGLsim***, an implementation of a full system simulator for the massively parallel machine Blue Gene/L[3]. ***BGLsim*** provides a simulation counterpart for almost every component in the real machine, being a very useful tool for software development and validation. Its high performance[4] allows the execution of full and unmodified operating systems and applications images. ***BGLsim*** is parallel simulator, each of its instances simulates a node of the real machine. All instances communicate between themselves through the virtual network interfaces and together they form a virtual Blue Gene/L machine. The simulation environment provided by ***BGLsim*** goes beyond the nodes and the interconnection networks, it also includes the control components that will be present in the real machine. This feature allows the validation of bring-up and control systems software that are external to the machine.

1 Introdução

A crescente complexidade dos sistemas de computação torna seu entendimento e projeto cada vez mais desafiador e trabalhoso. Na busca por melhor desempenho e sistemas mais robustos, a interação entre os diversos componentes se torna algo difícil de prever; tal caso acontece em sistemas de arquitetura paralela. O uso de simuladores no projeto de sistemas possibilita uma abordagem sistemática para o entendimento da interação entre *software* e *hardware*, como mencionado em [7].

A demanda de recursos para a execução de simulações tem uma relação direta com a complexidade do sistema estudado e com o nível de detalhe utilizado. Para uma simulação ser útil, é necessário que o tempo de execução seja prático e o nível de detalhe seja suficiente para se estudar o problema desejado. Uma forma de se reduzir o tempo de simulação é fazer com que o simulador seja paralelizado. Um estudo da relação custo/benefício de simulações paralelas de sistemas paralelos está presente em [6].

[1] Esta máquina é parte da família Blue Gene, atualmente em desenvolvimento pela IBM Reasearch

[2] A performance de simulação atingida pelo ***BGLsim*** é da ordem de 2 milhões de instruções por segundo

[3] This machine is part of the Blue Gene family, currently being developed by IBM Research

[4] The simulation performance achieved by ***BGLsim*** is 2 million instructions/second on Pentium III 1GHz host, characterizing a 500 times slowdown

A principal característica de um simulador de sistema completo é expor ao *software* sendo executado um ambiente muito a próximo ao *hardware*, sem entrar nos detalhes de circuitos lógicos, como os simuladores que usam descrição em VHDL fazem.

Simuladores de sistema completo, em geral, modelam o *hardware* no nível de arquitetura com detalhe suficiente para ser possível executar imagens não modificadas de programas como sistemas operacionais, *device drivers* e aplicações. Outra característica importante é o alto desempenho; em geral, simuladores desse tipo são de 3 a 6 ordens de magnitude mais rápidos que simuladores VHDL baseados em *software*.

Para a simulação de um sistema de computação ser completa, é necessário que os dispositivos no sistema também sejam simulados, incluindo a emulação de sua funcionalidade. Por exemplo, um dispositivo de rede deve ser capaz de se comunicar com a rede.

1.1 Motivação

A IBM está atualmente desenvolvendo uma família de computadores maciçamente paralelos chamada Blue Gene. A arquitetura desses computadores se baseia em unidades simples, geralmente compostas de processador, memória e dispositivos de interconexão. Essas unidades são chamadas células. Cada célula, através de seus dispositivos de interconexão, se conecta com outras formando uma rede de células. A partir da ligação de um grande número dessas células se obtém uma máquina paralela de alto desempenho e grande escalabilidade, pois o número de células pode variar de acordo com a configuração necessária.

O primeiro computador dessa família a ser implementado é o Blue Gene/L. As células desse computador serão baseadas num variante do microprocessador PowerPC para aplicações embutidas. A memória de cada célula é privada, só podendo ser acessada pelos processadores da célula. O desenvolvimento dessa máquina requer como ferramenta uma série de simuladores: simuladores que utilizam descrições em VHDL para o projeto de *hardware*, simuladores de rede para o dimensionamento das interfaces de interconexão e simuladores que propiciem um ambiente para o desenvolvimento de *software*.

A arquitetura inovadora e altamente paralela dessa máquina torna o desenvolvimento de *software* uma tarefa desafiadora e custosa. Por isso, atrelar o desenvolvimento de *software* de sistema[5] e aplicações ao desenvolvimento do *hardware* pode atrasar o processo de entrega da máquina, pois o *software* começaria a ser desenvolvido após o término do *hardware*. Assim, de forma a evitar esse atraso, é necessário que o desenvolvimento de *software* seja feito concorrentemente com o desenvolvimento do *hardware*. Para isso é necessário o uso de um simulador que possibilite a execução e validação de código.

Apesar de uma simulação seqüencial de sistemas paralelos ser possível, o tempo de execução se tornaria impraticável conforme o número de células ou nós fosse crescendo. Uma solução para esse problema é desenvolver um simulador paralelo, onde cada instância do simulador simula uma célula ou nó da máquina real. Através da execução de várias instâncias desse simulador (que podem ser executadas em diferentes máquinas) e da comunicação entre essas instâncias, a máquina final seria simulada. Esse é o principal assunto deste artigo.

1.2 Trabalhos Relacionados

A área de simulação de sistema completo sempre teve atenção tanto do meio acadêmico quanto da indústria. Isso se deve principalmente à necessidade de ferramentas que possibilitem o estudo de sistemas cada vez mais complexos, aliada ao aumento do desempenho e à queda do custo de sistemas de alto desempenho para serem *hosts* das simulações.

O artigo [14], publicado na revista Computer de Fevereiro de 2002, discute simuladores de alto desempenho para sistemas de computação. Também apresenta os principais simuladores de uso geral

[5] compiladores, sistemas operacionais, bibliotecas de tempo de execução, ambiente de depuração, etc.

disponíveis no momento. Não foi dado, porém, um enfoque muito grande a simuladores de sistema completo.

Em termos de simuladores disponíveis no momento, atualmente existem dois trabalhos que se relacionam diretamente com o ***BGLsim***. Um deles é o SimOS, cuja relação com o ***BGLsim*** é discutida em 1.2.1. O outro é o Simics e sua relação com o ***BGLsim*** é apresentada em 1.2.2.

Outro trabalho importante relacionado à simulação de sistemas de computadores é o simulador SimpleScalar [3], que é uma ferramenta bastante utilizada para simulação de processadores no nível de micro-arquitetura. Esse trabalho, apesar de não ser um simulador de sistema completo, apresenta contribuições úteis para o ***BGLsim***, como uma discussão sobre implementação de rotinas semânticas das instruções de ponto flutuante nativamente ou não-nativamente. Outra discussão interessante é sobre a situação em que se tem o sistema simulado e o *host* em *endianess*[6] diferentes, o que acarreta uma série de problemas no desenvolvimento de rotinas de simulação do acesso à memória.

Uma outra abordagem de simulação de sistemas de computadores é a instrumentação do código do programa que se pretende ensaiar e sua execução nativa. O artigo [15] apresenta o Augmint, um ambiente de simulação de sistemas multiprocessados baseados na arquitetura Intel x86. Esse ambiente é baseado em instrumentação de código nativo e atinge alto desempenho.

1.2.1 SimOS

O trabalho acadêmico mais diretamente relacionado com a dissertação proposta é [7], em que é proposto o uso de simuladores de sistema completo para o entendimento do comportamento de sistemas de computação; em particular, é apresentado um sistema chamado SimOS, desenvolvido pela Universidade Stanford. Também são ilustrados alguns modelos de máquinas comuns no mercado na época de sua publicação.

Por ser um sistema consagrado na comunidade acadêmica, e ser considerado o pioneiro em simulação de sistemas completos, foi um grande catalisador para o início do trabalho aqui proposto, ajudando na especificação inicial. Porém, em termos de sistemas paralelos, o SimOS não apresenta muitas funcionalidades, especialmente em sistemas celulares [16]. O ***BGLsim*** se relaciona com o SimOS em sua funcionalidade quando se olha para apenas um nó do sistema. A principal característica que difere o ***BGLsim*** do SimOS é seu paralelismo e sua abrangente completude, chegando ao nível do sistema de controle da máquina.

1.2.2 Simics

O Simics é um simulador de máquina completa e é o principal produto de uma empresa chamada Virtutech[18]. Esse produto tem alto desempenho e diversas funcionalidades interessantes, como a capacidade de simular uma rede Ethernet e seus respectivos nós, sincronizando as diversas máquinas de forma a não criar uma diferença de desempenho irreal. Assim como o ***BGLsim***, o Simics realiza a simulação de diversos nós de uma rede em *hosts* diferentes, diminuindo a carga de simulação por *host*. Uma diferença entre os dois, porém, está na escalabilidade, pois o ***BGLsim*** foi projetado para escalar na ordem de centenas de nós e o Simics suporta apenas algumas dezenas. Outra semelhança entre os dois é a existência de uma aplicação central que inicia e gerencia as diversas simulações em paralelo sendo executadas. Em seu principal artigo [5], que faz parte de uma série de artigos sobre simulação de alta performance, são apresentadas uma série de justificativas para o uso de simulação de sistema completo, enfatizando o ganho de tempo na paralelização do desenvolvimento de *hardware* e *software*. É importante mencionar que existe também uma semelhança muito grande entre os Simics e o SimOS [7], apesar do primeiro ser um produto comercial e o segundo ser um projeto acadêmico. Um ponto fraco é a difícil extensibilidade desse simulador e o fato de o código fonte não ser publicamente disponível.

[6] ordem em que os bytes de uma palavra são armazenados na memória

2 Arquitetura do Blue Gene/L

2.1 Projeto Blue Gene

O projeto Blue Gene tem como objetivo final produzir uma máquina de arquitetura celular [16] de desempenho da ordem de 1 petaflops. A primeira aplicação prevista para essa máquina é o dobramento de proteínas, uma aplicação computacionalmente intensa cuja escalabilidade foi demonstrada em [1].

Durante seu desenvolvimento, foi decidido que "Blue Gene" seria o nome de uma família de computadores de arquitetura celular desenvolvidos pela IBM Research. A primeira máquina a ser implementada é o Blue Gene/L, primeiro passo para se chegar na máquina de 1 petaflops. O Blue Gene/L também é baseado em conceitos de arquitetura celular, usando processadores da linha PowerPC e terá um desempenho da ordem de 200 teraflops.

A unidade básica que compõe o Blue Gene/L, uma célula dentro do conceito de arquitetura celular, será chamada de nó daqui para frente. O Blue Gene/L possui dois tipos de nós: nós de I/O e nós de computação; a diferença básica entre eles é a quantidade de memória e a quais redes de interconexão estarão conectados. Existem três tipos de rede de interconexão sob o controle da aplicação:

TORUS - uma rede de interconexão toroidal que engloba todos os nós de computação [17].

TREE - uma rede de interconexão em árvore que engloba todos os nós, incluindo nós de I/O e de computação [8].

Gigabit Ethernet - todos os nós possuem uma interface de rede desse tipo, porém apenas os nós de I/O estarão conectados a uma rede Ethernet [9].

O Blue Gene/L também disporá de uma quarta rede de interconexão fora do controle da aplicação. Essa interface chama-se JTAG[7] [11] e servirá para o controle e obtenção de informações de baixo nível do ASIC[8] de um nó, atuando sobre praticamente todos os componentes, excluindo a memória principal.

O Blue Gene/L pode ser visto como um conjunto de blocos chamados *psets*, que são compostos por um nó de I/O e 64 nós de computação. Apenas os nós de I/O são visíveis para o usuário; os nós de computação podem ser vistos como "aceleradores" computacionais dos nós de I/O. Dessa forma, a máquina expõe para o usuário um *cluster* de *psets*, proporcionando um ambiente muito próximo ao de *clusters* atualmente em uso.

Para acomodar diversas aplicações e usuários simultaneamente de uma forma flexível, o Blue Gene/L irá suportar particionamento. Através dessa funcionalidade será possível dividir a máquina em uma série máquinas virtuais menores, proporcionando gerenciamento e suporte a aplicações mais efetivos. Esse particionamento acontece tanto logicamente do ponto de vista do *software* quanto do ponto de vista elétrico, através de chaveadores eletrônicos que particionam os fios de interconexão.

2.2 Nó

De uma forma geral, um nó do Blue Gene/L pode ser visto como um SMP[9] de dois processadores, porém sem coerência de *cache*. Isso significa que se um processador altera um dado na memória principal que está presente no *cache* do outro processador, o *cache* do outro processador não vai ter seu conteúdo correspondente invalidado e vai ficar inconsistente com a memória. Assim, a falta de coerência de *cache* deve ser cuidadosamente gerenciada pelo *software*.

Um nó será composto de:

[7] padrão IEEE 1149.1 para portas de teste em circuitos integrados

[8] *Applicaion Specific Integrated Circuit - chip* especialmente projetado para uma aplicação

[9] *Symmetric Multi-Processor*

- dois processadores PowerPC 440
- subsistema de memória com 3 níveis de *cache* (L1/L2/L3)
- 256 MB de memória em nós de computação e 512 MB em nós de I/O
- interfaces de interconexão (*TORUS*, *TREE*, *Gigabit Ethernet* e JTAG)

Todos os componentes funcionais de um nó, exceto a memória, estarão fisicamente localizados dentro de um único *chip*, caracterizando o que é chamado de *system-on-a-chip*[4].

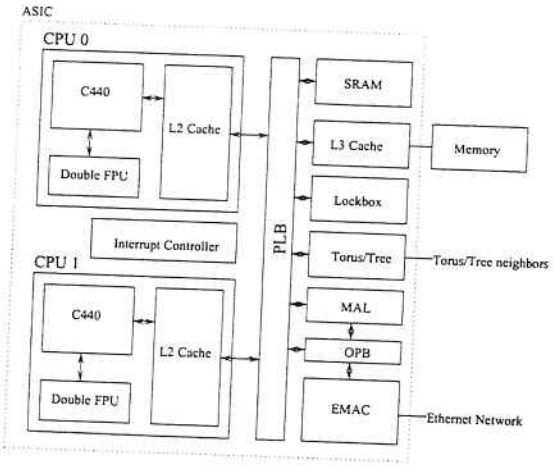

Figura 1: Diagrama simplificado dos componentes de um nó

A figura 1 ilustra, de maneira simplificada, como os componentes de um nó se relacionam. O quadro pontilhado com o título "ASIC" mostra quais componentes se encontram dentro do *chip* principal de um nó. Os componentes e as siglas serão expostos nas seções seguintes, onde serão fornecidos detalhes sobre os componentes de um nó.

CPU O microprocessador que será usado como CPU de um nó é um PowerPC 440 [10]. O PowerPC 440 é um processador RISC[10] de 32 *bits* projetado principalmente para aplicações embutidas. Suas principais características são: 32 registradores de propósito geral; *pipeline* de 7 estágios e 2 unidades de despacho; previsão dinâmica de *branches*; capaz de fazer uma multiplicação de inteiros de 32 *bits* em um ciclo.

FPU dupla Cada uma das duas CPUs de um nó disporá de uma unidade de ponto flutuante dupla. Essa unidade conecta-se ao processador através de uma porta chamada APU[11] e estende o conjunto de instruções do processador. Essas instruções são do tipo SIMOMD[12] e são capazes de realizar simultaneamente operações diferentes em dois elementos distintos de um vetor.

2.3 Controle

Devido ao grande número de nós presentes na máquina é necessário um sistema de controle elaborado. Esse sistema de controle é responsável por basicamente duas tarefas: IPL[13] e monitoramento não intrusivo.

[10] *reduced intruction set computer*
[11] *attached processing unit*
[12] *single instruction multiple operations multiple data*
[13] *initial program loading*

Os nós da máquina não possuirão disco, portanto todo I/O vai pela rede, por isso é preciso que seja carregado um programa inicial que ofereça funcionalidade suficiente para o resto do processo de *boot* da máquina.

De forma a manter o bom funcionamento da máquina, será necessário adquirir uma série de informações de baixo nível tanto sobre o próprio *chip* que constitui o nó quanto sobre os ventiladores, fontes de energia, temperatura do equipamento, entre outras. Essas informações também serão extraídas e gerenciadas pelo sistema de controle.

ETH chips *ETH chips* são dispositivos que fazem a interface entre uma rede IP sobre Ethernet de controle com a rede de serviços JTAG. Os *ETHchips* implementam em *hardware* uma pilha IP simplificada para comunicação com o mundo. As requisições vindas através da rede IP são traduzidas para comandos JTAG e enviadas para o nó ou dispositivo desejado. A resposta da requisição gerada pelo nó ou dispositivo é recebida e encapsulada pelo *ETHchip* e enviada para a rede IP.

3 *BGLsim*

O ***BGLsim*** é um simulador de sistema completo para o Blue Gene/L. A simulação implementada é dirigida por execução, com cada instrução do programa sendo executado pelo simulador sendo simulada. O nível de profundidade da simulação é o de arquitetura, em que o estado de alto nível (visível pelo *software*) do *hardware* é modelado.

O ambiente simulado provido pelo ***BGLsim*** inclui nós, redes de interconexão e *ETHchips*, além de diversos mecanismos de *escape* do ambiente simulado para o ambiente do *host*.

O principal elemento funcional do ***BGLsim*** é o simulador de um nó. O simulador de um nó possui tudo que um nó oferece dentro do seu *chip* (lembrando que um nó é um *system-on-a-chip*), além da memória principal.

Várias instâncias desse simulador de um nó podem ser executadas simultaneamente em um *cluster* de *hosts*. Essas diversas instâncias, desde que participando da mesma simulação, podem se comunicar através dos dispositivos de interconexão. Esses dispositivos de interconexão oferecem conectividade real entre os nós.

Outro elemento funcional do ***BGLsim*** é o simulador de *ETHchips*, que, como o componente real da máquina, disponibiliza serviços de controle e monitoramento de baixo nível dos nós da máquina.

Além dos simuladores de nós e de *ETHchips*, existem dois elementos funcionais que fazem parte do ***BGLsim*** que não possuem correspondentes na máquina real. São eles o *TAPserver* e o *EthernetGateway*, que, em conjunto, oferecem conectividade entre a rede Ethernet virtual do ***BGLsim*** e a rede Ethernet externa.

Apesar de um simulador VHDL do *chip* ser muito mais fiel que o ***BGLsim***, seu desempenho não possibilita a execução de longos trechos de código em um tempo prático. O ***BGLsim*** é 6 ordens de grandeza mais rápido que o simulador VHDL atualmente usado pela equipe de projeto do Blue Gene/L. Além disso o simulador VHDL do *chip* possibilita a execução de no máximo dois nós conectados entre si, enquanto testes com o ***BGLsim*** já chegaram a 125 nós.

A tabela 1 fornece uma idéia quantitativa da diferença de desempenho entre o ***BGLsim***[14] e o simulador VHDL. Para o primeiro item, "*boot* do Linux", a referência do fim do processo de *boot* é o aparecimento do *prompt*. O segundo item, "DGEMM", é um código de multiplicação de matrizes, que nesse experimento foi feita a multiplicação de duas matrizes 20x20. O terceiro item é um código de cálculo da raiz quadrada dos elementos de um vetor, nesse caso o vetor tinha 800 elementos. Para o segundo e terceiro ítens, apenas o trecho de código responsável pelos cálculos foi levados em conta.

[14] para comparação, o *host* utilizado para os experimentos foi um Pentium III de 1 GHz

workload	no. de instruções	***BGLsim***	VHDL
boot do Linux	165345715	119s	impraticável
DGEMM 20x20	39093	1s	1930s
SQRT 800	766	1s	40s

Tabela 1: Contagem de instruções executadas e tempo de simulação usando o ***BGLsim*** e o simulador VHDL.

3.1 Simulação de um nó

A simulação de um nó consiste em oferecer ao *workload* a impressão de estar sendo executado em um nó real do Blue Gene/L. Essa simulação envolve disponibilizar a própria CPU, memória e dispositivos.

O ***BGLsim*** possui uma estrutura de dados principal que descreve um nó. Essa estrutura de dados dispõe de ponteiros para tudo o que um nó possui, como memória, processadores e dispositivo.

A figura 2 ilustra em alto nível o modelo de um nó. Os *buses* PLB e OPB são modelados simplesmente como um *BUS*, que na verdade é apenas um mapeamento de endereços físicos e dispositivos.

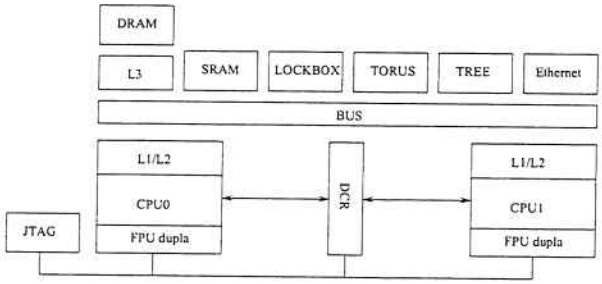

Figura 2: Diagrama em alto nível do modelo de um nó

3.1.1 CPU

Os nós do Blue Gene/L usarão como CPU o processador PowerPC 440. No ***BGLsim***, o modelo do processador é representado por uma estrutura de dados que contém todas as informações referentes ao estado do processador. Estas informações são:

- *program counter*
- 32 registradores de uso geral de 32 *bits* (GPRs)
- registradores de propósito especial (SPRs)
- registradores de *status*, controle e dados da FPU dupla
- TLB e *shadow* TLBs
- ponteiro para interrupções pendentes
- estruturas de dados dos *caches* L1 e L2

As estruturas de dados de componentes compartilhados pelos dois processadores de um nó ficam localizadas na estrutura de dados principal da máquina.

A figura 3 mostra um ciclo de simulação da CPU segue uma abordagem direta de *fetch-decode-execute*. No processo de simulação de um ciclo da CPU, a seguintes etapas são realizadas:

interrupts check verifica se há alguma interrupção pendente, caso haja, o contador de instruções é desviado

fetch busca a instrução atual no subsistema de memória

decode decodifica a instrução, determinando os parâmetros de execução e a rotina semântica correspondente no simulador

execute executa a rotina semântica determinada no passo anterior, que realiza as operações pertinentes à instrução

devices poll verifica se há algum evento a ser tratado nos dispositivos

timer update busca a instrução atual no subsistema de memória

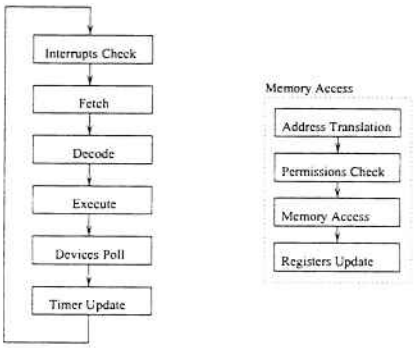

Figura 3: Ciclo de simulação da CPU

Os acessos à memória envolvidos no *fetch* da instrução e na execução de instruções de *load/store* são tratados pelo subsistema de memória. O processador PowerPC 440 não suporta modo real de endereçamento, portanto, todos os acessos à memória, incluindo acesso a dispositivos mapeados em memória, passam pelo processo de tradução de endereço.

Quando se está usando a opção de duas CPUs em um mesmo nó, a simulação é realizada executando um ciclo de CPU cada alternadamente. Assim, como a simulação de ambas as CPUs é feita em um mesmo processo e em uma única *thread* de execução, não há problema de *race condition*, pois tudo é serializado, e, portanto, as estruturas de dados estão sempre consistentes.

A unidade de ponto flutuante (ver **??**) dupla é modelada como dois conjuntos de registradores de ponto flutuante de precisão dupla e um conjunto de rotinas semânticas que emulam as instruções suportadas por essa unidade. As estruturas de dados que representam esses registradores residem na estrutura de dados do processador apresentada em 3.1.1. Na implementação desse modelo foi decidido que as operações de ponto flutuante propriamente realizadas durante a simulação seriam realizadas nativamente no *host*. Isso acelera significativamente a simulação, pois não é necessário simular as operações de ponto flutuante usando operações de ponto fixo. Essa decisão, porém, acarreta problemas principalmente na operação de multiplicação-e-soma, pois na unidade de ponto flutuante do Blue Gene/L essa operação é feita usando apenas uma instrução e a parte da soma é feita com a precisão interna de 80 *bits*, enquanto que o Pentium (*host* do ***BGLsim***) não possui instrução de multiplicação-e-soma e, portanto, essa instrução é simulada usando uma multiplicação e uma soma, acarretando um arredondamento intermediário que causa uma discrepância em relação valor previsto. Esse problema não foi resolvido, uma vez que essa discrepância não era significativa ao ponto de se justificar a implementação das rotinas semânticas das instruções de ponto flutuante usando operações de ponto fixo.

3.2 Simulação de múltiplos nós

A simulação de múltiplos nós implementada pelo ***BGLsim*** é basicamente um conjunto de instâncias do simulador de um nó rodando simultaneamente e comunicando entre si em um *cluster* ou rede de *hosts*. Mas não são apenas simuladores de nós que compõem uma simulação de múltiplos nós. Também fazem parte do ambiente os simuladores de *ETHchips*, os *gateways* Ethernet e os *TAPservers*. A figura 4 mostra como os componentes de uma simulação de múltiplos nós se relacionam. Note que estão presentes duas simulações no diagrama.

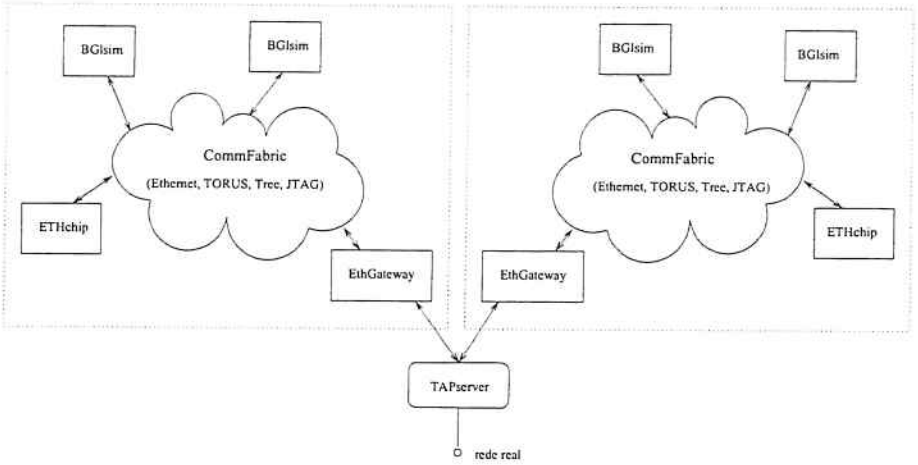

Figura 4: Visão geral de simulações de múltiplos nós

Cada um dos elementos mencionados acima é um processo no simulador. Esses processos se comunicam através de uma abstração de nfra-estrutura de serviços de comunicação chamada *CommFabric*. Essa abstração serve para dissociar os serviços de comunicação da implementação dos mesmos, de forma a ser mais portável e de fácil manutenção.

3.2.1 Infra-estrutura

A implementação de *CommFabric* usada pelo ***BGLsim*** usa MPI[15] [13] para realizar a comunicação entre os diversos processos participantes da simulação. Assim, todo o ambiente de simulação pode ser encarado como uma aplicação paralela MPI. Isso torna o gerenciamento dos diversos processos distribuídos algo menos complexo e mais robusto.

O *CommFabric* oferece um conjunto abrangente de serviços de comunicação de alto nível para os componentes de simulação. Dentre os serviços estão chamadas de função específicas para as diversas redes virtuais oferecidas pelo ***BGLsim***. Por exemplo, para a rede Ethernet, o modelo da interface de rede, quando deseja enviar um pacote para a rede, chama a função `sendEthernet(void *packet, int len)` do *CommFabric*, que, por sua vez, decide para onde o pacote deve ir e o envia.

Todos os processos envolvidos no ambiente de simulação, sejam eles simuladores de nós, *ETHchips* ou *gateways* Ethernet, possuem uma estrutura de dados que descreve toda a topologia da máquina sendo simulada e quais os processos responsáveis pela simulação dos diversos componentes. Isso permite que o *CommFabric* decida para onde as mensagens devem ser enviadas dentro do ambiente.

Tanto o envio quanto o recebimento de mensagens no ambiente de simulação são eventos assíncronos. Por isso, para evitar ter de verificar o recebimento de pacotes a cada ciclo ou conjunto de ciclos de simulação, existe uma *thread* de execução responsável por verificar a chegada mensagens e realizar roteamentos caso seja necessário.

[15] *message passing interface*

O pacote MPI[16] utilizado pela implementação do *CommFabric* foi o LAM-MPI[17]. Um aspecto interessante observado durante a implementação dos serviços do *CommFabric* foi o fato da biblioteca MPI não ser reentrante e, portanto, não ser *multi-threaded*. Por isso, foi necessário garantir que não haviam chamadas MPI sendo executadas ao mesmo tempo em *threads* diferentes dentro de um mesmo processo. Isso foi implementado através de um *mutex* global para chamadas MPI, que deveria ser adquirido antes de qualquer chamada e liberado logo após a chamada. Além disso, para evitar *deadlocks*, foi necessário fazer todas as chamadas de recebimento *non-blocking*, pois elas são executadas pela *thread* de comunicação freqüentemente.

3.3 Instrumentação

Para um ambiente de simulação de sistemas ser realmente útil, é necessário que seja possível extrair informações que permitam entender o comportamento de aplicações, compiladores e sistemas operacionais.

O ***BGLsim*** possui diversos mecanismos de extração de informações. Esses mecanismos englobam recursos de geração de *traces* de execução de programas, histograma de instruções, acessos à memória e identificação de processos sendo executados no sistema operacional usado sobre o simulador (atualmente Linux), entre outros.

3.3.1 Integração com o sistema operacional

Os programas executados sobre o simulador em geral rodam sobre um sistema operacional que também roda sobre o simulador. Porém, principalmente no caso do sistema operacional Linux, sempre existe mais de um processo rodando. Isso atrapalha quando se quer extrair informações sobre um programa, pois vários programas estão rodando e influindo na extração de informações do programa desejado.

Para contornar esse problema, o *kernel* do sistema operacional Linux [2] foi alterado para notificar, através de uma instrução especial, qual o processo sendo executado num dado momento. Com isso, é possível filtrar informações como *traces*, histograma de execução de instruções, etc, com no processo.

4 Validação

Uma das etapas mais importantes no desenvolvimento de um ambiente de simulação é a validação, pois é necessário ter uma garantia que a simulação disponibilizada é próxima o suficiente da realidade. Porém, quanto mais complexo o sistema simulado, mais difícil é a sua validação.

Como o ***BGLsim*** é um simulador de um computador, é necessário saber se o ***BGLsim*** é capaz de executar os programas que a máquina real executaria. Outro aspecto da validação é saber o quão tolerante é o ***BGLsim*** perante o futuro *hardware*, pois o inverso pode ocorrer: um programa pode ser executado corretamente no ***BGLsim*** e falhar quando executado no *hardware*.

Um aspecto que dificulta muito a validação do ***BGLsim***, é o fato da máquina real não estar disponível. Isso faz com que se tenha que assumir muitos fatores sem a certeza de sua real adequação.

A metodologia de validação do ***BGLsim*** em termos da simulação de um nó foi bem direta: o primeiro passo era a escrita e execução de programas de teste em linguagem *assembly* para ensaiar a simulação de instruções e dispositivos. Depois, através do uso de *cross*-compiladores, foram escritos programas mais elaborados, capazes de testar se seu comportamento estava correto. Também foram usados programas prontos para ensaiar o simulador, como o teste `Paranoia` para unidades de ponto flutuante. Um teste de validação muito importante foi o processo de *boot* do sistema operacional Linux.

[16] biblioteca de funções e utilitários de gerenciamento de aplicações

[17] *Local Area Multicomputer*[12], implementação *open-source* do padrão MPI 1.1 realizada pela Universidade de Notre-Dame

Além da validação da simulação de um nó foi necessário, também, validar a simulação das redes de interconexão. A validação das redes de interconexão e da infra-estrutura de comunicação entre os nós foi realizada através da execução de aplicações paralelas que fazem uso dos dispositivos de interconexão intensivamente.

Outra funcionalidade do ambiente de simulação que também deve ser validada é o sistema de controle, cujo principal componente é o simulador de *ETHchip*. Isso será discutido em ??.

Apesar da dificuldade, o ***BGLsim*** foi validado com um certo grau de profundidade que fosse possível assumir o ***BGLsim*** como um ambiente de simulação confiável o suficiente para o desenvolvimento e teste de *software*.

5 Conclusões e Trabalhos Futuros

O ***BGLsim*** está sendo uma ferramenta muito útil no desenvolvimento de *software* de sistema e aplicações para o Blue Gene/L. Seu desenvolvimento também culminou no descobrimento de alguns problemas na arquitetura da máquina, uma vez que o próprio desenvolvimento do ***BGLsim*** estava ocorrendo em paralelo com o processo de projeto do *hardware* do Blue Gene/L.

O desenvolvimento do ***BGLsim*** motivou o estudo de diversos conceitos de arquitetura de sistemas paralelos e principalmente de programação paralela, dado que o ***BGLsim*** não é uma aplicação paralela trivial.

O ***BGLsim*** foi muito útil para a depuração de um compilador capaz de gerar código para a unidade de ponto flutuante dupla, uma vez que era a única ferramenta capaz de executar código gerado para essa unidade.

O ***BGLsim*** mostrou-se escalável em termos de número de nós simulados, tendo sido ensaiado em configurações de até 125 nós em um *cluster* de *hosts* com 30 nós, sem uma queda significativa no desempenho em número de instruções por segundo por nó em relação à referência do desempenho de um nó isolado.

O ***BGLsim*** mostrou-se um simulador de alto desempenho, sendo capaz de executar até 2 milhões de instruções por segundo em um *host* Intel Pentium III de 1 GHz de *clock*.

5.1 Trabalhos Futuros

Um possível trabalho futuro para o ***BGLsim*** é um pequeno modelo de tempo capaz de fornecer uma estimativa, mesmo que grosseira, do número de ciclos usados por um dado programa. Esse modelo deverá se basear em informações provenientes dos modelos de cache para a determinação da latência de acessos à memória e, além disso, também deverá modelar a contenção em recursos como registradores, unidade funcionais, *buses* e dispositivos.

Outro trabalho futuro seria o desenvolvimento de um modelo de energia para o ***BGLsim*** que fosse capaz de prover uma estimativa de energia gasta no uso de um determinado sistema de *software*. Esse modelo está, de certa forma, relacionado com o modelo de tempo mencionado no parágrafo anterior, uma vez que o tempo que as operações levam para serem completadas tem uma relação direta com a energia que consomem.

Por fim, poderia ser desenvolvido um mecanismo de injeção de falhas, de forma a ser possível estudar como um sistema de *software* se comporta na presença de falhas. Isso é particularmente importante quando o sistema tem um grande número de componentes, como o Blue Gene/L, no qual componentes falharão freqüentemente e o *software* do sistema deve ser capaz de se manter funcional.

Referências

[1] Almasi, G. S., Cascaval, C., Castanos, J. G., Denneau, M., Donath, W., Eleftheriou, M., Giampapa, M., Ho, H., Lieber, D., Moreira, J. E., Newns, D., Snir, M.,

AND HENRY S. WARREN, J. Demonstrating the scalability of a molecular dynamics application on a petaflop computer. Tech. Rep. RC-21965, IBM Research, February 2001.

[2] BOVET, D., AND CESATI, M. *Understanding the Linux Kernel.* O'Reilly, October 2000.

[3] BURGER, D., AND AUSTIN, T. M. The simplescalar tool set, version 2.0. Tech. Rep. 1342, University of Wisconsin-Madison and Intel Corporation, June 1997.

[4] ET AL., G. A. Cellular supercomputing with system-on-a-chip. In *Proceedings of International Solid-State Circuits Conference (ISSCC)* (Yorktown Heights, NY, February 2002), IBM T J Watson Research Center and IBM Enterprise Server Group.

[5] ET AL., P. M. Simics: A full system simulation platform. *Computer Magazine sn* (February 2002).

[6] FALSAFI, B., AND WOOD, D. A. Cost/performance of a parallel computer simulator. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation(PADS '94)* (USA, 1994).

[7] HERROD, S., ROSENBLUM, M., BUGNION, E., DEVINE, S., BOSCH, R., CHAPIN, J., GOVIL, K., TEODOSIU, D., WITCHEL, E., AND VERGHESE, B. The SimOS simulation environment. Tech. rep., Computer Systems Laboratory - Stanford University, February 1996.

[8] HOENICKE, D. *Class Routed Network Interface.* IBM T.J.Watson Research Center, Yorktown Heights, NY, Maio 2002. v0.7.

[9] IBM - MICROELETRONICS RESEARCH TRIANGLE PARK, NC. *EMAC4 (Ethernet Core) User Manual.*

[10] IBM - MICROELETRONICS RESEARCH TRIANGLE PARK, NC. *PowerPC 440 User Manual.*

[11] Padrão ieee 1149.1 para portas de teste em circuitos integrados. `http://standards.ieee.org/reading/ieee/std\_public/description/testtech%/1149.1-1990\_desc.html`.

[12] LAM / MPI parallel computing. `http://www.lam-mpi.org/`.

[13] Mpi: A message-passing interface standard. `http://www-unix.mcs.anl.gov/mpi/`.

[14] MUKHERJEE, S. S., ADVE, S. V., AUSTIN, T., EMER, J., AND MAGNUSSON, P. S. Performance simulation tools. *Computer Magazine sn* (February 2002).

[15] SHARMA, A., NGUYEN, A.-T., AND TORRELLAS, J. Augmint - a multiprocessor simulation environment for intel x86 architectures. Tech. rep., Center for Supercomputing Research and Development - University of Illinois at Urbana-Champaign, March 1996.

[16] SIPPER, M. The emergence of cellular computing. *IEEE Computer Magazine sn* (July 1999).

[17] STEINMACHER-BUROW, B. D. *The Torus from the Software Point of View.* IBM T.J.Watson Research Center, Yorktown Heights, NY, 2002.

[18] Virtutech website. `http://www.virtutech.com`.

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PCS/9301 - Interligação de Processadores através de Chaves Ômicron - GERALDO LINO DE CAMPOS, DEMI GETSCHKO

BT/PCS/9302 - Implementação de Transparência em Sistema Distribuído - LUÍSA YUMIKO AKAO, JOÃO JOSÉ NETO

BT/PCS/9303 - Desenvolvimento de Sistemas Especificados em SDL - SIDNEI H. TANO, SELMA S. S. MELNIKOFF

BT/PCS/9304 - Um Modelo Formal para Sistemas Digitais à Nível de Transferência de Registradores - JOSÉ EDUARDO MOREIRA, WILSON VICENTE RUGGIERO

BT/PCS/9305 - Uma Ferramenta para o Desenvolvimento de Protótipos de Programas Concorrentes - JORGE KINOSHITA, JOÃO JOSÉ NETO

BT/PCS/9306 - Uma Ferramenta de Monitoração para um Núcleo de Resolução Distribuída de Problemas Orientado a Objetos - JAIME SIMÃO SICHMAN, ELERI CARDOSO

BT/PCS/9307 - Uma Análise das Técnicas Reversíveis de Compressão de Dados - MÁRIO CESAR GOMES SEGURA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9308 - Proposta de Rede Digital de Sistemas Integrados para Navio - CESAR DE ALVARENGA JACOBY, MOACYR MARTUCCI JR.

BT/PCS/9309 - Sistemas UNIX para Tempo Real - PAULO CESAR CORIGLIANO, JOÃO JOSÉ NETO

BT/PCS/9310 - Projeto de uma Unidade de Matching Store baseada em Memória Paginada para uma Máquina Fluxo de Dados Distribuído - EDUARDO MARQUES, CLAUDIO KIRNER

BT/PCS/9401 - Implementação de Arquiteturas Abertas: Uma Aplicação na Automação da Manufatura - JORGE LUIS RISCO BECERRA, MOACYR MARTUCCI JR.

BT/PCS/9402 - Modelamento Geométrico usando do Operadores Topológicos de Euler - GERALDO MACIEL DA FONSECA, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9403 - Segmentação de Imagens aplicada a Reconhecimento Automático de Alvos - LEONCIO CLARO DE BARROS NETO, ANTONIO MARCOS DE AGUIRRA MASSOLA

BT/PCS/9404 - Metodologia e Ambiente para Reutilização de Software Baseado em Composição - LEONARDO PUJATTI, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9405 - Desenvolvimento de uma Solução para a Supervisão e Integração de Células de Manufatura Discreta - JOSÉ BENEDITO DE ALMEIDA, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9406 - Método de Teste de Sincronização para Programas em ADA - EDUARDO T. MATSUDA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/9407 - Um Compilador Paralelizante com Detecção de Paralelismo na Linguagem Intermediária - HSUEH TSUNG HSIANG, LÍRIA MATSUMOTO SAITO

BT/PCS/9408 - Modelamento de Sistemas com Redes de Petri Interpretadas - CARLOS ALBERTO SANGIORGIO, WILSON V. RUGGIERO

BT/PCS/9501 - Sintese de Voz com Qualidade - EVANDRO BACCI GOUVÊA, GERALDO LINO DE CAMPOS

BT/PCS/9502 - Um Simulador de Arquiteturas de Computadores "A Computer Architecture Simulator" - CLAUDIO A. PRADO, WILSON V. RUGGIERO

BT/PCS/9503 - Simulador para Avaliação da Confiabilidade de Sistemas Redundantes com Reparo - ANDRÉA LUCIA BRAGA, FRANCISCO JOSÉ DE OLIVEIRA DIAS

BT/PCS/9504 - Projeto Conceitual e Projeto Básico do Nível de Coordenação de um Sistema Aberto de Automação, Utilizando Conceitos de Orientação a Objetos - NELSON TANOMARU, MOACYR MARTUCCI JUNIOR

BT/PCS/9505 - Uma Experiência no Gerenciamento da Produção de Software - RICARDO LUIS DE AZEVEDO DA ROCHA, JOÃO JOSÉ NETO

BT/PCS/9506 - MétodOO - Método de Desenvolvimento de Sistemas Orientado a Objetos: Uma Abordagem Integrada à Análise Estruturada e Redes de Petri - KECHI HIRAMA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/9601 - MOOPP: Uma Metodologia Orientada a Objetos para Desenvolvimento de Software para Processamento Paralelo - ELISA HATSUE MORIYA HUZITA, LÍRIA MATSUMOTO SATO

BT/PCS/9602 - Estudo do Espalhamento Brillouin Estimulado em Fibras Ópticas Monomodo - LUIS MEREGE SANCHES, CHARLES ARTUR SANTOS DE OLIVEIRA

BT/PCS/9603 - Programação Paralela com Variáveis Compartilhadas para Sistemas Distribuídos - LUCIANA BEZERRA ARANTES, LIRIA MATSUMOTO SATO

BT/PCS/9604 - Uma Metodologia de Projeto de Redes Locais - TEREZA CRISTINA MELO DE BRITO CARVALHO, WILSON VICENTE RUGGIERO

BT/PCS/9605 - Desenvolvimento de Sistema para Conversão de Textos em Fonemas no Idioma Português - DIMAS TREVIZAN CHBANE, GERALDO LINO DE CAMPOS

BT/PCS/9606 - Sincronização de Fluxos Multimidia em um Sistema de Videoconferência - EDUARDO S. C. TAKAHASHI, STEFANIA STIUBIENER

BT/PCS/9607 - A importância da Completeza na Especificação de Sistemas de Segurança - JOÃO BATISTA CAMARGO JÚNIOR, BENÍCIO JOSÉ DE SOUZA

BT/PCS/9608 - Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar Exceções em Sistemas de Frames com Múltipla Herança - BRÁULIO COELHO ÁVILA, MÁRCIO RILLO

BT/PCS/9609 - Implementação de Engenharia Simultânea - MARCIO MOREIRA DA SILVA, MOACYR MARTUCCI JÚNIOR

BT/PCS/9610 - Statecharts Adaptativos - Um Exemplo de Aplicação do STAD - JORGE RADY DE ALMEIDA JUNIOR, JOÃO JOSÉ NETO

BT/PCS/9611 - Um Meta-Editor Dirigido por Sintaxe - MARGARETE KEIKO IWAI, JOÃO JOSÉ NETO

BT/PCS/9612 - Reutilização em Software Orientado a Objetos: Um Estudo Empírico para Analisar a Dificuldade de Localização e Entendimento de Classes - SELMA SHIN SHIMIZU MELNIKOFF, PEDRO ALEXANDRE DE OLIVEIRA GIOVANI

BT/PCS/9613 - Representação de Estruturas de Conhecimento em Sistemas de Banco de Dados - JUDITH PAVÓN MENDONZA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9701 - Uma Experiência na Construção de um Tradutor Inglês - Português - JORGE KINOSHITA, JOÃO JOSÉ NETO

BT/PCS/9702 - Combinando Análise de "Wavelet" e Análise Entrópica para Avaliar os Fenômenos de Difusão e Correlação - RUI CHUO HUEI CHIOU, MARIA ALICE G. V. FERREIRA

BT/PCS/9703 - Um Método para Desenvolvimento de Sistemas de Computacionais de Apoio a Projetos de Engenharia - JOSÉ EDUARDO ZINDEL DEBONI, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9704 - O Sistema de Posicionamento Global (GPS) e suas Aplicações - SÉRGIO MIRANDA PAZ, CARLOS EDUARDO CUGNASCA

BT/PCS/9705 - METAMBI-OO - Um Ambiente de Apoio ao Aprendizado da Técnica Orientada a Objetos - JOÃO UMBERTO FURQUIM DE SOUZA, SELMA S. S. MELNIKOFF

BT/PCS/9706 - Um Ambiente Interativo para Visualização do Comportamento Dinâmico de Algoritmos - IZAURA CRISTINA ARAÚJO, JOÃO JOSÉ NETO

BT/PCS/9707 - Metodologia Orientada a Objetos e sua Aplicação em Sistemas de CAD Baseado em "Features" - CARLOS CÉSAR TANAKA, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9708 - Um Tutor Inteligente para Análise Orientada a Objetos - MARIA EMÍLIA GOMES SOBRAL, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9709 - Metodologia para Seleção de Solução de Sistema de Aquisição de Dados para Aplicações de Pequeno Porte - MARCELO FINGUERMAN, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9801 - Conexões Virtuais em Redes ATM e Escalabilidade de Sistemas de Transmissão de Dados sem Conexão - WAGNER LUIZ ZUCCHI, WILSON VICENTE RUGGIERO

BT/PCS/9802 - Estudo Comparativo dos Sistemas da Qualidade - EDISON SPINA, MOACYR MARTUCCI JR.

BT/PCS/9803 – The VIBRA Multi-Agent Architecture: Integrating Purposive Vision With Deliberative and Reactive Planning – REINALDO A. C. BIANCHI , ANNA H. REALI C. RILLO, LELIANE N. BARROS

BT/PCS/9901 – Metodologia ODP para o Desenvolvimento de Sistemas Abertos de Automação – JORGE LUIS RISCO BECCERRA, MOACYR MARTUCCI JUNIOR

BT/PCS/9902 – Especificação de Um Modelo de Dados Bitemporal Orientado a Objetos – SOLANGE NICE ALVES DE SOUZA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9903 – Implementação Paralela Distribuída da Dissecação Cartesiana Aninhada – HILTON GARCIA FERNANDES, LIRIA MATSUMOTO SATO

BT/PCS/9904 – Metodologia para Especificação e Implementação de Solução de Gerenciamento – SERGIO CLEMENTE, TEREZA CRISTINA MELO DE BRITO CARVALHO

BT/PCS/9905 – Modelagem de Ferramenta Hipermídia Aberta para a Produção de Tutoriais Interativos – LEILA HYODO, ROMERO TORI

BT/PCS/9906 – Métodos de Aplicações da Lógica Paraconsistente Anotada de Anotação com Dois Valores-LPA2v com Construção de Algoritmo e Implementação de Circuitos Eletrônicos – JOÃO I. DA SILVA FILHO, JAIR MINORO ABE

BT/PCS/9907 – Modelo Nebuloso de Confiabilidade Baseado no Modelo de Markov – PAULO SÉRGIO CUGNASCA, MARCO TÚLIO CARVALHO DE ANDRADE

BT/PCS/9908 – Uma Análise Comparativa do Fluxo de Mensagens entre os Modelos da Rede Contractual (RC) e Colisões Baseada em Dependências (CBD) – MÁRCIA ITO, JAIME SIMÃO SICHMAN

BT/PCS/9909 – Otimização de Processo de Inserção Automática de Componentes Eletrônicos Empregando a Técnica de Times Assíncronos – CESAR SCARPINI RABAK, JAIME SIMÃO SICHMAN

BT/PCS/9910 – MIISA – Uma Metodologia para Integração da Informação em Sistemas Abertos – HILDA CARVALHO DE OLIVEIRA, SELMA S. S. MELNICOFF

BT/PCS/9911 – Metodologia para Utilização de Componentes de Software: um estudo de Caso – KAZUTOSI TAKATA, SELMA S. S. MELNIKOFF

BT/PCS/0001 – Método para Engenharia de Requisitos Norteado por Necessidades de Informação – ARISTIDES NOVELLI FILHO, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/0002 – Um Método de Escolha Automática de Soluções Usando Tecnologia Adaptativa – RICARDO LUIS DE AZEVEDO DA ROCHA, JOÃO JOSÉ NETO

BT/PCS/0101 – Gerenciamento Hierárquico de Falhas – JAMIL KALIL NAUFAL JR., JOÃO BATISTA CAMARGO JR.

BT/PCS/0102 – Um Método para a Construção de Analisadores Morfológicos, Aplicado à Língua Portuguesa, Baseado em Autômatos Adaptativos – CARLOS EDUARDO DANTAS DE MENEZES, JOÃO JOSÉ NETO

BT/PCS/0103 – Educação pela Web: Metodologia e Ferramenta de Elaboração de Cursos com Navegação Dinâmica – LUISA ALEYDA GARCIA GONZÁLEZ, WILSON VICENTE RUGGIERO

BT/PCS/0104 – O Desenvolvimento de Sistemas Baseados em Componentes a Partir da Visão de Objetos – RENATA EVANGELISTA ROMARIZ RECCO, JOÃO BATISTA CAMARGO JÚNIOR

BT/PCS/0105 – Introdução às Gramáticas Adaptativas – MARGARETE KEIKO IWAI, JOÃO JOSÉ NETO

BT/PCS/0106 – Automação dos Processos de Controle de Qualidade da Água e Esgoto em Laboratório de Controle Sanitário – JOSÉ BENEDITO DE ALMEIDA, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/01/07 – Um Mecanismo para Distribuição Segura de Vídeo MPEG – CÍNTIA BORGES MARGI, GRAÇA BESSAN, WILSON VICENTE RUGGIERO

BT/PCS/0108 – A Dependence-Based Model for Social Reasoning in Multi-Agent Systems – JAIME SIMÃO SICHMAN

BT/PCS/0109 – Ambiente Multilinguagem de Programação – Aspectos do Projeto e Implementação – APARECIDO VALDEMIR DE FREITAS, JOÃO JOSÉ NETO

BT/PCS/0110 – LETAC: Técnica para Análise de Tarefas e Especificação de Fluxo de Trabalho Cooperativo – MARCOS ROBERTO GREINER, LUCIA VILELA LEITE FILGUEIRAS

BT/PCS/0111 – Modelagem ODP para o Planejamento de Sistemas de Potência – ANIRIO SALLES FILHO, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/0112 – Técnica para Ajuste dos Coeficientes de Quantização do Padrão MPEG em Tempo Real – REGINA M. SILVEIRA, WILSON V. RUGGIERO

BT/PCS/0113 – Segmentação de Imagens por Classificação de Cores: Uma Abordagem Neural – ALEXANDRE S. SIMÕES, ANNA REALI COSTA

BT/PCS/0114 - Uma Avaliação do Sistema DSM Nautilus –MARIO DONATO MARINO, GERALDO LINO DE CAMPOS

BT/PCS/0115 – Utilização de Redes Neurais Artificiais para Construção de Imagem em Câmara de Cintilação – LUIZ SÉRGIO DE SOUZA, EDITH RANZINI

BT/PCS/0116 – Simulação de Redes ATM – HSU CHIH WANG CHANG, WILSON VICENTE RUGGIERO

BT/PCS/0117 – Application of Monoprocessed Architecture for Safety Critical Control Systems – JOSÉ ANTONIO FONSECA, JORGE RADY DE ALMEIDA JR.

BT/PCS/0118 – WebBee – Um Sistema de Informação via WEB para Pesquisa de Abelhas sem Ferrão – RENATO SOUSA DA CUNHA, ANTONIO MOURA SARAIVA

BT/PCS/0119 – Parallel Processing Applied to Robot Manipulator Trajectory Planning – DENIS HAMILTON NOMIYAMA, LÍRIA MATSUMOTO SATO, ANDRÉ RIYUITI HIRAKAWA

BT/PCS/0120 – Utilização de Padrão de Arquitetura de Software para a Fase de Projeto Orientado a Objetos – CRISITINA MARIA FERREIRA DA SILVA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/0121 – Agilizando Aprendizagem por Reforço Através do uso de Conhecimento sobre o Domínio – RENÊ PEGORARO, ANNA H. REALI COSTA

BT/PCS/0122 – Modelo de Segurança da Linguagem Java Problemas e Soluções – CLAUDIO MASSANORI MATAYOSHI, WILSON VICENTE RUGGIERO

BT/PCS/0123 – Proposta de um Agente CNM para o Gerenciamento Web de um Backbone ATM – FERNANDO FROTA REDÍGOLO, TEREZA CRISTINA MELO DE BRITO CARVALHO

BT/PCS/0124 – Um Método de Teste de software Baseado em Casos Teste – SÉRGIO RICARDO ROTTA, KECHI HIRAMA

BT/PCS/0201 – A Teoria Nebulosa Aplicada a uma Bicicleta Ergométrica para Fisioterapia – MARCO ANTONIO GARMS, MARCO TÚLIO CARVALHO DE ANDRADE

BT/PCS/0202 – Synchronization Constraints in a Concurrent Object Oriented Programming Model – LAÍS DO NASCIMENTO SALVADOR, LIRIA MATSUMOTO SATO

BT/PCS/0203 – Construção de um Ambiente de Dados sobre um Sistema de Arquivos Paralelos – JOSÉ CRAVEIRO DA COSTA NETO, LIRIA MATSUMOTO SATO

BT/PCS/0204 – Maestro: Um Middleware para Suporte a Aplicações Distribuídas Baseadas em Componentes de Software – CLÁUDIO LUÍS PEREIRA FERREIRA, JORGE LUÍS RISCO BECERRA

BT/PCS/0205 - Sistemas de Automação dos Transportes (ITS) Descritos Através das Técnicas de Modelagem RM-OPD (ITU-T) e UML (OMG) – CLÁUDIO LUIZ MARTE, JORGE LUÍS RISCO BECERRA, JOSÉ SIDNEI COLOMBO

BT/PCS/0206 – Comparação de Perfis de Usuários Coletados Através do Agente de Interface PersonalSearcher – GUSTAVO A. GIMÉNEZ LUGO, ANALÍA AMANDI, JAIME SIMÃO SICHMAN

BT/PCS/0207 – Arquitetura Reutilizáveis para a Criação de Sistemas de Tutorização Inteligentes – MARCO ANTONIO FURLAN DE SOUZA, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/0208 – Análise e Predição de Desempenho de Programas Paralelos em Redes de Estações de Trabalho – LIN KUAN CHING, LIRIA MATSUMOTO SATO

BT/PCS/0209 – Previsões Financeiras Através de Sistemas Neuronebulosos – DANIEL DE SOUZA GOMES, MARCO TÚLIO CARVALHO DE ANDRADE

BT/PCS/0210 – Proposta de Arquitetura Aberta de Central de Atendimento – ANA PAULA GONÇALVES SERRA, MOACYR MARTUCCI JÚNIOR

BT/PCS/0211 – Alternativas de Implementação de Sistemas Nebulosos em Hardware – MARCOS ALVES PREDEBON, MARCO TÚLIO CA.RVALHO DE ANDRADE

BT/PCS/0212 – Registro de Imagens de Documentos Antigos – VALGUIMA VICTORIA VIANA ODAKURA MARTINEZ, GERALDO LINO DE CAMPOS

BT/PCS/0213 – Um Modelo de Dados Multidimensional – PEDRO WILLEMSENS, JORGE RADY DE ALMEIDA JUNIOR

BT/PCS/0214 – Autômatos Adaptativos no Tratamento Sintático de Linguagem Natural – CÉLIA YUMI OKANO TANIWAKI, JOÃO JOSÉ NETO

BT/PCS/0215 – Fatores e Subfatores para Avaliação da Segurança em Software de Sistemas Críticos – JOÃO EDUARDO PROENÇA PÁSCOA, JOÃO BATISTA CAMARGO JÚNIOR

BT/PCS/0216 – Derivando um Modelo de Projeto a Partir de um Modelo de Análise, com Base em Design Patterns J2EE – SERGIO MARTINS FERNANDES, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/0217 – Domínios Virtuais para Redes Móveis Ad Hoc: Um Mecanismo de Segurança – LEONARDO AUGUSTO MARTUCCI, TEREZA CRISTINA DE MELO BRITO CARVALHO

BT/PCS/0218 – Uma Ferramenta para a Formulação de Consultas Baseadas em Entidades e Papéis – ANDRÉ ROBERTO DORETO SANTOS, EDIT GRASSIANI LINO CAMPOS

BT/PCS/0219 – Avaliação de Performance de Arquiteturas para Computação de Alto Desempenho – KARIN STRAUSS, WILSON VICENTE RUGGIERO