número **1**
novembro 2003
ISSN 1678-8435

Revista de Engenharia de Computação e Sistemas Digitais_

Universidade de São Paulo
Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais - PCS

Revista
de Engenharia de Computação e Sistemas Digitais__

PCS

número **1**
novembro 2003
ISSN 1678-8435

Este é o primeiro número da ***Revista de Engenharia de Computação e Sistemas Digitais*** que terá periodicidade quadrimestral, e tem como objetivo abrir um novo espaço para a publicação de artigos na área. Resulta de um esforço do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo - PCS em divulgar trabalhos do próprio Departamento e da comunidade em geral, através da submissão de artigos por pesquisadores de outras instituições que atuam nas áreas de Computação e Sistemas Digitais.

Este primeiro número é constituído de artigos de pesquisadores do próprio PCS submetidos a um conselho editorial interno. Entretanto, a partir do segundo número, a revista contará com um grupo de pesquisadores conceituados do Brasil e do exterior para comporem o conselho editorial. Por esta razão conclamamos os colegas da área a submeterem seus artigos certos de que juntos construiremos uma revista útil a todos.

Moacyr Martucci Júnior

Diretor do Departamento de Engenharia
de Computação e Sistemas Digitais

Universidade de São Paulo
Escola Politécnica
Departamento de Engenharia de
Computação e Sistemas Digitais - PCS

A ***Revista de Engenharia de Computação e Sistemas Digitais*** (ISSN 1678-8435) é editada pelo Departamento de Engenharia de Computação e Sistemas Digitais - PCS, da Escola Politécnica da Universidade de São Paulo - USP.

Correspondência

Av. Prof. Luciano Gualberto, Travessa 3, nº 158
CEP 05508-900 - Faculdade de Engenharia Elétrica - São Paulo / SP / Brasil

http://revista.pcs.usp.br

Revista
de Engenharia de Computação e Sistemas Digitais

PCS

Índice

número **1**
novembro 2003

Revista
de Engenharia de Computação e Sistemas Digitais

PCS

Índice

Universidade de São Paulo
Escola Politécnica
Departamento de Engenharia de Computação e Sistemas Digitais - PCS

Revista
de Engenharia de Computação e Sistemas Digitais

Artigos

PCS

número **1**

novembro 2003

Universidade de São Paulo
Escola Politécnica

Departamento de Engenharia de
Computação e Sistemas Digitais - PCS

Avaliação de Performance de Arquiteturas para Computação de Alto Desempenho

Karin Strauss
Wilson V. Ruggiero
{kstrauss, wilson}@larc.usp.br

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

Resumo

Este trabalho apresenta a avaliação de desempenho de três componentes de arquitetura inovadora criados no Projeto Blue Gene, da IBM. Com a crescente demanda por poder computacional e a incapacidade de computadores baseados no modelo seqüencial em atender essa demanda, surgiram os computadores paralelos. Uma arquitetura surgida dessa evolução é conhecida como arquitetura celular. Ela baseia-se na simplicidade dos elementos de computação e no vasto paralelismo, sendo que o trabalho conjunto de muitas células resulta em grandes computações. Esse campo, porém, é relativamente novo e pouco explorado. Um dos projetos que pretendem estudar e construir máquinas de arquitetura celular é o Projeto Blue Gene, sendo desenvolvido no IBM T.J.Watson Research Center. Além da inovação na arquitetura da máquina como um todo, também foram propostas inovações específicas como a arquitetura dos próprios processadores, unidades de ponto flutuante e redes de interconexão. Foram escolhidos três dos componentes de arquitetura mais inovadora para o estudo, com o objetivo de avaliar sua capacidade antes mesmo de sua fabricação. Assim, é possível, se necessário, alterar o projeto dos componentes sem grande impacto no orçamento, já que os custos de refazer o processo de fabricação são evitados.

Abstract

In this work, we present a performance analysis of three key architectural features of the IBM Blue Gene Project. The demand for computational power continues to increase at a fast rate. Computers based on the sequencial processing model can no longer meet the performance expectations imposed by the market. Therefore, parallel computers have been used to address this demand. One of the approaches resulting from this evolution into massively parallel systems is known as "cellular computing". Its basic principles are simplicity and vast parallelism. One of the current research projects that are studying and building cellular architecture machines is the Blue Gene Project, at the IBM T.J.Watson Research Center. This project is innovative both from a system architecture perspective and from the perspective of specific technologies being used. New architectural solutions are being developed for the processors, the floating point units, and the interconnection networks. Our goal is to evaluate the behavior of three of the most innovative architectural components of Blue Gene, primarily from a performance perspective, before the machine is actually built. It is then possible to change the design of those components in response to the evaluation results without requiring another fabrication iteration.

1. Introdução

Poder computacional é cada vez mais necessário nos dias de hoje, tanto em aplicações comerciais quanto em aplicações científicas. Por isso, as grandes empresas de computação têm investido muito esforço no desenvolvimento de supercomputadores de grande poder computacional e de altíssimo desempenho. Projetos ambiciosos têm surgido a partir dos investimentos destas empresas, alguns até pretendendo produzir máquinas com poder maior do que o dos 500 maiores supercomputadores atuais somados.

Um desses projetos surgiu no centro de pesquisas T.J.Watson, da IBM. Trata-se do projeto Blue Gene [1][2][3],que visa desenvolver uma série de supercomputadores de arquitetura inovadora. O projeto baseia-se na idéia de se construir máquinas paralelas em forma celular, ou seja, compostas de blocos simples e de mesma estrutura (células), ligados apenas a alguns outros blocos vizinhos semelhantes [4]. Através da ligação desses blocos seria possível construir uma máquina paralela de alto desempenho e grande escalabilidade, já que, caso se necessite de mais poder computacional, basta adicionar mais células.

Além da inovação do ponto de vista de ligação entre os componentes do sistema paralelo e das redes de interconexão, pretende-se, no projeto Blue Gene, inovar em aspectos arquiteturais mais específicos, como na arquitetura do próprio processador e de unidades auxiliares.

Avaliar o sistema como um todo, de uma só vez, é uma tarefa bastante complicada: por isso, divide-se a avaliação em tarefas menores, que enfocam subsistemas mais restritos, tornando a análise mais simples.

Os primeiros subsistemas a passar por avaliação são, normalmente, os de arquitetura mais inovadora, justamente porque seu comportamento ainda não é dominado pelos projetistas. Em seguida, passa-se a fazer avaliações mais abrangentes, examinando a interação entre os diversos componentes do sistema.

Avaliar e validar o ganho de desempenho com certos aspectos arquiteturais do sistema é de grande importância para o sucesso do projeto e, para que isso seja possível antes mesmo de a máquina real existir, são usados simuladores [5][6]. Num projeto de grande porte como o Blue Gene, há vários grupos de pessoas distintos envolvidos. Cada um desses grupos está interessado em aspectos diferentes das máquinas. Assim, simuladores distintos são utilizados para a obtenção de dados em níveis de detalhes variados.

Pode-se, também, obter resultados dos simuladores mais detalhados e mais lentos para calibrar os simuladores mais rápidos e menos detalhados. Dessa forma, aumenta-se a precisão desses últimos, tornando possível estimar o desempenho do sistema com *workloads* reais em tempos razoavelmente pequenos. As avaliações de desempenho em um projeto como esse são numerosas e freqüentemente utilizam-se de um ou mais simuladores, com diferentes níveis de detalhes.

O objetivo principal da pesquisa foi avaliar o desempenho de três componentes diferentes das máquinas em desenvolvimento no projeto Blue Gene.

Os três componentes estudados foram cuidadosamente escolhidos por serem três fatores significativos para a estimativa do desempenho geral dos supercomputadores Blue Gene. Analisar o impacto de cada um dos componentes no desempenho do sistema, e os custos associados a eles, em momento prévio à fabricação das máquinas é importante pois, nessa fase, ainda é possível fazer alterações no projeto.

O primeiro componente é o processador Cyclops. Esse processador é composto por um número grande de unidades de execução, também chamadas de *threads*. Essas unidades de execução compartilham memória *cache*, memória principal e unidades de ponto flutuante. A forma como as *threads* compartilham o *cache* de dados é configurável através da forma de endereçamento.

O Cyclops tem uma arquitetura bastante inovadora, cujo comportamento ainda não é completamente conhecido e dominado. Além disso, é uma das opções de processador que pode ser empregado em supercomputadores Blue Gene. Algumas das grandes preocupações dos projetistas são qual banda de memória pode ser obtida com a utilização do *chip*, se é possível atingir o limite teórico calculado e quais políticas devem ser usadas para que isso aconteça. Assim, a avaliação de desempenho da banda de memória do *chip* é um tópico relevante de pesquisa, que pode ajudar os projetistas a estimar o desempenho geral de uma máquina construída com blocos básicos desse tipo.

O objetivo é, então, estudar o comportamento da taxa de transferência de dados entre as *threads* e a memória principal e verificar se é possível atingir o limite teórico calculado, com várias configurações de *cache* de dados e de distribuição de dados na memória principal.

O segundo componente é uma unidade de ponto flutuante dupla. Essa é uma unidade denominada pela própria IBM como SIMOMD (*Single Instruction, Multiple Operation, Multiple Data*), por ter instruções paralelas, cruzadas, assimétricas e complexas.

A unidade de ponto flutuante dupla também é um projeto bastante novo e pouco explorado. Sua arquitetura apresenta diferenças relevantes em relação a outras unidades de ponto flutuante paralelas, já que possibilita não só operações paralelas como também operações cruzadas, assimétricas e complexas. Isso facilita muitas operações freqüentemente utilizadas em aplicações científicas, incluindo diversas computações de álgebra linear (multiplicação de matrizes, por exemplo) e com números complexos. Para que seja possível aproveitar todo o potencial que a unidade pode oferecer, é necessário que haja suporte de compiladores, e que as aplicações sejam codificadas de forma adequada. Assim, a avaliação é feita sobre o conjunto arquitetura-compilador-aplicação. Como o projeto Blue Gene é bastante voltado à construção de máquinas de alto desempenho para aplicações científicas, e esse componente tem bastante influência no desempenho geral, é importante ter dados a seu respeito.

O objetivo é, então, estudar os ganhos de se ter uma unidade de ponto flutuante dupla como esta em relação a uma unidade simples similar. Nesse caso, a avaliação de desempenho não se restringe apenas à arquitetura. Devem ser também levados em conta o compilador e os algoritmos utilizados.

O terceiro componente é a forma de interconexão entre as células do sistema. Essas células comunicam-se através de várias redes distintas. A rede a ser estudada apresenta uma topologia toroidal e constitui a principal rede para aplicações do Blue Gene.

A arquitetura altamente paralela da máquina a ser construída implica em uma comunicação entre células muito intensa, já que a computação e os dados estão distribuídos entre eles. Esse comportamento de comunicação pode influenciar negativamente o desempenho da computação. Assim, o desempenho da rede de interconexão é, novamente, um fator importante para o desempenho geral da máquina, e deve ser estudado.

O objetivo é, então, avaliar o desempenho de um algoritmo distribuído bastante utilizado em computação científica, verificando a eficácia dessa rede de interconexão.

As demais seções deste artigo estão organizadas da seguinte forma: na seção 2 é apresentada a avaliação de desempenho do processador Cyclops; na seção 3 a avaliação de desempenho da unidade de ponto flutuante dupla é explicada; a seção 4 mostra a avaliação de desempenho da rede de interconexão toroidal estudada; na seção 5 são apresentadas as conclusões e comentados os trabalhos futuros.

2. Cyclops

O *Cyclops* é uma unidade básica que pode ser utilizada na construção de uma das máquinas Blue Gene. Trata-se de uma célula *SMP (Symmetric Multi-Processor)* com múltiplas linhas de execução (*daqui em diante chamadas de threads*), memória embutida e dispositivos de comunicação integrados.

O *Cyclops* é ainda uma arquitetura em desenvolvimento. A configuração do *chip* estudada neste trabalho será descrita a seguir.

O *chip* consiste em 128 *threads* com 64 registradores de 32 *bits* cada. Cada registrador tem precisão simples (32 *bits*) mas é possível agrupar dois registradores em um par para prover precisão dupla (64 *bits*). Além disso, cada *thread* tem um contador de instruções (*PC*) e uma unidade lógico-aritmética de ponto fixo.

Essas *threads* são organizadas em grupos de 4, chamados de *quads*. Cada *quad*, portanto, consiste nessas 4 *threads* que compartilham uma unidade de ponto flutuante e uma unidade de *cache* de dados de 16 KB. Cada dois *quads* compartilham um *cache* de instruções de 32 KB, de associatividade *8-way* e linha de 64 *bytes*. A memória principal, compatilhada por todas as *threads*, consiste em 8 MB, divididos em 16 bancos de 512 KB. O *chip* inclui, também, hardware de interconexão (A-switch e B-switch) e um controlador para acessar memória externa (opcional). Essa configuração pode ser observada na figura 1.

Se duas ou mais *threads* tentam utilizar o mesmo recurso em um mesmo ciclo, uma delas é selecionada como a

vencedora e aloca o recurso. As *threads* restantes esperam até que o recurso esteja disponível. A decisão de qual *threads* tem prioridade é feita utilizando-se a técnica de *round-robin*, para garantir que todas as *threads* terão a mesma chance de utilização de um recurso.

O *chip* Cyclops ainda possui um mecanismo de sincronização entre *threads* por *hardware*. Esse mecanismo permite que se faça sincronizações muito rápidas, da ordem de alguns ciclos após a última *thread* chegar à barreira.

Como já explicado, a memória principal do *chip* consiste em 16 bancos de 512 KB, num total de 8 MB. São utilizados apenas 24 *bits* para o endereçamento físico da memória. Os 8 *bits* restantes são utilizados para definir no *cache* em qual *quad* os dados serão colocados. Os dados podem ser colocados em um *cache* pertencente ao mesmo *quad* da *thread* que os resquisitaram ou em outro *cache* definido por esses 8 *bits*.

Como os dados podem ser colocados em qualquer um dos *caches*, dependendo do *software*, aparece um problema de consistência de dados. Esse problema só pode ser evitado com uma programação cuidadosa, já que não há suporte de *hardware* para isso.

O tempo de acesso em modo *burst* para dois blocos de 32 *bytes* subseqüentes em um banco de memória, ou seja, 64 bytes, é de 12 ciclos. Pode-se fazer acessos simultâneos a todos os bancos de memória. Assim, supondo uma freqüência de excitação de 500 MHz, temos uma banda teórica de 42GB/s (64 B*16 bancos*500 MHz/12 ciclos).

Para o estudo, foi utilizado o STREAM *benchmark* [7] [8] executado sobre um simulador de instruções específico para o chip, que inclui o modelo de *threads* e compartilhamento de recursos e que é capaz de fornecer estimativas de tempo. O STREAM foi desenvolvido especialmente para a medição de banda de memória e consiste em quatro operações com vetores: cópia (Copy), adição (Add), multiplicação por escalar (Scale) e multiplicação-e-soma (Triad).

Os experimentos indicaram que é possível atingir a banda máxima teórica quando se traz os dados para o *cache*

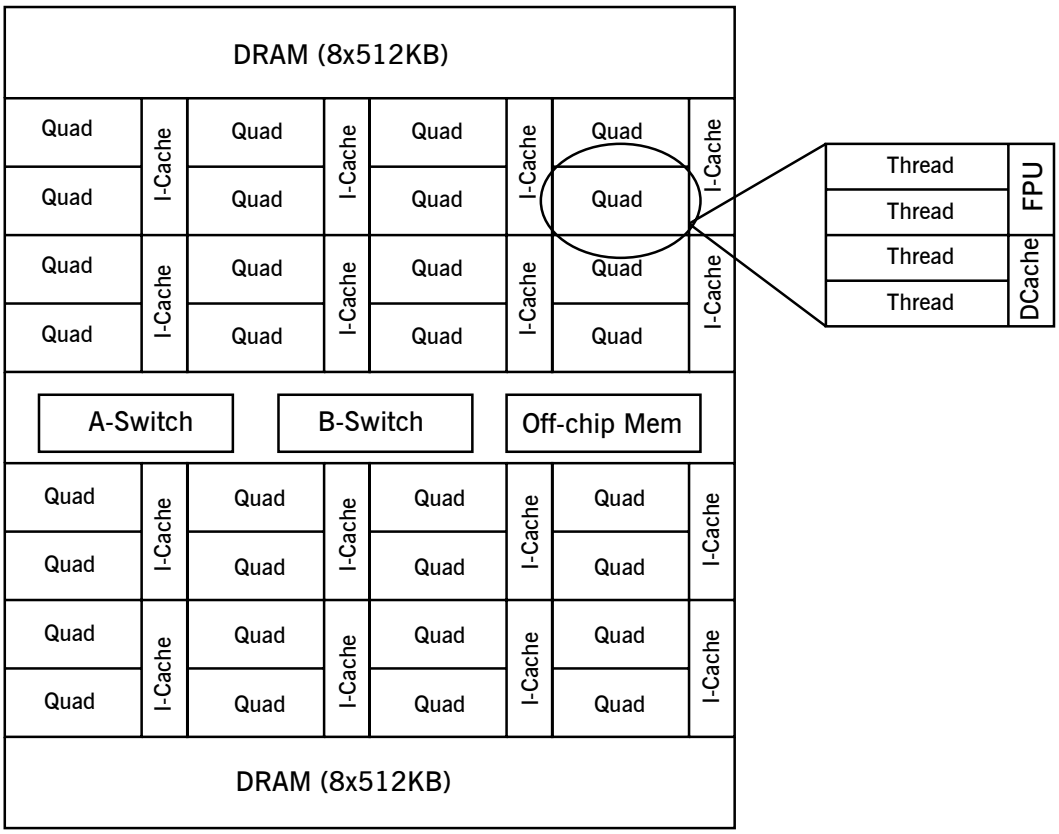


Figura 1: Diagrama de blocos do chip analisado

pertencente ao mesmo *quad* da *thread* que os acessaram. Pode-se comparar o desempenho do *chip* com o de uma máquina comercial, como mostra a figura 2.

Por restrições de espaço, não foi possível descrever todos os resultados obtidos. Para uma análise completa, o leitor interessado pode consultar [9].

Pode-se notar que os vetores usados com o SGI Origin 3800/400 são muito maiores, e que o Cyclops não tem memória suficiente para experimentos com vetores deste tamanho.

De qualquer forma, é importante notar que um único chip do Cyclops pode atingir banda de memória sustentável comparável à atingida por uma máquina comercial topo-de-linha.

3. Unidade de Ponto Flutuante Dupla

A unidade de ponto flutuante dupla estudada, chamada *PowerPC 440FP2* é uma extensão da unidade de ponto flutuante simples projetada para o processador *PowerPC 440*, chamada *PowerPC 440 FPU*. Ela será utilizada em uma das máquinas Blue Gene.

A arquitetura projetada baseia-se na duplicação da unidade original. Uma das unidades é, então, chamada de primária e a outra é chamada de secundária. As instruções originais, ou seja, as instruções normais da unidade de ponto flutuante simples, destinam-se à unidade primária. Foram adicionadas novas instruções ao *instruction-set*, tanto para operações simultâneas nas duas unidades quanto para operações na unidade secundária, estas similares às instruções primárias. Cada uma das unidades tem seu próprio banco de registradores, que podem ser endereçados separadamente. Também é possível endereçar um par de registradores de uma só vez. Pode-se, ainda, mover dados entre os bancos de registradores.

As novas instruções com as duas unidades incluem operações paralelas, cruzadas, assimétricas e complexas. A tabela 1 mostra alguns exemplos. As instruções assimétricas realizam operações diferentes mas relacionadas, nos dois caminhos de dados, enquanto que as instruções complexas podem ser

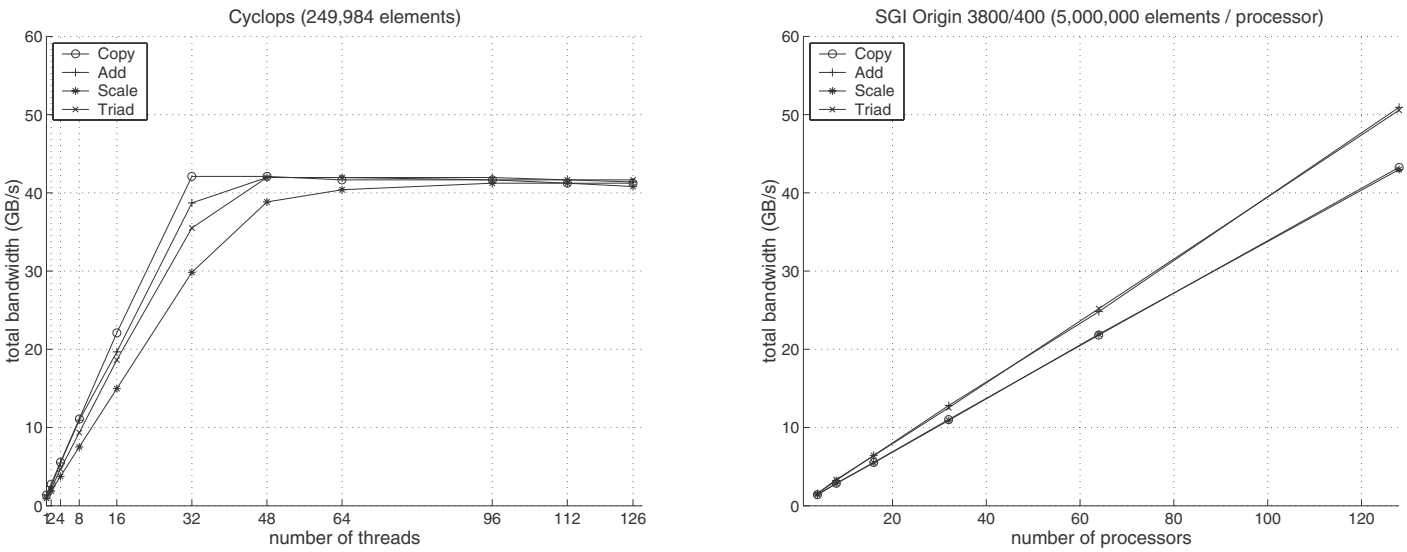


Figura 2: Comparação da performance de (a) Cyclops (tamanho do vetor: 249.984 elementos) *vs.* performance de (b) SGI Origin 3800/400 (tamanho do vetor: 5.000.000 elementos por processador)

simétricas ou assimétricas, mas são especificamente voltadas para computações com números complexos. O nome dado ao tipo de operações que a unidade é capaz de executar é SIMOMD, ou seja, *Single Instruction Multiple Operation Multiple Data*.

É possível, com essa unidade, fazer transferências de dados de palavras de 128 *bits*. Uma palavra de 64 *bits* é colocada no registrador primário selecionado na instrução e a outra é colocada no registrador secundário correspondente. Nesse caso, os registradores são encarados como pares. Se um dado é colocado no terceiro registrador primário, o dado subseqüente será colocado no terceiro registrador secundário. O equivalente pode acontecer se a operação for um *store*.

As operações de movimentação de dados podem ser paralelas, onde a palavra endereçada na instrução é colocada em um registrador primário, também selecionado

Classe	Exemplo		Operação na Unid. Primária	Operação na Unid. Secundária
Paralela	fpmadd	fT, fA, fB, fC	$P_T = P_A * P_C + P_B$	$S_T = S_A * S_C + S_B$
Cruzada	fxcpmadd	fT, fA, fB, fC	$P_T = P_A * S_C + P_B$	$P_T = S_A * S_C + S_B$
	fxmadd	fT, fA, fB, fC	$P_T = P_A * S_C + P_B$	$P_T = S_A * P_C + S_B$
Assimétrica	fxcpnpma	fT, fA, fB, fC	$P_T = -P_A * P_C + P_B$	$S_T = P_A * S_C + S_B$
Complexa	fxcxnpma	fT, fA, fB, fC	$P_T = -S_A * S_C + P_B$	$S_T = S_A * P_C + S_B$

Tabela 1: Exemplo de instruções da *PowerPC 440 FP2*

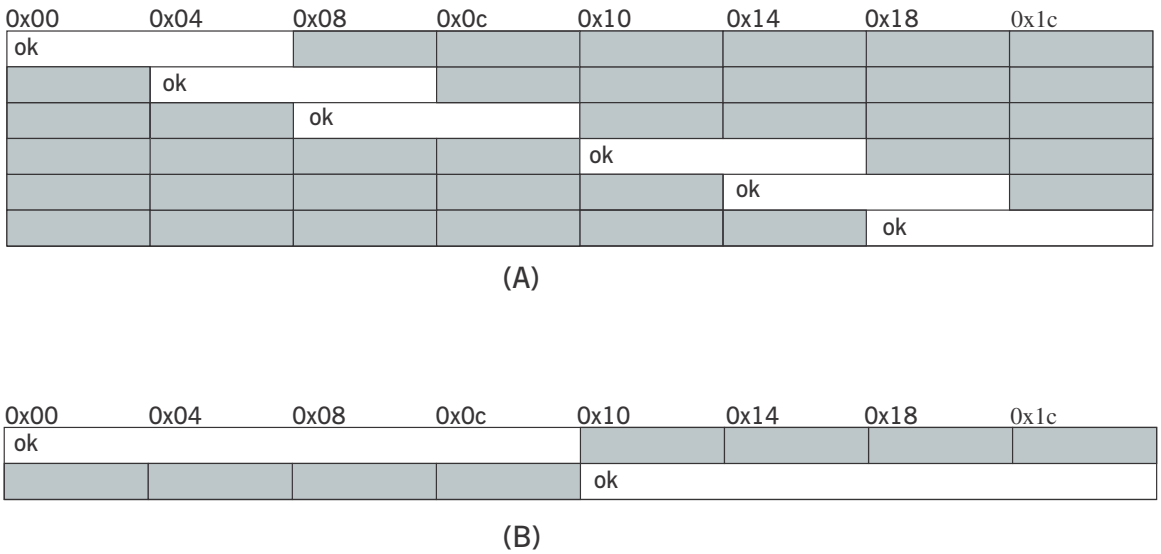


Figura 3: Restrições de alinhamento em uma linha de cache para o *PowerPC 440 FP2*. (a) mostra acessos de 64 *bits* (1 palavra de 64 *bits* ou duas palavras de 32 *bits*) e (b) mostra acessos de 128 *bits* (2 palavras de 64 *bits*).

na instrução, e a palavra consecutiva a ela é colocada no registrador secundário correspondente.

Outro modo de movimentação de dados é a operação cruzada. Nesse caso, a palavra endereçada é colocada em um registrador secundário selecionado na instrução e a palavra seguinte é colocada no registrador primário.

Para que esse tipo de operação seja eficiente, porém, os dados devem estar alinhados em relação a seu tamanho (dados de 32 *bits* alinhados em posições com endereço múltiplo de 4 *bytes*, dados de 64 *bits* alinhados em endereço múltiplo de 8 *bytes*) e precisa se encaixar completamente em uma região alinhada de 256 *bits*, ou 32 *bytes*, que é o tamanho de uma linha do *cache* do *PowerPC 440*. Além disso, uma tranferência não pode cruzar o meio de uma linha de cache. As transferências válidas estão ilustradas na figura 3.

As restrições podem ser mais bem ilustradas pela Figura 3. Elas aparecem porque a arquitetura de *cache* do processador força que transferências de dados só possam ser feitas a partir de uma das metades de uma linha de *cache*.

Se as restrições não forem seguidas, o tempo para que a operação termine aumenta muito (fica da ordem de milhares de ciclos, contra alguns ciclos se as restrições forem atendidas), já que é gerada uma interrupção de alinhamento e o *software* deve forçar o alinhamento dos dados de forma a atender às restrições.

O conjunto de *kernels* utilizados para a avaliação de desempenho da tríade arquitetura-compilador-algoritmo faz parte do BLAS (*Basic Linear Algebra Subprograms*)[10] que, como o próprio nome diz, é um conjunto de rotinas básicas de álgebra linear. Os *kernels* utilizados foram o daxpy (y[n] = a*x[n]+y[n], onde x e y são vetores), dgemv (y[m] = y[m] + A[m][n]*x[n], onde x e y são vetores e A é uma matriz) e dgemm (C[m][n] = C[m][n] + A[m][k]*B[k][n], onde A, B e C são matrizes). Esses *kernels* foram executados sobre um simulador baseado em descrição de *hardware* e o modelo simulado inclui processador e unidade de ponto flutuante.

Um dos resultados obtidos pode ser observado na figura 4.

A versão S e a versão 0 são o código do daxpy sem nenhuma otimização. A única diferença entre elas é o

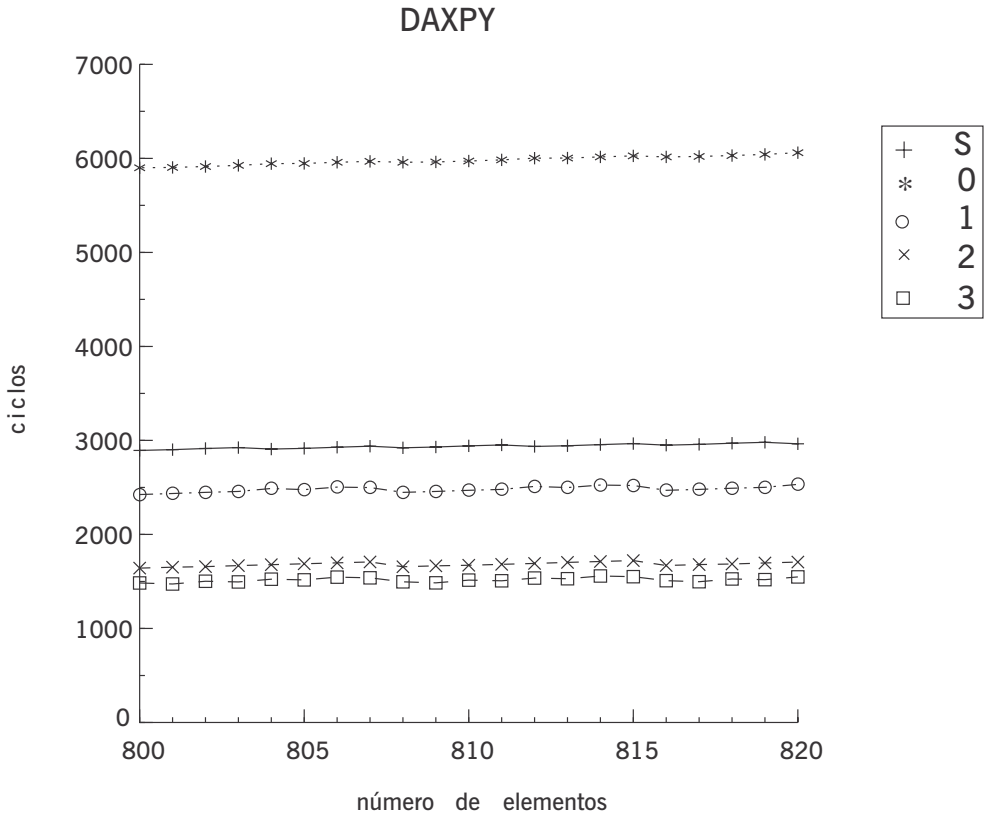


Figura 4: Resultados para o *kernel* **daxpy** para uma faixa contígua de tamanhos de problema (800-820)

compilador com que cada uma é compilada. A versão S é gerada com o compilador normalmente utilizado para gerar código para uma unidade PowerPC 440 FPU padrão. A versão 0 é gerada com o compilador otimizado para a unidade PowerPC 440 FP2. Essa versão tem uma performance inferior porque o compilador não tem informações de alinhamento sobre os vetores do *kernel*. Como se pode observar, o desempenho melhora na versão 1, em que são dadas ao compilador informações sobre alinhamento e sobreposição de vetores. No caso da versão 2, além dessas informações, o *unrolling* e o *scheduling* são feitos manualmente em linguagem C e o desempenho apresentado melhora um pouco mais. Na versão 3, os comandos em liguagem C são substituídos por comandos que controlam as instruções *assembly* geradas. O desempenho, porém, não é significativamente melhor do que na versão 2, o que mostra que o compilador fez um bom trabalho em escolher as instruções *assembly* a serem utilizadas na versão 2.

Os experimentos mostraram que a unidade pode atingir ganhos significativos em desempenho se devidamente programada.

Os resultados indicam que o compilador é capaz de aproveitar os novos recursos da unidade de ponto flutuante quando alimentado com dados sobre alinhamento e sobreposição de vetores e matrizes, principalmente para *kernels* mais simples, com poucos aninhamentos.

O compilador ainda encontra dificuldades em otimizar códigos mais complicados, com vários níveis de aninhamento. É possível, porém, gerar código otimizado para a unidade com o auxílio do recurso de *intrinsics* do compilador, onde o programador tem um maior controle sobre o código de baixo nível gerado, podendo escolher diretamente a seqüência de instruções *assembly* a serem utilizadas em determinados trechos do programa.

4. Rede de Interconexão Toroidal

Numa das máquinas Blue Gene, a rede de interconexão normalmente utilizada para troca de dados de aplicações é a rede toroidal. Assim, cada uma das células é ligada a suas vizinhas através de seis *links*, como mostrado na figura 5. Todas as células responsáveis pela computação de aplicações são interligadas em uma topologia toroidal de três dimensões.

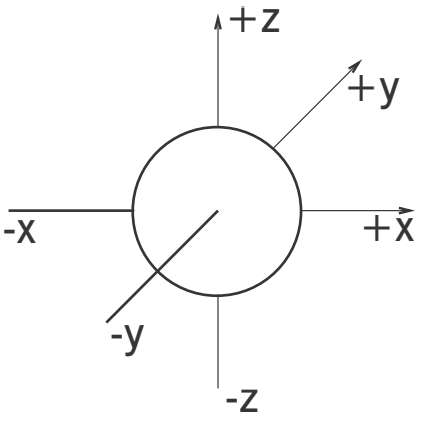


Figura 5: *Links* (com direção e sentido) de uma célula

Os pacotes têm formato proprietário e seu roteamento é feito por *hardware*. Cada pacote tem tamanho fixo de 256 *bytes* e cabeçalho de *hardware* de 8 *bytes*, incluído no pacote. O dispositivo de comunicação da rede toroidal é mapeado em memória.

Todas as transmissões e recepções entre processador e dispositivo de comunicação devem ser feitos em operações de 128 *bits*, ou 16 *bytes*. Assim, toda a interface entre processador e dispositivo é feita através da unidade de ponto flutuante dupla, a única unidade capaz de fazer transferências de dados tão largas.

Como as transmissões são feitas em blocos de 16 *bytes* e o cabeçalho de *hardware* só ocupa 8 *bytes*, decidiu-se utilizar os 8 *bytes* seguintes, já na área de dados, como cabeçalho de *software*. Assim, a primeira parte transmitida de um pacote inclui apenas cabeçalhos e através do cabeçalho de *software* pode-se passar informações a respeito dos dados sendo transmitidos.

O roteamento pode ser feito em modo determinístico, ou seja, é possível determinar qual o caminho seguido pelo pacote da origem para o destino. Outro modo de roteamento é o dinâmico e, nesse caso, não se pode determinar qual o caminho seguido por um pacote: ele depende do tráfego nos *links* de cada uma das células por onde o pacote passa. O modo de roteamento é determinado através de configurações feitas via *software* no cabeçalho do pacote.

Podem-se configurar outras características de roteamento através do cabeçalho do pacote como, por exemplo, os

chamados *hint bits*. Os *hint bits* são destinados a instruir o *hardware* a enviar o pacote em um determinado sentido preferencial.

Outra característica interessante disponível através dessa rede de interconexão é o *multicast* em linha. De acordo com um *bit* configurável do cabeçalho, o chamado *deposit bit*, pode-se instruir o *hardware* a depositar o pacote em cada uma das células por onde o pacote passa até alcançar seu destino.

Esse tipo de operação, porém, só pode ser feita se o caminho que o pacote segue é uma linha, sem mudança de direção. Assim, só se pode enviar pacotes *multicast* se a origem e o destino estiverem em uma mesma linha do toróide.

Como para cada par origem-destino em uma direção do toróide há duas formas de atingir o destino saindo da origem (sentido positivo ou negativo), usa-se os *hint bits* para especificar qual dos caminhos devem ser seguidos.

Como cada direção do toróide está fechada em anel, pode-se, também, enviar pacotes para todas as células em uma linha. Basta que uma célula envie um pacote para si mesma. O *hardware* envia o pacote à célula vizinha, que copia o pacote para si e também o envia à proxima célula. Quando o pacote chega a um extremo, ele é enviado ao outro extremo, dando a volta no anel. O processo se repete até que o pacote atinja a célula originadora (que é o próprio destino). Assim, pode-se fazer um *multicast* total na linha, como mostra a figura 6.

Apesar da limitação de manter o pacote em uma linha quando se usa os *deposit bits*, ainda é possível fazer *broadcasts* que atinjam todas as células, com algum suporte de *software*. Basta para isso que se marque o pacote como *broadcast* em algum campo da área de dados e se envie esse pacote com um *multicast* em linha em uma direção (suponhamos a direção x). Assim, todas as células daquela linha recebem esse pacote. Em seguida, o *software* verifica que aquele é um pacote de *broadcast* e o envia como *multicast* em linha em uma outra direção (digamos que seja a direção y). Assim, todos os pacotes ao longo de todos os anéis da direção inicial (x) recebem uma cópia do pacote. O *software* deve entrar novamente em ação e enviar agora um pacote de *multicast* em linha na direção restante (z, nesse caso). Dessa forma, cada uma das células do sistema recebe uma cópia do pacote inicial. O processo de *broadcast* descrito acima está ilustrado na figura 7.

O desempenho dessa rede de interconexão foi avaliado utilizando-se um simulador de sistema completo especialmente desenvolvido para a máquina. Ele modela

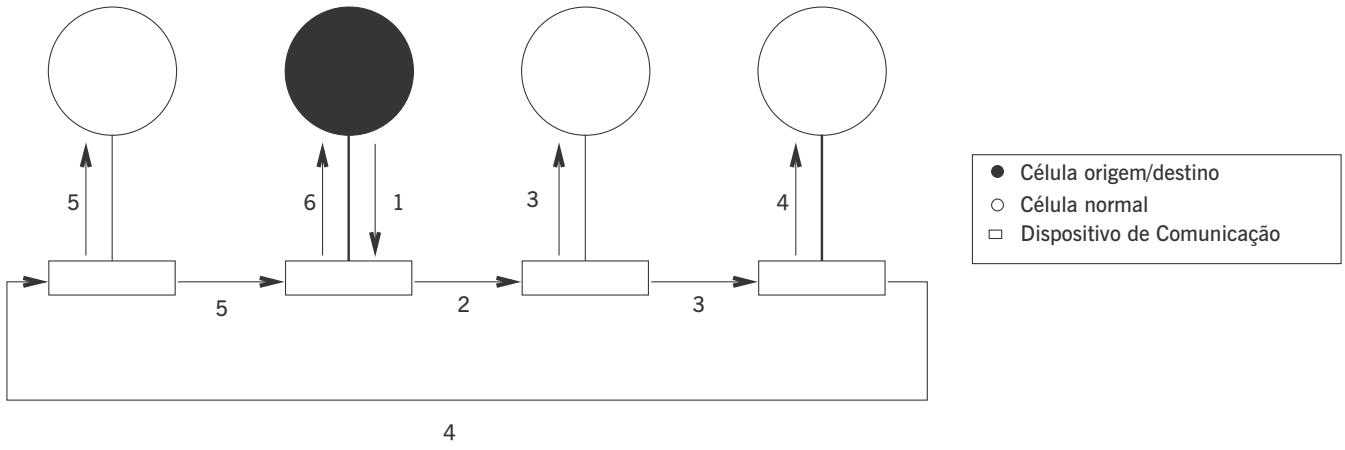


Figura 6: *Multicast* completo em linha: os números mostram o caminho do pacote, partindo da célula origem, sendo depositada e reenviada em cada uma das outras células e voltando para a célula inicial, que também é o destino

várias células e a interconexão que as liga e é capaz de efetivamente simular a troca de dados entre elas pela rede. Assim, é possível executar aplicações paralelas que se utilizem da rede de interconexão para troca de dados e colher informações sobre elas com o simulador.

A aplicação selecionada para a avaliação foi a multiplicação tridimensional de matrizes, adaptada de [11]. Essa aplicação tira proveito do recurso de *multicast* em linha ao distribuir as matrizes no início da computação. A figura 8 mostra como o algoritmo funciona.

A operação a ser feita é C = C + A * B, onde A é uma matriz de dimensões M x K, B é uma matriz de dimensões K x N e C é uma matriz de dimensões M x N. A é dividida em submatrizes A_{00}, A_{01}, A_{10} e A_{11}, cada uma delas de dimensões $\frac{M}{P} \times \frac{K}{P}$ onde p é o número de células presentes em uma das arestas do cubo, ou M x K. Em seguida, essas submatrizes são distribuídas entre as quatro células de uma das faces do cubo. B é dividida da mesma forma e distribuída em uma face ortogonal à face de A. A matriz A deve ser colocada de forma transposta em sua face, para que suas linhas ocupem colunas ao longo da face e as multiplicações entre partes de A e B possam ser feitas de forma mais simples, elemento a elemento em colunas. Essa distribuição pode ser observada na figura 8(a). A matriz C inicial também é dividida e colocada na base do cubo, na direção ortogonal às duas primeiras. Passa-se, então, à fase de distribuição dos elementos. As células pertencentes às faces de A fazem um *multicast* em linha com a sua submatriz, na reta perpendicular à face em que estão situadas, conforme é mostrado na figura 8(b). Simultaneamente, o mesmo acontece com as submatrizes contidas células da matriz B, como mostrado na figura 8(c). Em seguida, as células da base, que contêm elementos de A, B e C (inicial) iniciam a computação. Quando a computação termina, essas células enviam o resultado C' apenas para as células diretamente acima delas, como mostrado na figura 8(d). Essas células, por sua vez, dão início à sua parte da computação. Quando a computação termina nas células do topo, o resultado está distribuído entre as células pertencentes à face do topo do cubo, como mostrado na figura 8(e).

A figura 9 mostra alguns dos resultados obtidos.

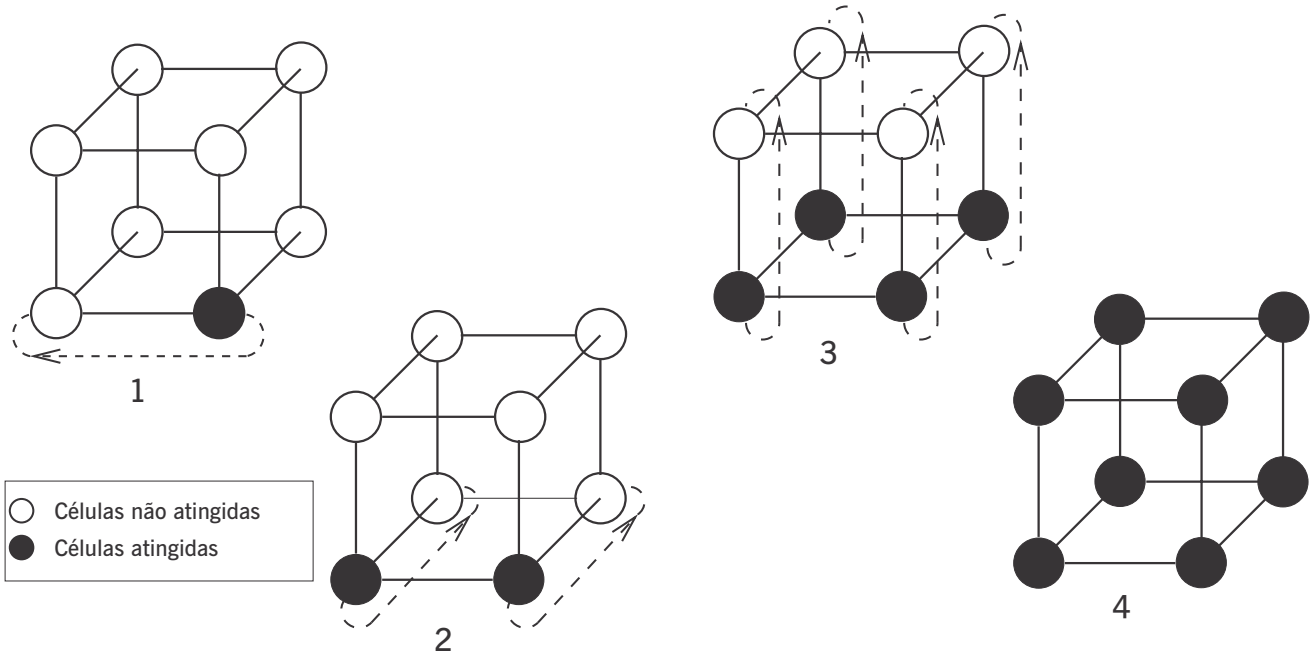


Figura 7: *Broadcast* com o auxílio de software: quatro etapas. (1) dados saem da origem, (2) dados terminam de percorrer eixo x, (3) dados terminam de percorrer eixo y e (4) dados terminam de percorrer eixoz.

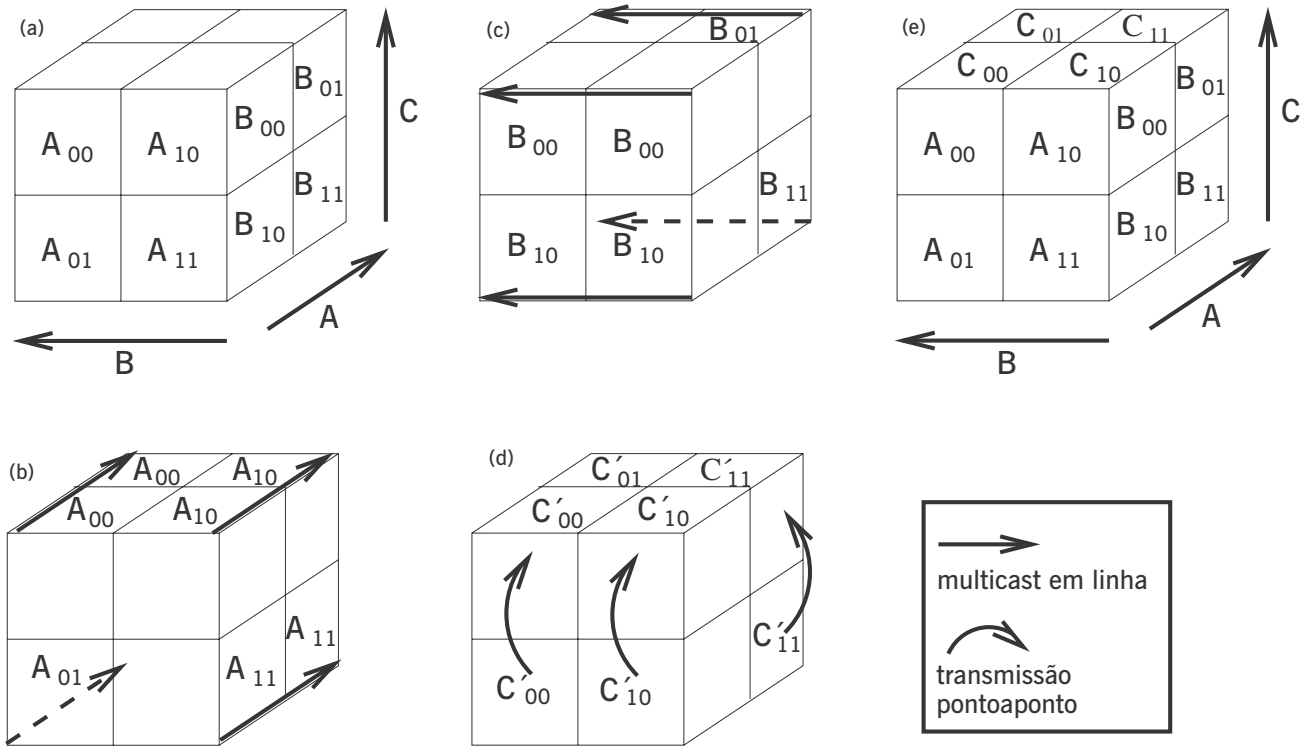


Figura 8: Algoritmo de multiplicação de matrizes tridimensional

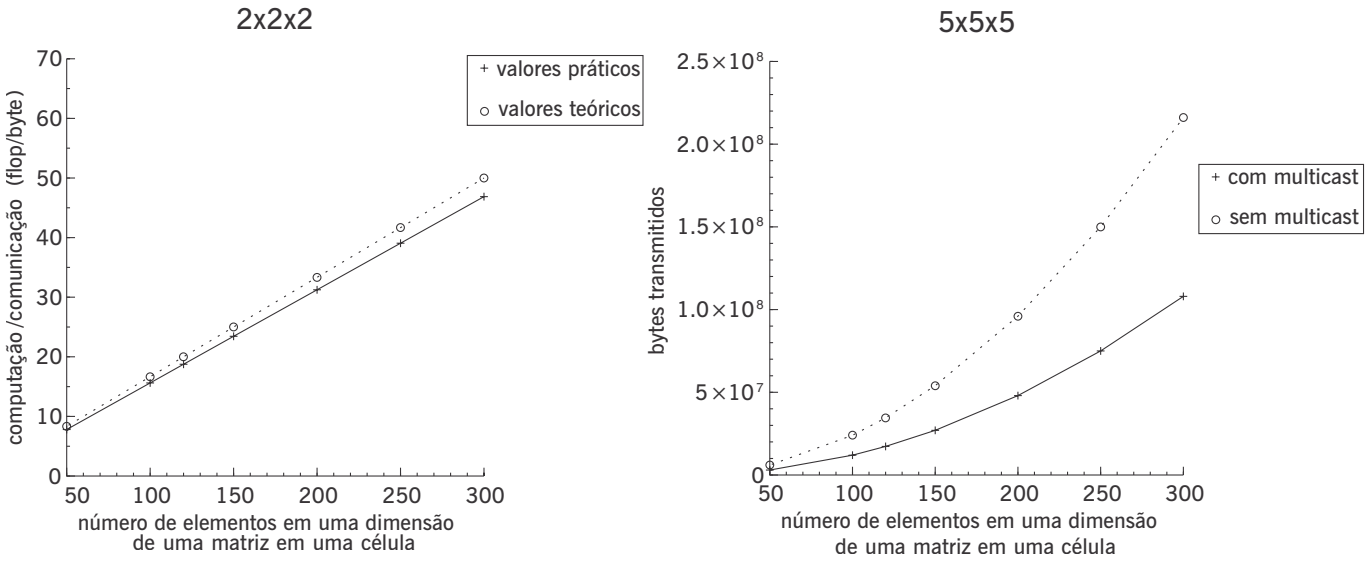


Figura 9: (a) Resultados para o **dgemm** tridimensional com número de células fixo e variação do tamanho do problema
(b) Comparação entre desempenho de sistemas que não dispõem do *multicast* em linha e de sistemas que dispõem do recurso

Foi possível observar pelos experimentos (fig. 9-a) que o *overhead* causado pelo cabeçalho do pacote de dados é relativamente baixo, não tendo um impacto significativo no desempenho do sistema.

Outro resultado importante é que o *multicast* em linha é uma grande ferramenta para algoritmos tridimensionais onde uma informação precisa ser replicada em um subconjunto de células, pois evita que seja necessário transmitir essa informação várias vezes (fig. 9-b). Vale ressaltar que o *multicast* em linha diminui a quantidade de comunicação necessária para a resolução do problema. Diminuindo a quantidade de informações enviadas, ganha-se tempo de processamento para a computação propriamente dita, tempo esse que estaria sendo utilizado para o envio das informações. É importante perceber que não necessariamente se diminui a ocupação nos *links* de comunicação, já que a informação transmitida por *multicast* trafega da mesma forma pelos *links*, ao longo de todas as células envolvidas.

5. Conclusões e Trabalhos Futuros

Os simuladores mostraram-se ferramentas muito valiosas para estudos de avaliação de desempenho.

Para este estudo foram utilizados diferentes simuladores para diferentes propósitos. Cada simulador tem seus pontos fortes e fracos, sendo úteis para obter resultados específicos diferentes.

Foi possível verificar que o Cyclops é capaz, na prática, de atingir os níveis teóricos de banda sustentável de memória, podendo ser comparado a máquinas comerciais topo-de-linha. O único fator limitante para o uso do *chip* individualmente é a quantidade de memória disponível. Assim, o uso do *chip* para aplicações paralelas que utilizam poucos dados em cada célula parece ser o indicado.

A unidade de ponto flutuante dupla estudada pode ser bastante utilizada para aplicações científicas. Ela provê uma maneira eficiente de aumentar o desempenho desse tipo de aplicação, sem aumentar muito a complexidade do *hardware*.

Porém, é muito importante que haja suporte de ferramentas para a geração de código para essa unidade. O compilador estudado é bastante eficiente para *kernels* simples. Conforme a complexidade dos *kernels* aumenta, o compilador apresenta mais dificuldades para gerar código eficiente.

Cabe, então, ao programador otimizar o código manualmente para a unidade, tarefa suportada pelo compilador através do recurso chamado de *intrinsics*.

A topologia toroidal mostrou-se adequada para a computação tridimensional de multiplicação de matrizes. Outras aplicações adequadas ao conceito de computação celular também parecem se adequar à topologia. O *overhead* observado não causa impacto significativo no desempenho como um todo.

O recurso de *multicast* em linha mostrou-se bastante eficiente e útil para aplicações onde é necessário transmitir a mesma informação para diversas células.

No caso do Cyclops, ainda é necessário estudar outros aspectos que não apenas a banda de memória atingida pelo sistema.

Outros benchmarks relacionados a aplicações científicas podem ser utilizados, como o BLAS, para a avaliação do compartilhamento das unidades de ponto flutuante entre as *threads*.

Além disso, o recurso de barreiras rápidas implementadas em *hardware* pode ser mais explorado utilizando aplicações altamente paralelizáveis que necessitem de sincronização.

No caso da unidade de ponto flutuante dupla, pode-se continuar o estudo com *kernels* que utilizem operações complexas, como por exemplo o zdgemm.

Kernels de álgebra linear esparsa, ou mesmo outros *kernels* bastante utilizados em aplicações científicas, como por exemplo o FFT *(Fast Fourier Transforms)*, podem também ser utilizados para aprofundar o estudo.

O estudo da rede de topologia toroidal, pode ser aprofundado utilizando outros *kernels* científicos paralelizáveis.

Se uma biblioteca MPI for desenvolvida para uso sobre essa rede (como planejado), um estudo interessante é a implementação do dgemm tridimensional sobre essa nova biblioteca, para avaliar o *overhead* introduzido por

ela. Ainda se pode utilizar o *benchmark* NAS [12], criado pela NASA, para avaliar os ganhos com a biblioteca MPI sobre a rede de topologia toroidal em relação a uma rede Ethernet.

Para avaliar o desempenho do conjunto unidade de ponto flutuante e rede de topologia toroidal, ainda se pode utilizar aplicações de processamento sísmico, química computacional e modelamento de clima do SPEChpg96 [13], (*Standard Performance Evaluation Corporation: High Performance Group*), *kernels* adaptados como a FFT complexa unidimensional e a decomposição LU blocada do SPLASH [14], (Stanford Parallel Applications for Shared Memory) e aplicações adaptadas também do SPLASH como simulação de oceanos e simulação de água com ou sem estruturas de dados especiais.

Referências

[1] Allen, F., et al. Blue Gene: A vision for protein science using a petaflop supercomputer. IBM Systems Journal 40, 2 (2001), 310—328.

[2] J.Lerner, E. Cellular architecture builds next generation supercomputers. em Think Research 1 (June 2002).

[3] Almasi, G., et al. Cellular supercomputing with system-on-a-chip. em ISSCC 2002 Proceedings (San Francisco, California, USA, 2002), IBM T.J.Watson Research and IBM Enterprise Server Group.

[4] Sipper, M. The emergence of cellular computing. em Computer Magazine 1} (July 1999).

[5] Herrod, S. A. em Using Complete Machine Simulation to Understand Computer System Behavior. PhD thesis, Stanford University, 1998.

[6] Mukherjee, S. S., Adve, S. V., Austin, T., Emer, J., and Magnusson, P. S. Performance simulation tools. em Computer Magazine 1 (Feb. 2002).

[7] McCalpin, J.~D.} \newblock Sustainable memory bandwidth in current high performance computers, 1995.http://home.austin.rr.com/mccalpin/papers/bandwidth.

[8] Memory bandwidth: STREAM benchmark performance results. http://www.cs.virginia.edu/stream/ref.html.

[9] Strauss, K., Avaliação de Performance de Arquiteturas para Computação de Alto Desempenho. Dissertação de Mestrado apresentada à Escola Politécnica da Universidade de São Paulo, Departamento de Computação e Sistemas Digitais. Agosto, 2002.

[10] Dongarra, J. J., Duff, I. S., Sorensen, D. C., and van der Vorst, H. A. em Solving Linear Systems on Vector and Shared Memory Computers. Society for Industrial and Applied Mathematics, 1991.

[11] Agarwal, R. C., Balle, S. M., Gustavson, F. G., Joshi, M., and Palkar, P. A three-dimensional approach to parallel matrix multiplication. *IBM* Journal of Research and Development 39, 5 (1995).

[12] NAS parallel benchmarks. http://www.nas.nasa.gov/Software/NPB}.

[13] Spec benchmarks. http://www.specbench.org.

[14] Splash-2: Stanford parallel applications for shared memory. http://www-flash.stanford.edu/apps/SPLASH/.

Aplicação dos Padrões ODP e TMN no Gerenciamento de Sistemas Corporativos Distribuídos: Sistemas de CRM

Sandro Antônio Vicente
Moacyr Martucci Jr.
{sandro.vicente, moacyr.martucci}@poli.usp.br

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

Resumo

Na medida que um sistema corporativo cresce e incorpora novas funcionalidades, cresce sua complexidade. Tais sistemas compreendem diversos componentes heterogêneos, geograficamente distribuídos, localizados em diversos domínios e seu gerenciamento apresenta-se como um novo desafio. Este artigo tem como objetivo propor a utilização de padrões abertos, definidos para a modelagem de sistemas distribuídos e gerenciamento de redes de telecomunicações, para permitir o gerenciamento de sistemas corporativos complexos. Especificamente, será ilustrada a utilização dos padrões ODP e TMN no desenvolvimento de arquiteturas de gerenciamento para sistemas de CRM.

Abstract

At the same rate that a corporate system grows in size and new capabilities, grows its complexity. Such systems consist of several heterogeneous components, geographically distributed across many domains. New challenges appeared on attempts to provide means to manage them. This paper aims to propose the use of open standards, defined in order to model distributed systems and to manage telecommunications networks, to address the problems of managing complex business systems. Specifically, it will be depicted the use of ODP and TMN standards to develop management architectures for CRM systems.

1. Introdução

Os atuais níveis de desenvolvimento de tecnologias de informação e telecomunicações permitiram a implementação de conceitos de marketing e administração. Resultante disso, sistemas extremamente complexos, distribuídos, assíncronos e heterogêneos vieram sendo compostos a partir da integração de soluções focalizando diversos propósitos.

A complexidade desses sistemas é ampliada no que se refere ao seu controle e gerenciamento, principalmente porque tais sistemas dificilmente são implementados por apenas um fornecedor e não se encontram totalmente dentro do domínio de uma instituição.

1.1 Motivação

Sistemas distribuídos, assíncronos e orientados por mensagens, demandam novos graus de organização por apresentarem as seguintes inconveniências [1]:

- Ninguém possui todo o sistema: cada vez mais componentes de uma aplicação localizam-se em

ambientes de outros fornecedores e até de clientes (PDAs, telefones móveis, etc.) e utilizam-se de infra-estruturas pertencentes a diversas organizações (concessionárias de telecomunicação, data-centers, etc). Isso dificulta a monitoração e gerenciamento de todo o sistema;

- O paradigma de comunicação entre subsistemas através de chamadas de procedimentos está mudando para o de intercâmbio de mensagens. Características inerentes a sistemas distribuídos tais como conexões intermitentes, referências dinâmicas, comunicação assíncrona e sujeita a latência, são suportadas por interações através de mensagens.

Enquanto a última característica apresentada pode ser endereçada por arquiteturas de objetos distribuídos, disponíveis através de tecnologias em uso comercial (DCOM, CORBA, DCE, Enterprise Java Beans, etc.), a primeira característica ainda necessita de pesquisas para sua implementação.

Por outro lado, padrões abertos podem ser aplicados no desenvolvimento de arquiteturas de gerenciamento que lidem com as complexidades de tais sistemas. Destacam-se os padrões TMN (Telecommunication Management Network) [2].

1.2 Objetivos

O objetivo deste trabalho é ilustrar a aplicação da arquitetura TMN no gerenciamento de sistemas corporativos complexos e distribuídos. Nesse contexto, o gerenciamento de tais sistemas deve permitir:

- o sincronismo de informações entre todos os componentes do sistema;
- a coleta de dados sobre componentes do sistema e suas interações. Por exemplo, taxas de QoS;
- a detecção de falhas e reconfiguração dinâmica através da habilitação e desabilitação de componentes do sistema.

Dentre os sistemas corporativos de grande complexidade que justificam a aplicação da arquitetura TMN para seu gerenciamento, as soluções de CRM (Customer Relationship Management) compreendem um interessante objeto de estudo sobre o qual o trabalho será realizado.

1.3 Estrutura do trabalho

Este trabalho está organizado em seis sessões. Na seção 2 são apresentados os conceitos básicos utilizados, na seção 3 é apresentada uma arquitetura genérica de sistemas de CRM, modelada nos padrões ODP. Na seção 4, os objetos obtidos na seção 3 são incorporados na arquitetura de gerenciamento TMN. Na seção 5 é apresentado um estudo de caso de gerenciamento de sistema de CRM através da arquitetura TMN e, na seção 6, são apresentadas as conclusões sobre o trabalho.

2. Conceitos básicos

2.1 Customer Relationship Management - CRM

O conceito de CRM origina-se do marketing e tem como objetivo a conquista e a fidelização de clientes através do acompanhamento personalizado de cada cliente (relacionamento um a um) [3]. A tecnologia permite a interação com o cliente através de pontos de contato automatizados ou semi-automatizados e também permite o acesso e a distribuição de informações para os clientes [4].

Distinguem-se três grandes grupos funcionais necessários a uma arquitetura de CRM: o operacional, o analítico e o colaborativo. O CRM operacional compreende os sistemas de front-office (sistemas de gestão de relacionamento com o cliente), de back-office (sistemas de gestão de negócios e sistemas legados) e de mobile-office (sistemas móveis usados no atendimento e vandas em campo). O CRM analítico compreende as tecnologias de análise e apoio a decisão (tais como: data warehouses, datamarts, datamining, gerenciadores de categorias e de campanhas). O CRM colaborativo compreende os canais com os clientes: URAs, CTI, DACs, web, face-a-face, etc. [5][3].

2.2 Open Distributed Processing - ODP

A recomendação ODP [6] tem como objetivo a especificação de sistemas distribuídos baseados em padrões abertos e cobre todas as características inerentes a tais sistemas [7]. Dentre os conceitos estabelecidos pelos padrões ODP, destaca-se o conceito de ponto de vista, que permite a representação do sistema segundo diferentes perspectivas. São identificados cinco pontos de vista: empresa, informação, computação, engenharia e tecnologia [6].

O ponto de vista de empresa preocupa-se com os objetivos e com o ambiente onde o sistema especificado opera. O ponto de vista da informação tem seu foco nos requisitos de informações do sistema e lida com os elementos de informação, seus estados e as mudanças de estado permitidas. O ponto de vista de computação preocupa-se com as funções de processamento do sistema e suas interfaces, abstraindo o ambiente distribuído de execução através de serviços de transparência. O ponto de vista de engenharia estabelece os serviços que permitem transparência de distribuição do sistema. O ponto de vista de tecnologia preocupa-se com a implementação de todo o sistema através de componentes de software e hardware específicos [6].

2.3 Telecommunications Management Network - TMN

TMN define uma rede de gerenciamento independente que interage com uma rede de telecomunicações permitindo seu gerenciamento, sendo que as especificações TMN definem diversas arquiteturas de gerenciamento, cada uma focalizando um diferente aspecto do gerenciamento, sendo elas as arquiteturas funcional, física e de informações, além. As especificações também propõem uma arquitetura de camadas lógicas [2].

A arquitetura funcional descreve os seguintes grupos funcionais responsáveis pelo gerenciamento de uma TMN: NEF (Network Element Function), que permite o gerenciamento dos elementos da rede; QAF (Q Adapter Function), que permite adaptação de componentes não-TMN a uma TMN; MF (Mediation Function), que permite o processamento e uniformização das informações dos elementos de rede; OSF (Object System Function), que executa o gerenciamento propriamente dito; e WSF (Work Station Function), que permite o suporte à interface homem-máquina. Também são especificados pontos de referência que distinguem os grupos funcionais entre si: m entre elementos não-TMN e QAFs, q entre NEFs, MFs e OSFs, w entre OSFs e WSFs e x entre OSFs de redes TMN distintas.

A arquitetura física implementa os grupos funcionais, definidos na arquitetura funcional, em blocos físicos: Elementos de rede (NE - Network Element), dispositivos de mediação (MD - Mediation Device), adaptadores Q (QA - Q Adaptor), sistemas de operações (OS - Operations System) e estações de trabalho (WS - Work Station). A relação entre os grupos funcionais e os blocos físicos é descrita na tabela 2.1. Na arquitetura física, também são definidas interfaces, que implementam os pontos de referência da arquitetura funcional: Q entre NEs, MDs e OSs, W entre OSFs e WSFs e X entre OSs de redes TMN distintas.

	NEF	MF	QAF	OSF	WSF
NE	M	O	O	O	**O***
MD		M	O	O	O
QA			M		
OS			O	M	O
WS					M

M - Mandatório
O - Opcional
O* - Opcional e presente somente se OSF e MF estiverem presentes

Tabela 2.1 - Relação entre grupos funcionais e blocos físicos

A arquitetura de informações permite a modelagem de componentes da rede TMN na forma de Objetos de Gerenciamento (MO - Managed Object) [2][8].

A arquitetura de camadas lógicas é proposta pela TMN [2] para organizar a estrutura de gerenciamento, executada através dos componentes funcionais OSF, em camadas lógicas em diferentes níveis de abstração, sendo essas as camadas de gerenciamento de elementos, de rede, de serviços e de negócios. A camada de gerenciamento de elementos trata dos elementos da rede individualmente. A camada de gerenciamento de rede trata a configuração de componentes da rede e suas interações. A camada de gerenciamento de serviços lida com os serviços desempenhados pela rede, que são acessados pelos usuários. A camada de gerenciamento de negócios permite consolidar informações sobre toda a rede e serviços, possibilitando a análise e determinação de estratégias para administrar o negócio de telecomunicações.

3. Arquitetura de CRM no modelo ODP

Nesta seção será ilustrada uma modelagem de uma arquitetura genérica de CRM desenvolvida a partir dos pontos de vista ODP de empresa, computação e informação. Supõe-se a existência de uma arquitetura de

objetos distribuídos que provenha as transparências desempenhadas pelo ponto de vista de engenharia. Possíveis tecnologias para a implementação dessa arquitetura estão fora do escopo deste trabalho.

A arquitetura de CRM ilustrada encaixa-se no esquema de um CRM Ecosystem, conforme proposto em [5], aderente à maioria das soluções de CRM em uso comercial. A modelagem dá ênfase aos componentes envolvidos na interação com o cliente que, além de essenciais para o relacionamento um a um, são as maiores fontes de complexidade dos sistemas de CRM.

3.1 Modelo de empresa

Nesta modelagem, o sistema de CRM é apresentado como uma composição de objetos de negócio, sendo que esses objetos são responsáveis pelos serviços do sistema de CRM (figura 3.1a). São definidas as seguintes classes de negócios: Customer, Channel, FrontOffice, BackOffice e Analytical:

- Customer - representação do cliente e suas preferências, localização, situação corrente, etc. Relaciona-se com o objeto Channel;
- Channel - serviço de canal de contato com Customer. Permite interação de instâncias do Customer com instâncias do FrontOffice e pode ser especializado por subclasses representando canais ativos e receptivos [3], que por sua vez podem ser especializados em grupos de dispositivos de contato específicos (atendentes, URAs, sessões web, etc.);
- FrontOffice - permite encapsular serviços para o gerenciamento de contato com o Customer, realizado através da classe Channel e recebe diretivas da classe BackOffice;
- BackOffice - encapsula sistemas de gestão de negócios, sistemas legados e repositórios de dados operacionais e recebe diretivas da classe Analytical;
- Analytical - encapsula os serviços necessários para a aplicação do CRM analítico e os repositórios de dados analíticos. Extrai informações do BackOffice, analisa essas informações, define estratégias e repassa-as ao BackOffice.

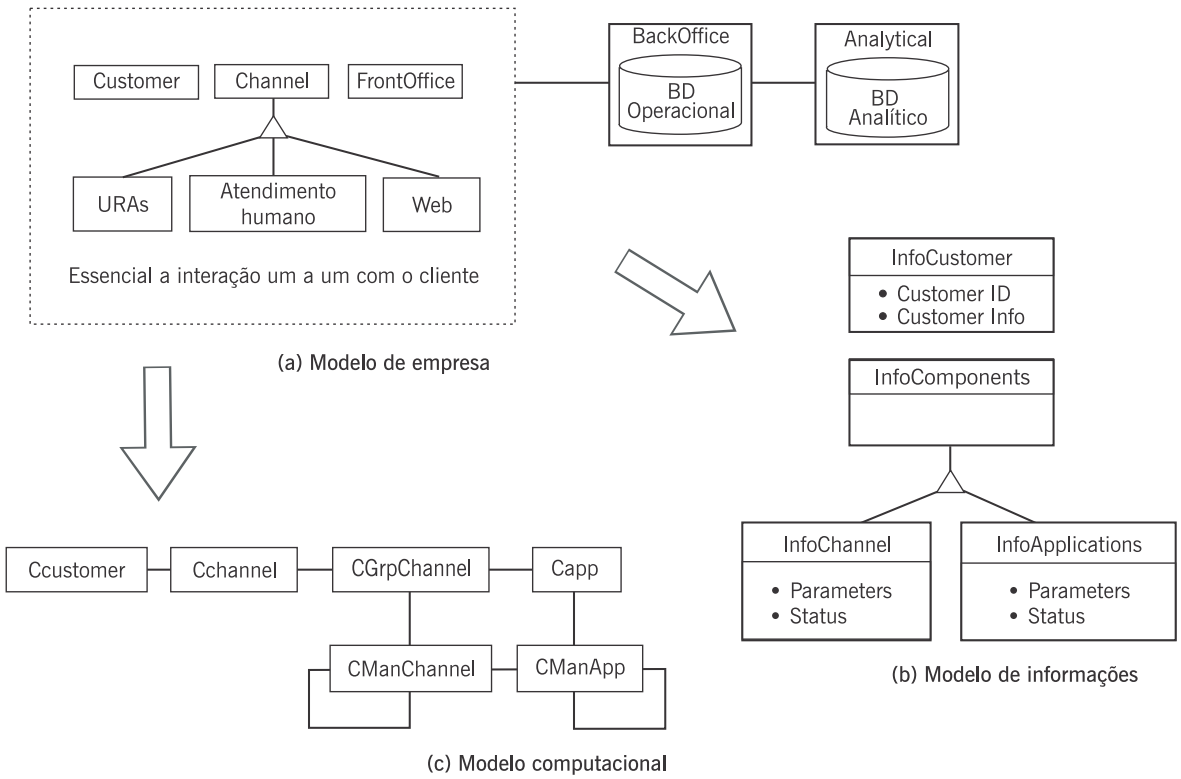


Fig. 3.1. Arquitetura de sistemas de CRM nos pontos de vista ODP

As classes Customer, Channel e FrontOffice são essenciais para a interação um a um, e são as maiores fontes de complexidade dos sistemas de CRM. A interação entre instâncias dessas classes envolve componentes geograficamente distribuídos de diferentes fornecedores. Os modelos de computação e informação contemplados a seguir são obtidos a partir do desenvolvimento dessas classes.

3.2 Modelo de informação

As informações sobre os componentes do sistema e as representações dos clientes são modeladas na forma de objetos de informação. Para o gerenciamento de sistemas de CRM, são destacadas as classes de informações InfoCustomer, InfoComponents, InfoChannel e InfoApplication (figura 3.1b):

- InfoCustomer - representa o cliente e suas informações características. Agrega o atributo CustomerID, que identifica unicamente o cliente, e o atributo CustomerInfo, que compreende as informações que caracterizam o cliente (estado atual, preferências, pendências, localização, etc.);
- InfoComponents - representa os demais componentes do sistema de CRM necessários a interação um a um. São especializados pelas subclasses InfoChannel e InfoApplication;
- InfoChannel - representa as informações sobre o canal de comunicação, agregando os atributos InfoChannel.Parameters (parâmetros que caracterizam o canal) e InfoChannel.Status (estado do canal);
- InfoApplication - representa as informações sobre a aplicação que interage com os canais de comunicação. Agrega os atributos InfoApplication.Parameters (parâmetros que caracterizam a aplicação) e InfoApplication.Status (estado da aplicação).

O detalhamento das informações modeladas neste ponto de vista são dependentes do negócio ao qual o sistema é aplicado, sendo que as classes de informações descritas envolvem características comuns a quaisquer sistemas de CRM.

3.3 Modelo de computação

Nesta modelagem, são identificadas as classes de computação que devem prover as funcionalidades necessárias para permitir a interação um a um. São destacados os classes de computação CChannel, CGrpChannel, CManChannel, CApp e CManApp (figura 3.1c):

- CCustomer - representação do cliente;
- CChannel - canal de interação com CCustomer;
- CGrpChannel - agrupamento de instâncias de CChannel, permitindo a modelagem de serviços que envolvem diversos canais com o cliente. Por exemplo, uma central de atendimento, que compreende diversas linhas telefônicas, ou um web server farm, que compreende diversas conexões web;
- CManChannel - gerenciador de canais. Relaciona-se com o CGrpChannel, efetuando o controle sobre um grupo de canais, e com o CManApp, para sincronizar informações entre aplicações e canais (instâncias de CApp e CChannel);
- CApp - aplicação de front-office que interage com os canais, provendo os serviços necessários ao cliente;
- CManApp - gerenciador de aplicações, efetua a requisição, restrição, ativação e desativação de instâncias da classe CApp.

As correspondências entre as classes de computação e negócios ocorrem da seguinte forma: a classe CCustomer localiza-se na classe Customer; as classes CChannel, CGrpChannel e CManChannel localizam-se em Channel, e as classes CApp e CManApp localizam-se em FrontOffice.

4. TMN no gerenciamento de sistemas de CRM

4.1 Arquitetura funcional

Os grupos funcionais TMN podem ser adaptados para o gerenciamento de sistemas de CRM, apresentando relação direta com o modelo computacional definido na seção 3.3. O termo componente designa as classes de computação CChannel, CGrpChannel, CApp e suas informações, modeladas pelas classes de informações InfoCustomer, InfoComponents, InfoChannel e InfoApplication. A seguir são identificadas as aplicações dos blocos funcionais TMN nos sistemas de CRM.

O NEF compreende funções para o intercâmbio de dados com e o gerenciamento dos elementos do sistema de CRM. Aplicações que incorporem essa funcionalidade gerenciam diretamente recursos como linhas de URAs, sessões de aplicações web e conexões com mainframes. Os componentes das classes CChannel, CGrpChannel e CApp têm acesso direto a essas funcionalidades, que permitem o gerenciamento detalhado dos recursos.

O QAF compreende as funções que permitem a adaptação de dispositivos e sistemas com interfaces não especificadas para os padrões TMN numa rede TMN. Por exemplo, permite que uma rede TMN gerencie um servidor web com interface SNMP ou um DAC com interface CTI CSTA.

O MF compreende as funções que executam a adaptação, o armazenamento, a filtragem e a concentração dos fluxos de informações oriundos dos blocos funcionais NEF e QAF, possibilitando uma visão de gerenciamento uniforme, integrada e independente de tecnologia dos componentes do sistema. Um exemplo é um terminal de atendimento e seu ramal telefônico (Posição de Atendimento - PDA) , recursos fisicamente distintos que devem estar sempre sincronizados. A mediação permite esta sincronização como se terminal e o ramal fossem um único componente.

O WSF permite a monitoração e o controle da rede TMN através de interfaces homem-máquina. Pessoas responsáveis pela operação dos componentes de um sistema de CRM pode usar essas funcionalidades para a monitoração e controle de centrais de atendimento, web server farms, sistemas de front-office e outros subsistemas.

O OSF desempenha o processamento da informação de gerenciamento, permitindo a monitoração, o controle e a coordenação da rede TMN. Essa funcionalidade é incorporada pelos componentes e pelos objetos das classes CManApp e CManChannel.

Os blocos funcionais OSF podem ser especializados de acordo com a camadas lógicas da arquitetura TMN em que se encontram, possibilitando o gerenciamento do sistema de CRM similarmente a uma rede de telecomunicações.

4.2 Arquitetura de informações

As classes informações correspondentes ao sistema de CRM são modeladas na forma de MOs e transitam de forma padronizada na rede. Assim, os MOs correspondem às classes de informação InfoCustomer, InfoComponents, InfoChannel e InfoApplication.

Os papéis de agente e gerente referem-se às interações entre os processos que gerenciam os MOs. Na arquitetura de gerenciamento proposta, os agentes sempre encontram-se em nível de abstração inferior (camada lógica inferior) ou igual ao do gerente. Admite-se que objetos de gerenciamento possam interagir entre si no mesmo nível de abstração. Nesse caso, a relação gerente-agente ocorre de acordo com o contexto de gerenciamento, sendo que ambos os papéis podem ser assumidos simultaneamente para o sincronismo de informações.

Por exemplo, componentes de uma central de atendimento (nível lógico de elementos) podem ser gerenciados por um sistema de CTI responsável por coordenar toda a central (nível lógico de rede), portanto esse servidor assume o papel de gerente sobre as URAs, atendentes e o DAC da central de atendimento. Mas duas centrais de atendimento, coordenadas por dois sistemas de CTI distintos (nivel lógico de rede), também podem interagir entre si para atender um cliente, de forma que o papel de gerente caberia à central que iniciar a interação com a outra central.

4.3 Arquitetura física

Os blocos funcionais são implementados em configurações em blocos físicos segundo a tabela 2.1. Componentes de gerenciamento do sistema de CRM são posicionados na arquitetura física de acordo com os grupos funcionais implementados. No caso de estudo da seção 5, na figura 5.3 são apresentados os elementos funcionais responsáveis pelo gerenciamento de uma central de atendimento e na figura 5.4, observa-se a implementação destes elementos funcionais em blocos físicos. Nesse caso, o servidor CTI é um bloco físico NE que comporta funcionalidades de gerenciamento pertencentes a todos grupos funcionais (NEF, OSF, MF, QAF e WSF).

4.4 Arquitetura de camadas lógicas *(LLA - Logical Layered Architecture)*

A dificuldade de se gerenciar sistemas complexos como os de CRM pode ser atenuada através da divisão das funcionalidades de gerenciamento em camadas lógicas,

com diferentes níveis de abstração. O modelo de referência de camadas lógicas proposto pela especificação TMN [2] pode ser utilizado no gerenciamento de sistemas de CRM. Este modelo propõe quatro camadas lógicas de gerenciamento: camada lógica de gerenciamento de elemento, camada lógica de gerenciamento de rede, camada lógica de gerenciamento de serviços e camada lógica de gerenciamento de negócios. Em cada uma dessas camadas, os grupos funcionais OSF exercem o gerenciamento num nível de abstração correspondente ao de camada lógica.

4.4.1 **Camada de gerenciamento de elementos**

Esta camada compreende, além dos OSF, os grupos funcionais NEF, QAF e MF [2]. Essa camada concentra as funcionalidades necessárias para gerenciar os componentes (seção 4.1) que compõem os sistemas de CRM. O grau de abstração é baixo, permitindo a visualização e controle detalhados dos componentes gerenciados. Por exemplo, é possível a visualização e o controle dos estados de um terminal de atendimento, a monitoração do tempo de utilização e os estados de um ramal telefônico, etc.

Aplicações de gerenciamento na camada de gerenciamento de elementos sempre assumem papel de agentes.

4.4.2 **Camada de gerenciamento de rede**

Nesta camada é realizado o controle e a monitoração de configurações de componentes do sistema de CRM e suas interconexões, sendo que cada uma dessas configurações representa uma rede de componentes.

O sistema de CRM pode ser visualizado como uma ou várias redes de componentes integrados. As funções de gerenciamento desta camada controlam as interações entre os componentes, permitindo que os canais de contato com o cliente estejam sincronizados com as aplicações adequadas para atendê-lo. Também executam ativação e desativação de componentes, detecção de falhas, supervisão de níveis de carga e performance nas interações entre os componentes.

As aplicações de gerenciamento da camada de gerenciamento rede realizam serviços para a camada de gerenciamento de serviços e assumem os papéis de agente (com relação a camada de gerenciamento de serviços) e de gerente (com relação a camada de gerenciamento de elementos). Também ocorrem interações de gerenciamento dentro desta camada com finalidade de permitir a sincronização de informações.

4.4.3 **Camada de gerenciamento de serviços**

Esta camada é responsável pelo gerenciamento dos serviços desempenhados pelo sistema de CRM. Cada serviço é desenvolvido através da colaboração entre elementos da camada de gerenciamento de rede e, no caso dos sistemas de CRM, os serviços são instâncias das classes de negócios vistas na seção 3.1 (classes Channel e FrontOffice).

Dessa forma, os serviços correspondentes a classe de negócio Channel são realizados por redes de objetos de computação das classes CChannel, CGrpChannel e CManChannel, e os serviços correspondentes a classe de negócio FrontOffice são realizados por redes de objetos de computação das classes CApp e CManApp.

Em sistemas de CRM, os serviços correspondem às centrais de atendimento, aos sistemas de back-office e aos web server farms. Para cada serviço, são gerenciadas as políticas de atendimento de clientes, o sincronismo de informações do cliente entre os componentes, o controle de carga entre os serviços (por exemplo, o controle de cargas entre centrais de atendimento), o controle de qualidade do serviço e a requisição da execução de algum plano de contingência diante da constatação de alguma anomalia.

4.4.4 **Camada de gerenciamento de negócios**

A camada de gerenciamento de negócios é a camada de maior nível de abstração. As funcionalidades dessa camada são predominantemente analíticas, envolvendo a concentração de informações de gerenciamento de todo o sistema de CRM e a aplicação de tecnologias de análise de dados (tais como: data warehouses, OLAP, data mining, etc.) sobre essas informações.

Estas funcionalidades permitem a visualização global do sistema de CRM, a definição de novas metas e a especificação de novos serviços. Com isto, pode-se orientar o desenvolvimento do CRM de acordo com as perspectivas de negócio da organização.

As aplicações de gerenciamento neste nível sempre assumem o papel de gerente.

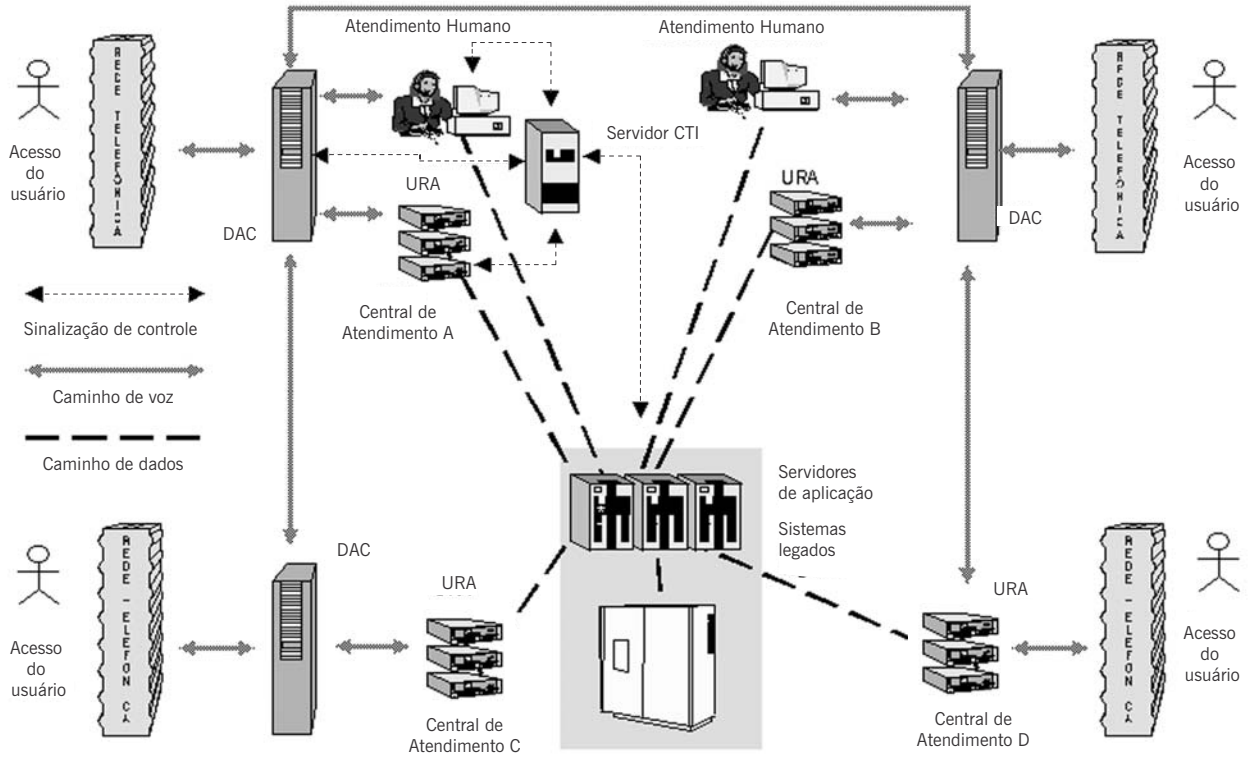


Fig. 5.1. Caso de estudo: central de atendimento distribuída

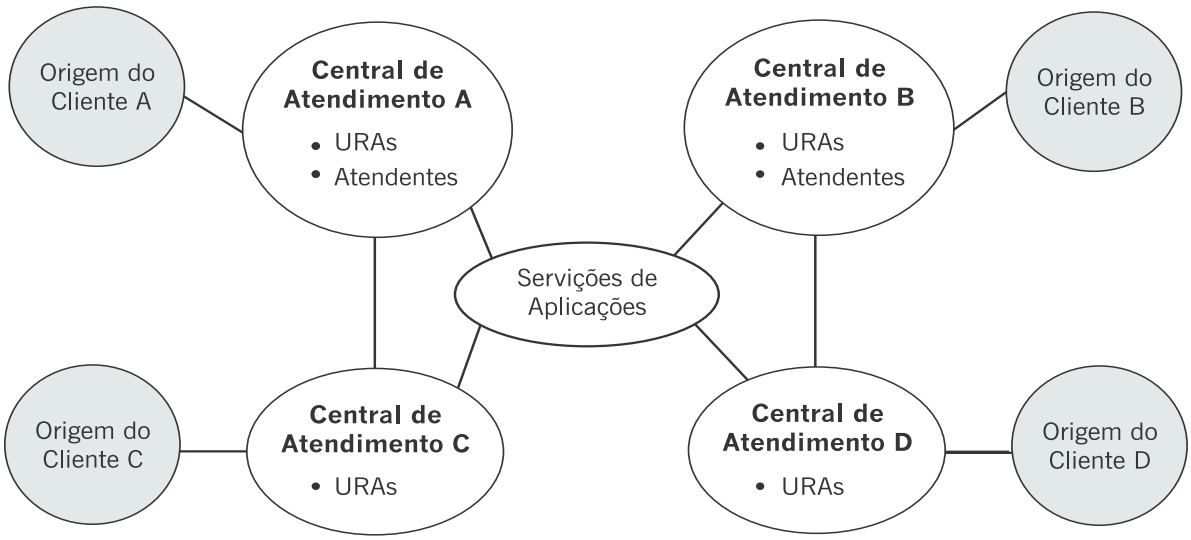


Fig. 5.2. Arquitetura de sistemas de CRM nos pontos de vista ODP

5. Estudo de caso

Nesta seção, será examinada a aplicação do gerenciamento TMN sobre uma solução de CRM que envolve a integração de diversas centrais de atendimento geograficamente distribuídas, integradas a um sistema de informações front-office que trata as consultas e operações de clientes, realizadas através das centrais de atendimento. Este sistema é ilustrado na figura 5.1.

Cada central está localizada em uma região geográfica e atende apenas a ligações locais. As centrais A e B possuem atendimento eletrônico (URAs) e atendimento humano (atendentes). As centrais C e D possuem apenas atendimento eletrônico. Quando uma ligação se origina em C ou D e necessita atendimento humano, ela é direcionada para uma central adjacente que provê o atendimento humano. As centrais A e B também podem compartilhar entre si posições de atendimento humano, de forma a balancear a carga de utilização de atendentes entre as duas centrais.

O sistema em estudo é compatível com a arquitetura genérica descrita na seção 3 e seu gerenciamento será analisado de acordo com os pontos de vistas utilizados na sua modelagem.

5.1 Modelo de empresa

No ponto de vista de empresa, esta arquitetura compreende as classes que modelam os clientes (Customer), o serviço de canal de contato (Channel) e a aplicação front-office (FrontOffice). No caso de estudo, as instâncias de Channel representam as centrais de atendimento e suas características; as instâncias de FrontOffice representam aplicações de gestão de interação com o cliente (sistemas de atendimento) e as instâncias de Customer representam os clientes, sendo esses associados ao objeto da classe Channel no momento do contato com sistema de CRM. Na figura 5.2, as centrais de atendimento, a aplicação e os clientes são exemplos de objetos de negócios cuja gerência é realizada pelo grupo funcional OSF, compreendido no nível de gerenciamento de serviços da LLA.

No nível de gerenciamento de negócios da LLA, todo o sistema é representado por um objeto de empresa, que

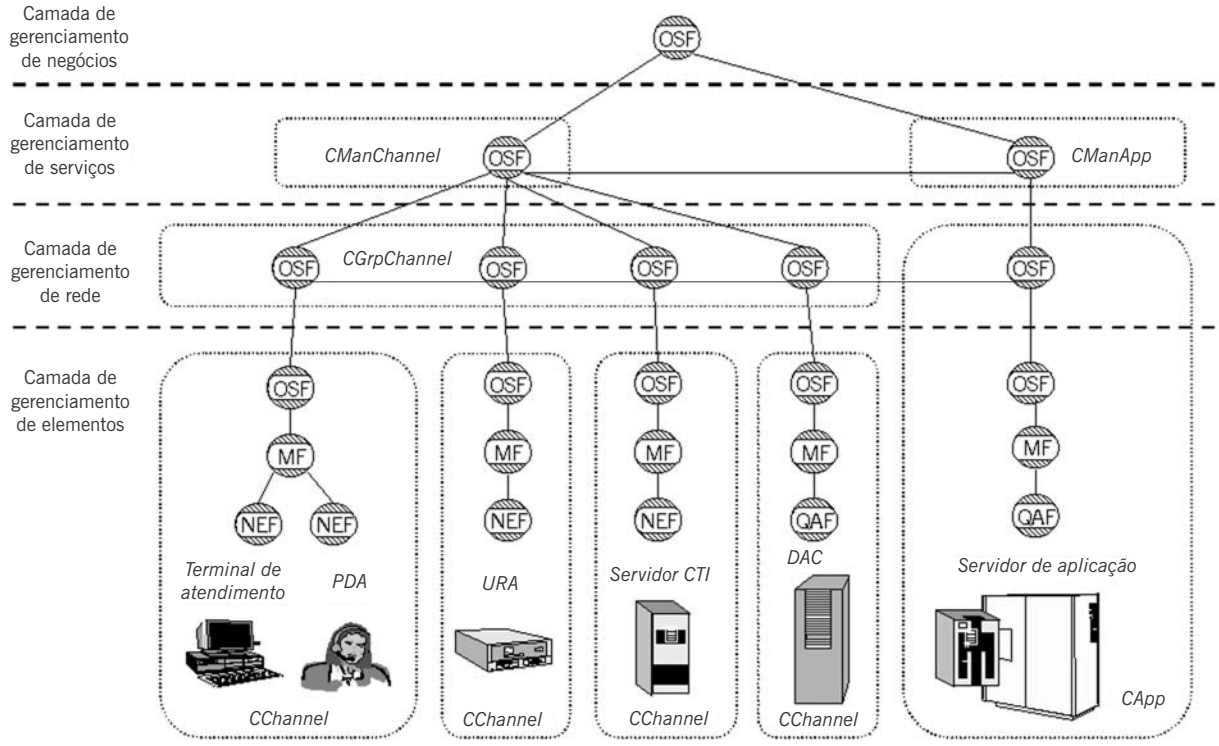


Fig. 5.3. Central A: Objetos de computação, grupos funcionais e LLA

concentra e analisa as informações de gerenciamento sobre todos os elementos do sistema de CRM e estabelece estratégias e diretivas para aperfeiçoar os serviços de atendimento. Tais estratégias podem compreender a contratação de recursos, a diminuição do tempo médio de atendimento e o remanejamento de serviços disponíveis apenas no atendimento humano para o atendimento eletrônico (URAs).

5.2 Modelo de computação

Na figura 5.3, é apresentado o modelo de computação correspondente a central de atendimento A, considerando a interação da central de atendimento com o sistema de front-office. Os grupos funcionais NEF e QAF possibilitam o gerenciamento de dispositivos físicos (tais como: URAs, estações de trabalho de atendentes, terminais de atendimento, e o DAC). Agregações dos grupos NEF e QAF com os grupos MF e OSF correspondem aos objetos de computação das classes CApp e CChannel, localizados no nível de gerenciamento de elementos da LLA.

No nível de gerenciamento de redes da LLA, os grupos OSF correspondem aos objetos da classe CGrpChannel, que gerenciam os diversos componentes da Central de Atendimento. O grupo funcional OSF correspondente a aplicação CApp encontra-se nesse nível e interage com os objetos da classe CGrpChannel, permitindo o sincronismo entre os componentes. No nível de gerenciamento de serviços da LLA, os OSFs correspondem aos objetos CManChannel e CManApp, que coordenam os serviços da central de atendimento (Channel) e da aplicação (FrontOffice). Estes serviços interagem entre si permitindo a interação um a um com o cliente em qualquer ponto de atendimento.

A figura 5.4 ilustra a implementação dos objetos de computação da central de atendimento A em blocos físicos da arquitetura TMN. Cada objeto de computação está associado a pelo menos um OSF, incorporando uma ou mais funcionalidades de gerenciamento. No caso, o servidor CTI compreende uma instância da classe CGrpChannel e instâncias da classe CChannel correspondentes aos dispositivos de contato (PDA, terminais, URAs e DAC).

5.3 Modelo de informação

As informações referentes a cada componente do sistema de CRM, localizados em cada nível lógico da LLA, são modeladas na forma de MOs

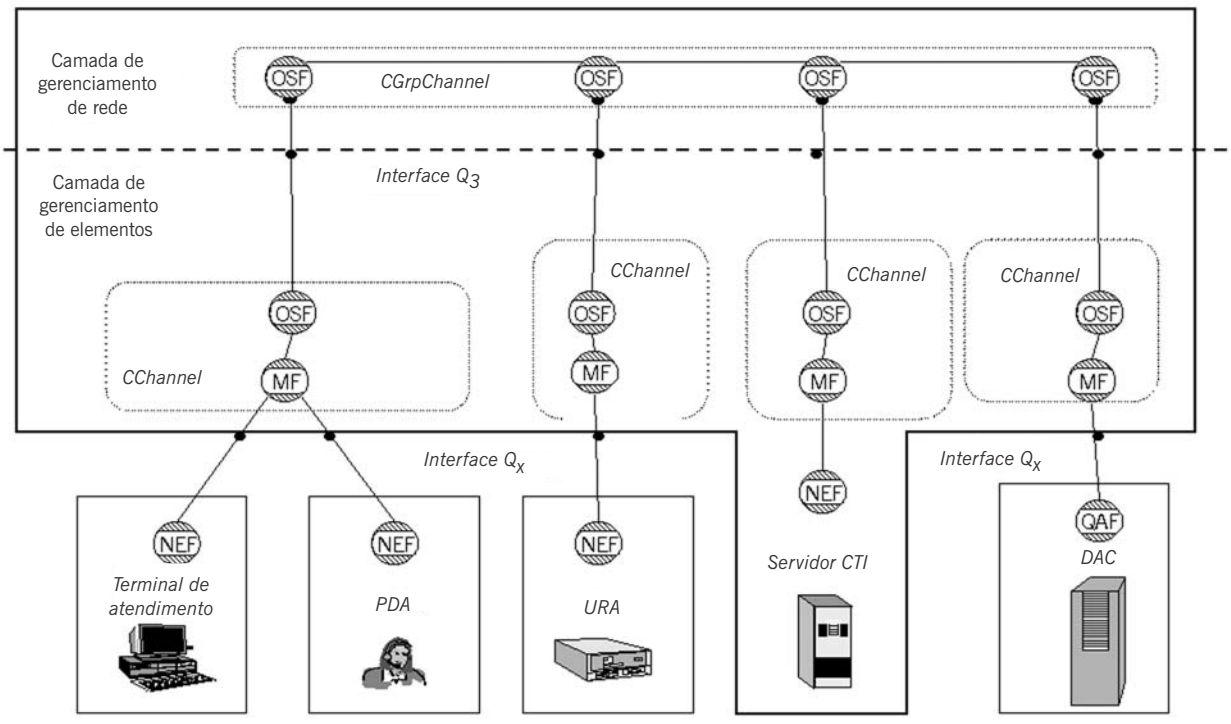


Fig. 5.4. Grupos funcionais compreendidos nos blocos físicos

No caso dos objetos de da classe de negócios Channel, as informações modeladas pelos MOs dizem respeito às capacidades da central de atendimento (número de atendentes, número de linhas de URAs, tempo de atendimento, centrais de transbordo, etc.). No caso de um objeto de computação da classe CChannel representando uma atendente, as informações modeladas pelos MOs podem compreender o nível do atendente (skill) e estado atual do ramal telefônico do atendente (busy, idle ou unavailable).

As funções de mediação (MF) podem ser utilizadas para agregar informações sobre os componentes localizados no nível lógico de gerenciamento de elementos, permitindo que estes sejam visualizados como entidades únicas nos níveis lógicos de gerenciamento de elementos ou de rede. Um exemplo é o par composto por terminal de atendente e PDA: o acesso e as operações sobre os estados dos dois dispositivos pode ser uniformizada através das funções de mediação como se o terminal e o PDA fossem uma entidade única e coesa.

6. Conclusões

Os sistemas de CRM são sistemas corporativos distribuídos de grande complexidade. Com a modelagem ODP, foi possível desenvolver uma arquitetura genérica para tais sistemas, compatível com a maioria das soluções de CRM em uso. Em seguida, verificou-se que tal arquitetura pode ser gerenciada de forma similar a um sistema de telecomunicações, e que, portanto, os padrões TMN podem ser utilizados para o gerenciamento dessa arquitetura.

Assim, verifica-se que os padrões TMN não são restritos ao gerenciamento de redes de telecomunicações. Esses padrões podem ser perfeitamente aplicados no gerenciamento de outros sistemas complexos e distribuídos, com a vantagem que o TMN baseia-se em padrões internacionais, abertos e extensíveis, o que facilita a manutenção e agregação de novas características ao sistema de CRM com baixos riscos de implementação

Referências

[1] JONES, N. Control the risks of living with untestable and unmanageable architectures. Gartner Research, 2000.

[2] ITU-T Recommendation M.3010: Principles for a telecommunication management network. 1996.

[3] GONÇALVES, A. P. C. Proposta de arquitetura aberta de central de atendimento. Dissertação (Mestrado) - EPUSP, 2001.

[4] WELLS, J. D.; FUERST, W. L.; CHOOBINEH, J. Managing information technology for one-to-one Customer interaction. Information & Management, 1998.

[5] SHAHNAM, E. The Customer Relationship Management Ecosystem. Meta Group Web edition Delta, 2000.

[6] ISO Recommendation X.901/ISO/IEC 10746-1: Information technology - Open Distributed Processing - Reference Model: Overview. 1998.

[7] ZNATY, S.; GENILLOUD, G.; GASPOZ, J. P.; HUBAUX, J. P. Networked Systems : Introducing Network Management Models into ODP. Proceedings of the IEEE Globecom'95, 1995.

[8] ITU-T Recommendation X.722. Structure of management information: Guidelines for the definition of managed objects. 1992. .

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

Luís Henrique de Barros Ceze
Wilson V. Ruggiero
{lceze, wilson}@larc.usp.br

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

Resumo

*Este artigo apresenta o **BGLsim**, um simulador paralelo de sistema completo para o Blue Gene/L[1] (Esta máquina é parte da amília Blue Gene, atualmente em desenvolvimento pela IBM Reasearch). 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] (A performance de simulação atingida pelo **BGLsim** é da ordem de 2 milhões de instruções por segundo) 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]. (This machine s part of the Blue Gene family, currently being developed by IBM Research). **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] (The simulation performance achieved by **BGLsim** is 2 million instructions/second on Pentium III 1GHz host, characterizing a 500 times slowdown) 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] *Esta máquina é parte da família Blue Gene, atualmente em desenvolvimento pela IBM Research*

[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 1 GHz host, characterizing a 500 times slowdown*

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 [1].

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 [2].

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 real.

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] (compiladores, sistemas operacionais, bibliotecas de tempo de execução, ambiente de depuração, etc.) 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.

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

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 [3] 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 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, 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 [4], 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] ordem em que os bytes de uma palavra são armazenados na memória 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 [5] 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 o [1], 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 [6]. 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 [7]. 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 [8], 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 [1], apesar do primeiro ser um produto comercial e o segundo ser um projeto

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

acadêmico. Um ponto fraco é a difícil extensibilidade desse simulador e o fato de o código fonte não ser publicamente disponível.

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 [6] 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 [9].

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 [10].

TREE **-** uma rede de interconexão em árvore que engloba todos os nós, incluindo nós de I/O e de computação [11].

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 [12].

O Blue Gene/L também disporá de uma quarta rede de interconexão fora do controle da aplicação. Essa interface chama-se JTAG[7] [13] 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:

- 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)

---

[7] *padrão IEE 1149.1 para portas de testes em circuitos integrados*

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

[9] *Symetric Multi-Processor*

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* [14].

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 [15]. 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.

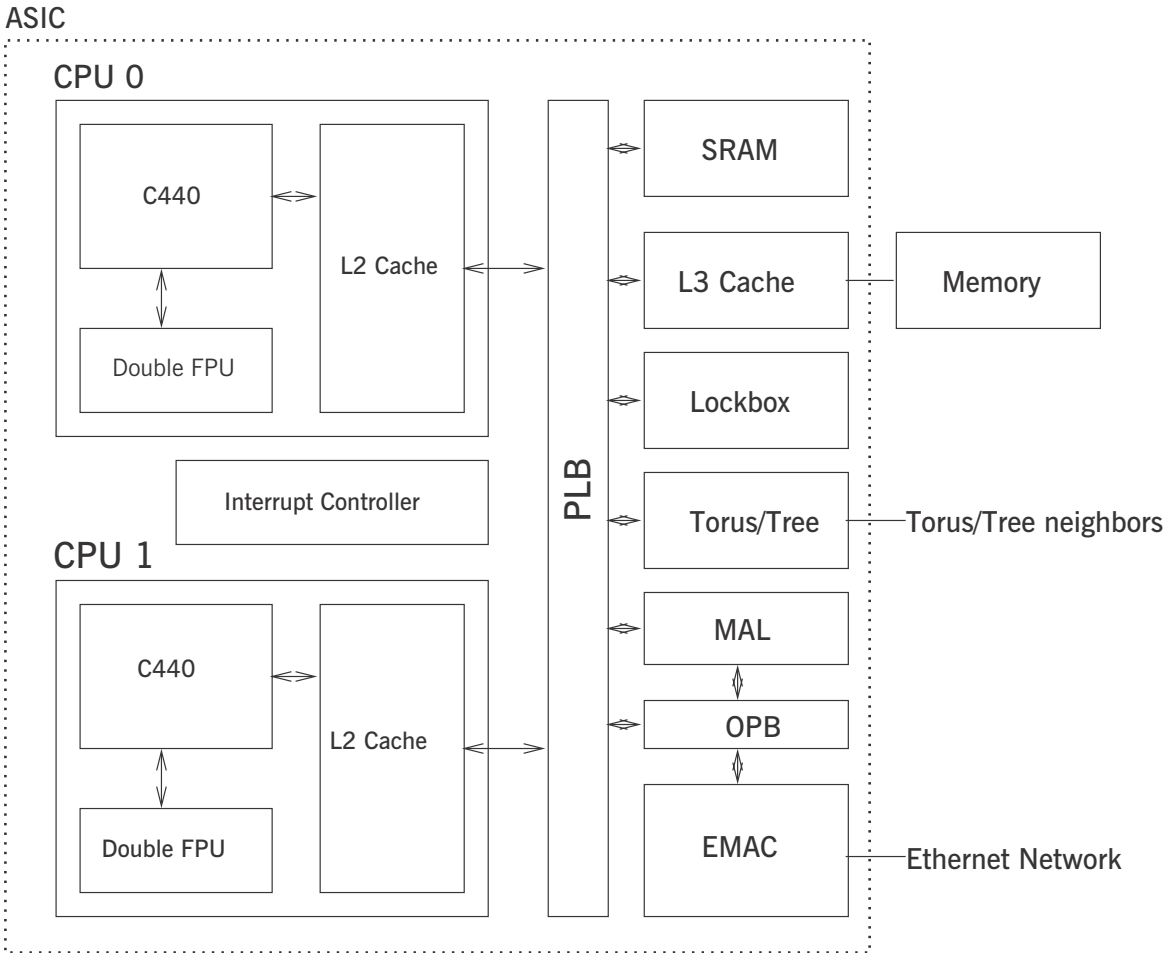


Figura 1: Diagrama simplificado dos componentes de um nó

[10] *reduced instruction set computer*

[11] *attached processing unit*

[12] *single instruction multiple operatioins multiple data*

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] (initial program loading) e monitoramento não intrusivo.

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 *ETH chips* 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 *ETH chip* 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. 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 BG\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 um simun]lador VDHL. O *host* utilizado para os experimentos foi um Pentium III de 1GHz. Para o primeiro item, "*boot* do

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.

[13] *initial program loading*

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

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 foram levados em conta.

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.

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
- 3 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

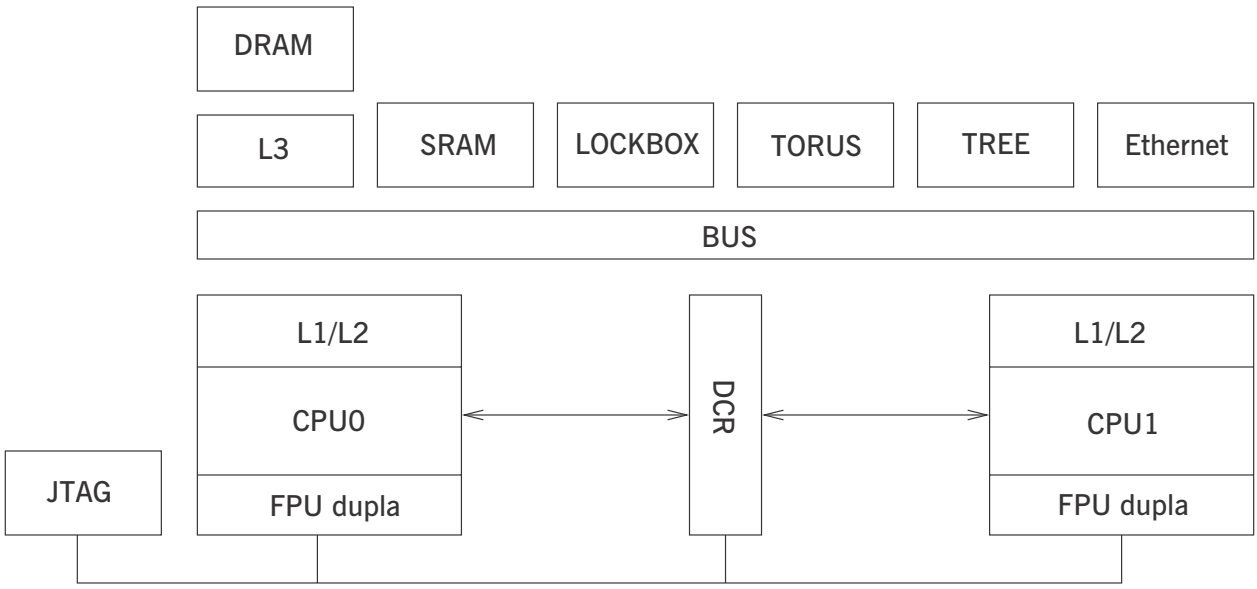


Figura 2: Diagrama em alto nível do modelo de um nó

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

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.

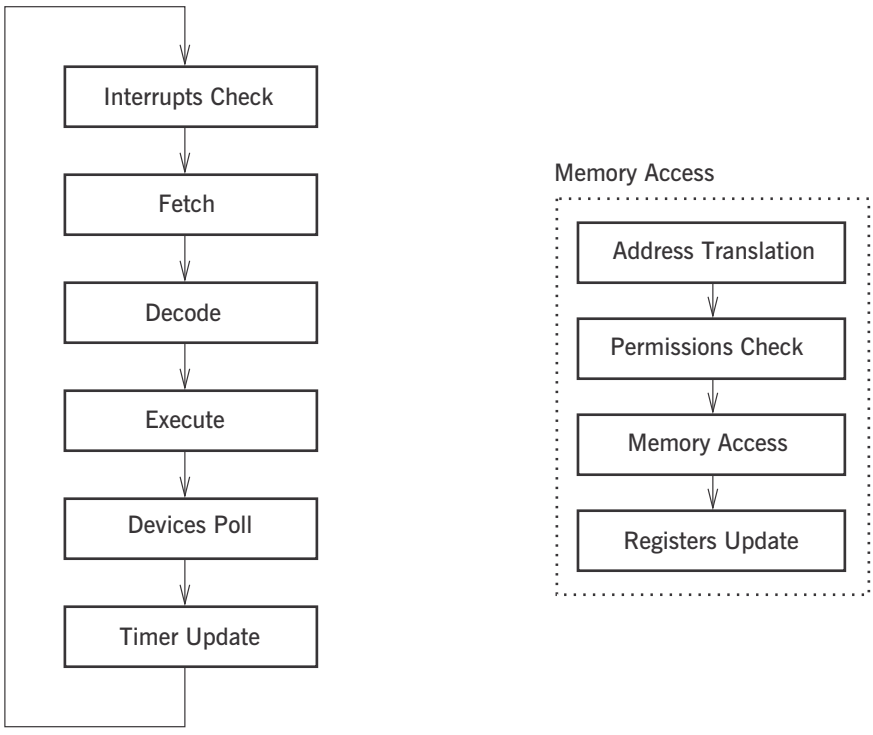


Figura 3: Ciclo de simulação da CPU

A unidade de ponto flutuante 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.

Cada um dos elementos mencionados acima é um processo no simulador. Esses processos se comunicam através de uma

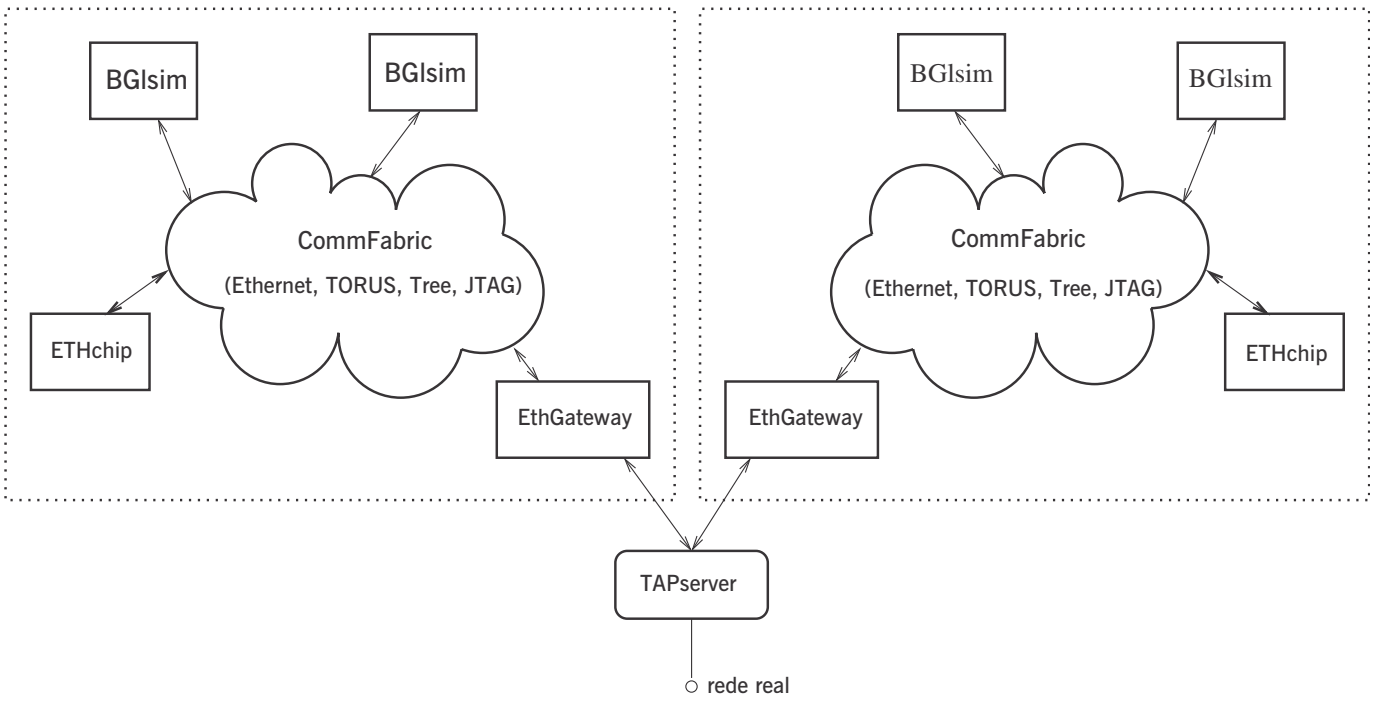


Figura 4: Visão geral de simulações de múltiplos nós

abstração de infra-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] [17] 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 send_Ethernet (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.

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 *dead-locks*, 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 [18] 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 base 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 de que a simulação disponibilizada é

[15] *message passing interface*

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

[17] *Local Area Multicomputer [16], implementação open-source do padrão MPI 1.1 realizada pela Universidade de Notre-Dame*

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 .

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.

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] Herrod, S., Roseblum, M., Bugnion, E., Devine, S., Bosch, R., Chaplin, J., Givil, K., Tedosiu, D., Witchel, E. and Verghese, B. The SimOS simulation environment. Tech. Rep., Computer System Laboratory – Stanford University, (February 1996).

[2] Falsafi, B., And Wood, D. A. Cost/performance of a parallel computer simulator. *In Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PAD´94)* (USA, 1994)

[3] Mukherjee, S. S., Adve, S. V., Austin, T., Emer, J., and Magnusson, P. S. Performance simulaton tool. *Computing Magazine sn* (February 2202).

[4] Burger, D., and Austin, T. M. The simplescalar tool set, version 2.0. Tech. Rep. 1342, University oh Wisconsin-Madison and Intel Corporation, June 1997.

[5] Sharma, A., Nguyen, A. T., and Torrelas, 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.

[6] Sipper, M., The emergency of cellular computing. *IEEE Computer magazine* sn (July 1999).

[7] Virtutech website. http://www.virtutech.com

[8] Et al., P. M. Simics: A full system simmlation platform. *Computer magazine sn* (February 2002).

[9] Almasi, G. S., Cascaval, C., Castanos, J. G., Denneau, M., Donath, W., Eletheriou, Giampapa, M., Ho, H., Lieber, D., Moreira, J. E., Newns, D., Snir, M., Ans Henry S. Warren, J. Demonstrating the scability of a molecular dynamics application on a petaflop computer. Tech Rep. RC-21965, IBM Researche, February 2001.

[10] Steinmacher-Burow, B. D., *The Torus for the Software Point of View.* IBM T. J. Watson Research Center, Yorktown Heights, NY, 2002.

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

[12] IBM – Microeletronics Research Triangle Park, *NC. EMAC4 (Ethernet Core) User Manual*

[13] Padrão ieee 1149.1 para protas de teste em circuitos integrados. http://standards.ieee.orh/reading/ieee/std_public/description/testtech%/1149.1-1990_desc.html.

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

[15] IBM – Microeletronics Research Triangle Park, *PowerPC 440 User Manual*

[16] Lam / Mpi parallel computing. http://www.lam-mpi.org/.

[17] Mpi: A message-passing interfece standard. http://www.lam-mpi.org/.

[18] Bobet, D and Cesati M. *Understanding the Linux Kernel*, O´Reilly, October 2000.

Adaptive Rule-Driven Devices
General Formulation and Case Study

João José Neto
joao.jose@poli.usp.br

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05.508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

Abstract

A formal device is said to be adaptive whenever its behavior changes dynamically, in a direct response to its input stimuli, without interference of external agents, even its users. In order to achieve this feature, adaptive devices have to be self-modifiable. In other words, any possible changes in the device's behavior must be known at their full extent at any step of its operation in which the changes have to take place. Therefore, adaptive devices must be able to detect all situations causing possible modifications and to adequately react by imposing corresponding changes to the device's behavior. In this work, devices are considered whose behavior is based on the operation of subjacent non-adaptive devices that be fully described by some finite set of rules. An adaptive rule-driven device may be obtained by attaching adaptive actions to the rules of the subjacent formulation, so that whenever a rule is applied, the associated adaptive action is activated, causing the set of rules of the subjacent non-adaptive device to be correspondingly changed. In this paper a new general formulation is proposed that unifies the representation and manipulation of adaptive rule-driven devices and states a common framework for representing and manipulating them. The main feature of this formulation is that it fully preserves the nature of the underlying non-adaptive formalism, so that the adaptive resulting device be easily understood by people familiar to the subjacent device. For illustration purposes, a two-fold case-study is presented, describing adaptive decision tables as adaptive rule-driven devices, and using them for emulating the behavior of a very simple adaptive automaton, which is in turn another adaptive rule-driven device.

Keywords

adaptive devices, rule-driven formalisms, self-modifying machines, adaptive decision tables, adaptive automata.

1. Introduction

Despite many adaptive devices have been created and used in the last decade, notations had not been elaborated in a way that the represented adaptive formalism be as close as possible to the original non-adaptive underlying formulation. Among several other reasons, self-modifying formalisms have not been extensively employed, as a consequence of the complexity of the existing formulations, which make them diffcult to use.

In this work we introduce a general proposal for the formulation of adaptive devices based on rule-driven subjacent non-adaptive ones. We expect that the proposed formulation be clear, intuitive and easy to learn, without adding complexity to reading, writing or interpreting the notations. We also expect that the proposal be general enough, in the sense that it be insensitive to the underlying non-adaptive formalism, so that all achieved results remain

valid even possibly changing the nature of the subjacent rule-driven system.

2. (Non-adaptive) Rule-driven devices

Formal state machines are popular mathematical tools used for describing and modeling actual real life systems. At each stage of their operation, these devices assume some configuration, which usually comprehends the contents of the whole set of information-holding elements and the current status of the device. In conjunction with the input stream yet to be processed, despite how the current configuration was reached, the device's configuration fully determines its further behavior. A formal machine operates by successively moving the device from one configuration to another, in response to stimuli consumed from its input.

Without loss of generality, we may state that such devices start their operation at some initial configuration that follows well-known fixed restrictions. After having processed the full input sequence of stimuli, the device reaches some configuration which may indicate that its whole input stream has been either accepted or rejected. Therefore, we may split the set of all possible configurations for the device into two partitions: one for the accepting configurations and the other for the rejecting ones.

A (*non-adaptive*) *rule-driven device* is any formal machine whose behavior depends exclusively on a finite set of rules which map each possible configuration of the device into a corresponding next configuration. The device is said to be *deterministic* if and only if, for any given initial or intermediate configuration and any input stimulus, its defining set of rules determines one and only one next configuration. The device is *non-deterministic* otherwise. Non-deterministic devices allow more than one valid move at each moment. So, their use requires that all possible next moves be tried as intermediate steps toward some final accepting configuration. Inefficiencies caused by such trial-and-error operation usually turn non-deterministic devices inadequate for sequential implementation. Therefore, in order to achieve efficiency, choosing deterministic equivalent devices is highly recommended. Let us define $ND = (C, NR, S, c_0, A\ NA)$ where:

- *ND* is some *rule-driven device*, described by a set of rules *NR*.
- *C* is its set of possible *configurations*, and $c_0 \in C$ is its *initial configuration*.
- *S* as the (finite) set of all possible *events* that are valid *input stimuli* for *ND*, with $\varepsilon \in S$.
- $A \subseteq C$ (resp. $F = C - A$) is the subset of its *accepting* (resp. *failing*) *configurations*.
- ε denotes "*empty*", and represents the *null* element of the set it belongs to.
- $w = w_1 w_2 ... w_n$ is a *stream of input stimuli*, where $w_k \in S - \{\varepsilon\}, k = 1, ..., n$ with *with* $n \geq 0$
- *NA* is a (finite) set, with $\varepsilon \in NA$, of all possible symbols to be output by *ND* as side effects to the application of the rules in *NR*. In practice, *output symbols* in *NA* may be mapped into procedure calls, so an output generated by applying any rule may be interpreted as a call to its corresponding procedure.
- *NR* is the set of *rules* defining *ND* by a relation $NR \subseteq C \times S \times C \times NA$. Rules $r \in NR$ have the form $r = (c_i, s, c_j, z)$, meaning that, in response to any *input stimulus* $s \in S$, *r* changes the *current configuration* c_i to c_j. , consumes *s* and outputs $z \in NA$ as a side-effect.

A rule $r = (c_i, s, c_j, z)$, with $r \in NR$; c_i; $c_j \in R$; $s \in S$; $z \in NA$, is said to be *compatible* with the current configuration *c* if and only if $c_j = c$ and *s* is either empty or equal to the device's current input stimulus. In this case, the application of a single compatible rule moves the device to configuration *cj* (denoted by $c_i \Rightarrow^s c_j$) and appends *z* to its output stream. Note that *s*, *z* or both may be empty. Let $c_i \Rightarrow^{\sim} c_m$, $m \geq 0$ denote $c_i \Rightarrow^{\varepsilon} c_1 \Rightarrow^{\varepsilon} c_2 \Rightarrow^{\varepsilon} ... \Rightarrow^{\varepsilon} c_m$, an optional sequence of empty moves. Let $c_i \Rightarrow^{\sim w_k} c_j$, denote $c_i \Rightarrow^{\sim} c_m \Rightarrow^{w_k} c_j$, an optional sequence of empty moves followed by a non-empty one consuming the symbol *wk*. An input stream $w = w_1 w_2 ... w_n$ is said to be *accepted* by *ND* when $c_0 \Rightarrow^{\sim w_1} c_1 \Rightarrow^{\sim w_2} ... \Rightarrow^{\sim w_m} c_n \Rightarrow^{\sim} c$ (for short $c_0 \Rightarrow^{w} c$ with $c \in A$). Complementarily, *w* is said to be *rejected* by *ND* when $c \in F$. The *language* described by *ND* is the set $L(ND) = \{w \in S^* | c_0 \Rightarrow^w c, c \in A\}$ of all streams $w \in S$ that are accepted by *ND*.

3. Adaptive devices

Define *T* as a built-in time counter that starts at 0 and is automatically incremented by 1 when a non-null adaptive action is executed. Each value *k* assumed by *T* may subscript the names of time-varying sets. In this case, it selects *AD*'s operation step *k*.

A device $AD = (ND_0, AM)$ is said to be adaptive whenever, for all operation steps $k \geq 0$, *AD* follows the behavior of ND_k until the execution of some non-null adaptive action starts its operation step $k + 1$ by changing its set of rules.

At any *AD*'s operation step $k \geq 0$, being ND_k the corresponding subjacent device defined by NR_k the execution of some non-null adaptive action evolves ND_k to ND_{k+1}. So, *AD* starts its operation step $k + 1$ by creating the set NR_{k+1} as an edited version of NR_k. Thereafter, *AD* will follow the behavior of ND_{k+1} until a further non-null adaptive action causes another step to start. This procedure iterates until the input stream is fully processed.

AD starts its operation at c_0, being $AD_0 = (C_0, AR_0, S, c_0, A, NA, BA, AA)$ its initial shape. At step $k \geq 0$, an input stimulus (always) moves *AD* to a next configuration and then starts its operation step $k + 1$ if and only if a non-adaptive action is executed. So, being *AD* at step *k*, with shape $AD_k = (C_k, AR_k, S, c_k, A, NA, BA, AA)$, the execution of a non-null adaptive action leads to $AD_{k+1} = (C_{k+1}, AR_{k+1}, S, c_{k+1}, A, NA, BA, AA)$. In this formulation:

- $AD = (ND_0, AM)$ is some adaptive device, given by an initial subjacent device ND_0 and an adaptive mechanism *AM*.
- ND_k is *AD*'s subjacent non-adaptive device at some operation step *k*. ND_0 is *AD*'s initial subjacent device, defined by a set NR_0 of non-adaptive rules. By definition, any non-adaptive rules in any NR_k mirror the corresponding adaptive ones in AR_k.
- c_k is the set of all possible configurations for *ND* at step *k*, and $c_k \in C_k$ is its starting configuration at step *k*. For $k = 0$, we have respectively c_0, the initial set of valid configurations, and $c_0 \in C_0$, the initial configuration for both ND_0 and *ND*.
- ε ("empty") denotes the absence of any other valid element of the corresponding set.
- *S* is the (finite, fixed) set of all possible events that are valid input stimuli for $(\varepsilon \in S)$.
- $A \geq C$ (resp. $F = C - A$) is the subset of its *accepting* (resp. *failing*) *configurations*.
- *BA* and *AA* are sets of adaptive actions, both containing the null action $(\varepsilon \in BA \cap AA)$
- $w = w_1 w_2 ... w_n$ is a stream of input stimuli, where $w_k \in S - \{\varepsilon\}, k = 1, ..., n$.
- *NA*, with $\varepsilon \in NA$, is a (finite, fixed) set of all possible symbols to be output by *AD* as side effects to the application of adaptive rules. Just like in nonadaptive devices, the output stream may be interpreted as a corresponding sequence of procedure calls.
- AR_k is a set of adaptive rules, given by a relation $AR_k \subseteq BA \times C \times S \times C \times NA \times AA$. In particular, AR_0 defines the initial behavior of *AD*. Adaptive actions map the current set of adaptive rules AR_k of *AD* into a new set $AR_k + 1$ by adding adaptive rules to AR_k and/or deleting rules from it. Rules $ar \in AR_k$ have the form $ar = (ba, c_i, s, c_j, aa)$, meaning that, in response to some input stimulus $s \in S$, *ar* initially executes the adaptive action $ba \in BA$; the execution of *ba* is aborted if it eliminates *ar* from AR_k; otherwise, it applies the subjacent non-adaptive rule $nr = (c_i, s, c_j, z) \in NRk$, as described before; finally, it executes adaptive action $aa \in AA$.
- Define *AR* as the set of all possible adaptive rules for *AD*.
- Define *NR* as the set of all possible subjacent non-adaptive rules for *AD*.
- $AM \subseteq BA \times NR \times AA$, defined for a particular adaptive device *AD*, is an *adaptive mechanism* to be applied at any operation step *k* to each rule in $NRk \subseteq NR$. *AM* must be such that it operates as a function when applied to any sub-domain $NR_k \subseteq NR$. That will determine a single pair of adaptive actions to be attached to each non-adaptive rule. The set $AR_k \subseteq AR$ may be

synthesized by collecting all adaptive rules obtained by merging such pairs of adaptive actions to the corresponding non-adaptive rules in NR_k. Equivalently, we may build NR_k by removing all references to adaptive actions from the rules in AR_k.

The algorithm below sketches the overall operation of any rule-driven adaptive device.

1. Set the device at its initial configuration.

2. Set the input stream at its leftmost event.

3. If there are no more events to be processed, then go to step 8 else feed the device by picking the next event to be processed.

4. Choose the next adaptive rule to be applied:
Given the current configuration $c_T \in C_T$ and stimulus $S_T \in S$, extracted from the input stream not yet processed, search the set NR_T for compatible rules, i.e., a set of rules that may be applied under the current circumstances, and collect them in a set $CR_T = \{ark \in AR_T | ar = (ba, c_T, s, c, z, aa), s \in \{s_T, \varepsilon\}; c, c_T \in C_T; ba \in BA; aa \in AA, z \in NA\}$ There are three cases to be considered:

   (a) if CR_T is empty, no moves at all are allowed for the device (because AR_T is incompletely specified), then the input stream is rejected by default.

   (b) if $CR_T = \{ar^k\}$ for some $ar^k = (ba^k, c_T, s, c^K, z^K, aa^k) \in AR_T$, then a single rule is available, for deterministic application. By doing so, the device will reach a well-defined next configuration $C_{T+1} = c$.

   (c) if $CR_T = \{ar^k = (ba^k, c_T, s, c^k, z^k, aa^k) | k = 1, 2, \ldots, m\}$, then all these *m* rules are equally allowed for being non-deterministically applied to the current configuration, so all of them are applied in parallel to the current configuration, as usual in the operation of non-deterministic devices. Obviously, in non-parallel environments, such parallelism must be exhaustively simulated, e.g. by applying some backtracking strategy.

5. If adaptive action ba^k is the null adaptive action ε in the rule being applied, then proceed to step 6, otherwise, apply the adaptive action ba^k to the current set of rules, yielding a new intermediate configuration for the device *AD*. Note that, in some cases, even the adaptive rule ar^k being applied may erase itself by executing ba^k. In such an extreme case, the application of *arp* is aborted by returning to step 4.

6. Apply the rule $nr_k = (c_T, s, c^k, z^k)$, just as defined by the underlying nonadaptive device, to the current (intermediate) configuration of *AD*, yielding another (intermediate) configuration.

7. If the adaptive action aa^k in the adaptive rule ar^k being executed is the null adaptive action ε, proceed to step 3, otherwise apply aa^k to the current set of adaptive rules, finally yielding the next configuration for the device. As in step 5, the execution of the adaptive action may also erase ar^k. In this case, however, no further action is needed.

8. If the current configuration is an accepting one, then accept the input stream, otherwise reject it, then stop.

4. Example

Decision tables are widely accepted tools among information systems programmers and software engineers. Adaptive decision tables constitute an interesting application of the concept of adaptive devices to the field of information systems. Adaptive decision tables may be defined as the class of adaptive devices that use traditional (non-adaptive) decision tables as their underlying formalism.

4.1 (Non-adaptive) Decision tables

Decision tables may be viewed as tabular devices that encode a set of rules represented by conditions and corresponding actions to be executed when those conditions are matched. In typical decision tables (see Table 1) rules are represented as columns while rows are employed to encode conditions (condition rows) and actions (action rows). In each rule, marked cells corresponding to each condition row refer to relevant conditions to be tested, and indicate whether that condition is to be tested for

true (*T*) or false (*F*) (non-marked conditions are not to be tested), while marked cells in action rows indicate that the corresponding action is expected to be performed in response to a match in all marked conditions.

		1	2	3	4	5	6	7	8	9
Condition Rows	c_1	T	T	F	-	-	F	T	T	-
	c_2	-	T	T	-	-	F	-	F	T
	...									
	c_n	-	-	-	-	T	-	F	F	-
Aspect Rows	z_1	F	T	T	T	F	F	F	F	T
	z_2	F	F	T	T	F	T	T	F	F
	...									
	z_n	F	F	F	F	T	F	F	T	T

Table 1. Structure of a typical non-adaptive decision table.

Rules in the decision table are encoded as follows:

Condition rows: all cells of the rule corresponding to conditions to be tested are filled with a boolean value (*T* or *F*) corresponding to the particular value to be tested for that condition. A special null mark (*i*) indicates that the associated condition is not to be tested.

Action rows: cells of the rule is chosen to be applied.

Operation: The operation of such non-adaptive decision tables is quite straightforward:

1. First, the status of the system is checked against the combinations of conditions stated in each of the rules encoded in the table.

2. If no rule matches the current status, then no action is executed at all.

3. If a single rule matches the current status, then we have a deterministic choice, so the matching rule is chosen to be applied.

4. If more than one rule match the current status, then we face a non-deterministic situation. Consequently, all such rules are to be applied in parallel. In practice, parallelism may be simulated, e.g. by some exhaustive backtracking strategy.

5. The selected rule is then applied by executing the set of all actions indicated with a boolean value *T* in the cells of the rule corresponding to action rows.

6. Once the selected rule has been applied, the decision table gets ready to be used again.

For instance, let us assume that condition c_1 is *F* and condition c_2 is *T*. In our decision table, obviously only the rule encoded in column 3 matches such status. Therefore, in this case the decision table will activate actions ra_1 and ra_2 for execution, as we may easily observe by inspecting the action rows specified in rule 3. Note that if rule 1 had been selected instead, no actions would have been called at all, since all action rows in rule 1 are filled with *F*. It is obvious from this simple example that decision tables are very easy to design and use.

Unfortunately, these classical devices are static, in the sense that their individual rules are all predefined and never change throught the operation of the device. Furthermore, the set of rules defining classical decision tables is not allowed to change, so in classical decision tables there is no dynamic inclusion or exclusion of rules. In order to provide more flexibility to this useful and popular tool, we may use it as the basic underlying non-adaptive formalism for building a far more powerful adaptive device.

4.2 **Adaptive decision tables**

Adaptive decision tables (see Table 2) are easily obtained from conventional (non-adaptive) ones by adding to them further rows encoding the adaptive actions to be performed before ("before-" adaptive action rows) and after the rule is applied ("after-" adaptive action rows).

When adaptive actions are executed, the adaptive table usually has its set of rules modified, therefore correspondingly changing the number of columns in the adaptive decision table. Note that with the chosen layout, however, the number of rows of the adaptive decision tables remains unchanged, since adaptive actions do not modify any of the items encoded in their rows. This property is truly valuable for implementation purposes.

For operating such an adaptive device, the subjacent non-adaptive decision table is first used for determining the rule(s) matching the current situation of the condition predicates. Then, the selected adaptive rule is performed by executing the indicated "before-" adaptive actions, then applying the subjacent non-adaptive rule, and finally executing the indicated "after-" adaptive actions.

		1	2	3	4	5	6	7	8	9
Condition Rows	c_1	T	T	F	-	-	F	T	T	-
	c_2	-	T	T	-	-	F	-	F	T
	...									
	c_n	-	-	-	-	T	-	F	F	-
Aspect Rows	z_1	F	T	T	T	F	F	F	F	T
	z_2	F	F	T	T	F	T	T	F	F
	...									
	z_n	F	F	F	F	T	F	F	T	T
"before" Adaptive Action rows	ba_1	T	F	F	F	F	F	F	T	F
	ba_2	F	F	T	F	T	F	F	T	F
	...									
	ba_n	F	T	F	F	F	F	F	F	F
"after" Adaptive Action rows	aa_1	F	T	T	F	F	F	F	T	F
	aa_2	F	F	F	T	F	T	F	T	F
	...									
	aa_n	F	F	F	F	F	T	F	T	F

Table 2. Structure of an adaptive decision table based on the non-adaptive table in fig. 1 as its subjacent device.

In some cases, when executing its adaptive actions, some adaptive rule being applied may even exclude itself. Whenever the currently used rule happen to be eliminated by its own before- adaptive action, its application is aborted, and the next rule to be applied is elected from the resulting set of rules.

The aspect of an adaptive decision table is shown in the example depicted ahead. The upper half of the table in the figure refers to the corresponding underlying non-adaptive decision table, while its lower half represents the attached adaptive mechanism.

Note that by associating adaptive actions in this way to the usual formulation of decision tables, no substantial changes are introduced for the user, since at first glance adaptive actions might be simply interpreted as additional standard actions to be executed in response to some particular matching of conditions. From a conceptual viewpoint, however, the execution of adaptive actions has significantly deeper implications, since it affects the decision table by allowing changes to be imposed to its own behavior.

4.3 **Application**

In the table 2 illustrated above, rule 3 is activated when condition c_1 is *F* and condition c_2 is *T*, regardless to the other conditions. Under this situation, the application of this adaptive rule operates as follows:

1. Execute adaptive action b_2 (which will probably change the device's set of rules) *before* the subjacent non-adaptive rule is applied.

2. Apply the underlying rule: actions r_1 and r_2 are performed just as they would be executed in the classical non-adaptive case.

3. Execute adaptive action *a*1 (probably changing the decision table again) *after* the application of the underlying non-adaptive rule.

Adaptive functions Adaptive actions may be defined by means of abstractions called adaptive functions, in a way correspondingly similar to that of function calls and function declarations in a usual programming language. Adaptive functions define generic abstractions while adaptive actions correspond to adaptive function specific calls. Adaptive actions customize the corresponding adaptive function abstraction by assigning arguments to their formal parameters according to each particular needs.

Specifying adaptive functions In order to state exactly how each adaptive action is expected to operate, we must provide some further information: the name of the corresponding adaptive function, the set of parameters to be used, the elementary adaptive actions to be applied and the exact way parameters, variables and generators are to be employed.

Thus, the specification of adaptive functions must include the following items:

- *name*: a symbolic name, used for referencing adaptive functions. When calling adaptive actions, the name of the corresponding adaptive function is used to select among available adaptive actions.

- *(formal) parameters*: a set of symbolic names that are used for referencing values passed as arguments to an adaptive function at the time it is called. All instances of the formal symbolic parameters, once replaced with the values associated to their corresponding arguments within the body of the adaptive function, may not be further modified throughout the execution of the adaptive function.

- *variables*: these are symbolic names used for holding values resulting from the application of some rule-searching elementary adaptive function. Variables are filled only once and their values remain unchanged during the execution of the adaptive function.

- *generators*: these elements are symbolic names that refer to new values each time they are used. Once generators are filled with some value, they do not change any more while the adaptive function is active.

- *body*: the body of an adaptive function encodes all editions needed to make the desired changes to the current set of rules of the decision table.

Elementary adaptive actions are editing primitives that allow either testing the rule set or specifying single modifications to the rules of an adaptive decision table. The body of an adaptive decision table consists essentially of a set of elementary adaptive actions.

There are three kinds of elementary adaptive actions that may be combined within the body of an adaptive function in order to specify its operation:

- *rule-searching elementary adaptive actions*: these actions do not modify the set of rules, but allow searching it for rules matching a given pattern.

- *rule-erasing elementary adaptive actions*: these actions remove rules that match a given pattern from the current set of rules.

- *rule-inserting elementary adaptive actions*: these actions allow adding a rule with a specified pattern to the current set of rules.

Encoding adaptive functions In order to encode adaptive functions, a format must be chosen for each of the component items listed above. It should be convenient that the format be similar to that of adaptive decision tables. We chose to include the specification of adaptive functions as part of the adaptive decision table itself, since adaptive functions are meaningless outside the environment they act on. The format adopted for encoding adaptive functions within adaptive decision tables will be informally introduced through the following example.

Overall format for adaptive decision tables Adaptive decision tables as defined here will be drawn as an extension of the notation already discussed for non-adaptive tables. So, the subjacent decision table is represented as usual and the adaptive mechanism is added by inserting the following elements in the rows:

- one heading row containing a tag for specifying the type of each column (*H*= header of the specification of an adaptive function; +, *i*, ? = including, excluding, nspecting elementary adaptive action; *S*= starting rule; *R*= normal rule; *E*= ending rule).

- one extra row for the names of each adaptive function used.

- one extra row for the names of each parameter, variable or generator used by the adaptive functions.

- in the example below, for better legibility of the table, assignments and comparisons referring to variables used by standard (non-adaptive) actions are denoted explicitly and not as function calls.

Similarly, the following additions have been done to the columns of the table:

- one header column for each adaptive function (tag = *H*) starting the specification of the adaptive function. This header must include a tag *B* or *A* in the cell corresponding to the name of the before- or after- adaptive functions, respectively, a tag *P* in each cell corresponding to a formal parameter, a tag *V* in

each cell corresponding to a variable and a tag *G* in each cell corresponding to a generator. Each header column is followed by a set of columns related to elementary adaptive actions. This set is finished when a starting rule or another header is found.

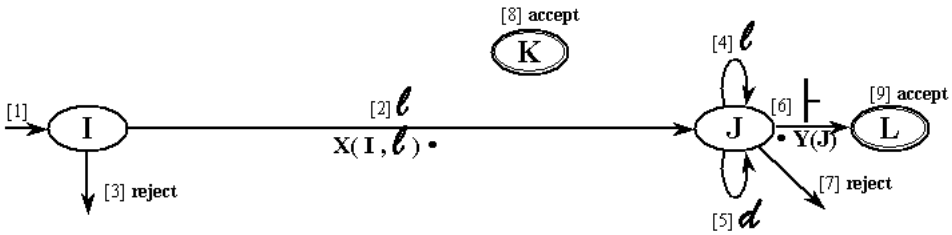
- in columns denoting elementary adaptive actions (tags +, *i* or ?) the cells corresponding to conditions are filled with the value the condition is to be tested against; cells corresponding to actions to be executed are marked; cells corresponding to assignments are filled with the value to be assigned. Required adaptive actions are marked, and the cells corresponding to their parameters are filled with a constant or the name of a variable, a parameter or a generator to be passed as an argument. Homonymous parameters must be avoided between adaptive functions called within the same rule.

- one column for the starting rule (tag = *S*) of the adaptive automaton, standing for the rule to be applied before any other. In this column, actions are activated in order to initialize all operating conditions for the device. A sequence of normal rules follow this column, ending with the ending rule, which closes the specification of the table.

- columns denoting normal rules (tag = *R*) specify all rules defining the decision table. Each normal rule is specified by filling condition cells to be tested with constants or names of variables, generators or formal parameters; actions cells and adaptive actions are specified just as described above for elementary adaptive actions.

- a single column, denoting the ending rule (tag = *E*), serves as a delimiter for the set of current rules in the adaptive decision table, and represents only a logical marker.

Illustrative Example Let us illustrate the encoding of adaptive functions by means of a low-complexity adaptive example. It is shown as a complete adaptive decision table in Fig. 1. In this example, we define two adaptive functions: The first adaptive function is named *X* and has two formal parameters, $p1$ and $p2$, and uses one generator, g_1, while the second one, *Y* , has one formal parameter, $q1$. Note that no variables are used in this example, but if there



rule numbers			1	2	3	4	5	6	7	8	9	10
tags		H + + + + + - H - +	S	R	R	R	R	R	R	R	R	E
subjacent table conditions	state=	p1 g1 g1 g1 g1 p1 q1 q1		I	I	J	J	J	J	K	L	
	input=	p2 ℓ d ⊢ p2 ⊢ ⊢		ℓ		ℓ	d	⊢				
	state:=	g1 J J L J L K	I	J		J	J	L				
subjacent table actions	get	✓	✓	✓		✓	✓					
	accept:=	✘			✘				✘	✓	✓	
adaptive funct. names	X	B ✓ ✓ ✓		✓								
	Y	✓ A ✓						✓				
parameters	p1	P g1 g1 p1		I								
	p2	P ℓ d p2		ℓ								
	q1	g1 P q1						J				
generators	g1	G										

Fig. 1. Initial adaptive decision table and the corresponding initial topology of the adaptive automaton

were variables, new corresponding rows would have been added, since they are denoted and used in the same way parameters and generators are.

Other features that were not used in this example are the calls to before- and after- adaptive actions within adaptive functions. If they were present, additional appropriately tagged columns would have been included for representing them.

The very simple adaptive decision table in Fig.1 illustrates the encoding of the information needed for implementing a simple adaptive device. A very brief flash of this adaptive decision table's operation is given in order to illustrate its behavior. In order to shorten the text and help visualizing the evolution of our adaptive device, the automaton represented by the adaptive decision table is represented graphically. New transitions are drawn in heavy lines. Bracketed numbers associated to transitions in the figures refer to the corresponding rule numbers in the adaptive decision table. The initial topology of our automaton is also shown in Fig. 1.

After consuming the first token ℓ from the input stream, one of the automaton's transitions is replaced by five new ones, as shown in Fig. 2.

After consuming one more token, the automaton's shape is changed again, resulting the shape in Fig. 3.

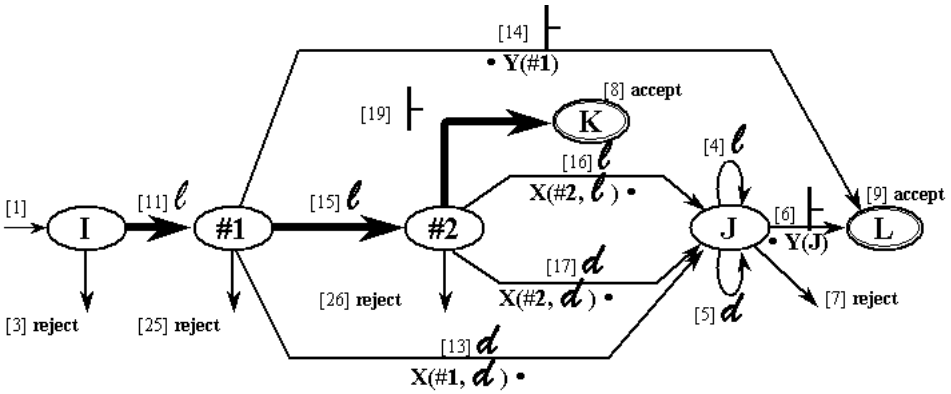
After accepting the whole identifier in the input stream, the resulting automaton is the one depicted in Fig. 4.

In this situation, if a second identifier $\ell d \vdash$ is processed by our adaptive device, the resultant topology of the adaptive automaton will be that shown in Fig. 5.

Note that after each identifier is fully processed, the path starting at state I and ending at state K will have the shape of a tree, each of whose leaves (transitions pointing to state K) correspond to one different valid identifier found in the input text.

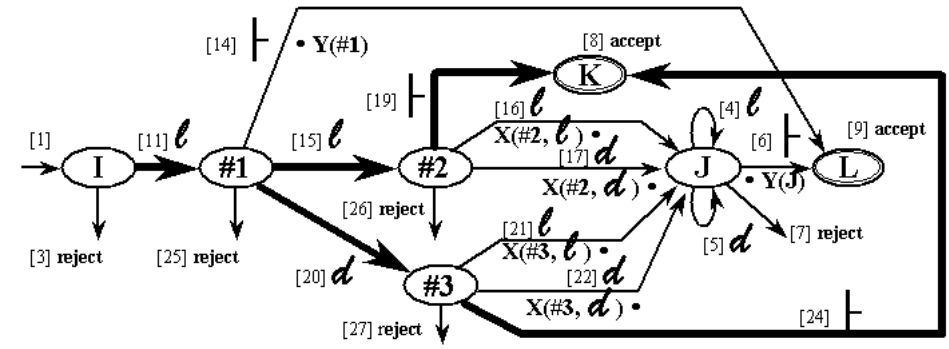
Not all possible options of the model have been explored in this example, but the given illustration is complex enough to be used as a guide for developing other projects with this technique.

This example illustrates the use of adaptive decision tables as an alternative way to implement adaptive automata-based logic. Applications of adaptive automata



											1	11	3	4	5	6	7	8	9	15	13	14	25	16	17	19	26	10					
Tag→	H	+	+	+	+	+	-	H	-	+	S	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	E				
state=		p1	g1	g1	g1	g1	p1		q1	q1		I	I	J	J	J	J	K	L	#1	#1	#1	#1	#2	#2	#2	#2					
input=		p2	ℓ	d	⊢		p2		⊢	⊢		ℓ		ℓ	d	⊢				ℓ	d	⊢		ℓ	d	⊢						
state:=		g1	J	J	L		J		L	K	I	#1		J	J	L				#2	J	L		J	J	K						
get		✓									✓	✓		✓	✓					✓												
accept:=						✘							✘				✘	✓	✓				✘				✘					
X	B		✓	✓			✓														✓				✓	✓						
Y					✓			A	✓							✓						✓										
p1	P		g1	g1			p1														#1				#2	#2						
p2	P		ℓ	d			p2														d				ℓ	d						
q1					g1			P	q1							J						#1										
g1	G																															

Fig. 2. Adaptive automaton after consuming $\ell\ell$



	1	11	3	4	5	6	7	8	9	15	20	14	25	16	17	19	26	21	22	24	27	10
Tag→	S	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	E
state=		I	I	J	J	J	J	K	L	#1	#1	#1	#1	#2	#2	#2	#2	#3	#3	#3	#3	
input=		ℓ		ℓ	d	⊢				ℓ	d	⊢		ℓ	d	⊢		ℓ	d	⊢		
state:=	I	#1		J	J	L				#2	#3	L		J	J	K		J	J	K		
get	✓	✓		✓	✓					✓	✓											
accept:=			✘				✘	✓	✓				✘				✘				✘	
X														✓	✓			✓	✓			
Y						✓						✓										
p1														#2	#2			#3	#3			
p2														ℓ	d			ℓ	d			
q1						J						#1										
g1																						

	Tag→	H	+	+	+	+	+	-	H	-	+
	state=		p1	g1	g1	g1	g1	p1		q1	q1
	input=		p2	ℓ	d	⊢		p2		⊢	⊢
	state:=		g1	J	J	L		J		L	K
	get		✓								
	accept:=						✘				
X		B		✓	✓			✓			
Y						✓			A	✓	
p1		P		g1	g1			p1			
p2		P		ℓ	d			p2			
q1						g1			P	q1	
g1		G									

Fig. 3. Adaptive automaton after consuming $\ell\ell$

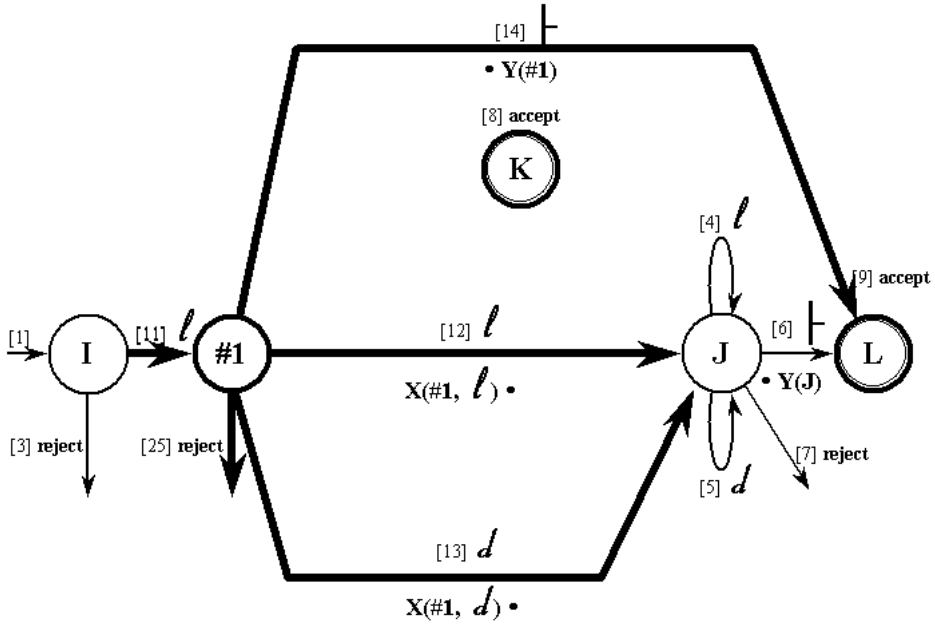


Fig. 4. Adaptive decision table after consuming $\ell\ell \vdash$ and the correspondig adaptive automaton

have been shown in [1]. The target of this decision table is to read an input sequence formed of letters and digits and to collect identifiers formed in the usual way, say, as a non-empty sequence of letters and digits starting with a letter, and ending with the special end-marker ⊢ . Whenever a new identifier is found, the adaptive decision table is adequately modified so that the identifier be thereafter registered as an already known one, while already known identifiers are simply accepted by the table without modifying it. So, it operates as a purely syntactic name-collecting device for which all needed available information is permanently encoded in the adaptive decision table.

Further descriptions of the operation of adaptive devices are found in [2].

5. Previous experience with adaptive techniques

Adaptive technology concerns to techniques, methods and disciplines referring to actual practical applications of adaptive devices. Historically, adaptive devices emerged from the field of formal languages and automata. Consequently, early applications of such devices have been in the area of rigorous definition of formal and computer languages. In this area, we may list the works by Burshtein [3], Shutt [4], Cabasino [5], Rubinstein [6], Neto [7] and Iwai [8]. For example, adaptive automata have been proposed as a practical formalism for the representation of languages with context dependencies [6], [7]. On another hand, adaptive grammars were also introduced as generative devices whose operation also allows their use in the rigorous definition of context-dependent languages [6], [8].

An early meta-system based on adaptive automata has been proposed and implemented which allowed many tests to be carried out in a comfortable form [9]. This work was a practical proof that adaptive engines might be useful and not so much difficult to design and implement. Burshtein [3], Shutt [4] and Christiansen [10] have proposed adaptive grammatical devices whose operations remind two-level Von Wijngaarden grammars [11], with enough power to express complex type-1 and type-0 languages. For use in the specification and analysis of real time reactive systems, we also find some works based on

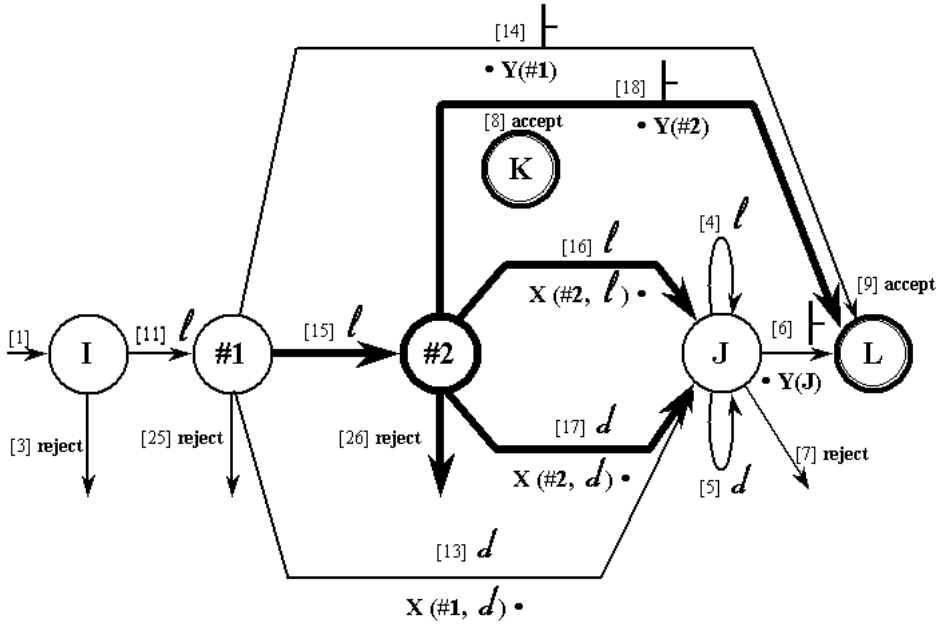


Fig. 5. Adaptive decision table after consuming both $\ell\ell \vdash$ and $\ell d \vdash$ and the corresponding adaptive automaton

adaptive versions of classical statecharts [12]. As an evolution of this work, in addition to reactive aspects, mechanisms based on Petri Nets have been added to adaptive statecharts for explicit analysis of synchronization aspects of concurrent adaptive systems [13].

Another extremely interesting manifestation of the power and practicality of adaptive devices for the specification and implementation of complex systems has been the formulation and use of adaptive Markov chains, which have been very successfully used in the design and physical implementation of a computer musicgenerating system [14]. Adaptive devices are also being tested in the field of decision-making systems. An adaptive-automata-powered system prototype was implemented as a tool to be used in the automatic generation and selection of solutions for problems with high computational complexity.

Artificial intelligence is a field that may be strongly benefited by adaptive technology, since adaptive devices have a built-in mechanism for acquiring, representing and manipulating knowledge. Although not been actually implemented, a proposal has been made for the application of adaptive automata in computer education, as a learning mechanism of a computer-aided tutoring system and as a model of the student's progress in such tutoring systems. So, learning is a natural feature shared by all adaptive devices.

In particular, a small experiment with regular languages has been reported that shows the potential of adaptive automata as a good device for constructing grammar inference systems [15]. Another area in which adaptive devices showed their strength is the specification and processing of natural languages. One of the works in this field employed adaptive automata as the main mechanism of an automatic tagging system for texts in Portuguese language. Further works are being currently developed in this direction, with many good intermediate results in the representation of syntactical context-dependent features of natural languages. Simulation and modeling of intelligent systems are also concrete applications of adaptive formalisms, as it was illustrated in the description of the control mechanism of an intelligent autonomous vehicle that collects information from its environment and builds a map for easier navigation. Many other applications for adaptive devices are possible in several fields. They must be extensively explored through the well-succeeded search for simple and effcient alternative ways to perform complex computation tasks.

6. Conclusions

As a result of this work, we have achieved, for adaptive devices, a formulation in which nothing beside the formal adaptive mechanisms is introduced, giving the proposed formulation the character of a simple extension of the underlying non-adaptive device. So, the integrity and even the intuition of the underlying formulation in which the adaptive device is based are preserved, as well as all their properties. Consequently, after getting familiarized with the concept of adaptive devices, users have no extra need of learning further concepts and notations, so they are free for using already designed and tested non-adaptive devices as a basis for easily building adaptive versions. From another viewpoint, with the proposed formulation users may directly identify the underlying nonadaptive device at any moment during the operation of an adaptive device, so simplifying its debugging effort and increasing the comprehension of the adaptive device by its designers. Our proposal has a very clean formulation, allowing the user to be permanently aware of all phenomena concerning the underlying mechanism. No new notations or concepts are introduced, except those involving adaptive features, therefore, our proposal offers a formulation that is indeed intuitive and easy to learn. It is also general to a large extent, since it does not depend on the nature of the underlying non-adaptive formalism chosen. We expect that this simple contribution encourage not only the revisiting and use of existent self-modifying formalisms, but also the formulation of new adaptive devices for solving problems that are hard to solve with usual non-adaptive tools.

Acknowledgements

Our sincere acknowledgement to the anonymous referees for their valuable questions and suggestions for the improvement of this paper.

References

[1] José Neto, J. Solving Complex Problems E±ciently with Adaptive Automata. CIAA 2000 - Fifth International Conference on Implementation and Application of Automata, July 2000 - London, Ontario, Canada.

[2] José Neto, J. Contribuição à metodologia de construçãode compiladores. SãoPaulo, 1993, 272p. Thesis (Livre-Docência) Escola Politécnica, Universidade de São Paulo. [In Portuguese]

[3] Burshteyn, B. Generation and recognition of formal languages by modifiable grammars. ACM SIGPLAN Notices, v.25, n.12, p.45-53, 1990.

[4] Shutt, J.N. Recursive adaptable grammar. M.S. Thesis, Computer Science Department, Worcester Polytecnic Institute, Worcester MA, 1993.

[5] Cabasino, S.; Paolucci, P.S.; Todesco, G.M. Dynamic parsers and evolving grammars. ACM SIGPLAN Notices, v.27, n.11, p.39-48, 1992.

[6] Rubinstein, R.S.; Shutt. J.N. Self-modifying finite automata: An introduction, Information processing letters, v.56, n.4, 24, p.185-90, 1995.

[7] José Neto, J. Adaptive automata for context-dependent languages. ACM SIGPLAN Notices, v.29, n.9, p.115-24, 1994.

[8] Iwai, M. K. Um formalismo gramatical adaptativo para linguagens dependentes de contexto. SãoPaulo 2000, 191p. Doctoral Thesis. Escola Politécnica, Universidade de SãoPaulo.[In Portuguese]

[9] Pereira, J.C.D; José Neto, J. Um ambiente de desenvolvimento de reconhecedores sintáticos baseados em autômatos adaptativos. II Brazilian Symposium on Programming Languages (SBLP'97), 3-5 September 1997, Institute of Computing, State University of Campinas, Campinas, SP, Brazil, pp.139- 50.[In Portuguese]

[10] Christiansen, H. A survey of adaptable grammars. ACM SIGPLAN Notices, v.25, n.11, p.33-44, 1990.

[11] Wijngaarden, A.V., et al, Revised report on the Algorithmic Language Algol 68, Acta Informatica, v.5, n.1-3, p.1-236, 1975.

[12] Almeida Junior, J.R. STAD - Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos. SãoPaulo 1995, 202p. Doctoral Thesis. Escola Politécnica, Universidade de SãoPaulo. [In Portuguese]

[13] Santos, J.M.N. Um formalismo adaptativo com mecanismos de sincronização para aplicações concorrentes. SãoPaulo, 1997, 98p. M.Sc. Dissertation Escola Politécnica, Universidade de SãoPaulo.[In Portuguese]

[14] Basseto, B. A.; José Neto, J. A stochastic musical composer based on adaptive algorithms. Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação. SBC-99 PUC-Rio, Vol 3, pp105-13, 19 a 23 de julho de 1999.

[15] José Neto, J.; Iwai, M.K. Adaptive automata for syntax learning. XXIV Conferencia Latinoamericana de Informática CLEI'98, Quito - Ecuador, Centro Latinoamericano de Estudios em Informatica, Pontificia Universidad Católica Del Ecuador, tomo 1, pp.135-146. 19 a 23 de Outubro de 1998.

Interaction in Educational Collaborative Virtual Environments[1]

Mario M. Kubo[2]
Romero Tori[2]
Cláudio Kirner[3]
{mario.kubo, tori}@poli.usp.br, ckirner@iae-sp.br

[2] PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

[3] Centro Universitário Adventista
Estrada de Itapecerica, 5.859
Jardim IAE - CEP: 05858-001 - São Paulo / SP / Brasil
Phone: + 55 11 - 5822 - 6166

Abstract

The educational collaborative virtual environment supposes the active participation of students and teachers, interacting highly and aiming the knowledge exchange and creation of new abilities. The learning becomes a process that one assists the other to reach the objective, by changing experiences, dialogues, discussion of ideas, accomplishment of group, and individual activities that can be shared with the group, allowing the creation of knowledge based on the collective involvement. In this context, this paper describes and discusses the aspects and methods of interaction between students and teachers in the educational collaborative virtual environments and presents an application based on the Virtual Teacher Project.

1. Introduction

The new educational trends make the limitations, still imposed nowadays by hardware and software and the individual difficulties yet persistent when manipulating computing systems, overcame by development of systems more and more adequated to the collaborative systems needs, particularly those directioned to education. The research of new ways for distance interaction, result valuation and users motivation is necessary.

At present, several enterprises, institutions and research centers devise the development of many applications destinated to education. These applications can be found in most variated forms and styles. A common point among them is the development of interfaces capable to support a relationship between teachers and students. In such application, that relationship can be done through synchronous and assynchronous communication. An assynchronous communication is that one which has no need of information exchange on real time, like when we use electronic mail or take part on a discussion list, whereas a synchronous communication is that one which needs information exchange on real time, like video and sound conferences and textual chats. Once the interaction

[1] *Kubo, M. M.; Tori, Romero; Kirner, Claudio. Interaction in collaborative educational environments. CyberPsycology & Behavior: Larchmont, NY, USA, v. 5, n. 5, p. 399-408, October 2002.*

between student and teacher regard to technology is extremely important to make the application a successful one, Virtual Reality raises as a new human – machine interaction way.

Virtual Reality integrated to collaborative systems originates what we call collaborative virtual environments. Making use of collaborative virtual environments theory, educational systems can be created by three-dimensional modeling of a multi user environment generated by computer, representing the environment to be shared between students and teachers (i.e. laboratories, study rooms or meetings).

The collaborative virtual environments, when used as interface on educational systems, are able to provide the following characteristics:

- Visualization of the information by its three-dimensional representation;
- Presentation of a large amount of information during a simulation;
- Support to several conventional and not conventional mechanisms;
- Naturalness interaction mechanisms between students and teachers;
- Simplicity to notice activities from others students and teachers contained at the virtual environment;
- The different tools composing the system can be represented at the virtual environment and used concurrently or alternately by students and teachers.

The goal of this paper is to describe teachers and students representation at the "Virtual Teacher Project" educational collaborative virtual environment [1] [2] [3].

The work is organized as the following description. Section 2 describes concepts about Virtual Reality and Education; Section 3 shows the Virtual Teacher Project; Section 4 describes representation forms of students and teachers at the educational collaborative virtual environments; Section 5 presents students and teachers aspects; Section 6 illustrates ways of interaction between students and teachers; Section 6 introduces one Virtual Teacher Project aplication developed; finally, Section 8 presents the conclusions.

2. Virtual Reality and Education

According to Kirner & Pinho [4], Virtual Reality can be seen as the most graphical Human-Computer Interaction evolved form available until now.

Virtual Reality comprises advanced interface technologies that allow user make immersion, navigation and interaction on a synthetic three-dimensional environment generated by computer, using multi sensorial channels [4] [5]. This new interface can actuate over two different approaches: on the users' motions and acts analysis, like a traditional interface, or provoking sensations on the user, as an answer to his actions, actuating on vision, hearing and tact.

A Virtual Reality important advantage over other Human-Computer Interaction forms is that the environment can be visualized from any angle and in real time. Behaviours and attributes can be assigned to environment objects, what propitiate answers and functions simulations of the real world. To support this kind of interaction, users are provided with not conventional devices, such as visualization and control helmets, gloves, spaceball and joystick. These devices give to the user a feeling that the application is working at the real three-dimensional environment, allowing environment exploration and natural handling of objects – for example, point, catch, drag and rotate [6] [7]. Another advantage of this kind of interface is that the intuitive user knowledgment about the physical world can be transferred to manipulate virtual world.

A Virtual Reality system wraps studies and resources linked to perception, hardware, software, Human-Computer Interaction human factors and applications. To build Virtual Reality systems is necessary to have some mastery on: not conventional I/O devices, high performance computers and good graphical capacity, parallel and distributed systems, three-dimensional geometrical modeling, real time simulation, navigation, collision detection, evaluation, social impact, interface design, and simple and distributed applications over several fields [4].

The virtual reality gives to educators a new way to teach with effectiveness and, to students, a strong motivation. The most obvious reason, as Byme claims [8], is that the virtual reality is a new and different way that lets people do things they couldn't do in our real world [9].

According to Souza [10] , we can assure that, virtual reality has the potential to modifying shape as people learn, lieing in the fact that it lets students explore the environments, processes or objects, not by means of books, pictures, movies, but by means of manipulation and analysis of its own aim of study. What lets students learn about a subject put inside its own context of its subject, and receive, in each action they take, a refreshment of such context.

According to Pantelidis [11], there are several reasons for the usage of a virtual reality in Education, which are:

- It gives much more motivation to the users;
- It has an illustration power for some processes and objects which is stronger than other types of media;
- It gives the objects not only a close analysis, but also a far one;
- It lets handicapped people do certain things that are not possible otherwise;
- It gives opportunities for understanding based in new perspectives;
- It lets the learner develop his work in his own time;
- It lets the learner proceed through an experience during a period of time that is not limited by the period of a regular student;

According to Stuart & Thomas [12] , the applications of the virtual reality can prepare students in order to:

- Explore existing places and things that students couldn't have access otherwise;
- Explore real things that, without alterations of the scale in the size and time, they could not be examined indeed;
- Create places and things with a natural or changed quality;
- Interact with people who are in remote places;
- Interact with people in a non-realistic way;
- Interact with virtual beings, such as representations of historical people;
- Create and manipulate abstract conceptual representations, such as data structure and mathematic functions.

3. Virtual Teacher Project

Virtual Teacher Project [1] [2] [3] has as main goal the development of a computing system directioned to Distance Teaching. The system shall allow interaction between teachers and students. Teachers and students will be virtually transported to a classroom or any other virtual environment that will be able to aim teachers and students offering to them the support of Distributed Systems and Virtual Reality technicals. Moreover, system will offer support to teachers, like virtual laboratories and study rooms. These support features represent a tool set that will not only help the teacher, but also allow the making of several experiences.

This project intends to spread knowledge and guidance to a higher number of persons, by integrating Distance Teaching and Virtual Reality fields. With the use of Virtual Reality technicals, it wishes create a low cost system that will be able to offer support to education all over the country. The project also intends to introduce and spread Virtual Reality potentiality as an educational support tool.

Virtual Teacher Project has been subdivided among three modules:

- Human-Computer Interface – it's related with students and teachers behaviour, based on their actions among themselves and with the equipments. The interface is responsible by mapping users actions with input devices, application processing and results presentation at the output devices;
- Remote Interaction – based on analysis and definition of interaction forms to naturalness communication between students and teachers.
- Distributed System Support – analysis and implementation of data distribution mechanisms between students and teachers at the system, having in view the several computer resources.

Figure 1. Avatar: Graphical representation of Students and Teachers

3.1. Avatar: Students and Teachers Representation

An avatar (Figure 1) is the representation of a user at the virtual environment. It may be a common user (student or teacher) or an automated entity that represents a process – limitated entity which executes actions inside a virtual world. The avatars at Virtual Teacher Project have the freedom of modifying objects and parameters and send messages to any other entity that belongs to the virtual environment (audio, video, text and image connections).

3.2. Teachers and Stundents' Aspects at the Educational Collaborative Virtual Environment

Figure 2 shows the relationship between students and teachers at the educational collaborative virtual environment, by using the Use Case Diagram – Unified Modelling Language (UML) [13]. At the educational collaborative virtual environment, students and teachers' roles suffer some variation when comparing with the traditional educational system.

Students are no more under teacher's dependence, they assume a more participative and interactive role with the knowledgment they're seeking for. This knowledgment is built in a gradative way through individual and group performance. Students learn, interacting with one anothers, even with the teacher, being able in this way to actively take part on the learning process. The experience and previously acquired knowledgment exchange enlarges the content proposed by teachers, making possible overcome any difficulty that some member of the group can eventually feel, when operating the system and increasing tool usage, by suggestions that should be sent by the participants of the work groups.

Teacher assumes an incentive role on the group performance, executing actions like: identify difficulties in common and suggest new approaches, offer guidance on the solution of the most difficult problems, provoke discussions and reasoning among the group members,

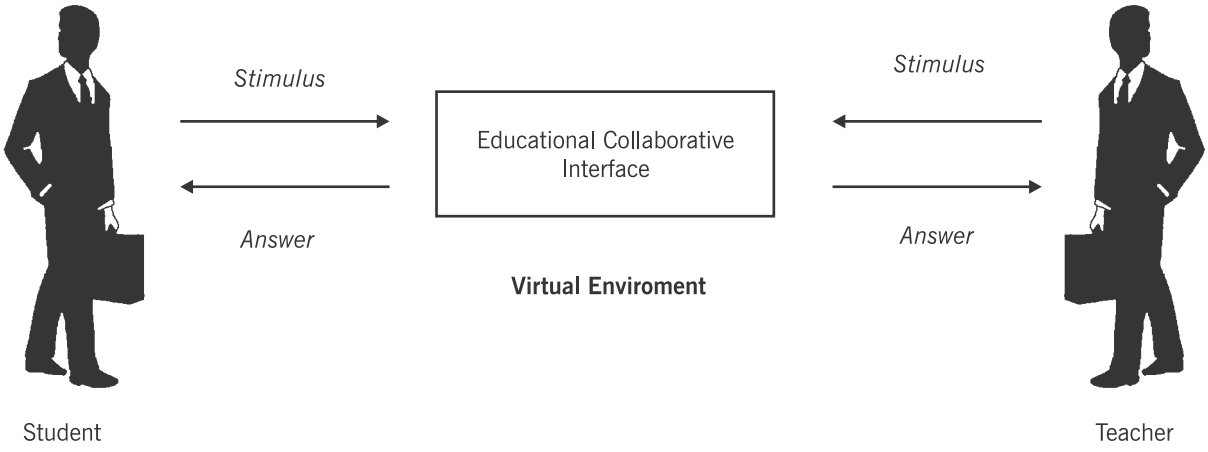


Figure 2. Virtual Environment Diagram Level 1

propose new paradigms, and during the most part of the time, take part of the activities as he/she was just a student. The teacher, due the high interaction grade among students, will became almost a member of the group, learning with the development of experiences.

The activity in groups at the educational collaborative virtual environment will break the concept of traditional work groups, where every member has to accomplish his/her own task, and will become a moment of integration, of ideas and experiences sharing, where every member will contribute to the learning process of the entire group.

At Virtual Teacher Project there are several possible situations. To make the learning process active and reach all the different learning styles, teachers and students' roles have to be revised: the teacher should have a posture different from the traditional one, he/she should exert himself/herself to offer as many different stimuluses as possible to help students in building their own knowledgment. In another words, learning will have its basis badsed on active participation, instead of a simple and pure transmission of a "ready" knowledge.

The student is the focus at Virtual Professor Project. He/she has his/her own characteristics kept, making possible a group forming within a context more adapted to its participants. This offers a richer environment, which favors the exchange of experiences and collaboration, feedbacking the process. The Virtual Teacher Project is able to contribute in offering new ways to support interaction among these groups, which will create their own identities.

Such communication forms intend to motivate students in adopting a more active, independent and responsible posture, allowing the improvement not only of oral and written communication skills, but also of research skill and ethical behaviour, keeping the students' individual style. By the other side, in face of these personal differences, it's necessary to considerate that an outrage of freedom and flexibility can lead to negligent, inconsequent, irresponsible or systematically anonymous postures. Groups' interdependency possibility also shows analogous positive and negative characteristics.

4. Students and Teachers Interaction

According Bishop et al. [14] and Kirner & Pinho [4], interaction between students and teachers in virtual environment regarding to real world have its basis based on usage of conventional or not conventional devices, communication process and technological mediation mechanisms. The general representation of interaction at a virtual environment can be seen at Figure 3.

Interaction in virtual environments can occur in two ways: single user and multi user. When in single user way, the

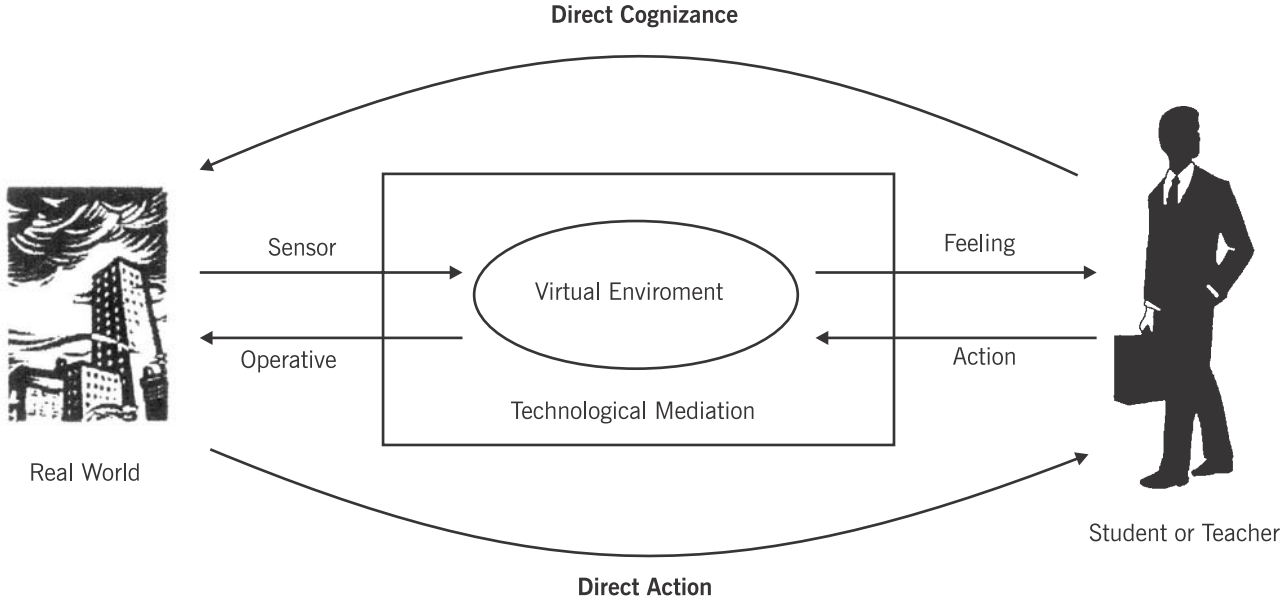


Figure 3. General Representation of Interaction at a Virtual Environment

student or teacher takes part alone on the system, whereas in multi user way, a plenty of teachers and students can participate on the same application.

Single user interaction may be defined as one of these four forms: viewer, with real participation, with simulated participation and without participation or possible supervision. Most times, the virtual environment represents a real world, except when talking about simulated participation, when the virtual environment can also be an imaginary one. At the Viewer case, we've got, for example, some particular situation of telepresence system that only makes inspections; the real participation case is a typical example of telepresence system, the simulated participation is a Virtual Reality case; and the last one (without participion) can have as example a robot with supervision skill.

Multi user interaction among several students and teachers may occur as one of the following three forms: communication between students and teachers; virtual environment sharing; and cooperative work made at the real world through the shared virtual world, according Figure 4.

In order to definy how teachers should relate with students, and vice-versa, using the collaborative virtual environment, we should analyze this by the following three points of view [2] [15].

- Cognitive: by cognitive point of view, the designer must observe how teachers and students think, learn and which are their skills in acquiring knowledgment;

- Physics: it analyzes how teachers and students look like (height, age, gender) and which sensorial and motor skills they've got. From this analysis, we can define the most convincent sensors and devices to apply on project.

- Social: it observes teacher or student's relashionship when he/she is at his/her social environment, including group structure and dynamic, power, politics, and which social skills he/she has. With this kind of observation,

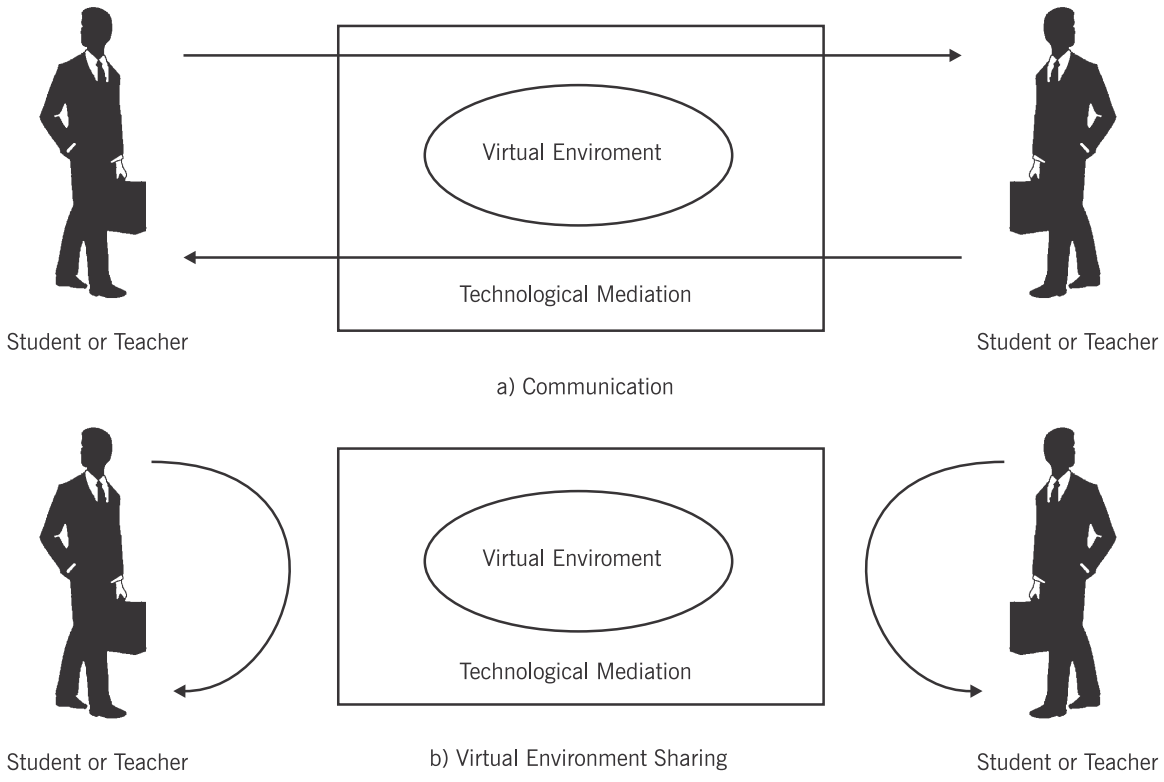


Fig 4. Multi user interaction

it's possible to create an environment control hierarchy, meaning that it's possible to know which kind of rights should exist within the environment, and how they'll be distributed among teachers and students.

Interaction study on Virtual Teacher Project is related to the necessary interaction forms analysis and definition in such a way that the system will be able to provide a naturalness communication between teachers and students. To acquire this goal, teachers and students' behaviour should be studied within a virtual environment.

According Hoffman & Mackin [16] and Martins et. al. [17], physical separation between teachers and students results in changes at the learning environment, generating others forms of relationships, such as student/interface, student/content, student/teacher and student/student interactions.

These interactions need to be used and known in an appropriate way in order to generate high quality and interactive courses on distance. Virtual Teacher Project is based on interactions definied by Hoffman & Mackin Martins et. al., described as it follows:

- Student/Interface interaction: it provides an access that allows apprentices receive not only training, but taking part of it. The apprentice/interface is the vital line between teacher and student; if it fails, training also fails. It's necessary make the technology as most transparent and friendly as possible;

- Student/Content interaction: it's called intellectual interaction. Here the student's understanding, perception and cognitive structures are transformed;

- Student/Teacher interaction: it determinates information flow from teacher to student or vice-versa;

- Student/Student interaction: it offers to students an opportunity of expand and applicate the lessons content in another way.

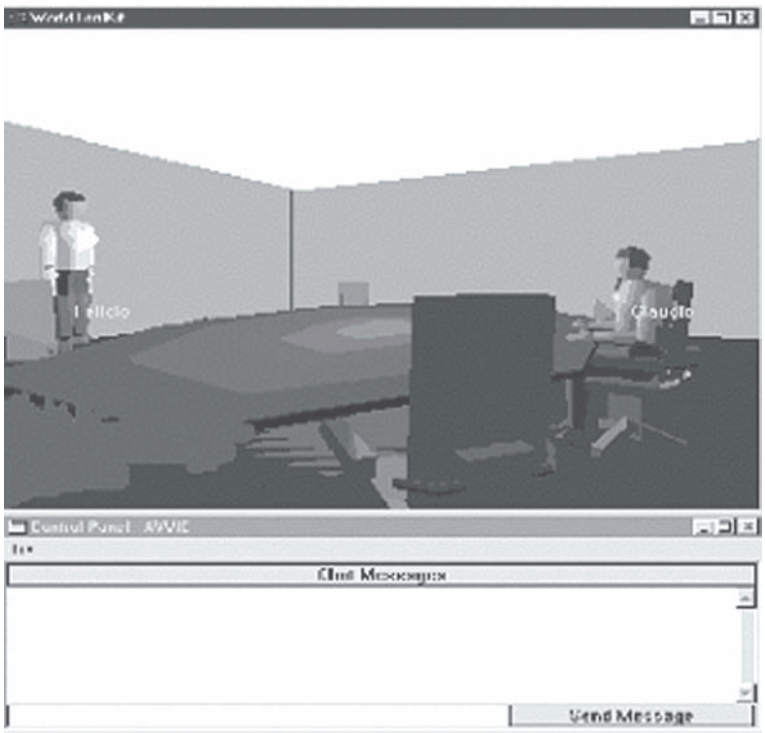


Figure 5. Educational Collaborative Virtual Environment

5. Educational Interactive And Collaborative Virtual Environment

Virtual Teacher Project aims to make a collaborative virtual environment, capable to support visualization and animation in three-dimensional spaces on real time. In this virtual environment, every computer receives a space description, detailing its objects as well the simulation process to be used. Each teacher or student's interaction (movements and actions) can be visualized by the remote students and teachers. This attendance is made by diffusing movements and actions to the others system machines, which will renderize images locally. This allows the use of wide area networks, based on the low traffic among computers.

The developed application is a virtual meeting room, where the several participants are put together around a table to talk about a common subject (Figure 5). Every participant is represented by an avatar (three-dimensional representation) who is able to move freely through the virtual environment.

The participant may use a Virtual Reality head mounted display or simply a monitor to visualize the virtual room; his/her moves are controlled by keyboard or mouse functions.

The user will be able to move his/her visualization point according his/her head usual movement (up, down, left and right) and he/she won't have any position restriction, being able to get up from chair and walk over the room. As it's usual in any collaborative environment, all participants see the avatar movement.

The tools used to build, synthetize and make the interactions of Virtual Teacher Project application were: WorldUp Modeler [18] that is a tool used to create and edit three-dimensional objects, which compose the virtual environment; WorldToolKit [19] that composes a function set used to make three-dimensional simulations and Virtual Reality applications; and WorldToWorld [20], that makes easier the development of multi user applications when integrated to WordToolKit and WorldUp Modeler.

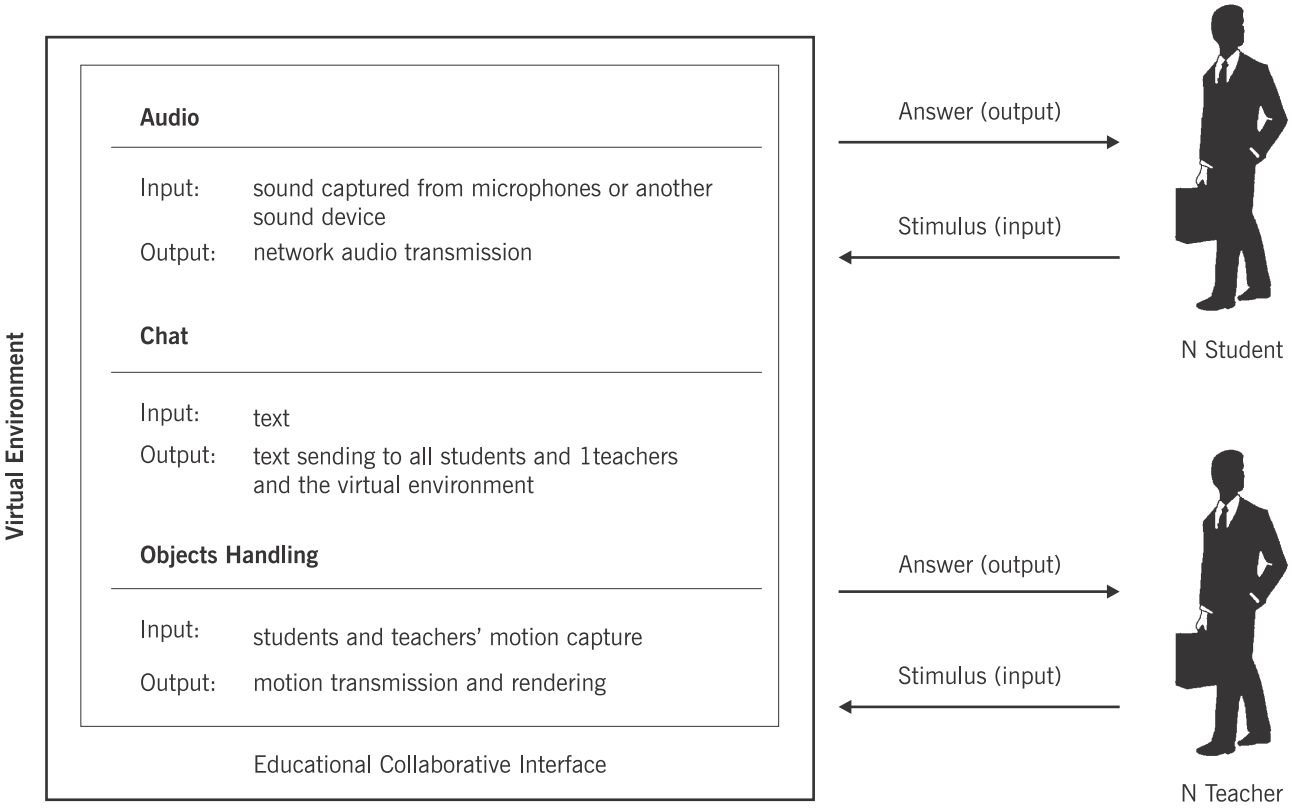


Figure 6. Communication Mechanisms Among Students and Teachers

Virtual Teacher Project is composed by the following features: communication between students and teachers, as showed at Figure 6; audio communication that aim for allowing real time voice transmisson among all students and teachers at the virtual environment and media transmission while navigating at the environment; text communication, called chat, that makes data communication by texts sent among the several students or teachers'application, having as a goal to make easier the information exchange between the participants, like messages, formulas, values, symbols and etc. Written messages will be automatically sent to all participants, and communications generated from student or teacher (avatar's) are transferred inside the virtual environment. Student and teacher are able to handle, change, create and remove objects from virtual environment.

6. Conclusion

Nowadays we can see Virtual Reality technology participating actively on several fields. In a special way, it's emphasized the usage of Virtual Reality technology on collaborative teaching. This kind of teaching aim for developing, through interpersonal activities, different situations that should favour the formation of new situations, ideas and solutions. Virtual Teacher Project allows making the collaborative teaching feasible with the use of Virtual Reality. Synchronous and assynchronous communications can be used to develop many activities, such as students' attendance, valuations, discussions, exhibitions, references etc. These actions can favour team spirit formation (when used in a convenient way), what is needed to make the educational collaborative virtual environment successful.

It was also discussed teacher and student's role at collaborative teaching. This kind of teaching is embedded in constructivism, where a student learns what he/she himself/herself builds and deduces. Virtual Teacher Project at this point is a very attractive tool, as it favours group formation, where student participates as an active element. Such participation can be done in both synchronous and assynchronous ways. It's good to emphasise that the interdependency of the group elements can be and should be explored (specially through the use of synchrounous tools) intending to create more diversified and real situations, with several solutions. Although this interdependency should be explored, every one's independency must be kept. In this way, the independency of the members to seek for different knowledgment sources, as well to think alone about some subjects, can lead to a growth not only personnal, but of the entire group, once that new ideas feed communication among group members. Teacher's role in a team is the same of a mentor, capable to valuate the student and promote his/her integration and interest. Teacher can use the several communication tools to promote these goals, considerating the profile of every student (i.e., maybe should be interesting communicate in private with a student through e-mail to not highlight some difficult that this student may have). Moreover, teacher also can play the role of a team member and, through his/her large experience, promote discussions and propose situations and even solutions.

This work is related to the continuous Project AVVIC [21], being developed by the Virtual Reality Team (GRV).

References

[1] Kubo, M.M., Deriggi, F., Kirner, C. (1999). A Model of Distributed Virtual Environment to Distance Education. International Conference on Engineering and Computing Education – ICECE'99/IEEE. Rio de Janeiro/ Brazil. 5pp.

[2] Kubo, M.M. (2000). Suporte de Comunicação para Sistemas Colaborativos Interoperáveis de Realidade Virtual. Dissertação de Mestrado. São Carlos – Brazil. 189pp.

[3] Kirner, C. (1998). Virtual Teacher Project. http:// www.dc.ufscar.br/ ~grv/pvirtual.htm.

[4] Kirner, C., e Pinho, M. (1996). Introdução à Realidade Virtual. Minicurso JAI/SBC. Recife – Brazil.

[5] Kalawsky, R. S. (1993). The Science of Virtual Reality an Virtual Environment. Addison-Wesley.

[6] Stuart, R., "The Design of Virtual Environmets", Fairfield, Pennsylvania, McGraw-Hill, 1996.

[7] Vince, J. (1995). Virtual Reality Systems. Reading. Massachusetts. Addison-Wesley.

[8] Byrne, C. M. (1995). The Use of Virtual Reality as Educational Tool. Washington University.

[9] Kalawsky, R. S. (1997). Exploring Virtual Reality Techniques in Education and Training: Technological Issues. Advanced VR Research Centre.

[10] Souza, P. C. (1997). Sistemas de Autoria para a Construção de Adventures Educacionais em Realidade Virtual. Masters project. UFSC. Florianopolis/SC - Brazil.

[11] Pantelidis, V. S. (1995). Reasons to Use Virtual Reality in Education. Documento On-line - URL: http:// eastnet.educ.ecu.edu/vr/ reas.html.

[12] Stuart, R., Thomas, J. C. (1991). The Implications of Education in Cyberspace. Multimedia Review, Summer, 2, pp. 17-27.

[13] D'Souza, D. F., Wills, A. C. (1998). Objects, Components and Frameworks with UML: The Catalysis Approach. Addison–Wesley. Published November, 785 pp.

[14] Bishop, G. et al. (1992). Reseach Directions in Vu Environments, Computer Graphics. ACM, 26(3):153–177.

[15] Vicentin, V. J., Kubo, M. M., Dizeró, W. J., Kirner, C. (1999). Sistemas Cooperativos de Realidade Virtual Usando Java e VRML. XXV Congresso Latino–Americano de Informática – CLEI'99. Assunção – Paraguai. 1085–1096 pp.

[16] Hoffman, J., Mackin, D. (1996). Intercative Television Course Design: Michael Moore's Learner Interaction Model, from the Classroom to Interactive Television. Internacional Distance Learning Conference (IDLCON). Washigton – DC.

[17] Martins, J. G., Rodriguez, A. M., Moço, S. S., Barcia, R. M. (1999). A Transformação do Ensino através do Uso da Tecnologia na Educação. V Workshop de Informática na Escola / XIX Congresso Nacional da Sociedade Brasileira de Computação – WIE'99/SBC'99. Rio de Janeiro, Brazil.

[18] Sense8 TM Corporation (1996). WorldUpTM – Reference Manual Release 4. Mill Valley – CA. 465pp.

[19] Sense8 TM Corporation (1997). WorldToolKitTM – Reference Manual Release 8. Mill Valley – CA. 888pp.

[20] Sense TM Corporation (1997). WorldToWorld Reference Manual. Mill Valley – CA. 95pp.

[21] Kirner, C. (1996). AVVIC – Virtual Environment for Shared Interactive Visualization. PROTEM Fase III – CNPq. URL: http://www.dc.ufscar.br/ ~grv

Previewing Air Traffic Conflict: System Modeling by Hybrid Automata

Ítalo Romani de Oliveira
Paulo Sérgio Cugnasca
{italo.oliveira, paulo.cugnasca}@poli.usp.br

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

Abstract

The air traffic control system has a demand for tools for aiding the human controller to take actions over an increasing number of simultaneous aircraft. A model to automatically verify safety of actions taken is presented here. The formalism used is hybrid automata, because the system has a combination of continuous and discrete variables. An example of conflict in route confluence is shown.

Keywords: *Hybrid Automaton, Air Traffic Control, Safety, Route, Conflict*

1. Overview

A great demand of navigation and traffic management support tools arises with the global implementation of the CNS/ATM (Communication, Navigation, Surveillance / Air Traffic Management) System, that represents a generalized upgrade in the technology of systems used in air transportation, with deadline in a few years. One of the main goals of the CNS/ATM is to optimize the system capacity, with respect to the arrivals and departures per minute rate, at the air transport terminals. An important bottleneck of this system is the number of simultaneous aircraft under the responsibility of a single ground controller. This number might be considerably increased or, at least, be compatible with a greater safety level, if the operator could use automatic previewing and verification tools.

System modeling is great challenge in the construction of such tools. This happens because every unsafe condition must be stated with precision. Because of it, the Hybrid Systems Theory seems to suit very well to describe and solve the problem. The terms from air traffic management are described in the appendix.

2. Hybrid Systems Theory

Hybrid Systems are systems that present a *discrete* behavior in some aspects, and a *continuous* one, in some others aspects. As an example of a hybrid system, we have an aircraft that may be on ground or flying. When it is on ground, its dynamic behavior follows a given

differential equations set; when it is flying, this set is distinct. In the same sense, the commands needed to maneuver an aircraft on flight are different from those needed on the ground. Because of this, it is said that an aircraft possesses different *modes* of operation, although, inside each mode, its behavior and its actuators follow a continuous differential behavior.

Several authors have proposed the application of Hybrid Systems on ATM modeling and similar areas. Some of the principal works are:

- ***Conflict resolution:*** Synthesis and verification of automatic air traffic conflict detection and resolution. See [1], [2] and [3].
- ***Sea Traffic:*** Verification of a sea traffic management model of harbor zones. See [4].
- ***Strings of Vehicles:*** Modeling a network automatic driver system to streams of vehicles in traffic ways. See [5].
- ***High Level Analysis of ATM:*** Systemic approach to the ATM system, being considered at its set of objects: ground control, airport, communications network, control software, aircraft, etc. See [6] and [7].
- ***Path Planning***: A calculation method that, given the initial and final points, determines a sub-optimal trajectory between them. See [8].

The *Theory of Hybrid Automata* is used to model a hybrid system. Given the system we hope to model, a *hybrid automaton* is built to each object or subsystem, so that each of its nodes represents an *operation mode*, or *location*, of that object or subsystem. For each location, its *validity conditions* are established, as well as its *flow differential equations*. To the set of nodes the possible *transitions* between them are defined, and so are the *pre* and *post conditions of each transition*.

Once the automaton is built, the unsafe states must be identified. Then, some algorithm performs the reachability analysis of these states.

3. Air Traffic Conflict

This work has a simple model of air traffic conflict problem in terminal areas and surroundings, specifically in the case of sequencing approach procedures.

Air traffic conflict is the situation where an aircraft invades the virtual cylinder of safe distance of another one. See Fig. 1.

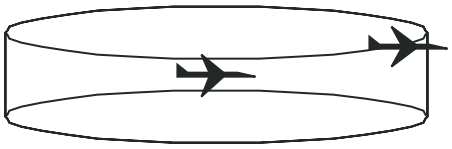


Fig. 1. Two aircraft at conflict situation.

The cylinder is attached to the aircraft on its center. The dimensions of this cylinder depends on the class of air space, and the class of flight being executed. When a conflict occurs, it is defined that the implicated aircraft are in an *unsafe situation*, because the risk of collision is no longer acceptable. This situation must be changed in order to achieve a safe state, where a correct distance is kept, holding the risk of collision at a low level. In the ATM context, there are two classes of scenarios related to conflict:

a) ***Conflict preview:*** given the actual conditions of a specific region of the air space, it is deduced that in a certain time interval a conflict will occur.

b) ***Conflict:*** two or more aircraft are already in a conflict situation.

And, respectively, two classes of actions to hold safety:

a) ***Anticipated conflict resolution:*** starting from a conflict preview, the controller executes the necessary actions to avoid the conflict.

b) ***Conflict resolution:*** as soon the effective conflict situation is detected, the controller takes the necessary actions to escape from it.

A safe traffic management requires pro-activity, which means that only the previewed scenarios and actions are considered ordinary.

The agents of conflict resolution are the ground traffic controller and the pilots, but the decision is taken by the controller, because she or he has more information available. The controllers are generally classified into two

There is not a formalized general method to preview conflict and to solve it beforehand. Actually, the method is based on some rules and heuristics, and on the controller's expertise that has seemed sufficiently reliable until now. Therefore, with the increase of air traffic, an overload on the controller may happen and, consequently, an increase of the risk on flights. The use of aid tools on the conflict preview and anticipated resolution might effectively contribute to raise the operational safety level.

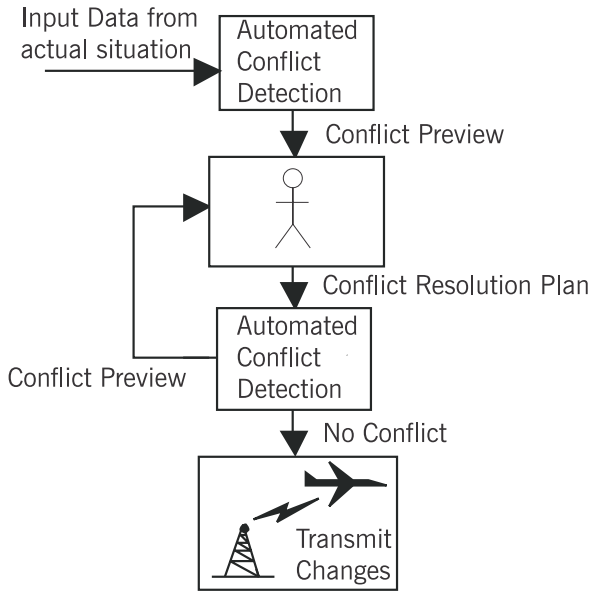


Fig. 2. Anticipated conflict resolution process

The cases below represents some important cases of potential conflict:

1. There are terminal areas that receive a traffic confluence from distinct area controls. As the coordination among the distinct ACCs is not perfectly integrated, the APPs controller frequently previews a conflict between an aircraft coming from the first ACC and an aircraft from the second, and must do its anticipated resolution in the terminal area, or close to it.

2. There are pre-defined entrances and exits at the terminal region (see Fig. 3). To each entrance, the area controller is responsible for sequencing aircraft coming from several directions, lining them along the same entrance route. At intense traffic periods, this sequencing, that is a route confluence situation, requires anticipated conflict detection and resolution.

A simple model with the hybrid automata of the aircraft and of the routes star of approach, will be presented. Each aircraft will have one automaton, and the verification will be done on the composite automaton of all aircraft.

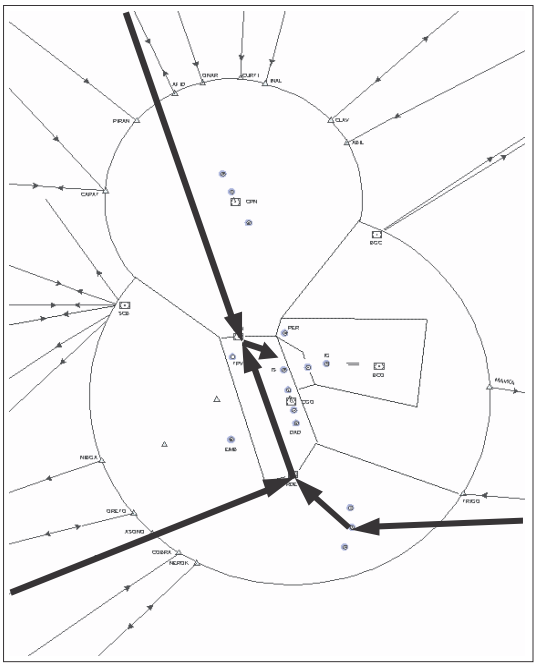


Fig. 3. Route confluence in the terminal area

4. The Model

The problem of conflict detection is modeled as follows. Both aircraft *a* and *b* in Fig. 4 are at a route confluence. The circle around each is a projection of the cylinder of safe separation. The reference points, indicated as triangles, are the reference points present on the navigation chart. A reference point is an abstract geographic point whose position is described in the chart. The aircraft must pass over each of these points at a given altitude. We assume that, given the conditions of the instant described in Fig. 4, a conflict will occur, if the speed, heading and flight level remain unchanged. Hence, the traffic controller will determine that these variables must be changed, which implies the execution of a deviation maneuver. Our objective is to verify if the safety conditions are held during this maneuver.

4.1 Routes

Lets start analyzing data and conditions of each route. Aircraft *a* and *b* follow the routes P_a and P_b, respectively, determined by the reference points R_a^i, R_b^i, that are equivalent to $i = 1, 2$. The segments S_a^i, S_b^i are the lines connecting the reference points. Each reference point belongs to a *transition window* W^i, which the aircraft must necessarily go through.

Fig. 5 represents the horizontal plan of routes. Therefore the transition windows are defined as a vertical rectangle, with defined width and height. The transition windows work as boundaries between locations (discrete states) of the automaton. In other words, each location of the automaton that is not the final state represents the spatial set $convex(W^i \cup W^{i+1})$.

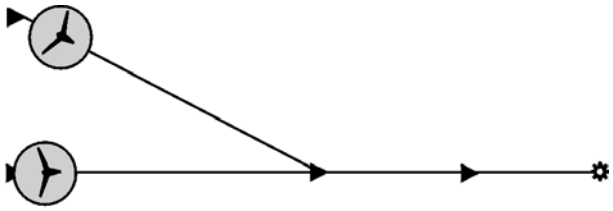


Fig. 4. Aircraft confluence of routes

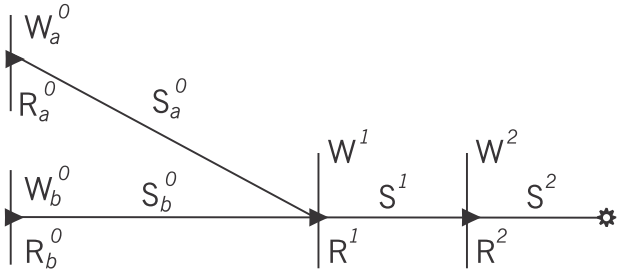


Fig. 5. Horizontal plan of routes

The dashed lines of Fig. 6 indicate $convex(W^1 \cup W^2)$, i.e. the convex hull of $W^1 \cup W^2$, a set that will be important to define one of the safety conditions. It will be also seen that a safe trajectory must go through every transition window at a given sequence.

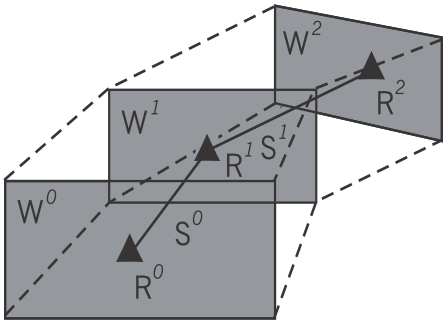


Fig. 6. Transition windows and convex hull

4.2 Aircraft

The aircraft variables, that determine its behavior, are presented at table 1.

Note on Fig. 7 that the original cylinder of safety was substituted by the parallelepiped *B*. This change was done to preserve the linearity conditions necessary to automatic verification of automata. As the original cylinder, the horizontal dimensions of the box are far bigger than the height. Other polyhedra can be used, but we preferred to use a simple one.

The safe conditions of operation are described below. The violation of any of them implies falling into an unsafe state.

1. Conflict absence, i.e., given the aircraft *a* and *b*, then $(sb \notin B_a(s_a)) \wedge (sa \notin B_b(s_b))$. This means that no aircraft invaded the safety box of another. The violation of this condition is the event that will be called *get_conflict*.

2. As the aircraft goes through W^i, then $s \in T^i$, where $T^i = convex(W^i \cup W^{i+1})$. The succession of these sets for $i = 0, 1,..., n$ results in the *tube* of navigation inside which the aircraft must hold along all its trajectory. The violation of this condition is the event called *escaping*.

3. Being the aircraft into T^i, then $a_L^i \leq a \leq a_R^i$. This defines a horizontal heading considered safe to the aircraft, according to the navigation chart. When this condition is violated, the *misaligning* event occurs.

4. Being W_n the last transition window, then the aircraft must go through it at a time lower than t_{max}. The violation of this condition is the event labeled *timeout*.

Table 1. Aircraft variables

Variable	Meaning (initial value provided)
v	Scalar speed
α	Aircraft heading.
$s = (x,y,z)$	Spatial position of the aircraft, in function of time t and speed u, from a given absolute referential.
r^i	Reset[1] of the heading at i-th window. It is a discrete variable, with the possible values 0 : without change; 1 : default heading of segment i; 2 : heading of the reference point R^{i+1}. The default heading of segment S_i is the heading of the straight line linking the points R^i and R^{i+1}.
Variable	**Meaning (initial value calculated)**
$u = (x',y',z')$	Aircraft speed. The initial value is $x = v.\sin a^0$, $y = v.\cos a^0$, $z = z'^0$. The value of z'^i is calculated from the difference between z^0 and the next programmed flight level. Being at W^i, one may determine the next programmed flight level by the following procedure: 1) If $r^i = 0$ or $r^i = 1$, then $z^{i+1} = z^i$ 2) If $r^i = 2$, then $z^{i+1} = z(R^{i+1})$
B	Safety Box of the aircraft. Given the length l, height h and width w, the aircraft will hold inside the parallelepiped B with these dimensions, which aligning of the horizontal edge is given by the heading a. This means that B is function of the speed u and the position s. This parallelepiped will always stay; vertically aligned with the Y-axis.

[1] A reset of a variable is an attribution of a new given value at a state transition of the system.

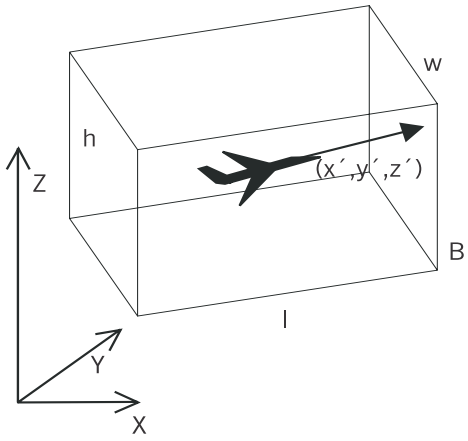


Fig. 7. Aircraft variables

4.3 System Automata

From the above definitions, the basic automaton of the system can be presented, as in Appendix B. The subscripts *a* and *b* refer to the aircraft *a* and *b*, respectively.

Aircraft *b* has a similar automaton. To the verification of event *get_conflict*, that is synchronized between the automata of both aircraft, it is necessary to verify the composite automata. Let H_a and H_b be the automata of aircraft *a* and *b*, respectively. We build the composite automata and perform the conjunct verification. The unsafe states are represented by the ones labeled ***escaped***, ***misaligned***, ***conflict*** and ***out_of_time***.

4.4 Deviation Maneuvers

The deviation maneuvers are determined by the input variables v, r_a^i e r_b^i. These variables indicate where there will or not be change of heading, according to the possible options in Table 1.

5. Execution Example

Let us observe an example of deviation maneuver verification. Initially, at instant t^0, aircraft *a* and *b* are at the positions indicated at Fig. 9, each in its safety box. Both headings are initially pointing to the point R^1. The speeds are identical. Doing the verification to these initial conditions, the result is that the system will fall into the ***conflict*** state at a given time t_c. Note at Fig. 9 that *a* invaded the safety box of *b*.

Some plausible maneuvers to avoid conflict are:

1. Hold the speeds and head *a* to the left end of W^i, and then to R^2. In this case, $r_a^1 = 2$.

2. Decrease speed of *a* and head it straight to R^2. Here, $r_a^1 = 0$

It is important to observe that the change on speed and flight level will alter the fuel consumption. Thus, the desired maneuvers should avoid or minimize extra consumption.

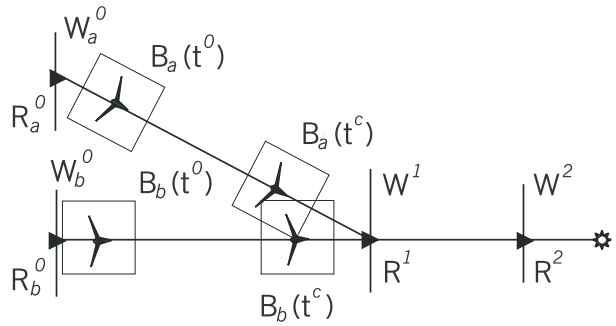


Fig. 8. Conflict situation

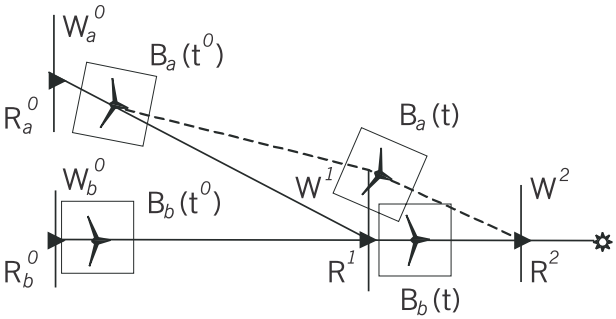


Fig. 9. Deviation maneuver without change of scalar speed

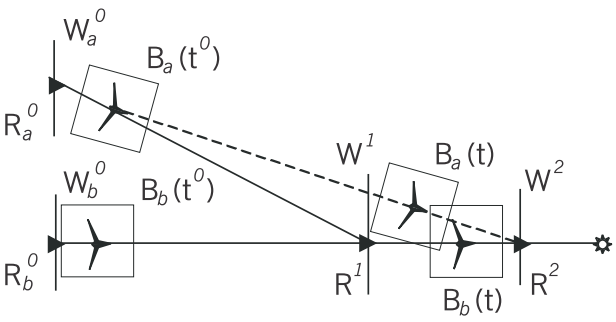


Fig. 10. Deviation maneuver with lowering speed of *a*.

6. Optimization

To performance achievements it is desirable that, once identified the class of maneuver to be performed, some performance aspect can be optimized. For example, in the second deviation maneuver shown, it is possible to find the maximum speed to the aircraft *a* that holds the safety conditions. In hybrid automata, this is done using parametric analysis.

7. Final Remarks

This model permits a great flexibility in representing routes. It approximates routes into linear segments, what is very compatible with the models used by the operators of air navigation. On the other hand, in order to make this system really operational, it will be needed to make an interface or input processor to turn the input of route data more suitable. Also, some refinements must be done, like inclusion of states representing awaiting orbits, and the possibility of changing the flight level in maneuvers.

It would be of high utility to conduct a research on the complexity of the problem and practical experiments of verification of systems with many aircraft.

References

[1] Bonifácio, A. L.: Verificação e Síntese de Sistemas Híbridos, MSc. Dissertation, Instituto de Computação Unicamp, 2000.

[2] Tomlin, C.; Lygeros, J.; Sastry, S.: Synthesizing Controllers for Non-linear Hybrid Systems. LNCS 1386, Springer-Verlag, 1998.

[3] Tomlin, C.; Lygeros, J.; Sastry S.: Computing Controllers for Non-linear Hybrid Systems. LNCS 1569, Springer-Verlag, 1999.

[4] GodHavn, J. M.; Lauvdal, T.; Egeland, O.: Hybrid Control in Sea Traffic Management Systems. LNCS 1066, Springer-Verlag, 1996.

[5] Lygeros, J.; Lynch, N. A: Strings of Vehicles: Modeling and Safety Conditions. LNCS 1386, Springer-Verlag, 1998.

[6] Lynch, N.: High-Level Modeling and analysis of an Air-Traffic Management System. LNCS 1569, Springer-Verlag, 1999.

[7] Lygeros, J., Pappas, G. J.; Sastry, S.: An Approach to the Verification of the-TRACON Automation System. LNCS, Springer-Verlag, 1998.

[8] Egerstedt, M., Koo, T. J., Hoffmann, F., Sastry, S.: Path Planning and Flight Controller Scheduling for an Autonomous Helicopter. LNCS 1569, Springer-Verlag, 1999.

[9] Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.-H.,. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, Hybrid Systems, Lecture Notes in Computer Science 736, pages 209-229. Springer-Verlag, 1993.

[10] Alur, R., Henzinger, T.A., Ho, P.-H. Automatic symbolic verification of embedded systems. In Proceedings of the 14th Annual Real-time Systems Symposium, pages 2-11. IEEE Computer Society Press, 1993. Full version appears in IEEE Transactions on Software Engineering, 22(3): 181-201, 1996.

[11] Henzinger, T. A., Ho, P.-H., Wong-Toi, H.: HyTech: the next generation. In Proceedings of the 16th Annual Real-time Systems Symposium, pages 56-65. IEEE Computer Society Press, 1995.

[12] Henzinger, T. A., Ho, P.-H., Wong-Toi, H.: A user guide to HyTech. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steen, editors, TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science 1019, pages 41-71. Springer-Verlag, 1995.

[13] Henzinger, T. A., Nicollin, X., Sifakis, J., and Yovine, S. Symbolic model checking for real-time systems. Information and Computation, 111(2):193-244, 1994.

[14] Henzinger, T. A., Wong-Toi, H.: Linear phase-portrait approximations for non-linear hybrid systems. In R. Alur and T.A. Henzinger, editors, Hybrid Systems III, Lecture Notes in Computer Science. Springer-Verlag, 1995.

Appendix A: Glossary

ACC: Air Traffic Control Center. An area control center established to provide air traffic control service.

APP: Approach Control. A terminal control center established to provide air traffic control service.

ATM: Air Traffic Management.

CNS: Communication, Navigation and Surveillance.

Flight level: Band of altitude of the aircraft. The international standard assigns 100 ft for each level. So, an aircraft flying at level 150 has altitudes between 15000 and 15099 ft.

Location: In a hybrid automaton, it represents a discrete state. Every transition between locations is done by edges, graphically represented as arrows in the automaton.

Navigation Chart: Pre-defined sequence of reference points, which the aircraft must pass over. A chart determines the route that an aircraft must follow.

Heading: Angle of direction of the aircraft at the horizontal plan, measured from the North.

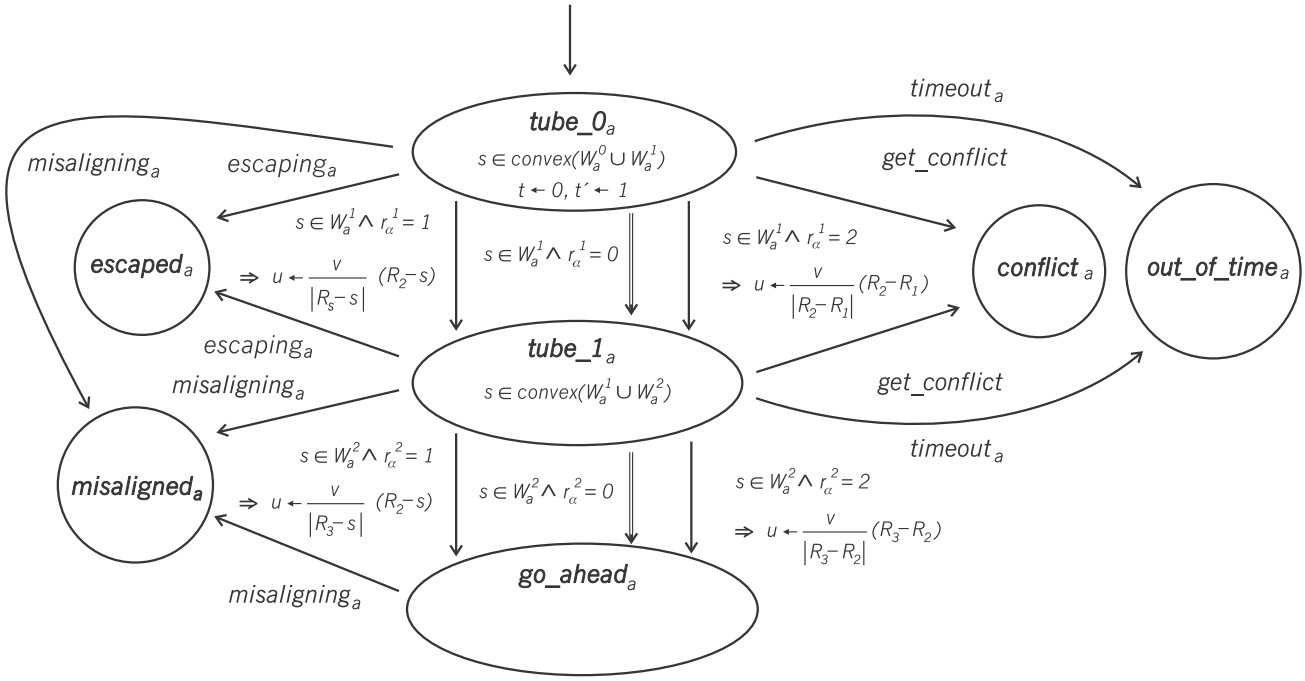
Route: Navigation tube based on a navigation chart.

Trajectory: Path performed by the aircraft. We assume in this model that the trajectories are the union of linear segments, whose vertices belong to the transition windows.

Transition window: A transition window W^i is defined as a vertical rectangle, with defined width and height. The transition windows work as boundaries between locations. In other words, each location of the automaton that is not final represents a set like *convex* ($W^i \cup W^{i+1}$), in space. A safe trajectory must go through every transition window, at a given sequence.

Appendix B: A Basic Automaton

Hybrid automaton H_a of the aircraft *a*



WebBee — a Web-based Information Network on Bees

A.M. Saraiva[1]
V.L. Imperatriz-Fonseca[2]
R.S. Cunha[1]
E.A.Cartolano-Júnior[1]
{antonio.saraiva, renato.cunha, etienne.cartolano}@poli.usp.br., vlifonse@ib.usp.br

[1] PCS Departamento de Engenharia de Computação e Sistemas Digitais
Laboratório de Automação Agrícola (Agricultural Automation Laboratory)
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C2-54
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5366

[2] DEG Departamento de Ecologia
Laboratório de Abelhas (Bee Laboratory)
IBUSP Instituto de Biociências da Universidade de São Paulo
Rua do Matão, Travessa 14, no. 321
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 7533

Abstract

There is a growing awareness on the importance of the world's biodiversity and on its fast decline due to many factors, including human activities. Among the huge number of animal and plant species on Earth, it is estimated that only 10% are known and this is one of the main obstacles for developing conservation programs. In order to increase the knowledge and understanding of that diversity, there is a worldwide effort for digitizing and integrating the information already available but geographically disperse, which will eventually result in an electronic Catalogue of Life, accessible on the Internet. Many pollinators, including bees, are among those threatened species and their decline is especially dangerous as they are crucial for maintaining global biodiversity and for crop production. This paper presents a Web-based information system that was developed to organize knowledge and facilitate sharing of information on bees. It is aimed at different audiences, from researchers to basic level students, from policy-makers to beekeepers, who need high quality information on bees. A database is a central part of the system, storing data from different investigations conducted by a network of researchers and research groups. That includes data from bee species, plant species visited by the bees, and a great deal of other information on their behavior and characteristics. It also includes data collected automatically by data loggers and by a weather station in several experiments on thermoregulation and flight activity of the colonies. Data is stored in different formats such as texts, photos and videos. The system can be accessed on the Internet with a simple Web browser and was implemented with a MySQL DBMS, Apache Web server and PHP scripts. Developed initially to hold the information from one research group, it was afterwards proposed as a platform for the integration of the information of different groups, forming an information network. Currently groups from other countries are willing to use it as their platform, configuring it as an international portal on bees.

***Keywords**: bees, information system, Internet, database, biodiversity*

1. Introduction

Bees are the main pollinators of many plant species and so they directly influence agricultural production and the environment. The use and conservation of bees as important agricultural crop pollinators was considered high-priority by the Convention on Biological Diversity [1] which, at the Conference of the Parties, in Nairobi, May 2000, approved the São Paulo Declaration on Pollinators, which identified the causes and consequences of pollinators decline [2]. The International Pollinators Initiative was formulated as a priority for Agricultural Biodiversity. In COP6 of CBD, in Haya, May 2002, a Plan of Action was developed, including natural areas as well as agroecosystems for pollinators´ protection.

Besides that, beekeeping can be considered a sustainable development activity [3] that can be very important for increasing the income of small farmers, and it is also being used for agricultural crops pollination inside greenhouse, improving the quality of the products and increasing the yield [4]. It is therefore very important to increase beekeeping of native bees for commercial purposes, but there is a lack of information on the subject to the general public.

At the same time, there is a lot of research to be done on bees, as well as on all the other species. As pointed out by Edward Wilson, who created the word “biodiversity”, currently there is a proposal and an effort for a global biodiversity census to be developed within the next 25 years. Considering that what is known so far is only 10% of the world’s species, and that this knowledge was obtained basically during the last 250 years, beginning with Carolus Linnaeus’ *Systema Naturae*, in can be inferred that the remaining 90% of the species (of the estimated 10 to 20 million species) will have be discovered and described in 10 % of the time spent until now [5].

It is already clear that information technology has an important role to play on that effort. A new discipline, Bioinformatics, or more specifically Biodiversity Informatics, is in charge of providing the tools for the task of increasing our knowledge on biodiversity, thus enabling us to conserve it. That involves a wide spectrum of systems and equipment, from instrumentation to information systems and databases, as summarized by Saraiva [6].

Taking all that into account, WebBee was proposed and developed. It comprises an instrumentation system for automatic data acquisition. It is also an information system where all those automatic data plus the information and knowledge gained throughout the years is organized and made available via Internet, facilitating the access and contributing to its dissemination.

As in many other fields, the knowledge on different species is geographically scattered through different research groups and institutions. It is necessary to integrate those many different sources of information in order to enhance the system and to cover as many species as possible. To cope with that aim, WebBee is an information network, in the sense that many different sources provide information for the system, and it is also a portal for the different research groups on bees, thus stimulating cooperative work.

2. System overview

The system architecture can be seen in Figure 1. A data acquisition system collects data from sensors installed inside the colonies at the Bee Laboratory, Instituto de Biociências, where a weather station monitors the climatic conditions at the site. Both data sets are sent via Internet to a central database server computer at the Agricultural Automation Laboratory, Escola Politécnica, where many other data items are also stored. At this point, the system is open to the Internet via a Web server, so that the researchers and the general public can access its content [7].

The data collected at each colony are air temperature and humidity inside the hive, and bee flux at the hive entrance. Together with the weather data, it allows studying the effect of abiotic conditions on the flight activity of the bees and on the temperature regulation inside the colonies.

From the point of view of the information system, WebBee is composed of three modules: Services Module, Maintenance Module and Web Module (Figure 2). The Web Module is the user interface to the system at any point in the Internet. The Maintenance Module is used by the system managers (there can be many of them, at the different partners institutions) for data input. The Service Module is the core of the system

and contains the database. Client-server architecture was used, with message exchange between client software at the Maintenance and Web Modules and server software at the Services Module.

The Services Module is based on MySQL DBMS (http://www.mysql.com) and on Apache Web server (http://www.apache.org) plus a PHP interpreter (http://www.php.net). The Maintenance Module was built with MS Visual Basic upon MS Windows environment, which is the operational system used at the Bee Laboratory. The Web Module uses any Web browser that supports HTML 4.0 and Javascript 1.2.

At the Maintenance Module, a dedicate software written in Visual Basic makes data queries to the MySQL server via an ODBC driver. The server, in turn, searches the data on the database and returns the dataset required to the client software.

At the Web Module a browser asks for PHP pages to the Web server. These are forwarded to the PHP interpreter for execution and, if they contain any database query, the interpreter will forward the query to the MySQL database server, who will return the demanded data. These data will be formatted into an HTML page by the PHP interpreter, according to the PHP page instructions and will send this new HTML page back to the Web server and then back to the browser (client).

The system is strongly based on software with free distribution for non-commercial purposes. The only exception is MS Visual Basic which, however, will be replaced by a PHP interface for the Maintenance module as discussed below.

3. Services Module: the Database

The database is the core of the Services Module. It must handle data obtained from different studies and sources, and in different formats and media. As an example, information on species taxonomy may comprise texts and images, while behavioral information may also require information in video format. Regarding the data source, most of the data available was manually collected and will be manually input to the system. However, the system is supposed to receive data acquired directly from a weather station and from instrumentation systems installed inside the colonies. [8].

The modeling process was based on the needs expressed by the experts on the application domain and on existing systems. The data items considered for SinBiota, the information system for the Biota Program of FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo – The State of São Paulo Research Foundation) [9] was one of the main references for defining species data items. Figure 3 shows an E-R (entity-relationship) diagram that succinctly presents the database structure. From the point of view of the contribution of this system to the building of a Catalogue of Life, some important data modeled were those related to bee species (taxonomy, geographic distribution, etc), their relation to plant species (plants visited, taxonomy, etc) and to the ecosystems. Other data items are present which do not find a counterpart in other systems, such as automatically acquired data and their metadata (data from the sensors, equipment, etc), since WebBee counts on an instrumentation system.

4. Maintenance Module

The Maintenance Module is the system interface for the manager, which is responsible for the input of data into the system. The main functions of the Module are to connect to the database server, to authenticate the user, to provide query mechanisms, and to format and present data.

The interface (Figure 4) is organized in windows and menus which is a well known metaphor to most users. MS Visual Basic allows the use of MDI (multiple document interface) technology, so that many (child) windows can be presented inside the client area of a main (parent) window, simultaneously and in an organized way. Other windows are presented in modal form. Related information is presented in formularies where it is possible to edit data in an integrated way. Comboboxes and listboxes were also used to facilitate data input. With the Maintenance Module, the user can edit the database content and input related files (texts, images and videos) that are stored in a directory structure instead of in the database.

In a first phase, researchers did data entry only locally, from the Bee Laboratory. However, as other groups

have joined the initiative forming the network of research groups, remote data entry became necessary. Although the same software can be used in this case, it must be installed at the computers of those users. To avoid that operation, which is a burden and sometimes can be troublesome, a new piece of software is being developed for the maintenance module. It is also based on PHP and will run on the server computer, thus avoiding the need to install any software at the client side, which will require only the browser.

5. Web Module: the Internet Operating Interface

Among the different types of information conveyed by WebBee, special emphasis is put on those related to bee species and to automatically acquired weather data.

The bee species information is presented in record cards containing images and texts specific for each species. The images cover different aspects of the body (front, side, leg and wing views) for the queen, the worker and the male, besides images of the nest and hive entrance. They can be used for the user to identify the bee species and are enhanced by texts that explain the images.

As the number of bees is very large, the information is presented with the aid of scripts that dynamically build the Web page for a species requested among a catalogue that can be sorted by popular or scientific name (Figure 5). Within the record card, the user can browse through many images, which have a resolution adequate not to hinder the access due to low speed network connections. Each record card is divided into two parts: the main part or window, which shows a selected image and its text; and the browser part, where miniature images can be seen and selected for zooming in and hence forming a new main window with its text area (Figure 6).

The numerical data from the instrumentation system is another important feature of WebBee. The variables air temperature, air relative humidity, air pressure, wind speed, rain, solar radiation and ultra-violet radiation can be selected and presented as a graphic for a selectable period of time (Figure 7), or can be exported as a table for use in spreadsheets.

6. An information network on bees: present and future

WebBee started as an information system with the aim of organizing information available at the Bee Laboratory. However, very soon it became clear that its scope should be widened for many reasons: the diverse research groups have information that complement each other; the task of providing the huge amount of information for such a system would benefit from distributed contributors being responsible for each part of the job; the problem of organizing the information for internal use and the issue of making it available to the public is common to all of them; both from a Brazilian point of view, where there is a lack of quality information in Portuguese with respect to native bees, and from a global point of view, where a global species catalogue is necessary, the integration of scattered information is essential.

For those reasons, WebBee was open to the community and was proposed as a common platform for research groups on bees in Brazil at the meeting 5º Encontro Sobre Abelhas (Fifth Meeting about Bees), in Ribeirão Preto, Brazil, in September, 2002 [10], no matter if as a centralized database or as integrated distributed databases. The domain webbee.org.br was registered so as to provide an institutionally neutral site. Three other Brazilian research groups have joined the initiative: Embrapa Amazônia Oriental, Empresa Baiana de Desenvolvimento Agrícola and State University of Feira de Santana (Bahia State), and many others are expected to do it. Recently, in South Africa, on May, 2003, at the workshop "Best practices guide on pollinators conservation for policy makers", under the auspices of ITIS – International Taxonomy Information System, researchers from many developing countries (South Africa, Kenya, Ghana, India, China, Pakistan, Nepal) showed their willingness to adopt WebBee as their system for publishing information on the Web on their native bee species. It should be stressed that those are important countries from the biodiversity point of view, and are involved in national and regional pollinators initiatives.

Therefore, the system is evolving to become a global portal on bees. To cope with this new scope and taking into account the need to integrate it to other national or global initiatives (Species link, Catalogue of life, etc) WebBee will be redesigned to allow its integration to other sources and users of information and to

distributed processing under Web Services architecture and technologies, as proposed in Saraiva [6].

7. Conclusion

With WebBee, the information collected along the development of many research projects and groups that were dispersed in many places and media was organized in a centralized database. That information is now more available and can be better used by all those involved and/or interested in bees, from researchers to students, agricultural producers and policy makers.

The database will grow as new data is generated and each researcher can input its data through a unified interface and format, which helps promote content homogeneity and consistency. Multimedia contents can be viewed by automatically launching external viewers such as Adobe Acrobat, MS-Word, MS Player, etc. Data acquired in real time by instrumentation systems and stored on the database can be remotely monitored and graphically viewed allowing interaction and collaboration between geographically distant researchers.

An important result is the possibility of dynamic updating of web pages as new data is fed into the system, dispensing the need to edit HTML pages. For instance, each new species included into the database will automatically be presented when the correspondent PHP page is loaded. This will have a significant impact on information availability without burdening the researchers with Web page preparation. To eliminate the installation of any software at the user (manager) computer, the Maintenance Module will be re-written on PHP, simplifying the access and unifying the user interface.

The increasing interest of research groups from various countries in using WebBee and the need to integrate it to other national and global initiatives will lead to a redesign of the system. Web services architecture and technologies are the choice for its evolution.

Acknowledgements

This project was partially funded by the Brazilian agencies: Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq; Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP; Financiadora de Estudos e Projetos, FINEP; and Universidade de São Paulo- Pró-Reitoria de Pesquisa (Programa SIAE).

It is a joint project of Bee Laboratory (Laboratório de Abelhas), Departamento de Ecologia, Instituto de Biociências, and Agricultural Automation Laboratory (Laboratório de Automação Agrícola), Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica, both from Universidade de São Paulo, Brazil.

References

[1] CBD. Convention on Biological Diversity. Available at: http://www.biodiv.org. Accessed: 21 jan. 2003.

[2] Kevan, P.G.; Imperatriz-Fonseca, V.L. Pollinating Bees – The conservation link between agriculture and nature. Brasília. Ministry of Environment. 2002. p.303-8.

[3] Cortopassi-Laurino M. *et al*. Stingless bee rearing as an activity for sustainable development. In: International Apicultural Congress, 37., Durban, 2001. Proceedings. Durban, South Africa, 2001. Doc.343. / CD-ROM /.

[4] Velthius, H.H.W. The historical background of the domestication of the bumble-bee, *Bombus terrestris*, and its introduction in Agriculture. In: Kevan & Imperatriz-Fonseca eds.,Pollinating bees: the conservation link between Agriculture and Nature. 2002. p. 177-184. /also available on-line at www.webbee.org.br, under IBP/

[5] Wilson, E.O. The encyclopedia of life. Trends in Ecology and Evolution. v.18, n.2, feb. 2003. p.77-80.

[6] Saraiva, A.M. Tecnologia da informação na agricultura de precisão e biodiversidade: estudos e proposta de utilização de *Web services* para desenvolvimento e integração de sistemas. 2003. 187p Tese (Livre Docência) – Escola Politécnica, Universidade de São Paulo. São

Paulo, 2003. (English title: Information technology in precision agriculture and biodiversity: studies and proposal of utilization of Web services for systems development and integration)

[7] Cunha, R.S. *et al*. An Internet-based monitoring system for behavior studies of stingless bees. In: European Conference of the European Federation for Information Technology in Agriculture, Food and the Environment, 3., Montpellier, 2001. Proceedings. Montpellier: ENSAM, 2001, p.279-284.

[8] Cunha, R.S, *et al.* WebBee - A Web-based information system for research on stingless bees. In: World Conference of Computers in Agriculture and Natural Resources, Foz do Iguaçu, 2002. Proceedings. St. Joseph: American Society of Agricultural Engineers, 2002. v.1, p.676-682.

[9] SINBIOTA. SinBiota – Sistema de Informação Ambiental para o Estado de São Paulo. Available at: http://sinbiota.cria.org.br. Accessed: 21 jan. 2003. (English title: Environmental Information System for the State of São Paulo)

[10] Saraiva, A.M.; Imperatriz-Fonseca, V.L. WebBee: Uma rede de informações sobre biodiversidade brasileira em abelhas nativas. In: Encontro sobre Abelhas, 5., Ribeirão Preto. 2002. Anais. Ribeirão Preto: FFCLRP/FMRP/USP, 2002. p.108-113. (English title: WebBee: an information network on Brazilian biodiversity on stingless bees)

Figures

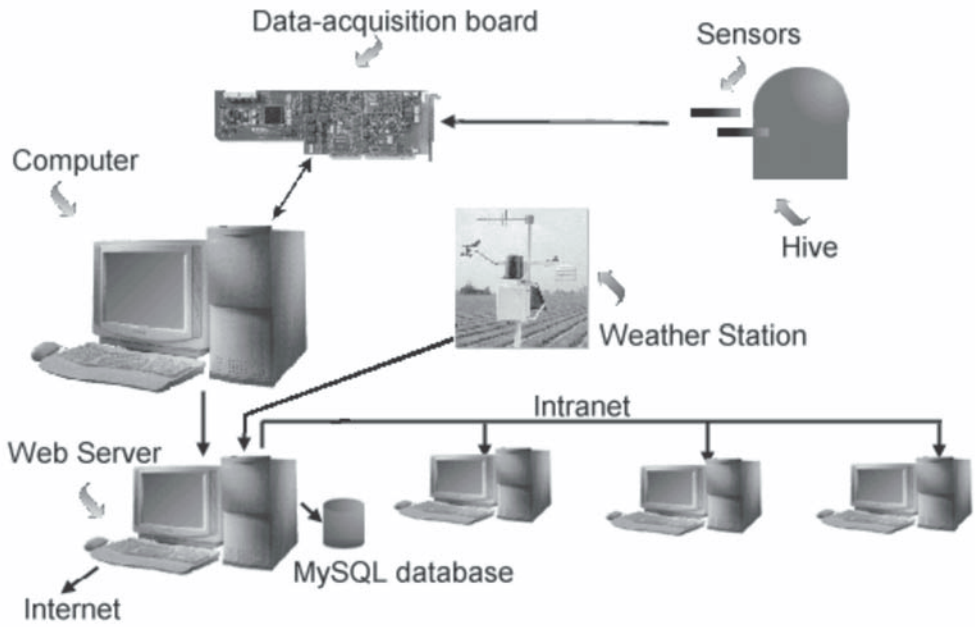


Figure 1. System architecture

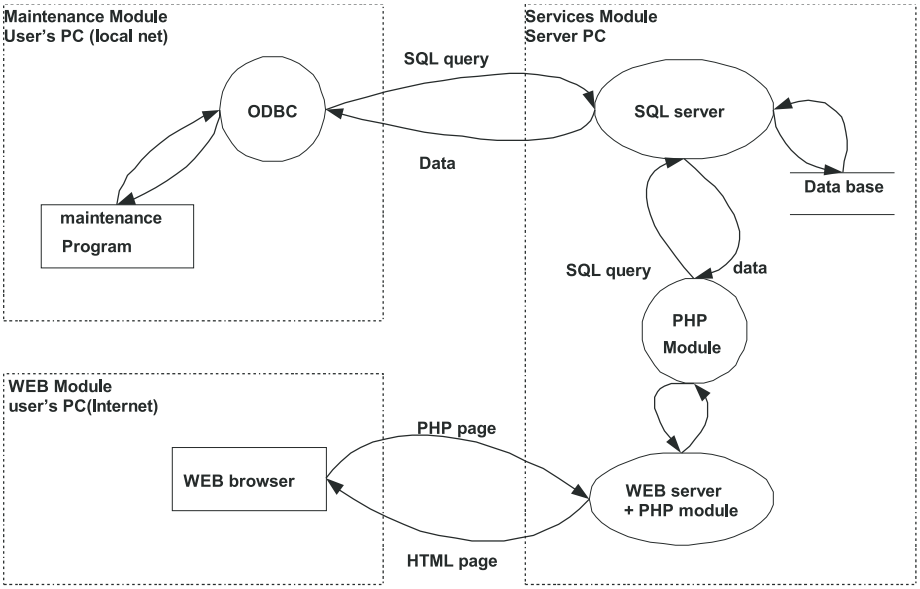


Figure 2. Software architecture: three modules

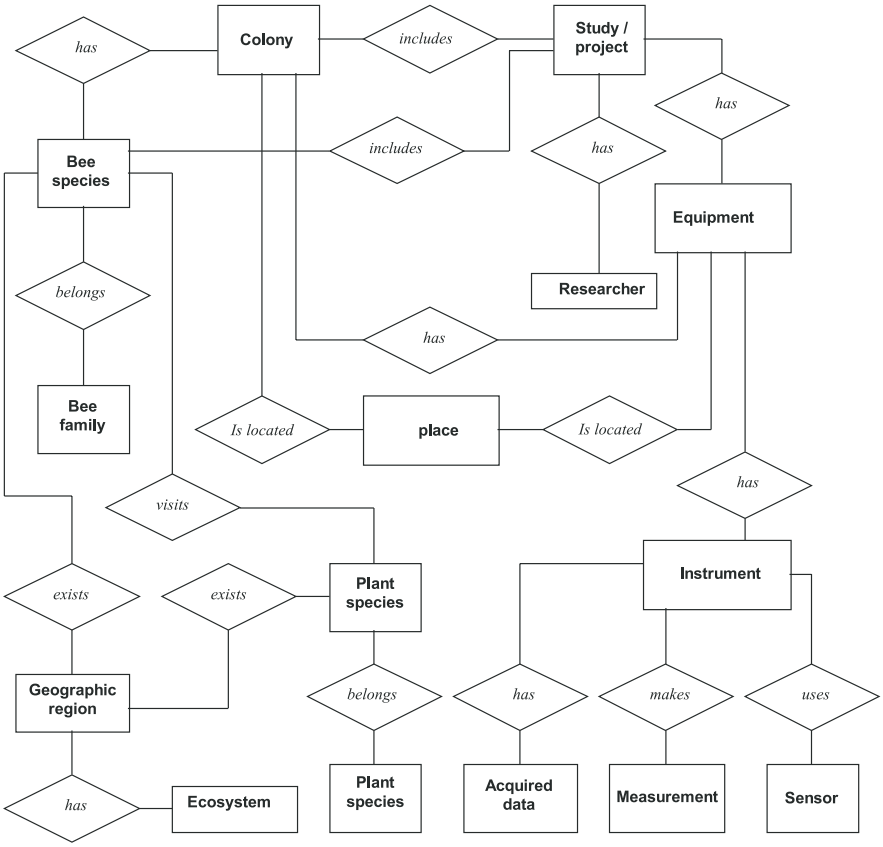


Figure 3. Simplified Entity-Relationship diagram of the database

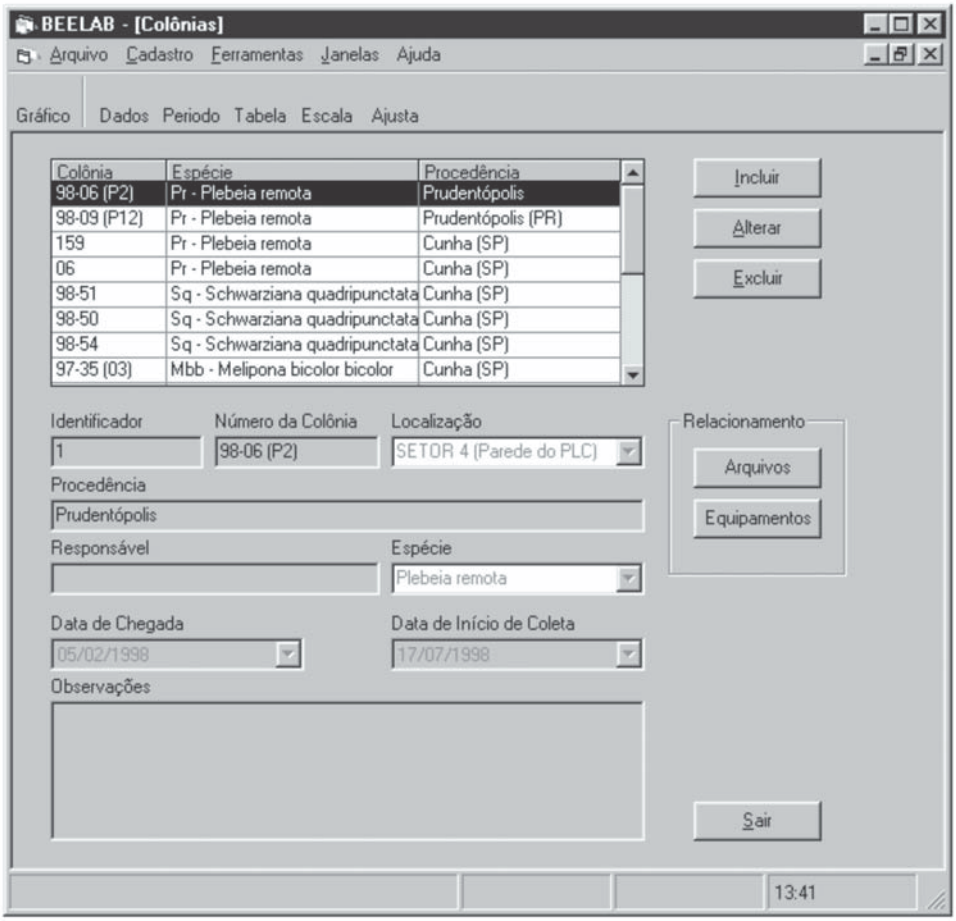


Figure 4. Example window of the Maintenance Module

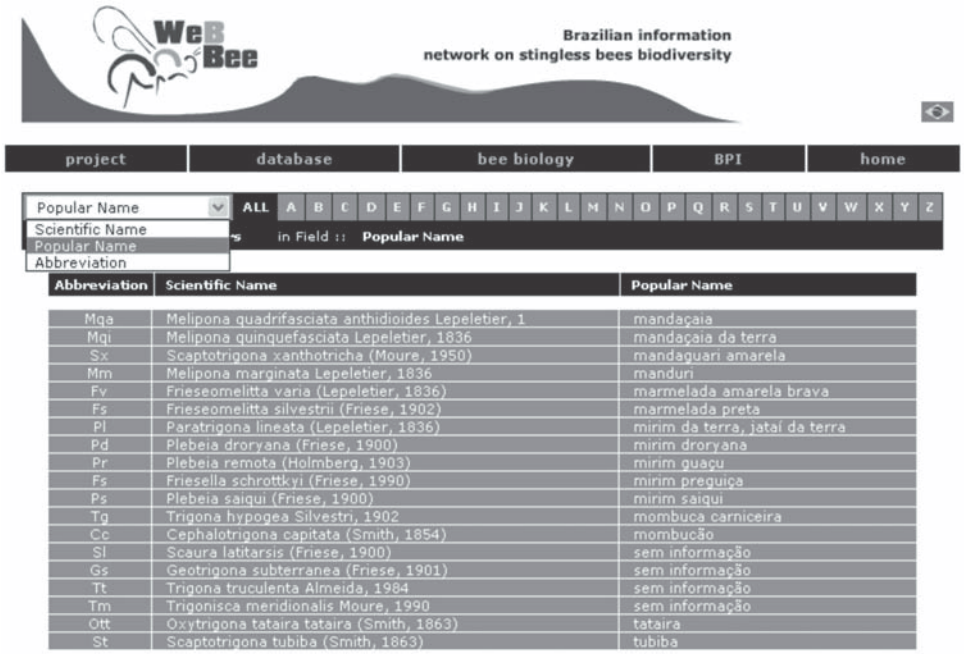


Figure 5. Example window of the Web Module.

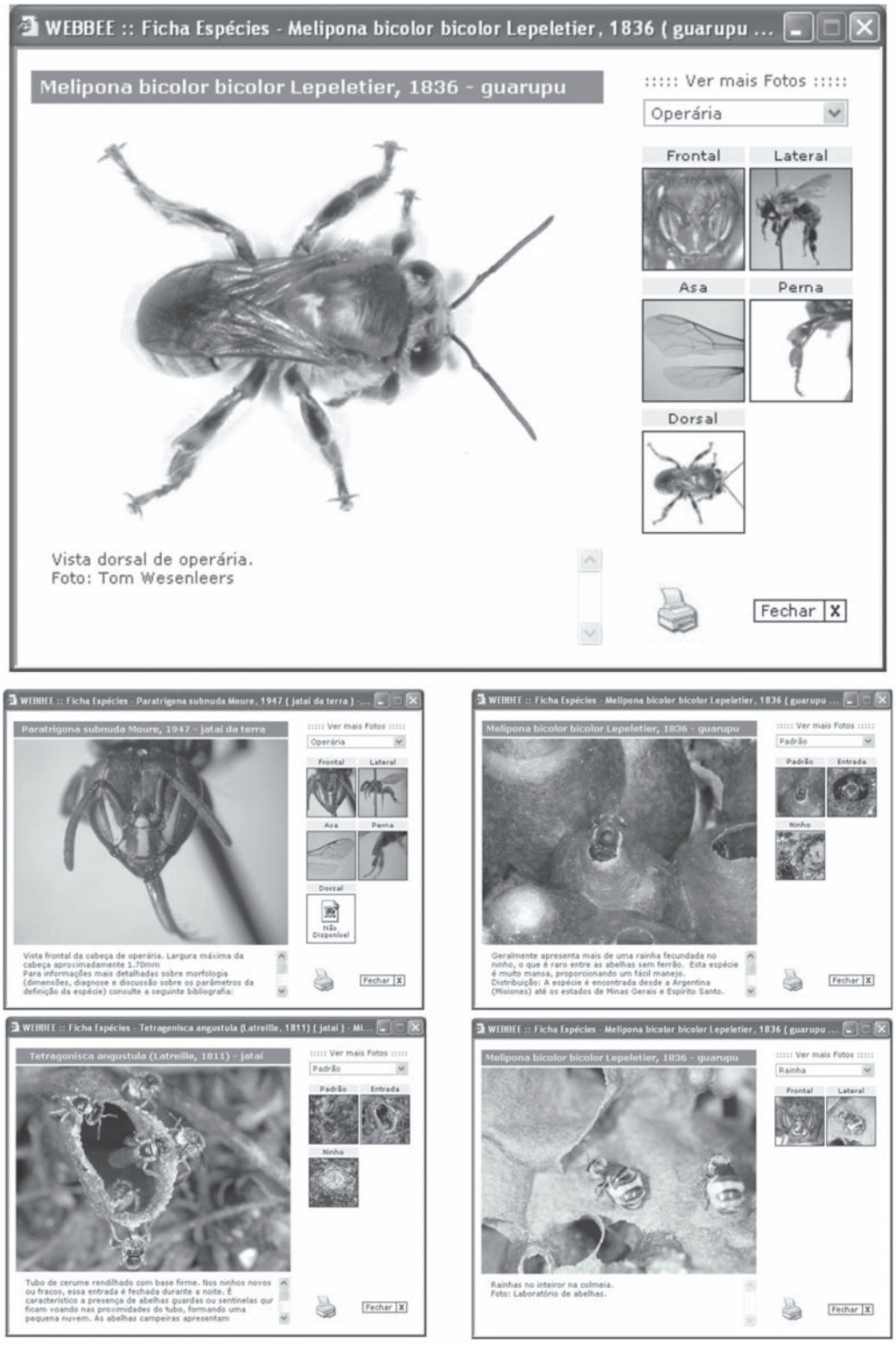


Figure 6. Example of a species record card and its many images and texts.

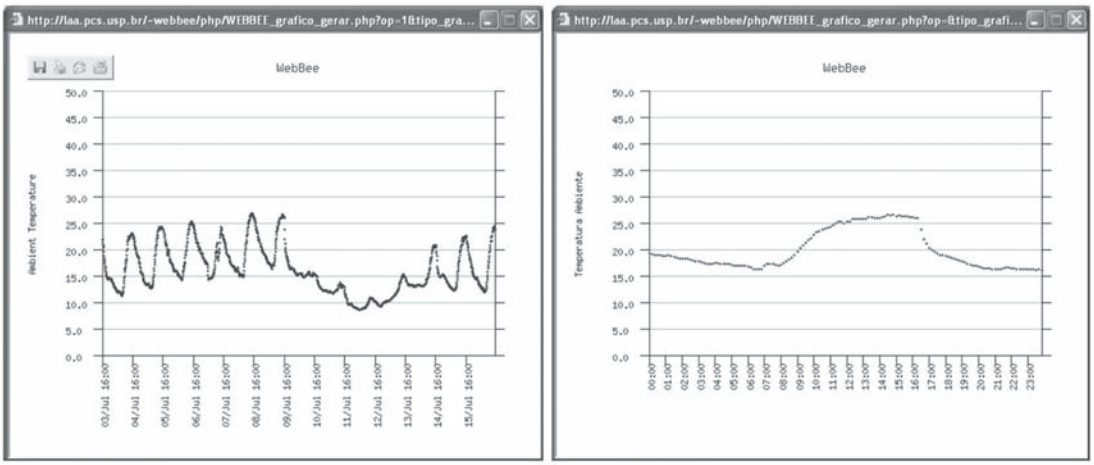


Figure 7. Example of an air temperature graphic for many days(left) and for a single day (right).

Revista
de Engenharia de Computação e Sistemas Digitais

Comunicações

PCS

número **1**

novembro 2003

Universidade de São Paulo
Escola Politécnica

Departamento de Engenharia de
Computação e Sistemas Digitais - PCS

TVoD: Sistema Multimídia sob Demanda para Distribuição de Material Digital das TVs Educativas

Regina Melo Silveira[1]
Rubens Ramires Fonseca[1]
Reinaldo Matsushima[1]
Sergio Rodrigo de Almeida[1]
Mauricio Monteiro[2]
Celso Hatori[2]
Ivan Negro Isola[2]
Graça Bressan[1]
Tereza Cristina M. B. Carvalho[1]
Wilson Ruggiero[1]

[1] LARC Laboratório de Arquitetura e Redes de Computadores
PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091- 5280

[2] TV Cultura de São Paulo – Fundação Padre Anchieta
Rua Cenno Sbrighi, 378 - CEP: 05036-900S - São Paulo / SP / Brasil

Este projeto tem como participantes o Laboratório de Arquitetura e Redes de Computadores (LARC) da Escola Politécnica da USP e a TV Cultura/Fundação Padre Anchieta, tendo recebido financiamento do CNPq, edital 09/2001 - SocInfo/ProTEM 01/2001, com duração de dois anos, tendo início em 2002.

As redes metropolitanas de alta velocidade (REMAV's), implementadas com o apoio do CNPq e da RNP, possibilitam a implantação de sistemas de transmissão sob demanda de conteúdos multimídia com qualidade de estúdio e compõe a espinha dorsal da Internet 2, sendo que o LARC foi um dos participantes na implantação deste projeto.

Tendo por base a infra-estrutura da REMAV, este projeto tem por objetivo disponibilizar um patrimônio cultural constituído por áudio e vídeo disponível na TV Cultura. A TV Cultura está implantando o NUDOC-MB - Núcleo de Documentação e Referencia de Música Brasileira com o apoio da Fundação Vitae. Este Núcleo será um centro de documentação e pesquisa de Música Brasileira, clássica e moderna, e contará com acervo de partituras e documentos, que já conta com um grande volume em doações, além de registros musicais em áudio e vídeo provenientes da Discoteca e Fitoteca da Rádio e TV Cultura.

O projeto consiste da definição e implantação de uma infra-estrutura para a digitalização de material de áudio e vídeo analógico, de forma a poder disponibilizar conteúdo junto a RMAV-SP. Além disso, o projeto permitirá fazer pesquisa e desenvolver uma metodologia de produção digital, edição não-linear, digitalização,

compressão, armazenamento, organização da hierarquia de memória e de acesso seguro a esses acervos. Essa metodologia será transferida para a TV Cultura e, a partir da integração do sistema de vídeo sob demanda a RMAV-SP, o material digitalizado poderá ser acessado por estudantes, professores e pesquisadores de diversas universidades brasileiras.

O projeto está sendo executado considerando os seguintes aspectos:

- Criação de uma metodologia para a migração de mídias analógicas para digital, incluindo o armazenamento, organização e busca do conteúdo, e avaliação dos padrões MPEG-1, MPEG-2 ou MPEG-4 para armazenamento.

- Pesquisa especifica para a otimização da arquitetura hierárquica de armazenamento e sistema de indexação de conteúdo, com o objetivo de implementar um sistema multimídia sob demanda mais eficiente com relação ao armazenamento, busca e recuperação do material armazenado. Para isto serão considerados padrões como o MPEG-7 e o MHEG, com o objetivo de implementar um sistema multimídia sob demanda mais eficiente com relação à busca e recuperação.

- Implementação de uma infra-estrutura para armazenamento, busca e distribuição de áudio e vídeo sob demanda.

- Avaliação do desempenho do sistema de multimídia sob demanda implementado, levando em consideração banda utilizada e a qualidade de serviço oferecida.

O projeto se encontra atualmente em fase de elaboração de protótipo.

Monte Carlo Aplicado à Avaliação de Perigos de Colisão entre Aeronaves em Pistas de Aproximação Paralelas e pouco Espaçadas

Mestrando:Paulo Ogata
Orientador: Prof. Dr. João Batista Camargo Júnior

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

O pressuposto básico da filosofia CNS/ATM (*Communications, Navigation, Surveillance/Air Traffic Management*) desenvolvida pela organização internacional da aviação civil (OACI), tem como princípio básico o atendimento da demanda do tráfego aéreo previsto para as próximas décadas. Neste contexto, no futuro, será primordial a construção de uma nova plataforma de forma não estruturada para o setor do transporte aéreo, além da introdução de procedimentos mais flexíveis aplicáveis aos pilotos e aos controladores de tráfego.

Os níveis de segurança nesta nova plataforma, com respeito aos riscos de colisão entre as aeronaves, serão atendidos pela utilização de dispositivos avançados instalados nos aeroportos e nas aeronaves e baseados nos sinais de satélites de posicionamento global. Estes novos equipamentos permitirão sinalizar aos pilotos e/ou controladores de vôo, acerca dos possíveis conflitos gerados pela perda de separação (lateral, longitudinal ou vertical), possibilitando dessa forma, a tomada de decisão para a execução das manobras corretivas.

A análise de risco pode ser aplicada para a avaliação dos níveis de segurança na área terminal. O cenário da área terminal é conhecido na literatura técnica como "operação de aproximação ou aterrissagens independentes e simultâneas em pistas paralelas e pouco espaçadas" ou CSPA (*Closely Spaced Parallel Approaches*).

Genericamente, CSPA, corresponde à expressão utilizada para referenciar o processo envolvendo aeronaves que operam conjuntamente e muito próximas uma da outra durante a fase de aproximação em pistas de aterrissagens paralelas que estejam separadas de 4.300 pés (~ 1.310 metros) ou menos.

Vários aeroportos possuem pistas de aterrissagens paralelas, cujo espaçamento entre as linhas centrais das pistas variam de aeroporto a outro e, portanto, com diversas restrições nos procedimentos de aterrissagens, de acordo com as condições meteorológicas presentes no momento da aproximação.

Uma taxa de aterrissagem ótima é aquela baseada em ótimas condições meteorológicas e, uma taxa reduzida, aquela em condições de tempo adversas como baixa visibilidade, ventos não favoráveis ou elevadas precipitações.

O aumento na capacidade das pistas paralelas em termos de um maior número de aproximações independentes será vital para suportar o crescimento esperado do tráfego aéreo, principalmente em péssimas condições meteorológicas, sem que sejam gerados em contrapartida, aumento no congestionamento nos circuitos de espera ou demasiados atrasos nas decolagens nos aeroportos de origem ou nas aterrissagens nos aeroportos de destino.

Com base nestas considerações, uma possível linha de pesquisa é a modelagem de uma ferramenta de auxílio à tomada de decisão e fundamentada na avaliação dos perigos de colisão entre aeronaves. O objetivo da ferramenta é permitir a análise da maximização da taxa de aterrissagens nas operações independentes em pistas CSPA, de acordo com a diminuição do espaçamento entre as linhas centrais das pistas paralelas.

Esta ferramenta é composta de um modelo dinâmico de duas aeronaves em aproximação em pistas CSPA, cujas incertezas inerentes ao posicionamento são modelados por meio de distribuições de probabilidade adequadas e incluem a separação lateral e longitudinal da aeronave em relação à trajetória nominal, a velocidade, a proa, entre outras.

As trajetórias das aeronaves são geradas pelo Método de Monte Carlo, cujos resultados permitem estimar a probabilidade de colisão entre as duas aeronaves, de acordo com o espaçamento entre as pistas paralelas. Para cada simulação, a ocorrência do estado perigoso é caracterizada em termos da distância mais próxima entre as trajetórias das duas aeronaves devido ao desvio de uma delas em direção a trajetória da outra. Tecnicamente, esta distância é conhecida como "*miss distance*" e utilizada na literatura técnica com o valor de 500 pés.

Esta ocorrência é definida como um evento "quase acidente", sendo computado como uma situação crítica de conflito. Portanto, para um determinado espaçamento entre as pistas paralelas, a probabilidade de ocorrência do estado perigoso pode ser calculada dividindo-se o número de ocorrência das situações críticas de conflito pelo número de simulações (trajetórias) processadas pelo Método de Monte Carlo.

Localização de Falhas em Sistemas Distribuídos através de Redes Neurais

Mestrando: Cláudio Hayashi Macoto
Orientador: Prof. Dr. João Batista Camargo Júnior
Bolsista de Iniciação Científica: Frederico Araujo

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

As redes de comunicação têm se tornado cada vez mais importantes para a sociedade. Esse aumento de importância veio acompanhado de um gigantesco acréscimo de complexidade. No início da evolução das redes de comunicação, alguns simples testes bastavam para encontrar um defeito. No entanto, atualmente, o tamanho das redes e a grande variedade de protocolos, equipamentos e sistemas tornam extremamente complexa a tarefa de se localizar a origem primária de uma falha. Esse aumento de complexidade faz com que a automação do processo de localização de falhas, torne-se difícil e seja necessário em muitos casos o envolvimento de um ser humano no processo. Por outro lado, a complexidade dessa tarefa também torna extremamente difícil para um ser humano localizar uma falha sem o auxílio de um sistema automatizado. Esses sistemas são denominados sistemas de gerenciamento de falhas.

O principal objetivo do gerenciamento de falhas é reagir a eventuais mudanças de comportamento dos sistemas, de forma a manter o correto funcionamento e garantir um determinado nível de qualidade de serviço. Essas mudanças são percebidas a partir de certos eventos e indicadores, que aqui denominaremos genericamente de alarmes.

A ocorrência de uma única falha em um sistema de comunicação pode gerar diversos alarmes. O processo de localização de falhas recebe como entrada um conjunto de alarmes e deve apresentar na saída o conjunto de elementos falhos que causaram esses alarmes. A localização dessa falha pode ser realizada de diversas formas. Uma delas é através da criação de um modelo de dependências que exiba as probabilidades de falha primária de cada elemento do sistema e exiba também as dependências entre cada elemento. Essa abordagem apresenta um problema: as probabilidades de falha não são constantes, e precisam ser atualizadas. Além disso, é extremamente difícil atribuir-lhes valores confiáveis.

Uma abordagem baseada em redes neurais reduziria a dependência com relação às probabilidades de falha, pois o sistema aprenderia com a ocorrência de falhas e as respectivas manutenções. Assim a pesquisa em andamento visa avaliar o potencial das redes neurais na localização de falhas.

Os resultados obtidos nessa pesquisa mostram que o uso das redes neurais no gerenciamento de redes se mostra bastante promissor. As simulações demonstraram que as redes neurais podem apresentar resultados bastante semelhantes a outras formas de localização de falhas. Considerando as propriedades de generalização, adaptação e tolerância a dados incompletos, pode-se concluir que as redes neurais se mostram uma ferramenta promissora para integrar futuros sistemas de gerenciamento de redes. Os próximos passos incluem a avaliação da capacidade de aprendizado das redes neurais quando aplicadas ao processo de localização de falhas. Para realizar essa avaliação utilizaremos um modelo bastante simplificado de rede de comunicação onde estarão representadas as taxas de comunicação entre os elementos, esse modelo será utilizado para determinar quais alarmes seriam gerados no caso de uma falha. Através de simulações verificaremos se a rede neural é capaz de se aprimorar com o processo de aprendizado por reforço apontando os elementos falhos que geraram um certo conjunto de alarmes.

Análise de Risco do Sistema ADS-B através de Redes de Petri Fluidas Estocásticas

Mestrando: Engº Lúcio Flávio Vismari

Orientador: Prof. Dr. João Batista Camargo Jr

Grupo de Análise de Segurança – PCS/EPUSP

Instituições de Apoio: IPV – Instituto de Proteção ao Vôo ,
CGNA – Centro de Gerenciamento de Tráfego Aéreo e Atech Tecnologias Críticas.

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

A utilização de computadores em sistemas de supervisão e controle de processos críticos quanto à segurança é um assunto de longa data. Por isso, este paradigma possui, em seu ciclo-de-vida, metodologias de análise e de desenvolvimento bem consolidadas quanto as áreas de segurança de sistemas de hardware e de segurança de software deste paradigma.

Devido a necessidade de redução de custos visando o aumento de competitividade, alterações no paradigma anterior tornam-se cada vez mais presentes. Enquanto, anteriormente, eram utilizados componentes dedicados e sistemas de padrão proprietário, atualmente utilizam-se componentes comercias – os chamados COTS (*Commercial-of-the-Shelf*) e sistemas de padrão abertos. Além disso, a alta complexidade dos processos atuais produz arquiteturas de sistemas de supervisão e controle altamente distribuídas e, portanto, baseadas em comunicação .

O novo paradigma dos sistemas de supervisão e controle de processos críticos quanto à segurança possui inúmeros novos modos de falha. Para estes serem detectados, são necessárias metodologias de análise que contemplem as novas características destes sistemas. Por isso, este trabalho de mestrado pretende mapear as novas características deste atual paradigma e apresentar uma metodologia de modelamento que possa contemplá-las em uma análise de segurança (safety). Como aplicação, serão utilizadas as características mapeadas e a metodologia de modelamento definida na análise de risco de em algum sistema que se enquadre no novo paradigma de sistemas de supervisão e controle de processos críticos quanto à segurança.

O estado-de-arte deste trabalho demonstrou as Redes de Petri Fluidas Estocásticas como uma boa solução para a metodologia de modelamento dos sistemas propostos. Além disso, definiu-se que o sistema a ser analisado na aplicação será o ADS-B (Automatic Dependable Surveillance - Broadcasting). O ADS-B é um dos sub-sistema de vigilância do próximo sistema mundial de gerenciamento de tráfego aéreo: o CNS/ATM (Communication, Navigation, Surveillance/Air Traffic Management).

Este trabalho espera contribuir, em senso amplo, com a melhoria das metodologias de análise de segurança, especificamente às metodologias de modelamento de sistemas e análise de risco. Com a aplicação realizada espera-se somar esforços na garantia de níveis aceitáveis de segurança do próximo sistema de gerenciamento de tráfego aéreo (CNS/ATM).

Revista
de Engenharia de Computação e Sistemas Digitais

Espaço da Graduação

PCS

número **1**

novembro 2003

Universidade de São Paulo
Escola Politécnica

Departamento de Engenharia de
Computação e Sistemas Digitais - PCS

Sistema de Busca em Áudio baseada em Descritores mpeg-7 gerados por Algoritmos de Reconhecimento de Padrão

Projeto de Formatura:
Reinaldo Matushima
Carlos E. M. Costa
Daniel M. Hiramatsu
Engenharia de Computação Cooperativo
Orientadora: Profa. Dra. Regina Melo Silveira

LARC Laboratório de Arquitetura e Redes de Computadores
PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091- 528

Hoje, mais do que nunca, com o grande avanço da Internet e de todas as tecnologias a ela associada, está sendo disponibilizado um volume cada vez maior de informações, principalmente no que se relaciona às mídias áudio-visuais. Ao contrário das pesquisas por textos, não há modos de se pesquisar por mídias de áudio a partir de uma característica específica se não há nenhuma informação associada já cadastrada. Estas restrições dos mecanismos de buscas atuais limitam e dificultam em muito o acesso a este universo potencial de informações.

O objetivo do projeto que está sendo desenvolvido é justamente especificar e implementar um mecanismo de busca aplicado sobre arquivos de áudio que reconheça características de arquivos de música de forma automatizada sem a necessidade de uma extração manual das informações. Este projeto se encontra dentro do escopo de um projeto da TV Cultura/LARC para a digitalização de um rico acervo de música brasileira.

Para a implementação deste sistema serão seguidas as especificações do padrão MPEG-7 desenvolvido pela ISO, tecnologia lançada para a descrição de qualquer tipo de conteúdo multimídia.

O MPEG-7 vem para criar um padrão para a representação das informações sobre mídia digital com flexibilidade na informação e permitindo a globalização e interoperabilidade destes dados.

Numa visão mais ampla, o projeto proposto, mais do que prover um mecanismo avançado de busca em arquivos de áudio visa a partir do uso de tecnologias recentes, como é o caso do MPEG-7, proporcionar uma metodologia de pesquisa e análise que possibilite sua aplicação em ambientes que vão além da própria Internet, que é justamente o que o próprio MPEG-7 propõe.

O escopo do projeto a ser realizado envolve o desenvolvimento de todo o sistema de busca, desde

os mecanismos de análise dos arquivos de áudio para reconhecimento de padrões musicais até o processo de geração dos descritores segundo o padrão estabelecido pelo MPEG7, como forma de indexação das mídias.

O sistema contará com uma base de arquivos de áudio e serão definidos algoritmos de análise e reconhecimento de padrões que serão executados sobre estes arquivos, obtendo as características do áudio. Para as características serem associadas de forma organizada e padronizadas aos arquivos de áudio, os sistemas usará o formato de descritores de mídia especificados no MPEG-7. Uma consulta realizada no sistema utilizará estes descritores para buscar os arquivos que correspondam ao que foi solicitado pelo usuário.

Devido ao grande universo de possibilidades de análise de áudio, e à grande complexidade dos mecanismos de reconhecimento de padrões existentes, optou-se pelo enfoque à pesquisa e estudo de mecanismos de reconhecimento de áudio que possibilitem a avaliação do gênero musical. A opção foi tomada, pois tende a direcionar o processo de pesquisa para um caminho mais específico, acelerando e possibilitando a obtenção de resultados mais efetivos.

Este projeto apresenta, sem dúvida, grandes desafios, tanto por sua multidisciplinaridade, como pelo fato de suportar esta tecnologia ainda recente que é o MPEG7.

Estágio Integrado Poli - Indústria

Por Reginaldo Arakaki
Coordenador de Estágios do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP
reginaldo.arakaki@poli.usp.br,

PCS Departamento de Engenharia de Computação e Sistemas Digitais
EPUSP Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 - 158, sala C1-46
Cidade Universitária - CEP: 05508-900 - São Paulo / SP / Brasil
Phone: + 55 11 - 3091 - 5583

O Estágio Integrado Poli-Indústria consiste num método de parceria envolvendo três elementos vitais para a formação de futuros profissionais de alta capacitação: Escola-Aluno-Empresa. A escola, responsável pela formação acadêmica conceitual. As empresas, responsáveis pela aplicação prática dos conhecimentos, desenvolvimento e aproveitamento dos alunos. Os alunos, futuros responsáveis por dirigir tais empresas são os elementos de integração. Visando o benefício dos três elementos, o Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP e seus professores elaboraram uma proposta baseada em planejamento conjunto entre a Poli e as empresas, para que os estagiários da Poli sejam incorporados nas grandes empresas da indústria, e tenham a sua formação profissional conduzida e adequada de acordo com as necessidades corporativas, aliadas com fundamentos conceituais da escola. Como suporte, os professores suportam os alunos na forma de tutoria.

Este método visa oferecer para as empresas mais opções para a formação dos alunos da Poli em termos de excelência profissional a baixo custo. Para tanto, fundamenta-se nos seguintes elementos:

- O aluno deve ser assistido durante a sua formação técnica e de maturidade como ser humano;
- A Escola tem a sua grade de formação conceitual acadêmica baseada em critérios didáticos e conceituais, avalizados por instituições oficiais e pela sua tradição histórica;
- A empresa tem a necessidade de formar recursos competentes na execução e na gestão de seus processos de negócios. Para isso, seus profissionais devem desenvolver a percepção crítica deste processo de negócio;

Desta forma, com base no planejamento prévio realizado pelos coordenadores nas empresas, com o apoio dos professores do PCS, o aluno tem as suas atividades integradas entre conceitos e projetos na indústria, de maneira uniforme: as aplicações práticas sustentadas por conceitos e conceitos experimentados na prática. Com esse planejamento é possível que o aluno, ao se formar, tenha até três anos de experiência profissional na empresa e está apto a trabalhar de maneira engajada e integrada com as equipes.

Com objetivo de efetivar a referida parceria, o Departamento PCS/EPUSP organizou um evento em 15 de abril de 2003, nas instalações cedidas pelo Banco Real ABN AMRO BANK. Estiveram presentes representantes de conceituadas instituições: Booz Allen, Microsoft, Banco Unibanco, Itautec Philco, Banco Santander, Banco Real ABN AMRO, Mckinsey, Accenture, Scopus, DTS Altran e EPUSP. A estratégia é que, a partir deste evento, cada instituição tenha um professor representante na EPUSP para evoluir na elaboração de planos de formação de recursos humanos, especializados para as empresas. Novas empresas serão convidadas a participarem de eventos similares.

O Setor de Estágios do PCS teve contribuições importantes de seus professores. Em especial, alguns participaram desta iniciativa com muito envolvimento: Prof. Dr. Edson Fregni, Prof. Dr. Paulo Sergio Cugnasca e Prof. Dr. João Batista Camargo apoiados pela Chefia do Departamento Prof. Dr. Moacyr Martucci Jr.

Informações aos Autores

A ***Revista de Engenharia de Computação e Sistemas Digitais*** é uma publicação do ***Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica - PCS - da USP***, de periodicidade quadrimestral, que tem como escopo a divulgação de artigos relacionados a esta área, com enfoque acadêmico. A submissão de trabalhos deve ser realizada através do site:

http://revista.pcs.usp.br

e segue as seguintes regras:

1. Os artigos submetidos à publicação nesta revista não devem ter sido submetidos, apresentados ou publicados em outros lugares e não devem estar sujeitos a copyright de publicação, a menos que seja explicitamente indicado e autorizado.

2. Tipos de conteúdos aceitos:

   a. Artigos de até 20 páginas, em formato A4 com espaçamento duplo e respectivas figuras;

   b. Resumos de projetos de alunos de graduação, resultantes de bolsas de iniciação científica e projetos de formatura de graduação – máximo de 500 palavras.

   c. Comunicações de pesquisas em andamento ou mesmo concluídas – resumos contendo objetivos, metodologia, resultados esperados e alcançados, além das entidades financiadoras – máximo de 500 palavras.

3. Formato dos originais:

   a. Título, nome e contato dos autores e resumo (em português e em inglês) de até 250 palavras.

   b. A informação completa dos autores deve aparecer no final do artigo. Inclui formação, título, instituição à qual cada autor está vinculado. Também devem ser informadas as subvenções ao projeto de pesquisa ao qual o artigo está relacionado.

   c. As referencias bibliográficas devem estar no padrão abaixo descrito:

      [1] X. Li, S. Paul and M.H. Ammar. Layed Video Multicast with Retrnsmission (LVMR): Evaluation of Hierarchical Rate Control. In IEEE INFOCOM'98, San Francisco, , California, pages 1062-1072, 1998.

      [2] M.D. Amorim, O.C.M.B. Duarte and G. Pujolle. Na Improved MPEG behavioral Analysis with Autonomous Parameter-Extracting Algorithm for Strict Video Applications. In IEEE ICC'2000, New Orleans, USA, pages 831-835,2000.

Universidade de São Paulo
Escola Politécnica
Departamento de Engenharia de Computação e Sistemas Digitais - PCS