MODELAGEM E ANÁLISE DE DESEMPENHO DE UMA ARQUITETURA DE FLUXO DE DADOS

Sérgio Roberto Pereira da Silva[1]
Departamento de Informática
Universidade Federal de Maringá
Av. Colombo, 3690
87020-900 Maringá PR
(0442) 26-2727 R324
srpsilva@brfuem.bitnet

Arthur João Catto[2]
Departamento de Ciência da Computação
UNICAMP
Caixa Postal 6065
13081-970 Campinas SP
(0192) 39-8442
catto@dcc.unicamp.br

RESUMO

Este artigo relata o desenvolvimento, a validação e a aplicação de um modelo analítico baseado na Teoria Geral de Redes para o estudo da arquitetura de fluxo de dados de Manchester.

O modelo resultante foi validado utilizando-se o modelo-Q, um método analítico determinístico apoiado por redes de Petri. O mesmo método foi empregado para se analisar o modelo e avaliar algumas alternativas de projeto, visando a identificação e a remoção de pontos de estrangulamento do desempenho da arquitetura.

Como resultado, além da obtenção um modelo satisfatório para a arquitetura de fluxo de dados de Manchester, demonstrou-se a potencialidade do modelo-Q para a identificação de problemas e avaliação de alternativas de projeto para esta classe de arquiteturas.

ABSTRACT

This paper reports the development, validation and application of an analytical model based on the General Net Theory for studying the Manchester data flow architecture.

The resulting model was validated using the Q-model, an analytical deterministic method supported by Petri nets. The same method was used to analyze the model and to evaluate some design alternatives, aiming at identifying and removing performance bottlenecks in the architecture.

As a result, besides obtaining a satisfactory model for the Manchester data flow architecture, the potential of the Q-model for identifying problems and evaluating design alternatives for this class of archietctures was demonstrated.

---

[1]Professor Assistente, Mestre em Ciência da Computação, Unicamp, 1991.
[2]Professor Assistente Doutor, Ph.D. in Computer Science, University of Manchester, 1981.

1 INTRODUÇÃO

Atingir o desempenho e a competitividade exigidos dos supercomputadores atuais importa em grandes investimentos econômicos e tecnológicos. Nos últimos anos, com a aproximação dos limites físicos superiores para os componentes de hardware, tem-se buscado com maior ênfase melhorar o desempenho dos sistemas explorando o paralelismo existente nas aplicações. Nesse contexto, a simultaneidade e a concorrência na realização de tarefas passam a ser consideradas chaves para a obtenção de máquinas de grande potência [GPK82].

Por outro lado, os bons resultados conseguidos através da adaptação do modelo computacional de von Neumann para o tratamento do paralelismo regular não parecem estender-se aos casos onde essa regularidade inexiste. Por essa razão, alguns dos maiores grupos de pesquisa mundiais vêm pesquisando há vários anos outros modelos computacionais mais adequados a ambientes altamente paralelos. Em sua maioria, as propostas surgidas agrupam-se em duas classes básicas: a dos modelos dirigidos pela demanda de resultados e a dos modelos dirigidos pela disponibilidade de dados [TBH82].

Entre os modelos dirigidos pela disponibilidade de dados destaca-se o de fluxo de dados, que já é suportado por algumas implementações, como a Máquina de Fluxo de Dados de Manchester (*MFDM*), por exemplo [GKW85, GW83]. Os resultados obtidos pelo grupo de Manchester, entre os quais um protótipo e uma variedade de trabalhos publicados, oferecem uma boa visão das opções de projeto e dos pontos fortes e fracos da implementação.

Entre as qualidades atribuídas ao modelo de fluxo de dados está a desnecessidade de serem explicitadas a sincronização entre tarefas e a partição do algoritmo, que ficam por conta do "hardware". Apesar de sua elegância, o modelo é novo e a experiência com ele limitada para que surjam implementações com eficiência comparável à das máquinas von Neumann mais potentes. Além disso, algumas questões específicas desta classe de modelo ainda necessitam ser adequadamente resolvidas.

Para o estudo das questões associadas ao desenvolvimento de arquiteturas vem sendo dada atenção crescente a técnicas de modelagem analítica como alternativa à utilização de simuladores e protótipos. Entre as razões para essa opção está o fato de essas técnicas terem rigoroso embasamento teórico e apresentarem níveis de complexidade e custo de desenvolvimento razoáveis [All80]. Dentre as técnicas analíticas mais conhecidas encontram-se as redes de Petri, que são técnicas gráficas associadas a um formalismo matemático, muito utilizadas na modelagem de sistemas de eventos discretos [Rei85].

Este artigo relata o desenvolvimento, a validação e a aplicação de um modelo analítico determinístico baseado em redes de Petri para o estudo da arquitetura de fluxo de dados de Manchester.

2 O MODELO DE FLUXO DE DADOS

O modelo de fluxo de dados [Den85, AGP78, GWG78] é uma abstração clássica para a representação de algoritmos paralelos, na qual programas são expressos como grafos orientados denominados *grafos de fluxo de dados* (*GFDs*). Num *GFD*, cada *vértice* representa uma função primitiva e cada *aresta* uma dependência de dados entre dois vértices.

Durante a execução de um *GFD*, um vértice torna-se *habilitado* para execução ao receber um conjunto completo de argumentos. Todo vértice habilitado *dispara*, aplicando sua função aos argumentos recebidos, após o que transfere os respectivos resultados a seus descendentes diretos no *GFD*.

O modelo não faz qualquer suposição quanto ao número de vértices ativos num determinado instante, o que depende apenas das características do *GFD* executado. Por outro lado,

supõe-se que todo vértice habilitado dispare e que todo disparo de vértice e transferência de resultados tenham duração finita.

O modelo de fluxo de dados possui duas variantes principais: estática [Den85] e dinâmica [AGP78, GWG78]. A variante dinâmica, por ser menos restritiva, serviu de base à maioria das implementações conhecidas.

3 A ARQUITETURA DE FLUXO DE DADOS DE MANCHESTER

A *MFDM* [GW80, GKW85] implementa o modelo de fluxo de dados dinâmico através de um anel com cinco unidades ligadas em "pipeline" (Fig. 1), onde os dados e resultados circulam representados em fichas rotuladas. Nos *GFD*s suportados, os vértices implementam funções unárias ou binárias e são capazes de produzir zero, um ou dois resultados[1].

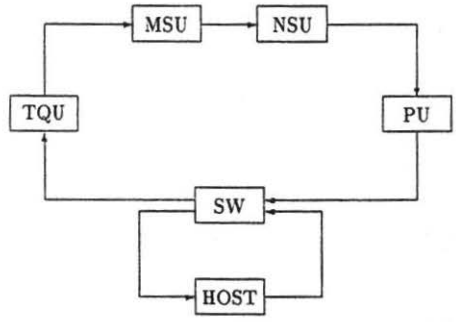

Figura 1: Máquina de Fluxo de Dados de Manchester

As unidades são síncronas internamente e são dotadas de relógios independentes. A intercomunicação é toda assíncrona, havendo "buffers" de entrada e saída em todas as unidades.

A Matching Store Unit (*MSU*) é responsável por agrupar as fichas que se dirigem a um mesmo vértice binário, habilitando-o para execução quando ambas estiverem disponíveis. As fichas que se dirigem a vértices unários não necessitam de agrupamento e, portanto, passam direto para o "buffer" de saída.

O tratamento das fichas que se dirigem a vértices binários depende da disponibilidade das respectivas parceiras na *MSU*. Quando a parceira de uma ficha de entrada está presente, ambas são agrupadas num pacote que é enviado ao "buffer" de saída. Quando a parceira está ausente, a ficha de entrada fica armazenada temporariamente.

A *MSU* é uma memória pseudo-associativa, implementada por uma tabela de espalhamento[2] mapeada sobre uma memória de acesso aleatório.

Os pacotes de dados formados pela *MSU* são enviados para a Node Store Unit (*NSU*), onde fica armazenado o *GFD* em execução. A partir de cada pacote de dados recebido da

[1] Na versão final do protótipo foram incluídas instruções capazes de gerar mais do que dois resultados.
[2] "hash table"

MSU a *NSU* prepara um pacote executável, agregando aos dados recebidos uma cópia do vértice por eles endereçado.

Esses pacotes executáveis são enviados à Processing Unit (*PU*), onde são processados e dão origem a zero, uma ou duas novas fichas contendo os respectivos resultados. A *PU* dispõe de dois estágios: o primeiro executa algumas operações de alta velocidade que não podem ser realizadas de forma distribuída; o segundo é formado por um conjunto paralelo de 1 a 20 Function Units (*FU*s), que executam a maior parte das instruções.

Uma vez que a duração e o número de resultados das instruções são variáveis, o desempenho da máquina na execução de um programa depende não só da topologia do *GFD*, mas também da distribuição dos tipos de instruções presentes no grafo.

A Switch Unit (*SW*) permite a comunicação com um computador hospedeiro ("Host") que se encarrega da entrada dos programas e dados e da saída dos resultados.

A Token Queue Unit (*TQU*) comporta-se como uma fila simples e foi adicionada ao anel para permitir a uniformização do fluxo de fichas, compensando eventuais irregularidades nas respectivas taxas de produção e consumo.

A uniformidade do fluxo no anel é também prejudicada pelo armazenamento das fichas não-emparelhadas pela *MSU*, o que provoca o aparecimento de bolhas entre essa unidade e a *NSU* e afeta consideravelmente o desempenho da máquina.

A decisão de restringir o número de arestas nos vértices dos *GFD*s buscou simplificar a implementação da *MSU* e da *NSU* e limitar a largura do barramento paralelo requerido pelos pacotes transmitidos.

Durante seu desenvolvimento, a *MFDM* recebeu a adição de três outras unidades visando a melhoria do desempenho ou a eliminação de alguns problemas tardiamente identificados. Essas unidades, rapidamente mencionadas abaixo, não aparecem na Fig. 1.

Uma Overflow Unit (*OU*) ligada à *MSU*, mas externa ao anel principal, encarrega-se do tratamento de eventuais estouros da capacidade de armazenamento temporário.

Uma Structure Store Unit (*SSU*), ligada ao anel através da *SW*, é responsável pelo tratamento de estruturas de dados [KG86, SK86]. O mecanismo de tratamento é inspirado no conceito de I-structures [AT80].

A proliferação descontrolada de atividades paralelas em arquiteturas de granularidade fina tem sido apontada como uma das principais causas da degradação de seu desempenho durante a execução de muitos programas com paralelismo elevado. Para evitar esse problema foi implementada uma Throttle Unit (*TU*), capaz de limitar o grau máximo de paralelismo explorado pela máquina em qualquer instante [Rug87].

4 MODELAGEM DA UTILIZAÇÃO DE RECURSOS NA MFDM

Na construção do modelo conceitual da *MFDM* [dS91] foram feitas diversas simplificações:

- Foram modeladas apenas as unidades básicas, ficando ausentes do modelo a *OU*, a *SSU* e a *TU*.
- Foram modeladas apenas as características básicas da *MSU*, em prejuízo de um suporte completo às Matching Functions — informações carregadas pelas fichas com o objetivo de diversificar as opções para emparelhamento oferecidas pela *MSU* e ampliar as possibilidades de utilização da máquina [Cat81].
- Foram modeladas apenas instruções capazes de gerar zero, um ou dois resultados.

Após considerar algumas metodologias para análise de desempenho derivadas da Teoria Geral de Redes [MCB84, HV87, Taz88], resolveu-se adotar a desenvolvida por Tazza, que se baseia na avaliação de um modelo de redes de lugar/transição restrito [Taz88], com as adaptações introduzidas por Fernandes [Fer90]. As restrições impostas ao modelo visam evitar problemas que poderiam impossibilitar a análise, como bloqueios perpétuos, por exemplo. As adaptações visam torná-lo aproximativo, para permitir o tratamento de casos onde ocorrem bloqueios na posse de um recurso.

Procurando limitar o número de recursos e parâmetros do modelo, foram feitas as seguintes suposições simplificadoras adicionais:

- Todas as instruções têm a mesma duração.
- As ações de emparelhamento e armazenamento temporário realizadas na *MSU* sobre as fichas destinadas a vértices binários, são igualmente distribuídas.
- O sistema encontra-se em regime estacionário com paralelismo constante (PI), desconsiderando-se as flutuações que naturalmente ocorrem durante a execução de um *GFD*.
- As variações do fluxo na *MSU* e na *PU* compensam-se mutuamente, isto é, o número de fichas consumidas é igual ao de fichas produzidas.

Foram tomados como parâmetros do modelo:

- o tempo médio de permanência das fichas em cada uma das cinco unidades;
- a porcentagem de fichas destinadas a vértices unários, que passam direto pela *MSU* (Pby);
- o número de FUs;
- o paralelismo médio do programa[3] (PI).

As simplificações e hipóteses formuladas acabaram por levar a um modelo com 6 recursos e 8 parâmetros.

De acordo com a metodologia adotada, as variações nos valores dos parâmetros são modeladas através da ponderação dos resultados apresentados por um conjunto adequado de redes independentes [Fer90]. Esse mecanismo foi adotado para tratar os diferentes comportamentos possíveis para a *MSU* e a redução de fluxo que ocorre após essa unidade. Ao final, cada resultado foi obtido a partir da combinação das avaliações de doze redes distintas.

Para reduzir o número de redes a ser avaliado em cada caso, decidiu-se simplificar ainda mais o modelo. Alguns recursos foram removidos, passando a ser considerados subtarefas do sistema. Dessa forma, a eles não corresponderam subcadeias nem lugares de espera e a rede final foi simplificada.

Essa decisão reduziu o volume de dados gerados e facilitou a análise, embora introduzisse o risco de perder-se a visualização de recursos que poderiam estar bloqueando o desempenho do sistema. No entanto, a decisão foi mantida, uma vez que essa verificação poderia voltar a ser realizada facilmente com a reinclusão dos recursos suspeitos.

O modelo final aparece na Fig. 2.

[3] O paralelismo médio do programa é representativo da influência do paralelismo efetivo sobre o desempenho do sistema [GW83].

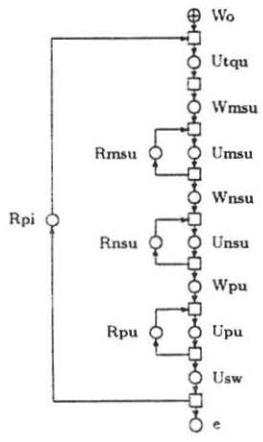

Figura 2: Modelo-Q1 correspondente à *MFDM* básica.

4.1 VALIDAÇÃO DO MODELO DE RECURSOS

Não existe uma coerência absoluta entre os valores dos parâmetros na documentação disponível sobre a *MFDM*. Nos casos de dúvida, realizou-se um teste preliminar antes da opção por um dos valores disponíveis. Os valores adotados, que provêm de [GKW85] exceto nos casos indicados, foram os seguintes:

- Tempo de permanência na $SU = 200$ ns.
- Tempo de permanência na $TQU = 375$ ns.
- Tempo de permanência na *MSU* (de acordo com a documentação de projeto [dS82]):
  - Fichas destinadas a vértices unários = 180 ns.
  - Fichas destinadas a nós binários, emparelhadas = 360 ns.
  - Fichas destinadas a nós binários, armazenadas temporariamente = 540 ns.
- Tempo de permanência na $NSU = 500$ ns.
- Tempo de permanência na $PU = 7700$ ns[4].

Os demais parâmetros necessários à avaliação do modelo dependem dos programas executados, e foram estimados a partir do conjunto de programas usados para avaliar o protótipo [GKW85, GW83]. A Fig. 3, por exemplo, mostra os resultados relativos a um programa de integração numérica (*LAPLACE*), com $Pby = 70\%$ e $PI = 50$. Observa-se que, nesse caso, os desvios entre as curvas experimental e do modelo foram sempre inferiores a 10%.

A Fig. 4 mostra os resultados obtidos para um programa de soma de inteiros com recursividade dupla (*SUM*), onde $Pby = 61\%$ e PI assume os valores 1, 5, 10, 20, 50 e 150. Observa-se

[4] A menos de uma constante (592 ns), este valor depende do número de *FU*s disponíveis e da respectiva taxa de processamento. Supôs-se a existência de 12 *FU*s com taxa de processamento de 0.14 Mips, que é a média dos valores apresentados em [GKW85] e leva a um tempo de permanencia nas *FU*s de 7108 ns.)

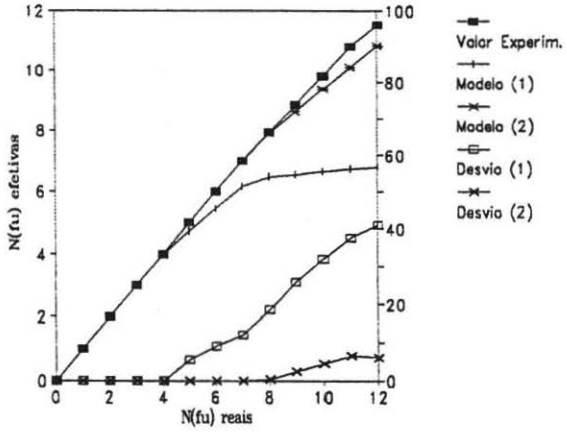

Figura 3: Desempenho e desvio do modelo para o programa *LAPLACE*.

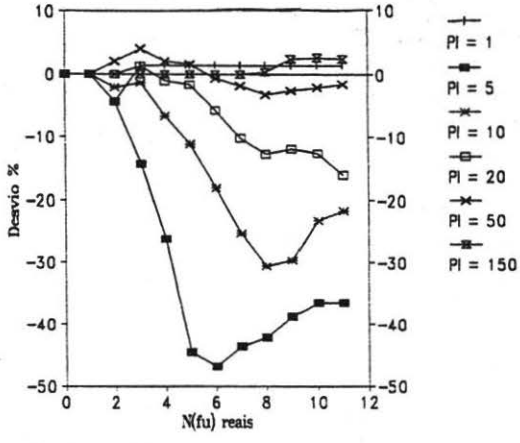

Figura 4: Desvio do modelo para o programa *SUM* para diversos valores de *PI*.

nesse caso que, embora o modelo tenha bom comportamento para $PI = 1$ e $PI > 50$, para valores intermediários do paralelismo médio ($5 \leq PI \leq 20$), a resposta não é boa, mesmo quando o número de FUs ativas é pequeno.

Uma possível razão para essa diferença está no fato de que nesse caso a distribuição do paralelismo varia muito durante o processamento [GW83]. Essa variação ocasiona o aparecimento de muitas bolhas no "pipeline", em conseqüência do grande número de fichas que precisam ser retidas na *MSU* devido à ausência de suas parceiras. Isto, por sua vez, resulta numa queda de desempenho das FUs e provoca uma redução da taxa de processamento medida.

Flutuações do paralelismo efetivo durante o processamento não foram consideradas quando da construção do modelo. Pelo contrário, elas foram eliminadas pela principal simplificação introduzida: a hipótese de que o sistema estaria em regime estacionário com taxa de trabalho igual ao paralelismo médio do programa.

Essas considerações são coerentes com os altos desvios percentuais encontrados no programa *SUM*, para os casos de $5 \leq PI \leq 20$. O paralelismo nesse programa concentra-se em aproximadamente 15% do tempo de processamento, fazendo com que os efeitos do surgimento de bolhas acentuem-se quando o paralelismo médio é restrito. Como o paralelismo efetivo nos 85% restantes do tempo de processamento é muito baixo — cerca de apenas 30% do valor médio — o desempenho das FUs fica muito prejudicado [GW83]. Esse fenômeno não é sentido com tanta intensidade para valores altos de PI pois, apesar do grande número de bolhas, o paralelismo efetivo ainda supera o número de FUs ativas.

Resultados semelhantes foram observados também para outros programas, com diversos valores de PI. As conclusões foram invariavelmente as mesmas, demonstrando que a perda de representatividade do modelo está restrita a casos onde a distribuição do paralelismo durante o processamento distancia-se da uniforme [dS91].

Embora ainda se investiguem formas mais efetivas para o tratamento de flutuações do paralelismo pelo modelo, os resultados já obtidos foram considerados compatíveis com a finalidade do trabalho. Assim, passou-se à fase de avaliação da *MFDM* empregando-se o modelo de recursos com os parâmetros aqui definidos.

4.2 ANÁLISE DE DESEMPENHO

Para avaliar a *MFDM* levantaram-se curvas de desempenho e de utilização de recursos para diversas situações. Destas, retiraram-se informações sobre pontos de estrangulamento, identificados pela variação da inclinação das curvas de desempenho e pelo afastamento das curvas de utilização dos recursos dos respectivos valores ideais. Para a localização dos pontos de estrangulamento foram também analisadas as curvas do tempo médio de espera para atendimento e da população média de fichas à espera de atendimento em cada recurso.

Uma vez que nesta fase não se consideram *GFD*s específicos, os parâmetros do modelo apresentados na seção anterior foram complementados por:

- Paralelismo médio $PI = 100$.
- Porcentagem de fichas destinadas a vértices unários $Pby = 60\%$.

O valor adotado para PI justifica-se pelo interesse no desempenho da arquitetura em programas com alto grau de paralelismo. O valor adotado para Pby aproxima-se da média encontrada nos programas incluídos no pacote de avaliação da *MFDM* [GKW85].

Devido ao interesse na avaliação da influência do acréscimo de FUs à PU sobre o comportamento da *MFDM*, as curvas para análise foram traçadas em função do número de FUs ativas. O número máximo de FUs ativas foi arbitrariamente fixado em 25.

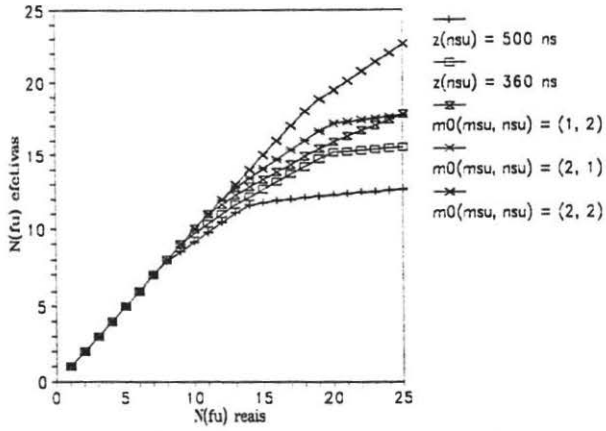

Figura 5: Eficiência no uso das FUs.

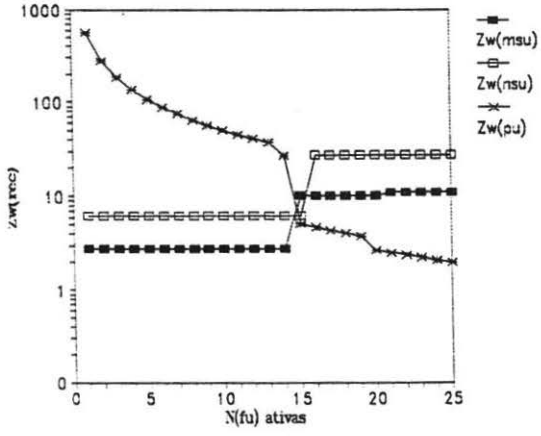

Figura 6: Tempo médio de espera para atendimento com $PI = 100$ e $Pby = 60\%$.

A curva de eficiência do modelo (Fig. 5) apresenta duas deflexões. Para identificar a unidade responsável pela queda do desempenho utilizou-se o gráfico do tempo médio de espera para o atendimento em um recurso (Fig. 6), com as seguintes conclusões:

- As unidades não possuem um mesmo regime de trabalho.
- Com até 14 *FU*s ativas o gargalo do sistema está na *PU*.
- Com 15 *FU*s ativas o gargalo desloca-se para a *MSU*.
- Com 16 *FU*s ativas o gargalo desloca-se para *NSU*.

Para estimar as razões desse comportamento deve-se observar que:

- A taxa de processamento da *MSU* depende da porcentagem de fichas destinadas a vértices unários (*Pby*) e também da parcela das fichas destinadas a vértices binários que encontram sua parceira na memória.
- A taxa de processamento na *NSU* é constante.
- O número de fichas que chegam à *NSU* e à *PU* é menor que o das que chegam à *MSU*, devido aos emparelhamentos que esta realiza.
- A existência de um maior número de *FU*s ativas provoca o aumento na taxa de processamento efetivo da *PU*.
- O modelo-Q desenvolvido é determinístico e fornece uma avaliação do melhor caso na análise do desempenho do sistema.

Assim, os saltos observados na Fig. 6 decorrem, primeiramente, do modo determinístico e aproximativo com que o modelo está construído, incorporando as variações dos tempos de processamento na forma de redes alternativas, cujos resultados são ponderados pelas respectivas probabilidades associadas. Além disso, mesmo as curvas ponderadas estão representadas em função do número de *FU*s ativas, não sendo possível representar frações de *FU*s.

Dessa forma, o crescimento do número de *FU*s ativas leva a taxa de processamento nominal da *PU* a atingir a das demais unidades, fazendo com que o gargalo em uma rede alternativa específica mude para outra unidade. No primeiro salto, com 14 *FU*s ativas, atinge-se o valor da taxa da *MSU* na rede em que todos os emparelhamentos fracassam, o que somente afeta as redes onde a *MSU* é o gargalo. Ao acrescentar-se mais uma *FU*, a taxa de processamento da *PU* iguala-se à da *NSU*, provocando um novo salto, desta vez na curva desta unidade. Observa-se ainda que o fenômeno se repete nas curvas da *MSU* e da *PU*, ao se atingirem 20 *FU*s ativas. Esses saltos estão relacionados com o fato de a taxa de processamento da *PU* atingir a da *MSU*, na rede em que nenhum emparelhamento fracassa.

O gráfico de utilização dos recursos do sistema (Fig. 7) fornece ainda outras informações, menos visíveis nos demais gráficos. Assim, é possível verificar, por exemplo, que, para até 7 *FU*s ativas, a utilização da *PU* é máxima, decaindo a partir daí. Essa redução indica ociosidade de *FU*s, decorrente do aparecimento de bolhas no "pipeline", provocadas pelo armazenamento de fichas na *MSU*. A partir de 7 *FU*s a *PU* não trabalha mais à sua capacidade nominal, embora o acréscimo de novas *FU*s continue a melhorar o desempenho do sistema até se chegar a 14 *FU*s (Fig. 5 e 7).

Ao se atingirem 15 *FU*s ativas, a taxa de processamento da *PU* supera a da *MSU* na rede onde todos os emparelhamentos fracassam: o efeito das bolhas passa a ser sentido com maior

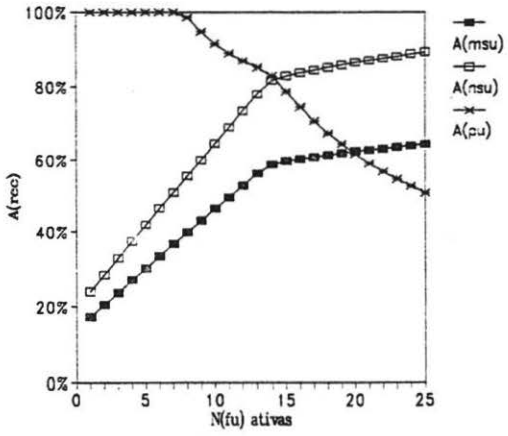

Figura 7: Utilização dos recursos do sistema com $PI = 100$ e $Pby = 60\%$.

intensidade, provocando a passagem temporária do gargalo para *MSU*. A partir de 16 *FU*s ativas, mesmo com o aumento do número de bolhas, a *NSU*, cuja taxa de processamento já é menor que a *MSU*, passa a ser o gargalo, ao ser superada também pela *PU*.

Essa avaliação diverge parcialmente dos resultados reportados por Ghosal ([GB90, GB87]) a partir da análise de um modelo estocástico de redes de filas. Essas divergências serão tratadas em maior detalhe na Seção 5.

4.3 ANÁLISE DE ALTERNATIVAS DE IMPLEMENTAÇÃO

Identificadas as unidades responsáveis pela queda do desempenho do sistema, foram consideradas duas soluções alternativas: a redefinição do tempo de processamento da *NSU* e a alteração do número de unidades disponíveis. Ambas foram examinadas modificando-se os parâmetros do modelo de recursos desenvolvido.

No primeiro caso, fixaram-se os tempos de permanência na *NSU* em 360 ns e na *TQU* em 288 ns, o que, na prática, corresponde a equalizar as taxas de processamento da *TQU*, *MSU* e *NSU*.

Os resultados mostraram uma melhora na eficiência do uso das *FU*s, embora as deflexões da curva persistissem, indicando a existência de estrangulamentos. Da análise do gráfico do tempo médio de espera para atendimento concluiu-se que a razão das deflexões continuava sendo a mesma e que o gargalo se deslocara para a *MSU* [dS91].

No segundo caso, dobrou-se o número de centros de serviço da *NSU*. O tempo de permanência na unidade foi outra vez fixado em 360 ns, de modo a equilibrar as taxas de processamento e permitir um uso mais eficiente das *FU*s. Essa escolha também tornou possível comparar a eficácia das duas alternativas examinadas. Na Fig. 8 pode-se verificar que as unidades responsáveis pelo estrangulamento do desempenho são a *PU* e a *MSU*, conforme o número de *FU*s ativas, uma vez que o tempo médio de espera na *NSU* é constante e menor que o das outras unidades.

Em ambos os casos, a primeira deflexão encontrada não é visível nas curvas de tempo médio de espera. Ela decorre do início da formação de bolhas no "pipeline", o que provoca

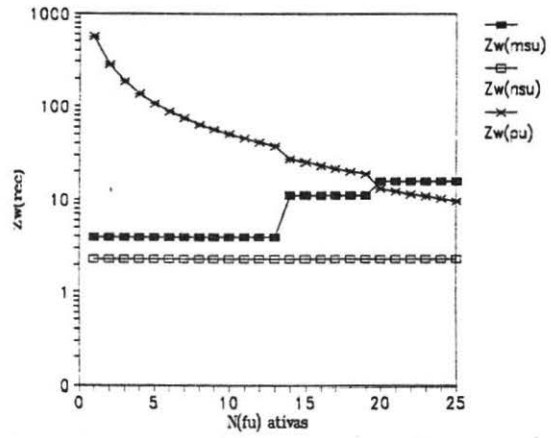

Figura 8: Tempo médio de espera para atendimento com as taxas de processamento da *MSU*, e *TQU* equalizadas e 2 centros de serviço na *NSU*.

uma queda no desempenho da *PU* e o desvia de 100%.

A eliminação do estrangulamento que existia na *NSU* simplesmente deslocou o gargalo para a *MSU* sem produzir um grande aumento da eficiência no uso das *FU*s. Para atacar este problema, decidiu-se variar o número de centros de serviço na *MSU* e acompanhar seu comportamento. Esta possibilidade foi examinada supondo-se a existência de um ou dois centros de serviço na *NSU*, mantendo-se equilibradas as taxas de processamento das unidades. As curvas de eficiência obtidas aparecem, também, na Fig. 5.

A Fig. 9 mostra que, dobrando-se apenas o número de centros de serviço na *MSU*, o tempo médio de espera para atendimento na unidade torna-se constante e menor que nas demais unidades. Desta forma o gargalo retorna para a *PU* ou a *NSU*, conforme o número de *FU*s ativas.

No segundo caso, quando se considera a existência de dois centros de serviço na *MSU* e na *NSU*, e são equalizadas as taxas de processamento, o gargalo é causado exclusivamente pela *PU*. Este arranjo corresponde à maior eficiência no uso das *FU*s, que permanece em 100% até 18 *FU*s ativas. Para quantidades maiores de *FU*s ativas continua a ocorrer um desvio devido à formação de bolhas no "pipeline", que deverá ser estudado em trabalhos posteriores.

As análises desenvolvidas até agora demonstram o potencial do modelo proposto. No entanto, também está previsto o exame de outras possibilidades, o que poderá exigir modificações no modelo de recursos desenvolvido. Entre as características que se pretende incorporar ao modelo encontram-se:

- O tratamento de variações no paralelismo durante a execução de um *GFD*.
- A avaliação do desempenho em função da porcentagem de fichas destinadas a vértices unários e do paralelismo do *GFD*.

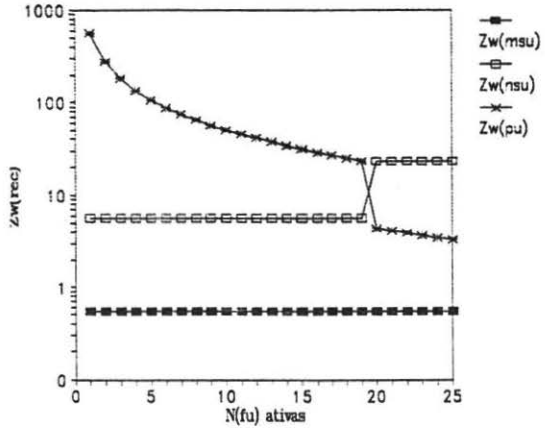

Figura 9: Tempo médio de espera para atendimento com as taxas de processamento da *MSU*, *NSU* e *TQU* equalizadas e 2 centros de serviço na *MSU*.

5 CONCLUSÕES E RECOMENDAÇÕES

A demanda por processamento de alto-desempenho levou ao desenvolvimento de modelos alternativos de computação e destes a novas arquiteturas de computadores. Em muitos casos, o projeto dessas arquiteturas e mesmo a construção dos respectivos protótipos aconteceram sem uma avaliação adequada dos fatores determinantes de sua viabilidade técnica e econômica.

É extremamente importante que a análise desses fatores seja realizada o mais cedo possível, de modo a evitar que possíveis inadequações atinjam estágios mais avançados do trabalho. Uma forma consagrada para se realizar esses estudos é através da criação e avaliação de modelos analíticos, dada sua eficácia e rapidez de desenvolvimento.

Wolf — o projeto do Grupo de Fluxo de Dados da Unicamp — tomou como ponto de partida o projeto da Máquina de Fluxo de Dados de Manchester (*MFDM*), a cujo desenvolvimento esteve intimamente ligado. O desenvolvimento da *MFDM* apoiou-se em um simulador e um protótipo, duas ferramentas com custo de produção, utilização e modificação bastante elevado, que oferecem limitada flexibilidade para a avaliação de alternativas de projeto. Essa pouca flexibilidade levou o grupo a tomar decisões de grande impacto em estágios primários do desenvolvimento, sem amparo analítico ou experimental adequado, o que acabou por acarretar deficiências na arquitetura final.

Este trabalho visou dotar Wolf de um modelo analítico, capaz de analisar a arquitetura da *MFDM*, buscando identificar seus pontos de estrangulamento. Para isso recorreu-se às redes de Petri, como opção à abordagem clássica por redes de filas.

A opção pelo uso de redes de Petri justifica-se pela sua adequação à modelagem de sistemas paralelos, por não exigirem o conhecimento de parâmetros muitas vezes indisponíveis na fase de projeto e pelas facilidades para seu aprendizado e análise.

Os resultados apresentados demonstram que, apesar das simplificações, o modelo representou de forma satisfatória a arquitetura e permitiu a identificação de alguns pontos

de estrangulamento. Para que se pudesse chegar a essas conclusões, os resultados obtidos foram comparados não só com os oferecidos pelo protótipo da *MFDM* [GKW85, GW83], mas também com os obtidos por Ghosal utilizando um modelo estocástico de redes de filas [GB90, GB87].

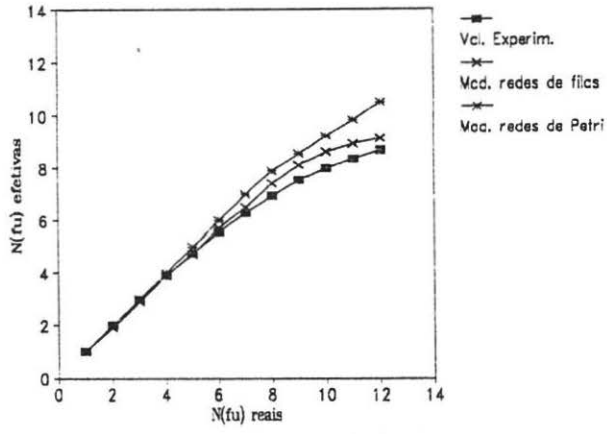

Figura 10: Comparação entre os resultados dos modelos de Redes de Filas e da Teoria Geral de Redes.

Em seu trabalho, Ghosal adotou simplificações que reduzem os problemas enfrentados pelas redes de filas no tratamento de problemas com tarefas paralelas. A Fig. 10 exemplifica um caso extremo das diferenças entre os resultados dos dois trabalhos. Observa-se que, para programas com baixos valores de paralelismo médio — no caso $PI = 15$ — o modelo-Q utilizado aqui apresenta um desvio do valor real significativamente mais alto, para um grande número de FUs ativas.

Isso pode ser justificado pelo fato de que, na análise do modelo de redes de filas, o valor do paralelismo médio é tomado como a média de uma distribuição normal, pois sua análise é estocástica. Por outro lado, no modelo de redes de Petri adotado, esse valor é tomado como exato e determinístico. Dessa forma, a análise do modelo de redes de filas consegue levar em conta a variação real existente no paralelismo de um programa, o que não ocorre na análise desenvolvida no atual estágio do modelo de redes de Petri.

É possível modelar a variação do paralelismo em um programa utilizando-se métodos aproximativos de análise. Nesse caso considera-se um certo conjunto de valores possíveis para o paralelismo médio e as respectivas probabilidades de ocorrência e ponderam-se os resultados para determinar o desempenho final da rede. Esse método, apesar de determinístico, leva em consideração a variação real do paralelismo do programa e pode aproximar-se mais da curva experimental. Essa alternativa deverá ser considerada em desdobramentos futuros visando ampliar o escopo da análise aqui apresentada.

Tanto a modelagem como a análise foram beneficiadas pela adequação da técnica adotada ao problema a ser resolvido. A validação do modelo demonstrou a necessidade de aperfeiçoamentos para os casos de baixo paralelismo médio. Este problema foi estudado e uma solução preliminar proposta, embora ainda não implementada.

[GB87] D. Ghosal and L. N. Bhuyan. Analytical Modeling and Architecture Modifications of a Data Flow Computer. *ACM Computer Architecture News*, 15(2):81–89, Fev. 1987.

[GB90] D. Ghosal and L. N. Bhuyan. Performance Evaluation of a Dataflow Architecture. *IEEE Transactions on Computers*, 39(5), Maio 1990.

[GKW85] J. R. Gurd, C. C. Kirkham, and I. Watson. The Manchester Prototype Dataflow Computer. *Communications of the ACM*, 28(1):34–52, Jan. 1985.

[GPK82] D. D. Gajski, D. A. Padua, and D. J. Kuck. A Second Opinion on Data Flow Machines and Languages. *IEEE Computer*, 15(2):58–69, Fev. 1982.

[GW80] J. R. Gurd and I. Watson. Data Driven System for High Speed Parallel Computing - Part 2: Hardware Design. *Computer Design*, 19(7):97–106, Jun. 1980.

[GW83] J. R. Gurd and I. Watson. Preliminary Evaluation of a Prototype Dataflow Computer. In *Proceedings of the* 9^{th} *IFIP World Computer Congress*, pages 545–551, Set. 1983.

[GWG78] J. R. Gurd, I. Watson, and J. R. W. Glauert. A Multilayered Data Flow Computer Architecture. Technical Report, Department of Computer Science, University of Manchester, Jul. 1978.

[HV87] M. A. Holliday and M. K. Vernon. A Generalized Timed Petri Net Model for Performance Analysis. *IEEE Transactions on Software Engeneering*, SE-13(12):1297–1310, Dez. 1987.

[KG86] K. Kawakami and J. R. Gurd. A Scalable Dataflow Structure Store. In *Proceedings of the* 13^{th} *Annual Symposium on Computer Architecture*, pages 243–250, Tokyo, Jun. 1986. ACM.

[MCB84] M. A. Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, Maio 1984.

[Rei85] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag, Berlin, 1985. 160 p.

[Rug87] C. A. Ruggiero. *Throtlle Mechanisms for the Manchester Dataflow Machine*. Ph.D.Thesis, Department of Computer Science, University of Manchester, Manchester, Jul. 1987.

[SK86] J. Sargeant and C. C. Kirkham. Stored Data Structures on the Manchester Dataflow Machine. In *Proceedings of the 13th Annual Symposium on Computer Architecture*, pages 235–242, Tokyo, Jun. 1986. ACM.

[Taz88] M. Tazza. *Análise Quantitativa de Sistemas*. III Escola Brasileiro-Argentina de Informática, Curitiba PR, 1988.

[TBH82] P. C. Treleaven, D. R. Brownbridge, and R. P. Hopkins. Data-Driven and Demand-Driven Computer Architecture. *ACM Computing Surveys*, 14(1):93–143, Mar. 1982.

Foram também identificados alguns pontos de estrangulamento da arquitetura e avaliadas algumas alternativas para sua eliminação. Dessa avaliação concluiu-se que será necessário aprimorar tanto a *NSU* quanto a *MSU* para que a arquitetura da *MFDM* possa explorar adequadamente o seu potencial teórico.

Em princípio, o problema encontrado na *NSU* é de ordem tecnológica e poderá ser resolvido com o emprego de memórias de tecnologia mais recente, com menor tempo de acesso. Viu-se, porém, que essa alternativa simplesmente deslocará o problema do gargalo para a *MSU*, onde a solução tornar-se-á mais complexa.

Na seqüência deste trabalho prevê-se o aperfeiçoamento do modelo para permitir o tratamento de paralelismo variável e a representação das unidades ora suprimidas, além da investigação de melhorias como a otimização da *MSU* e do sistema de comunicação.

AGRADECIMENTOS

Os autores agradecem ao Prof. Dr. Maurizzio Tazza pelo auxílio nos assuntos ligados à Teoria Geral de Redes, a Paulo Fernandes pelas discussões sobre modelagem de recursos utilizando as redes de Petri e ao Prof. Dr. Phillipe Navaux pelo acesso ao sistema SQ1, desenvolvido por seu grupo de pesquisa na UFRGS.

REFERÊNCIAS

[AGP78] Arvind, K. P. Gostelow, and W. Plouffe. An Asynchronous Programming Language and Computing Machine. Technical Report, Department of Information and Computer Science, University of California, Irvine, Cal., Dez. 1978.

[All80] A. O. Allen. Queueing Models of Computer Systems. *IEEE Computer*, 13(4):13–24, Abr. 1980.

[AT80] Arvind and R. E. Thomas. I-Structures: An Efficient Data Type for Functional Languages. Technical Memo 178, Laboratory for Computer Science, MIT, Cambridge, Mass., Jun. 1980.

[Cat81] A. J. Catto. *Nondeterministic Programming in a Dataflow Environment.* Ph.D.Thesis, Department of Computer Science, University of Manchester, Jun. 1981.

[Den85] J. B. Dennis. Models of Dataflow Computation. In M. Broy, editor, *Control Flow and Data Flow: Concepts of Distributed Programming*, pages 345–398. Springer-Verlag, Berlin, 1985.

[dS82] J. G. D. da Silva. *The Matching Unit of the Manchester Data-Flow Computer: A Pseudo Associative Store with Hardware Hashing.* Ph.D.Thesis, Department of Computer Science, University of Manchester, Jan. 1982.

[dS91] S. R. P. da Silva. Modelagem e Análise de Desempenho de uma Arquitetura de Fluxo de Dados. Dissertação de Mestrado, IMECC, UNICAMP, Campinas SP, Abr. 1991.

[Fer90] P. H. L. Fernandes. Avaliação de Desempenho de Interconexões de Processadores — Modelos Analíticos para Alocação Simultânea de Recursos. Dissertação de Mestrado. CPGCC-II, UFRGS, Porto Alegre RS, Dez. 1990.