TESE
998

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

Avaliação de Desempenho de Hardware Paralelo

MAURILIO PEREIRA COUTINHO

Dezembro / 98

Escola Federal de Engenharia de Itajubá

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca
MAUÁ
BIM

AVALIAÇÃO DE DESEMPENHO DE HARDWARE PARALELO

Maurílio Pereira Coutinho

Orientador: Prof. Germano Lambert Torres
Co-orientador: Prof. Luiz Eduardo Borges da Silva

Dissertação apresentada à Escola Federal de Engenharia de Itajubá, para obtenção do título de Mestre em Engenharia Elétrica.

Dezembro/98

"Dedico este trabalho à minha esposa
Pitty e aos meus filhos Lilian e Nino,
pela paciência e compreensão,
com muito amor."

SUMÁRIO

Resumo 6

Abstract 7

Glossário 8

Lista de Tabelas 11

Lista de Figuras 12

Capítulo 1 - Introdução e Objetivos

1.1 - Avaliação de Desempenho 14

1.2 - Tópicos Principais para o Desempenho 15

1.2.1 - Avaliação para Seleção 15

1.2.2 - Previsão de Desempenho 16

1.2.3 - Monitoração de Desempenho 16

1.3 - Avaliação de Desempenho de Hardware 16

1.4 - Avaliação do Desempenho de Máquinas Paralelas 17

1.5 - Sumário 18

Capítulo 2 - Instrumentação do Desempenho em Sistemas de Computação

2.1 - Introdução 20

2.1.1 - Especificação 20

2.1.2 - Coleta de Dados 21

2.1.3 - Análise de Dados e Visualização 21

2.2 - Especificação 22

2.2.1 - Métricas de Desempenho 23

2.2.2 - A Carga de Trabalho ("Workload") 25

2.3 - Coleta de Dados 27

2.3.1 - Metodologia 28

2.3.2 - Ferramentas para Medição de Desempenho 31

2.4 - Análise de Dados e Visualização 37

2.4.1 - Análise de Dados 38

2.4.2 - Visualização 39

2.5 - Instrumentação da Medida de Desempenho em Multiprocessadores 41
2.6 - Exemplos de Instrumentação para Monitoração 42
2.6.1 - Multiprocessdor Kendall Square Research KSR1 42
2.6.2 - Multiprocessador Stanford DASH 43
2.6.3 - Multiprocessador Cray Y-MP 45
2.7 - Sumário 46

Capítulo 3 - Arquitetura de Computadores Paralelos
3.1 - Introdução 48
3.2 - Hardware de Sistemas Computacionais Paralelos 49
3.2.1 - Níveis de Abstração 49
3.2.2 - Harwdare de Sistemas Computacionais 50
3.2.3 - Explorando o Paralelismo em Hardware 51
3.2.4 - Características de Desempenho 54
3.2.5 - Interconexões 56
3.2.6 - Modelo Unificado para Hardware Paralelo 60
3.3 - Classificação da Arquitetura de Hardware 66
3.3.1 - Taxonomia de Flynn 66
3.3.2 - Outras taxonomias 67
3.3.3 - Desenvolvimento de Sistemas taxonômicos para arquiteturas de Computador 70
3.3.4 - Taxonomia baseada na estrutura fisica dos sistemas 71
3.4 - Multiprocessadores e Multicomputadores 73
3.4.1 - Multiprocessador de Memória Compartilhada 74
3.4.2 - Multicomputadores de Memória Distribuída 78
3.4.3 - Multiprocessadores de Memória Virtual Compartilhada . 80
3.4.4 - Sistemas de Cache para Multiprocessadores 81
3.5 - Resumo 95

Capítulo 4 – Alguns Sistemas Multiprocessados
4.1 - Introdução 96
4.2 - Multiprocessador Vetorial Cray Y-MP 96
4.3 - Multiprocessador Stanford DASH 100

4.4 - Multiprocessador Kendall Square Research KSR1-32 103
4.4.1 - Características Gerais 103
4.4.2 - Elementos de Processamento 105
4.4.3 - Elementos de Memória 108
4.4.4 - Elementos de Interconexão 109
4.5 - Comentários 111

Capítulo 5 - Conclusões 112

Referências 115

RESUMO

Atualmente, processamento paralelo é uma área de pesquisa de grande interesse. O estudo das relações entre modelos diferentes para solucionar grandes problemas computacionais em vários computadores paralelos é essencial para o uso eficiente da tecnologia disponível. Avaliar o desempenho de sistemas existentes, capacita-nos a compreender estes sistemas e a ajustar os algoritmos para fazer melhor uso do hardware disponível. Usando esta informação, deficiências no projeto do hardware e do software podem ser identificados e registrados para o melhoramento de futuras gerações de máquinas. Portanto o estudo de avaliação de desempenho de sistemas paralelos tem um papel fundamental no processo de compreensão dos problemas de mapeamento das aplicações nestas máquinas. Esta dissertação apresenta como um estudo de avaliação de desempenho é caracterizado e delineia uma taxonomia de sistemas de hardware paralelo e um modelo de hardware paralelo que unifica a medida de desempenho e o tratamento dos vários grupos na taxonomia é introduzido. Três sistemas de hardware paralelo representando o estado-da-arte são descritos em termos deste modelo.

ABSTRACT

Parallel processing is currently the subject of a great deal of research. The study of the relationship between different approaches to solve large scale computational problems on the diverse parallel computers is essential for the efficient use of the available technology. Evaluating the performance of existing systems, both enable us to understand these systems and to tune algorithms to make better use of available hardware. Using this information, deficiencies in hardware and software design can be identified and earmarked for improvement in the future machine generations. Therefore, performance evaluation study of parallel computer plays a fundamental role in the process of understanding the mapping application problems on these machines. This thesis outlines a taxonomy of parallel hardware systems and a model of parallel hardware that unifies performance measurement and treatment of the various groups in the taxonomy is introduced. Three state-of-the art parallel hardware systems are described in the terms of this model.

Glossário

ALLCACHE: Conjunto de Memórias do KSR, comportando-se como um único Cache Global.

ALU : Unidade Lógica e Aritmética

Arrays: Combinação sequencial de módulos de hardware

Benchmark: Programa Utilizado para avaliação de Desempenho de um processador ou sistema computacional

Block Frame: Conjunto ou blocos de localizações sucessivas armazenadas no cache

Bottleneck: Ponto de qualquer sistema considerado como empecilho à melhoria do desempenho

C : Unidade de Controle

CAM : Contend Addressable Memory

CC-NUMA : Cache-Coherent- Non Uniforme Memory Access

CEU : Unidade de Co-Execução

Clock : Tempo de referência para sincronização das operações do processador

COMA : Cache-Only Memory

CPU: Unidade Central de Processamento

D: Unidade de Dados

DASH : Directory Architecture for Shared Memory

DDM : Data Diffusion Machine

DMA: Acesso Direto à Memória

DRAM: Memória RAM Dinâmica

E/S: Unidade de Entrada/Saída

FPGA: Field Programmable Gate Array

FPU: Unidade de Ponto Flutuante

Gate: Implementação de Bloco Lógico

Gate Array: Conjunto de gates utilizados para implementar uma lógica combinacional ou sequencial

Hit: Acesso à informação no Cache com sucesso

IBM : International Business Machines

IEEE: Institute of Electrical and Electronics Engineers

I-Memória: Memória de Instruções

IPU: Unidade de Processamento de Inteiros

ISC: Conexão Inter-Estágio

Junção PN: Região de depleção de um semicondutor

K: Unidade de Controle

Kernels : Trechos de programas reais usados para avaliação de desempenho

KSR: Kendall Square Research

L: Undade Lógica

LC: Cache Local

Log: Arquivo para registro de eventos

Loop Unroll : Descompactação de um laço de ciclo repetitivo

LRU: Least Recently Used

M: Unidade de Memória

ME: Elemento de Memória

MFLOPS: Milhões de Operações de Ponto Flutuante por segundo

MFLOP/s: Milhões de Operações de Ponto Flutuante por segundo

MHz: MegaHertz

MIMD: Múltiplas Instruções e Múltiplos dados

MIN: Rede de Múltiplos Estágios

MIPS: Milhões de Instruções por Segundo

Mi/s: Milhões de Instruções por Segundo

MISD: Múltiplas Instruções e Dado Único

Miss: Falha na operação de acesso ao cache

MMU: Unidade de Gerenciamento de Memória

Mp: Memória Principal ou Primária

NORMA: Non-Remote Memory Access

NUMA: Non-Uniform Memory Access

OP-Memória: Memória de Dados ou de Operandos

P: Processador

Pc: Processador Central

PCU: Unidade de Controle de Programa

PE: Elemento de Processamento

PMS: Notação Processador-Memória-Chave

Script: Programa em Linguagem de Comandos do Sistemas Operacional Unix

SIMD: Instrução Única e Múltiplos Dados

SISD: Instrução Única e Dado Único

SRAM: Memória RAM Estática

SVA: Endereço Virtual do Sistema

T: Transdutor

TLB : Translation Lookaside Buffer

UMA: Acesso à Memória Uniforme

WB : Write-Back

Workload: Conjunto de programas, comandos, scripts, etc. que formam a carga de trabalho para avaliação de um sistema computacional

WT: Write-Through

X: Transdutor

Xilinx: Fabricante de Gate Arrays

XIU: Unidade de Entrada/Saída

X-Window: Ambiente de Janelas

Lista de Tabelas

Tabela 2.1 - Comparação do Nível de Desempenho 23
Tabela 2.2 - Contadores da PMON 44
Tabela 2.3 - Grupo de Contadores HPM para CRAY Y-MP 47
Tabela 3.1 - Niveis Hierárquicos de Sistemas Digitais 50
Tabela 4.1 - Resumo de Configurações Típicas 111

Lista de Figuras

Fig. 2.1 : Experimento de Análise de Desempenho 20

Fig. 2.2 : Esquema Básico de um Monitor de Hardware 34

Fig. 2.3: Modelos de Medidas em hardwrae nos nós de sistemas paralelos 35

Fig. 2.4: Diagrama de Bloco Lógico do Monitor de Desempenho 45

Fig. 3.1: Notação Abreviada para Registro Lógico 51

Fig.3.2: Diagrama Função-Tempo de uma Estrutura Pipeline com n estágios . 53

Fig.3.3: Descrição de um sistema computacional no nível PMS 61

Fig. 3.4: Nível PMS com controle local 62

Fig. 3.5: Classificação de Flynn de arquiteturas de computadores 68

Fig. 3.5: Classificação de Flynn de arquiteturas de computadores (cont.) 69

Fig.3.6: Taxonomia Tera para Multiprocessadores e Multicomputadores 71

Fig.3.7: Representação da Transição de Estado 72

Fig.3.8: Sistema Genérico Multiprocessado 74

Fig.3.9: Modelos de Multiprocessador NUMA 76

Fig.3.10: Modelo de Multiprocessador COMA 77

Fig.3.11: Modelo Genérico de um Multicomputador com troca de mensagens . 79

Fig.3.12: Modelo de Memória Compartilhada Virtual 80

Fig.3.13: Organização de Cache com mapeamento direto 83

Fig.3.14: Organização de Cache Totalmente associativo 84

Fig.3.15: Organização de Cache Set Associativo 85

Fig.3.16: Inconsistência apresentada por uma variável compartilhada numa operação de escrita 87

Fig.3.17: Protocolos para caches coerentes: write-invalidate e write-broadcast . 89

Fig.3.18: Conceito básico de um esq. de coerência de cache baseado em diretório 90

Fig.3.19: Ordem de acesso a eventos de memória num sistema uniprocessado e num sistema multiprocessado 92

Fig.3.20: Modelo de consistência de memória sequencial 94

Fig.4.1: Organização do Sistema Cray Y-MP8I/8128 97

Fig.4.2: Diagrama lógico esquemático da rede crossbar entre 8 elementos de processamento e os 128 elementos de memória no Cray Y-MP8I/8128 100
Fig.4.3: Diagrama de bloco do protótipo DASH 2X2 101
Fig.4.4: Hierarquia de Memória da máquina DASH 103
Fig.4.5: Sistema de Memória ALLCACHE: Visão lógica e física 104
Fig.4.6: A arquitetura do KSR-1 com um anel para comunicação 105
Fig.4.7: Organização de Elemento de Processamento do KSR1 108
Fig.4.8: Processadores requisitante e replicante na mesma "Search Engine:0" . 110

Capítulo 1

INTRODUÇÃO E OBJETIVOS

1.1 AVALIAÇÃO DE DESEMPENHO

De acordo com Ferrari et al. [1], em sistemas de computação, o desempenho expressa a eficiência com a qual as máquinas conduzem suas funções, enquanto, a avaliação trabalha com as técnicas utilizadas com a finalidade de analisar quão eficientemente estes serviços são produzidos.

Hoje em dia, o desempenho é um dos fatores-chave que necessita ser levado em consideração nas diversas etapas dos processos computacionais. Desde a sua concepção, passando pelo seu desenvolvimento, configuração e ajuste, os sistemas de computação necessitam observar este quesito. E mais, é necessário uma avaliação quantitativa do desempenho do sistema durante todo o ciclo de sua vida útil.

Certas medidas de desempenho já podem ser consideradas como tradicionais, tais como:

- as taxas operacionais, dadas normalmente em MIPS (Milhões de Instruções por Segundo) ou em MFLOPS (Milhões de Operações de Pontos Flutuantes por Segundo),
- o mix de instruções ("instruction mixes"),
- a caracterização vetorial,
- o instruções "load" versus instruções "store",
- as entradas e saídas das subrotinas,
- os conflitos de bancos de memória e
- os atrasos na execução das instruções.

Muitas vezes, a instrumentação para coletar tais dados já está no lugar em alguns sistemas, possibilitando uma avaliação "on-line" do desempenho dos sistemas.

O principal desafio a ser encarado na avaliação de desempenho de sistemas computacionais é que os métodos necessários para esta avaliação devem acompanhar a evolução de projetos dos novos sistemas trazidos pelo rápido avanço tecnológico.

Enquanto a avaliação de desempenho do computador é de grande importância prática para os fabricantes e gerentes de centros de computação, ela se constitui também numa área de considerável atividade de pesquisa, tanto nas indústrias como nas universidades. O número de livros e artigos recentes, que tratam deste tema, atesta para o interesse nesta área [2,3].

1.2 Tópicos Principais para o Desempenho

Basicamente, existem três tópicos principais para o estudo do desempenho de um sistema de computação [4]: a avaliação para seleção, a previsão de desempenho e a monitoração de desempenho.

1.2.1 Avaliação para seleção

Neste caso, a escolha de um sistema ou de componentes de um sistema inclui o desempenho como o aspecto mais relevante dentro do critério de seleção. Entretanto, outros aspectos, tais como manutenção, expansibilidade ou obsolescência do equipamento são também significativos. O processo de seleção inclui a avaliação tanto de hardware como de software.

1.2.2 Previsão de Desempenho

A previsão de desempenho é orientada para avaliação da viabilidade do projeto. O objetivo é estimar a possibilidade de sua realização, seu desempenho e o binômio custo/impacto do projeto antes mesmo do desenvolvimento final e/ou da integração completa do sistema.

1.2.3 Monitoração do Desempenho

A monitoração fornece os dados de desempenho em um sistema já existente com o objetivo de identificar os possíveis gargalos, calculando, de antemão, o impacto do binômio custo/desempenho das mudanças sugeridas. A monitoração ainda pode ser conduzida com o propósito de se obter um perfil do uso corrente do sistema. Este procedimento pode predizer quando a capacidade do sistema alcançará níveis de desempenho insatisfatórios.

1.3 Avaliação de Desempenho de Hardware

O conceito de desempenho de hardware é importante durante a fase de projeto e influenciará toda hierarquia de um sistema: Isto ocorre desde o nível físico do semicondutor até a integração do sistema. Por exemplo, o tempo para drenar uma carga na junção PN no nível mais baixo de projeto digital pode estar relacionado hierarquicamente com o tempo de execução de uma ou mais instruções, sendo que um conjunto de instruções (estabelecendo uma aplicação particular) pode determinar o desempenho do sistema.

A hierarquia das medidas de desempenho pode ser concebida em quatro níveis: de aplicação, de algoritmo, de software de sistema e de projeto de hardware.

Assim, a determinação do desempenho do hardware fornece tanto a validação do projeto como orienta os desenvolvimentos em outros níveis. Por exemplo,

compreendendo os pontos fortes e fracos do sistema de hardware, os projetistas de software podem desenvolver uma implementação e uma interface com o usuário que maximiza o desempenho do hardware disponível para o usuário no final.

1.4 Avaliação do Desempenho de Máquinas Paralelas

As máquinas paralelas são fundamentalmente diferentes das máquinas convencionais. Assim, o seu desempenho não pode ser previsto através de uma simples analogia, ou através de extrapolações, como no caso das máquinas convencionais. Os sistemas tradicionais uniprocessados são caracterizados por arquitetura monolítica com dados processados seqüencialmente. Por outro lado, as máquinas paralelas têm características complexas e de paralelismo em vários níveis de implementação, tais como "pipelining", replicação de hardware, memória intercalada e técnicas de processamento concorrente. A complexidade desses sistemas se manifesta por si só, através do aumento na complexidade do desempenho como também da complexidade de seus programas.

A interação de elementos de hardware e de software na arquitetura dessas máquinas resultam em um grande espaço para o estudo do comportamento de desempenho dos sistemas e de seus potenciais gargalos [5]. E mais, isto também implica em uma quantidade enorme de testes que devem ocorrer, antes que esses sistemas possam ser construídos de forma viável e com um custo competitivo. Portanto, é necessário se ter ferramentas e métodos eficazes para identificar possíveis problemas e compreender as características de um dado sistema. Com isto, a instrumentação passa a ser uma característica essencial em sistemas paralelos, pois é através dela que são aquisitados os valores que permitem realizar todos os testes necessários.

1.5 Sumário

Esta dissertação está relacionada com a definição de uma estrutura experimental sobre a qual os estudos sistemáticos da avaliação de desempenho de sistemas computacionais podem ser realizados.

O processamento paralelo é atualmente um assunto de grande interesse de pesquisa. O estudo dos relacionamentos entre diferentes modelos para solucionar grandes problemas computacionais nos diversos computadores paralelos da atualidade é essencial para o uso eficaz das tecnologias disponíveis.

Avaliando o desempenho dos sistemas existentes, pode-se compreender o seu funcionamento, permitindo ajustar os algoritmos para fazer o melhor uso do hardware disponível. E mais, usando tais informações, as deficiências dos projetos de hardware e de software podem ser identificadas e refeitas no sentido de melhorar as gerações futuras da máquina. Por conseguinte, o estudo da avaliação de desempenho de computadores paralelos tem um papel fundamental no processo de compreensão dos problemas de aplicação de mapeamento dessas máquinas.

O Capítulo 2 inicia com a descrição de aspectos gerais da avaliação do desempenho. Em seguida, tópicos específicos relacionados à avaliação de desempenho de multiprocessadores são discutidos. Este capítulo conclui com a apresentação de diferentes técnicas e ferramentas de desempenho.

O Capítulo 3 examina os pontos fundamentais do hardware de sistemas paralelos. A classificação das máquinas paralelas baseadas na estrutura física do sistema é descrito utilizando um modelo unificado para hardware paralelo proposto. Finalmente, o paralelismo em multiprocessadores é analisado.

O Capítulo 4 descreve três exemplos de sistemas multiprocessados, a saber, Cray Y- MP, o Kendall Square Research KSR1 e o Stanford DASH, baseado no modelo unificado de hardware proposto nesta dissertação.

O Capítulo 5 discute outros trabalhos que podem ser desenvolvidos no futuro e apresenta as conclusões deste trabalho.

Capítulo 2

INSTRUMENTAÇÃO DO DESEMPENHO EM SISTEMAS DE COMPUTAÇÃO

2.1 INTRODUÇÃO

A análise de desempenho de um sistema de computação, inclui as seguintes tarefas: especificação das medidas desejadas, coleta de dados, análise dos dados e visualização, conforme apresentado na Figura 2.1.

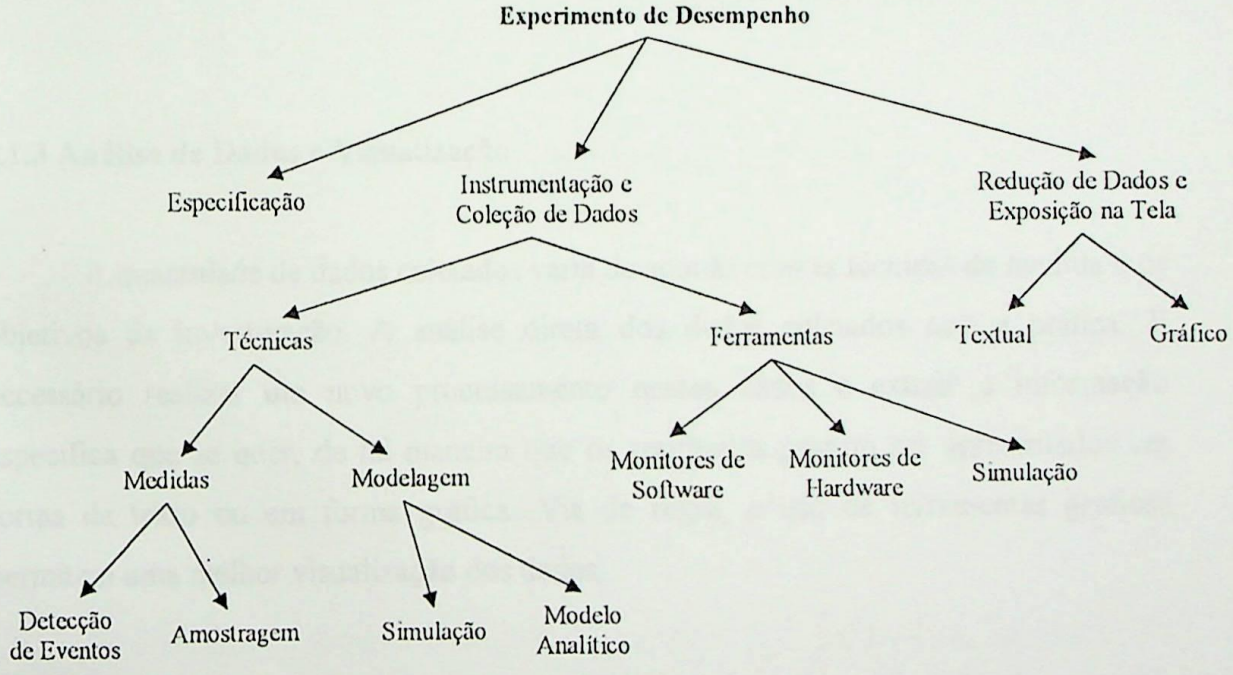


Fig. 2.1 - Experimento de análise de desempenho

2.1.1 Especificação

Após decidir o objetivo da avaliação de um sistema de computação, define-se um conjunto de índices de desempenho e as estratégias para se proceder a avaliação.

Também se deve incluir a definição de quais técnicas e ferramentas serão utilizadas. Alguns índices não podem ser obtidos diretamente, assim, é necessário especificar como esses podem ser derivados das medidas reais que foram coletadas pelas ferramentas de monitoração.

2.1.2 Coleta de Dados

O número elevado de parâmetros e o grande volume de dados envolvidos em computação paralela exigem que a instrumentação para coleta de dados seja inserida de forma seletiva e apropriada. Se o computador já está em operação, esta instrumentação pode ser em hardware ou software. Caso contrário, ele é instrumentalizado através do uso de simuladores. Em geral, os dados coletados são armazenados para os processamentos subsequentes.

2.1.3 Análise de Dados e Visualização

A quantidade de dados coletados varia de acordo com as técnicas de medida e os objetivos da investigação. A análise direta dos dados coletados não é prática. É necessário realizar um novo processamento nesses dados e extrair a informação específica que se quer, de tal maneira que os resultados possam ser apresentados em forma de texto ou em forma gráfica. Via de regra, o uso de ferramentas gráficas permitem uma melhor visualização dos dados.

Neste capítulo são discutidos os aspectos gerais da análise do desempenho instrumentalizada. Nesse contexto, diferentes técnicas e ferramentas são descritas. Em particular, a instrumentação de desempenho em sistemas paralelos é examinada. Finalmente, três sistemas de monitoração desenvolvidos no Sistema Kendall Square Research KSR1-32, multiprocessador Stanford DASH e Cray Y-MP são descritos.

2.2 ESPECIFICAÇÃO

Antes que o estudo da avaliação de desempenho seja conduzido, três questões básicas devem ser respondidas [6]:

1) "Que tipo de programa, dados e comandos devem ser utilizados? Que áreas são representadas? Quanto de trabalho está envolvido?".

2) "O que deve ser medido e que tipo de questões podem ser endereçadas pela execução deste conjunto 'programa, dados e comandos'?"

3) "Quanto de confiança pode-se ter nos resultados?"

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

Para responder a estas questões precisamos definir os valores das seguintes variáveis:

1. As características físicas do sistema que estão sendo avaliadas.

2. As condições de operação do sistema no momento em que a avaliação é feita. Em especial, as características da carga de trabalho ("workload") precisam ser conhecidas.

3. As métricas de desempenho do sistema.

O conjunto de programas, dados e comandos que os usuários submetem ao sistema computacional, a fim de conduzir um estudo de avaliação de desempenho, compreendem a carga de trabalho ("workload") do sistema. Portanto, as características de uma carga de trabalho e a métrica utilizada dependem dos objetivos de um estudo de avaliação de desempenho.

Tendo especificado os objetivos, os instrumentos a serem utilizados na próxima fase devem ser determinados. Nesse momento é possível especificar os parâmetros

necessários e a forma de sua representação, tais como, por exemplo: valores simples, distribuição, descritores da distribuição, entre outros..

2.2.1 Métricas de Desempenho

Não é fácil generalizar que métricas devem ser utilizadas com o objetivo de caracterizar o desempenho. A escolha depende dos níveis a serem considerados. Em sistemas de computação paralela, mesmo o modelo de computação utilizado pelo hardware é importante para a seleção da métrica do desempenho. A Tabela 2.1 mostra um subgrupo de uma importante métrica de desempenho para os três níveis: hardware, software e algoritmo, em três diferentes implementações de hardware de um modelo de computação: Cray X-MP, o sistema Cedar da Universidade de Illinois, e o Hipercubo Intel IPSC [7].

Tabela 2.1. Comparação do Nível de Desempenho

Nível	Cray X-MP	Illinois Cedar	Intel iPSC
Hardware	Vector start-up Conflitos de memória	Contenção da Rede Interação Vector/cache	Velocidade do processador Comunicação Latência e Largura de Faixa
Software	Compilador	Compilador	Suporte OS
Algoritmo	Vetorização	Acesso à memória compartilhada	Padrão de comunicação

Constitui erro comum quantificar o desempenho do sistema com um simples valor. Um certo número de métricas mais familiares têm sido utilizados com esse propósito pelos fabricantes e vendedores. Exemplos disto são:

- Avaliação do pico de desempenho ("Peak Performance") em Mi/S (Milhões de instrução por segundo) ou Mflops/S (Milhões de Operações de Ponto-Flutuante por segundo),

- Speed-up e
- Speed-up escalado.

Entretanto, o uso dessas métricas simples obscurecem a importância da interação dos níveis de desempenho e da dinâmica do equilíbrio [6,8,9,10]. A importância de escolher a métrica de desempenho correta que caracteriza cada nível de abstração é apresentada por Hockney, em [10]. Com este objetivo, as métricas de desempenho de hardware, que são derivadas da informação básica de temporização, são definidas a seguir.

Segundo Hockney, em [11,12], a descrição de dois parâmetros, r_∞ e $n_{1/2}$, para um "vector pipelines" é feita no sentido de se determinar quanto de trabalho útil é realizado durante o tempo de "vector start-up". O parâmetro r_∞ representa o desempenho máximo ou assintótico em Mflops/S. Para um sistema genérico isto ocorre assintóticamente para vetores de comprimento infinito, portanto o subescrito; Este desempenho máximo performance é caracterizado pela tecnologia usada no sistema computacional. O parâmetro $n_{1/2}$ determina o comprimento do vetor para se obter metade do desempenho máximo. Este parâmetro é uma medida da quantidade de paralelismo na arquitetura do sistema computacional e não é afetada por mudanças na tecnologia. O tempo t gasto para executar uma instrução vetorial em um vetor de comprimento n é dada por [11]:

$$t = r_\infty^{-1}(n+n_{1/2}) \qquad (2.1)$$

Uma relação similar é derivada para sistemas paralelos. Neste caso, deve-se introduzir a medida do custo de sincronização dos processadores de modo que as operações aritméticas sejam executadas na seqüência correta. Assim $s_{1/2}$ é definida como a quantidade de operações aritméticas que poderiam ter sido executadas (na máxima taxa de r_∞)durante o tempo gasto para sincronização. Se tomarmos t_0 como o tempo gasto para sincronização, o qual depende das facilidades de hardware fornecidas pelo sistema computacional e as facilidades de software fornecidas pelo sistema operacional e linguagem de programação, o parâmetro $s_{1/2}$ é igual a

$$s_{1/2} = r_\infty t_0 \qquad (2.2)$$

Isto significa a quantidade de operações aritméticas perdidas por causa da necessidade de sincronização. Assim como n e $n_{1/2}$, quando temos $s=s_{1/2}$, somente metade do tempo útil é gasto executando operações aritméticas. A outra metade do tempo é gasta em operações de sincronização. Este parâmetro é chamado de tamanho do grânulo para metade do desempenho. Relacionando estes parâmetros com t temos:

$$t = r_\infty^{-1}(S+S_{1/2}) \qquad (2.3)$$

Uma vez determinados os valores de métrica do desempenho, a análise qualitativa dos mesmos deve ser executada. Esta análise consiste em estabelecer se tais valores são bons ou necessitam ser melhorados.

2.2.2 A Carga de Trabalho ("Workload")

Os resultados da avaliação de desempenho são sensíveis à natureza da carga de trabalho ("workload"), que está sendo processado pelo sistema. Pequenas mudanças na definição da carga de trabalho podem, algumas vezes, levar a conclusões bastante diferentes [13]. A despeito dos objetivos do estudo da avaliação de desempenho, um experimento de desempenho precisa ser reproduzido para outros sistemas ou configurações. Portanto, os resultados dos estudos de avaliação de desempenho não podem ser expressados ou caracterizados por valores independentes da carga de trabalho.

A caracterização da carga de trabalho é o primeiro passo para qualquer estudo de avaliação de desempenho. Ela é usualmente feita em termos de parâmetros que podem

afetar o comportamento do sistema, tais como tempo de CPU, operações de entrada e saída (E/S), espaço de memória, conjunto de instruções, desvios ("branches"), entre outros [1,9]. Especificando os objetivos do estudo de avaliação de desempenho, é possível determinar os parâmetros que devem ser utilizados no experimento, bem como suas representações.

Uma carga de trabalho teste pode ser composta de aplicações, programas, passos de programas, kernels, ciclos, frações de código processado em modo escalar, scripts e comandos. Pode-se classificá-la de várias maneiras, dependendo das características a serem consideradas:

1. Carga de Trabalho Real: consiste de programas reais e seus dados processados durante um certo intervalo de tempo.

2. Carga de Trabalho Sintética: pode ser constituída de pequenos trechos de código extraídos de um programa real ou a mistura de trechos de programas reais e trechos construídos de maneira proposital. A primeira forma são chamado de kernels, tais como Livermore Loops [14,15] e o Linpack [16,17]. A segunda forma são chamados de benchmarks sintéticos, tais como Whetstone [18] e Dhrystone [19].

Os kernels são mais utilizados para isolar características individuais do desempenho de uma máquina e explicar as razões de diferenças de desempenho para programas reais. Benchmarks sintéticos tentam imitar a freqüência média de operações e operandos de uma grande variedade de programas. Diferentemente da carga de trabalho real nenhum usuário executa kernels e benchmarks sintéticos porque eles não computam coisa alguma que o usuário poderia usar. Estes benchmarks certamente são melhores que as taxas de pico cotadas pelos vendedores, mas eles ainda estão longe de descrever com precisão a totalidade de aspectos da arquitetura de grandes sistemas computacionais. É difícil ter total confiança na capacidade de poucas linhas de código ser representativo de dezenas de milhares de linhas de código, especialmente se o

algoritmo implementado não estiver nem mesmo sendo usado na carga de trabalho do sistema avaliado.

Embora a carga de trabalho real seja potencialmente mais representativa e mais fácil de implementar, na prática, elas são raramente utilizadas, devido principalmente a: limitada capacidade de reprodução dos padrões de recursos exigidos, longa duração das sessões de medida, falta de flexibilidade e portatibilidade e confidencialidade dos dados dos programas reais [4].

Entretanto, tem-se visto ultimamente iniciativas de grupos representativos de fabricantes de computadores e de universidades com o objetivo de reunir códigos de aplicações científicas relevantes que sejam independentes da instalação com o propósito de conduzir avaliação de desempenho. Como exemplo, pode-se citar que em 1988 o SPEC Consortium ("Systems Performance Evaluation Cooperative") foi formado com o objetivo de definir um benchmark baseado em aplicações similares para estações de trabalho de alto desempenho[20]. Um esforço semelhante, chamado "Perfect Club" ("Performance Evaluation for Cost-Effective Transformations") liga, ao mesmo tempo, universidades e companhias interessadas em computação paralela [21]. O Perfect Benchmark consiste em 13 programas oriundos de diversas áreas da engenharia. Cerca de 60.000 linhas de FORTRAN foram portados com sucesso para mais de 30 máquinas [22].

2.3 Coleta de Dados

A coleta de dados refere-se à metodologia e às ferramentas empregadas para obter o conjunto real de dados necessários para projetar o desempenho.

2.3.1 Metodologia

Os métodos pelos quais os índices podem ser obtidos são subdivididos em duas principais categorias: técnicas de medidas e técnicas de modelagem.

2.3.1.1 Técnicas de Medidas

A medição é realizada em duas operações chamadas de gatilhamento ("triggering") e coleta dos dados. O gatilhamento é a detecção de um evento pré-definido durante a execução do programa que faz com que a informação seja coletada. A informação pode ser coletada por rastreamento ("tracing") ou por amostragem.

A forma de rastreamento das informações envolve reportar todas as ocorrências de um evento no momento da sua ocorrência. A forma de amostragem relata um valor sob demanda do monitor.

1. Técnica de Rastreamento - Esta técnica é baseada na interceptação e gravação de todos os eventos de um dado tipo num certo momento. Quando este evento está associado a uma função do programa, diz-se que é um evento de software. Um evento desse tipo ocorre quando um programa alcança um certo estágio da sua execução, por exemplo, quando uma operação de entrada e saída (E/S) se inicia. Similarmente, um evento de hardware está associado com o aparecimento de um ou mais sinais no circuito de um componente do sistema independente da execução do programa. Esta técnica pode resultar numa quantidade de dados excessiva, que é diretamente proporcional ao número de tipos de eventos que serão interceptados e suas freqüências [23].

2. Técnica de Amostragem - A técnica da amostragem é usada sempre que a medida de todos os dados que caracterizam um evento seja impossível, impraticável ou

necessitar de um grande esforço computacional ou humano para sua programação. Ao invés de examinar o conjunto todo, o método analisa apenas parte dele. A partir daí, é possível inferir alguns parâmetros que são característicos do conjunto, com intervalo de tempos regulares, o sistema é interrompido para detectar o estado de alguns de seus componentes. Se o número de amostras coletadas for grande, esta técnica pode produzir resultados que sejam suficientemente precisos. A técnica de amostragem tem a vantagem de produzir um número de dados bem menor, abreviando e simplificando assim a sua análise.

2.3.1.2 Técnicas de Modelagem

Tais técnicas não requerem a existência ou disponibilidade de um sistema para ser avaliado. Essa característica é importante, por exemplo, durante a fase do projeto de um sistema ou para determinar o efeito da modificação do sistema antes que a mudança seja feita (previsão de desempenho). Por outro lado, têm uma desvantagem fundamental: como utilizam modelos, que são inevitavelmente representações parciais e aproximadas da realidade, elas fornecem resultados cuja credibilidade é duvidosa, ao menos até que os argumentos válidos que suportem a precisão do modelo sejam fornecidos. Há dois tipos de técnicas de modelagem: modelos de simulação e modelos analíticos.

1. Modelos de Simulação - O modelo de simulação reproduz no tempo certos aspectos do comportamento dinâmico de um sistema, permitindo medidas "reais" no modelo através de ferramentas de simulação. Isto permite a compreensão, pelo menos parcial, do funcionamento do sistema [24]. Uma vez que o modelo seja construído, pode ser tratado como um sistema real. A precisão de um modelo depende do nível de detalhes usados para representar o sistema. Assim, o modelo deve incluir todos os aspectos relevantes do sistema, com grau de detalhe compatível com a precisão exigida. Quanto mais detalhes incluídos, mais preciso o modelo será e mais caro computacionalmente será construí-lo e usá-lo.

Um processo de simulação utiliza "traces" de aplicações reais que podem ser manipulados para corresponder com a configuração específica ou a uma carga de trabalho a ser avaliada ("Trace Driven Simulation"). Em um outro modelo, as operações reais de um sistema computacional são estabelecidas e a distribuição de probabilidades são usadas para descrever o desempenho dos componentes do sistema a ser avaliado.

"Trace Driven Simulation" é uma das técnicas mais usadas para avaliação de arquiteturas. Em particular, tem sido de muito valor para analisar memória cache [25,26]. Os dados com o qual uma simulação de cache é conduzida, consiste em um rastreamento de todas as referências de instruções e dados obtidos com a execução de um ou mais aplicações.

A simulação fornece excelentes resultados para avaliação de alternativas para uma nova máquina ou software, mas o esforço e o custo computacional de preparar os simuladores para um sistema específico é, em geral, inviável. Isto é especialmente verdadeiro para arquiteturas paralelas. Entretanto, para projetar desempenho nos sistemas propostos, o custo da simulação é geralmente razoável, dado à extensão de tal empreendimento.

2. Modelos Analíticos - É a representação matemática de um sistema de computação. Tais técnicas estão se tornando bem populares porque são relativamente de baixo custo, de aplicabilidade geral e de uso fácil e flexível. A clássica Teoria das Filas estende a classe dos sistemas que podem ser modelados por técnicas analíticas. Por exemplo, Browne e outros propõe, em [27], o uso de modelos baseados em redes de Petri para introduzir abstrações na representação das estruturas de controle na modelagem de estruturas de computação paralela.

   Entretanto, nem todos os problemas podem ser resolvidos por esta técnica. A necessidade para o modelo ser matematicamente solucionável limita o grau de detalhamento do modelo e, consequentemente, a exatidão do modelo analítico. Nesse caso, a simulação ainda permanece como a única técnica adequada de modelagem.

2.3.2 Ferramentas para Medição de Desempenho

As ferramentas de medição são dispositivos usados para quantificar os dados coletados durante o processo de monitoração. Tais ferramentas são primariamente orientadas para analisar o comportamento de um sistema existente a fim de melhorar o seu desempenho. Outras razões incluem a determinação das características de uma carga de trabalho e fornecimento de dados para os modelos de sistema. O nível de sofisticação destas ferramentas pode variar de uma simples observação humana da máquina a um sofisticado hardware, um conjunto software/firmware, ou monitores híbridos. Estes são conectados ao sistema através de interface de instrumentação apropriada. Estas interfaces podem ser seções de código inseridas no programa, temporizadores, registros, sinais de barramento no bastidor ("backplane"), e assim por diante.

É sabido que qualquer instrumentação perturba, de alguma forma, o desempenho do sistema em estudo. A extensão desta perturbação é determinada pelo volume das informações coletadas, pelo hardware de monitoração, pelo software do sistema, pela arquitetura do sistema, e pelo paradigma de programação [28].

As ferramentas de medição podem ser divididas em três categorias:

- ferramentas de pré-processamento ("pre-run tools"): tais como simuladores, usados para projetar o desempenho,
- ferramentas de processamento em tempo real ("run-time tools"): tais como os monitores para coletar medidas em tempo de execução com o objetivo de predição ou melhora do desempenho,
- ferramentas de pós-processamento ("post-run tools"): tais como ferramentas de modelagem, para exibir os dados coletados e/ou usar tais dados para previsão ou melhoraria do desempenho.

2.3.2.1 Programas Monitores

Os programas monitores têm a capacidade de registrar os eventos dos componentes do sistema. Exemplos desses registros incluem arquivos do tipo log ("log-files") para contabilidade do sistema, rotinas de interrupção-interceptação e sistemas de amostragem. O uso de tais ferramentas requer a inserção de códigos embutidos, chamados "sondas de software" ("software probes") em pontos críticos do sistema operacional, dos ambientes de execução ou nos programa de aplicação. Devido ao fato de serem rotinas estranhas a estes sistemas, os programas de monitoração aumentam a carga de trabalho e, consequentemente, alteram seu desempenho.

O nível em que a instrumentação é colocada, determina o tipo de informação que pode ser coletada e exibida. A instrumentação do sistema operacional pode ser usada para coletar informações, tais como mensagens enviadas e recebidas, criação de processo, escalonamento, "page-fault", chaveamento de contexto, acesso à memória e chamadas do sistema. A instrumentação do ambiente de execução pode fornecer informações que dizem respeito ao estado das filas em execução, à obtenção e liberação de "locks", à entrada e a saída de sessões paralelas, à chegada e partida das barreiras ("barriers"), ao despacho de conjunto de ciclos e às chamadas e retorno de procedimento. Este tipo de informação pode ser usado para associar gargalos de desempenho com sessões específicas do código da aplicação e, também, visualizar aspectos do próprio ambiente de execução. A instrumentação de aplicações podem fornecer a associação entre eventos do sistema operacional e porções particulares do código de fonte. De forma adicional, pode fornecer informações sobre eventos e construções de alto nível nos programas de aplicação.

A inserção de sondas de software pode ser automatizada, tal como no sistema MPtrace [29], que automaticamente modifica o programa em estudo para produzir dados de rastreamento ("trace"), enquanto ele é executado. Similarmente, o sistema IPS [30] não exige modificação do programa do usuário; toda instrumentação é automaticamente inserida durante o tempo de compilação.

2.3.2.2 Técnicas de Hardware

As técnicas de hardware são ferramentas não intrusas que contêm dispositivos eletrônicos a serem conectados a pontos específicos do sistema onde eles detectam de forma passiva os sinais que caracterizam o fenômeno que está sendo observado. A principal característica dos monitores de hardware é que eles são independentes e externos ao sistema que está sendo medido. Diferentemente dos programas monitores, a interferência provocada pelo uso de monitores de hardware numa medida de desempenho é mínima. O uso de ponta de prova de alta impedância conectada a um ponto específico do sistema torna esta perturbação desprezível para o sistema em estudo.

Os monitores de hardware são capazes de registrar uma larga variedade de eventos de hardware e software, mas são limitados em sua habilidade de relatar atividades internas de um programa que está sendo executado e de detectar as causas para o conjunto de respostas.

A Figura 2.2 mostra a estrutura de uma ferramenta de hardware simples. O dispositivo detecta sinais e os passa para módulos lógicos, os quais realizam operações lógicas seqüenciais e combinacionais para derivar o sinal a ser medido ("pattern matching"). Os resultados são enviados para contadores, que podem contar pulsos representando eventos elementares ou gravar o tempo entre dois eventos elementares. Periodicamente o sistema operacional lê o conteúdo dos contadores e os armazena em localizações de memórias especiais ou registros da CPU. Subseqüentemente, estas localizações poderão ser acessadas pelos programas monitores.

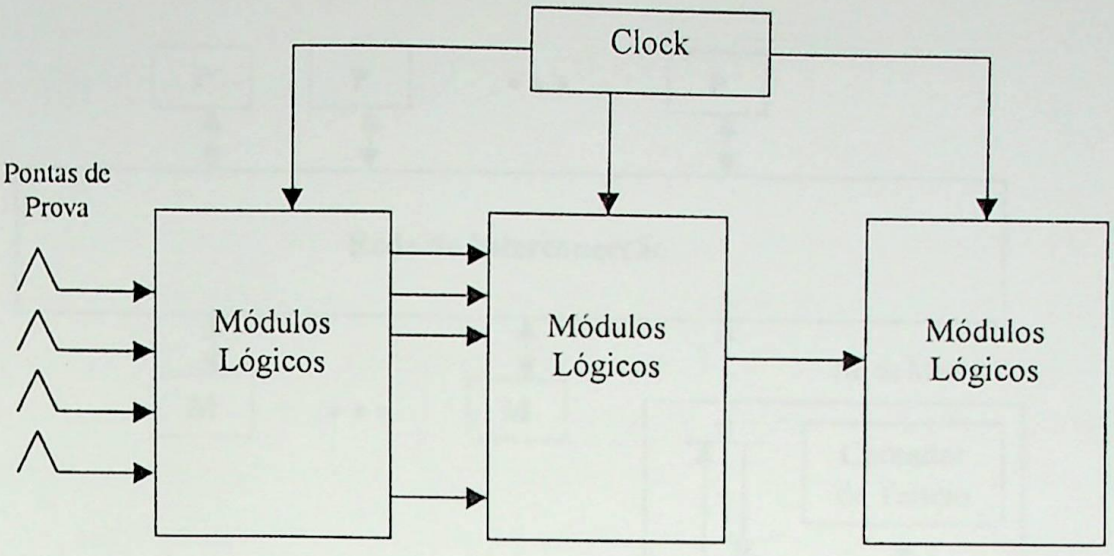
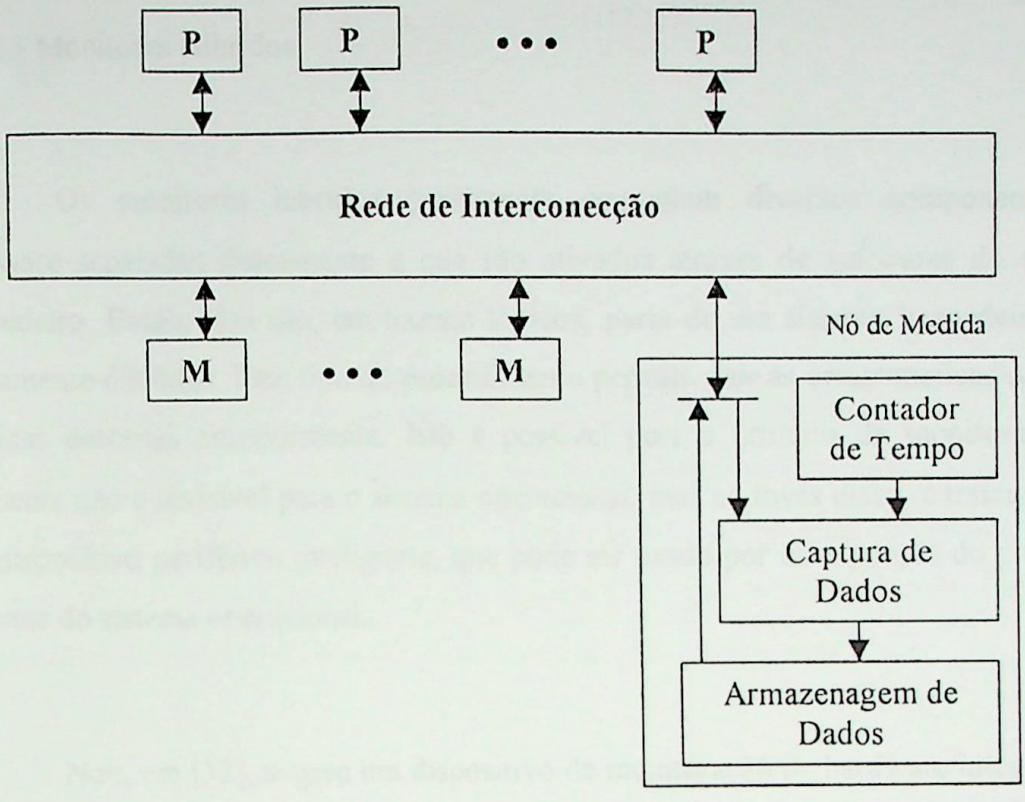


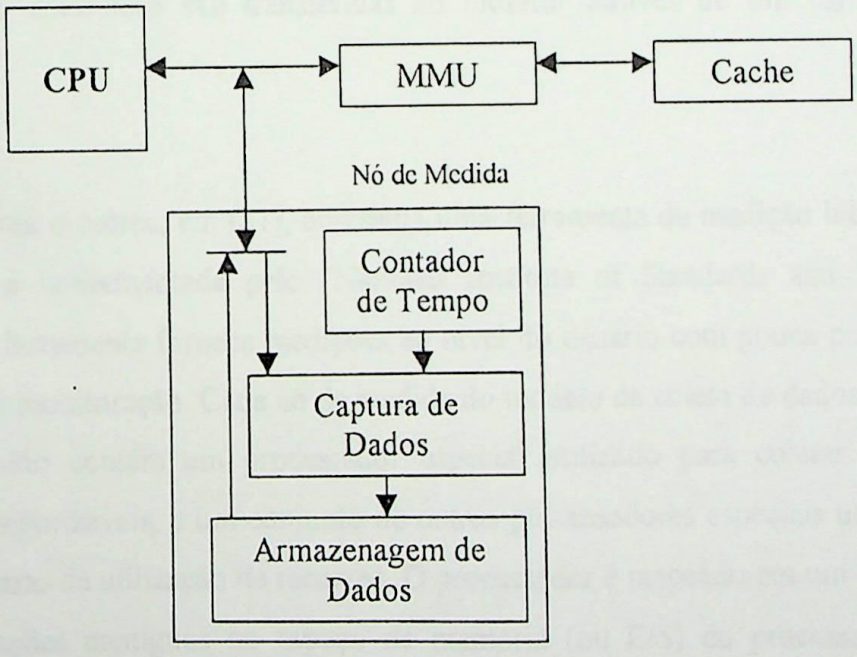
Fig. 2.2 - Esquema básico de um monitor de hardware

O registro de eventos de hardware de uma máquina paralela pode ser centralizado ou distribuído [31], como mostrado na Figura 2.3. No esquema centralizado, o sistema de monitoração coleta todos os dados; enquanto, em um esquema distribuído, os eventos são capturados localmente em cada processador.

As informações que podem ser obtidas no nível de hardware incluem medidas como Mi/s, Mflop/s, amostras do contador de programa, TLB e cache misses, tráfego de memória, amostras de endereços de dados e instruções. Por exemplo, um processador de monitoraração de desempenho foi projetado para o IBM Research Parallel Processor Prototype (RP3) para coletar tais informações [32].



(a)



(b)

Fig. 2.3 - Modelos de medidas em hardware nos nós de sistemas paralelos:

(a) centralizado, nó de medida acessado globalmente

(b) nó de medida conectado a CPU e by-passing da MMU e do cache

2.3.2.3 Monitores Híbridos

Os monitores híbridos geralmente empregam diversos componentes de hardware separados fisicamente e que são ativados através de softwares do sistema hospedeiro. Então, eles são, em termos lógicos, parte de um sistema hospedeiro, mas fisicamente distintos. Este tipo de procedimento permite unir as características das duas técnicas descritas anteriormente. Isto é possível pois o artifício de monitoração de hardware não é invisível para o sistema operacional, mas ao invés disso, é tratado como um dispositivo periférico inteligente, que pode ser usado por uma porção do programa monitor do sistema operacional.

Nutt, em [32], sugere um dispositivo de monitoração de hardware integrado ao sistema hospedeiro, como um dispositivo periférico. Nesta implementação, um minicomputador é usado para controlar as funções do monitor de hardware enquanto as informações adicionais são transferidas ao monitor através de um canal de dados dedicados.

Mink e outros, em [31], apresenta uma ferramenta de medição híbrida. Ela foi concebida e implementada pelo "National Institute of Standards and Technology" (NIST). A ferramenta fornece medições ao nível do usuário com pouca perturbação do sistema sob monitoração. Cada nó de medida do modelo de coleta de dados centralizado ou distribuído contém um processador especial utilizado para coletar amostras de eventos temporizáveis, e um conjunto de outros processadores especiais utilizados para obter amostras da utilização de recursos. O processador é mapeado em um certo número de localizações contíguas no espaço de memória (ou E/S) do processador medido, fornecendo o acesso adequado aos registros, contadores e as especificações de comando.

2.3.2.4 Simuladores

Um simulador é uma ferramenta que modela o comportamento dinâmico do sistema para um determinado período de tempo, que é dividido em intervalos discretos e ajustáveis [1].

Existem dois tipos de simuladores: os comerciais e os de propósito especial. Os simuladores comerciais são distribuídos por companhias especializadas e adaptáveis para uma grande quantidade de sistemas, configurações e aplicações. Os simuladores de propósito especial são concebidos e construídos para fins muito mais específicos que aqueles dos simuladores comerciais. Este é o caso das máquinas paralelas.

So e outros, em [34], descrevem um simulador de desempenho para multiprocessadores, chamado PSIMUL, usado para compreender a execução de programas paralelos em nível de instrução. Este simulador foi inicialmente utilizado na análise de desempenho durante o desenvolvimento da máquina RP3 [35].

2.4 Análise de Dados e Visualização

A correta interpretação do desempenho de um sistema não depende somente da facilidade com que o usuário pode compreender e lidar com as diversas características dos sistemas de avaliação do desempenho, nem somente da habilidade das ferramentas de instrumentação e da coleção de dados. Há de se considerar a análise dos dados de desempenho e as ferramentas de apresentação que são devotadas ao problema de identificação de áreas e componentes que requerem atenção.

2.4.1 Análise de Dados

Uma vez que a fase de coleta dos dados tenha produzido uma lista de eventos, pode-se iniciar a sua análise. Este processo inclui o cálculo de estatísticas, a ordenação dos eventos e a detecção de eventos abstratos de mais alto nível a partir da lista de eventos de mais baixo nível.

Os eventos coletados podem ser um conjunto de instruções do cache, um procedimento de chamada de função, um acesso a uma variável compartilhada ou qualquer evento de alto nível definido do usuário.

A análise efetuada no registro de eventos coletados pode ser tão simples, quanto uma conversão de formato, ou tão complexa, quanto a detecção e ordenação de eventos abstratos de alto nível a partir de uma lista de eventos primitivos. Muitos sistemas realizam análises de dados como preparação para armazenagem do histórico do evento em alguma forma, tais como gráficos hierárquicos ou bases de dados relacionais. Alguns sistemas podem simplificar o processo de análise através do pré-processamento de arquivos de rastreamento ("trace files"), determinando parâmetros como escala de tempo e número de processadores. Naturalmente, isto pode ser feito apenas em sistemas depois do processamento ("post-mortem systems"). Em geral, o número de análises que podem ser efetuadas em tempo-real pelos sistemas é limitado devido a limitações de tempo de execução.

Os sistemas de avaliação de desempenho analisam os dados coletados sem nenhum tratamento prévio, com a finalidade principal de determinar a métrica e a estatística do desempenho. Adicionalmente, os dados de desempenho podem ser analisados para propósitos de mudanças on-line nos programas paralelos ou distribuídos afim de melhorar seu desempenho. A extensão desta análise é limitada pela relação entre o custo de tal operação e o ganho de desempenho conseguido pelas adaptações dela resultantes.

2.4.2 Visualização

O propósito da visualização pode ser a correção ("debugging") de um programa ou sistema, a avaliação e otimização de seu desempenho ou o fornecimento de uma documentação eletrônica através de uma estrutura de dados na tela, programas de animação ou outro meio de gráfico que permite vizualizar o conhecimento do funcionamento de um sistema, programa ou algoritmo. Kraemer e Stasko [28] definem o termo visualização de computação paralela ("parallel computation visualization") como sendo a visualização gráfica do comportamento e desempenho de computadores paralelos e programas de computadores.

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

O processo de visualização está se tornando, cada vez mais, uma ferramenta efetiva para compreender o comportamento de um sistema e/ou a execução de um programa, para auxiliar na correção de problemas e na avaliação de desempenho. A qualidade das interfaces gráficas, cada vez mais utilizando recursos como gráficos 2D ou 3D e diagramas de estado de transição, permite aos usuários localizar problemas ou gargalos do sistema. Por outro lado, os antigos relatórios em forma de texto, descrevendo a execução de um sistema de computador paralelo, a despeito de serem seqüenciais, podem ser de difícil assimilação e interpretação.

Diversos sistemas de avaliação de desempenho fornecem na tela também informações estatísticas. Por exemplo, as ferramentas Tapestry [7] e Paragraph [36] fornecem estas informações utilizando gráfico de barras, diagramas de Kiviat[1] e matrizes de visualizaçâo.

Tang, em [23], descreve e implementa duas ferramentas de monitorização para o sistema "Manchester Dataflow Machine". Elas apresentam a monitorização de dados através de gráficos e telas animadas usando os recursos do sistema Sun View. A primeira ferramenta, chamada Tree, executa a monitorização ao nível do programa, mostrando na tela as diversas etapas de sua execução em uma máquina paralela. A

[1] O diagrama de Kiviat permite aos usuários conhecer o grau de utilização de cada processador [1,33].

segunda ferramenta, denominada Graph, faz a monitoração ao nível da máquina, mostrando os diversos estados da máquina paralela.

Mallory e Reed, em [7], descrevem uma ferramenta de visualização do desempenho pós-execução ("post-run"), baseada em técnicas de hipervisualização ("hyperview"). É usado para processador paralelo distribuído configurado como um hipercubo. A implementação é baseada no sistema X-Windows e permite a apresentação na tela simultaneamente das diversas partes do sistema via uma variedade de diferentes vistas.

Em [37], uma ferramenta de monitoração gráfica, denominada MIN-Graph, baseado no sistema X-Windows e implementada nas máquinas BBN GP1000 e BBN TC2000 é descrita. Min-Graph é um sistema integrado de coleção de dados, análise e ambiente de visualização com a finalidade de fornecer ao usuário uma idéia do desempenho dinâmico de um sistema multiprocessado baseado na operação MIN ("MIN- based multiprocessor system").

Kohn e Willians, em [40], apresentam a ferramenta ATExpert, desenvolvida pela CRAY Research, Inc. Ela permite exibir o desempenho de programas paralelos. Trabalhando atrás das telas estão diversos sistemas especialistas, os quais simulam a execução de um programa e ajudam ao usuário a compreender do desempenho de um programa paralelo. Esta ferramenta descreve visualmente o desempenho paralelo em todos os níveis de um programa, guia o usuário na identificação de áreas de problemas no código de fonte, explica porque os problemas estão ocorrendo e sugere ações para resolvê-los.

Kitchin and O'Neill, em [39], apresentam o Gist, que é uma ferramenta de gravação de eventos com orientação gráfica, desenvolvido para as máquinas BBN Butterfly com memória compartilhada hierarquizadas. Ele é um programa para a monitoração de software, que gera arquivos do tipo log dos eventos dos programas significativos, os quais podem ser exibidos graficamente como rastreamento

simultâneos de eventos para cada processador. O Gist possui três características principais: uma coleção de funções para registro de eventos para programa em C ou em FORTRAN BBN Butterfly; uma função que transfere os registros de eventos para o computador de interface ("front-end host") e um programa que exibe na tela o histórico de eventos através de um ambiente X-Windows.

2.5 Instrumentação da Medida de Desempenho em Multiprocessadores

Diferentemente dos sistemas uniprocessados, em um sistema multiprocessado, a relação de tempo e de utilização de recursos entre os vários caminhos de instruções, que contribuem para a solução de um problema, devem ser mantidos. Chamadas para o serviço de medidas de um sistema operacional poderia atrasar um processador, permitindo que outros prossigam, perturbando seriamente a execução. A soma das perturbações que podem ser toleradas está relacionada ao tamanho da granularidade do programa, isto é, o quanto de computação é feita entre os tempos em que os processos precisam interagir e que medidas devem ser tomadas. Em geral, quanto mais longo esse período, maior o tempo de perturbação que pode ser tolerado. Vários modelos de monitores de hardware e software para medir o tempo de execução e a utilização de recursos para sistema multiprocessados estão apresentados por Carpenter, em [40]. Nesta implementação são descritos contadores de taxas, tais como cache hit, ocupação de barramento, tempo morto do processador ("processor idle"), e assim por diante, usando um monitor de hardware.

Um importante aspecto de qualquer sistema de instrumentação é a habilidade para ordenar eventos para visualização. A coleção de dados de programas paralelos diferem de coleção de dados de programas seriais, pois nos primeiros existe uma seqüência múltipla de eventos. Estas seqüências precisam ser ordenadas e agrupadas. Marcadores de tempo ("timestamps") são freqüentemente usados com o objetivo de ordenar os eventos. Se a análise e os estágios de apresentação na tela necessitam dos

marcadores de tempo, estes valores precisam ser precisos e consistentes. O sistema de hardware para capturar e gravar os traces dos programas gerados pelo Intel iPSC/2 Hypercube, chamado HYPERMON, de acordo com Mallony e Reed, em [41], tem a capacidade de aceitar, simultaneamente, eventos de todos os nós do sistema e atribuir os marcadores de tempo para a gravação dos eventos.

2.6 Exemplos de Instrumentação para Monitoração

2.6.1 Multiprocessador Kendall Square KSR1

O Kendall Square Research KSR1 é um multiprocessador de memória compartilhada escalavel, disponível comercialmente, o qual é formado como uma hierarquia de anéis [42]. Cada anel contém 32 processadores. Cada processador contém uma unidade chamada de monitor de evento ("event monitor unit – EMU"), construído para registrar vários eventos e intervalos. Ele é conectado a vários sinais de barramento em cada processador. O trabalho da EMU é contar eventos e lapsos de tempo relativos as atividades do sistema de memória que não são diretamente visíveis para o processador. Ele incrementa esses contadores quando os eventos ocorrem. O sistema operacional verifica a unidade através de interrupções temporizadas, afim de realizar a leitura dos contadores da EMU e atualizar os seus contadores internos de 64-bits.

Os tipos de eventos que são registrados incluem cache hits / misses locais e quão distante na hierarquia um pedido tem que percorrer para ser satisfeito. O EMU também acumula o número de ciclos do processador envolvido em tais eventos. Esses contadores podem ser lidos em pontos apropriados do código e ajudar a caracterizar ninhos de ciclos ou outras sessões de códigos. Como os eventos são contados por hardware, o custo desta medida é baixo. Pelo fato de que estes eventos são monitorados por cada processador (modelo distribuído), torna-se fácil para o usuário definir o tipo de paralelização que melhor se enquadra a sua aplicação e otimizar a localidade de referência.

A biblioteca KSR de monitorização do desempenho, PMON, fornece a interface para o EMU, para os registros de hardware, e para o kernel do sistema operacional, afim de obter dados de desempenho. As rotinas da biblioteca PMON garantem acesso para o conjunto de dados da própria PMON e retornam um buffer com 18 contadores como mostrado na Tabela 2.2. Os dados da PMON fornecem a possibilidade de uma análise mais aprofundada do desempenho de um programa no KSR1. O objetivo é fornecer uma fotografia dos dados relativos aos desempenhos individuais, de tal maneira que os vários pedaços formem um quadro consistente do desempenho global.

É importante mencionar que esses contadores incluem dados a cerca das próprias rotinas da biblioteca PMON e sobre as atividades do sistema operacional durante a execução da aplicação. Estas informações são mantidas numa base "per-thread" e o kernel cuida da armazenagem e restauração dos dados da PMON durante o contexto de chaveamento de um processo para outro. Entretanto, os contadores continuam a registrar os eventos, enquanto as atividades do sistema operacional e do kernel continuam a acontecer, incluindo os serviços de interrupção.

2.6.2 Multiprocessador Stanford DASH

A máquina Stanford Dash é um multiprocessador de memória compartilhada, escalável, com protocolo de cache coerente baseado em diretório distribuído, que consiste em um conjunto de nós de processamento (agrupamento) conectados através de uma malha de rede de interconexão. Por sua vez, cada nó de processamento é um multiprocessador baseado em barramento. O protótipo consiste em 64 processadores de 33 MHz MIPS R3000/R3010, entregando até 1600 Mi/s e 600 Mflop/s escalar [43].

Tabela 2.2 - Contadores do PMON [64]

Contador	Nome	Fonte	Descrição
1	iuser_clock	%X register	No. of User Clock Cycles
2	iwall_clock	%X register	No. of Elapsed Clock Cycles
3	iinstruction_counter	computed	Not Available
4	iceu stalls time	Event Monitor	Total time CEU stall
5	ixiu ins inst time	Event Monitor	Total time of inserted instructions initiated by XIU
6	icache ins inst time	computed	Total time of inserted instructions initiated by CCU
7	ipg hit	Event Monitor	No. of Cache page hits
8	isp hit	Event Monitor	No. of Cache subpage misses
9	ipg miss	Event Monitor	No. of Cache page misses
10	isp miss	Event Monitor	No. of Subcache misses
11	isp miss time	Event Monitor	Total time of subpage misses
12	idsc miss	Event Monitor	No. of data subcache misses
13	iael	Event Monitor	No. of packets that travelled AE:1
14	iprefetches	Event Monitor	Total No. of Prefetches
15	iprefetches miss	Event Monitor	Prefetches of data which were not in Cache
16	iprefetch prt hit	Event Monitor	Prefetches which hit a full prt
17	ithrea migração	Kernel	No. of times the thread migrated between cells

Para possibilitar maiores informações sobre a execução das aplicações no protótipo, cerca de 20% da placa de controle do diretório ("directory control board") em cada agrupamento foi dedicado ao monitor de desempenho de hardware [44]. A integração do monitor de desempenho com a lógica do diretório permite medidas não invasivas do sistema completo, sem um hardware extra para teste. O desempenho do hardware é controlado por software (procedimento híbrido) e fornece informações sobre as características de referência do software e da utilização dos recursos de hardware. O monitor de hardware pode rastrear e contar uma variedade de eventos dos barramentos, diretórios e da rede. A seleção de evento é controlada por um gate array Xilinx (FPGA) programável por software. Isto permite uma seleção flexível de eventos e um pré-processamento sofisticado de eventos.

O diagrama de bloco da lógica de desempenho é mostrado na Figura 2.4. Consiste de três blocos principais. O primeiro, o FPGA, seleciona e pré-processa os

eventos a serem medidos e controla o resto da lógica de desempenho. O segundo bloco, composto por dois bancos de 16K x 32 SRAM's e pela lógica de incrementação, que conta a ocorrência de eventos. O terceiro bloco, uma memória DRAM de 2M x 36, que captura e armazena 36 ou 72 bits de informação em cada transação do barramento.

Esta lógica permite a correlação dos speed-ups da aplicação como referência ao comportamento e monitoração da utilização de recursos de hardware. Os resultados de tal análise em um protótipo de 48 processadores são apresentados, por Lenoski e outros em [44].

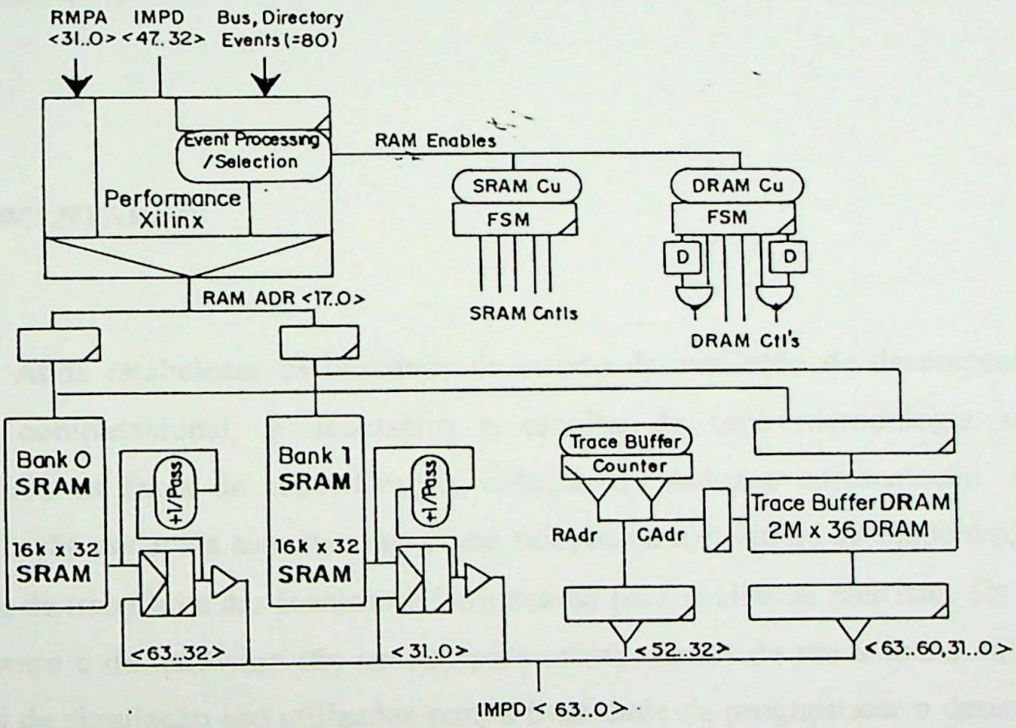


Fig. 2.4 - Diagrama de Bloco Lógico do Monitor de Desempenho

2.6.3 Multiprocessador Cray Y-MP

O multiprocessador Cray Y-MP de memória compartilhada foi lançado no mercado em 1988 com até oito processadores em um único sistema, como o sucessor do multiprocessador Cray X-MP [45]. Cada CPU Y-MP contém um conjunto de

contadores de hardware que podem ser atualizados durante a execução do programa para coletar certas estatísticas de forma não obstrutiva. Esta ferramenta é chamada de monitor de desempenho de hardware (HPM). O HPM monitora o tempo de execução, a utilização do estágio de emissão de instrução ("instruction issue") e o número de operações com ponto-flutuante, com inteiros e de memória. A Tabela 2.3 lista os 32 eventos que podem ser monitorados. Para qualquer execução analisada, um dos quatro grupos de 8 eventos deve ser selecionado, mas um grupo particular pode ser mudado, se desejado por uma chamada do sistema. Além do mais, para os valores retornados pelos contadores, o valor atual dos ciclos da máquina (em tempo-real) para o qual a monitoração foi realizada, é também acumulada pelo sistema e está disponível como um nono evento [46].

2.7 Comentários

Após estabelecer os objetivos de estudo da avaliação de desempenho de um sistema computacional, é necessário a escolha de uma metodologia sistemática envolvendo as fases de especificação, coleção de dados e apresentação. A fase da especificação endereça aspectos tais como seleção da métrica de desempenho, definição da carga de trabalho e das técnicas e ferramentas para avaliar as métricas. Os monitores de software e de hardware são usados para coletar dados de um sistema existente. As técnicas de simulação são utilizadas com a finalidade de prognosticar o desempenho de um sistema não existente ou não disponível. As ferramentas gráficas e textuais são essenciais para visualização dos rastreamentos produzidos pelas diferentes ferramentas de monitoração. A disponibilidade de gráficos e de ambientes com janelas ("windows environment") nas máquinas de interface com o usuário permite diversas tipos de representação gráfica amigáveis e flexíveis.

Tabela 2.3 - Grupo de Contadores HPM para CRAY Y-MP[46]

Grupo Monitor	Contador de Desempenho	Descrição	Incremento por CP
j=0		**Número de:**	
	0	Instructions issued	1
	1	CPs holding issue	1
	2	Fetches of instruction buffer	1
	3	I/O references, ports 3	1
	4	CPU references ports 0,1,2	3 max
	5	Floating Point add operations	1
	6	Floating Point multiply operations	1
	7	Floating Point reciprocal operations	1
j=1		**Hold issue conditions:**	
	0	Semaphores	1
	1	Shared registers	1
	2		
	3	A registers and Functional Units	1
	4	V registers	1
	5	V Functional Units	1
	6	Scalar Memory	1
	7	Block Memory	1
j=2		**Número de:**	
	0	Fetches of Instruction Buffer	1
	1	Scalar References	1
	2	Scalar Conflicts	1
	3	I/O references	1
	4	I/O conflicts	1
	5	Block references, B+T+V	3 max
	6	Block conflicts, B+T+V	3 max
	7	Vector Memory References	3 max
j=3		**Número de:**	
	0	000-017 instructions	1
	1	020-137 instructions	1
	2	140, 157, 175 instructions	1
	3	160 – 174 instructions	1
	4	176 – 177 instructions	1
	5	Vector Integer Operations	3 max
	6	Vector Floating-point operations	3 max
	7	Vector memory references	3 max

Capítulo 3

ARQUITETURA DE COMPUTADORES PARALELOS

3.1 INTRODUÇÃO

A partir dos desenvolvimentos de Hwang e Briggs, em [78], o processamento paralelo pode ser definido como:

"Processamento paralelo é uma forma eficiente de processamento de informação que enfatiza a exploração de eventos concorrentes no processo de computação... Isto representa uma relação de custo/benefício para melhorar o desempenho do sistema, através de atividades concorrentes no computador."

Portanto, os computadores paralelos são aqueles que enfatizam a utilização de eventos concorrentes, afim de melhorar o desempenho do sistema.

Tais eventos concorrentes podem ser explorados em vários níveis de processamento, usando diversas técnicas, tais como "pipelining", replicação de processador, paralelismo funcional, e assim por diante. Como resultado, uma grande variedade de arquiteturas paralelas têm sido propostas, bem como diversas delas já foram construídas, pelo menos de forma experimental. Apesar da diversidade dos modelos, há considerável similaridade nos aspectos que caracterizam sistemas de computadores paralelos.

Embora não exista uma classificação única que possa classificar de maneira correta os sistemas de computação paralela, neste capítulo optou-se por revisar a taxionomia de Flynn, uma das formas de classificação mais utilizadas [47]. Adicionalmente são descritas uma classificação alternativa baseada na estrutura física dos sistemas e um modelo de hardware paralelo que unifica o tratamento dos vários grupos na taxonomia .

Posteriormente, este capítulo apresenta os sistemas de memória compartilhada virtual, com a descrição de alguns exemplos de sistemas existentes. Esta arquitetura oferece uma alternativa aos problemas de memória compartilhada tradicional e aos sistemas de memória distribuída.

3.2 Hardware Para Sistemas Computacionais Paralelos

3.2.1 Níveis de Abstração

A complexidade dos sistemas de computação em geral é melhor compreendida quando estes são organizados em diferentes níveis. A análise de cada nível facilita a compreensão ordenada de suas funções. Os progressos dos níveis mais primitivos da hierarquia para níveis mais elevados são acompanhados pela criação de uma série de abstrações. Pela supressão de detalhes desnecessários, cada abstração contém somente a informação relevante para o nível mais elevado. As abstrações fornecem os caminhos conceituais ao longo dos quais somente passam uma pequena porção de informações.

A Tabela 3.1 ilustra os quatro níveis pelos quais um computador pode ser descrito [48]. Cada sistema é caracterizado por um conjunto de componentes e um conjunto de caminhos que os combinam para montar estruturas. O comportamento do sistema é formalmente descrito de acordo com o comportamento dos componentes e de suas combinações específicas. Linguagens distintas são usadas para representar o sistema (i.e., os componentes, os modos de combinação e as leis e comportamento). Estas linguagens distintas refletem as propriedades especiais dos tipos de componentes e da forma com que eles se combinam.

Tabela 3.1 - Níveis Hierárquicos de Sistemas Digitais [48]

Nível PMS				Estruturas Componentes	computadores, sistemas de múltiplos processadores processadores, memórias, controladores, chaves transdutores, operadores, links
Nível Programa	Subnível Linguagem de Alto Nível	Sistemas de Aplicação		Estruturas Componentes	SPSS, simulador de sistema de potência, resolvedor de equações diferenciais parciais rotinas de biblioteca matemática, rotinas de formatação
		Rotinas de Aplicação		Estruturas Componentes	funções matemáticas, rotinas de plotagem subrotinas, alocação de memória
		Sistemas de Tempo de Execução		Estruturas Componentes	alocação de memória, entrada/saída, sistema de arquivos chamada de sistema operacional
		Sistemas Operacionais		Estruturas Componentes	schedulers, alocadores, comunicação subrotinas, corotinas, programas
	Subnível ISP			Estruturas Componentes	conjunto de instruções estado de memória, estado do processador, cálculo de endereçamento efetivo, decodificação de instrução, execução de instrução
Nível Projeto Lógico	Subnível Transferência de Registro	Controle	Microprogramação	Estruturas Componentes	microprogramas, microrotinas microsequenciador, microstore
			Hardwired	Estruturas Componentes	sequenciador máquinas sequenciais
		Via de Dados		Estruturas Componentes	unidades aritméticas, arquivo de registro registradores, operadores de dados
	Subnível Circuito de Chaveamento	Seqüencial		Estruturas Componentes	contadores, geradores funcionais, registradores flip-flops, latches, delays
		Combinacional		Estruturas Componentes	decodificadores, codificadores, operadores de dados gates
Nível Circuito				Estruturas Componentes	amplificadores, delays, clocks, gates transistores, relés, resistores, capacitores

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ BIM

3.2.2 Hardware de Sistemas Computacionais

No nível Transferência de Registro, os componentes são os registros e as transferências funcionais entre registros. Um registro é um dispositivo que retém o valor de um conjunto de bits. A transferência funcional consiste em portas lógicas e fios de interconexão entre as portas. O comportamento do sistema é dado pelos valores dos registros sequencialmente no tempo. O sistema executa operações discretas, enquanto os valores dos vários registros são combinados de acordo com algumas regras e, então,

armazenados em outro registro (portanto transferidos). A lei da combinação pode ser qualquer, desde uma simples transferência não modificada, do tipo A=B, até uma combinação lógica tipo A = B OU C ou aritmética tipo A=B+C.

Os sistemas do nível de transferência de registro são normalmente visualizados como tendo dois componentes: dados e controle. Os dados são compostos de registros, operadores e via de dados ("data path"). A parte do controle fornece a seqüência de tempo dos sinais que evocam atividades na via de dados. A parte do controle pode ser implementado através de máquinas de estado "hardwired" ou por microprogramação.

Usando esta representação, os sistemas complexos de hardware podem ser modelados como sistemas de transferência de registros síncronos, nos quais os registros são armazenados ("latched") a cada intervalo de tempo ("Ciclo de Máquina"), sendo carregados com uma função combinacional de seu estado de saída e das entradas externas definidas durante um intervalo de tempo anterior. O intervalo de tempo é limitado pelo atraso de tempo associado (latência) de cada componente, registro ou bloco de função. Portanto, é possível omitir registros, considerando simplesmente a latência da porta como seu estado. A Figura 3.1 ilustra a notação abreviada definida para qualquer registro lógico de um sistema de hardware, de tal maneira que se possa construir sistemas complexos a partir de **módulos**, como proposto por Gurd [49].

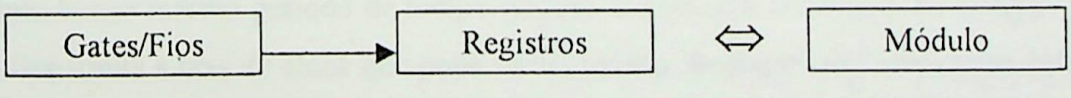


Fig. 3.1 - Notação abreviada para Registro Lógico

3.2.3 Explorando o Paralelismo em Hardware

Em um determinado tempo, a tecnologia determina as constantes de tempo mais importantes que ditam o desempenho de uma implementação. As implementações computacionais podem exceder o desempenho disponível através da tecnologia, introduzindo paralelismo em sua organização. O grau de paralelismo é o número de operações que estão ocorrendo simultaneamente. O paralelismo em uma estrutura é

também uma medida de sua complexidade. Para se ter uma estrutura altamente paralela implica numa estrutura de controle juntamente com múltiplas vias de dados (e operações) que podem ser ativas simultaneamente.

Existe a possibilidade de se construir estruturas paralelas de duas formas diferentes: "arrays" seriais e "arrays" paralelos.

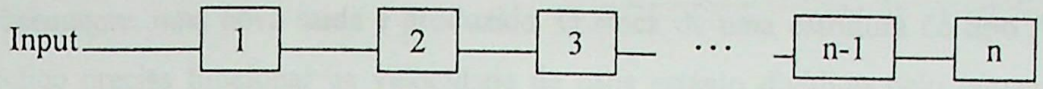
3.2.3.1 Arrays Seriais

O array serial (ou "pipeline") é uma combinação de sucessivos módulos, onde a saída de um módulo é a entrada para o próximo módulo.

O princípio da estrutura pipeline envolve a quebra de um processo em pequenos passos para então executá-los em paralelo, num estilo de linha de produção. A cada ciclo de clock, um novo dado de entrada é fornecido ao array pipeline. Cada um dos passos torna-se um estágio ("pipe stage"), ou passo no pipeline, resultando no padrão de execução como mostrado na Figura 3.2. Após a primeira entrada alcançar o final do pipeline (tempo de latência), uma operação completa é produzida a cada ciclo de clock.

Na implementação pipeline, o clock precisa funcionar numa velocidade compatível com o estágio mais lento. Em uma estrutura pipeline bem balanceada, cada estágio terá o mesmo período de tempo (clock). Entretanto, o número de estágios num pipeline limita a taxa de clock que pode ser alcançada. Registros são requeridos entre os estágios, adicionando o tempo de setup mais os atrasos através dos registros em cada período de clock. Uma vez que o ciclo de clock seja tão pequeno quanto a soma do custo para armazenagem de todos os módulos, o pipelining não é mais útil.

Existem três áreas principais onde a técnica de pipelining pode ser aplicada: processamento de instrução, referências de memória e operações aritméticas de ponto flutuante.



Tempo	Estágio 1	Estágio 2	Estágio 3		Estágio n-1	Estágio n
0	-	-	-		-	-
1	Job1	-	-		-	-
2	Job2	Job1	-		-	-
3	Job3	Job2	Job1		-	-
4	Job4	Job3	Job2		-	-
5	Job5	Job4	Job3		-	-
....	...	....	.....		-	-
n-1	Job n-1	Job n-2	Job n-3	...	Job 1	-
n	Job n	Job n-1	Job n-2	...	Job 2	Job 1
n+1	Job n+1	Job n	Job n-1	...	Job 3	Job 2
n+2	Job n+2	Job n+1	Job n	...	Job 4	Job 3
n+3	Job n+3	Job n+2	Job n+1	...	Job 5	Job 4

Fig. 3.2 - Diagrama Função-Tempo de uma Estrutura Pipeline com n Estágios

3.2.3.2 Arrays Paralelos

O array paralelo é uma replicação de múltiplos módulos, cada um agindo independentemente. Isto é chamado de replicação espacial.

Uma razão comum para o uso de arrays paralelos é para acelerar o desempenho de um módulo pipeline complexo [11]. Existem duas maneiras para implementar tais arrays:

a. Array Cíclico ("cyclic array"): que são módulos complexos replicados, fornecendo várias unidades independentes, as quais permitem que tais módulos operem em diferentes e sucessivas entradas arranjadas através da unidade de controle. Em um array com *n* módulos, após os primeiros *n* passos de tempo ("time-steps"), chamado

de tempo de latência, uma saída é produzida. Então, após cada passo de tempo subsequente, uma nova saída é produzida. O clock de uma estrutura do tipo Array Cíclico precisa funcionar na velocidade de cada estágio dividido pelo número de estágios da estrutura.

uso de arrays cíclicos é particularmente eficiente para aumentar a velocidade do tempo de acesso à memória. A técnica apropriada é conhecida como Memória Intercalada ("Memory Interleaving") e é muita utilizada nos supercomputadores de hoje.

b. Lockstep Array: Neste caso, módulos idênticos são replicados, fornecendo um array de elementos de processamento sob um controle comum, todos realizando a mesma operação simultaneamente, em diferentes entradas. Considerando um array com n módulos, após os primeiros n passos de tempo, n saídas são produzidas simultaneamente. Após isso, n saídas são produzidas em cada conjunto subsequente de passos de tempo. O clock de um "lockstep array" precisa funcionar na velocidade de cada estágio da estrutura.

uso de arrays requer replicação total de módulos de hardware, elevando-se o custo para construir tais estruturas.

3.2.4 Características de Desempenho

3.2.4.1 Speed-up e Eficiência

Seja $Op(n)$, o número total de operações realizadas por uma estrutura paralela de n-módulos, e $T(n)$, o tempo de execução de cada passo em cada unidade. Assuma $T(1) = Op\ (1)$ em uma estrutura de um único módulo. O fator Speed-up é definido de acordo com Hwang [50] como:

$$S(n) = T(1) / T(n) \qquad (3.1)$$

Portanto, estruturas paralelas exibem um fator de speed-up de processamento próximo do número de módulos paralelos.

A eficiência do sistema para uma estrutura paralela de n-módulos é também definida por Hwang [50]:

$$E(n) = S(n)/n = T(1)/(nT(n)) \qquad (3.2)$$

A eficiência é uma indicação do real valor de speed-up de desempenho alcançado quando comparado com o valor máximo. A eficiência máxima é alcançada quando todos os n módulos são completamente utilizados durante o período de execução.

As principais razões que impedem as estruturas paralelas de terem um speed-up linear de processamento são [49]:

1. as entradas não podem ser fornecidas toda vez que são necessáriass;
2. as saídas não podem ser removidas toda vez que são produzidas;
3. um módulo recebe uma entrada, mas não produz qualquer saída ou "stalls" ("internal bubbles");
4. comunicação sincronizada necessária entre módulos.

3.2.4.2 Escalabilidade

O desempenho de um sistema de hardware paralelo depende de um grande número de fatores, todos afetando a escalabilidade da estrutura de hardware e o número de tarefas envolvidas. A mais simples definição de escalabilidade é que o desempenho de um sistema de hardware aumenta linearmente com respeito ao número de módulos usados para uma determinada aplicação.

Com a finalidade de explorar o paralelismo para obtenção de um maior desempenho, deve-se levar em consideração tanto as estruturas paralelas escaláveis quanto os algoritmos escaláveis. A análise pode ser realizada sob diferentes restrições, por exemplo, no crescimento do tamanho do problema (carga de trabalho) ou no tamanho da máquina (número de processadores). Uma boa compreensão de escalabilidade pode ajudar na avaliação de desempenho dos sistemas de hardware paralelo para uma larga faixa de aplicações. Assumindo que os algoritmos podem ser

escalados sem limite, a escalabilidade das estruturas de hardware podem ser limitadas de acordo com Gurd [49]:

1. latência e tempos de setup do registro numa estrutura pipeline;
2. custos dos dispositivos de comutação para rotear a entrada/saída para ou do módulo correto num array cíclico;
3. sistemas de comunicação complexos para canais de entrada / saída num lockstep array.

Como alcançar uma construção balanceada entre estas restrições práticas é o maior desafio para as pessoas que concebem as estruturas de hardware paralelo dos dias de hoje.

3.2.5 Interconexões

Como descrito no item 3.2.3, a replicação de módulos não é viável, a menos que as conexões possam ser estabelecidas entre eles, de modo programável. Estas conexões podem ocorrer usando:

a. conexões ("wring") entre gates e registros;
b. conexões ("wiring") entre componentes ;
c. redes de chaveamento, i.e, uma coleção de dispositivos de chaveamento, com uma dada topologia de conexão.

Enquanto poucos componentes podem se comunicar eficientemente usando fios, tais como sistemas usando barramento, para um grande número de componentes, a inerente seqüencialidade desses métodos de comunicação produz gargalos ("bottleneck") no sistema. Neste caso, é preferível considerar sistemas nos quais os dados são chaveados entre módulos. Este chaveamento pode acontecer ou para uma rede fixa, também chamadas de redes estáticas, na qual a distância cresce com o número de módulos, ou para uma rede de conexão programável, também chamadas de redes dinâmicas, nas quais o custo segue uma lei quadrática e consideráveis problemas de fiação estão associados [49].

As redes de chaveamento fornecem um conjunto de interconexões ou de mapeamento entre dois conjuntos de nós. Para n entradas e m saídas, existem $n \times m$ conexões bem definidas. As funções de mapeamento comumente vistas entre módulos, incluem deslocamentos ("shifting"), rotações, permutações (uma a uma), broadcast (uma para todas), multicast (muitos para muitos), comunicação personalizada (uma para muitos), e assim por diante.

A eficiência de comunicação da rede é crítica para o desempenho de hardware paralelo. O melhor desempenho é realizado por uma rede de baixa latência, com uma alta taxa de transferência de dados (e assim, uma grande banda de comunicação).

3.2.5.1 Propriedades das Redes

Em geral, uma rede é representada por um grafo com um número finito de nodos (módulos), ligados direta ou indiretamente. O número de nodos no grafo é chamado de **tamanho da rede**. Algumas propriedades comuns incluem:

a. Grau do nodo: expresso pelo número de conexões(ligações ou canais) que incidem num nodo;
b. Diâmetro da Rede: expresso pelo comprimento máximo do caminho mais curto entre qualquer dois nodos. O comprimento do caminho é medido pelo número de ligações "traversed".
c. Largura da Bisseção: é expressado pelo número mínimo de canais ao longo do corte ou bisessão da rede.

O parâmetro Grau do Nodo reflete o número de portas de E/S do nodo e portanto o custo por nodo. Também indica quantidade de contenção que pode ocorrer em qualquer ponto crítico da rede. Portanto, deveria ser mantido constante e tão pequeno quanto possível. Este parâmetro constante é muito desejado para obter a modularidade na construção dos blocos de sistemas escaláveis. O Diâmetro da Rede expressa a latência potencial da interconexão fornecendo o número máximo de hops entre dois nodos quaisquer. Ele deve ser mantido tão pequeno quanto possível do ponto de vista

da comunicação, e a Largura da Bisseção fornece um bom indicador da máxima largura de faixa de comunicação ao longo da bisseção de uma rede.

3.2.5.2 Desempenho da Rede

O desempenho de uma rede de interconexão é afetado pelos seguintes fatores:

1. Latência da rede: Refere-se ao pior caso de tempo de atraso para uma mensagem ser transferida através da rede;
2. Largura de Faixa: Refere-se à taxa de transferência de dados máxima transmitida através da rede;
3. Complexidade de Hardware: Refere-se aos custos da implementação, tais como para os fios, chaves, conectores, arbitração e a lógica de interface;
4. Escalabilidade: Refere-se à habilidade de uma rede ser modularmente expansível, apresentando um desempenho escalável com o aumento de recursos da máquina.

3.2.5.3 Redes de conexão estática

Redes estáticas usam ligações diretas ponto-a-ponto que são fixas, uma vez construídas. As topologias mais comuns são arrays lineares, anel, árvore, estrela, "fat tree", malha, toroidal, e hipercubo. Os arrays lineares são a topologia de mais simples conexão e impõem comunicação ineficiente quando o número de nodos torna-se grande. Um anel é obtido pela conexão dos dois nodos terminais de um array linear com uma ligação extra. Esta ligação pode ser unidirecional ou bidirecional. Aumentando-se o grau do nodo mais ligações são adicionadas. Como resultado, o diâmetro da rede é diminuido. Com um grau do nodo constante, a árvore binária é uma estrutura escalável. Fat tree é uma topologia de árvore modificada, onde a largura do canal é aumentada assim que a comunicação se move das folhas para a raiz. Como resultado, a topologia fat tree alivia o pesado tráfico para a raiz, presente na topologia de árvore convencional. Malha é uma topologia popular, a qual tem sido implementada nos sistemas computacionais, tais como Illiac IV, MPP, DAP, CM-2 e Intel Paragon, com variações.

O hipercubo é uma topologia binária n-cube que consiste em 2^n nodos distribuídos ao longo de n dimensões, com dois nodos por dimensão. O grau do nodo aumenta linearmente, com respeito à dimensão, sendo difícil considerar a arquitetura hipercubo escalável. Muitas outras arquiteturas, tais como árvores binárias, malhas, etc, podem ser embutidas num hipercubo.

3.2.5.4 Redes de Conexão Dinâmica

São usadas para aplicações de propósito geral, porque a conexão dinâmica pode implementar todo padrão de comunicação baseado na demanda do programa. Ao invés de usar conexões fixas, chaves ou árbitros precisam ser usadas ao longo dos caminhos de conexão, para fornecer a conectividade dinâmica. Em ordem crescente de custo e desempenho, as redes de conexões dinâmicas incluem sistema de barramentos, redes de interconexão de múltiplos estágios - MIN, e rede "crossbar".

O preço destas redes são atribuídas aos custos dos fios, chaves, árbitros e conectores necessários. O desempenho é atribuído pela largura de faixa da rede, taxa de transferência de dados, latência da rede e padrões de comunicação suportados.

Um sistema de barramento é essencialmente uma coleção de fios e conectores utilizados para transação de dados entre vários módulos conectados ao barramento. O barramento é usado somente para uma transação em um determinado tempo. Por esta razão, é chamado de barramento de contenção ou barramento de tempo compartilhado entre múltiplos módulos. Exibe um baixo custo e fornece uma largura de faixa limitada, se comparado com outras redes de conexões dinâmicas. Muitos padrões industriais e do IEEE estão disponíveis.

Um módulo de chave **a** × **b** tem **a** entradas e **b** saídas. Cada entrada pode ser conectada a uma ou mais saídas (mapeamentos uma-a-uma e uma-para-muitos). Redes de multi-estágios (MINs) usam um certo número de chaves **a** × **b** em cada estágio. As conexões inter-estágios fixas são usadas entre as chaves de estágios adjacentes. As chaves podem ser ajustadas dinamicamente para estabelecer as conexões desejadas

entre as entradas e saídas. Diferentes classes de MINs diferem no módulo de chave e no padrão da conexão inter-estágio (ISC) utilizada. O módulo mais simples seria a chave 2x2. Os padrões ISC frequentemente utilizados incluem o "perfect shuffle", "butterfly", "multiway shuffle", "crossbar", "cube connection", e assim por diante. A maior vantagem das MINs encontra-se na sua escalabilidade com construção modular. As desvantagens são a latência causada pelos sucessivos estágios, o número de fios e a complexidade do chaveamento.

Numa rede crossbar, as chaves de cruzamento ("crosspoint switches") podem estar ligadas ou desligadas, fornecendo a conexão dinâmica entre um par (fonte, destino). Redes Crossbar fornecem a mais alta largura de faixa e capacidade de interconexão, mas são mais caras para construir, porque a complexidade do hardware aumenta com o quadrado de **n**, quando **n** é o número de entradas (ou saídas) [50]. Para um tamanho de rede pequeno, torna-se uma escolha desejada.

3.2.6 Modelo Unificado para Hardware Paralelo

O primeiro estágio no desenvolvimento de qualquer classificação de sistema computacional é a habilidade em descrever de maneira razoável, precisa e concisa os aspectos essenciais de uma arquitetura particular. Isto requer uma notação apropriada. Exemplos de tais notações estão descritas por Siewierek e outros, em [48], e por Hockney e outros, em [11]. No item **3.2.6.2** uma notação que possibilita descrever um sistema hardware paralelo ser descrito usando um modelo abstrato unificado é fornecida. Tal notação é usada em **3.3**, para fornecer uma classificação de sistemas computacionais paralelos e no Capítulo 4, para descrever os computadores paralelos usados no estudo de avaliação de desempenho proposto nesta dissertação. Contudo, antes de descrever os elementos propostos na notação, vamos rever alguns conceitos de uma notação similar chamada PMS.

3.2.6.1 Trabalho Anterior – Notação PMS

Da Tabela 3.1, o quarto e último nível de abstração dos sistemas computacionais é chamado "processor-memory-switch", ou simplesmente PMS [48]. Esta é a visão que tomamos de um sistema computacional quando consideramos apenas seu comportamento mais agregado. A notação PMS ignora toda a estrutura mais fina do processamento da informação considerando um sistema que consiste em componentes que trabalham num meio homogêneo, chamado informação. A informação vem em pacotes, chamados Unidades de Informação, e são medidas em bits ou outra unidade equivalente.

No PMS, há sete tipos de componentes básicos distinguidos pelo tipo de operação que cada um realiza: Memória, ligação, controle, chave, transdutor, dado-operação e processador. Especificações adicionais associadas com cada componente estão incluidas na notação, tais como tempo de ciclo, organização da memória, latência da memória, tecnologia, capacidade, etc. Informações específicas a cerca da notação PMS pode ser encontrada também em Siewiorek [48].

Os sete tipos de componentes são conectados para compor os computadores digitais de programas armazenados ("stored program"), como ilustrado na Figura 3.3. Pc indica o processador central e Mp a memória primária.

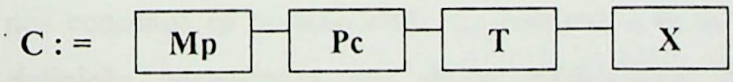


Fig. 3.3 - Descrição de um Sistema Computacional no nível PMS

Todos os componentes consistem em um conjunto de operações discretas, e quando evocadas, realizam alguma transformação discreta de estado. Cada componente pode ter uma unidade de controle local associada, que permite que ele execute operações primitivas. Esta representação pode ser vista na Figura 3.4. Portanto, componentes podem ser decompostos em arranjos de outros componentes. Estes subcomponentes são capazes de executar operações primitivas nos dados. Por exemplo, um processador Pc pode ser decomposto em vários componentes, tais como Unidade

Lógica e Aritmética, como mostrado no diagrama da Figura 3.4. Pode ainda conter memórias locais temporárias , tais como registros ou memória cache.

A informação é distribuída através dos componentes pelas ligações e compreende duas partes: a parte do código e a parte do dado. A unidade de informação é medida em bits.

A notação PMS suprime todos os detalhes sobre o modelo lógico de computação. Na verdade, é mais uma abstração do nível lógico do que do nível de programa.

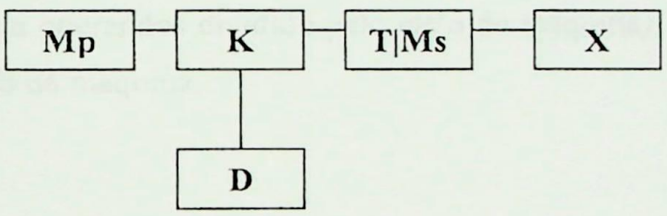


Fig.3.4 - Nível PMS com controle local

3.2.6.2 Notação proposta

Baseado nos conceitos da notação PMS, foi proposta uma nova notação que contenha uma definição generalizada dos vários componentes de um sistema computacional abstrato. O objetivo da notação proposta é unificar o tratamento de diferentes estruturas de hardware para sistemas paralelos. Esta notação pode vir a servir, em estudos futuros, para definir métricas para um estudo de avaliação de desempenho baseado em estruturas de hardware paralelo [51,52].

A notação considera um sistema de hardware como um número de componentes que são baseados no modelo transição-estado, mais as interconexões que ligam os componentes. Cada componente pode ser decomposto como um conjunto de "Unidades Funcionais", que são capazes de executar operações primitivas nos dados. A notação

reconhece dois tipos de componentes, classificados por sua funcionalidade: Aqueles que predominantemente executam operações, chamados Elementos de Processamento (PE's), e aqueles que predominantemente armazenam dados, chamados Elementos de Memória (ME's). Portanto, o estado total do sistema distribuído através dos componentes consiste em duas partes: a parte do código e a parte dos dados. As unidades de informação de cada uma são instruções e operandos, respectivamente. O tamanho de ambas as unidades são tratados como equivalentes.

Outros parâmetros usados para descrever aspectos dos componentes de hardware incluem Ciclo de Máquina ("Machine Cycle"), o qual é definido como intervalo mínimo durante o qual um operando é produzido ou consumido em qualquer componente do sistema. A "Capacidade do Processamento" de um componente é o número máximo de interconexões internas que transportam dados durante um ciclo de máquina (expresso em operandos dividido pelo ciclo de máquina). O Tempo de Ciclo é a duração de um ciclo da máquina.

a) Elementos de Processamento (PE's)

São caracterizados por componentes capazes de interpretar instruções a fim de executar uma seqüência de operações. Um Elemento de Processamento contém Unidades Funcionais de Processamento e Memória Cache. O número total desses elementos é **P**, e para cada Elemento de Processamento é atribuído um índice **p**, onde $\mathbf{1 \leq p \leq P}$.

1. **Unidades Funcionais de Processamento**: São unidades computacionais individuais, possivelmente paralelas, dentro de cada elemento de processamento. Funções típicas executadas por tais unidades incluem operações de ponto flutuante escalares ou, operações de gerenciamento de memória e operações de comunicação.
2. **Memória Cache**: É caracterizada por qualquer memória conectada diretamente às Unidades Funcionais de Processamento. Isto inclui memória cache de dados e de instruções, registros, stack, e assim por diante.

Um Elemento de Processamento é caracterizado como segue:

1. **Capacidade de Processamento de Instrução**: O número máximo de instruções que podem ser emitidas num ciclo de máquina (em instruções/ciclo de máquina).
2. **Capacidade de Processamento de Dados**: O número máximo de operandos que podem ser armazenados num ciclo de máquina (em operandos por ciclo de máquina)
3. **Capacidade do Cache de Instruções** (em instruções).
4. **Capacidade do Cache de Dados** (em operandos).
5. **Taxa de acesso ao Cache de Instruções** (em Instruções por ciclo de máquina).
6. **Taxa de Acesso ao Cache de Dados** (em Operandos por ciclo de máquina).

Todas as características são taxadas no valor máximo.

b) Elementos de Memória

A memória principal constitui o volume primário de memória para o sistema, e é normalmente dividido em múltiplos elementos de memórias. Ela pode ser compartilhada, distribuída ou hierárquica. Memória compartilhada pode ser centralizada ou distribuída. Sistemas de memória distribuída são caracterizadas por vários Elementos de Memória locais e são tratados como parte da memória principal, exceto para partes que podem ser claramente identificadas como memória cache.

Por convenção, o número total de Elementos de Memória é **M**, e para cada elemento é atribuído um índice **m**, onde $\mathbf{1 \leq m \leq M}$.

Um Elemento de Memória pode conter hardware, possivelmente unidades paralelas, chamadas Unidades Funcionais de Memória, que verdadeiramente realizam as operações de memória, incluindo leitura, escrita, transferência, incrementação, e "test-and-set".

Um Elemento de Memória é caracterizado pelos seguintes parâmetros:

1. Capacidade do Elemento de Memória (em operandos).

2. Largura de Faixa do Elemento de Memória: o qual representa a taxa de acesso máximo (em operandos por ciclo de máquina).
3. Latência do Elementos de Memória: o qual representa a atraso mínimo, antes que o Elemento de Memória responda ao pedido (em ciclos de máquina).

c) Interconexões

As interconexões de um sistema incluem:

1. Interconexões entre os componentes, tais como Elementos de Processamento e/ou Elementos de Memória.

2. Interconexões com os componentes de um sistema, tais como Unidades Funcionais de Processamento e memórias cache, Unidades Funcionais de Processamento dentro dos Elementos de Processamento, Unidades Funcionais de Memória e níveis da memória principal e Unidades Funcionais de Memória distintas.

3. Interconexões com o mundo exterior incluindo conexões entre outros Elementos de Processamento ou Elementos de Memória do sistema.
4. As unidades de informação transferidas podem ser operandos e instruções. Como o fator crítico é movimento de dados, a unidade de transferência a ser considerada será apenas **operandos**.
5. Uma interconexão é caracterizada pelo seguinte:

6. Largura de Faixa de dados em operandos por ciclo de máquina;

7. Latência dos dados: É o atraso mínimo para um operando viajar de um terminal da interconexão para a outra extremidade (em ciclo de máquina).

3.3 Classificação da Arquitetura de Hardware

A recente proliferação de tecnologias de processamento paralelo tem incluído novas arquiteturas de hardware paralelo, interconexão de tecnologias e paradigmas de programação. A diversidade deste campo apresenta um obstáculo substancial para os usuários que desejam compreender que tipos de arquiteturas paralelas existem e como a relação entre elas define um esquema regular. Muitos esquemas de classificação para arquiteturas de computadores têm sido propostos nas últimas décadas, e especificamente para arquiteturas de computadores paralelos. Dasgupta [53] tem discutido algumas das características desejáveis para sistemas taxonômicos mais "adequados". Este item revê alguns desses sistemas de classificação e descreve uma nova taxonomia baseada na estrutura física dos sistemas.

3.3.1 Taxonomia de Flynn

A classificação macroscópica de arquiteturas generalizadas de computadores proposta por Michael J. Flynn [47,54], é baseada em como as máquinas relacionam suas instruções com os dados que estão sendo processados. Não há relação com a estrutura das máquinas. Esta classificação produz quatro classificações amplas, de acordo com as seqüências de dados ou de instruções sejam únicas ou múltiplas. A Figura 3.5 ilustra as quatro classes.

- **SISD** – Seqüência de Instrução única / Seqüência de dados única: Define uma máquina serial convencional de von Neumann .
- **MISD** –Seqüência de Instruções múltiplas / Seqüência de dados única: Esta classe implica que muitas instruções estão operando sobre um único dado simultaneamente. Esta possibilidade hipotética é considerada impraticável.
- **SIMD** – Seqüência de Instruções única / Seqüência de dados múltiplos: Envolve múltiplos Elementos de Processamento simultaneamente executando a mesma instrução em diferentes dados. Esta classe inclui os processadores vetoriais.
- **MIMD** – Seqüência de instruções múltiplas / Seqüência de dados múltiplos: Envolve múltiplos Elementos de Processamento, autonomamente executando

diversas instruções sobre diversos dados. Esta classe inclui todas as formas de configurações de multiprocessadores.

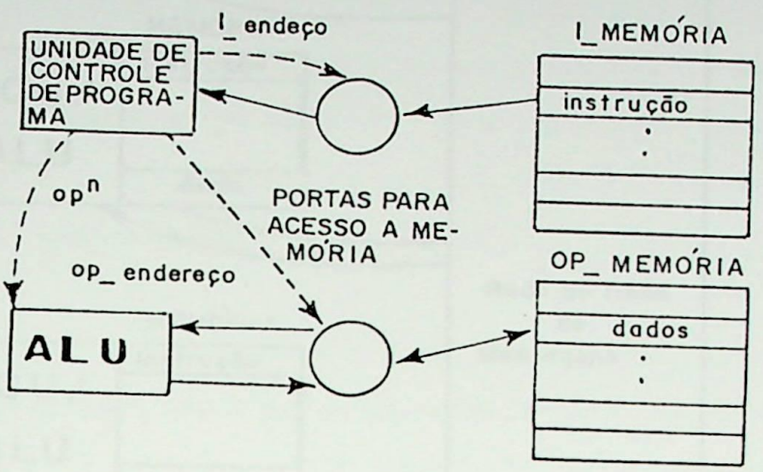
Apesar das deficiências da taxonomia de Flynn, esta classificação continua sendo popular principalmente porque nenhuma real alternativa foi elaborada.

3.3.2 Outras taxonomias

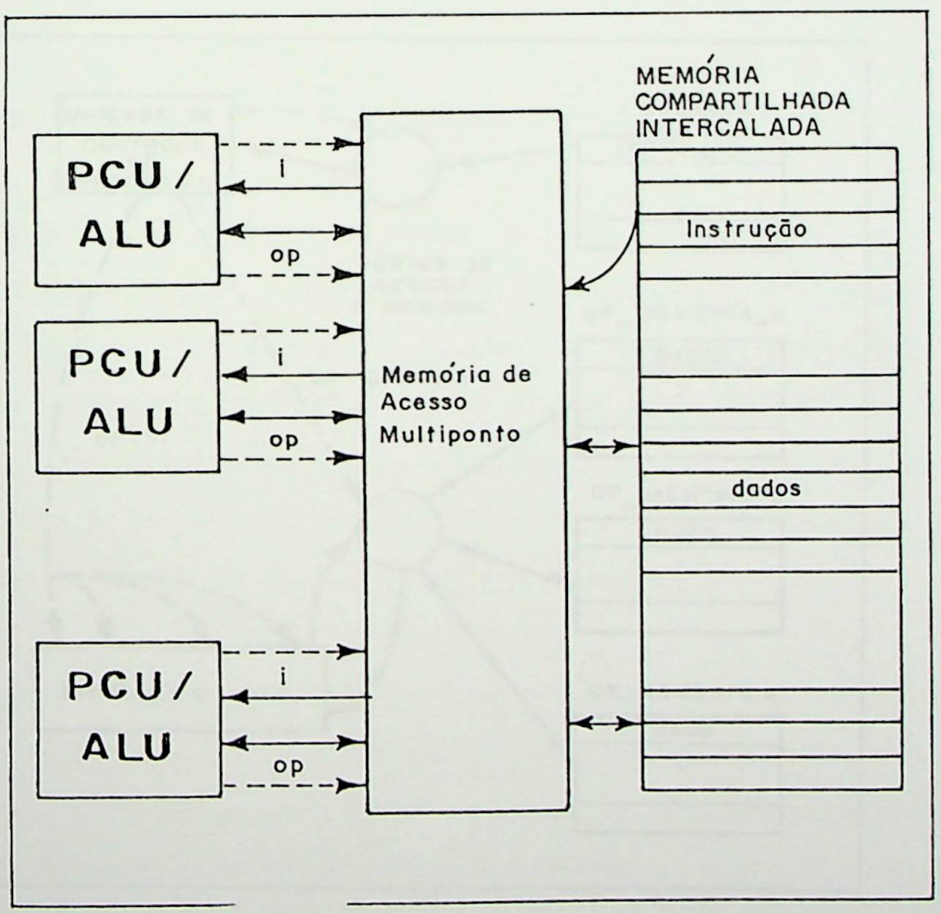
Embora a classificação de Flynn [47,54] permaneça em uso, ela tem-se mostrado ineficiente em diferenciar muitas das novas arquiteturas desenvolvidas. Numerosas alternativas têm sido propostas. A classificação de Shore, mostrada por Hockney em [11], é baseada em como o computador é organizado, a partir de suas partes constituintes, i.e. , unidade de controle, unidade de processamento, memória de instrução e memória de dados. Seis diferentes tipos de máquinas foram reconhecidas. Entretanto, algumas categorias, tais como o processador com "vector pipeline" não são satisfatoriamente cobertos pela classificação.

Skillicorn [55] produziu um sistema de classificação baseado no número de instruções e elementos de processamento no sistema, nas relações entre eles, no modelo de memória para instruções e dados e no modo com que múltiplos elementos de processamento interagem. A classificação derivou 28 arquiteturas de computador que podem tratar desde computadores sem processadores de instrução ("dataflow architectures") até aqueles com múltiplas instruções e elementos de processamento. A classificação é baseada em arquiteturas abstratas.

Dasgupta [53] introduziu um sistema taxonômico de arquiteturas que inclui aspectos tais como estrutura hierárquica, nomeclatura, poder de prognóstico e capacidades explanatórias. Este modelo foi influenciado por notações e termos químicos, e exibe considerável poder de classificação.

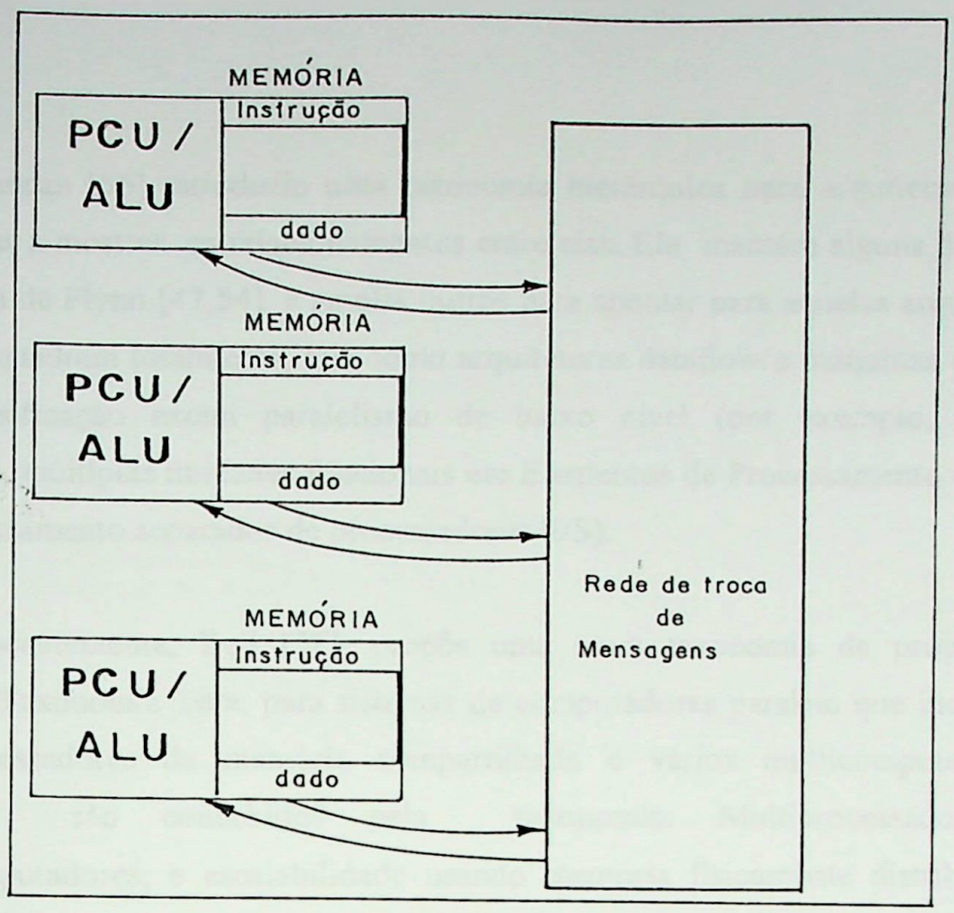


(a)

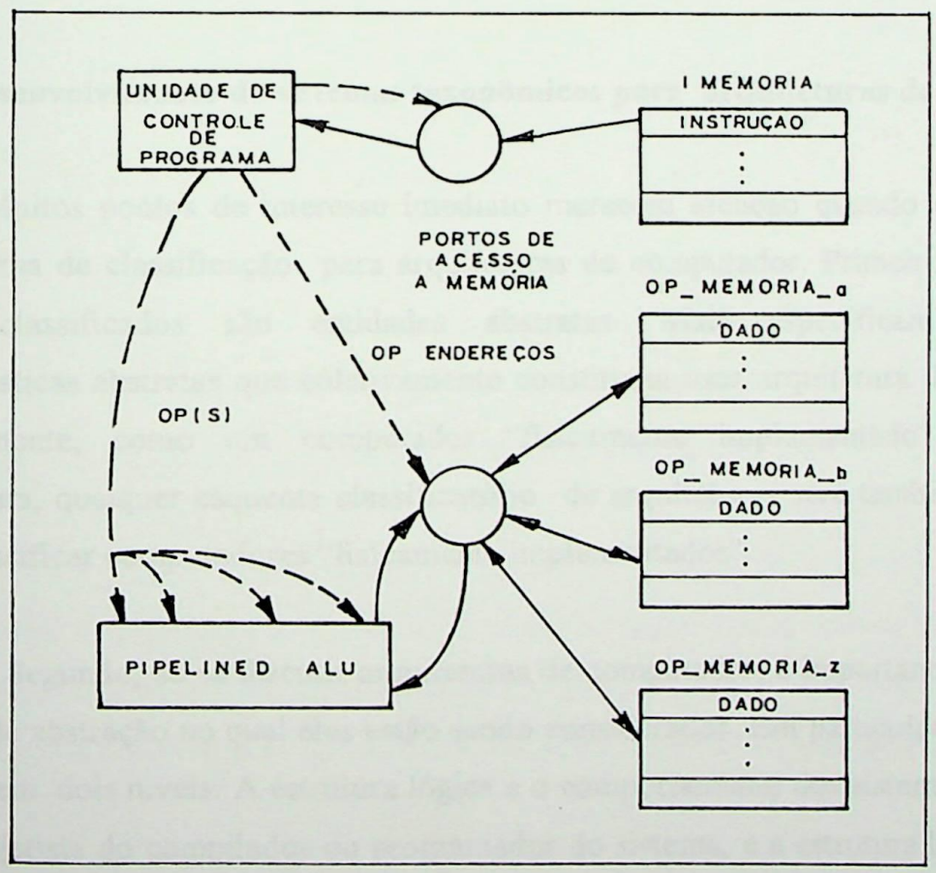


(b)

Fig. 3.5 - Classificação de Flynn de arquiteturas de computadores: (a) SISD, (b) MIMD compartilhada, (c) MIMD distribuída, (d) SIMD.



(c)



(d)

Fig. 3.5 - Classificação de Flynn de arquiteturas de computadores: (a) SISD, (b) MIMD compartilhada, (c) MIMD distribuída, (d) SIMD. (continuação)

Duncan [56] introduziu uma taxonomia hierárquica para arquiteturas paralelas para ajudar a mostrar os relacionamentos entre elas. Ela mantém alguns dos conceitos do sistema de Flynn [47,54], e amplia outros para apontar para aquelas arquiteturas que não se enquadram totalmente, tais como arquiteturas dataflow e máquinas de redução. Esta classificação exclui paralelismo de baixo nível (por exemplo, pipeline de instruções, múltiplas unidades funcionais em Elementos de Processamento e Elementos de Processamento separados de processadores E/S).

Recentemente, Bell [57], propôs uma nova taxonomia de propósito geral, chamada Taxonomia Tera, para sistemas de computadores paralelo que inclui somente multiprocessadores de memória compartilhada e vários multicomputadores Dois atributos são concebidos pela taxonomia: Multiprocessadores versus Multicomputadores; e escalabilidade usando memória fisicamente distribuída versus memória centralizada. A Figura 3.6 ilustra esta classificação.

3.3.3 Desenvolvimento de sistemas taxonômicos para arquiteturas de computador

Muitos pontos de interesse imediato merecem atenção quando desenvolvendo um sistema de classificação para arquiteturas de computador. Primeiro, os objetos a serem classificados são entidades abstratas. Mais especificamente, aquelas características abstratas que coletivamente constituem uma arquitetura de computador. Naturalmente, como um computador "fisicamente implementado" possui uma arquitetura, qualquer esquema classificatório de arquiteturas será também um sistema para classificar computadores "fisicamente implementados".

Segundo, ao se discutir arquiteturas de computador, é importante compreender o nível de abstração no qual eles estão sendo considerados. Em particular, os projetistas distinguem dois níveis: A estrutura lógica e o comportamento dos sistemas como vistos pelo projetista do compilador ou programador do sistema, e a estrutura lógica, controle e comportamento dos componentes de hardware de sistemas integrados. O primeiro nivel consiste em componentes tais como a hierarquia de armazenagem visível externamente, o conjunto de instruções, o formato das instruções e os modos de endereçamento.

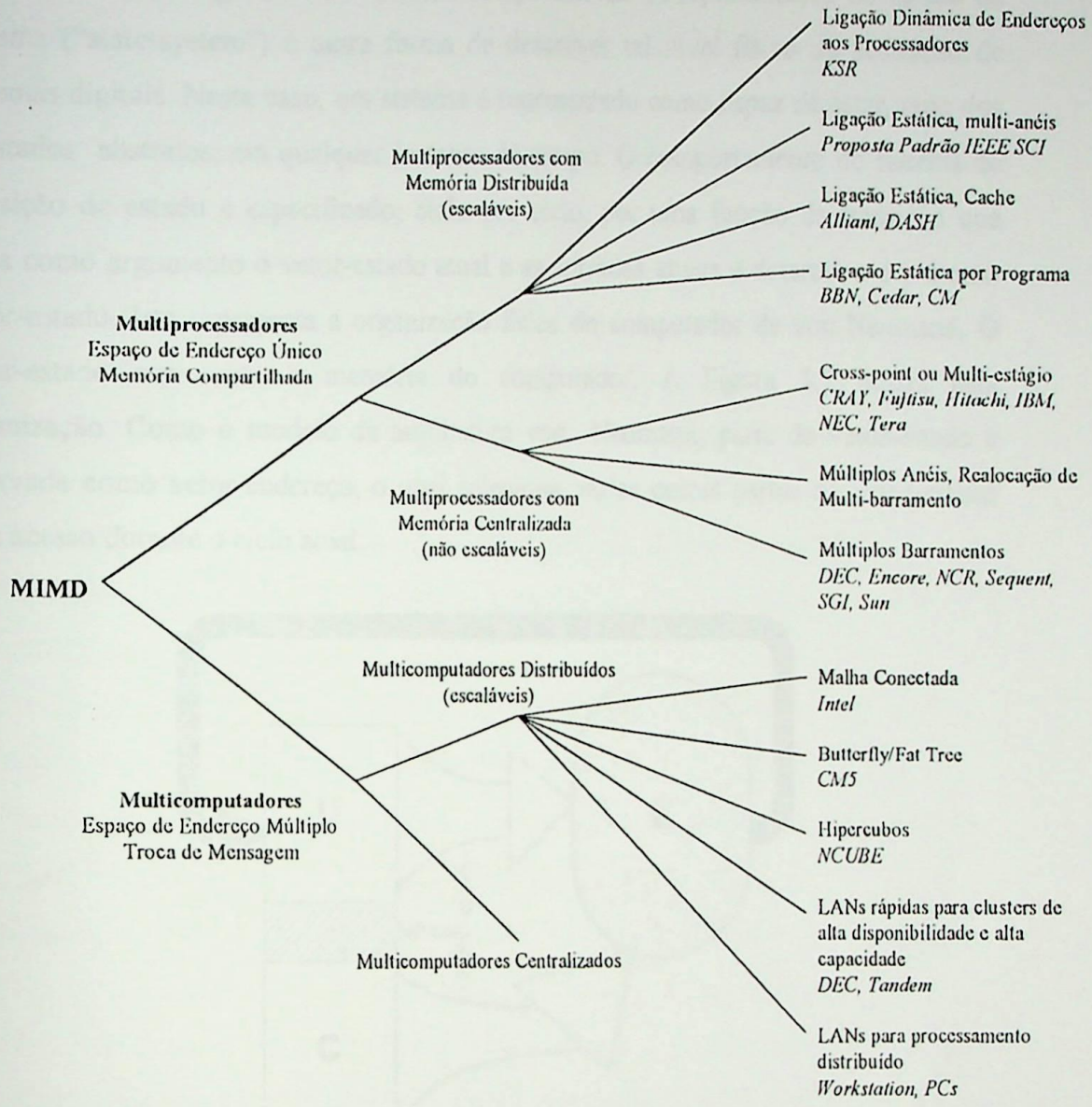


Fig. 3.6 - Taxonomia Tera para multiprocessadores e multicomputadores [57]

O segundo nivel considera a natureza do ciclo de instrução (interpretação e execução), a organização interna e as relações entre componentes de memória e as unidades funcionais e os meios pelo qual o processamento da instrução é controlada. A discussão da taxonomia proposta está totalmente confinada a este nível.

3.3.4 Taxonomia baseada na estrutura física dos sistemas

No item 3.2.2 foi utilizada a representação de transferência de registro para

descrever o nível lógico de um sistema computacional. A representação do estado do sistema ("state-system") é outra forma de descrever tal nível físico de abstração de sistemas digitais. Neste caso, um sistema é representado como capaz de estar, num dos **n** estados abstratos, em qualquer instante de tempo. O comportamento do sistema de transição de estado é especificado, ciclo por ciclo, por uma função de transição que toma como argumento o vetor-estado atual e as entradas atuais e determina o próximo vetor-estado. Isto representa a organização física do computador de von Neumann. O vetor-estado representa a memória do computador. A Figura 3.7 ilustra esta organização. Como o modelo da arquitetura von Neumann, parte do vetor-estado é reservada como vetor-endereço, o qual seleciona várias outras partes do vetor-estado para acesso durante o ciclo atual.

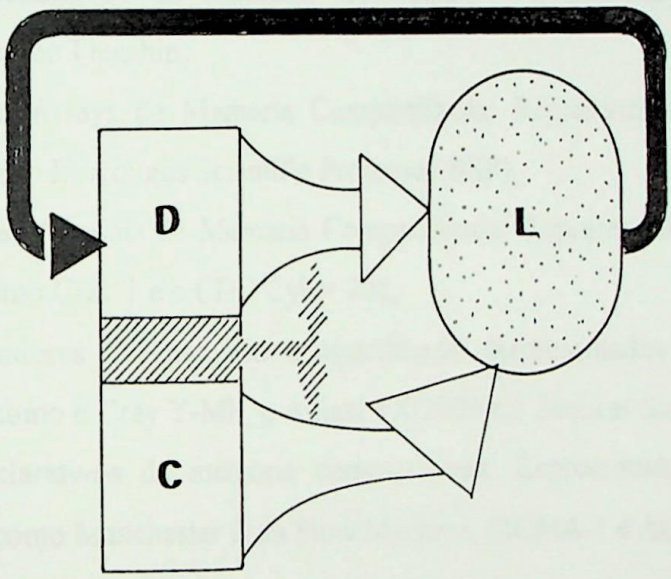


Fig. 3.7 - Representação da Transição de Estado

Os sistemas de transição de estado paralelos desenvolvem tais modelos, numa das seguintes formas, que podem corresponder ao aumento do tamanho do vetor-endereço, expandindo desse modo o total do vetor-estado que pode ser acessado durante cada ciclo de instrução [49]:

a. Divide o vetor-estado em vetores-estado-local , cada um com seu próprio vetor-endereço local. Este é o modelo de memória distribuída;
b. Replica o vetor-endereço- de tal maneira que muitas instruções separadas possam acessar o vetor-estado simultaneamente. Este é o modelo de memória compartilhada.

Estes dois modelos fisicos levam à seguinte classificação de máquinas paralelas, baseadas primariamente se o sistema contém apenas memória distribuída, ou se há alguma quantidade de memória compartilhada endereçável globalmente. Sete classes de máquinas paralelas são identificadas [51]:

a. Processadores Arrays com Memória Distribuída: Representados pelas máquinas SIMD, tais como Illiac IV, AMT DAP, e TMC CM2;
b. Multicomputadores de Memória Distribuída: Representados pelas máquinas MIMD, tais como Intel iPSC/860, N-Cube, Meiko Computing Surface, Multiprocessador Stanford Dash e Kendall Square Research KSR1- 32, IBM RS-6000/SP2;
c. Sistemas Declarativos de Memória Distribuída: Representados por máquinas MIMD, tais como Flagship;
d. Processadores Arrays de Memória Compartilhada: Representados por máquinas SIMD tais como Burroughs Scientific Processor(BSP);
e. Processadores Vetoriais de Memória Compartilhada: Representados por máquinas SIMD tais como Cray 1 e o CDC Cyber 205;
f. Multiprocessadores de memória compartilhada: Representados pelas máquinas MIMD, tais como o Cray Y-MP, o Alliant FX/2800 e o Sequent Simetry;
g. Sistemas declarativos de memória compartilhada: Representados por máquinas MIMD, tais como Manchester Data Flow Machine, SIGMA-1 e ALICE;

3.4 Multiprocessadores e Multicomputadores

A diferença entre o hardware de sistemas de multiprocessadores e multicomputadores reside no fato de como o sistema tem e mantém um espaço único de endereço simples e uma única memória coerente, e se mensagens explicitas são necessárias para acessar memória em outros nodos da computação. Como resultado, esses modelos fisicos são distinguidos por terem memória compartilhada ou memória distribuída não compartilhada. O conceito de memória virtual tem sido usado para ligar esses dois modelos, a fim de obter estruturas de hardware escaláveis. Esta seção descreve a organização da arquitetura desses três modelos.

3.4.1 Multiprocessador de Memória Compartilhada

Arquiteturas de memória compartilhada realizam a coordenação interprocessador, através de uma memória global compartilhada onde cada processador pode endereçar. Tais arquiteturas envolvem múltiplos processadores de propósito geral compartilhando memória. Um sistema generalizado de multiprocessador está ilustrado na Figura 3.8. Cada processador Pi está junto com sua própria memória local e cache privada. Múltiplos processadores são conectados aos módulos de memória compartilhada, através de uma rede de memória-interprocessador. Tal ilustração combina aspectos de diferentes modelos que serão introduzidos nesta seção.

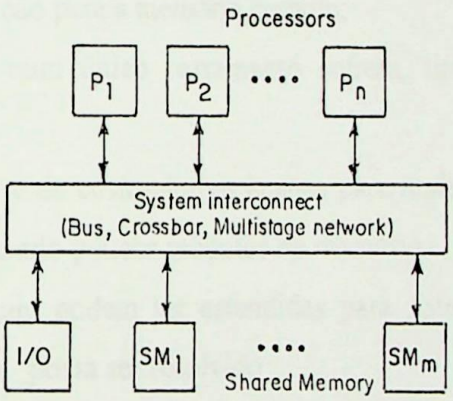


Fig. 3.8 - Sistema Multiprocessado genérico

Sistemas de multiprocessadores são convenientes para aplicações de propósitos gerais em ambientes multi-usuário, onde a programabilidade é a maior preocupação. As duas maiores limitações dos multiprocessadores são a falta de escalabilidade e a latência para acesso à memória remota. Estas limitações tornam mais dificeis construir arquiteturas escaláveis usando modelos de memória compartilhada centralizada [57].

Os multiprocessadores de memória compartilhada podem ser descritos em três modelos, baseados em como a memória e os recursos periféricos são compartilhados ou distribuídos : Modelo UMA ("Uniform Memory Access"), Modelo NUMA ("Non-Uniform Memory Access) e Modelo COMA ("Cache Only Memory") [58].

3.4.1.1 Modelo UMA

Em um modelo de multiprocessador UMA, a memória física é uniformemente compartilhada por todos os processadores, que têm igual tempo de acesso a todas as palavras da memória. Os multiprocessadores deste modelo são chamados de sistemas fortemente acoplados, devido ao alto grau de recursos compartilhados.

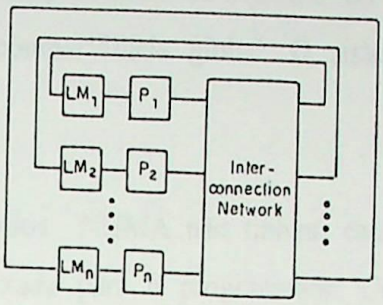
O modelo UMA é apropriado para aplicações de propósito geral e de tempo compartilhado por múltiplos usuários. A sincronização e a comunicação entre processadores são feitas implicitamente, através de variáveis compartilhadas na memória comum. A escalabilidade do modelo UMA é limitada devido a:

1□ Problema de contenção para a memória comum;
2□ Sistemas baseados num único barramento sofrem, tipicamente, de saturação de barramento;
3□ Uma rede crossbar é de custo-efetivo apenas para multiprocessadores com poucos processadores acessando poucos módulos de memória.
4□ Redes de multiestágio podem ser estendidas para sistemas maiores, desde que o problema da latência possa ser resolvido.

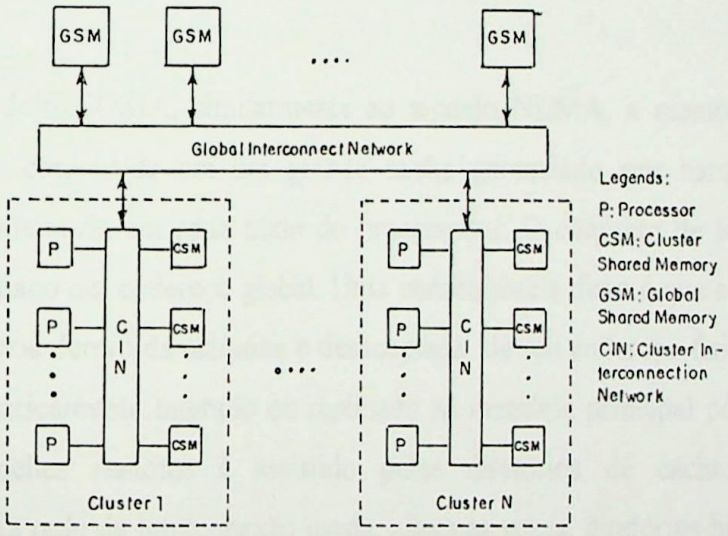
3.4.1.2 Modelo NUMA

Um multiprocessador NUMA constitui-se de um sistema de memória compartilhada, no qual o tempo de acesso varia com a localização da palavra de memória. A memória compartilhada está fisicamente distribuída em todos os processadores (memória local). A coleção de todas as memórias locais formam um espaço de endereço global acessível a todos os processadores. A Figura 3.9 ilustra dois exemplos de tal modelo.

Escola Federal de Engenharia de Itajubá - EFEI
Biblioteca MAUÁ



(a)



(b)

Fig. 3.9 - Modelos de Multiprocessador NUMA: (a) Memórias locais compartilhadas (p.ex. BBN Butterfly), (b) Modelo de Agregação Hierárquico (p.ex. Cedar system) [50]

É mais rápido acessar uma memória local com processador local. Entretanto, o tempo acesso à uma memória ligada a um nodo remoto é mais longo devido à demora adicional através da rede de interconexão. Memória globalmente compartilhada pode ser adicionada a um sistema de multiprocessador, a fim de obtermos um sistema estruturado hierarquicamente, como ilustrado na Figura 3.9(b). Neste caso, há três tipos de acessos padrão de memória: acesso à memória local, à memória global e a memória

remota. O acesso mais rápido é o da memória local; o mais lento é o acesso da memória remota. Os processadores estão divididos em vários grupos ("clusters"). Cada grupo é por si só um multiprocessador UMA ou NUMA. Os grupos estão conectados aos módulos de memória compartilhada global. O sistema todo é considerado um multiprocessador NUMA.

Os primeiros modelos NUMA não tinham caches coerentes. A questão de manter coerência era deixada para o programador. Uma variação de tal modelo, chamada CC-NUMA, ou "Cache Coherent Nonuniform Memory Access" usa memória compartilhada distribuída e diretórios de cache para manter o cache coerente. Exemplo desse modelo inclui o Multiprocessador Stanford Dash [43].

3.4.1.3 Modelo COMA

No modelo COMA, similarmente ao modelo NUMA, a memória principal é distribuída, e convertida em um grande cache gerenciado por hardware. Não há hierarquia de memória em cada nodo do processador. O conjunto de todos os caches formam um espaço de endereço global. Uma conseqüência disto é que a localização de um item de dados dentro da máquina é desacoplada de seu endereço fisico, e o item de dados é automaticamente migrado ou replicado na memória principal por hardware. O acesso aos caches remotos é assistido pelos diretórios de caches distribuídos. Dependendo da rede de interconexão usada, algumas vezes, diretórios hierárquicos são utilizados para ajudar a localizar cópias de cada bloco. A localização dos dados inicialmente não é crítica, porque eles serão eventualmente migrados para onde serão usados. Este modelo é ilustrado na Figura 3.10.

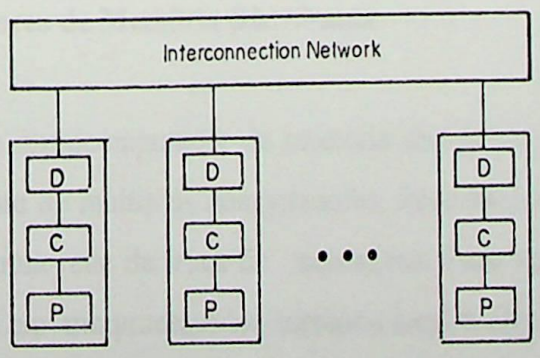


Fig. 3.10 - Modelo de Multiprocessador COMA

A principal vantagem das máquinas COMA é que o usuário não tem que se preocupar com a colocação explicita e movimentação dos dados entre as memórias físicas, mesmo do ponto de vista do desempenho, desde que os dados automaticamente migram (ou são replicados) para onde eles serão usados, e nesse sentido, é apresentado com o modelo de programação familiar de uma máquina de memória compartilhada e centralizada com caches. Entretanto, as máquinas COMA também possuem muitas desvantagens. O uso de uma estrutura de diretório hierárquico para localizar um dado na ocorrência de um "miss", aumenta a latencia para requisições que não sejam satisfeitas no nodo local. O protocolo de coerência de cache é mais complexo porque é necessário assegurar que a última cópia de um item de dado não seja trocada na memória principal global.

Exemplos de tais modelos incluem o "Kendall Square Research Machine" KSR1 [43] e o "Swedish Institute of Computer Science's Data Diffusion Machine" (DDM) [58].

Stenstrom e outros [65] descrevem as vantagens e desvantagens relativas das arquiteturas COMA e CC-NUMA e as características das aplicações que são comuns para fazer com que um execute melhor que o outro. Num artigo recente [60], é descrito o desempenho paralelo de dois representantes dessas classes de arquitetura: O multiprocessador Stanford Dash e o Kendall Square Research KSR1, usando um conjunto de importantes Kernels computacionais e aplicações científicas reais. É examinado as diferenças de desempenho resultantes da natureza CC-NUMA/COMA das máquinas assim como as diferenças específicas das implementações .

3.4.2 Multicomputadores de Memória Distribuída

Um sistema de multicomputador de memória distribuída é ilustrado na Figura 3.11. O sistema consiste de múltiplos computadores, frequentemente chamados nodos, interconnectados por uma rede de troca de mensagens. Cada nodo é um computador autônomo, consistindo em um processador, memória local e, algumas vezes, discos ou periféricos de E/S. Todas as memórias locais são privadas e acessíveis apenas pelos

processadores locais. Por esta razão, multicomputadores são chamados de "no-remote-memory-access machines" (NORMA).

As topologias de rede estáticas usadas para construir multicomputadores incluem o anel, árvore, malha, hipercubo, e assim por diante. Vários padrões de comunicação são exigidos entre os nodos, tais como um-para-um, broadcasting, permutações e multicast.

A rede de troca de mensagens fornece, conexões ponto-a-ponto estáticas entre os nodos. As mensagens são roteadas entre os nodos. Um protocolo send/receive (enviar/ receber) precisa ser incorporado para garantir programação consistente com velocidade de mensagens uniforme. Essas mensagens são passadas explicitamente e podem ficar escondidas ou não do programador, através de hardware e do compilador. Programas e seus dados devem ser alocados aos processadores, a fim de minimizar o custo ("overhead") da passagem de mensagens. Como resultado, um novo modelo de programação e ferramentas são necessários para multicomputadores.

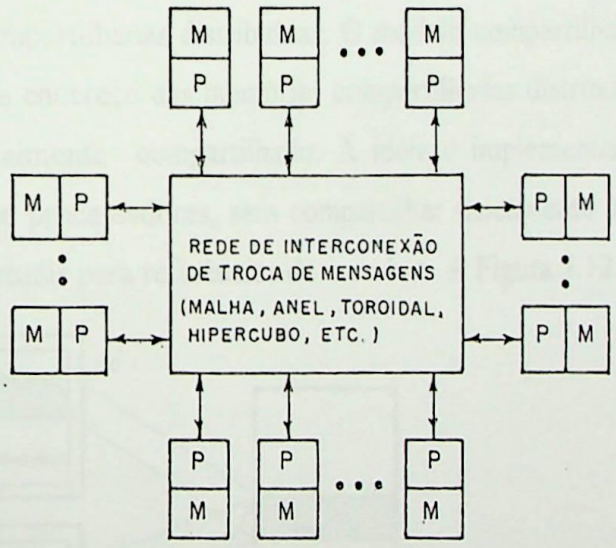


Fig. 3.11 - Modelo Genérico de um multicomputador com troca de mensagens.

Com nodos de processador/memória distribuídos, essas máquinas tendem a exibir um desempenho escalável. Esta é a maior vantagem que multicomputadores desfrutam sobre as máquinas de memória compartilhada. Entretanto, a falta de programabilidade continua sendo o maior obstáculo à adoção de multicomputadores em larga escala.

3.4.3 Multiprocessadores de Memória Virtual Compartilhada

A idéia tradicional da memória virtual é expandir o uso de memória física entre os vários programas, com a ajuda do sistema de armazenagem secundária (por exemplo, discos). Todos os programas podem ser carregados para dentro e fora da memória física, dinamicamente sob a coordenação do sistema operacional. Tal mecanismo fornece ao usuários espaço de memória "ilimitado" para trabalhar e torna a tarefa de programação muito mais fácil.

Em sistemas de multiprocessadores, tais como nos modelos NUMA e COMA, a tarefa de gerenciar o movimento dos códigos e dados entre as unidades de memória distribuída pertence ao programador. O trabalho é similar em estilo, à tarefa de gerenciar a migração de dados entre a memória primária e a secundária antes da introdução da memória virtual. Multiprocessadores com memória compartilhada virtual extendem o conceito de memória virtual para o processamento altamente paralelo.

Como visto na seção 3.3.1, os modelos de Processador NUMA e COMA usam memórias compartilhadas distribuídas. O modelo compartilhado virtual combina todos os espaços de endereço das memórias compartilhadas distribuídas em um único espaço virtual globalmente compartilhado. A idéia é implementar memória compartilhada numa rede de processadores, sem compartilhar fisicamente a memória. O sistema usa endereços virtuais para referências da memória. A Figura 3.12 ilustra este modelo.

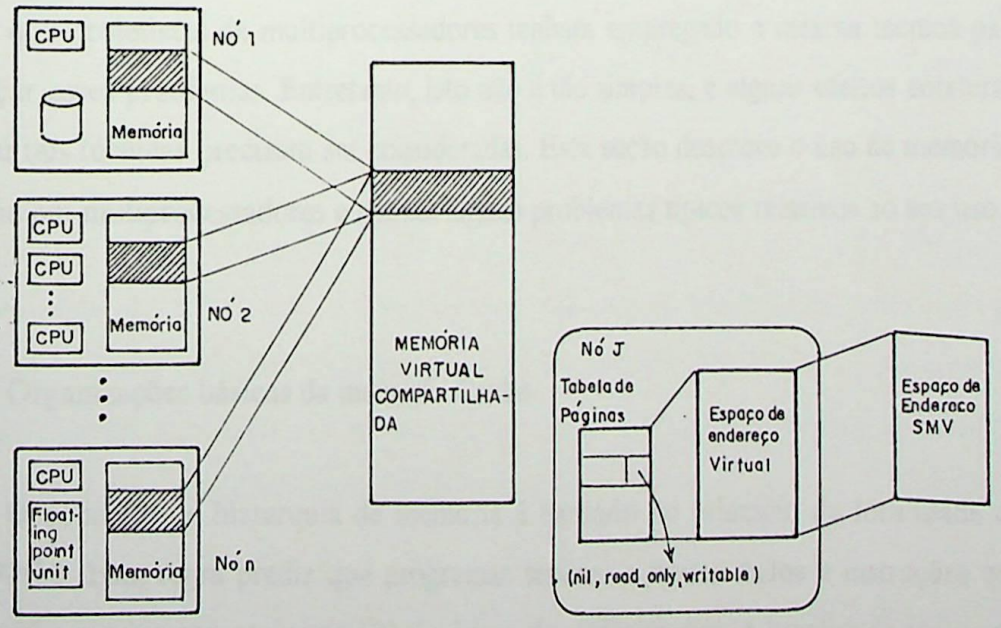


Fig. 3.12 - Modelo de Memória Compartilhada Virtual [50]

Há dois esquemas para implementar este modelo [49]:

1□ Memória Compartilhada Distribuída: A Cada processador é dada uma porção da memória virtual compartilhada para declarar seus endereços. Trata-se do modelo NUMA.

2□ Memória Compartilhada Virtual: Neste caso, existe somente uma grande memória cache em cada processador. A memória compartilhada é completamente virtual. Isto é o modelo COMA.

3.4.4 Sistemas de Cache para Multiprocessadores

Como foi mencionado na seção 3.3.1, os três maiores problemas relativos aos multiprocessadores de memória compartilhada são: Contenção de memória, contenção da comunicação e tempo de latência. Esses problemas todos contribuem para aumentar o tempo de acesso à memória e, por essa razão, a velocidade de execução do processador diminui.

Em uniprocessadores tradicionais, as memórias cache servem como uma forma importante de reduzir o tempo médio de acesso à memória. Como o cache é mais rápido que a memória principal, o processador pode executar mais rápido, quando ele acessa o cache. A largura de faixa é aumentada e a latência é reduzida. Portanto, é natural que projetistas de multiprocessadores tenham empregado a mesma técnica para endereçar esses problemas. Entretanto, isto não é tão simples, e alguns efeitos colaterais por usar tais técnicas precisam ser consideradas. Esta seção descreve o uso de memórias de cache em mutiprocessadores e aponta alguns problemas típicos relativos ao seu uso.

3.4.4.1 Organizações básicas da memória Cache

O conceito de hierarquia de memória é baseado no princípio da localidade ou regra 90/10. Esta regra prediz que programas tendem a reusar dados e instruções que teriam sido usados recentemente [9]. Isto tem duas dimensões: a localidade temporal, que estabelece que, itens acessados mais recentemente provavelmente serão acessados

num futuro próximo; e a localidade espacial, que estabelece que itens cujos endereços estão próximos um do outro tendem a ser conjuntamente referenciados. A hierarquia de memória é organizada em diferentes níveis, cada nível menor, mais rápido e com custo por byte mais caro que o nível abaixo. Os níveis de hierarquia englobam um ao outro. Todos os dados em um determinado nível são encontrados em um nível mais abaixo.

Cache é o nome do primeiro nível de hierarquia de memória, entre o Elemento de Processamento e o Elemento de Memória. As memórias cache estão bem documentadas [61-64]. Esta seção trata das estratégias para escrita e dos esquemas de alocação. Algumas caracteristicas de desempenhos discutidos.

a) Estratégias para escrita ("Write Strategies")

Há duas estratégias quando se escreve no cache:

1. "Write-through" (WT): O dado é escrito imediatamente tanto para o bloco no cache como para o bloco no nível de memória mais abaixo.
2. "Write-back" (WB): O dado é escrito imediatamente somente para o bloco no cache. O bloco modificado é escrito para a memória principal somente quando ele for trocado por outro bloco no cache.

Para reduzir a frequência de re-escrita de blocos no momento da sua troca, um bit de sinalização chamado "dirty bit" é usado para indicar se o bloco foi ou não modificado enqùanto no cache. Se não foi, o bloco não é re-escrito, pois o nível mais abaixo tem a mesma informação do cache.

O cache WT requer mais ciclos de barramento ou da rede para acessar a memória principal. Entretanto, a principal memória sempre tem a cópia mais atual do dado. Em contraste, o cache WB permite à CPU continuar sem esperar pelo acesso à memória. Desde que todas as operações de escrita não passem para a memória principal, WB usa menos largura de faixa da memória.

Portanto, multiprocessadores utilizam WB para reduzir o tráfico de memória para o processador e WT para manter o cache e a memória consistentes.

b) Caches com Mapeamento direto e caches associativos

A transferência de informação da memória principal para o cache é conduzida em unidades de blocos de cache ou linhas de cache. Três esquemas de alocação de blocos são apresentadas abaixo. Cada esquema tem seu próprio mérito e limitações.

Blocos em caches são chamados "block frames", a fim de distinguí-los dos correspondentes blocos na memória principal. O cache contém **m** block frames , onde $\mathbf{m= 2^r}$, e a memória principal contém **n** blocos, onde $\mathbf{n= 2^s}$. Cada bloco (ou block frame) é assumido ter **b** palavras, onde $\mathbf{b= 2^w}$. Portanto, o cache consiste de $\mathbf{m*b=2^{(r+w)}}$ palavras. A memória principal tem $\mathbf{n*b=2^{(s+w)}}$ palavras endereçadas por **(s+w)** bits. Quando os blocos são divididos em $\mathbf{v=2^t}$ conjuntos, $\mathbf{k=m/v=2^{(r-t)}}$ blocos estão em cada conjunto.

Cache de Mapeamento Direto

Esta organização de cache é baseada no mapeamento direto de $\mathbf{n/m=2^{(s-r)}}$ blocos de memória, separados por distância igual de um "block frame" no cache. Cada bloco tem somente um lugar em que ele pode residir no cache. O mapeamento é usualmente definido como *endereço do "Block-frame " **módulo** número de blocos no cache*. Isto está ilustrado na Figura 3.13.

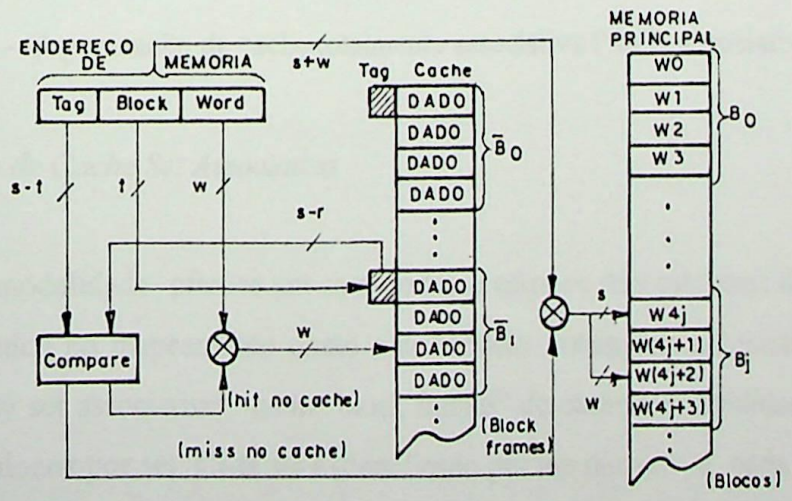


Fig. 3.13 - Organização de Cache com mapeamento direto [50]

A relativa simplicidade do hardware usado para implementar o mapeamento direto é uma vantagem. Entretanto, o rígido mapeamento pode resultar numa taxa de "hit" mais pobre que o mapeamento associativo.

Cache totalmente associativo

Nesta organização de cache, cada bloco na memória principal pode ser colocado em qualquer dos "frames" disponíveis, como ilustrado na Figura 3.14. O nome "fully associative cache" é derivado do fato de uma pesquisa no cache requer que o "tag" seja comparado com todos os "tags" do cache. Uma memória associativa (CAM – Contend Addressable Memory) é necessária para alcançar a comparação paralela com todos os tags simultaneamente. Como resultado, há um aumento do custo da implementação do hardware.

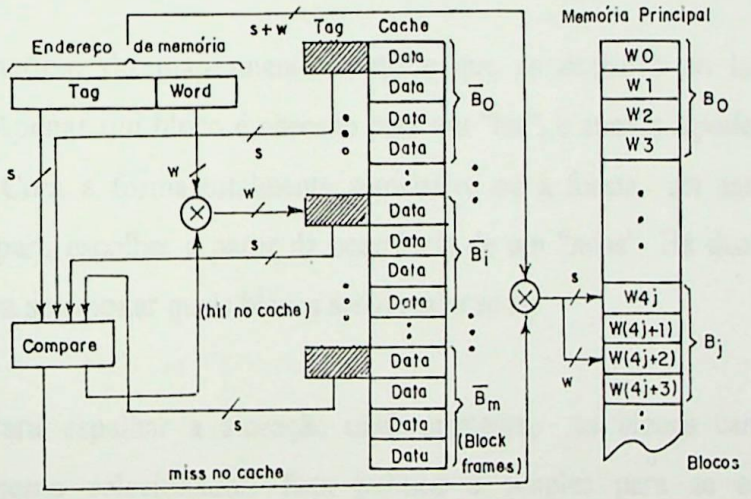


Fig. 3.14 - Organização de cache totalmente associativa ("fully associative") [50]

Organização de Cache Set Associativo

Esta modalidade oferece um compromisso entre os dois extremos de projeto de caches baseados no mapeamento direto e no modelo totalmente associativo. Em um cache "K-way set associative" os **m** "block frames" do cache são divididos em **v=m/k** sets, com **k** blocos por set. Cada set é identificado por um número **d**, onde 2^d=**v**. Isto é ilustrado na Figura 3.15. Na prática, o tamanho **k** do "set"é igual a 2, 4, 8, 16, ou 64 ,

dependendo do compromisso entre tamanho do bloco w, a capacidade do cache m e outros fatores de custo/desempenho.

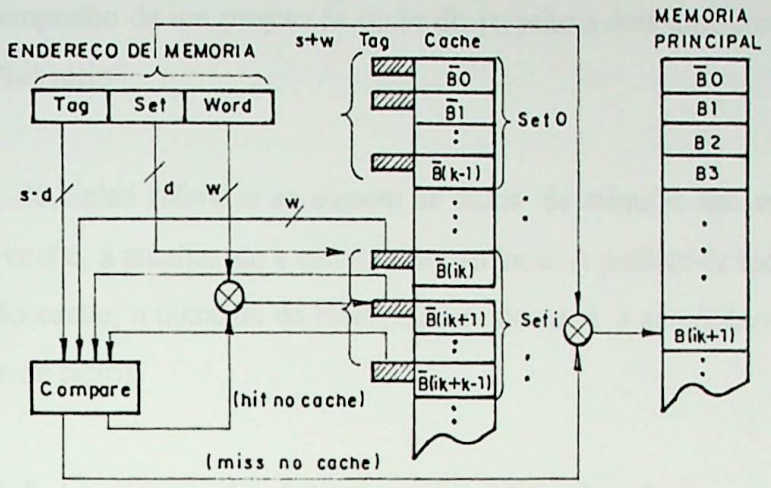


Fig. 3.15 - Organização de Cache Set Associativo [50]

c) Política de Realocação

Um benefício do mapeamento direto é que as decisões do hardware são simplificadas: Apenas um bloco é checado para um "hit", e apenas aquele bloco pode ser realocado. Com a forma totalmente associativa ou a forma set associativo, há muitos blocos para escolher a partir da ocorrência de um "miss". Há duas estratégias empregadas para selecionar quais blocos serão realocados:

1. Random: Para espalhar a alocação uniformemente, os blocos candidatos são randomicamente selecionados. Esta política é simples para se construir em hardware.

2. Least-Recently Used (LRU): Para reduzir a chance de retirar informações que serão novamente necessárias, o acesso aos blocos são registrados (contador). O bloco a ser realocado é aquele que não foi usado pelo período de tempo mais longo (Localidade Temporal).

A política de realocação exerce uma maior influência nos caches menores que nos maiores, quando há mais escolha sobre o que pode ser realocado.

d) Características do desempenho de caches

O desempenho de um projeto de cache diz respeito a dois aspectos: O contador de ciclos e o "hit ratio".

- Contador de ciclos refere-se ao número de ciclos de máquina necessários para o acesso ao cache, a atualização e controle de coerência. A política de escrita usada, o tamanho do cache, o tamanho do bloco, número do set, e a associatividade afetam o contador de ciclos.

- O "hit ratio" determina quão efetivamente o cache pode reduzir o tempo total de acesso à memória. Isto é afetado pelo tamanho do cache e pelo tamanho do bloco, de diferentes maneiras. Geralmente, o "hit ratio" aumenta com respeito ao aumento do tamanho do cache. Entretanto, uma taxa de 100% jamais acontecerá. Também a taxa de hit melhora quando o tamanho do bloco aumenta, devido a localidade espacial ao referênciar blocos maiores de dados/instruções.

Depois de um certo ponto, chamado tamanho de bloco ótimo, a taxa de hit decresce com o aumento do tamanho do bloco.

Simulação orientada através de "traces" ("trace driven simulation") e modelamento analítico são dois modelos complementares para estudar o desempenho de cache. Ambos devem ser aplicados em conjunto para prover uma avaliação de desempenho satisfatória. Um estudo detalhado de desempenho de cache é descrito por Przybylski e outros, em [65]. É investigado a relação de compromisso entre tamanho do cache e CPU/ciclo de tempo do cache, "set associativity" e tempo de ciclo, e entre tamanho de bloco e velocidade da memória principal.

3.4.4.2 Problema de Coerência do cache

Caches em um ambiente de multiprocessamento introduzem o problema de coerência de cache. Quando múltiplos processadores mantêm localmente cópias no seus caches individuais de uma certa localização da memória compartilhada, qualquer

modificação feita localmente desta localização pode resultar em uma visão globalmente inconsistente da memória. Esquemas de coerência de cache previnem este problema mantendo um estado uniforme para cada bloco de dados armazenados no cache. Inconsistências de cache podem ser causadas por compartilhamento de dados, migração de processos ou operação de E/S.. A Figura 3.16 ilustra um problema de coerência de cache no compartilhamento de dados de uma variável que pode ser atualizada. Antes da atualização, as três cópias da variável compartilhada X são consistentes. Depois de P1 atualizar a variável X em X' no seu poderemos ter uma inconsistência entre as três cópias.

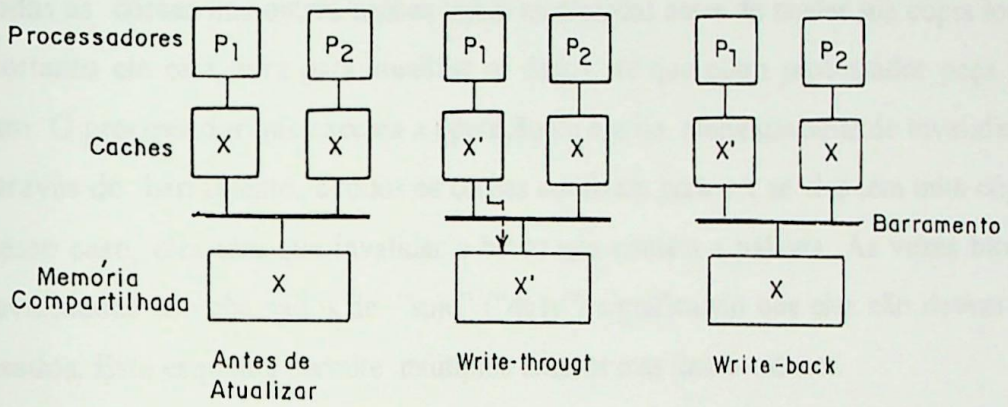


Fig. 3.16 - Inconsistência apresentada por uma variável compartilhada numa operação de escrita.

Um grande número de protocolos de coerência de cache para resolver este problema foram propostos por Stenstrom, em [66], e por Agarwal e outros, em [67]. Em geral, um protocolo de coerência de cache consiste em um conjunto de possíveis estados nos caches locais, o estado na memória compartilhada, e as transições de estado causadas pelas mensagens transportadas através da rede de interconexão para manter a memória coerente. Estes protocolos são baseados em software, hardware ou uma combinação de ambos para implementação. Há dois tipos de protocolos de hardware: o protocolo snoopy e o protocolo baseado em diretório.

a) Protocolo Snoopy

O Protocolo Snoopy obtém consistência de dados entre os caches e memória compartilhada através de um mecanismo de observação do barramento ("bus

watching"). Todo cache que também tem uma cópia dos dados de um bloco de memória física tem uma cópia da informação sobre ele. Estes caches normalmente estão conectados a um barramento de memória compartilhada e todos os controladores de cache monitoram ou "bisbilhoteiam" ("snoopy") o barramento para determinar se ou não eles possuem uma cópia do bloco compartilhado.

Protocolos Snoopy são de dois tipos que dependem do que acontece numa operação de escrita ("write"):

1. "Write Invalidate": O processador que executa a operação de escrita faz com que todas as cópias nos outros caches sejam invalidadas antes de mudar sua cópia local; Portanto ele está livre para atualizar os dados até que outro processador peça por isto. O processador que executa a operação de escrita emite um sinal de invalidação através do barramento e todos os caches conferem para ver se eles têm uma cópia; nesse caso, eles têm que invalidar o bloco que contém a palavra. Às vezes blocos invalidados são chamados de "sujo" ("dirty") significando que eles não devem ser usados. Este esquema permite múltiplos leitores mas único escritor.

2. "Write Broadcast": Em lugar de invalidar todo bloco que é compartilhado, o processador que realiza a operação de escrita envia ("broadcast") os novos dados pelo barramento; todas as cópias são então atualizadas com o novo valor. Este esquema envia continuamente operações de escrita ("write broadcast") para dados compartilhados enquanto apaga todas as outras cópias ("write invalidate") de forma que haja somente uma única cópia local para as subseqüentes operações de escrita. Protocolos "Write-Broadcast" normalmente permitem que blocos sejam identificados ("tagged") como compartilhado ("broadcast") ou privado (local).

A Figura 3.17 ilustra estes dois protocolos.

Os Protocolos Snoopy são populares porque pequenas configurações de multiprocessadores podem conviver com as restrições de largura de faixa impostas por um único barramento compartilhado para memória. O barramento compartilhado torna a implementação das ações de "broadcast" diretas. Porém, tal esquema não escalará além do número de processadores que podem ser acomodados em um barramento

(provavelmente não mais que 20). Exemplos de sistemas que implementam tal protocolo são o Sequent Simmetry Multiprocessor, Alliant FX multiprocessors, DEC Firefly multiprocessor e o Xerox Palo Alto Research Dragon Center Multiprocessor Workstation.

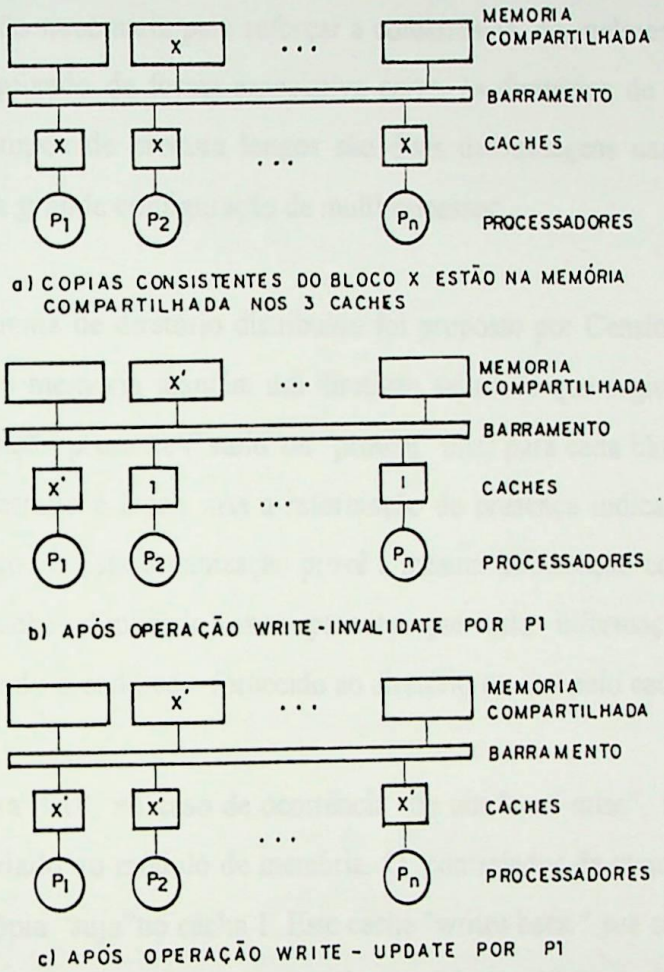


Fig. 3.17 - Protocolos para caches coerentes: write-invalidate e write-broadcast [50]

b. Protocolos Baseados em Diretório

Protocolos baseados em diretório mantêm um diretório em separado associado com a memória principal que armazena o estado de cada bloco da memória principal. Vários protocolos baseados em diretório diferem principalmente em como o diretório mantém informação e que informação ele armazena. Esta Informação pode incluir quais caches têm cópias do bloco, se está sujo ("dirty"), e assim por diante. As entradas do diretório podem ser de forma que diferentes pedidos possam ir para diferentes

memórias, reduzindo a contenção. Porém eles retêm a característica de que o estado compartilhando de um bloco sempre está em uma única localização conhecida.

Tang [68] propôs o primeiro esquema de diretório que usou um diretório central que contém duplicatas de todos os diretórios de cache. Este diretório central, provendo toda a informação necessária para reforçar a consistência, normalmente é muito grande e deve ser pesquisado de forma associativa como os diretórios de cache individuais. Contenção e tempos de procura longos são duas desvantagens usando um diretório central para uma grande configuração de multiprocessor.

Um esquema de diretório distribuído foi proposto por Censier e Feautrier [69]. Cada módulo de memória mantém um diretório separado que registra o estado ("dirt bit") e a informação presente ("valid"ou "present" bits) para cada bloco de memória. A informação de estado é local, mas a informação de presença indica quais caches têm uma cópia do bloco. Esta organização provê a mesma informação como o esquema de diretório de cache duplicado, mas permite que esta informação seja acessada diretamente usando o endereço fornecido ao diretório central pelo cache que requisitou.

Na Figura 3.18, no caso de ocorrência de um "read-miss" no cache 2 resulta num pedido enviado ao módulo de memória. O controlador de memória retransmite o pedido para a cópia "suja"no cache 1. Este cache "writes back " sua cópia. O módulo de memória pode prover uma cópia para o cache requisitante. No caso da ocorrência de um "write-hit" no cache 1, um comando é enviado ao controlador de memória que envia invalidações a todos os caches (cache 2) marcados no vetor de presença que reside no diretório Dl.

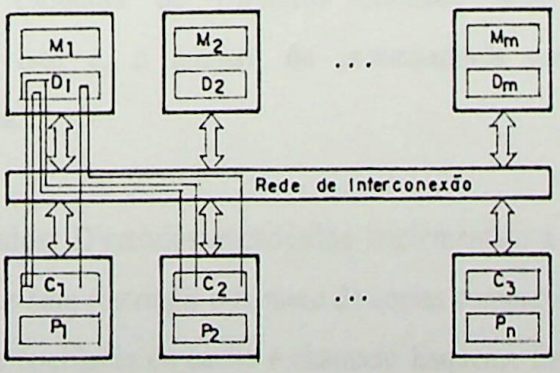


Fig. 3.18 - Conceito básico de um esquema de coerência de cache baseado em diretório [50]

Há três categorias diferentes de protocolo de diretório: Diretórios Mapa Completo ("Full Map Directories"), Diretórios Limitados ("Limited Directories")e Diretórios Encadeados ("Chained Directories").

1. Diretórios Mapa Completo: Este protocolo implementa entradas de diretório com 1 bit por processador e um bit sujo. Cada bit representa o estado do bloco no cache do processador correspondente. Se o bit sujo está ativado ("set"), então um e somente o bit de um processador está ativado ("set") e o processador pode escrever no bloco. Um cache mantém dois bits de estado por bloco. Um bit indica se um bloco é válido, e outro indica se um bloco válido pode ser escrito. O protocolo de coerência de cache tem que manter os bits de estado na memória do diretório e esses no cache consistente. Como resultado, o tamanho da entrada de diretório associado com cada bloco é proporcional ao número de processadores e a memória consumida pelo diretório é proporcional ao tamanho de memória multiplicado pelo tamanho do diretório. Este protocolo provê um salto superior para o desempenho de coerência de cache baseado em diretório centralizado. Porém, eles são caros para implementar, especialmente para multiprocessores que contém muitos processadores [66].

2. Protocolos Limitados: Estes protocolos são projetados para resolver o problema de tamanho de diretório. Eles restringem o número de ponteiros de cache para menos que o número real de caches. Então, o crescimento do diretório é limitado a um fator constante. Estes protocolos são considerados escaláveis com respeito ao "overhead" de memória porque o recurso necessário para implementá-los cresce aproximadamente linearmente com o número de processadores no sistema. Porém, o sucesso de um esquema de Diretório Limitado depende do grau de compartilhamento, isto é, o número de processadores que simultaneamente compartilham dados.

3. Diretórios Encadeados: Diretórios encadeados implementam a escalabilidade dos Diretórios Limitados sem restringir o número de cópias compartilhadas de blocos de dados. Este tipo de coerência de cache é chamado Esquema Encadeado ("Chained Scheme") porque mantém o registro de cópias compartilhadas de dados mantendo uma cadeia de ponteiros de diretório. Este esquema dirige comandos de consistência

para somente aqueles caches tendo cópias de um bloco. O "Scalable Coherence Interface" (IEEE P1596) propõe um esquema de diretório encadeado [70].

3.4.4.3 Modelos de Consistência de Memória

O problema de inconsistência de memória surge quando a ordem de acesso à memória difere da ordem de execução. As tradicionais máquinas de Von Neumann mapeiam uma seqüência SISD em uma seqüência de execução semelhante. Como resultado, acessos de memória são consistentes com a execução de programa (Veja Figura 3.19(a)). Esta propriedade é chamada " consistência sequencial" [71].

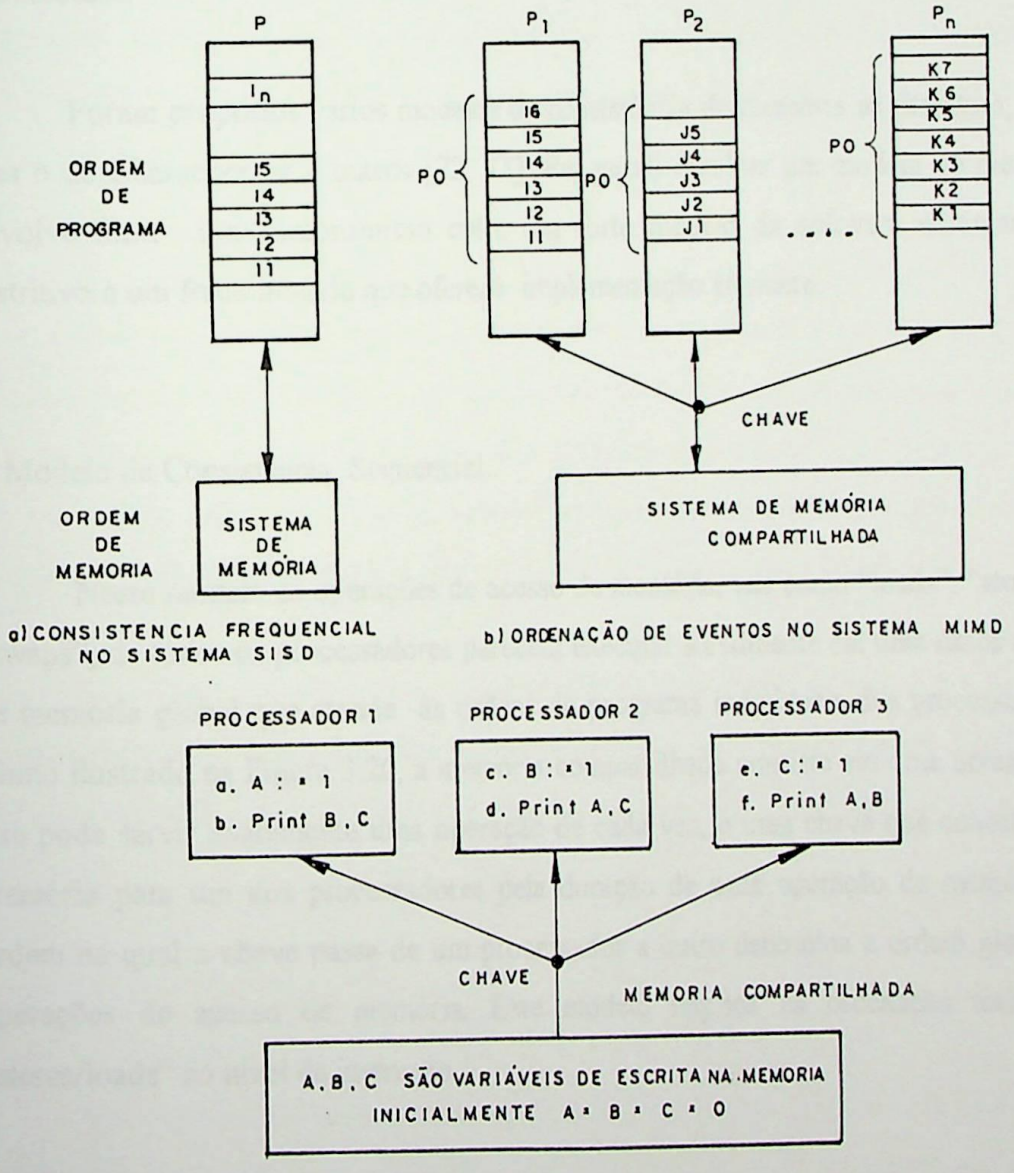


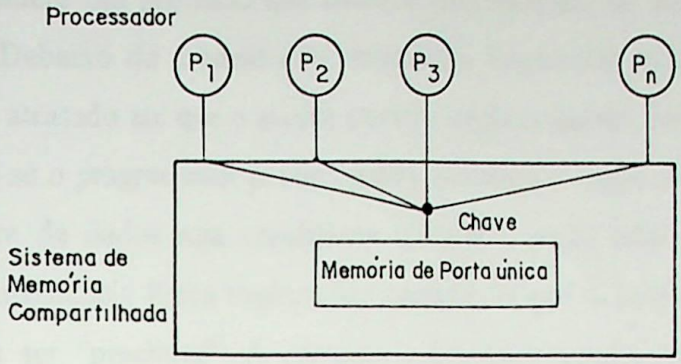
Fig. 3.19 - Ordem de acesso à eventos de memória num sistema uniprocessado e num sistema multiprocessado [50]

Em um multiprocessor com memória compartilhada, há múltiplas seqüências de instruções em diferentes processadores, como mostrado em Figo. 3.19(b). Diferentes modos de intercalar seqüências de instruções MIMD numa seqüência de acessos de memória global levam a diferentes comportamentos da memória compartilhada. Modelos de consistência especificam a ordem pela qual os eventos de um processo deveriam ser observados através de outros processos na máquina. Ordenação de eventos podem ser usados para declarar se um evento de memória é legal ou ilegal, quando vários processos estão acessando um conjunto comum de localizações de memória. A ordem de um programa é a ordem pela qual acessos de memória acontecem para a execução de um único processo, contanto que nenhuma reordenação de programa acontecesse.

Foram propostos vários modelos de consistência de memória na literatura, entre eles o de Gharachorloo e outros [72, 73]. Em geral, escolher um modelo de memória envolve fazer um compromisso entre um forte modelo de software minimamente restritivo e um fraco modelo que ofereça implementação eficiente.

a. Modelo de Consistência Sequencial

Neste modelo as operações de acesso de memória, tais como "loads", "stores" e "swaps", de todos os processadores parecem executar serialmente em uma única ordem de memória global que atende às ordens de programa individuais dos processadores. Como ilustrado na Figura 3.20, a memória compartilhada consiste em uma única porta que pode servir exatamente uma operação de cada vez, e uma chave que conecta esta memória para um dos processadores pela duração de cada operação de memória. A ordem na qual a chave passa de um processador a outro determina a ordem global de operações de acesso de memória. Este modelo implica na ordenação total de "stores/loads" ao nível de instrução.



Escola Federal de Engenharia
Biblioteca
MAUÁ
BIM
de Itajubá - EFEI

Fig. 3.20 - Modelo de Consistência de memória seqüencial [50]

Lamport, em [71], define consistência seqüencial como se segue: Um sistema de multiprocessor é sequencialmente consistente se o resultado de qualquer execução é o mesmo como se as operações de todos os processadores fossem executadas em alguma ordem seqüencial, e as operações de cada processador individual aparecessem nesta seqüência na ordem especificada por seu programa.

A ordenação rígida de todos os acessos à memória compartilhada no modelo de consistência seqüencial conserva a ordem do programa em todos os processadores. Um processador não pode emitir outro acesso até que o acesso mais recente à memória compartilhada pelo processador tenha sido globalmente realizada. Isto pode requerer a propagação de todos os acessos da memória compartilhada para todos os processadores, o qual são bastante demorados e caros.

Enquanto conceitualmente simples, este modelo impõe severas restrições nos acessos pedentes que um processo pode ter e inibindo, efetivamente, muitas otimizações de hardware, tais como "buffering" e pipelining, que poderiam aumentar desempenho.

b. Modelo de Consistência Fraca

Um modelo de consistência mais relaxado pode ser derivado relacionando a ordenação das requisições aos pontos de sincronização no programa. O modelo de consistência fraco proposto por Dubois e outros [74] está baseado na idéia acima e assegura que aquela memória só está consistente nos pontos de sincronização. Como um

exemplo, considere um processo que atualiza uma estrutura de dados dentro de uma seção crítica. Debaixo do modelo de Consistência Seqüencial, todo acesso dentro da seção crítica é atrasado até que o acesso anterior esteja completo. Mas tais demoras são desnecessárias se o programador já tem certeza que nenhum outro processo pode contar que a estrutura de dados seja consistente até que a seção crítica tenha terminado. Modelo de Consistência Fraca explora isto permitindo que os acessos dentro da seção crítica possam ser "pipelined". A correção é alcançada garantindo-se que todos os acessos prévios são executados no princípio e fim de cada seção crítica.

A vantagem dos modelos menos rígidos é o aumento do potencial desempenho. A desvantagem é a complexidade de hardware e um modelo de programação mais complexo. O programador deve estar atento das restrições impostas.

Uma extensão deste modelo é o "release consistency model" [73] implementado no protótipo do multiprocessor DASH [43]. Este modelo explora mais informações sobre sincronização classificando-os nos acessos de obter ("acquire") e liberar ("release"). Em contraste ao modelo consistência fraca, o modelo de consistência liberada ("release") não requer que acessos que seguem uma operação de "release" sejam atrasados até que a operação ocorra. Similarmente, não é requerido que uma operação "acquire" seja atrasada para seus acessos prévios.

3.5 Comentários

Neste capítulo é estudado os vários aspectos relacionados ao hardware paralelo. Especificamente é proposto uma taxonomia de máquinas paralelas baseada na estrutura física em lugar do modelo de programação implementada pelo paradigma. Finalmente, uma classe de máquinas paralelas, o multiprocessadores, é analisada e o problema de coerência de cache e os modelos de consistência de memória são endereçados.

Capítulo 4

ALGUNS SISTEMAS DE MULTIPROCESSADORES

4.1 INTRODUÇÃO

Esta capítulo descreve três sistemas de multiprocessadores implementados usando diferentes modelos. Primeiramente, um multiprocessador de memória compartilhado, representado pelo Cray Y-MP, é descrito. Esta classe de máquina representa o modelo tradicional de supercomputador com memória centralizada e fortes características vetoriais. A seguir é examinado dois modelos para implementar máquinas de alto desempenho escaláveis com memória compartilhada fisicamente distribuída. Primeiro, a máquina Stanford DASH é descrita como exemplo de uma máquina de memória distribuída com cache coerente baseado em diretório (CC-NUMA). Segundo, o Sistema Kendall Square Research KSR1 é descrito como um exemplo de arquitetura COMA. Neste caso a memória principal é um grande cache administrado por hardware, com o endereço lógico desacoplado do endereço fisico. Este sistema é um forte candidato a ser um marco representativo para futuras implementações de Processadores Massivamente Paralelos [75].

O modelo de hardware paralelo unificado introduzido no capítulo anterior é utilizado para caracterizar estes sistemas.

4.2 MULTIPROCESSADOR VETORIAL CRAY Y-MP

A arquitetura do processador Cray Y-MP é muito similar a do seu antecessor Cray X-MP [75]; A diferença principal é que o Processador Cray Y-MP pode endereçar um espaço de memória maior (32 bits de endereço no Y-MP contra 24 bits no X-MP).

Um diagrama de bloco esquemático para o Cray Y-MP8I/8128 é mostrado em Figura 4.1. O sistema é configurado com até oito Elementos de Processamento distintos

que compartilham a memória central, a seção de E/S, a seção de comunicação e o clock de tempo real. A Memória Principal com múltiplas portas ("multi-ported memory") forma uma estrutura hierarquica de 3 niveis consistindo em duas seções, cada uma contendo oito bancos onde cada banco contém oito sub-bancos. Há assim um total de 128 bancos intercalados ("interleaved") ou Elementos de Memória, cada um com capacidade de 1M operandos de 64 bits. Como a arquitetura de memória deste modelo contém só duas seções, somente dois dos caminhos para a memória podem ser ativos por vez ao invés dos quatro caminhos ativos previamente mencionados.

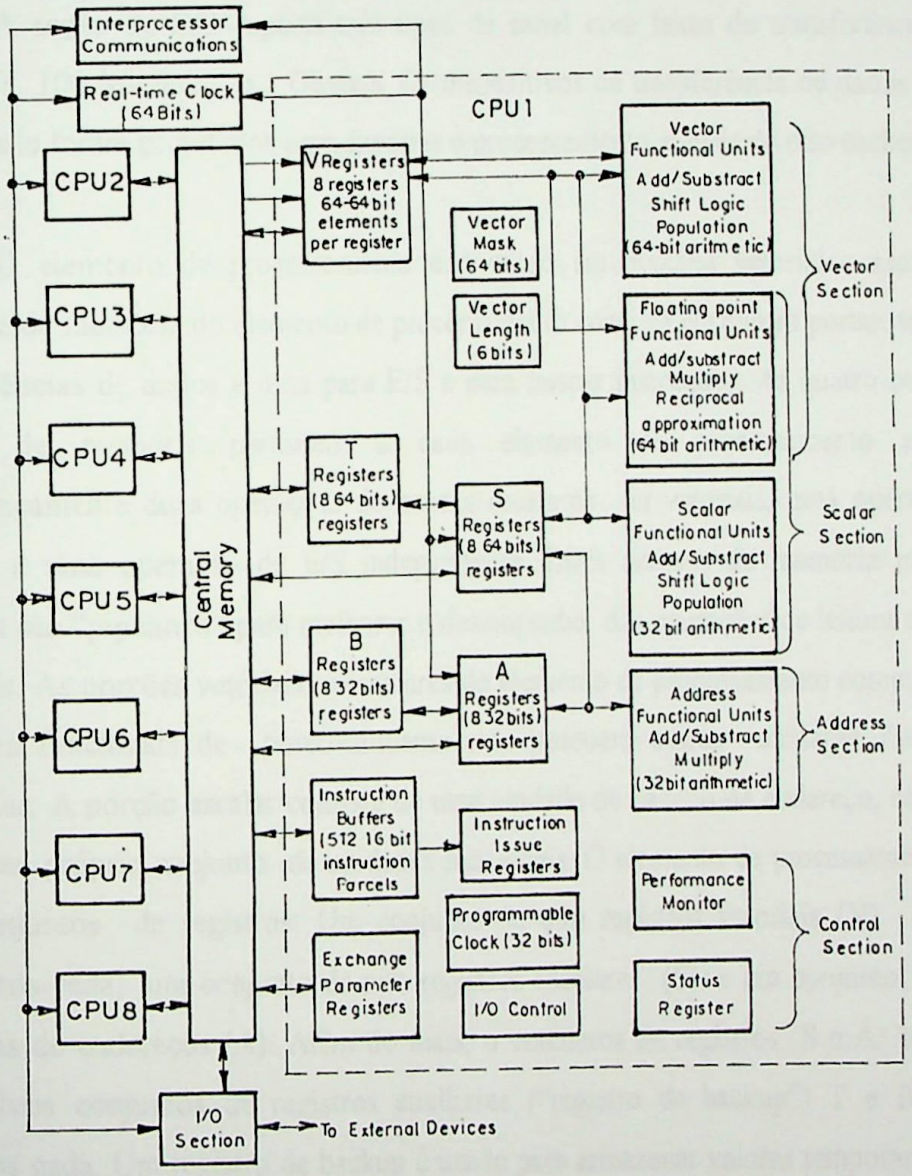


Fig.4.1 - Organização do sistemas Cray Y-MP8I/8128 [50]

A seção de comunicação entre elementos de processamento contém agrupamentos de registros compartilhados para propósitos de sincronização. Cada

agrupamento consiste em oito registros endereços, oito registros escalares, e oito registros de semáforo. Note que a comunicação de dados vetoriais entre elementos de processamento é feita através da memória compartilhada.

O clock de tempo real consiste de um contador de 64 bits que incrementa a contagem a cada ciclo de máquina. Como o clock avança de forma síncrona com a execução do programa, ele pode ser usado para temporizar a execução das instruções com uma contagem exata.

A seção de E/S suporta três tipos de canal com taxas de transferência de 6 Mbytes/s. 100 Mbytes/s, e 1 Gbyte/s. Os dispositivos de transferência de dados de alta velocidade foram projetados para suportar o processamento através de oito caches.

O elemento de processamento é dividido em porções vetorial e escalar. A interface de memória do elemento de processamento consiste em quatro portas: três para transferências de dados e uma para E/S e para buscar instruções. As quatro portas de acesso de memória permitem a cada elemento de processamento executar simultaneamente duas operações de buscas escalares ou vetoriais, uma operação de "store" e uma operação de E/S independente. Estes acessos de memória paralelos também são "pipelined" para melhorar o desempenho das operações de leitura e escrita vetoriais. As porções vetoriais e escalares do elemento de processamento compartilham unidades funcionais de ponto-flutuante mas possuem outras unidades funcionais separadas. A porção escalar consiste de uma unidade de cálculo de endereço, cada uma tendo seu próprio conjunto de unidades funcionais. O elemento de processamento tem três conjuntos de registros: Um conjunto de oito registros vetoriais (V) com 64 elementos cada, um conjunto de oito registros escalares (S) e um conjunto de oito registros de endereços (A). Além do mais, o conjuntos de registros S e A têm seus respectivos conjuntos de registros auxiliares ("registro de backup") T e B de 64 registros cada. Um registro de backup é usado para armazenar valores temporariamente quando o correspondente conjunto de registros primário está cheio e um registro precisa de ser desocupado para dar espaço para outro valor. As unidades funcionais do elemento de processamento são totalmente pipelined.

O elemento de processamento Cray Y-MP tem instruções de um "parcel", dois

"parcel" e de três "parcel". Um parcel corresponde a 16 bits. O elemento de processamento possui uma arquitetura LOAD/STORE, isto é, a memória é acessada explicitamente. e só por instruções de transferência de dados. Todas as instruções de computação são instruções do tipo registro para registro. O elemento de processamento pode buscar instruções de um "parcel" a uma taxa de pico de uma instrução por ciclo de máquina; Instruções de dois "parcel" e de três "parcel" precisam de dois ciclos de máquina. O elemento de processamento tem um Cache de Instrução, chamado "I-Buffer", que é usado para armazenar 512 instruções de 16 bits ao mesmo tempo. O Cache de Dados de 656 operandos de 64 bits é representado pelos conjuntos de registros V, S, A, T e B. O tempo de ciclo é de 6 ns.

O Elemento de Processamento é altamente "pipelined" a fim de explorar o paralelismo de grânulo fino. O compilador tenta identificar e aumentar paralelismo de grânulo fino no código para tirar proveito desta caracteristica. Por exemplo, o compilador desmenbra os laços dos ciclos ("loop unroll") para aumentar paralelismo, e o software enfileira as operações de memória para amenizar as longas latências da memória (i.e., ele pré-carrega na iteração atual alguns dos valores de memória que serão utilizados na próxima iteração). O compilador da Cray Reasearch limita a quantia de desmembramentos de ciclos ao tamanho do Cache de Instruções para evitar repetidos "misses"do I-buffer para instruções de ciclo.

As interconexões entre os 8 elementos de processamento e os 128 elementos de memória no Cray YMP81/8128 são implementadas com uma rede "crossbar" de múltiplos estágïos, logicamente descrita na. Figura 4.2. Os blocos principais são chaves "crossbar" 4 por 4 e 8 por 8 e demultiplexadores 1 para 8. A rede é controlada por uma forma de circuito de comutação onde todos os conflitos são trabalhados antes do acesso à memória e todos os pedidos de uma determinada porta retornam à porta na ordem que eles foram requisitados.

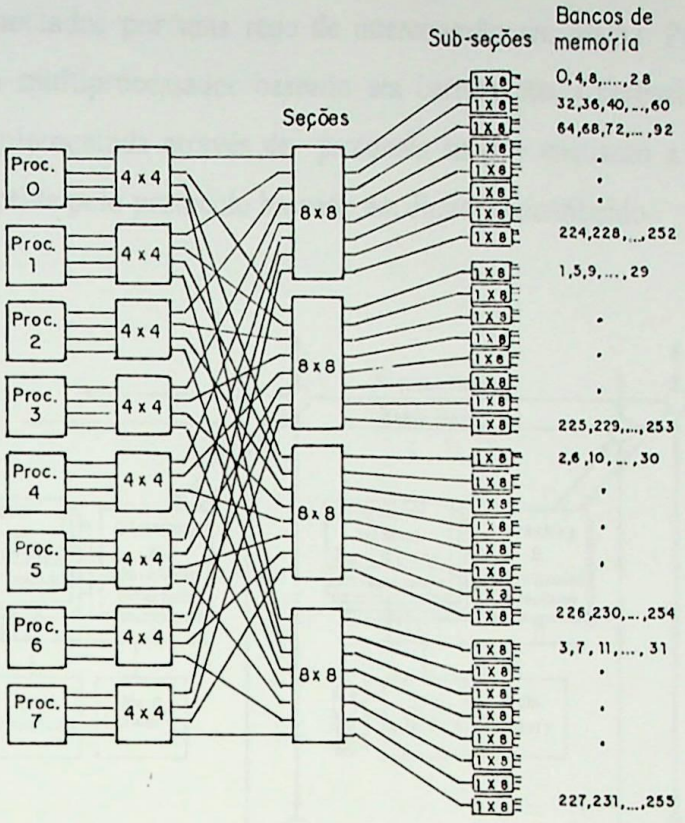


Fig. 4.2 - Diagrama lógico esquemático da rede crossbar entre os 8 elementos de processamento e os 128 elementos de memória no Cray Y-MP8I/8128 [50]

4.3 MULTIPROCESSADOR STANFORD DASH

Este é um sistema multiprocessado experimental que foi desenvolvido por Hennessy e outros colaboradores da Universidade de Stanford desde 1988 [44]. O nome DASH é uma abreviação de "Directory Architecture for Shared Memory"; (Arquitetura de diretório para Memória Compartilhada). A premissa fundamental atrás da máquina DASH é que é possível construir um processador de alto-desempenho escalável com um espaço de endereço único, caches coerentes e memória distribuída. A coerência baseada em diretório fornece ao DASH a facilidade do uso de arquiteturas de memória compartilhada, enquanto mantém a escalabilidade das máquinas de troca de mensagens.

A arquitetura do DASH tem uma estrutura de dois níveis, como mostrada na

Figura 4.3. No nível mais alto, a arquitetura consiste em um conjunto de agrupamentos ("clusters") conectados por uma rede de interconexão em malha. Por sua vez, cada "cluster" é um multiprocessador baseado em barramento. Coerência de Cache nos "clusters" é implementada através de protocolo snoopy enquanto a coerência entre "clusters" é mantida pelo protocolo baseado em diretório distribuído.

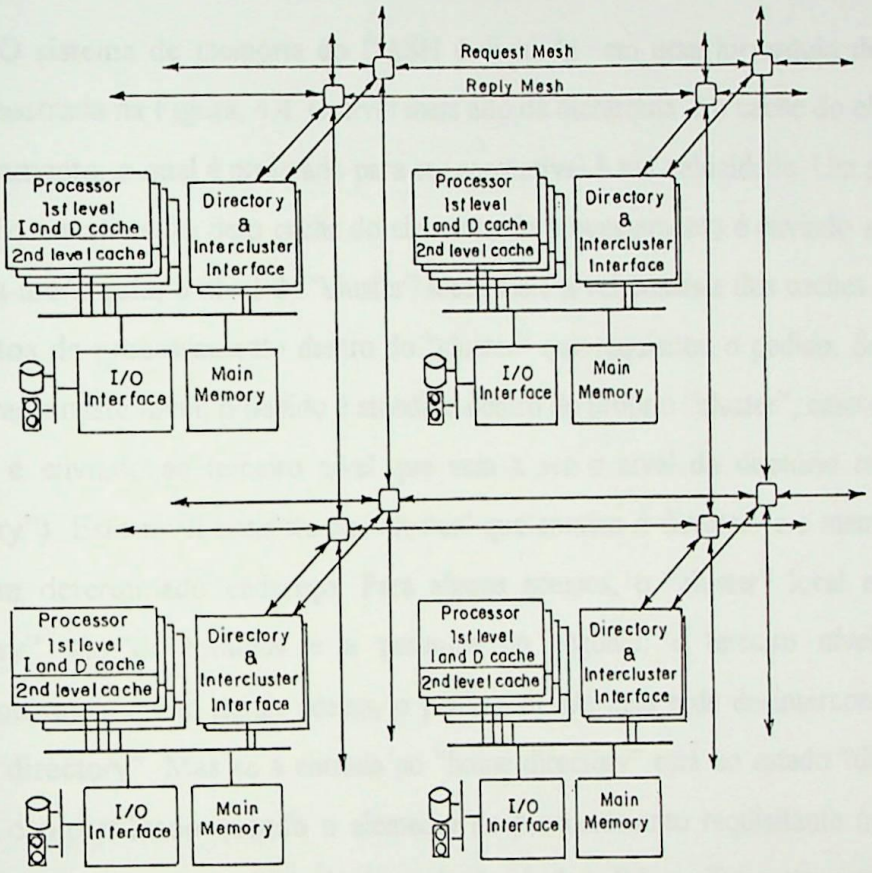


Fig.4.3 - Diagrama de bloco do protótipo DASH 2X2 [44]

O protótipo consiste em 16 "clusters", cada um com quatro elementos de processamento, num total de 64 elementos de processamento. Cada elemento de processamento consiste de um MIPS R3000, uma unidade funcional de ponto flutuante R3010, um Cache de Instruções de primeiro nível de 16 K operandos e um Cache de Dados de primeiro nível tipo "write-through" com 16 K operandos. O elemento de processamento tem 32 registros de propósito geral de 32 bits e a Unidade Funcional de Ponto-Flutuante tem 16 registros de 64 bits. O tamanho de Operando é 32 bits (4 bytes). O Cache de Dados de primeiro nível comunica-se com um Cache de segundo nivel tipo "write-back"de 64K operandos. A interface entre o cache de primeiro nível e o cache de segundo nível consiste em um buffer de leitura e um buffer de escrita com capacidades

para armazenar 4 operandos. Ambos os caches de primeiro e segundo níveis são do tipo "direct mapped" e o tamanho da linha de cache é de 16 bytes ou 4 operandos. O tamanho da Memória Principal é 7 M operandos para cada de seus 16 "clusters", i.e., aproximadamente 1.75 M operandos por elemento de processamento. A rede conectando estes 16 "clusters" forma uma topologia do tipo "wormhole routed 2-D". A largura de canal é 16-bits. O tempo de ciclo é 30-ns [44].

O sistema de memória do DASH é dividido em uma hierarquia de 4 niveis, como mostrada na Figura. 4.4. O nível mais alto da hierarquia é o cache do elemento de processamento, o qual é projetado para ser compatível à sua velocidade. Um pedido que não pode ser atendido pelo cache do elemento de processamento é enviado ao segundo nível da hierarquia, o nível do "cluster" local. Este nível consiste dos caches dos outros elementos de processamento dentro do "cluster" que requisitou o pedido. Se o dado é encontrado neste nivel, o pedido é atendido dentro do próprio "cluster", caso contrário o pedido é enviado ao terceiro nivel que vem a ser o nível do diretório raiz ("home directory"). Este nível consiste no "cluster" que contém o diretório e a memória fisica para um determinado endereço. Para alguns acessos, o "cluster" local e o "home directory" são os mesmos e a pesquisa no segundo e terceiro nível acontece simultaneamente. Em geral, porém, o pedido viajará pela rede de interconexão até o "home directory". Mas se a entrada no "home directory" está no estado "dirty" ou no estado compartilhado quando o elemento de processamento requisitante necessita de acesso exclusivo, o quarto nível, chamado de nível de "cluster" remoto, será acessado. O nível de "cluster" remoto responde diretamente ao nível de "cluster" local enquanto também atualiza o diretório. As latências de acesso para um "cache hit", acesso à memória local e acesso à memória remota são aproximadamente 1, 30, e 100 ou 135 ciclos de máquina respectivamente(dependendo se um "miss" é satisfeito no "home directory" ou não).

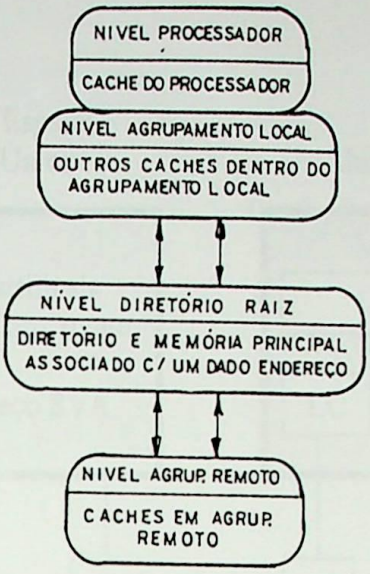


Fig.4.4 - Hierarquia de Memória da máquina DASH [44]

Além de caches coerentes que reduzem a latência, a máquina DASH suporta várias outras técnicas para esconder e minimizar a latência da memória como o Modelo de Consistência "Release" e o "prefetching" controlado por software.

4.4 MULTIPROCESADOR KENDALL SQUARE RESEARCH KSR1-32

4.4.1 Características Gerais

O KSR1 é um multiprocessador comercial utilizando modelo de memória compartilhada e implementando Arquitetura "Cache-Only-Memory" (COMA) [42,58], no qual a memória principal é um grande cache de único nível administrado em hardware, chamada ALLCACHE [76,77]. Este projeto representa a confluência dos conceitos de cache e memória virtual compartilhada que exploram a localidade necessária para implementar sistemas distribuídos escaláveis. Nesta arquitetura, cada processador acessa o mesmo espaço de endereço global, mas as localizações endereçadas estão fisicamente distribuídas entre os processadores (Figura 4.5) Este espaço de endereço é chamado Endereço Virtual de Sistema (SVA). A arquitetura genérica do KSR suporta um espaço de SVA de 2^{64} bytes. Contudo, a implementação do KSR1 suporta um espaço de SVA de 2^{40} bytes [77].

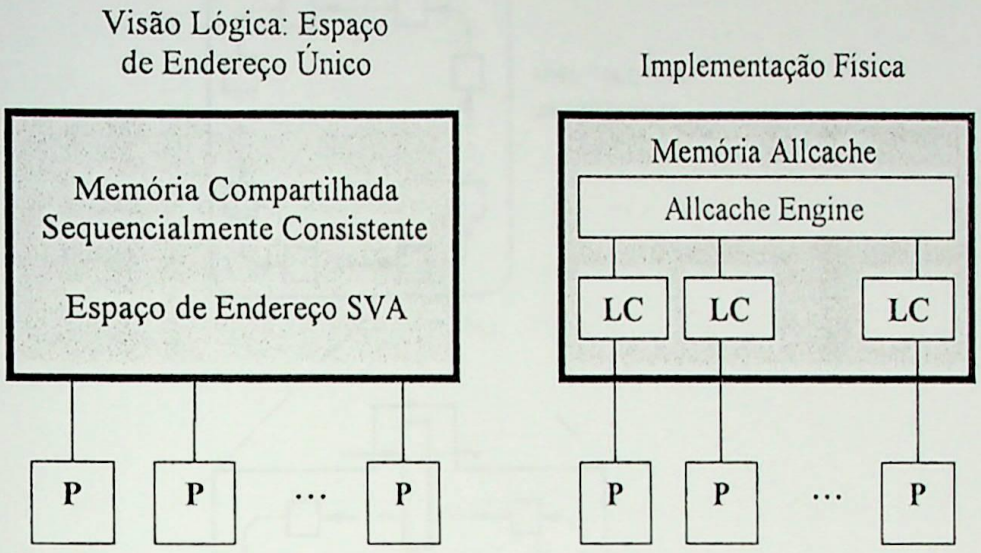


Fig.4.5 - Sistema de memória ALLCACHE: Visão lógica e física.

O KSR1 é organizado numa hierarquia de múltiplos anéis interconectados [75]. A escalabilidade é obtida conectando-se 32 elementos de processamento para formar um anel que opera a 1 Gb/sec (128 milhões de acessos por segundo). A largura de faixa da interconexão escala linearmente, pois cada "slot" do anel possa conter uma transação. O KSR1 utiliza uma hierarquia de dois níveis para interconectar 34 anéis de pacotes com elementos de processamento localizados nos anéis ao nível de folha. Cada anel ao nível de folha tem 32 elementos de processamento formando uma configuração massiva (1088 elementos de processamento). A Figura4.6 apresenta esta organização. O projeto do anel suporta um número arbitrário de níveis, permitindo até ultracomputadores possam ser construídos [75].

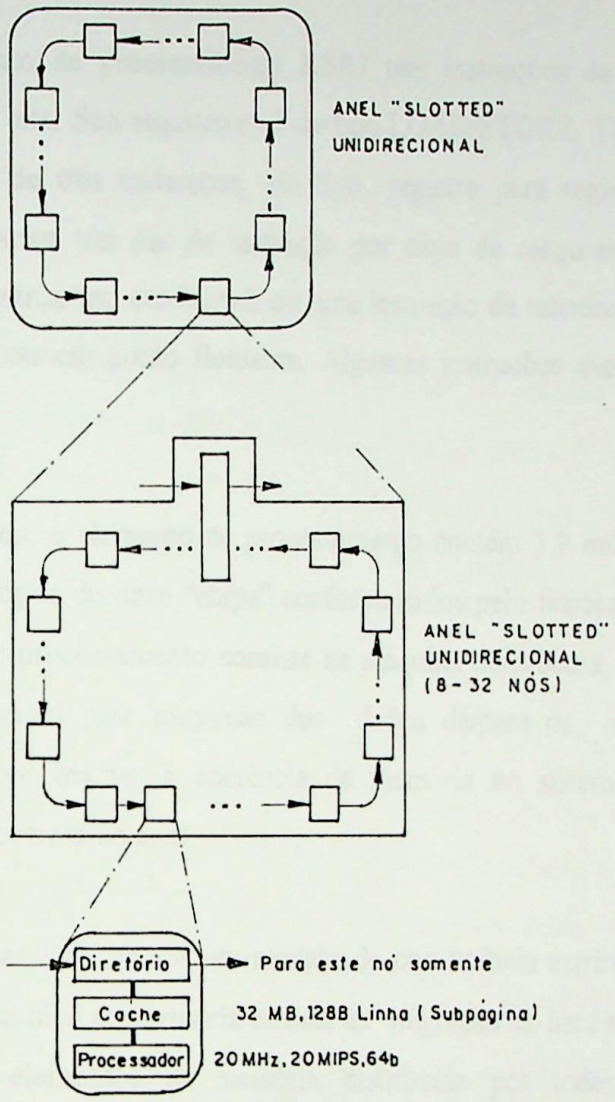


Fig. 4.6 - A arquitetura do KSR-1 com um anel para comunicação

4.4.2 Elementos de Processamento

Cada Elemento de Processamento compreende um cache primário, chamado Cache Local, e um processador superescalar de 64 bits com aproximadamente o mesmo desempenho que um IBM RS6000 operando à mesma taxa de clock [75] . Cada Cache Local tem uma capacidade de 32 Mbytes. Cada elemento de processamento contém 64 registradores de ponto flutuante e 32 registradores de ponto-fixo de 64 bits sendo projetado para realizar operações escalares e vetoriais. Por exemplo, 16 operandos de 64 bits podem ser pré-buscados ("prefetch") a cada vez. O processador também tem também um Sub-cache de 0.5 Mb capaz de satisfazer 20 milhões de acessos por

segundo. O ciclo de máquina é de 50 ns.

O elemento de processamento KSR1 tem instruções de 32 bits e manipula operandos de 64 bits. Sua arquitetura é do tipo LOAD/STORE. Todas as instruções de computação são de três endereços do tipo registro para registro. O Elemento de Processamento busca um par de instrução por ciclo de máquina, podendo executar, portanto, até 2 instruções: um branch ou uma instrução de referência à memória e uma operação inteira ou em ponto flutuante. Algumas instruções executam mais de uma operação.

Sem caches, o elemento de processamento contém 3.9 milhões de transistores divididos em seis tipos de doze "chips" confeccionados pelo fabricante. Três quartos de cada elemento de processamento consiste na máquina-de-procura, ou "search engine", o qual é responsável pela migração dos dados de/para os outros elementos de processamento, por manter a coerência de memória no sistema usando diretórios distribuídos e o controle do anel.

A máquina KSR provê um modelo de consistência estritamente seqüencial e administração dinâmica de memória através da migração de hardware e replicação de dados entre os elementos de memória distribuída por todos os elementos de processamento que utilizam os mecanismos da memória ALLCACHE. Com consistência seqüencial, todo elemento de processamento retorna o valor mais recente de um valor escrito, e resultados da execução em múltiplos elementos de processamento aparecem como o intercalamento de operações de elementos de processamento individuais quando executado numa máquina "multithreaded". Portanto, as latências não podem ser escondidas do elemento de processamento o qual deve esperar ("stall") por todo o período de duração de uma operação de escrita. Com ALLCACHE, um endereço se torna um nome, e este nome migra automaticamente ao longo do sistema e é associado com o elemento de processamento num estilo como do cache quando necessário. São feitas cópias de uma determinada célula por hardware e enviada a outras células para reduzir período de acesso. Chamanos de célula ao conjunto elemento de memória (ALLCACHE local) e elemento de processamento (subcache e unidade de processamento de instruções). Um elemento de processamento pode pré-buscar ("pre-fetch") dados dentro do seu cache local e pós-armazenar-los ("post-store") em outras

células. O hardware é projetado para explorar localidade espacial e temporal.

O Elemento de Processamento tem quatro unidades funcionais de processamento completamente "pipelined" onde uma operação pode ser iniciada em qualquer unidade funcional em cada ciclo de máquina. A Unidade de Co-execução (CEU) busca todas as instruções, controla a busca e armazenagem dos dados , controla fluxo de instrução ("branch") e faz a aritmética necessária para o cálculo do endereço. A Unidade de Flutuante-ponto (FPU) executa a aritmética de ponto-flutuante. A unidade de processamento de inteiros (IPU) executa aritmética com valores inteiros. A Unidade de Entrada/Saída Externa (XIU) executa o acesso direto à memória (DMA), Entrada/Saída programada, e contém vários registros de temporização. As quatro unidades funcionais de processamento operam em paralelo.

O Elemento de Processamento executa uma busca de instrução por ciclo de máquina, a menos que uma parada ("stall") aconteça causado pelo custo ("overhead") do gerenciamento do subcache e do cache. Qualquer instrução de load/store pode causar uma parada para permitir que o sistema recupere dados do cache local ou de uma célula remota. Assim, as paradas estão relacionadas com a proximidade dos dados relacionados.

Há 3 tipos de registros: Registros Gerais, Registros de Controle e Localizações de Controle. Os Registros Gerais são usados para armazenar valores computationais e endereços. Os Registros de Controle e as Localizações de Controle armazenam todas as outras informações acessíveis privilegiadas do programa. Os Registros Gerais podem ser carregados do subcache ou armazenados no subcache. Há quatro conjuntos de registros relacionados às unidades de processamento funcionais: o CEU tem 32 registros de 40-bits; o IPU tem 32 registros de 64-bits; O FPU tem 64 registros de 64-bits; e o XIU tem 64 registros de 64 bits. No total há 192 registros. O grande número de registros, juntamente com instruções "load/store" rápidas e "pipelined" além da capacidade para iniciar duas instruções simultaneamente, permite ao KSR1 sobrepor computações com referências de memória. Por exemplo, uma instrução "Load" e uma outra instrução independente podem ser buscadas e serem completadas em paralelo. O esboço da CPU é apresentado na Figura 4.7.

Cada Elemento de Processamento do KSR-l é conectado a um subcache que é cache de primeiro nível de 0.5 Mb dividido em 0.25 Mb de subcache de instrução e 0.25 Mb de subcache.de dados. Durante cada ciclo, um par de instruções é buscado do subcache de instrução, e uma palavra de 64 bits pode ser carregada de ou pode ser armazenada no subcache de dados.

Cada subcache consiste em 128 blocos de 2048 bytes, cada bloco dividido em 32 sub-blocos de 64 bytes ("cache line"). Trata-se de uma implementação "two-way set associative cache" com 64 "sets".

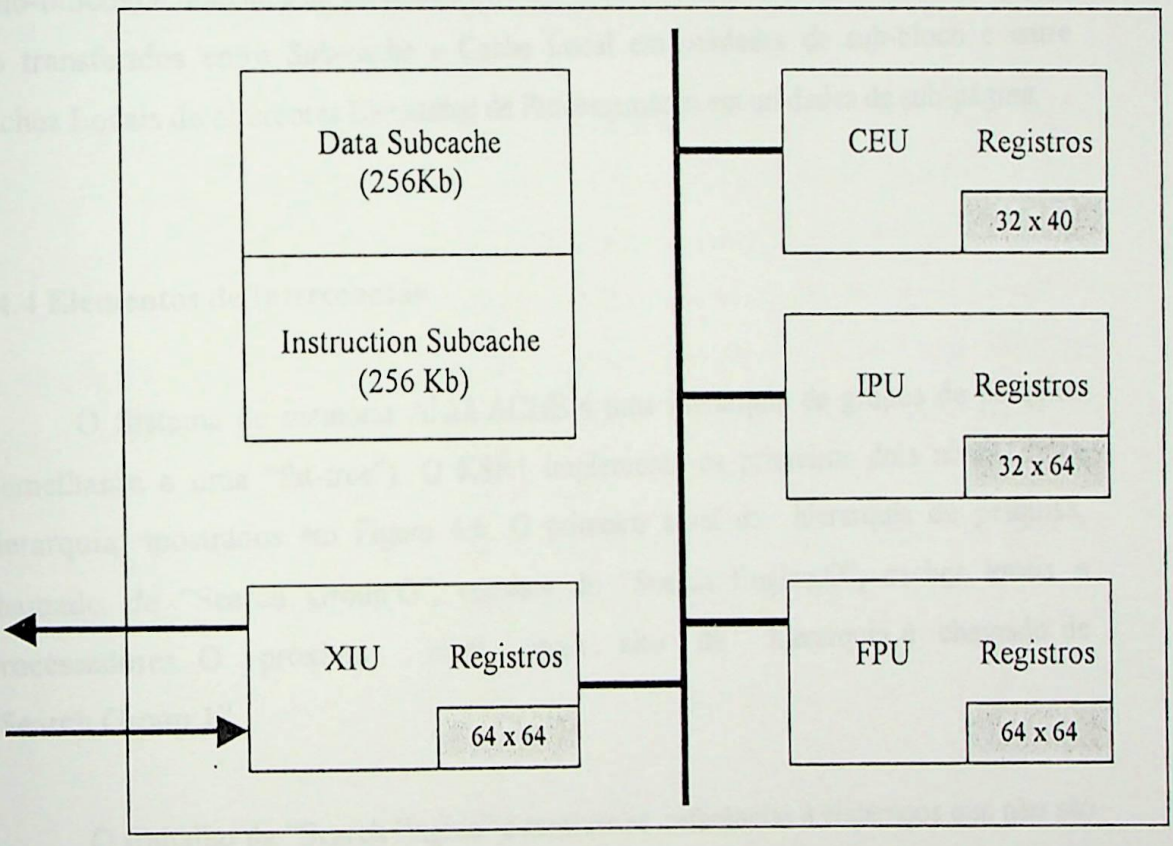


Fig. 4.7 - Organização de Elemento de Processamento do KSR1

4.4.3 Elementos de memória

Se o programa ou dados não cabem no subcache do elemento de processamento, a construção hierárquica do sistemas de memória ALLCACHE entra em ação. O nível de hierarquia acima do subcache é o elemento de memória chamado de Cache Local

com 32 Mb associado com cada Elemento de Processamento. O Cache Local consiste em 2048 páginas de 16 Kbytes. Cada página é dividida em 128 sub-páginas de 128 bytes ("cache line"). É uma implementação de "16-way set associative cache" com 128 conjuntos.

As latências são como segue: 2 ciclos de máquina para o sub-cache, 18 ciclos de máquina para o Cache Local e 175 ciclos de máquina para acessar uma célula remota no anel local [77].

É importante distinguir entre as unidades de gerenciamento do subcache (blocos e sub-blocos) e unidades de gerenciamento do Cache Local (páginas e subpages). Dados são transferidos entre Sub-cache e Cache Local em unidades de sub-bloco e entre Caches Locais de diferentes Elementos de Processamento em unidades de sub-página.

4.4.4 Elementos de Interconexão

O Sistema de memória ALLCACHE é uma hierarquia de grupos de pesquisa (semelhante a uma "fat-tree"). O KSR1 implementa os primeiros dois níveis desta hierarquia, mostrados em Figura 4.6. O primeiro nível da hierarquia de pesquisa, chamado de "Search Group:O", consiste do "Search Engine:O", caches locais e processadores. O próximo nível mais alto da hierarquia é chamado de "Search Group:1".

O trabalho da "Search Engine" é resolver as referências à endereços que não são achados no Cache Local do requisitante (ou é achado mas não com o estado da "subpage' adequado). O "Search Engine" tem que achar o endereço indicado no conjunto de Caches Locais e retorná-lo ao Cache Local do requesitante no estado apropriado. Também tem que deixar os estados da subpage dos outros Caches Locais consistente com o estado devolvido ao requesitante.

A "Search Engine:O" consiste de uma interconexão de roteamento, Diretórios de Cache Locais, e Caches Locais. Esta interconexão de roteamento do "Search Engine:O", chamado "Ring:O", é um meio de comunicação de pacotes, "pipelined" e

circular. Cada "slot" contém um pacote que consiste em um cabeçalho de 16-bytes (endereço e comando) e uma sub-página com 128 bytes de dados. O "Ring:O" é construído da parte da interconexão de roteamento presente em cada célula. A "Search Engine:1", chamado "Ring:1", opera de maneira semelhante ao "Ring:O".

Cada Diretório de Cache Local contém uma entrada para cada uma das 2048 páginas dentro daquele cache. A entrada especifica o SVA da página e o estado de todas as 128 sub-páginas. A célula que conecta o "Search:Engine:O" ao próximo nível mais alto da hierarquia, chamado de "ALLCACHE Router and Directory" (ARD), contém um diretório para toda "Search Engine:O"; há uma entrada no diretório de ARD para toda sub-página alocada em todo Cache Local no "Search Engine: O".

A comunicação entre elementos de processamento para acesso da memória remota é obtida por um processo de pesquisa. Quando o requesitante e aquele que deve responder ao pedido estão no mesmo "Ring:O", a pesquisa é restringida ao único "Ring:O" conectado. Diretórios de Caches locais mostram quais endereços podem ser achados no Cache Local conectado. Quando um Diretório de Cache Local determina que o endereço está presente em seu Cache Local - ele é chamado de Cache Local que responde - o pacote é extraído do anel. O Cache Local que responde re-insere o pacote com os dados dentro do anel, e o pacote continua a circular pelo "Ring:0" de volta ao Cache Local e ao elemento de processamento que requisitou. Veja Figura. 4.8.

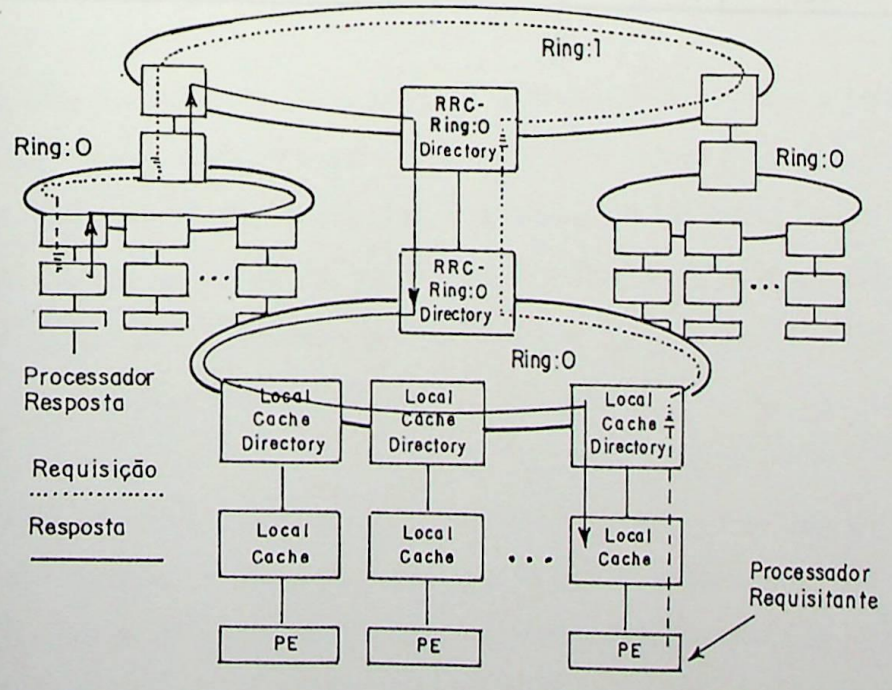


Fig. 4.8 - Processadores requisitante e que responde na mesma "Search Engine:0" [50]

4.5 Comentários

Foram descritos três sistemas de multiprocessadores implementados com modelos de arquiteturas diferentes. O Cray Y-MP, o Stanford DASH Multiprocessor e o Kendall Square Research KSRl foram caracterisados usando o modelo de Hardware Unificando desenvolvido no capítulo anterior. Um resumo de uma configuração típica de cada máquina, no estilo do modelo de hardware unificado, é apresentado na Tabela 4.1.

Tabela 4.1 - Resumo das configurações típicas

Caracteristícas	Unidade	Cray Y-MP8I/812	Stanford DASH	KSR1-32
P		8	64	32
M		128	16	32
IPCmax	Instruções/ciclo de máquina	1	1	2
DPC max	Operandos/ciclo de máquina	2	1	2
ICCmax	Instruções	512	16384	65536
DCCmax	Operandos	656	81968	32960
ICAmax	Instruções/ciclo de máquina	2	1	2
DCAmax	Operandos/ciclo de máquina	8	1	2
MECmax	Operandos	1 M	≈6.5M	4M
MBW	Operandos/ciclo de máquina	0.25	0.13	≈0.5
MLT	Ciclos de Máquina	3	30	18
DBW	Operandos/ciclo de máquina	2	0.03	6.7
DLT	Ciclos de Máquina	8	135	175

Capítulo 5

CONCLUSÕES

A avaliação de desempenho de computadores é necessária durante todo o ciclo de vida de um sistema computacional. Nosso trabalho procurou apresentar o estado da arte das diversas metodologias para realizar tal estudo, levando em conta vários fatores, dentre os quais podemos citar o grande desafio de acompanhar a evolução dos projetos de novos sistemas computacionais proporcionados pelo rápido avanço tecnológico. Este levantamento veio demonstrar a grande efervescência desta área, destacada pela elevada produção científica de universidades e indústrias com o desenvolvimento acelerado de novos equipamentos de alto desempenho visando os grandes desafios computacionais ("the computational grand challenges") em divesras áreas tais como Modelamento do Clima, Dispersão da Poluição, Genoma Humano, Modelamento de Semicondutores, Modelamento de Supercondutores, Visão e Sistemas Cognitivos, dentre outros.

Devido a grande diversidade de tópicos relacionados ao estudo de avaliação de desempenho, nosso trabalho teve como proposta focar uma classe específica de sistemas computacionais de alto desempenho: As máquinas paralelas. A complexidade desses sistemas não se manifesta somente através da complexidade do estudo da avaliação de desempenho mas também na complexidade de seus programas. Nossa proposição foi, então, buscar os conceitos fundamentais de desempenho de hardware e definir uma metodologia que levasse em conta a estrutura física do sistema ao invés do modelo de computação que a estrutura implementa. Esta interação de elementos de hardware e de software habilita-nos a estudar e compreender melhor os gargalos e a vida útil dos sistemas fornecendo subsídios para a validação do projeto e orientando os desenvolvimentos em outros níveis.

No Capítulo 2 procuramos relacionar os aspectos gerais mais importantes para o estudo de avaliação de desempenho que incluem a especificação das medidas desejadas, coleta de dados, análise dos dados e visualização. Procurou-se focar em

tópicos específicos relacionados à avaliação de desempenho de multiprocessadores. O que podemos depreender do material apresentado é que a definição e utilização de uma instrumentação adequada além de um conjunto de métricas de desempenho para a identificação e a caracterização de um dado sistema, passam a ter um papel fundamental no processo de mapeamento das aplicação para o hardware paralelo, permitindo o correto entendimento dos custos e benefícios de cada opção.

O Capítulo 3 e o Capítulo 4 foram devotados em examinar os pontos fundamentais do hardware de sistemas paralelos, a fim de definir uma classificação das máquinas paralelas baseadas na estrutura física do sistema. Além de revisar outros modelos de classificação utilizados até o momento, introduzimos um modelo que unifica o tratamento do hardware paralelo. Em termos de estrutura física, todo sistema pode ser visualizado com um sistema de transição de estado ("state-transition"). Através deste modelo torna-se fácil a compreensão dos fatores importantes que determinam o desempenho, abstraindo-se dos detalhes de uma implementação particular e concentrando atenção nas características essenciais.

Assim a classificação das máquinas paralelas ficou baseada, principalmente, se o sistema contém somente memória distribuída ou se há alguma quantidade de memória compartilhada globalmente endereçável. Sete classes foram identificadas.

Sob o ponto de vista da proposição de um modelo unificado, embora haja muitas diferenças entre as classes dos sistemas identificados, suas estruturas de hardware são semelhantes, em virtude de terem sido construídas utlizando componentes similares.

Também no Capítulo 3 procuramos definir estas estruturas fundamentais, com as quais os diversos sistemas são construídos, criando uma notação adequada para os componentes e suas interconexões. Três unidades básicas de hardware foram identificadas, definidas e caracterizadas através de parâmetros de hardware: Elementos de Processamento, Elementos de Memória e Interconexões. Além disso alguns parâmetros gerais foram definidos a fim de unificar o tratamento das diversas características do sistema: Ciclo de Máquina, Capacidade de Processamento e Tempo de Ciclo.

Três exemplos de estado da arte de sistemas de hardware paralelo foram descritos no Capítulo 4 em termos do Modelo proposto, a saber o Cray Y- MP, o Kendall Square Research KSR1 e o Stanford DASH. Estes sistemas representam, respectivamente, o modelo tradicional de Multiprocessadores com Memória Compartilhada e os modelos escaláveis de Multicomputadores com Memória Compartilhada Virtual e Memória Compartilhada Distribuída. Diversos aspectos relacionados a estes dois últimos modelos foram discutidos, tais como esquemas para cache coerente e Modelos de Consistência da Memória. Ao final apresentamos um resumo das características destas três configurações sob o ponto de vista do modelo unificado e de seus parâmetros característicos.

Futuros trabalhos envolveriam a caracterização de novos sistemas paralelos de tal forma a demonstrar a eficácia do modelo proposto. Além disso, este modelo lança as bases para a definição de um ambiente controlado sob o qual investigações sistemáticas sobre computação paralela poderiam ser conduzidas. Isto passa pela especificação de um conjunto de medidas de desempenho proximamente relacionadas com as características parametrizadas neste trabalho. A definição de métricas apropriadas para caracterização do desempenho total da implementação ao nível de hardware estabelece o benefício quantitativo da escolha de diferentes implementações em níveis superiores de abstração.

REFERÊNCIAS

[1] Ferrari,D., Serazzi,G., Zeigner,A, *"Measurement and Tunning of Computer Systems"*, Prentice Hall,Inc., Englewoods Cliffs, NJ,USA,1983.

[2] Heildelberger,P. ,Lavenberg,S., "Computer Performance Evaluation Methodology", *IEEE Transactions on Computer*, Vol. C-33, No. 12, December 1984.

[3] Michael Hord,R., *"Understanding Parallel Supercomputing"*, IEEE Press, 1999.

[4] Calzarossa, M., Serazzi,G., "Workload charcterization for Supercomputers", Special Topics in Supercomputing 4: Performance Evaluation of Supercomputers, J.L. Martin (ed), Elsevier Science Pub., North Holland, 1988

[5] Simmons,M., Koskela,R., Bucher,I., *"Instrumentation for Future Parallel Computing Systems"*, Addison-Wesley Pub. Company, Redwood City, CA, USA,1989.

[6] Kuck,D., Sameh,A H., "A supercomputing Performance Evaluation plan", Technical Report 692, Center for Supercomputing Research and Developement, University of Illinois, USA, 1987.

[7] Mallony,AD. Reed,D., "Visualizing parallel computer system performance", ACM Press Frontier Series: *Instrumentation for future parallel computing systems*, Eds. M. Simmons, R. Koskela, and I. Bucher, Addison-Wesley Pub. Company, Redwood City, CA,USA,1989.

[8] Snelling,D.F., "How long is a Megaflop?", Internal Report, Centre for Novel Computing, Department of Computer Science, University of Manchester.

[9] Hennessy,J.L., Patterson,D.A, *"Computer Architecture: A Quantitative Approach"*, Morgan and Kauffmann Pub., Inc., Palo Alto, CA, USA, 1990.

[10] Hockney,R., "A framework for Benchmark analysis", *Supercomputer*, March 1992, pp. 9-24.

[11] Hockney, R.W. and Jesshope,C.R., *"Parallel Computers 2: Architecture, programming and algorithms"*, IOP Pub.Ltd., Bristol, UK, 1988.

[12] Hockney, R., "Problem related performance parameters for supercomputers", *Special Topics in Supercomputing 4: Performance Evaluation of Supercomputers*, J.L. Martin (Ed.), Elsevier Science Pub. B.V., North Holland, 1988.

[13] Buzen,J.P., "Fundamental operational laws of computer system performance", *Acta Informatica*, Vol. 7, No.2, 1976.

[14] McMahon, F.H., "The Livermore Fortran Kernels: A computer test of numerical performance range", Lawrence Livermore National Lab., CA, USA, 1986.

[15] McMahon,F.H., "The Livermore Fortran Kernels test of numerical performance range", *Special Topics in Supercomputing 4: Performance Evaluation of Supercomputers*, J.L., Martin (ed), Elsevier Science Pub. B.V., North Holland, 1988.

[16] Dongarra,J.J., "Performance of various computers using Standard Linear Equations Sofwtare in a Fortran Environment", Tech. Rep. MCS-TM-23, Argonne National Lab., April 1987.

[17] Dongarra,J.J., "The Linpack Benchmark: An Explanation", *Evaluating Supercomputers*, Applied Information Tech. Reports, Aad van der Steen (Ed.), Chapman and Hall, London, 1990.

[18] Curnow,H., Wichman,B., "A Synthetic Benchmark", *Comput. J.*, Vol. 19, No.1, 1976, pp. 43-49.

[19] Weiker,R.P., "Dhrystone: A synthetic system programming benchmark", *Comm. ACM*, Vol 17, No. 10, October 1984, pp. 1013-1030.

[20] Uniejewski,J., "SPEC Benchmark Suite: Designed for today's advanced systems", *SPEC Newsletter*, Fall 1989, Vol. 1, Issue 1.

[21] Berry,M. et Al., "The Perfect Club Benchmarks: Effective Performance Evaluation of Supercomputers", CSRD Report No. 827, Center for Supercomputing Research and Developement, University of Illinois at Urbana-Champaign, May 1989.

[22] Cybenko, G., Kipp,L., Porter,L., Kuck, D., "Supercomputer Performance Evaluation and the Perfect Bechmarks", International Conference on Supercomputing, June 1990, ACM Press, pps. 254-266.

[23] Tang, H.W.H., "Monitoring tools for parallel systems,MSc Thesis, Department of Computer Science, University of Manchester, September 1987.

[24] Lucas Jr., H.C., "Performance Evaluation and Monitoring", *Computing Surveys*, Vol. 3, No. 3, September 1971.

[25] Smith,AJ., "Cache Memories", *ACM Computing Surveys*, Vol. 14, No.3 September 1982, pps. 473-530.

[26] Wang,W.-H., Loup Baer,J.-L, "Efficient Trace-Driven simulation methods for cache performance analysis", Conference on Measurement & Modelling of Computers Systems, Boulder, Colorado,USA, May 22-25, 1990, ACM Press Conf. Proceedings, pps. 27-36.

[27] Browne, J.C., Adiga, A.K., "Graph Structured Performance Models", *Special Topics in Supercomputing 4: Performance Evaluation of Supercomputers*, J.L. Martin (Ed.), Elsevier Science Pub. B.V., North Holland, 1988.

Escola Federal de Engenharia
Biblioteca
MAUÁ
Itajubá - EFEI

[28] Kraemer,E., Stasko,J.T., "The Visualization of Parallel Systems : An Overview", *Journal of Parallel and Distributed Computing*, Vol. 18, N. 2, June 1993, pps 105-117.

[29] Eggers,S.J., Keppel, D.R., Koldinger,E.J., Levy,H.M., "Techniques for Efficient Inline Tracing on a Shared Memory Multiprocessor", *Performance Evaluation Review*, 18, 1, May 1990, pps. 37-47.

[30] Miller,B.P., Yan,C.-Q, "IPS: An Interactive and Automatic Performance Measurement Tool for Parallel and Distributed Programs", *Proceedings of 7th International Conference on Distributed Computing Systems*, University of Wisconsin, September 1987, pps. 482-489.

[31] Mink,A, Carpenter,R., Nacht,G., Oberts,J., "Multiprocessor Performance Measurement Instrumentation", *IEEE Computer*, Vol. 23, No. 9, September 1990, pps. 63-75.

[32] Kimelman, D., Ngo,T., "Program Visualization for RP3: Am Overview", *Technical Report, IBM Research Division*, T.J. Watson Research Center, 1990.

[33] Nutt,G.J., "Tutorial: Computer System Monitors", *IEEE Computer*, Vol. 8, No. 11, Nov. 1975, pp. 51-61.

[34] So, K., Darema,F., George,D.A, Norton,V.A, Pfister,G.F., "PSIMUL- A system for parallel simulation of parallel systems", *Special Topics in Supercomputing 4: Performance Evaluation of Supercomputers*, J.L. Martin (Ed.), Elsevier Science Pub. B.V., North Holland, 1988.

[35] Pfister,G.P. et al., "The IBM Research Parallel Prototype (RP3): Introduction and Architecture", *Proceedings 1985 International Conf. On Parallel Processing*, pps. 764-771.

[36] Heath, M.T., Etheridge,J.A., "Visualization the Performance of Parallel Programs", IEEE Software, Vol. 8, No. 5,September 1991, pps. 29-39.

[37] Zhang, X, Nalluri, N.S., Qin,X., "MIN-Graph : A Tool for Monitoring and Visualizing MIN-Based Multiprocessor Performance", *Journal of Parallel and Distributed Computing*, Vol.18, No. 2, June 1993, pps. 231-241.

[38] Kohn,J., Williems, W., "ATExpert", *Journal of Parallel and Distributed Computing*, Vol.18, No. 2, June 1993, pps. 205-222.

[39] Kitchin,P., O'Neill,G., "Performance Evaluation on a hierarchical shared memory machine", *Evaluating Supercomputers*, Applied Information Technology Reports, Aad van der Steen (Ed.), Chapman and Hall, London, 1990.

[40] Carpenter,R.J., "Performance Measurement Instrumentation at NBS", *ACM Press Frontier Series: Instrumentation for future parallel computing systems*, Eds. M. Simmons, R. Koskela, and I. Bucher, Addison-Wesley Pub.Company, Redwood City,CA,USA,1989.

[41] Mallony, AD., Reed,D.A, "A hardware-based performance monitor for the Intel iPSC/2 Hypercube", *International Conference on Supercomputing*, June 1990, ACM Press Conf. Proceedings.

[42] Steven, F., Burckhardt III, H., Rothnie, J., "KSR1: Bridging the gap between shared memory and MPPs", *Proceedings of Compcon'93*, San Francisco, CA, Feb 1993, pps 285-294.

[43] Lenoski D., Gharachorloo, J. L., Gupta. A., Hennessy, J.. Horowitz, M.. I am M., "Design of Scalable Shared-Memory Multiprocessors: The DASH Approach" *Proceedings of COMPCON'90 IEEE* , Computer Society Press, February 1990, pps. 62-67.

[44] Lenoski, D. et Al, "The DASH Prototype: Logic Overhead and Performance", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 1, January 1993, pps. 41-61.

[45] Simmons, M.L. et all, "A Performance Comparison of Four Supercomputers", *Communications of the ACM*, Vol. 35, No. 8, August 1992, pps. 116-124.

[46] Nelson,H., "Experience with Performance Monitors", *ACM Press Frontier Series: Instrumentation for future parallel computing systems*, Eds. M. Simmons, R. Koskela, and I. Bucher, Addison-Wesley Pub.Company, Redwood City,CA,USA,1989.

[47] Flynn, M. J., "Some Computer Organizations and their effectiveness". *IEEE Trans. on Computers*, C-21, No. 9, Sept 1972, pp. 948-960.

[48] Siewiorek, D. P., Bell, C. G., Newell, A., *"Computer Structures: Principles and examples*", McGraw-Hill Book Company, USA, 1982.

[49] Gurd, J. R., "Lecture notes of CS503". Centre for Novel Computing, Dept. of Computer Science, University of Manchester, 1992.

[50] Hwang. K., "*Advanced Computer Architecture: Parallelism, Scalability, Programmability*", McGraw-Hill. Inc., USA, 1993.

[51] Gurd J. R., Cooper, M. D., Hedayat, G., Nisbet, A., O'Boyle, M. F. P., Snelling, D. F., Bohm, A. P. W., "A Framework for Experimental Analysis of Parallel Computing", Centre for Novel Computing Department of Computer Science, University of Manchester. Nov. 1992.

[52] Snelling, D., Cooper, M. D., Nisbet, A., "Experimental guidelines for a Shallow mapping study", TR CNC/NC/TR11991/043.001, Centre for Novel Computing, Department of Computer Science. University of Manchester, Dec. 1991.

[53] Dasgupta,S., " A hierarchical Taxonomic System for Computer Architectures" Computer March 1990.

[54] Flynn M. J., "Very high-speed computing systems", *Proc. IEEE*, Vol. 54, 1966, pp. 1901 1909.

[55] Skillicorn. D. B.. "A taxonomy for computer architectures", *Computer*, Vol. 21, No. 11, November 1988.

[56] Duncan, R., "A survey of parallel computer architectures". *IEEE Computer*, February 1990, Vol. 23, No. 2, pps. 5-16.

[57] Bell, G.. "Ultracomputers: a teraflop before its time", *Communications of the ACM*, August 1992. Vol. 35,No. 8, pps. 27-47.

[58] Hargersten, E., Landin, A., Haridi, S., "DDM - A Cache-Only Memory Architecture". *IEEE Computer*, September 1992, pps. 44-54.

[59] Stenstrom,P., Joe,T., Gupta, A.,"Comparative Performance Evaluation of Cache Coherent NUMA and COMA architectures", Proc.of the 19th Annual International Symposium on Computer Architecture, May 1992, pps. 80-91.

[60] Singh., J.P., Joe, T.,Hennessy, J.L., Gupta, A.. " An Empirical Comparison of the Kendall Square Research KSR- 1 and Stanford DASH Multiprocessors", Center Integrated Systems, Computer Systems Laboratory, Stanford University, CA, USA, 1993.

[61] Smith. A.J., "Cache Memories"? ACM Computing Survey, ps. 473-500, Sept. 1982.

[62] Smith-A.J.,"Second Bibliography on Cache Memories", *Computer Architecture News*. Vol. 19, No.4, June 1991, pps. 154- 182.

[63] Ramamoorthy,G.- Choudhary,A.N.,"A bibliography for Multiprocessor C ache Memories" *Computer Architecture News*, Vol. 19, No.4, June 1991, pp. 138-153.

[64] Moreira,J.E., "Cache Systems for Multiprocessors", Jornada EPUSP/IEEE em Sistemas de Computação de Alto Desempenho",25-28 March 1991, EPUSP São Paulo, Brazil.

[65] Przybylski, S., Horowitz, M., Hennessy, J., "Performance tradeoffs in cache design", Proc. 1st Annual International Symposium Computer Architecture, 1988. pp. 290-298.

[66] Stenstrom, P., "A survey of Cache Coherence for Multiprocessors", *IEEE Computer*, June 1990, pps. 12-24.

[67] Agarwal, A., Simoni, R., Hennessy, J., Horowitz, M., "An Evaluation of Directory Schemes for Cache Coherence", Proc. 15th Annual International Symposium Computer Architecture, 1988, pp. 280-289.

[68] Tang. C., K., "Cache design in the tightly Coupled Multiprocessor System" AFPS Conf. Proceedings. National Computer Conf., NY, NY, pp. 479-753, June 1976.

[69] Censier. L., M., Feautrier, P., "A new solution to coherence problems in multicache systems", *IEEE Trans. on Computers*, C-27(12):1112-1118, December 1978.

[70] James, D.,V., Laundrie, A.,T., Gjessing, S., Shi, G.,S.. "Distributed-Directory Scheme: Scalable Coherent Interface", *IEEE Computer*, June 1990, pp. 74-77.

[71] Lamport, L., "How to make a multiprocessor computer that correctly executes multiprocess programs", *IEEE Trans. on Computers*, Vol. 28, No. 9, 1979, pp. 241-248.

[72] Gharachorloo, K., Lenoski, D., Laudon, J., Gibbons, P., Gupta, A.. Hennessy. J.-" Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors". Proc.of the 17th A~mual International Symposium on Computer Architecture, June 1990. pps. 15-26.

[73] Gharachorloo, G., A., Hennessy, J.,"Performance Evaluation of Memory Consistency Models for Shared-Memory Multiprocessors", Proc. of the 4th international Conference on Architectural Support for Programming Languages

and Operating Systems", April 1991, pps. 245-257.

[74] Dubois, M., Scheurich, C., Briggs, F., "Memory access buffering in multiprocessors", Proc. of the 13th Annual International Symposium on Computer Architecture", June 1986, pps. 434-442.

[75] Bell, G., "Ultracomputers: a teraflop before its time"? *Communications of the ACM*, August 1992, Vol. 35,No. 8, pps. 27-47.

[76] Frank, S., "Virtual Memory to Allcache Memory", Virtual Shared Memory Symposium, Centre for Novel Computing, University of Manchester, September 1992.

[77] Kendall Square Research Corp., *"KSR Parallel Programming"*, October 1991, Waltham, MA, USA.

[78] Hwang, K., Briggs, F. A., *"Computer architecture and Parallel Processing"*, McGraw-Hill International Editions, 1984.1.