Arquitetura de Computadores Paralelos

**Matheus Barbosa Souza**

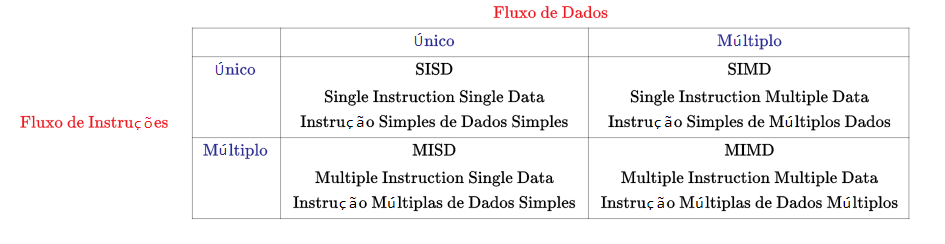
Curso de Ciência da Computação – Universidade Presidente Antônio Carlos (UNIPAC) – Campus de Barbacena

[mbscbsjbs@gmail.com](mailto:mbscbsjbs@gmail.com)

**Resumo.** O grande interesse por problemas cada vez mais complexos tem levado à necessidade de computadores cada vez mais potentes para resolvê-los. Entretanto, limitações físicas e econômicas têm restringido o aumento da velocidade dos computadores sequenciais, ou seja, computadores que executam instruções em série, uma após a outra pela CPU. Por outro lado, os problemas computacionais usualmente podem ter algumas de suas partes divididas em pedaços que teriam como ser solucionados ao mesmo tempo ou processadas em paralelo. Processamento paralelo é então uma forma pela qual a demanda computacional é suprida por meio do uso simultâneo de recursos computacionais como processadores para solução de um problema. A computação paralela é caracterizada pelo uso de várias unidades de processamento ou processadores para executar uma computação de forma mais rápida. É baseada no fato de que o processo de resolução de um problema pode ser dividido em tarefas menores, realizadas simultaneamente por meio de algum tipo de coordenação. O conceito foi originalmente introduzido no CDC 6600 em 1964 pela CDC (control data corporation).

**1. Taxonomia de Flynn**

A taxonomia de Flynn baseia-se no fato de um computador executar uma sequência de instruções de dados, diferencia-se o fluxo de instruções e o fluxo de dados.



**Figura 1. Classificação de Flynn**

**2. Classes de Arquiteturas**

**2.1 SISD (Single Instruction Single Data): fluxo único de instruções sobre um único conjunto de dados.**

Nesta classe, um único fluxo de instruções opera sobre um único fluxo de dados. Isso corresponde ao processamento sequencial característico da máquina de Von Neumann e que compreende os computadores pessoais e estações de trabalho. Apesar dos programas estarem organizados por meio de instruções sequenciais, elas podem ser executadas de forma sobreposta em diferentes estágios (pipelining). Arquiteturas SISD caracterizam-se por possuírem uma única unidade de controle, podendo possuir mais de uma unidade funcional. Um exemplo seria seu computador pessoal com um processador convencional.

**2.2 SIMD (Single Instruction Multiple Data): fluxo único de instruções em múltiplos conjuntos de dados.**

Esta classificação corresponde ao processamento de vários dados sob o comando de apenas uma instrução. Em uma arquitetura SIMD, o programa ainda segue uma organização sequencial. Para possibilitar o acesso a múltiplos dados é preciso uma organização de memória em diversos módulos. A unidade de controle é única, e existem diversas unidades funcionais. Nesta classe estão os processadores vetoriais e matriciais.

Quanto às facilidades de hardware para armazenamento, essas normalmente são classificadas como:

● Processor Array

● Vector Pipeline

Exemplo de computadores com a arquitetura processor array são as máquinas ILLIAC IV (Universidade de Illinois), Thinking Machine CM- 2 e MASPAR MP-1216.

Exemplo de computadores com a arquitetura vector pipeline são as máquinas IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10. As GPUs, também, estão sob essa classificação.

**2.3 MISD (Multiple Instruction Single Data): fluxo múltiplo de instruções em um único conjunto de dados.**

Um conjunto de dados é colocado concorrente em múltiplas unidades de processamento. Cada UP opera de maneira independente via conjuntos independentes de instruções. Algumas utilizações de uma configuração MISD poderiam ser:

● Filtros de múltiplas frequências operando um mesmo sinal

● Múltiplos algoritmos de criptografia tentando a quebra de uma mensagem codificada

Não se tem conhecimento de arquitetura de máquinas comercial com múltiplas instruções trabalhando com um único conjunto de dados concorrente. Em 1971, uma máquina denominada como C.mmp computer foi desenvolvida na Universidade de Carnegie-Mellon.

**2.4 MIMD (Multiple Instruction Multiple Data): fluxo múltiplo de instruções sobre múltiplos conjuntos de dados.**

Esta classe é bastante genérica, envolvendo o processamento de múltiplos dados por parte de múltiplas instruções. Neste caso, várias unidades de controle comandam suas unidades funcionais, as quais têm acesso a vários módulos de memória. Qualquer grupo de máquinas operando como uma unidade (deve haver certo grau de interação entre as máquinas) enquadra-se como MIMD. Alguns representantes desta categoria são os servidores multiprocessados, as redes de estações e as arquiteturas massivamente paralelas.

**3. Paralelismo**

**3.1 Paralelismo Implícito**

Faz uso de compiladores que detectam o paralelismo existente em um código seqüencial gerando código paralelo automaticamente. Fica deste modo oculto ao programador o acréscimo de complexidade introduzido pelo paralelismo, mantendo-se a sintaxe da codificação seqüencial, ganhando, porém grande complexidade a tarefa de elaboração do compilador paralelizador.

**3.2 Paralelismo Explícito**

ocorre quando o paralelismo fica a cargo do programador, que sabe construir programas paralelos e faz uso de linguagens e ferramentas de programação que lhe oferecem suporte. Nessa proposta o programador é responsável por especificar o que pode/deve ser executado em paralelo, exigindo que, além de dominar o algoritmo, o programador conheça as características operacionais da arquitetura paralela. Além disso, nesse caso, o projeto do compilador paralelizador tem sua complexidade reduzida, porém ainda engloba aspectos mais sofisticados que o projeto de compiladores para códigos sequenciais.

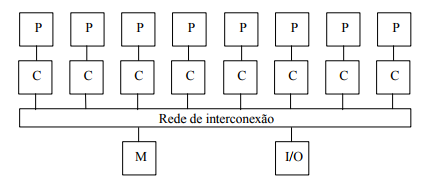
**4. Multiprocessadores**

**4.1 Multiprocessadores UMA (Uniform Memory Access – Acesso à Memória Uniforme).**

Cada CPU tem o mesmo tempo de acesso a todos os módulos de memória.

– Se for tecnicamente impossível, a velocidade das palavras de memória mais rápidas são reduzidas para o valor das mais lentas – Uniforme.

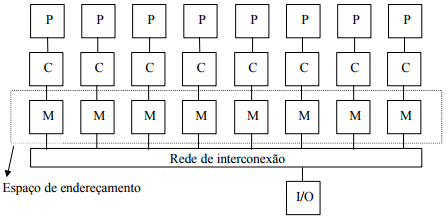
– Uniformidade torna o desempenho previsível.



**Figura 2. Arquitetura de um multiprocessador do tipo SMP (UMA)**

**4.2 Multiprocessadores NUMA (NonUniform Memory Access – Acesso Não-Uniforme à Memória).**

A cada processador é associado um banco de memória. Nesse caso, o sistema operacional trata cada banco separadamente, pois cada banco tem um custo de acesso diferente, dependendo de qual o processador a que está associado e onde está sendo executado o processo que tenta acessar a memória.



**Figura 3. Arquitetura de um multiprocessador do tipo NUMA**

**4.3 Multiprocessadores COMA (Cache Only Memory Access – Acesso Somente à Memória Cache).**

São multiprocessadores baseados em memórias cache de alta capacidade, em que a coerência é conseguida em hardware com a atualização simultânea em múltiplos nós dos dados alterados. Esse tipo de arquitetura é bastante complexo e faz com que estas máquinas tenham um custo elevado.

**5. Multithreading no chip**

Permite que a CPU gerencie múltiplos threads de controle ao mesmo tempo na tentativa de mascarar as protelações (retardações) que ocorrem quando em uma referencia à memória não encontra nada nos cachês L1 e L2 e o processador fica esperando. Em suma se o thread um tiver bloqueado, a CPU pode executar o thread dois, de modo a manter o hardware sempre ocupado. O Multithreading no Chip pode ocorrer de três formas: Multithreading de granulação fina: esse mascara as protelações executando os threads segundo uma política de alternância circular, com thread diferente em ciclos consecutivos. Cada estágio do pipeline executa um thread. Multithreading de granulação grossa: Pipeline muda de thread a cada stall e Multithreading simultâneo: Cada stall pode se identificado isoladamente. Um exemplo prático de Multithreading é o Processador Pentium IV 3.06 GHz – Seu valor está em fazer com que um segundo thread utilize hardware, que não fosse por isso, ficaria inutilizado. Foram aumentados 5% da área do Chip e ganhou-se 25% em desempenho para muitas aplicações. A idéia principal é permitir que dois threads ou processos executem ao mesmo tempo.

**6. Hiper-Threading**

Usa os recursos do processador com mais eficiência, permitindo que múltiplos threads sejam executados em cada núcleo. Por ser um recurso de desempenho, também aumenta a produtividade do processador, melhorando o desempenho geral do software encadeado.

O Hyper Threading é uma tecnologia, portanto, que permite que os processadores multicore (dois, quatro, seis, oito e, atualmente, já em 10 e 12 núcleos) tenham acesso a um recurso de agendamento de tarefas muito mais esperto, no sentido de que é capaz de encaminhar as threads aos núcleos da CPU de forma muito mais ágil em busca da diminuição desses períodos de ociosidade.

Processadores com Hyper Threading disponível atingem performance muito melhor em aplicações que dependem de alto grau de paralelismo de processamento de dados (ou, como vimos, que são capazes de passar um conjunto de instruções à CPU que podem ser divididas e processadas simultaneamente pelos núcleos). Exemplos de aplicações desse tipo são editores de vídeo, jogos, softwares de modelagem, simuladores e etc.

**7. Multicomputadores**

**7.1 MPPs**

Máquinas maciçamente paralelas (MPP - Massively Parallel Processors) são

multicomputadores construídos com milhares de processadores comerciais (P/C)

conectados por uma rede proprietária de alta velocidade. A expressão

“maciçamente paralela” indica a proposta dessas máquinas: a obtenção de alto

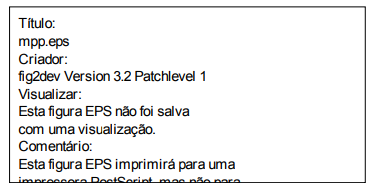
desempenho através da utilização de um grande número de processadores, os quais,

devido ao fator custo, acabam sendo processadores de poder computacional médio ou

pequeno. Essa é uma alternativa à proposta dos PVPs, por exemplo, em que o alto

desempenho é obtido através da utilização de poucos processadores vetoriais de grande

poder computacional.



**Figura 4. Arquitetura de um MPP (Massively Parallel Processor)**

**7.2 Cluster**

Um cluster, ou aglomerado de computadores, é formado por um conjunto de computadores que se utiliza de um tipo especial de sistema operacional classificado como sistema distribuído. Muitas vezes, é construído a partir de computadores convencionais (personal computers), os quais são ligados em rede e comunicam-se por meio do sistema, trabalhando como se fossem uma única máquina de grande porte. Há diversos tipos de cluster. Um tipo famoso é o cluster da classe Beowulf, constituído por diversos nós escravos gerenciados por um só computador. Ele pode ser definido também como o cruzamento de uma trilha com um setor de um disco formatado. Um HD (hard disc) possui vários clusters que serão utilizados para armazenamento dos dados de um determinado arquivo. Com essa divisão em trilhas e setores, é possível criar um endereçamento que visa facilitar o acesso a dados não contíguos, assim como o endereçamento de uma planilha de cálculos ou, como um exemplo mais simples, o tabuleiro do jogo "batalha naval".

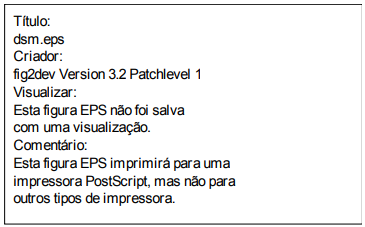
**8. Computação em Grade**

A computação em grade é um grupo de computadores em rede que trabalham em conjunto, como um super computador virtual, para executar tarefas grandes, como analisar grandes conjuntos de dados e modelagem do clima. Por meio da nuvem, é possível montar e usar vastas grades de computadores por períodos de tempo e fins específicos, pagando somente pelo seu uso e economizando tempo e gastos com a aquisição e implantação de recursos necessários. Ao dividir as tarefas entre várias máquinas, o tempo de processamento é reduzido de forma significativa, aumentando a eficiência e diminuindo os recursos desperdiçados.

Ao contrário da computação paralela, os projetos de computação em grade não apresentam nenhuma dependência de tempo associada a eles. Eles utilizam computadores que fazem parte da rede apenas enquanto estiverem ociosos e os operadores podem realizar tarefas não relacionadas à grade a qualquer momento. A segurança deve ser levada em conta ao utilizar grades de computadores, visto que os controles sobre os nós de membros costumam ser dispersos. A redundância também deve ser incorporada, pois muitos computadores podem ser desconectados ou falhar durante o processamento.

**9. Memória Compartilhada Distribuída**

Máquinas com memória compartilhada distribuída (DSM – Distributed Shared Memory) são sistemas em que, apesar de a memória se encontrar-se fisicamente distribuída através dos nós, todos os processadores podem endereçar todas as memórias. Isso se deve à implementação de um único espaço de endereçamento. Essa implementação pode ser feita em hardware, em software ou ainda com a combinação dos dois. A distribuição da memória, por sua vez, pode ser resultado da escolha de uma arquitetura multiprocessada com memória entrelaçada distribuída (máquinas NUMA) ou de uma arquitetura de multicomputador com memórias locais (máquina NORMA). Em ambos os casos, a máquina resultante é considerada CC-NUMA se tiver coerência de cache implementada em hardware ou SC-NUMA se a implementação for em software.

****

**Figura 5. Arquitetura de uma máquina DSM (Distributed Shared Memory)**

**10. Abordagem**

**10.1 DSM Linda**

Sistemas DSM baseados em páginas como IVY e o Treadmarks usam o hardware MMU para causar exceções de acesso as páginas faltantes. Uma delas é a Linda, que fornece processos em varias maquinas com uma memoria compartilhada distribuída com alto grau de estruturação. Essa memoria é acessada por meio de um pequeno conjunto de operações primitivas que podem ser adicionadas a linguagens existentes, tal como C e FORTRAN, para formar linguagens paralelas nesse caso, C-linda e FORTRAN-Linda. O conceito unificador fundamental de linda é um espaço abstrato de tuplas, que é global para o sistema inteiro e acessível a todos os seus processos.

**10.2 Memória Compartilhada ORCA**

Uma abordagem um pouco diferente para a memória compartilhada no nível de aplicação em um multicomputador é usar como unidade de compartilhamento objetos totalmente desenvolvidos, em vez de apenas tuplas. O sistema baseado em objetos que da a ilusão de memoria compartilhada é denominado Orca. Orca é uma linguagem de programação tradicional, baseada em Modula 2, a qual foram adicionadas duas novas característica: objetos e a habilidade de criar novos processos. Um objeto Orca é um tipo de dados abstrato, análogo a um objeto em java ou a um pacote em Ada. Ele encapsula estruturas de dados internas e métodos escritos pelo usuário denominados operações.

**10.3 DSM Globe**

A maioria dos sistemas DSM, Linda e Orca executam em sistemas locais. Todavia, também é possível construir um sistema de memória compartilhada no nível de aplicação em um multicomputador que executa em âmbito mundial. No sistema Globe , um objeto pode estar localizado no espaço de endereço de vários processos ao mesmo tempo, possivelmente em diferentes continentes. Para acessar os dados de um objeto compartilhado, processos usuários devem passar por seus métodos, o que permite que diferentes objetos tenham diferentes estratégias de implementação. O que torna Globe um tanto ambicioso é sua meta de ser ampliado para um bilhão de usuários e um trilhão de objetos (possivelmente móveis). Localizar objetos, gerenciá-los e administrar a ampliação é crucial. O Globe faz isso com uma estrutura geral na qual, mesmo assim, cada objeto pode ter sua própria estratégia de replicação, estratégia de segurança e assim por diante. Isso evita o problema do tamanho único presente em outros sistemas e ao mesmo tempo, conserva a facilidade de programação oferecida pela memória compartilhada.

**11. MPI (Message-Passing Interface)**

Message Passing Interface (MPI) é um padrão para comunicação de dados em computação paralela. Existem várias modalidades de computação paralela, e dependendo do problema que se está tentando resolver, pode ser necessário passar informações entre os vários processadores ou nodos de um cluster, e o MPI oferece uma infraestrutura para essa tarefa. No padrão MPI, uma aplicação é constituída por um ou mais processos que se comunicam, acionando-se funções para o envio e recebimento de mensagens entre os processos. Inicialmente, na maioria das implementações, um conjunto fixo de processos é criado. Porém, esses processos podem executar diferentes programas. Por isso, o padrão MPI é algumas vezes referido como MPMD (multiple program multiple data). Elementos importantes em implementações paralelas são a comunicação de dados entre processos paralelos e o balanceamento da carga. Dado o fato do número de processos no MPI ser normalmente fixo, neste texto é enfocado o mecanismo usado para comunicação de dados entre processos. Os processos podem usar mecanismos de comunicação ponto a ponto (operações para enviar mensagens de um determinado processo a outro). Um grupo de processos pode invocar operações coletivas (collective) de comunicação para executar operações globais. O MPI é capaz de suportar comunicação assíncrona e programação modular, através de mecanismos de comunicadores (communicator) que permitem ao usuário MPI definir módulos que encapsulem estruturas de comunicação interna.

**Referências:**

Tanenbaum, A. S. Organização Estruturada de Computadores. Rio de Janeiro, RJ: Livros Técnicos e Científicos Editora, 2001. OCLC 709563955.

Acesso: Novembro/2017

Carlos Eduardo Medeiros de Carvalho, “O Processamento Paralelo e a Taxonomia de Flynn”

Março/2017

Web: https://pt.linkedin.com/pulse/o-processamento-paralelo-e-taxonomia-de-flynn-carlos-eduardo.

Acesso: Novembro/2017