Modelos de Computação Paralela

Lucas Cyulik[1]

[1] Departamento de Informática – Universidade Tecnológica Federal do Paraná (UTFPR)
CEP 80.230-901 – Curitiba – PR – Brasil

lcyulik@alunos.utfpr.edu.br

Resumo. *O presente artigo tem como objetivo apresentar um estudo a respeito dos principais tópicos de Computação Paralela. Neste artigo são abordados o modelo de Von Neumann, a definição de paralelismo, assim como os níveis de paralelismo. Além disso, são descritas as arquiteturas e os modelos de execução Parallel Random Access Machine (PRAM), Bulk Synchronous Parallel Model (BSP), Coarse Grained Multicomputer (CGM) e LogP de computação paralela.*
Palavras-Chave: *Paralelismo, Von Neumann, Computação Distribuída, Modelos, Níveis de Paralelismo, PRAM, BSP, CGM, LogP.*

1. Introdução à Computação Paralela

Computação paralela é um assunto muito recorrente nos estudos de Sistema Distribuídos, pois opera sob o princípio de que grande problemas podem ser divididos em problemas menores, que então podem ser resolvidos concorrentemente em paralelo. Este artigo abordara de forma breve os diferentes modelos e arquiteturas de computação paralela, e assim relacionando com a importância de sistemas distribuídos.

Porem antes de avançar na definição de computação paralela, precisa-se discutir como surgiram as bases nas quais os modelos atuais se apoiam. Na literatura, um modelo muito aceito é o de Von Neumann, que estabeleceu os conceitos básicos de organização dos computadores eletrônicos. O modelo de von Neumann é formado basicamente pelo conjunto: processador, memória e dispositivos de E/S [Backus 1978]. O processador executa instruções sequencialmente de acordo com a ordem ditada por uma unidade de controle [Goldman 2003]. Segundo Goldmann (2003), as operações realizadas pelo modelo de Von Neumann são:

1. Busca e decodificação das instruções;
2. Cálculo dos endereços dos operandos;
3. Busca dos operandos na memória;
4. Cálculo com os operandos;
5. Armazenamento dos resultados na memória.

Nem todas as operações realizadas por um computador descrito por esse modelo executam todos os passos acima listados. Esse modelo simplifica bem o funcionamento de um computador, tornando-o um modelo universal [Goldman 2003].

O processamento paralelo interliga vários nós de processamento de maneira que um processo de grande consumo seja subdividido por vários nós, que por sua vez deverão cooperar entre si, buscando eficiência, através da quebra do paradigma de execução sequencial do fluxo de instruções ditado por Von Neumann [Cap and Strumpen 1993].

Tal abordagem permite a redução de custos e a melhoria de desempenho dos processos computacionais [Cap and Strumpen 1993].

O conceito de executar tarefas paralelamente é simplesmente a distribuição de subtarefas entre processadores, tendo assim um trabalho conjunto, no qual cada processador resolve um pequeno problema por vez [Cap and Strumpen 1993]. São várias as medidas de desempenho usadas na resolução de um problema em computação paralela. As mais comuns são [Karp and Flatt 1990]:

- Speed-up: também conhecido como Fator de Aceleração, é definido para um sistema paralelo homogêneo como a razão entre a velocidade de processamento do sistema com um processador e do sistema paralelo com *m* processadores;
- Eficiência: Mede a fração de tempo que um processador é, de fato, utilizado. É definida como o *speed-up* dividido pelo número de processadores;
- Custo: é a soma do tempo que cada processador utilizou. É o equivalente a um processador sequencial hipotético com o desempenho equivalente a soma dos processadores em paralelo;
- Taxa de ocupação: Inclui todo o tempo gasto por um processador para executar o algoritmo, levando em consideração não só o processamento, mas também a comunicação. É calculado individualmente para cada processador.

A computação paralela permitiu um aumento do desempenho na execução de tarefas, mas além disso, se consolidou também pelas restrições tecnológicas, pois problemas complexos, que exigiam processamento de muitos dados em pouco tempo serviram de motivadores para evolução das técnicas de processamento paralelo [Cap and Strumpen 1993].

São vários os problemas que surgem quando se trata de executar tarefas em paralelo, os principais se devem a comunicação entre os diversos processadores, que gera complicações de sincronismo entre as tarefas, de separação das tarefas, memória disponível e concorrência[Karp and Flatt 1990].

1.1. Níveis de Paralelismo

Existem diferentes níveis no qual pode ocorrer paralelismo, como:

- Nível de instruções: uso de *pipeline*, técnica que tira proveito do fato de que uma instrução de máquina pode ser dividida em uma sequência de etapas intermediárias. Como cada instrução está realizando apenas uma dessas etapas em um instante de tempo, mais de uma instrução pode ser executada em um mesmo instante de tempo desde que elas não estejam na mesma etapa.
- Nível de processos: consiste na em multiprocessadores e/ou multi-computadores, com organização de uma ou mais centrais de processamento para execução de uma tarefa.
- Nível de tarefas: uso de *Threads*, onde o software utilizado é que o responsável pela execução das tarefas em paralelo. E pode ser utilizado em multicomputadores ou outras arquiteturas.

2. Arquitetura de Máquinas para Computação Paralela

São dois os critérios principais seguidos para classificação de máquinas paralelas: o fluxo de instruções e fluxo de dados. Segundo Goldman (2003) são quatro as categorias que se destacam:

- SISD (*Single Instruction Single Data*): representa uma máquina sequencial, com o fluxo de dados e de instruções únicos;
- MISD (*Multiple Instructions Single Data*): múltiplos fluxos de instruções e um único de dados;
- SIMD (*Single Instruction Multiple Data*): vários fluxos de dados com um único fluxo de instruções;
- MIMD (*Multiple Instruction Multiple Data*): vários fluxos de instruções e de dados.

A classificação apresentada acima representa a maior parte das arquiteturas de computador usadas até hoje. No caso de flexibilidade na execução de algoritmos paralelos, as arquiteturas MIMD se destacam por apresentarem bom desempenho e pelo fato de o processamento ser assíncrono. As arquiteturas que seguem a categoria SIMD são mais simples e facilitam a programação de programas paralelos.

3. Modelos de Computação Paralela

São diversos os modelos de arquitetura existentes para computação paralela, o que dificulta a concepção de modelos que sejam simples o bastante para permitir um fácil entendimento e uso [Goldman 2003]. A comparação comumente feita na literatura é em relação ao modelo de Von Neumann e o que ele foi para a computação sequencial.

3.1. *Parallel Random Access Machine* (PRAM)

O modelo de computação paralela mais conhecido é PRAM (Parallel Random Access Machine) que consiste de um conjunto de processadores idênticos que executam suas tarefas de maneira síncrona. Existem duas características marcantes nesse modelo:

1. Memória compartilhada;
2. Vários processadores trabalhando sincronicamente.

O fato de o modelo utilizar diversos processadores e apenas uma memória, evidência um problema: concorrência para acesso a determinada posição da memória [Menezes and Setubal 1995]. Por se tratar de um modelo simples e fácil quando comparado aos outros, ele é muito usado no cenário teórico, sendo referência para comparação de algoritmos paralelos.

O modelo PRAM pode ainda ser divido em 4 submodelos, que se diferenciam na maneira como lidam com o acesso a memória. Os submodelos são:

- Leitura Exclusiva, Escrita Exclusiva: leitura e escrita não concorrentes;
- Leitura Concorrente, Escrita Exclusiva: apenas um processador pode escrever por vez na memória;
- Leitura Exclusiva, Escrita Concorrente: apenas um processador pode fazer leitura por vez na memória;
- Leitura Concorrente, Escrita Concorrente: ambas as operações serem realizadas de maneira concorrente pelos processador.

As principais vantagens para utilização do modelo PRAM são: a fácil análise de complexidade e a fácil implementação do modelo possibilita um foco maior na estruturação do problema, não precisando se ater tanto ao modelo [Menezes and Setubal 1995].

3.2. *Bulk Synchronous Parallel Model* (BSP)

O modelo BSP (Bulk Synchronous Parallel Model) foi concebido com o pretexto de representar uma padronização para a computação paralela, da mesma forma que a arquitetura de Von Neumann foi para a computação sequencial [Menezes and Setubal 1995]. Dessa maneira permitindo a concepção de algoritmos ao mesmo tempo portáveis e eficazes [Goldman 2003].

O modelo BSP funciona seguindo uma sequencia de passos dentro de intervalos de tempo, sendo que a cada final de intervalo, há uma sincronização geral entre os processadores, a qual deve ser feita pelo hardware. O BSP consiste dos seguintes componentes:

- Processadores e módulos de memória;
- Roteador para transporte das mensagens ponto a ponto entre os componentes;
- Mecanismos para sincronização entre todos os componentes.

Super-passo é o momento da atribuição das tarefas para cada componente de processamento, como leituras e escritas e transmissões de mensagens, onde cada processador é atribuído um conjunto de operações independentes, consistindo de uma combinação de passos de computação, usando dados disponibilizados localmente no início do superpasso. A Barreira de Sincronização define o momento para que haja uma sincronização global entre todos os componentes, assim, caso algum processador não tenha terminado sua tarefa, são atribuídos outros intervalos de tempo para execução das tarefas faltantes [Menezes and Setubal 1995].

3.3. *Coarse Grained Multicomputer* (CGM)

O modelo CGM (Coarse Grained Multicomputer) tem como característica ser muito parecido com o BSP, porem, mais simples. O CGM consiste em processadores idênticos, cada um com uma memória local, conectados por meio de uma interconexão arbitrária ou por memória compartilhada, e assim cada processador pode trocar mensagens com seus vizinhos imediatos [Dehne et al. 1996].

O tamanho de cada memória local é um exemplo de restrição que ocorre nesse modelo. Outro exemplo é que cada processador pode enviar ou receber um número limitado de dados. Assim, o desempenho do CGM pode ser calculado por meio do número de rodadas de comunicação. Os principais parâmetros considerados no CGM são:

- O número de processadores;
- Tamanho do problema.

A relação entre estes é o que determina a granularidade grossa, dai que vem o nome do modelo em inglês ”coarse grained”.

3.4. LogP

O modelo LogP é considerado um sucessor do modelo BSP, por manter há a presença de uma comunicação por rede ou memória compartilhada e máquinas. A diferença está no fato de o LogP ser totalmente assíncrono em relação a execução das tarefas, com isso neste modelo não existe nenhuma sincronização global entre os processadores.

Alguns parâmetros são importantes para o LogP:

- Capacidade da rede na transmissão das mensagens;
- A topologia da rede de interconexão não importa;
- O número de processadores e memórias locais;
- Custo de comunicação de um processador para enviar ou receber uma mensagem.

O custo de comunicação é determinado por três fatores que dão origem ao nome: L, o e g. O ”o”, representa o tempo em que um processador fica sem executar nenhuma ação após ter enviado ou recebido uma mensagem. O ”g”é o *gap*, que representa intervalo entre o envio de duas mensagens. O ”L”é a latência, que corresponde ao tempo entre o final do envio de uma mensagem e a recepção da mensagem por parte do destinatário [Goldman 2003].

Este modelo é considerado de baixo nível, por exemplo, quando comparado com o modelo PRAM, por conta dos diversos fatores que devem ser considerados na implementação do algoritmo [Menezes and Setubal 1995]. O modelo LogP consegue representar bem o comportamento de máquinas reais [Goldman 2003], tendo assim um grande uso na prática.

4. Conclusão

O estudo dos quatro modelos de Computação Paralela apresentados anteriormente (PRAM, BSP, CGM e LogP) permitiu perceber várias semelhanças entre os modelos, fica claro que a proposta de estudo destes modelos se deu pela importância que eles representam na evolução dos modelos de execução para Computação Paralela.

Cada modelo apresentado possui uma visão diferente em relação a computação paralela. O PRAM é mais voltado para análise e comparação de algoritmos. O BSP e CGM são modelos voltados para a programação. O BSP apresenta sincronização que facilita na elaboração de um algoritmo. O LogP, por apresentar diversos outros parâmetros além dos considerados no BSP e CGM, apresenta um grau de dificuldade maior para o projetista de algoritmos que terá como base o modelo LogP.

Vale ressaltar novamente a importância que a computação paralela apresenta ao longo deste anos de existência, permitindo ser mais viável resolver problemas anteriormente muito demorados para se executar, como na areá da bioinformática (para o enovelamento de proteínas) e na economia (para simulações de matemática financeira).

Referências

Backus, J. (1978). Can programming be liberated from the von neumann style?: a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641.

Cap, C. H. and Strumpen, V. (1993). Efficient parallel computing in distributed workstation environments. *Parallel Computing*, 19(11):1221–1234.

Dehne, F., Fabri, A., and Rau-Chaplin, A. (1996). Scalable parallel computational geometry for coarse grained multicomputers. *International Journal of Computational Geometry & Applications*, 6(03):379–400.

Goldman, A. (2003). Modelos para computação paralela. *Escola Regional de Alto Desempenho*, 3.

Karp, A. H. and Flatt, H. P. (1990). Measuring parallel processor performance. *Communications of the ACM*, 33(5):539–543.

Menezes, R. P. d. and Setubal, J. C. (1995). Modelos de computacao paralela e projeto de algoritmos.