Computação Paralela

Andrei E. H. Danelli[1], **Guilherme Fusieger**[2], **Luiz P. Reche**[3]

[1]Universidade Federal da Fronteira Sul (UFFS)
Caixa Postal 89815-899 – Chapecó – SC – Brasil
andrei.danelli@estudante.uffs.edu.br, guifusieger@gmail.com, luiz.reche@estudante.uffs.edu.br

Resumo. *Este relatório explica de forma simplificada como a Computação Paralela funciona, destacando que os computadores paralelos podem resolver certos problemas muito mais rapidamente do que os computadores sequenciais, que só conseguem fazer uma única operação por vez, enquanto os paralelos conseguem realizar varias operações de forma paralela. As pesquisas realizadas têm o objetivo de esclarecer o funcionamento de cada tópico, que incluem Circuitos Booleanos Uniformes, a Classe NC e a P-Completude.*

1. Introdução

Atualmente, com equipamentos mais poderosos e com uma demanda de processamento maior sobre dados, surge necessidades de se haver conceitos e métodos para se melhorar a velocidade de processamento, dessa forma, entregando tais dados de forma mais rápida e consistente, com isso, surge o conceito de Computação Paralela, que basicamente melhora o processamento de múltiplas tarefas, especificamente em situações que há alto número de dados, utilizando-se do conceito de paralelismo, algo que até existe em computadores sequenciais mas em escala menor. Basicamente, computadores sequenciais executam instruções individualmente, já na computação paralela vai ser utilizado vários elementos processadores em uma única computação.

Para aprofundar mais o que forma a computação Paralela, vamos ver conceitos de Circuitos Booleanos Uniformes, que é um modelo matemático para circuitos lógicos digitais, conceito de Classes NC, que são um conjunto dos problemas de decisão de computadores paralelos, e por fim, P-Completude, que também é um conceito relacionado a problema de decisão.

2. Circuitos Booleanos Uniformes

Circuitos booleanos uniformes são uma classe de circuitos computacionais usados na teoria da complexidade e em algoritmos paralelos. Eles são definidos de forma que o circuito para cada entrada de comprimento n possa ser gerado por uma máquina de Turing em tempo polinomial em n. Isso permite uma definição mais rigorosa e prática para a análise de complexidade dos circuitos. Circuitos booleanos têm certas vantagens e desvantagens como um modelo de computação paralela. No lado positivo, o modelo é simples de descrever, tornando as provas mais fáceis. No lado negativo, circuitos são difíceis de programar pois os processadores individuais são muito fracos.

2.1. Máquina de Acesso Aleatório Uniforme Paralela (MAAP)

O modelo Máquina de Acesso Aleatório Paralela Uniforme (MAAP), mais conhecido em inglês como Parallel Random Access Machine (PRAM), é uma abstração teórica utilizada

para estudar algoritmos paralelos. Este modelo assume que vários processadores operam em paralelo e compartilham uma memória comum, onde cada processador pode acessar qualquer posição de memória em tempo constante. O modelo PRAM é usada por projetistas de algoritmos para modelar o desempenho, complexidade de tempo, onde o número de processadores assumido, normalmente, é também indicado.

2.2. Conflitos de leitura/escrita

Existem várias variantes do modelo **PRAM**, baseadas nas permissões de leitura e escrita na memória compartilhada e com isso apresenta várias limitações quando se trata de aplicabilidade prática e realismo. Alguns pontos negativos do modelo **PRAM** são: Irrealismo no Acesso à Memória Compartilhada, Sincronização e Comunicação, Escalabilidade Limitada e outros. Porém os conflitos por acessar memória compartilhada são resolvidos com as seguintes formas abaixo.

1. **Leitura exclusiva gravação exclusiva (EREW - Exclusive read exclusive write)** - Toda célula de memória só pode ser lida ou escrita por um único processador de cada vez.
2. **Leitura concorrente gravação exclusiva (CREW - Concurrent read exclusive write)** - múltiplos processadores podem ler uma mesma célula de memória mas somente um processador pode escrever de cada vez.
3. **Leitura exclusiva gravação concorrente (ERCW - Exclusive read concurrent write)** - nunca considerada.
4. **Leitura concorrente gravação concorrente (CRCW - Concurrent read concurrent write)** - Múltiplos processadores podem ler e escrever. Uma CRCW PRAM é as vezes chamada de máquina de acesso randômico concorrente.

2.3. Código de exemplo

Logo abaixo temos um exemplo na linguagem de descrição de hardware **Verilog**, o qual implementa um módulo chamado **FindMax** para encontrar o valor máximo em uma matriz de dados em apenas 2 ciclos de relógio. O módulo opera da seguinte forma:

- Estado COMPARE: No primeiro ciclo, compara todos os pares de elementos na matriz data. Se um elemento é menor que outro, marca-o em um array auxiliar m.
- Estado MERGE: No segundo ciclo, identifica o elemento que não foi marcado (ou seja, o maior valor) e o armazena na saída maxNo.
- Estado DONE: Indica que a operação está completa.

Este design utiliza memória **CRCW** para permitir gravações simultâneas sem conflitos, já que a lógica garante que o mesmo valor é escrito nas mesmas posições de memória.

```
module FindMax #(parameter int len = 8)
                (input bit clock, resetN, input bit[7:0] data[len], output bit[7:0] maxNo);
    typedef enum bit[1:0] {COMPARE, MERGE, DONE} State;

    State state;
    bit m[len];
    int i, j;

    always_ff @(posedge clock, negedge resetN) begin
        if (!resetN) begin
            for (i = 0; i < len; i++) m[i] <= 0;
            state <= COMPARE;
        end else begin
            case (state)
                COMPARE: begin
                    for (i = 0; i < len; i++) begin
                        for (j = 0; j < len; j++) begin
                            if (data[i] < data[j]) m[i] <= 1;
                        end
                    end
                    state <= MERGE;
                end

                MERGE: begin
                    for (i = 0; i < len; i++) begin
                        if (m[i] == 0) maxNo <= data[i];
                    end
                    state <= DONE;
                end
            endcase
        end
    end
endmodule
```

Figura 1 - Implementação Módulo FindMax. — Máquina de acesso randômico paralelo

3. Classes NC

A classe NC é uma classe de complexidade referente a problemas computacionais que podem ser resolvidos por meio de circuitos booleanos paralelos, ou seja, em circuitos que podem executar diversas tarefas simultaneamente. A classe NC está relacionada a varias outras classes de complexidade, tais como, Classe P, sendo problemas solúveis em tempo polinomial e a classe L, que é de problemas solúveis em espaço logarítmico. Para se provar a relação da classe NC com outras classes utilizamos teoremas específicos, tais como, NC^1 subconjunto de L, NL subconjunto de NC^2 e NC subconjunto de P.

3.1. Teorema 1: NC^1 subconjunto de L

NC^1 representa problemas que possuem soluções com profundidade logarítmica e tempo polinomial e L representa problemas com solução em espaço logarítmico, com isso, podemos provar que NC^1 está em L da seguinte forma: Em uma entrada W de tamanho n, podemos construir o n-ésimo circuito do conjunto que decide A, posteriormente executamos uma busca em profundidade partindo da saída do circuito, no decorrer da execução da DPS, o algoritmo precisa guardar o caminho encontrado até então e resultados parciais obtidos, para isso, utilizasse de um espaço logarítmico, considerando que o circuito possui profundidade logarítmica também, portanto, podemos concluir que qualquer problema em NC^1 pode ser resolvido em L, pois como o circuito possui profundidade logarítmica e pode ser construída em um espaço logarítmico.

3.2. NL subconjunto de NC²

NC² representa problemas que possuem soluções com profundidade logarítmica ao quadrado, O((log n)²), e tamanho polinomial, já NL, representa problemas com solução em uma maquina de Turing não determinística com espaço logarítmico. Nesse caso para se provar que um problema em NL pode ser resolvido por NC², podemos utilizar da construção de um grafo de configurações, onde basicamente iremos localizar se há um caminho que vá da configuração inicial à configuração de aceitação, o algoritmo assim contendo um tamanho polinomial e uma profundidade logarítmica ao quadrado. Com essas características, sabemos que problemas em NL podem ser resolvidos por um NC².

3.3. NC Subconjunto de P

NC representa problemas que podem ser resolvidos por circuitos booleanos com profundidade polilogaritimica, O((log n)^k) e com tamanho polinomial, e P problemas que pode ser resolvido por algoritmos determinísticos em tempo polinomial. Para provar que problemas em NC podem ser resolvidos em P, pegamos um problema qualquer em NC, e usamos um transdutor de espaço logarítmico para gerar um circuito Cn, que corresponde a uma entrada de comprimento n, posteriormente, um algorítimo de tempo polinomial pode executar tal transdutor para gerar o circuito, e posteriormente simular o mesmo, dessa forma, como o circuito tem tamanho polinomial, a simulação também poderá ser feita em tempo polinomial, provando assim que NC é subconjunto de P.

4. P-completude

P-completude é um conceito em teoria da complexidade computacional que se refere a problemas que são, de certo modo, os mais difíceis dentro da classe de problemas que podem ser resolvidos em tempo polinomial(classe P). Um problema é considerado P-completo se ele é um problema na classe P, e também, todo problema em P pode ser reduzido a ele por uma redução em tempo polinomial. Sua relação com a computação paralela é interessante, visto que um problema P-completo é considerado difícil de paralelizar de forma eficiente, o que significa que não é esperado que exista um algoritmo paralelo que resolva o problema significativamente mais rápido do que o melhor algoritmo sequencial. Os problemas P-completos são importantes porque se acredita que esses problemas não estão em NC, assim, não se espera que exista uma forma eficiente de paralelizar esses problemas a ponto de serem resolvidos em tempo logarítmico com um úmero polinomial de processadores.

4.1. Definição

Uma linguagem é P-completa se:

1. B$\in$P;
2. Toda A em P é redutível em espaço log a B.

Uma linguagem B é P-completa se B pertence a classe P, ou seja, deve ser solucionável por uma máquina de Turing em tempo polinomial, e todo problema A em P deve ser redutível ao problema B através de uma redução que pode ser computada em espaço logarítmico. Isso se traduz em uma transformação eficiente que converte instâncias de A em instâncias de B usando apenas uma quantidade de memória que cresce logaritmicamente com o tamanho da entrada.

4.2. Teorema

Se $A \leq_L B$ e B está em NC, então A está em NC.
Esse teorema afirma que um problema A pode ser reduzido a um problema B por uma redução log-space(denotada $A \leq_L B$), e B pertence a classe NC(problemas que podem ser resolvidos em tempo polilogarítmico com um número polinomial de processadores), onde A pertence também a NC, indicado que a classe NC é fechada sob reduções log-space.

$$\text{VALOR-CIRCUITO} = \{(C,x) \mid C \text{ é um circuito booleano e } C(x)=1\}.$$

O problema de VALOR-CIRCUITO é definido como o problema de determinar se um circuito booleano C produz a saída 1 para uma entrada x específica((C,x) tal que C(x)=1). Esse problema é mostrado como P-completo, sendo um dos problemas mais difíceis na classe P sob reduções log-space. Isso implica que se pudermos resolver esse problema de maneira eficiente usando computação paralela(em NC), então qualquer problema na classe P pode ser resolvido de maneira eficiente usando computação paralela, o que é considerado improvável pela suposição de que $P \neq NC$. Para a computação paralela, a P-completude de VALOR-CIRCUITO indica que esse problema é candidato para estudar os limites da paralelização eficiente dentro da classe P. Problemas P-completos são vistos como intrinsecamente difíceis de paralelizar, e o entendimento desses problemas ajuda a definir o que pode ou não ser alcançado com algoritmos paralelos.

Referências

Arora, S. (2007). P-completness. In Arora, S., editor, *Computational Complexity: A Modern Approach*, pages 110–111. Princeton University.

Sipser, M. (2004). Introdução à teoria da computação. In Sipser, M., editor, *Computação Paralela*, pages 424–429. Cengage.

Wikipédia (2023). Máquina de acesso randômico paralelo. https://l1nk.dev/j5UVd.

[Sipser 2004] [Wikipédia 2023] [Arora 2007]