Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação de São Carlos
SSC-143 - Programação Concorrente

Relatório - Trabalho I

Docente:
Dr. Paulo Sergio Lopes de Souza

Alunos:

Eder Rosati Ribeiro	8122585
Pedro Puzzi	6513497
Wesley Tiozzo	8077925

Setembro de 2017

Sumário

1 Introdução

Em álgebra linear, o algoritmo para solução de sistemas lineares conhecido por **Eliminação de Gauss-Jordan** é uma versão da Eliminação de Gauss que zera os elementos acima e abaixo do elemento de pivotação, conforme ele percorre a matriz. Em outras palavras, a eliminação de Gauss-Jordan transforma a matriz em uma forma escalonada por colunas reduzida.

2 Pseudocódigo do algoritmo sequencial

```
FormaEscalonadaReduzida(A[n][m]: Real ){
INÍCIO
    // Número de linhas
    nrows = n: Inteiro;
    // Número de colunas
    ncols = m: Inteiro;
    // Pivô
    linhaDoPivo = 0: Inteiro;
    ENQUANTO (linhaDoPivo < nrows)
        d, m: Real;
        c, r: Inteiro;
        // Para cada linha
        PARA r de 0 até nrows FAÇA
            // Calculando divisor e múltiplo
            d = A[linhaDoPivo][linhaDoPivo];
```

```
                    m = A[r][linhaDoPivo] / A[linhaDoPivo][linhaDoPivo];
                    // Para cada coluna
                    PARA c de 0 até ncols FAÇA
                        SE r == linhaDoPivo ENTÃO
                            A[r][c] /= d;  // ajusta a linha do pivô
                        SENÃO
                        // ajusta as linhas que não são a linha do pivô
                        A[r][c] -= A[linhaDoPivo][c] * m;
                        FIM SENÃO
                        FIM SE
                    FIM PARA
                FIM PARA
                linhaDoPivo++;
                ESCREVER (“fim do grande passo: ”, linhaDoPivo - 1);
        FIM ENQUANTO
FIM
}
```

3 Paralelismo

O primeiro passo no desenvolvimento do algoritmo paralelo se refere a decomposição do problema em tarefas que possam ser executadas concorrentemente. Uma das formas de decomposição pode ser ilustrada na forma de um grafo direcionado e acíclico (grafo de dependências de tarefas) com nodos que correspondem às tarefas e arestas a qual indicam o resultado de uma tarefa para o processamento da próxima.

4 Representação gráfica

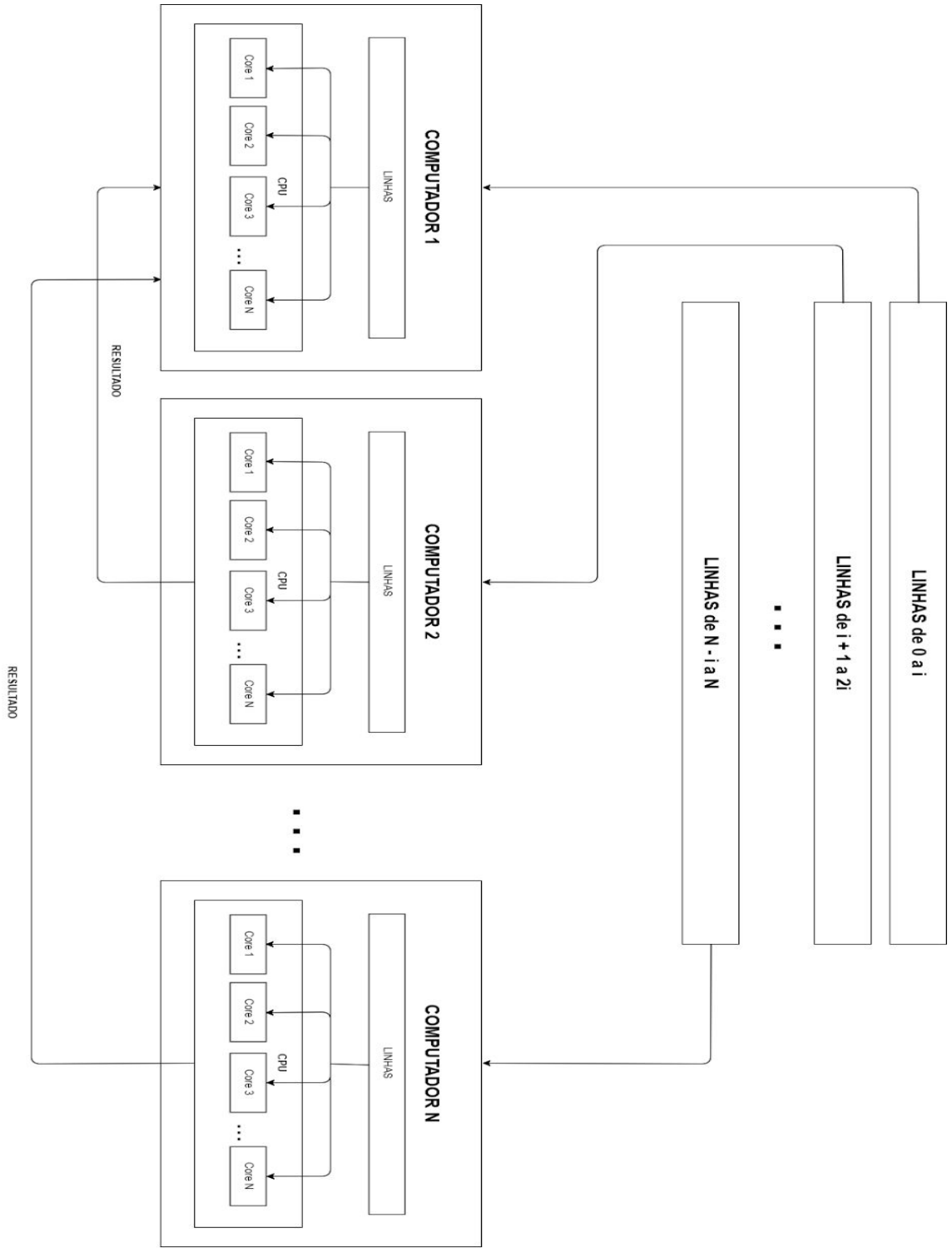

Figura 1: Representação gráfica

5 Particionamento

Este passo se refere a quebrar o programa em blocos de trabalho que possam ser distribuídos em tarefas de granularidade fina que possam ser executadas concorrentemente.

Primeiramente, é preciso entender em linhas gerais o funcionamento do algoritmo sequencial

O algoritmo recebe uma matriz aumentada de tamanho n x m. Ele precisa fazer então n "grandes passos", um para cada linha, que resulta no ajuste da linha n (linha pivô) e na multiplicação dessa linha pivô para zerar as colunas correspondente a esse elemento.

Exemplo de uma matriz aumentada 3x4:

5	-6	-7	7		1	-1.2	-1.4	1.4	
3	-2	5	-17	---->	0	1.6	9.2	-21.2	----->
2	4	-3	29		0	6.4	-0.2	26.2	

1	0	5.5	-14.5		1	0	0	2
0	1	5.75	-13.25	---->	0	1	0	4
0	0	-37	111		0	-0	1	-3

Dentro desse grande passo, são necessárias n x m operações, porque cada elemento da matriz vai ser ajustado dentro desse grande passo. Então para esse pequeno exemplo teríamos 3 grandes passos e 12 pequenos, resultando em 36 operações aritméticas básicas.

O grande passo (loop mais externo, que é repetido de 1 até n linhas) são dependentes uns dos outros, para se realizar o passo seguinte é necessário que o passo anterior esteja concluído.

Já os pequenos passos, são independentes dos dados, ou seja, é possível realizar as nxm operações separadamente, e isso leva a uma abordagem de particionamento por dados.

A partir dessas conclusões, decidimos particionar o problema de duas maneiras. Primeiramente fazemos um particionamento das tarefas onde cada tarefa é ou o ajuste da linha pivô ou é a multiplicação da linha pivô pelas outras n-1 linhas.

Depois particionamos essas linhas recebidas por cada um dos processos em operações atômicas (somas e subtrações) e esse particionamento é onde exploramos mais fortemente o paralelismo deste algoritmo.

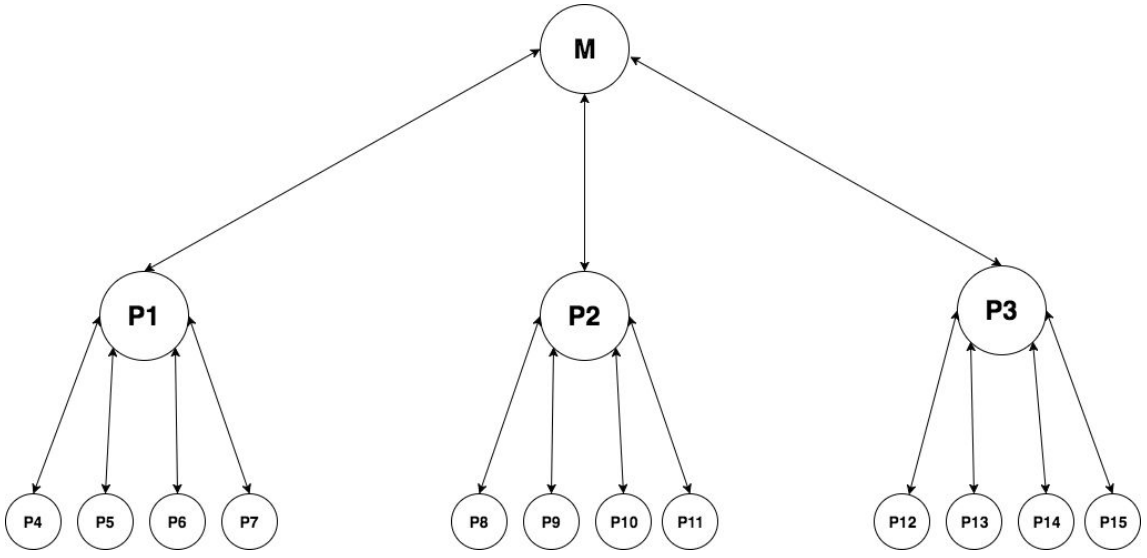

Figura 2: Particionamento (Exemplo para uma matriz 3x4)

- Checklist de particionamento:

1. A partição define pelo menos uma ordem de magnitude de mais tarefas do que processadores no computador alvo? Primeiramente a abordagem foi de 1 para 1, mas isso foi revisto ao longo dos outros passos.
2. A partição previne computação redundante e requisitos de armazenamento? Sim, cada tarefa receberá apenas os dados que lhe são necessários.

3. Há tarefas de tamanho similar? Sim, possui o mesmo tamanho.
4. O número de tarefas mantém proporção com o tamanho do problema? Sim, e isso indica que a solução proposta é escalável em função do tamanho do problema.

6 Comunicação

Este passo se refere a determinação de um padrão de comunicação determinado pela dependência de dados entre as tarefas. A maioria das aplicações paralelas não são tão simples e precisam que as tarefas possam compartilhar dados umas com as outras.

Nessa etapa da metodologia, devemos nos atentar às informações necessárias para cada tarefa realizar a computação que lhe foi atribuída.

Temos em princípio uma tarefa mestre que tem como função dividir nosso problema e enviar uma mensagem para os outras tarefas contendo suas respectivas partes. Essa mesma tarefa mestre também é responsável por esperar o encerramento das outras tarefas (fim do chamado "grande passo"), reunir os resultados e dar início ao passo seguinte.

Cada tarefa receberá da tarefa mestra a linha pivô e a linha que deve ser ajustada em relação a linha pivô e deve devolver ao mestre esse resultado também por troca de mensagens.

Como foi proposto rodar a aplicação em Cluster Multicore, devemos realizar a troca de mensagens através de por exemplo uma rede ethernet. O computador mestre deve se comunicar com os outros de forma bidirecional, e não é necessário que os outros computadores estejam ligados entre si porque eles realizam tarefas independentes.

Há a necessidade de esperar as tarefas terminarem e reportarem seus resultados ao mestre, o que pode ser garantido com o uso de abordagens como o semáforo.

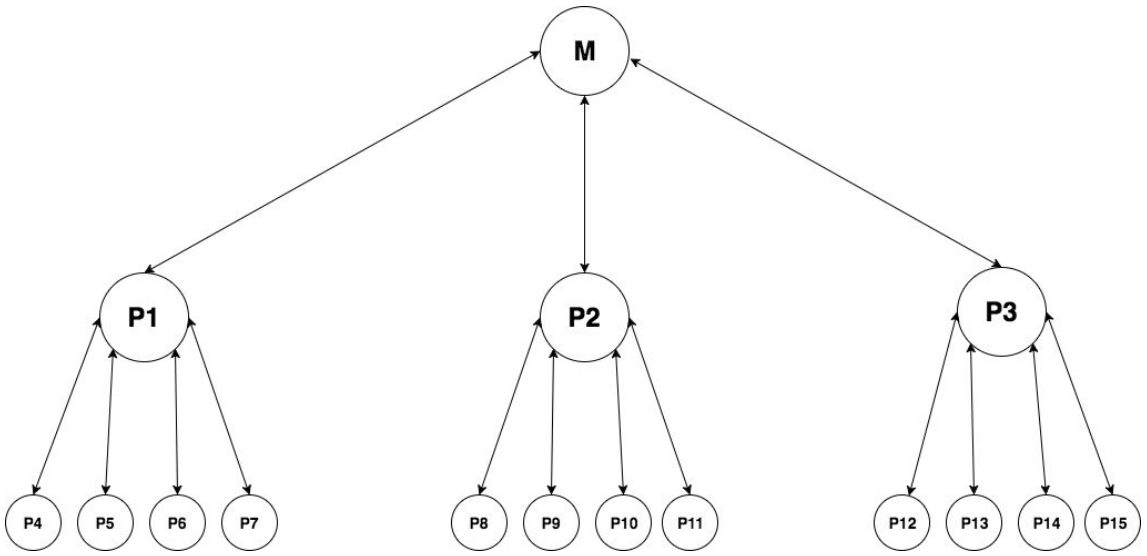

Figura 2: Comunicação (Exemplo para uma matriz 3x4)

- Checklist de Comunicação:

1. Todas as tarefas possuem o mesmo número de operações de comunicação? Sim, a única que é diferente é a tarefa mestre.
2. Cada tarefa se comunica apenas com um pequeno número de vizinhos? Sim, apenas com o mestre.
3. As operações de comunicação podem ser realizadas concorrentemente? Sim.
4. A computação associada com diferentes tarefas pode ser realizada concorrentemente? Sim .

7 Aglomeração

Esta seção se refere a combinação de grupos de tarefas de granularidade fina para a formação de poucas tarefas de granularidade grossa, assim então reduzindo os requisitos de comunicação.

Granularidade grossa é quando há grandes trechos de código com comunicação baixa, ou seja, há menor potencial de exploração de paralelismo.

A partir do grafo de dependência, podemos perceber que as tarefas do "pequeno passo" podem ser aglomeradas em um processo só ao invés de termos diversos processos. Essa prática nos ajuda a diminuir o overhead gerado na comunicação entre eles.

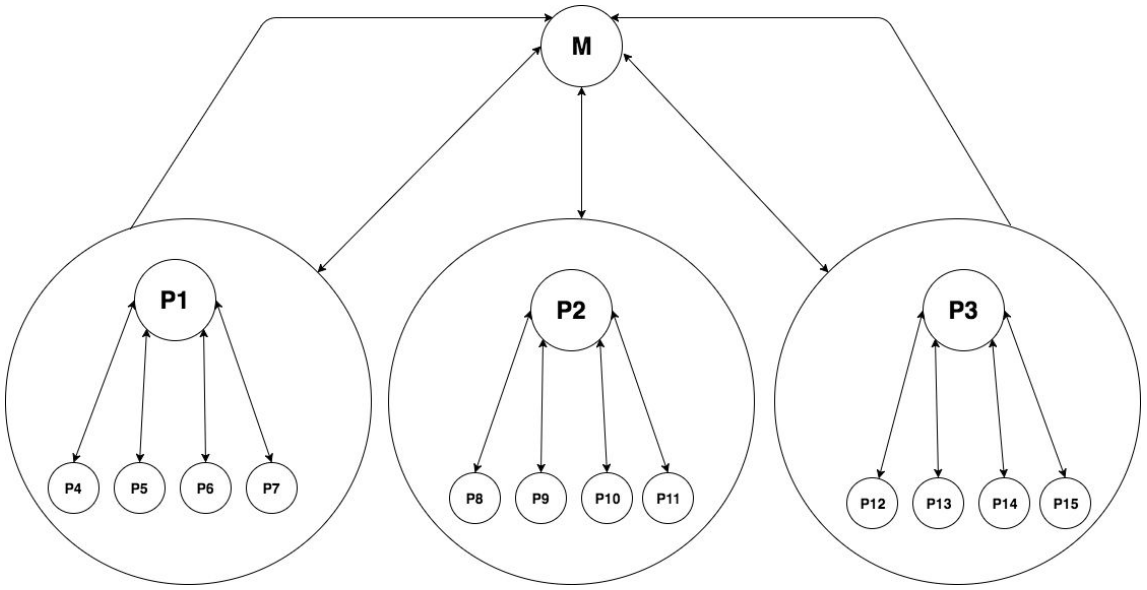

Figura 3: Aglomeração das tarefas (Exemplo para uma matriz 3x4)

- Checklist de aglomeração:

1. A aglomeração reduziu custos de comunicação através do aumento da localidade? Sim.
2. Se a aglomeração possui computação replicada, foi verificado se os benefícios desta replicação excedem os custos, para uma variedade de tamanhos de problemas e contagens de processadores? Não possui computação replicada.

3. Se a aglomeração replica dados, foi verificado se isso não compromete a escalabilidade do algoritmo em relação a restrição da variedade de tamanhos de problemas ou contagens de processadores que os mesmos podem endereçar?
4. A aglomeração permitiu tarefas com custos de computação e comunicação similar? Sim.
5. O número de tarefas ainda mantém proporção com o tamanho do problema? Sim, a mesma proporção.
6. Se a aglomeração eliminou oportunidades para a execução concorrente, foi verificado se há concorrência suficiente para os computadores atuais e os computadores alvo? Ela não eliminou.
7. O número de tarefas pode ser reduzido ainda posteriormente, sem introduzir desbalanceamento de carga, aumento de custo da engenharia de software, ou redução da escalabilidade? Não sei.
8. Se está sendo paralelizado um programa sequencial, foi considerado o custo de modificações necessários para o código sequencial? Não

8 Mapeamento

Esta seção se refere ao mapeamento das tarefas de granularidade grossa em múltiplos processadores, o qual é um passo crítico em relação a minimizar overhead de processamento paralelo.

O mapeamento foi pensado conforme a figura (FIGURA)

Após realizadas as etapas de particionamento, comunicação, aglomeramento e levando em consideração o ambiente escolhido para rodar a aplicação, devemos agora então mapear cada tarefa a um nó do cluster.

A princípio, cada linha da nossa matriz será enviada a nó do cluster, e isso implica então que para uma matriz com N linhas, precisaríamos de N computadores, o que é uma situação irreal.

Podemos contornar isso atribuindo um maior número de linhas passadas ao nosso cluster, como podemos ver na figura, e isso fica também a cargo do nosso computador mestre. Isso implica em um menor paralelismo, mas em contrapartida também temos menos overhead gerado pela comunicação.

- Checklist de mapeamento:

1. Se for considerado um design SPMD para um problema complexo, foi considerado um algoritmo baseado na criação e remoção de tarefas dinâmicas? Não, pois o algoritmo tem um número definido de passos em relação a entrada
2. Se for considerado um design baseado na criação e remoção de tarefas dinâmicas, foi considerado um algoritmo SPMD? O algoritmo é SPMD por definição
3. Se for utilizado um esquema de balanceamento de carga centralizado, foi verificado se o gerente não se tornará um bottleneck (congestionamento, barreira)? Fora do escopo

4. Se for utilizado um esquema de balanceamento de carga dinâmico, foi avaliado os custos relativos a diferentes estratégias? Fora do escopo
5. Se for utilizado métodos probabilísticos ou cíclicos, há um número de tarefas suficientemente grande para assegurar o balanceamento de carga? Fora do escopo

9 Referências

[1] FOSTER, Ian. Designing and building parallel programs. Boston: Addison Wesley Publishing Company, 1995.

[2] GRUPTA, A., Kumar, V., Grama, A., & Karypis, G. (2003). Introduction to Parallel Computing.

[3] BIEZUNER, Rodney Josué. Notas de Aula do Ciclo Básico: Sistemas Lineares. UFMG, 2010.

[4] LAGES, Elon. Algebra linear. IMPA, Rio de Janeiro, 2009, cap. 9.

[5] "Método de Gauss-Jordan - Prof. Dr. Waldeck Shutzer" , https://www.dm.ufscar.br/profs/waldeck/

[6] "Wikipedia - Gauss Elimination", https://en.wikipedia.org/wiki/Gaussian_elimination