PCAM Menor valor maior que Mat[i,k]

- Daniel Sá Barretto Prado Garcia 103734344
- Felipe Guilermmo Santuche Moleiro 10724010
- Laura Alves de Jesus 10801180
- Tiago Marino Silva 10734748

Particionamento

O problema consiste em encontrar o menor valor maior que M[i, k] para todas as linhas de uma matriz quadrada M de tamanho n, sendo k um valor passado como parâmetro. Será feito um particionamento por dados sobre a matriz. Esta matriz será particionada em n linhas e cada linha será dividida em partes iguais, podendo resultar em até n partes por linha. Assim, cada linha irá gerar até n tarefas para si e, após o término dessas tarefas geradas, é feita uma redução por linha de ordem logarítmica para encontrar o menor valor. Então, cada tarefa gerada terá uma parte da linha pela qual foram geradas, na qual buscarão pelo menor valor maior que M[i, k] (sendo 0 <= i < n) dentro dessa parte e salvarão o valor encontrado para que seja calculado o valor mínimo maior que M[i, k] da linha com a operação de redução mencionada anteriormente. Desta forma, as tarefas podem ser executadas em paralelo e, após a execução de todas de uma mesma linha, é feita uma comunicação para encontrar o valor mínimo entre os calculados.

Comunicação

A comunicação é feita da seguinte forma: a matriz M é uma variável global compartilhada por todas as tarefas sem problemas, pois todas as tarefas apenas executarão operações de leitura sobre ela. Cada tarefa terá um mínimo local privado que salva o mínimo valor maior que M[i, k] encontrado em sua parte da linha. Após isso as threads se comunicam através de uma redução por linha, em que as threads executando em uma mesma linha vão ter seus mínimos locais comparados para encontrar o mínimo da linha inteira. Finalmente, o valor mínimo da linha inteira é guardado em um vetor compartilhado de resultados de tamanho N.

Aglomeração

Como estamos utilizando apenas um nó do cluster que é uma máquina MIMD com memória compartilhada, a aglomeração será feita da seguinte forma: Serão geradas T threads, em que T é o um valor baseado no número de núcleos de processamento lógicos existentes na máquina sendo executada.

Após a leitura da entrada, teremos n linhas com n posições cada. Iremos dividir a matriz em T blocos, ou seja, cada thread será responsável por calcular o mínimo maior que M[i,k] em (N/T) linhas da matriz. Por termos aglomerados todos os elementos de uma linha juntos em uma única thread, não é necessário fazer uma redução por linha já que o resultado encontrado considera a linha inteira e não

precisa se comunicar com outras threads executando na mesma linha, evitando esse overhead de comunicação, já que o número de unidades de processamento existentes não é muito grande. Se estivéssemos trabalhando com um cluster inteiro em que cada máquina é uma MIMD, poderíamos dividir as linhas entre os nós de um cluster e dividir cada linha entre os núcleos de um nó, e dessa forma usaríamos o reduction em cada linha, entretanto por utilizarmos apenas um nó, o método mais eficiente é não particionar a linha, ou seja, particionar uma linha em uma única parte, já que o número de linhas é bem maior que o número de unidades de processamento, evitando o custo de fazer uma redução. Por exemplo, se há 8 linhas e temos 4 núcleos de processamento lógicos, serão geradas 4 threads lógicas, em que cada thread lógica achará um mínimo valor maior que o valor de M[i, k] para cada linha em um bloco de (8/4) linhas.

Mapeamento

O mapeamento das Threads em cada núcleo de processamento lógico é feito pelo SO e não depende da implementação. Entretanto, ao criar um número de threads iguais ao número de núcleos de processamento lógico, espera-se que com sorte o SO escalone cada thread para um núcleo, utilizando toda a capacidade do processador de uma só vez.