Atividade 1 - PCAM

João Pedro A.S. Secundino (10692054); João Pedro Uchôa Cavalcante (10801169); Luís Eduardo Rozante de Freitas Pereira (10734794); Sérgio Ricardo G. B. Filho (10408386);

O objetivo desta atividade é descrever o PCAM desenvolvido para a resolução da atividade proposta em sala. O projeto a seguir divide o problema em dois subproblemas funcionais (baseados na decomposição funcional). Cada um destes subproblemas é, em seguida, decomposto utilizando a técnica da decomposição por dados. A equipe optou por este esquema híbrido de decomposição por acreditar que este provê uma maior clareza de detalhes sobre as etapas necessárias para o alcance da solução.

1. O problema

Objetivo: Preencher um vetor e encontrar o seu maior elemento.
Restrições:

1. Gere os números usando o valor 1 para todos os elementos e em seguida substitua o valor da posição do meio do vetor (*tamanho_vetor / 2*) pelo valor de *tamanho_vetor*;
2. Não use a diretiva for do *omp* (não confundir com o for do C);
3. Calcule os limites das iterações dos *loops* de cada *thread* no seu código explicitamente;
4. Paralelize a geração dos números no vetor e também a busca do maior valor;
5. Sincronize com uma diretiva *barrier*, entre a geração dos números e a busca do maior valor;
6. Na paralelização dos *for's*, distribua as iterações estaticamente por blocos contínuos:
   - *Thread* 00 – gera elementos do bloco 0 (0 até *tam/num_threads*);
   - *Thread* 01 – gera elemsdo bloco 1 (*tam/num_threads* até (*tam/num_threads*)*2);
   - ...
7. Não use a cláusula *reduce* e, sim, otimize o uso da memória compartilhada;
8. Imprima na tela o maior valor encontrado no vetor, como feito em sala de aula.

2. Design

O problema pode ser, inicialmente, divido em duas etapas fundamentais: (1) preencher o vetor e (2) buscar o maior elemento do mesmo. O projeto de paralelização destas duas funcionalidades é detalhado a seguir.

2.1 Problema do preenchimento do vetor

2.1.1 P - Particionamento

O particionamento desta funcionalidade consiste na criação de $\boldsymbol{N}$ tarefas, sendo $\boldsymbol{N}$ o tamanho do vetor. Cada tarefa possui como objetivo preencher uma posição deste vetor. Este preenchimento deve seguir a seguinte regra: aquelas tarefas correspondentes à posição ***i != N/2 - 1*** do vetor deverão preencher ***v[i]*** com o valor 1. A tarefa responsável pela posição ***i == N/2 - 1*** deverá preencher ***v[i]*** com o valor $\boldsymbol{N}$.

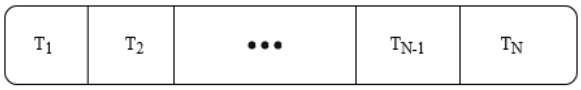

Figura 1: Esquema do particionamento de dados. Cada tarefa T_i está associada à posição $v[i]$.

2.1.2 C - Comunicação

A comunicação está associada ao fornecimento dos dados de entrada para a realização de cada tarefa. Cada tarefa recebe ***i***, $\boldsymbol{N}$ e ***v[i]***. Com estes dados, cada tarefa terá informação suficiente para atribuir o valor correto à posição ***v[i]***. As tarefas, após o término das suas execuções, deverão ser sincronizadas de forma a manter a consistência de dados para a próxima etapa descrita na Seção 2.2.

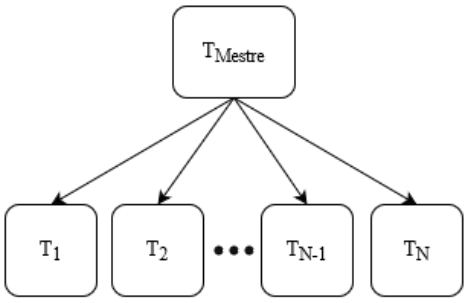

Figura 2: Esquema da comunicação entre as tarefas. A tarefa mestre T_m é a responsável por enviar os dados necessários para as subtarefas.

2.1.3 A - Aglomeração

Sugere-se o agrupamento das $\boldsymbol{N}$ tarefas em $\mathbf{P}$ processos, sendo $\boldsymbol{P \geq n\_processadores}$ (número de processadores). Desta forma, cada tarefa (após a aglomeração) será composta de $\frac{N}{P}$ subtarefas, as quais não necessitam se comunicar entre si. No caso de a divisão não

ser exata (resultar em resto R), as tarefas restantes serão distribuídas entre os processos de forma semelhante a uma fila circular, na qual cada um dos R processos recebe uma subtarefa a mais para realizar.

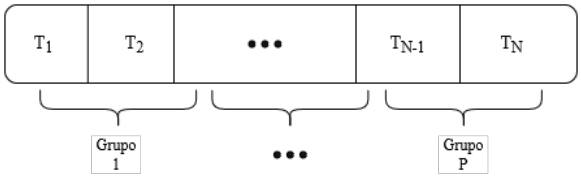

Figura 3: Esquema da aglomeração das tarefas.

2.1.4 M - Mapeamento

De acordo com Ian Foster[1], o problema do mapeamento não acontece em sistemas que possuem memória compartilhada e oferecem escalonamento automático de tarefas. Portanto, nesta atividade, como o computador no qual o programa será executado possui as características citadas, o problema do mapeamento não será abordado durante a implementação do algoritmo.

Em caso de o programa ser executado em uma máquina que não possui estas características, o $\boldsymbol{P}$ processos podem ser atribuídos às unidades de processamento a partir de uma fila circular. Se $\boldsymbol{P} == \boldsymbol{n\_processadores}$, cada unidade de processamento recebe uma quantidade equivalente de carga de trabalho. Caso não, os processos restantes serão adicionados às unidades na ordem em que elas aparecem na fila.

2.2 Problema da busca pelo maior valor

2.2.1 P - Particionamento

Utilizando o particionamento por dados, cada tarefa tem como objetivo verificar o valor de uma posição do vetor e compará-lo com o maior valor global. Caso o valor em $\boldsymbol{v[i]}$ seja maior que o global, este último deve ser atualizado. No total, serão geradas $\boldsymbol{N}$ tarefas, cada uma com acesso a um elemento $\boldsymbol{v[i]}$. Esse particionamento é esquematizado pela Figura 1.

2.2.2 C - Comunicação

Existirão duas comunicações nesta funcionalidade: a primeira está relacionada ao fornecimento de informação para a execução de cada tarefa e a segunda à busca do maior elemento do vetor. A primeira (local), esquematizada na Figura 2, acontecerá durante a criação de cada tarefa. Cada tarefa $\boldsymbol{i}$ receberá a posição correspondente do vetor que deve ser analisada. A segunda (global), esquematizada na Figura 4, acontecerá quando as tarefas necessitarem comparar seus valores com o maior valor global. A tarefa mestre deverá possuir, ao fim da execução de todas as outras, o maior elemento do vetor $\boldsymbol{V}$.

1. FOSTER, Ian. Designing and building parallel programs: concepts and tools for parallel software engineering. Addison-Wesley Longman Publishing Co., Inc., 1995.

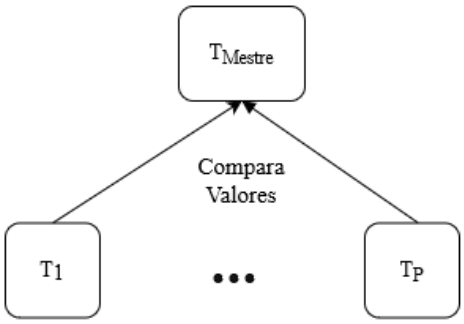

Figura 4: Esquema da comunicação global necessária durante a busca do maior elemento do vetor.

2.2.3 A - Aglomeração

Sugere-se o mesmo agrupamento descrito na Subseção 2.1.3.

2.2.4 M - Mapeamento

Sugere-se o mesmo agrupamento descrito na Subseção 2.1.4.

3. Esquema geral da aplicação

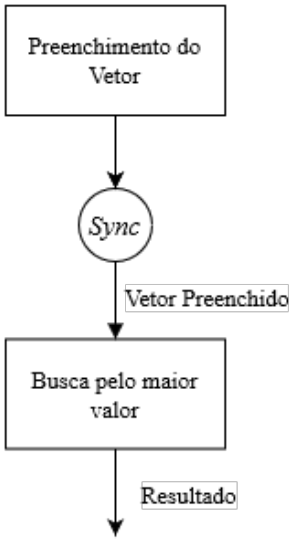

Figura 5: Esquema final da aplicação.