Metodologia PCAM - Atividade 1

Bruno Baldissera Carlotto	10724351
Bruno Gazoni	7585037
Gabriel Eluan Calado	10734453
João Villaça	10724239
Matheus Steigenberg Populim	10734710

Particionamento

O vetor de tamanho *tam* é particionado unidimensionalmente em **T** pedaços, sendo **T** o número de threads a serem empregadas na paralelização. Busca-se o maior elemento em cada pedaço através de busca sequencial e em seguida compara-se os **T** resultados para atingir o maior elemento do vetor original. Um problema sequencial **O(n)** é assim paralelizado em **O(n/T)**.

Comunicação

É necessário utilizar uma barrier explícita para sincronização de threads, para buscar o maior elemento em um vetor que já foi particionado a partir do original e corretamente preenchido. Sem o uso de barriers, não podemos garantir a presença de todos os elementos necessários no momento da comparação. É utilizada uma região *single* para inicializar o vetor com os maiores elementos dos subvetores, visto que é uma tarefa pequena que se tornaria redundante em múltiplas threads.

Aglomeração

As tarefas são aglomeradas formando ***T*** *threads*, e cada *thread* é responsável por encontrar sequencialmente o maior valor de um subvetor de tamanho *(tam/T)*, que tem como início a posição *(tam/T)*(id_da_thread)*, e como fim *(tam/T)*(id_da_thread + 1)*. Ao longo da execução, cada *thread* **i** armazena e utiliza a variável maiores**[i]** para armazenar o maior elemento de cada subvetor. Ao final da execução, o vetor maiores**[i]** é percorrido sequencialmente para se descobrir o maior valor entre todos.

Mapeamento

As *threads* são escalonadas estaticamente, através de um ***for***. Cada *thread* percorre toda a região de código paralelo, mas a região do subvetor que ela deve percorrer está no escopo de um ***if*** que compara se o *id* da *thread* é o responsável por executar a i-ésima região de código.