SSC0903 - Computação de Alto Desempenho
Docente: Paulo Sérgio Lopes de Souza
Aluno PAE: Thiago Durães

Trabalho 2 - Descrição do modelo PCAM para o Problema do Caixeiro Viajante

Grupo responsável

Carlos André Martins Neves	NºUSP.: 8955195
Felipe de Oliveira	NºUSP.: 10284970
Ivan Costa de Souza	NºUSP.: 9292754
Jonathan Ferreira de Mello	NºUSP.: 10377754
Kevin Felipe Kühl Oliveira	NºUSP.: 10309715

São Carlos, Junho de 2020

PCAM

O problema do caixeiro viajante possui um grafo de cidades. A busca nesse grafo que enumera todas as soluções possíveis geram uma estrutura de árvore. Nas folhas dessa árvore temos todos os circuitos possíveis que o caixeiro viajante pode fazer. Cada uma dessas folhas possui um custo associado. O circuito de menor custo é a solução ótima.

Particionamento

O problema possui N cidades e inicialmente a única “tarefa” é nó raiz da árvore. A estratégia de particionamento do problema é expandir em largura essa árvore de busca o máximo possível; a granularidade mais fina seria expandir essa árvore até enumerar todas as folhas. O cálculo do custo do circuito de cada folha seria a carga de trabalho mais fina que pode ser atingida. Dessa forma, (N-1)! folhas são geradas.

Comunicação

A comunicação pode ser implementada em dois níveis, uma primeira comunicação global, usando memória distribuída e passagem de mensagem; e uma segunda comunicação usando memória compartilhada.

Na primeira comunicação (memória distribuída), utilizando a arquitetura mestre-escravo, existem 3 comunicações chaves para a implementação do algoritmo. Primeiro, um broadcast da matriz custo do processo mestre para todos os outros processos. Segundo, uma alocação e distribuição das (N-1)! tarefas para os P processos. Por fim, a terceira e última passagem de mensagem quando os P processos terminarem o seu processamento e encontrarem seu ótimo local; Os P ótimos locais são agregados no processo mestre via passagem de mensagem.

Na segunda comunicação (memória compartilhada), as T tarefas que cada processo recebe podem ser particionadas expandindo esses nós em largura e salvando eles em espaços separados da memória; cada espaço de memória pode ser reservado para cada núcleo de processamento sem condições de disputa.

Aglomeração

A carga de trabalho pode ser aglomerada em granularidades maiores no processo master, i.e., não é necessário que o mestre faça a expansão completa da árvore de busca. A expansão que o mestre faz pode gerar tarefas de diferentes tamanhos. A carga de trabalho de uma tarefa está associada com sua profundidade na árvore de busca; quanto menor a profundidade, maior a carga de trabalho. Veja como exemplo o nó raiz, de profundidade 0 e possui a maior carga de trabalho.

Além da expansão da árvore de busca, há um pedaço de memória em que cada linha de execução armazena o seu ótimo local. Há o problema de agrupar esses ótimos locais e achar o melhor entre eles. Esse passo precisa que as linhas de execução acabem toda sua carga de trabalho e armazenem o ótimo local encontrado. Subsequentemente, em uma única linha de execução, é eleito o melhor desses ótimos.

Num cluster homogêneo, uma boa estratégia é expandir o problema raiz em T tarefas de mesma carga e distribuir essas T tarefas da forma mais uniforme possível entre os P processos (um algoritmo de alocação como o round-robin é suficiente para fazer essa distribuição).

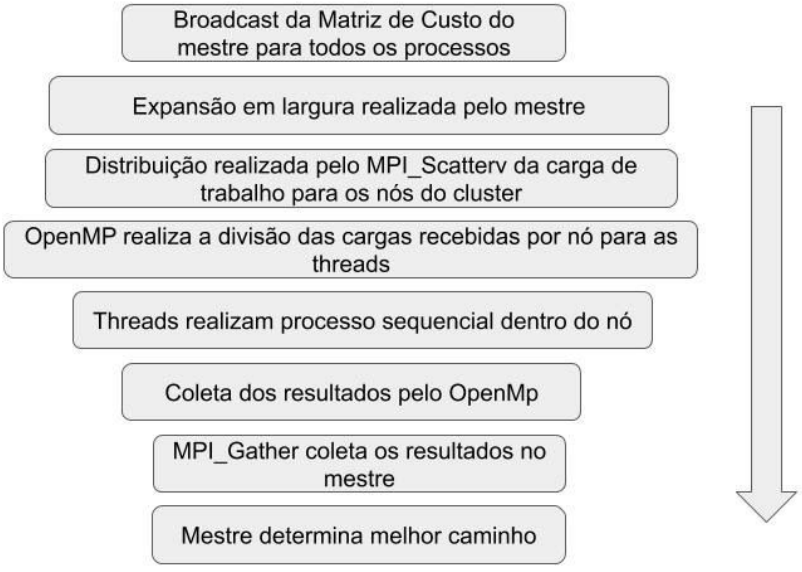
Num cluster heterogêneo, a expansão do problema inicial em T tarefas é mais complicada, uma vez que os P processos possuem diferentes poder de computação. Dessa forma a expansão do problema inicial deve gerar várias tarefas de tamanhos variados e essas tarefas devem ser distribuídas levando em consideração sua carga e o poder de processamento do processo de destino.

Mapeamento

O cluster que computa o problema possui característica homogênea, pois todos os nós possuem as mesma características de hardware. O mapeamento é feito para P==13 processos, no qual um processo é atribuído a cada elemento de processamento multicore(o cluster possui 13 nós com 3.6GHz e 8 núcleos); cada processo atribui uma thread para cada um de seus 8 núcleos.

Se o cluster fosse heterogêneo o mesmo mapeamento de processos e threads pode ser feito, porém é necessário que se faça um escalonamento dinâmico de novas cargas de trabalho (possivelmente de tamanhos variados) para os elementos que vão ficando ociosos.

Overview da lógica de implementação



Fonte: Imagem produzida pelo grupo