UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

PROCESSAMENTO DISTRIBUÍDO USANDO MPI

Sadi Júnior Domingos Jacinto

Professor orientador: Odorico Machado Mendizabal

Florianópolis
2019

Sadi Júnior Domingos Jacinto

PROCESSAMENTO DISTRIBUÍDO USANDO MPI

Análise do uso de processamento distribuído utilizando MPI, requerido pelo professor da disciplina Programação Paralela e Distribuída, Odorico Machado Mendizabal, necessário para obtenção de nota.

Professor orientador: Odorico Machado Mendizabal

Florianópolis

2019

Conteúdo

1 RESUMO

Muitos problemas interessantes de otimização não podem ser resolvidos de forma exata, utilizando a computação convencional (sequencial) dentro de um tempo razoável, inviabilizando sua utilização em muitas aplicações reais.

Embora os computadores estejam cada vez mais velozes, existem limites físicos e a velocidade dos circuitos não pode continuar melhorando indefinidamente. Por outro lado nos últimos anos tem-se observado uma crescente aceitação e uso de implementações paralelas nas aplicações de alto desempenho como também nas de propósito geral, motivados pelo surgimento de novas arquiteturas que integram dezenas de processadores rápidos e de baixo custo.

Dito isso, o presente relatório tem por objetivo analisar o desempenho de uma aplicação de ordenação de vetores de inteiros em sua implementação paralela, utilizando o MPI para troca de mensagens entre diferentes processos.

2 DEFINIÇÃO DO ESTUDO

O presente relatório buscará analisar o desempenho de uma aplicação responsável por ordenar um vetor de inteiros, cujo tamanho é definido pelo usuário e cujos valores são gerados de forma aleatória, em sua implementação paralela em comparação com sua implementação sequencial.

Para isso, os seguintes critérios foram seguidos:

- A aplicação foi inteiramente escrita na linguagem C.
- Foi utilizada a biblioteca OpenMPI para realizar a comunicação entre os diferentes processos.
- O algoritmo de ordenação usado foi o *bucket sort*, para a divisão dos dados, e o *quick sort* para a real ordenação dos dados.
- Tanto a versão paralela quanto a versão sequencial ordenam o mesmo conjunto de dados (copiados para ambas as versões) e utilizam o mesmo número de *buckets*, com o objetivo de os resultados não serem prejudicados por diferenças nas implementações[1].
- Foram escritos duas aplicação para realizar essa análise, uma delas dando prioridade para o processamento dos dados e outra priorizando a otimização de memória.

Tais critérios serão melhor explicados ao longo do relatório.

[1] afora o paralelismo, é claro.

3 MPI (*MESSAGE PASSING INTERFACE*)

O MPI é um padrão de interface para a troca de mensagens em máquinas paralelas com memória distribuída. Apesar de alguns pensarem dessa forma, o MPI não é um compilador ou um produto específico.

No padrão MPI, uma aplicação é constituída por um ou mais processos que se comunicam, acionando-se funções para o envio e recebimento de mensagens entre os processos. Inicialmente, na maioria das implementações, um conjunto fixo de processos é criado. Porém, esses processos podem executar diferentes programas. Por isso, o padrão MPI é algumas vezes referido como MPMD (*multiple program multiple data*).

Elementos importantes em implementações paralelas são a comunicação de dados entre processos paralelos e o balanceamento da carga. Dado o fato do número de processos no MPI ser normalmente fixo, neste texto é enfocado o mecanismo usado para comunicação de dados entre processos. Os processos podem usar mecanismos de comunicação ponto a ponto (operações para enviar mensagens de um determinado processo a outro). Um grupo de processos pode invocar operações coletivas (collective) de comunicação para executar operações globais. O MPI é capaz de suportar comunicação assíncrona e programação modular, através de mecanismos de comunicadores (communicator) que permitem ao usuário MPI definir módulos que encapsulem estruturas de comunicação interna.

Os algoritmos que criam um processo para cada processador podem ser implementados, diretamente, utilizando-se comunicação ponto a ponto ou coletivas. Os algoritmos que implementam a criação de tarefas dinâmicas ou que garantem a execução concorrente de muitas tarefas, num único processador, precisam de um refinamento nas implementações com o MPI.

4 BUCKET SORT

5 SOBRE A IMPLEMENTAÇÃO

6 RESULTADOS

7 CONCLUSÕES