Mestrado de Eng.ª Informática e Sistemas de Informação

Cadeira de Técnicas de Computação Paralela

Trabalhos de Avaliação

1. Introdução

Estes enunciados constituem os temas de avaliação da Cadeira de **Técnicas de Computação Paralela** do 2º Semestre do Mestrado de Engenharia Informática e Sistemas de Informação.

Cada trabalho deverá ser realizado por um grupo de no máximo dois alunos. O trabalho deverá ser uma realização original dos membros do grupo, devendo na sua discussão cada aluno responder individualmente a questões sobre a realização do mesmo.

Uma das condições para a aceitação do trabalho para efeitos de avaliação é o respectivo programa não apresentar erros de compilação, implementar a funcionalidade descrita no enunciado escolhido e ser acompanhado de um relatório de execução devidamente identificado.

São apresentados de seguida 3 temas dos quais 2 (à escolha) deverão ser obrigatoriamente realizados pelos alunos.

A entrega dos trabalhos deverá ser realizada até **30 de Junho de 2008**, devendo a defesa do trabalho decorrer durante o mês de Julho. A demonstração das funcionalidades implementadas deverá ser realizada nas instalações da Universidade Lusófona.

2. Programação Paralela em plataforma de Memória Partilhada

2.1 Objectivo

Pretende-se com este trabalho aplicar as técnicas de Computação Paralela apresentadas nas aulas teóricas a plataformas de memória partilhada (*threads*).

Assim, o trabalho deverá efectuar a paralelização de um algoritmo série conhecido ou desenvolvido pelos alunos, utilizando uma ou várias técnicas de decomposição referidas. Para além da implementação do programa paralelo, o trabalho deverá mencionar a metodologia seguida, o grafo de dependências do algoritmo, assim como o respectivo caminho crítico.

Uma previsão dos ganhos de desempenho deverá ser igualmente apresentada.

Na medida do possível, o trabalho deverá realizar uma breve análise dos ganhos de desempenho obtidos, embora seja reconhecido não haver possibilidade de acesso a hardware com as características necessárias para tirar partido de graus de paralelismo máximo superior a 2.

2.2 Indicações para a realização do trabalho

O trabalho deverá constar das fases seguintes:

1. Escolha e programação do algoritmo sequencial
2. Determinação do grau de paralelismo utilizando as técnicas de decomposição indicada nas aulas
3. Mapeamento em tarefas e desenho do respectivo grafo de dependência
4. Mapeamento das tarefas em *threads* analisando os aspectos de comunicação de dados e agregação dos resultados finais
5. Programação do algoritmo paralelo utilizando *multithreading*
6. Avaliação possível dos ganhos de desempenho

Aconselham-se os alunos a desenvolver o programa em ambiente Linux e linguagem C, utilizando as APIs Posix Threads ou OpenMP, embora outras linguagens possam ser utilizadas (Java, C#, etc…).

Algoritmos sugeridos:

- Multiplicação de matrizes com dimensão mínima de 1.000x1.000.
- Ordenação de um conjunto de números gerados aleatoriamente com uma dimensão mínima de 1.000.000 elementos.
- Contagem das ocorrências de palavras de um texto em ASCII com pelo menos 100.000 palavras.
- Outros que permitam uma paralelização efectiva da computação

3. Programação Paralela em plataforma de *Message Passing*

3.1 Objectivo

Pretende-se com este trabalho aplicar as técnicas de Computação Paralela, apresentadas nas aulas teóricas, a plataformas de troca de mensagens.

Assim, o trabalho deverá efectuar a paralelização de um algoritmo série conhecido ou desenvolvido pelos alunos, utilizando uma ou várias técnicas de decomposição referidas. Para além da implementação do programa paralelo, o trabalho deverá mencionar a metodologia seguida, o grafo de dependências do algoritmo, assim como o respectivo caminho crítico.

Uma previsão dos ganhos de desempenho deverá ser igualmente apresentada.

O trabalho deverá realizar uma análise dos ganhos de desempenho obtidos, em função do número de nós utilizado na execução do algoritmo.

3.2 Indicações para a realização do trabalho

O trabalho deverá constar das fases seguintes:

1. Estudo e instalação de uma plataforma de programação baseada em mensagens, (MPI / MPICH2 aconselhado) num conjunto de máquinas a escolher pelos alunos.

2. Escolha e programação do algoritmo sequencial

3. Determinação do grau de paralelismo utilizando as técnicas de decomposição indicada nas aulas

4. Mapeamento em tarefas e desenho do respectivo grafo de dependência

5. Mapeamento das tarefas em jobs MPI, analisando os aspectos de comunicação de dados e agregação dos resultados finais

6. Programação do algoritmo paralelo utilizando a interface de programação MPI

7. Avaliação dos ganhos de desempenho em função do número de nós utilizados

Este trabalho implica a instalação de um ambiente de execução baseado em mensagens, que será em principio o MPICH2. Para a realização desta tarefa serão dadas instruções nas aulas, mas os alunos deverão realizar uma parte significativa do trabalho de forma autónoma.

Existem várias implementações do standard MPI, mas uma das versões mais documentadas e actual é sem dúvida a que é distribuída pelo **Argonne National Laboratory**, cujo site é: http://www.mcs.anl.gov/research/projects/mpich2/index.php

Existem também várias versões de MPICH2 para inúmeras plataformas, inclusive para Windows, mas aconselha-se a utilização da versão genérica para "*commodity clusters*" baseados em máquinas Linux por ser a mais bem documentada e estável.

A instalação do ambiente de execução pode ser realizado num conjunto de máquinas do laboratório (ver enunciado seguinte), ou nos computadores pessoais dos alunos, de forma nativa ou utilizando máquinas virtuais. Neste ultimo caso um conjunto de precauções adicionais deverá ser tomado, nomeadamente no que diz respeito ao endereçamento IP das máquinas virtuais.

Os algoritmos a programar poderão ser os mesmos do enunciado 1, tendo em conta o facto de que o paradigma de programação é completamente diferente, sobretudo no método de acesso aos dados.

4. Instalação e configuração de um Cluster

4.1 Objectivo

Pretende-se com este trabalho realizar a configuração e instalação de um *cluster* de 6 máquinas de tipo PC, disponíveis no Laboratório.

O resultado deste trabalho deverá ser uma plataforma hardware e software de execução

paralelo de tarefas, que possa eventualmente servir de suporte à execução de trabalhos do enunciado anterior.

Dadas as suas características abrangentes, este trabalho poderá ser realizado por dois grupos (no máximo 4 alunos) que deverão partilhar entre si as tarefas abaixo descritas.

4.2 Indicações para implementação

Este trabalho deverá constar das fases seguintes:

1. Análise do hardware existente e verificação do seu estado de funcionamento
2. Configuração do hardware numa rede fisicamente separada, com suporte eventual de um esquema de endereçamento próprio
3. Escolha de um sistema operativo de base, que à partida deverá ser de tipo Linux
4. Escolha de uma plataforma de Clustering adequada ao hardware disponível
5. Instalação do Sistema Operativo, com as funcionalidades adequadas ao tipo de cluster pretendido, com eventual partilha de recursos de storage por NFS.
6. Instalação da plataforma de Clustering e sua validação funcional
7. Utilização de alguns exemplos de trabalhos realizados para validação do desempenho do sistema

Possíveis plataformas de Clustering para instalação:

- **Torque/Open PBS**: plataforma de gestão de recursos, de submissão e escalonamento de jobs em batch, baseada em mecanismos de tipo Unix. É actualmente distribuído pela Cluster Resources mas existem outras versões open source. (http://www.clusterresources.com/pages/products/torque-resource-manager.php
- **Sun Grid Engine**: plataforma de gestão de recursos, submissão e escalonamento de jobs, baseada em Java, necessita de uma instalação completa do Java da SUN (http://gridengine.sunsource.net).
- **Condor**: é uma plataforma extremamente evoluída, fornecendo simultaneamente as funcionalidades das anteriores para além de uma interface de programação de tipo MPI. Permite também o aproveitamento de ciclos de CPU de workstations de utilizadores num ambiente distribuído. (http://www.cs.wisc.edu/condor)
- **MPICH2**: como se viu, é essencialmente uma plataforma de programação e execução baseada em mensagens, suportada por algumas das plataformas anteriores (com algumas restrições). Este ambiente também pode ser instalado em modo nativo (http://www.mcs.anl.gov/research/projects/mpich2/index.php).