Provinhas - Programação Concorrente
Turma de Terça-Feira

1 - Diferença entre computação paralela e distribuída.

RESPOSTA : computação paralela se refere a programs executando de forma paralela, ou seja, várias instruções no mesmo instante de tempo. Essas instruções podem ser executadas em paralelo em uma plataforma com memória compartilhada (UMA - *Uniform Memory Acess*) ou distribuída (NUMA - *NonUniform Memory Acess*). Computação distribuída refere-se a programas distribuídos, ou seja essencialmente plataformas NUMA sob alguma rede de interconexão baseada em pilhas de protocolos. A

2 - Diferença entre máquinas SIMD e Clusters.

RESPOSTA : ver provinhas TURMA 1. SIMD (*Single Instruction Multiple Data*).

3 - Defina o conceito de Granulação (Granularidade).

RESPOSTA : granularidade de uma aplicação paralela é a relação entre o número de tarefas nas quais a aplicação pode ser decomposta e a quantidade de trabalho associada a cada uma dessas tarefas. Traduzindo para o contexto de plataformas de programação e exemplificando, uma aplicação com *granularidade fina* é constituída de muitos processos executando trechos reduzidos de código em paralelo, enquanto uma aplicação com *granularidade grossa* consiste em um número reduzido de processos em paralelo com uma maior carga de trabalho associada a cada um. Aplicações finamente granuladas podem ser mapeadas mais facilmente em plataformas robustas, onde há muitos processadores disponíveis. Contudo, o custo de interação entre processos (comunicação e sincronização) pode acabar se tornando um gargalo na escalabilidade da aplicação. Por outro lado, aplicações pouco granuladas tendem a sofrer mais com o desbalanceamento de carga entre processos e mau uso de recursos disponíveis, fatores que também comprometem o desempenho. Buscar o equilíbrio entre esses contrapontos constitui um dos principais desafios no desenvolvimento de aplicações paralelas.

4 - Qual a diferença entre interação Rendezvous, Rendezvous Estendido e Coletiva.

RESPOSTA :

5 - O paradigma de programação mestre-escravo pode ser também SPMD ou MPMD? Justifique.

RESPOSTA : Sim. Para o paradigma ser utilizado como SPMD (*Single Program Multiple Data*), pode-se por exemplo colocar cada escravo como uma cópia do mestre, com cada processo executando sobre um bloco de dados e sincronizando o trabalho para o mestre através de alguma interação de redução. Para o paradigma ser utilizado como MPMD (*Multiple Program Multiple Data*), basta que escravos e mestre possuam alguma espécie de distinção de funcionalidade e operem sobre dados distintos.

6 - Faça o código usando FORK/JOIN dado o grafo de dependências.

RESPOSTA : Obter grafo dessa turma. Ver grafo da TURMA 1.

7 - Explique *down(&sem)* e a principal característica que garante sincronização entre processos com semáforos.

RESPOSTA : *down(&sem)* é uma operação para decrementar o valor de uma variável semáforo. A principal característica de semáforos e suas operações down e up é a atomicidade de execução, de maneira que somente um processo terá acessado ao recurso protegido pelo semáforo por vez.