Provinhas (Programação Concorrente)
Turma de Segunda-feira

1 - Diferenciar Programação Paralela de Programação Concorrente.

RESPOSTA : Uma aplicação é concorrente quando seus processos associados - sendo leves ou não - estão executando apenas um a cada vez em um dado instante de tempo (conceito de pseudo-paralelismo relacionamento ao chaveamento desses processos no tempo). Já uma aplicação paralela possui diversos processos iniciados executando em um mesmo instante de tempo. Esses processos poderão ser concorrentes entre si, de acordo com o tipo de mapeamento entre processos e processadores adotado em função de plataforma de programação disponível, segundo alguma política de balanceamento de carga. Dessa forma, todo programa paralelo pode ser inerentemente concorrente.

2 - Diferenciar máquinas SMP (ou UMA) de um Cluster.

RESPOSTA : SMP são plataformas computacionais multiprocessadas, baseadas em barramento e de um único sistema de memória principal centralizado e acessível de maneira igual por cada processador, o que origina o termo UMA (*Uniform Memory Acess*). Já clusters por sua vez são plataformas compostas por nós, com memória distribuída, elementos processadores não necessariamente homogêneos e comunicação entre processadores via E/S (rede), o que o caracteriza uma plataforma NUMA (*Non-Uniform Memory Acess*).

3 - Diferenciar Programação Sequencial de Paralela.

RESPOSTA : em um programa executado de forma sequencial, apenas uma dada instrução de código é executada a cada instante de tempo. Já em um programa executado em paralelo, múltiplas instruções de código podem ser executadas simultaneamente, de acordo com a plataforma de programação disponível.

4 - Definir o paradigma de programação MPMD.

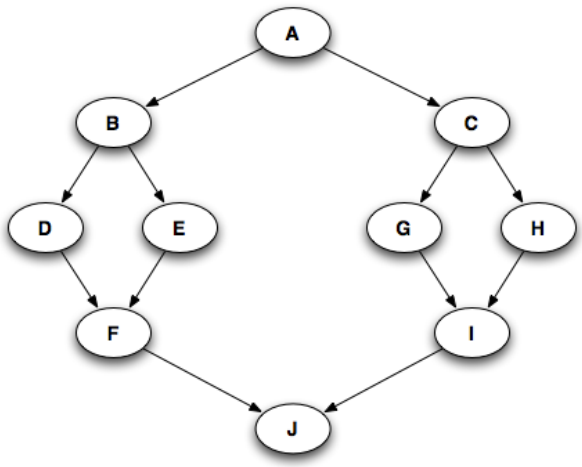
RESPOSTA : MPMD (*Multiple Program Multiple Data*) se refere ao paradigma que coloca processos heterôgeneos para executar sob dados distintos de uma dada aplicação.

5 - Diferenciar Speedup Linear e Superlinear.

RESPOSTA : um Speedup Linear é caracterizado quando o desempenho aferido de uma dada aplicação aumenta proporcialmente (linearmente) com o número de processadores que possam estar disponíveis para a execução da mesma. Já o Speedup Superlinear é caracterizado por um desempenho aferido superior ao fator de uma proporção simples, por exemplo uma relação quadrática ou exponencial. Tais índices de desempenho em geral são dependentes de melhorias consideráveis em nível de hardware na plataforma de programação, como aprimoramento (ou eventual adoção) de memórias cache e um ou mais níveis, paralelização de E/S, dentre outros fatores.

6 - Montar pseudocódigo com FORK/JOIN a partir do Grafo de Dependências.

RESPOSTA :




```
        NI = 2; NF = 2; NJ = 2;

        FORK b;
        C;
        FORK h;
        G;
        JOIN i, NI, QUIT;

h:      H;
        JOIN i, NI, QUIT;

i:      I;
        JOIN j, NJ, QUIT;

b:      B;
        FORK d;
        E;
        JOIN f, NF, QUIT;

d:      D
        JOIN f, NF, QUIT;

f:      F;
        JOIN j, NJ, QUIT;

j:      J;
```

7 - Diferenciar técnicas de Sincronização entre Processos : Sleep/Wakeup e Semáforos.

RESPOSTA : Uma sincronização entre processos do tipo Sleep/Wakeup ocorre quando existem chamadas de sistema que permitem que interrupções de software (traps) possam ser direcionadas de um processo a outro, alterando o estado do processo alvo para o desejado. Dessa maneira, a sincronização ocorre com os processos hora sendo forçados a 'dormir' (ou seja, bloqueando) ora sendo notificados a 'acordar' (ou seja, sendo colocados na fila de processos prontos para executar), de acordo com a lógica da aplicação. O problema com tal técnica de sincronização para dadas aplicações é o fato de o processo poder ser escalonado no exato instante em que o sinal de sleep/wakeup seria enviado, de maneira que o processo alvo é considerado pelo emissor da trap logicamente como dormindo ou acordado, quando efetivamente ele não está no estado desejado. Isso ocorre porquê o escalonamento de processos não garante a entrega confiável de sinais, ou seja, o sinal enviado acaba se perdendo no chaveamento entre processos.

Esse problema não ocorre com semáforos, que são estruturas de dados nativas do sistema operacional e que, em geral com auxílio de instruções de hardware, são acessadas e manipuladas de maneira atômica pelos processos associados, de maneira que a sincronização ocorre sem falhas. Um semáforo é normalmente utilizado para controlar o acesso a uma região de memória sob disputa de vários processos, podendo ser tipo tipo semáforo contador (*counting*) ou binário (*lock/unlock*). De acordo com o valor atual do semáforo, o acesso ao recurso é liberado ou não a cada processo ao longo do tempo. Semáforos em geral possuem pelo menos duas operações associadas : *up(&sem)* e *down (&sem)*, executadas pelos processos em condição de disputa e que alteram o valor da variável semáforo de acordo com a lógica da aplicação. Adicionalmente, Mutexes são um tipo especial de semáforo binário, com algumas características adicionais.

8 - Explicar função do POSIX *pthread_cond_wait(cond,mutex).*

RESPOSTA : a função *pthread_cond_wait(pthread_cond_t* ****cond****, pthread_mutex_t* ****mutex****)* faz com que a thread corrente tente ganhar o controle do *mutex* indicado. Esse está inicialmente bloqueado, e será desbloqueado para a thread chamadora, que irá bloquear segundo a variável de condição *cond*. Após o retorno com sucesso, o controle da variável mutex é da thread corrente. *(Retirado do website do POSIX)*.