1. **Qual a importância da computação paralela nos dias atuais?**

É importante pois possibilita melhora no desempenho e economia de tempo, além do que várias tarefas feitas na internet precisam ser feitas de forma simultânea em diferentes dispositivos, assim a computação paralela é a melhor alternativa.

1. **Qual a diferença entre paralelismo de dados e de tarefas?**

No paralelismo de dados, os dados, utilizados na solução do problema entre os núcleos, são divididos e cada core realiza operações em cada parte dos dados. Já no paralelismo de tarefas, as tarefas são divididas e os problemas são resolvidos entre os cores.

1. **Qual a diferença entre memória compartilhada e distribuída?**

Na computação paralela um programa é formado por tarefas onde essas obrigatoriamente cooperam para resolver os problemas. Na computação distribuída um programa pode cooperar com outros para resolver o problema, não sendo uma ação obrigatória.

1. **O que é o princípio da localidade? Como isso ocorre nos programas e nos dados?**

É quando o acesso a um local é seguido por um acesso a um local próximo. Os programas e dados utilizados por um processo de computação tendem a estar concentrados em uma região relativamente pequena da memória e a serem acessados repetidamente a partir dessa região.

1. **O que é mapeamento de cache e quais os tipos existentes?**

O mapeamento de cache é uma técnica de gerenciamento de memória cache que determina como os blocos de dados da memória principal (RAM) são associados e armazenados na memória cache. Existem 3 tipos de mapeamentos:

*- Mapeamento associativo total:* que é quando uma nova linha pode ser colocada em qualquer lugar da cache;

- *Mapeamento direto:* quando cada linha da cache possui um local exclusivo na cache a qual será atribuído;

- *Mapeamento associativo por conjunto(N-way Set Associative):* cada linha de cache pode ser colocada em um dos n locais diferentes da cache.

1. **Como funciona a memória virtual e o que é cache TLB?**

A memória virtual funciona como cache para armazenamento secundário, ela explora o princípio da localidade espacial e temporal, e mantém apenas as partes ativas dos programas em execução na memória principal. O TLB é uma cache específica usada para acelerar o processo de tradução de endereços virtuais para endereços físicos. Em sistemas de memória virtual, os programas em execução geram endereços virtuais que precisam ser mapeados para endereços físicos na memória principal. O TLB armazena pares de endereços virtuais e físicos previamente traduzidos, permitindo que o sistema encontre rapidamente a correspondência correta e evite atrasos associados à tradução de endereços em tempo real.

1. **A granularidade relaciona o tamanho das unidades de trabalho enviadas aos processadores e pode ser dividida em três níveis. Cite e descreva estes níveis.**

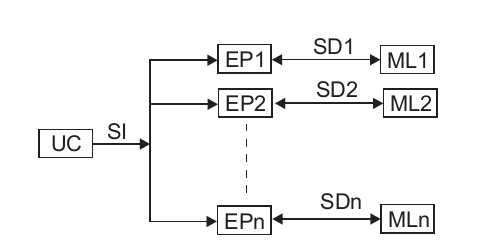
- *Granularidade Fina:* paralelismo de baixo nível; nível de instruções ou operações; grande número de processos pequenos e simples; fácil de conseguir balanceamento de carga eficiente; o tempo de computação nem sempre compensa os custos de criação, comunicação e sincronismo; difícil conseguir aumentar o desempenho;

- *Granularidade média;* paralelismo de nível médio; algo entre nível de instruções ou operações e nível de programas; dezenas de processos de tamanho médio;

- *Granularidade Grossa:* paralelismo de alto nível; nível de programas; geralmente observado em plataformas com poucos processadores, grandes e complexos; o tempo de computação compensa os custos de criação, comunicação e sincronização; oportunidade de conseguir melhorias significativas no desempenho; difícil de conseguir balanceamento de carga eficiente.

1. **(Petrobrás – 2010) A taxonomia de Flynn é utilizada para classificar sistemas de processamento paralelo.**

A figura abaixo apresenta uma das categorias definidas por Flynn:



UC: unidade de controle

SI: sequência de instruções

EP: elementos de processamento

SD: sequência de dados

ML: memória local

Qual das arquiteturas a seguir pertence à categoria mostrada acima?

a) ( ) SMP.

b) ( ) NUMA.

c) ( ) Cluster.

d) ( ) Uniprocessador.

e) (x) Processadores vetoriais.

1. **(POSCOMP - 2011) O gerenciamento de processos em sistemas modernos é feito, quase sempre, com o uso de preempção de processos através de técnicas de compartilhamento de tempo. O que a introdução de processadores com vários núcleos altera nesse gerenciamento?**

a) ( ) Torna-se possível eliminar a condição de corrida em processos concorrentes executados em paralelo.

b) ( ) Torna-se possível o uso de sistemas operacionais multitarefas.

c) ( ) Torna-se possível o uso de threads para a execução de processos concorrentes.

d) ( ) Torna-se possível separar os demais mecanismos de gerenciamento do sistema operacional do gerenciamento de processos.

e) (x) Torna-se possível a paralelização efetiva de processos concorrentes.

1. **Considerando a taxonomia proposta por Duncan, responda:**
2. **Qual a diferença entre as arquiteturas síncronas e assíncronas?**

- Arquitetura Síncrona: Nesse tipo de arquitetura, as operações são executadas em sincronia, seguindo um relógio central. Todas as unidades de processamento executam instruções ao mesmo tempo, avançando de uma etapa para a próxima de acordo com um relógio global.

- Arquitetura Assíncrona: Em uma arquitetura assíncrona, as operações não são coordenadas por um relógio global. Em vez disso, cada unidade de processamento opera de forma independente e se comunica com outras unidades apenas quando necessário. Não há um ritmo de operação uniforme para todas as unidades.

**b. A classificação proposta por Duncan resolve alguns problemas apresentados na classificação proposta por Flynn. Quais são esses problemas?**

- A classificação de Flynn categoriza os sistemas de acordo com a quantidade de fluxos de dados e controle, mas não lida diretamente com questões de coordenação e sincronização entre esses fluxos.

- A taxonomia de Duncan estende a classificação de Flynn ao incluir as arquiteturas síncronas e assíncronas, abordando a questão da sincronização e coordenação entre as unidades de processamento.

- Isso é importante porque em sistemas assíncronos, não há um controle centralizado ou relógio global, o que pode ser útil para resolver problemas de escalabilidade, tolerância a falhas e economia de energia, por exemplo.

- Portanto, a taxonomia de Duncan aprimora a classificação de Flynn, tornando-a mais abrangente e relevante para sistemas de computação modernos, especialmente aqueles com arquiteturas altamente paralelas e heterogêneas.

Em resumo, a classificação de Duncan resolve problemas relacionados à falta de consideração de coordenação e sincronização entre unidades de processamento na classificação de Flynn, adicionando a distinção entre arquiteturas síncronas e assíncronas, o que é crucial em sistemas paralelos modernos.

1. **(INPE - 2014) Suponha um determinado problema científico que possua solução tanto por programação sequencial como por programação paralela. Identifique a melhor resposta considerando as afirmações abaixo:**

I. É possível estabelecer que um programa sequencial é mais eficiente que sua versão paralela, uma vez que é mais fácil programá-lo.

II. A comunicação e a sincronização entre diferentes sub tarefas é tipicamente uma das maiores barreiras para atingir grande desempenho em programas paralelos.

III. A lei de Amdahl afirma que uma pequena porção do programa que não pode ser paralelizada limitará o aumento de velocidade geral disponível com o paralelismo.

a) ( ) Todas as afirmações estão corretas.

b) (x) As afirmações II e III são as únicas corretas.

c) ( ) A afirmação I é a única correta.

d) ( ) As afirmações I e III são as únicas corretas.

e) ( ) Todas as afirmações estão erradas.

1. **(INPE - 2014) Um código que possua “paralelismo implícito”, refere-se a (selecione a resposta que melhor responda à definição):**

a) ( ) O paralelismo vem do fabricante.

b) ( ) Atribuição de que o código em compilação não pode ser distribuído.

c) ( ) A compilação força o uso de memória compartilhada.

d) ( ) A distribuição pelos nós computacionais exigirá mais memória.

e) (x) Cabe ao compilador e ao sistema de execução detectar o paralelismo potencial do programa.

1. **(Petrobrás – 2010) Para as 5 regras básicas de mapeamento de processos abaixo, indique como deve acontecer para um sistema paralelo:**

**a) Atribuição de tarefas:** No contexto de sistemas paralelos, a atribuição de tarefas refere-se à distribuição de tarefas ou computações para diferentes processadores ou núcleos de CPU. Isso é feito para que várias tarefas possam ser executadas em paralelo, aproveitando os recursos de processamento disponíveis.

**b) Repartição do trabalho:** A repartição do trabalho envolve dividir uma tarefa ou computação em partes menores que podem ser distribuídas entre os processadores. Essa divisão do trabalho é fundamental para aproveitar ao máximo o paralelismo disponível em um sistema paralelo, permitindo que várias partes do trabalho sejam executadas simultaneamente.

**c) Interação inter processos:** Em sistemas paralelos, as interações entre processos (ou threads) podem ser necessárias quando partes diferentes do trabalho precisam compartilhar dados ou coordenar a execução. O gerenciamento adequado das interações entre processos é importante para evitar problemas de concorrência, como condições de corrida.

**d) Tarefas com interações elevadas:** Tarefas com interações elevadas são aquelas em que os processos ou threads precisam se comunicar ou cooperar intensivamente. Isso pode ser mais desafiador em sistemas paralelos devido a possíveis problemas de sincronização e concorrência. É importante projetar cuidadosamente como essas tarefas serão executadas para garantir um desempenho eficiente.

**e) Tarefas do caminho crítico:** As tarefas do caminho crítico são aquelas que têm o maior impacto no tempo total de execução de um programa ou sistema. Em sistemas paralelos, é vantajoso dedicar recursos adicionais, se disponíveis, às tarefas do caminho crítico para acelerar a execução geral do programa.

1. **Para os valores de tempo de execução abaixo de um processo paralelizado, encontre os valores de speedup e eficiência:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1 CPU** | **2 CPUs** | **4 CPUs** | **8 CPUs** | **16 CPUs** |
| **T(p)** | 1000 | 510 | 265 | 160 | 100 |
| **S(p)** |  |  |  |  |  |
| **E(p)** |  |  |  |  |  |

Qual a “melhor” quantidade de processadores para executar esse processo paralelo? Por quê?

1. **Um programa gasta 4% do processamento com a parte sequencial (que não pode ser paralelizada).**
2. **Segundo a Lei de Amdhal, qual o Speedup esperado para um sistema com 8 processadores?**

Speedup Total= 1/(F + ((1−F)/P))) -> 1/(0.04 + ((1−0.04)/8))) -> 1/(0.04 + 0.12) -> 1/(0.16) -> 6.25

1. **E ainda, qual o Speedup máximo que pode ser obtido?**

Speedup Máximo= 1/(F) = 1/0.04 = 25

**c. Segundo a Lei de Gustafson, qual o Speedup esperado para um sistema com 8**

**processadores?**

Speedup Total = S(P) = P − α \* (P − 1) -> 8 − 0.04 \* (8 − 1) -> 8 - 0.04 \* 7 -> 8 - 0.28 -> 7.72

1. **Qual a diferença conceitual entre estas leis?**

Lei de Amdahl:

- A Lei de Amdahl se concentra nas limitações impostas pela fração sequencial ou não paralelizável de um programa ou algoritmo.

- Ela parte do pressuposto de que uma parte fixa do programa não pode ser paralelizada (F), e o desempenho é limitado por essa fração, mesmo quando se aumenta o número de processadores.

- A Lei de Amdahl enfatiza que, à medida que mais processadores são adicionados, o ganho de desempenho será cada vez mais limitado pela fração sequencial.

Lei de Gustafson:

- A Lei de Gustafson adota uma perspectiva diferente, focando na escalabilidade do tamanho do problema em sistemas paralelos.

- Ela sugere que, em sistemas paralelos, é possível aumentar o tamanho do problema (ou seja, a quantidade de trabalho) à medida que mais processadores estão disponíveis.

- A Lei de Gustafson argumenta que o ganho de desempenho pode ser obtido escalando o tamanho do problema, de modo que o tempo total de execução ainda seja reduzido, mesmo que uma parte fixa do programa não seja paralelizável.

Em resumo, enquanto a Lei de Amdahl está mais preocupada em identificar e limitar o impacto da fração sequencial de um programa, a Lei de Gustafson enfatiza a capacidade de aumentar a carga de trabalho para aproveitar melhor o paralelismo disponível. Ambas as leis são úteis para entender diferentes aspectos do desempenho em sistemas paralelos e podem ser aplicadas em contextos específicos, dependendo das características do problema e do sistema.

1. **Compare paralelismo implícito e explícito.**

Paralelismo implícito:

- Quando é responsabilidade do compilador e do sistema de execução: Detectar o paralelismo do programa; Atribuir as tarefas para execução paralela; Controlar e sincronizar toda execução.

- Vantagens: Mais geral e mais flexível; Programador não precisa se preocupar com detalhes da execução paralela.

- Desvantagens: Difícil chegar em uma solução eficiente em todos os casos.

Paralelismo explícito:

- Quando é responsabilidade do programador: Detectar o paralelismo do programa; Atribuir as tarefas aos processadores; Controlar a execução, indicando os pontos de sincronização; Conhecer a arquitetura do computador, de forma a otimizar o desempenho.

- Vantagem: Programadores eficientes produzem soluções muito eficientes.

- Desvantagens: Pouco portável em diferentes arquiteturas; Programador responsável por todos os detalhes da execução.

1. **Quais as técnicas de decomposição de tarefas e como elas funcionam?**

As técnicas são:

- Decomposição recursiva:

Utilizada em algoritmos que possam ter uma solução do tipo dividir e conquistar;

O problema inicial é decomposto em subproblemas de resolução independente, que tem tratamento idêntico ao inicial;

Essa decomposição é feita recursivamente até atingir a granularidade pretendida;

- Decomposição de dados:

A ideia é identificar a decomposição atráves dos dados que estão sendo computados;

Esses dados são divididos em subconjuntos.

Cada tarefa é responsável por processar um subconjunto de dados que lhe é atribuído;

Muito utilizada no tratamento de grandes quantidades de dados;

A forma como os dados são decompostos condiciona o desempenho da solução;

Exemplo: dados de entrada, dados de saída e dados intermediários.

- Decomposição exploratória:

Aplicada a problemas em que a decomposição está associada as fases de execução do algoritmo, e não ao conjunto de dados;

Problemas que envolvem a exploração de espaços de possíveis soluções;

Exemplos: Análise combinatória, problemas de otimização discreta, prova de teoremas, jogos e criptoanálise.

- Decomposição especulativa:

Em outras aplicações, as dependências entre tarefas não são conhecidas à priori.

Dependem de resultados intermediários.

Impossível identificar tarefas dependentes.

Podem ser realizadas duas aproximações:

1. Conservadora

Só considera tarefas independentes quando há garantia de não existir dependências;

Pode-se obter níveis de paralelismo reduzidos.

1. Otimista

Cria tarefas independentes mesmo quando não existe a garantia, confiando em mecanismos de avaliação posteriori das dependências.

Necessário mecanismos de controle e rollback, para desfazer algum trabalho pela antecipação.

1. **Indique e explique as 4 condições de exclusão mútua.**

1 Dois ou mais processos não podem estar simultaneamente na mesma região crítica.

2 Nenhum processo fora da região crítica pode bloquear a execução de outro processo.

3 Nenhum processo deve esperar infinitamente (starvation) para ter acesso a uma região crítica.

4 A solução não deve fazer considerações sobre o número de processadores nem das velocidades dos mesmos.

1. **Nos mecanismos de sincronização e comunicação, como funciona a troca de mensagens síncrona e assíncrona? Compare estes mecanismos.**

Comunicação Síncrona

O processo que transmite a mensagem bloqueado até que receba uma confirmação de recebimento da mensagem.

Comunicação Assíncrona

O processo transmissor envia a mensagem (que deve ser armazenada em um buffer) e continua sua execução.

1. **Como podemos utilizar paralelismo nas situações abaixo?**
2. Processamento de Imagens:

- Divisão de Imagens: Em processamento de imagens, uma imagem pode ser dividida em várias partes, e cada parte pode ser processada em paralelo. Isso é útil para aplicar filtros, transformações ou análises em imagens grandes.

- Filtragem Paralela: Em operações de filtragem, cada pixel da imagem pode ser processado independentemente, o que permite que vários pixels sejam processados em paralelo por diferentes núcleos ou processadores.

- Processamento de Vídeo: Em vídeo, cada quadro pode ser tratado como uma imagem separada, permitindo o paralelismo ao processar quadros em sequência.

b. Processamento de Banco de Dados:

- Paralelização de Consultas: Consultas SQL complexas podem ser divididas em subconsultas que podem ser executadas em paralelo por diferentes processadores ou threads. Isso acelera a recuperação de dados.

- Índices Paralelos: A construção e manutenção de índices em bancos de dados podem ser paralelizadas para melhorar o desempenho das operações de consulta.

- Transações Paralelas: Em sistemas de banco de dados com suporte a transações, várias transações podem ser processadas em paralelo, desde que não interfiram umas nas outras.

c. Processamento de Matrizes:

- Multiplicação de Matrizes: A multiplicação de matrizes é uma operação intensiva em termos computacionais e pode ser facilmente paralelizada, com diferentes partes da matriz sendo processadas em paralelo.

- Algoritmos de Álgebra Linear: Operações comuns de álgebra linear, como decomposições LU, QR e SVD, podem ser paralelizadas para acelerar o processamento.

- Processamento de Sinais e Imagens: Em campos como processamento de sinais e imagens, as operações matriciais podem ser aceleradas por meio do paralelismo para melhorar o desempenho em tarefas como filtragem e transformações.