CAD - Lista 03

Pedro Antonio Maciel Soraiva - 20230335B

① - SMP apresenta processadores idênticos com memória compartilhada; acesso uniforme a memória; sistema operacional único e apresenta uma escalabilidade limite pelo nº de processadores. Em comparação com um uniprocessador, o SMP apresenta maior desempenho, escalabilidade incremental, mais simples que usar um cluster, alta disponibilidade em caso de um processador falhar e menor latência de comunicação.

② - Coerência de cache é como se ~~(contenh)~~ mantém a consistente de dados em multiplos processadores que possuem copias de dados em caches locais. A diferenças de esquemas de hardware e software é no tempo de execução (hardware) e compilação (software); a abordagem de software é mais simples - evita dados compartilhados no cache, entretanto é menos eficiente no uso do cache. No esquema de hardware, há protocolos de diretório (controle de escrita e leitura), protocolos de monitoração (mantem a coerência de cache entre processadores), Protocolo MESI (Modified, Exclusive, Shared, Invalid)

③ Clusters são um sistema de computação baseado em nós independentes mas conectados por um middleware, seria como se nos juntassemos varias PCs em um so, pela rede. A vantagem é tolerancia a falhas, facil de escalar, balanceamento de carga e você consegue agregar recurso computacional que seria mal utilizado.

④ - UMA: Unified Memory Acess: Tempo de acesso uniforme como o nome diz, mais simples, menos escalavel, geralmente usado SMP.

NUMA (Non UMA): Tempo variado de acesso a memória depende da distância do processador, mais complexo, maior escalabilidade, geralmente usado em cluster

CC-NUMA (Cache Coherence NUMA): Protocolos de software e hardware garante coerência de informações, escalabilidade de desempenho em níveis alta de paralelismo

⑤ - SISD (Single Instruction, Single Data): Ele executa uma instrução em um dado; não apresenta paralelismo; usado em computadores de Von Neumann tradicionais.

SIMD (Single Instruction, Multiple Data): Um fluxo de instrução para multiplos fluxo de dados; apresenta paralelismo (como a gpu), ideal para operações singulares em conjunto grande, como a soma de constante em grande array.

MISD (Multiple Instruction, Single Data): Varias fluxo de instrução para um fluxo de dados; não apresenta aplicações praticos

MIMD (Multiple Instruction, Multiple Data): Varias fluxos de instruções e de dados; apresenta paralelismo; flexivel, usado em clusters, sistemas distribuidas e multiprocessadores.

⑥ - Memória compartilhada: programação intuitiva e facil de aplicar, com balanceamento dinâmico; problemas de escalabilidade limitada, problemas com sincronização e coerência de cache.

Memória distribuida: Mais complexo; mais escalavel, tolerante a falha e com maior custo beneficio; problema com

sobrecarga de comunicação e duplicação de dados.

(7): Particionamento: Dividimos o problema em tarefas menores e executáveis para paralelismo
- Comunicação: Quais tarefas comunicam e conectam entre si, para minimizar a latência
- Aglomeração: Agrupamos tarefas que comunicam em "clusters" para serem processadas, a ideia é equilibrar o paralelismo e os comunicações entre atividades
- Mapeamento: Atribuir os aglomerados para processadores, considerações as comunicação entre processadores e topologias das maquinas, garantido uso total do hardware.

(8) - O particionamento pode ser decomposto em funcional (tarefas) baseado nas tarefas que devem ser executadas e funções que devem ser coordenadas entre si, pode ser definido um pipeline ou fluxo lógico de como deve ser executada cada parte do problema. No paralelismo de dados, como deve ser executados e comunicados os dados em paralelismos massivos e escaláveis, como deve ser feita operações em cada dado e como serão sincronizadas no final
- Paralelismo de dados → divide os dados → quais devem ser processados e como → sincronização final como necessário
- Paralelismo de instrução → processo dividido em partes → como associados a cada instrução → coordenação da execução dessas partes

⑨ - Expressão: $(X/k - Y * j * g + Z * g * h)$

constantes: k, j, g e h

vetores: X, Y, Z (tamanho m)

- Paralelismo de tarefas

tarefa 1.1: calcula X/k

tarefa 1.2: calcula $Y \cdot j \cdot g$

tarefa 1.3: calcula $Z \cdot g \cdot h$

tarefa 2: soma os resultados das tarefas 1.1, 1.2 e 1.3

- Paralelismo de Dados:

1- Calcularmos duas novas constantes $a = j * g$ e $b = g * h$

Vamos considerar os vetores, então podemos fazer operações pelo indice, da seguinte maneira

$$Resultado_i = X[i]/k + Y[i] * a + Z[i] * b$$

Que ao ser executado, teriamos algo como

$$resultado = \sum_{i=0}^{m-1} X[i]/k + Y[i] * a + Z[i] * b$$

E para paralelizarmos, cada thread de uma gpu poderia executar uma operação com os elementos i. Se houver m-1 threads, haveria uma resolução quase instantânea.