| | | | |
|---|---|---|---|
| DISCIPLINA | **Computação Paralela e Distribuída** | | |
| CURSO | **Engenharia Informática** | | |
| DISCENTE | | | |
| Nº MATRÍCULA | | TURMA: | DATA: **Segunda-feira, 17/04/2023** |

- **Não é permitido o uso de dispositivos electrónicos**
- **Responda apenas o que é perguntado**
- **Justifique todas as suas respostas**

## Grupo I (2.0 + 2.0 = 4.0 valores)
## Arquitectura de Memória Partilhada

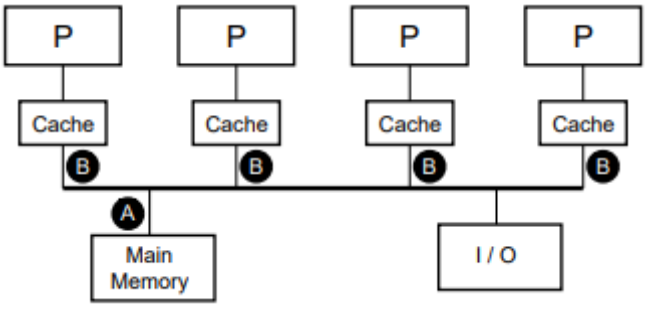
**1.** Considere um sistema de memória partilhada, como mostrado na figura abaixo, com um nível de cache privado para cada um dos p processadores. Para melhorar o desempenho do sistema, a adição de um segundo nível de cache está em estudo.

**Qual é o principal gargalho?**

**Discuta a vantagem relativa das duas opções a seguir:**

A. colocar uma única cache de tamanho C próximo à memória principal, portanto, partilhado por todos os processadores.

B. utilizar p caches de tamanho C/p, privados para cada processador, colocados após o cache de nível 1.




**2.** A coerência de cache é uma questão importante em sistemas de memória partilhada. **Descreva o problema e em que situações ocorre**. **Discuta as soluções para resolver o problema** da coerência de cache e compare os méritos relativos as alternativas de solução.

## Grupo II (1.0 + 1.5 + 0.5 + 2.0 + 2.0 + 1.0 = 8.0 valores)
## Programação concorrente com OpenMP

**1.** Escreva o código para paralelizar o ciclo abaixo com recurso a comandos OpenMP:

```
for(i = 2; i < MAX; i++)
      a[i] += a[i-2] + 5;
```

**2.** Optimize o seguinte código e escreva uma implementação paralela eficiente em OpenMP.

```
n = 7;
for (i = 0; i < N; i++){
  n += 2;
  A[i] = f(n);
}
sum = 0;
for (i = 0; i < N; i++){sum += A[i];}
```

**3.** Discuta a diferença entre `#pragma omp critical` e `#pragma omp atomic`.

4. Considere o seguinte código OpenMP, assuma que foi executado em um sistema com 4 threads (`OMP_NUM_THREADS=4`):

```
#define M 16;
#pragma omp parallel for private(j)
  for (i = 0; i < M; i++ ){
    for ( j = M - (i+1); j < M ; j++ ) {
      // Esta função tem um tempo de computação de 2s
      f(i, j, ...) ;
  }
}
```

**a)** Preencha a tabela a seguir com uma possível alocação de threads das primeiras iterações do ciclo (índice i), assumindo um escalonamento estático definido pela directiva OpenMP `schedule(static)`. Indique qual thread executa cada iteração e quanto tempo essa iteração leva.

Determine o tempo de execução aproximado por thread e o tempo total de execução.

| #iter | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Thread | | | | | | | | | | | | | | | | |
| Tempo/iter (s) | | | | | | | | | | | | | | | | |

| | Thread 0 | Thread 1 | Thread 2 | Thread 3 |
|---|---|---|---|---|
| Tempo de execução de thread individual | | | | |
| Tempo de execução total | | | | |

**b)** Responda a questão anterior assumindo um escalonamento dinâmico definido pela directiva OpenMP `schedule(dynamic,2)`.

| #iter | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Thread | | | | | | | | | | | | | | | | |
| Tempo/iter (s) | | | | | | | | | | | | | | | | |

| | Thread 0 | Thread 1 | Thread 2 | Thread 3 |
|---|---|---|---|---|
| Tempo de execução de thread individual | | | | |
| Tempo de execução total | | | | |

**c)** Justifique qual dos escalonamentos anteriores seria melhor para um caso genérico (número variável de iterações e threads).

## Grupo III (1.0 + 2.0 + 2.0 + 3.0 = 8.0 valores)
## Versão Serial do Trabalho Prático

Considere a sua solução para o trabalho prática de Computação Paralela e Distribuída (classificação/organização de documentos):

**1.** Apresente as estruturas de dados para manter os documentos (D), os armários (C) e suas associações.

**2.** Dadas as estruturas de dados definidas acima, escreva a função **`f1`** em C que faça uma atribuição round-robin inicial dos documentos D aos armários C, onde cada documento é colocado no armário correspondente ao resto da divisão do índice do documento i pelo número de armários C, isto é, `i%C`.

**Nota:** Não há limite para o número de documentos em um armário e nem há nenhuma exigência sobre o balanceamento da distribuição de documentos por armários.

**3.** Dadas as estruturas de dados definidas acima e outras adicionais que considerar relevantes, escreva a função **`f2`** em C que para cada armário, calcule a "média" de cada assunto de seus documentos.

**4.** Dadas as funções (**`f1`** e **`f2`**) e estruturas de dados definidas acima, escreva a função **`f3`** que organiza os documentos nos armários. Para tal, utilize o seguinte algoritmo aproximado para resolver este problema:

**Passo 1**: fazer uma atribuição inicial dos documentos D aos armários C usando a função **`f1`**;

**Passo 2**: para cada armário, calcule a "média" de cada assunto de seus documentos, usando a função **`f2`**;

**Passo 3**: para cada documento, calcule a distância às médias de cada armário e mova o documento para o armário com menor distância;

Use o seguinte modelo:

$$d' = \sum_{i=0}^{S-1} (a_i - b_i)^2,$$

**Passo 4**: vá para a **Passo 2** até que nenhum documento troque de armário.

Bom trabalho!