# Universidade Federal do ABC
# MCZA020-13 - Programação Paralela
# 2019.Q1

**Lista de Exercícios 2**

Prof. Emílio Francesquini

11 de março de 2019

Lista de termos cuja definição você **deve** saber:

- Arquitetura de von Neumann e gargalo (*bottleneck*) de von Neumann.
- Sistema operacional, muti-tasking, processo, thread.
- CPU: pipelining, instruction leval parallelism (ILP), multiple issue, hardware multithreading (SMT/Hyperthreading).
- Memória pricipal: memória virtual, endereço real *vs.* endereço virtual, páginas de memória, tabelas de página.
- Caches: caches L1, L2,..., blocos/linhas de cache, cache miss, cache hit, false sharing, coerência de cache (snooping e directory based).
- Taxonomia de Flynn: SISD, SIMD, MIMD, MISD.
- Topologias de interconexão: toroidal, hipercubos, aneis, grafo completo
- Comunicação: latência, banda.
- Software: SPMD, problemas vergonhosamente paralelos, balanceamento de carga, não determinismo, race conditions, seções críticas, exclusão mútua.
- Desempenho: speedup, eficiência, speedup linear, lei de Amdahl, escalabilidades forte e fraca.

## Exercícios

1. [PP] A adição de caches e memória virtual à um sistema computacional baseado na arquitetura de von Neumann faz com ele deixe de ser um sistema SISD? O que dizer sobre a adição de *pipelining*? *Multiple issue*? *Hardware multithreading*?

2. [PP] Suponha que um processador vetorial possua um sistema de memória que leva 10 ciclos para carregar uma palavra de 64 bits a partir da memória. Quantos bancos de memória seriam necessários para que uma sequência de operações de *load* possam, em média, precisar de apenas um ciclo para completar?

3. Descreva o que são localidades espacial e temporal no contexto de acessos à memória.

4. O que é coerência de cache? Por que o *false sharing* de linhas de cache pode se tormar um problema ao desempenho de programas com diversas threads?

5. [PP] Considere o problema de soma de números de ponto flutuante discutido em sala. Quando discutimos este problema, assumimos que cada uma das unidades funcionais levava o mesmo tempo. Suponha que a operação de *fetch* e de *store* levem 2 nanosegundos cada, e que o resto das operações sejam completadas em apenas um nanosegundo. Nessas condições responda:

   (a) Quanto tempo levaria uma soma entre números de ponto flutuante?

   (b) Quanto tempo levaria a soma de 1000 pares de pontos flutuantes sem o uso de *pipeline*?

   (c) Quanto tempo levaria a soma de 1000 pares de pontos flutuantes utilizando um *pipeline*?

   (d) O tempo para se efetuar um *fetch* ou um *store* pode variar bastante dependendo da localização do dado na hierarquia de memória (caches L1, L2, ..., RAM). Suponha que um *fetch* no nível 1 leve 2 nanosegundos, no nível 2 leve 5 nanosegundos e um *fetch* diretamente da memória principal leve 50 nanosegundos. O que ocorre no *pipeline* quando há um *miss* na L1 para um dos operandos? O que ocorre quando há um *miss* na L2?

6. Classificamos diversos sistemas utilizando a taxonomia de Flynn (SISD, SIMD, MIMD, MISD). Explique sucintamente o que cada categoria representa. Nas nossas discussões não demos nenhum exemplo sobre a categoria MISD. Como tal sistema trabalharia? Dê exemplos de possíveis usos para tal arquitetura.

7. Um multiprocessador consiste de 10 processadores, cada um com pico de execução de 200 MFLOPs. Qual a performance do sistema, medido em MFLOPs, quando 10% de um algoritmo é sequencial e 90% é paralelizável?

8. Desenhe um esboço das topologias de interconexão para hipercubos de dimensão 1, 2, 3 e 4.

9. Mostre que a largura da bissecção de um hipercubo de dimensão $d$ é $2^{d-1}$

10. Descreva um algoritmo paralelo para determinar o elemento máximo de um vetor $A$ de $n$ elementos, para uma topologia de anel com $p$ processadores. Considere que cada processador possui uma parte ($\frac{n}{p}$ elementos) do vetor $A$.

    (a) Detalhe este algoritmo.

    (b) Apresente as complexidades de tempo, tempo local e mensagens deste algoritmo.

11. Deseja-se um algoritmo paralelo que realize a multiplicação de um vetor $A$ de $n$ elementos por uma constante $c$ em uma máquina com topologia de grafo completo (completamente conexa) com $p$ processadores. Considere que cada processador possui uma parte ($\frac{n}{p}$ elementos) do vetor $A$.

    (a) Escreva este algoritmo.

    (b) Apresente as complexidades de tempo local e de mensagens deste algoritmo.

12. Escreva um algoritmo paralelo para computar o `OU lógico` de $n$ bits armazenados num vetor $B$, para a topologia de hipercubo com $n = 2^d$ processadores.

13. [PP] Suponha que um programa precisa executar $10^{12}$ instruções para resolver um problema em particular. Suponha também que um sistema computacional com um só processador possa resolver este problema em $10^{16}$ segundos (ou, aproximadamente, 11,6 dias). Então, na média, tal processador é capaz de executar $10^6$ (ou um milhão) de instruções por segundo. Agora suponha que o programa tenha sido paralelizado para execução em um sistema distribuído. Suponha ainda que se o sistema paralelo utilizar $p$ processadores, cada processador irá executar $10^{12}/p$ instruções e que cada processador precisa enviar $10^9(p-1)$ mensagens. Finalmente, suponha que não há nenhuma sobrecarga adicional na execução do programa paralelo. Ou seja, o programa irá completar a sua execução assim que cada um dos processadores tiver executado

todas as instruções a ele atribuídas, enviado todas as mensagens e que não haja nenhum atraso para a entrega das mensagens.

(a) Suponha que leve $10^{-9}$ segundo para enviar uma mensagem. Quanto tempo levaria para o programa executar utilizando 1000 processadores se cada processador for tão rápido quando o processador *single-core* original onde o programa sequencial foi executado?

(b) Suponha que seja necessário $10^{-3}$ segundo para enviar uma mensagem. Quanto tempo levaria para o programa executar com 1000 processadores?

14. [PP] *Speedups* e eficiência.

(a) Suponha que o tempo de execução de um programa sequencial seja dado por $T_{\text{sequencial}} = n^2$, com unidades de tempo dadas em microsegundos. Suponha ainda que uma versão paralela deste mesmo programa tenha o tempo de execução $T_{\text{paralelo}} = n^2/p + \log_2(p)$. Escreva um programa que determine os *speedups* e eficiências deste programa para diversos valores de $n$ e $p$. Rode o seu programa com $n = 10, 20, \ldots, 320$ e $p = 1, 2, 4, \ldots, 128$. O que ocorre com os *speedups* e eficiências conforme $p$ cresce e $n$ se mantém fixo? O que ocorre quando $p$ é fixado e $n$ varia?

(b) Suponha que $T_{\text{paralelo}} = T_{\text{sequencial}}/p + T_{\text{overhead}}$. Também suponha que fixamos $p$ e aumentamos o tamanho do problema.

- Mostre que caso $T_{\text{overhead}}$ cresça mais lentamente que $T_{\text{sequencial}}$ então a eficiência paralela crescerá juntamente com o tamanho do problema.
- Mostre que se, por outro lado, $T_{\text{overhead}}$ crescer mais rapidamente que $T_{\text{sequencial}}$, então a eficiência paralela cairá conforme aumentarmos o tamanho do problema.

15. Explique com suas próprias palavras o que é um programa escalável. Qual a diferença entre escalabilidade forte e fraca?

16. Suponha que $T_{\text{sequencial}} = n$ e que $T_{\text{paralelo}} = n/p + \log_2(p)$ com tempos em microsegundos. Se aumentarmos $P$ em um fator de $k$, encontre uma fórmula para saber quanto precisamos aumentar $n$ para que possamos manter a eficiência constante. Por qual fator devemos aumentar $n$ se dobrarmos o número de processos de 8 para 16? Este programa paralelo é escalável?

17. Um programa com speedup linear é fortemente escalável? Explique sua resposta.

18. [PP] Bob tem um programa que ele quer medir o desempenho baseado em duas entradas. A entrada chamada de `input1` e a entrada chamada de `input2`. Para ter uma ideia do que esperar quando ele adicionar as chamadas de medição de tempo no seu código, ele decidiu rodar o seu programa com as duas entradas usando o utilitário do Unix `time` para ter uma estimativa do tempo.

```
$ time ./ bobs prog < input data1
real 0 m0 .001 s
user 0 m0 .001 s
sys 0 m0 .000 s
$ time ./ bobs prog < input data2
real 1 m1 .234 s
user 1 m0 .001 s
sys 0 m0 .111 s
```

A função de medição de tempo que Bob está pensando em usar tem a precisão de milissegundos. O Bob estaria correto em utilizar essa função para medir o tempo de execução para a entrada `input1`? E quanto à entrada `input2`? Por quê?