Computação Heterogénea de Alto Desempenho Lab 1

Manuel Santos - 2019231352

22 September 2023

1 Defina ou expanda cada um dos seguintes termos ou acrónimos

- ***Flops, gigaflops, teraflops, petaflops, exaflops, zettaflops***: o acrónimo flops significa **FL**oating-point **O**perations **P**er **S**econd (operações de ponto flutuante por segundo). Este valor é utilizado como medida da performance de supercomputadores, ou seja, como *benchmark* destes. Flops é a unidade básica. Os prefixos métricos giga, tera, peta, exa e zetta indicam múltiplos desta unidade. Os valores destes são giga = 10^9, tera = 10^{12}, peta = 10^{15}, exa = 10^{18} e zetta = 10^{21}.
- ***Throughput***: *throughput* refere-se ao volume médio de dados que passa por uma rede durante período de tempo específico. Este conceito é importante na área de High-Throughput Computing (HTC), onde recursos de computação distribuídos são utilizados para aplicações que exigem grande poder de computação durante um longo período de tempo.
- ***Benchmark***: *benchmark* é um teste que mede a performance de um componente de hardware, software, ou de um computador. Estes testes servem para comparar o quão bom é um desempenho de um produto face a outros similares.
- ***Parallel Processing***: *parallel processing* refere-se ao uso de vários núcleos de um processador para efetuar várias operações ao mesmo tempo, isto é, em paralelo. Ao contrário da computação em série, a arquitetura paralela pode dividir um processo nos seus componentes e realizar cada um destes componentes simultaneamente.
- ***Moore's Law***: *Moore's Law*, ou Lei de Moore, refere-se à observação, feita por Gordon Moore (co-fundador da Intel) em 1965, de que o número de transistores num circuito integrado dobraria a cada dois anos. Esta observação
- ***Strong Scaling***: *scaling* refere-se à capacidade de hardware e software de fornecer um maior poder computacional, à medida que a quantidade de recursos é aumentada. Como tal, *strong scaling* refere-se à relação entre paralelização efetiva de um processo face ao número de *cores* (recursos) utilizados. Esta relação é descrita pela Lei de Amdahl.
- ***Starvation***: *starvation* refere-se à condição em que um processo computacional não pode ser suportado pelos recursos disponíveis. Esta condição pode ocorrer devido à falta de recursos do computador ou à existência de vários processos que competem pelos mesmos recursos.
- ***Latency***: *latency* refere-se ao atraso entre a intridução de dados num processo e a obtenção de resultados, no contexto da computação. Latency pode ainda referir-se ao atraso na comunicação em redes de computadores, ou seja, o tempo que os dados demoram a percorrer a rede.

- ***ALU***: *ALU* ou ***A**rithmetic **L**ogic **U**nit* refere-se à unidade funcional de um computador digital que efetua operações lógicas e aritméticas. Esta unidade encontra-se dentro do CPU.

- ***von Neumann Architecture***: arquitetura computacional na qual a maioria dos computadores de uso genérico se baseia. Esta inclui o uso de ciclos fetch-decode-execute para ler, descodificar e executar instruções. Nesta arquitetura os dados e as instruções são guardadas no mesmo espaço de forma binária.

- ***DRAM vs SRAM***: a *DRAM (**D**ynamic **RAM**)* é um circuito integrado de memória feito de milhares de transístores e condensadores. Na maioria dos computadores, a DRAM é composta por um transístor e um condensador. Este par cria uma célula de memória que representa um bit de dados. A SRAM é mais cara, mais rápida, e ocupa mais espaço que a DRAM, o que faz com que a SRAM seja usada para criar uma memória mais próxima do processador, a cache. enquanto que a DRAM é usada para fazer os módulos de RAM, fornecendo uma muito maior quantidade de RAM ao sistema.

- ***SIMD***: ***S**ingle **I**nstruction **M**ultiple **D**ata (SIMD)* refere-se à aplicação de uma operação específica a um conjunto de dados em simultâneo.

- ***SIMT***: ***S**ingle-**I**nstruction-**M**ultiple-**T**hread (SIMT)* refere-se a um método de escalonamento utilizado em CPU's. Esta tira partido da utilização de várias *threads* por core num sistema *multi-core* para alcançar um grande performance no processamento em paralelo.

- ***SPMD***: ***S**ingle **P**rogram – **M**ultiple **D**ata (SPMD)* refere-se ao paradigma de processamento paralelo em que todos os processos que participam na execução de um programa trabalham cooperativamente para executar o programa. No entatno, a qualquer instante de execução, diferentes processos podem executar diferentes fluxos de instruções e agir em diferentes dados e em diferentes secções do programa. De tal forma, estes processos podem autoprogramar-se dinamicamente, de acordo com a carga de trabalho do programa.

- ***VLSI***: ***V**ery **L**arge-**S**cale **I**ntegration (VLSI)* refere-se ao processo de criar circuitos integrados através da combinação de grandes quantidades de transístores num único chip ou *waffer* de silício.

2 Qual é considerado o requisito principal que diferencia HPC de outros computadores? Que outros requisitos são importantes?

HPC apresentam uma capacidade de processamento muito superior à de computadores de uso genérico. Esta capacidade é utilizada para resolver problemas computacionalmente intensivos. Outros pontos a ter em conta aquando a utilização de HPC são:

- capacidade de executar cálculos que requerem muita memória;

- o mesmo programa necessita de ser corrido muitas vezes com dados diferentres;

- o programa em execução demora muito tempo a correr, mas pode ser corrido mais rapidamente quando uma aqrquitetura em paralelo é utilizada;

3 Indique seis técnicas para melhorar throughput performance num sistema

- **Processamento Paralelo**: dividir processos em sub-tarefas menores que possam ser executadas em paralelo;
- **Alocação de Recursos Apropriada**: alocar recursos, como CPU e memória e largura de banda de rede, de forma adequada. Este processo permite identificar *bottlenecks* e alocar recursos onde estes são mais necessários;
- **Otimização de Algoritmos**: otimizar algoritmos e estruturas de dados de forma a tirarem o máximo partido dos recursos disponíveis de forma eficiente;
- **Escalabilidade**: projetar o sistema de forma a que seja fácil adicionar recursos para lidar com cargas de trabalho mais intensivas quando necessário;
- **Compressão de Dados**: utilizar técnicas de compressão que permitam reduzir o volume de dados transmitidos no sistema ou armazenados em disco;
- **Cache**: armazenar em cache dados utilizados com mais frequência, de forma a dimiuir as leituras de memória ao disco.

4 Descreva o que entende por "end of Moore's law"

O aumento de performance de processadores single-core tem diminuído consideravelmente desde meados do ano 2005. Existem diversos fatores responsáveis por esta diminuição. Aquele que mais se salienta passa pela dimensão dos transístores atuais, que estão à beira do limite físico possível.

Este fator leva também ao fim do *Dennard Scalin*, uma "lei" da computação que nos diz que há medida que a dimensão de um transístor diminui, também o seu consumo energético diminui, levando a um consumo por área do circuito relativamente constante, ainda que os transistores estejam mais próximos. A aproximação do limite físico levou a que o *bottleneck* atual não seja o consumo mas sim a capacidade de dissipar a energia térmica produzida.

Como tal, e tendo em conta estes fatores, a Lei de Moore, na sua forma original, tem vindo a não ser cumprida nos últimos anos.

5 Qual era o computador mais rápido no ano em que nasceu? Que tecnologias eram usadas nesse computador? Quão mais rápido é hoje o computador mais veloz do mundo?

De acordo com a página top500.org, em Junho de 2001 (nasci a 29 de Maio de 2001) o computador mais rápido era o *ASCI White, SP Power3 375 MHz*, criado pela IBM e alojado no *Lawrence Livermore National Laboratory*. Este alcançou um máximo de 7,304.00 GFlop/s, apresentando no entanto um máximo teórico de 12,288.00 GFlop/s. Para alcançar tais valores, dispendia de 8,192 cores.

Atualmente, o computador mais rápido do mundo é o *Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11*, criado pela HPE e alojado no *DOE/SC/Oak Ridge National Laboratory*. Apresenta um máximo de 1,194.00 PFlop/s, e podendo alcançar um máximo teórico de 1,679.82 PFlop/s,

ou seja, valores de uma ordem de magnitude 6 vezes maior (1,000,000) do que aqueles apresentados pelo computador mais rápido do mundo há 22 anos. No que toca ao número de cores, dispõem de 8,699,904 cores, sensivelmente 1000 vezes mais.

Para os exercícios que se seguem foram utilizados as expressões a, e,d e c para efetuar os cálculos necessários.

$$s = \frac{1}{1 - f + \frac{f}{g}} \tag{a}$$

$$s = \frac{1}{1 - f + \frac{f}{g} + \frac{n \times v}{T_O}} \tag{b}$$

$$s = \frac{T_O}{T_A} \tag{c}$$

$$f = \frac{T_F}{T_O} \tag{d}$$

Foram ainda utilizados os esquemas 1, 2, 3 e 4 como ajuda à compreensão.

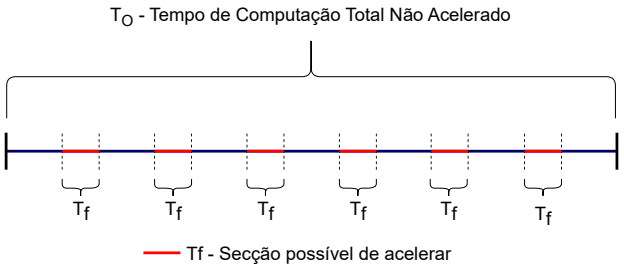

Figure 1: Cenário não acelerado

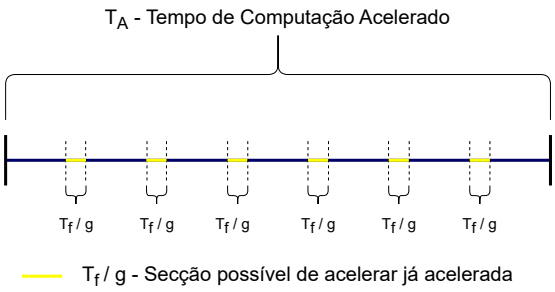

Figure 2: Cenário acelerado

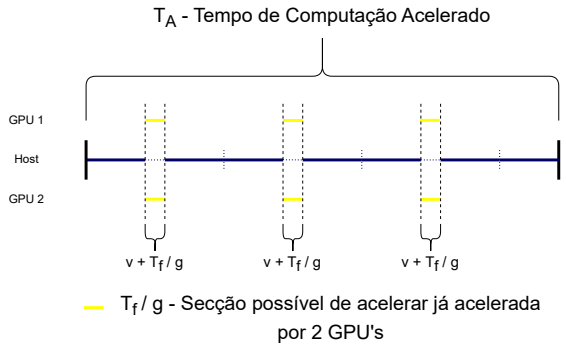

Figure 3: Cenário acelerado por 2 GPU's

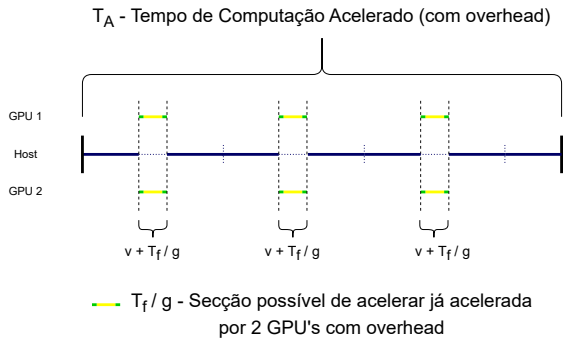

Figure 4: Cenário acelerado por 2 GPU's com overhead

6 Assuma que o cálculo da raiz quadrada é responsável por 20% do tempo gasto por um benchmark gráfico. As instruções de vírgula flutuante perfazem 50% do tempo total de execução. Compare as duas alternativas e explique qual é mais vantajosa. Considere as seguintes alternativas:

6.1 Tornar o cálculo da raiz quadrada 10 vezes mais rápido

Sabemos que f = 0.2 e g = 10. Como tal, tem-se que:

$$s = \frac{1}{1 - f + \frac{f}{g}} = \frac{1}{1 - 0.2 + \frac{0.2}{10}} \approx 1.2195\times$$

6.2 Tornar todas as instruções de vírgula flutuante 60% mais rápidas

Sabemos que f = 0.5 e g = 1.6. Como tal, tem-se que:

$$s = \frac{1}{1 - f + \frac{f}{g}} = \frac{1}{1 - 0.5 + \frac{0.5}{1.6}} \approx 1.2308\surd$$

Como tal, e face aos resultados obtidos, a segunda opção, ou seja, *"Tornar todas as*

instruções de vírgula flutuante 60% mais rápidas" é a mais vantajosa, pois permite obter um maior *speedup*.

7 Considere que tem de acelerar um programa que se encontra a 30% de conseguir executar em tempo-real, numa CPU dual-core, demorando 1.3 seg a processar 60 frames (o objetivo consiste em atingir 60 frames por segundo). Após análise do profiling temporal do programa sequencial, descobriu-se que, entre outras funcionalidades, executa 6 FFTs que representam 40% do tempo total de processamento.

7.1 No sentido de acelerar o programa, admita que vamos usar 2 GPUs para processar as 6 FFTs. Sabendo que o potencial de aceleração de cada GPU é 22x, qual é o speedup obtido (ignore tempos de transferência de dados entre host e device)? Consegue atingir o objetivo inicial?

Sabemos que f = 0.4 e g = 2 × 22 = 44 (estão a ser utilizadas 2 GPU's). Como tal, tem-se que:

$$s = \frac{1}{1 - f + \frac{f}{g}} = \frac{1}{1 - 0.4 + \frac{0.4}{4}} \approx 1.6418$$

$$s = \frac{T_O}{T_A} \Leftrightarrow T_A = \frac{T_O}{s} = \frac{1.3}{1.6418} \approx 0.7918[s]$$

$$\frac{60}{0.7918} \approx 75.78FPS\surd$$

Segundo os resultados obtidos, é possível atingir o objetivo inicial de alcançar 60 FPS.

7.2 Admita agora que os tempos de transferência host ⇔ device representam 35ms em cada sentido (host ⇒ device e device ⇒ host), para cada GPU. O objetivo é atingido?

Para este caso, vou admitir que a transferência de dados do host para as GPU's é feita em simultâneo entre o host e as GPU's.

Sabendo que existem 2 overheads por cada utilização da GPU (host ⇒ device e device ⇒ host) e que temos 2 GPU's disponíveis, concluo que $n = \frac{6}{2(GPU's)} \times 2(overheads) = 6$ e que v = 35 ms. Como tal, tem-se que:

$$s = \frac{1}{1 - f + \frac{f}{g} + \frac{n \times v}{T_O}} = \frac{1}{1 - 0.4 + \frac{0.4}{44} + \frac{6 \times 35 \times 10^{-3}}{1.3}} \approx 1.2976 \quad \text{(e)}$$

$$s = \frac{T_O}{T_A} \Leftrightarrow T_A = \frac{T_O}{s} = \frac{1.3}{1.2976} \approx 1.002[s]$$

$$\frac{60}{1.002} \approx 58.88FPS\times$$

Segundo os resultados obtidos, não é possível atingir o objetivo inicial de alcançar 60 FPS quando existe um overhead de 35ms em cada sentido (host ⇒ device e device ⇒ host).

8 Suponha que certo modo de execução de um programa é melhorado 10 vezes. O novo modo de funcionamento é usado 50% do tempo do programa original.

8.1 Determine o ganho de rapidez (speedup) obtido pela utilização do modo melhorado de funcionamento.

Sabemos que f = 0.5 e g = 10. Como tal, tem-se que:

$$s = \frac{1}{1 - f + \frac{f}{g}} = \frac{1}{1 - 0.5 + \frac{0.5}{10}} \approx 1.8181$$

8.2 Que percentagem de tempo de execução original foi convertido para usar o novo modo de funcionamento?

$$f = \frac{T_F}{T_O} = 0.5 = 50\%$$

References

[1] *ACM Digital Library*. https://dl.acm.org/doi/10.5555/1074100.1074135.

[2] Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. "Chapter 7 - High-Throughput Computing: Task Programming". In: *Mastering Cloud Computing*. Ed. by Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. Boston: Morgan Kaufmann, 2013, pp. 211–252. ISBN: 978-0-12-411454-8. DOI: https://doi.org/10.1016/B978-0-12-411454-8.00007-3. URL: https://www.sciencedirect.com/science/article/pii/B9780124114548000073.

[3] Frederica Darema. *SPMD Computational Model*. https://link.springer.com/referenceworkentry /10.1007/978-0-387-09766-4_26.

[4] *June 2001 — TOP500*. https://www.top500.org/lists/top500/2001/06/.

[5] August Liljenberg. *The End of Moore's Law - Farsight*. https://farsight.cifs.dk/are-we-at-the-end-of-moores-law/. Oct. 2022.

[6] *Moore's Law*. https://www.intel.com/content/www/us/en/newsroom/resources/moores-law.html.

[7] *O que so DRAM e SDRAM?* https://student.dei.uc.pt/~hecosta/sramdram.htm.

[8] *Parallel Computing And Its Modern Uses — HP® Tech Takes*. https://www.hp.com/us-en/shop/tech-takes/parallel-computing-and-its-modern-uses. Oct. 2019.

[9] *Resource Starvation - Glossary — CSRC*. https://csrc.nist.gov/glossary/term/ resource_starvation.

[10] *Scalability: strong and weak scaling – PDC Blog*. https://www.kth.se/blogs/pdc/2018/11/scalability-strong-and-weak-scaling/. Nov. 2018.

[11] *SIMD (Single Instruction Multiple Data Processing)*. https://link.springer.com/referenceworkentry/10.1007/978-0-387-78414-4_220.

[12] *Throughput vs Latency - Difference Between Computer Network Performances - AWS*. https://aws.amazon.com/compare/the-difference-between-throughput-and-latency/. [Online; accessed 2023-09-22].

[13] *VLSI Design - Digital System*. https://www.tutorialspoint.com/vlsi_design/vlsi_design_digital_system.html.

[14] *Von Neumann architecture - Architecture - Eduqas - GCSE Computer Science Revision - Eduqas - BBC Bitesize*. https://www.bbc.co.uk/bitesize/guides/zhppfcw/revision/3.

[15] *We're not prepared for the end of Moore's Law*. https://www.technologyreview.com/2020/02/24/905789/were-not-prepared-for-the-end-of-moores-law/. Feb. 2020.

[16] *What is a Benchmark?* https://www.computerhope.com/jargon/b/benchmar.htm.

[17] *What's the Computing Difference Between a TeraFLOPS and a PetaFLOPS? — HP® Tech Takes*. https://www.hp.com/us-en/shop/tech-takes/computing-difference-between-teraflops-and-petaflops. July 2019.

[18] *When to use HPC cluster — High Performance Computing*. https://www.hpc.iastate.edu/guides/introduction-to-hpc-clusters/when-to-use-hpc-cluster.

[19] Yuanxu Xu et al. *The Analysis of Generic SIMT Scheduling Model Extracted from GPU*. https://link.springer.com/chapter/10.1007/978-3-642-41635-4_2.