

Implementação de um escalonador de processos em GPU

Guilherme Martins

guilhermemartins@usp.br

6 de abril de 2017

Conteúdo

- 1 Introdução
 - Apresentação
 - Conceitos
- 2 Objetivos
- 3 Metodologia
- 4 Cronograma
- 5 Referências

Guilherme Martins

Graduado em Sistemas de Informação pela Universidade Federal de Viçosa. Atualmente, mestrando em Sistemas Distribuídos e Computação Concorrente sob orientação do professor Paulo S. L. de Souza.

Área de Atuação

- Programação paralela e concorrente;
- Programação híbrida;
- Pesquisa Operacional.

Contato

guilhermemartins@usp.br
martins_guilherme@live.com



O escalonamento de processos (*process scheduling*) é uma das tarefas fundamentais de um Sistema Operacional. O objetivo de um algoritmo escalonador é definir a melhor ordem de alocação dos processos na CPU no intuito de maximizar a quantidade de *jobs* processados e minimizar o tempo de processamento [1].

Existem diversas sequências possíveis para ordenar os processos, levando em consideração diversos critérios como tamanho da tarefa (LPT), ordem de entrada (FIFO), divisão igualitária (FSS), entre outros [2].

Representação de fluxo de processamento

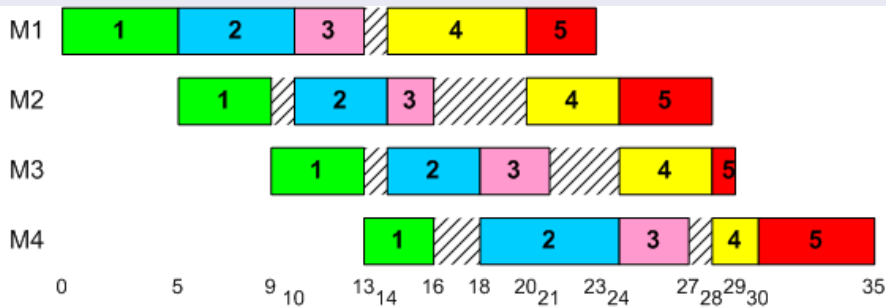


Figura 1: Flow Shop or Assembly Line Work Flow, fonte:

<http://www.bizharmony.com/Blog/September-2013/monetize-your-abandoned-or-dormant-technologies.aspx>

É evidente que existem inúmeras combinações possíveis para o sequenciamento destes processos, que podem gerar melhores ou piores resultados, baseados nas métricas estabelecidas anteriormente.

GPU

GPU (*Graphics Processing Unit*) ou Unidade de Processamento Gráfico é uma classe de microprocessadores especializados no processamento gráfico em computadores. Define-se computação com GPU a utilização conjunta da placa gráfica e da CPU no intuito de acelerar o processamento de aplicações ou dados.

<http://www.nvidia.com.br/object/what-is-gpu-computing-br.html>

Nvidia GeForce GTX Titan Z (desktop)



Figura 2: Nvidia GeForce GTX Titan Z (desktop), fonte:
<http://www.nvidia.com.br/object/geforce-gtx-titan-z-br.html>

Nvidia GeForce GTX 980M (Notebook)

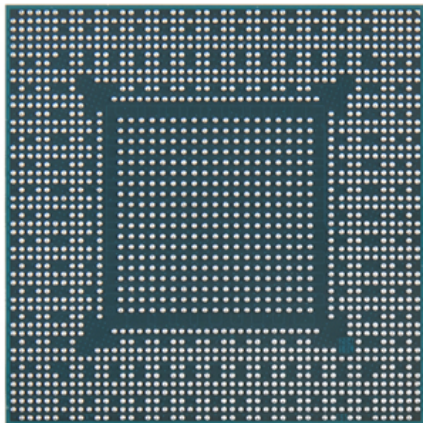


Figura 3: Nvidia GeForce GTX 980M (Notebook), fonte: <http://www.nvidia.com.br/object/geforce-gtx-900m-graphics-cards-br.html>

AMD Radeon R9 nano (desktop)



Figura 4: AMD Radeon R9 nano (desktop), fonte:
<http://www.amd.com/pt-br/products/graphics/desktop/R9>

AMD Radeon R9 m200 (notebook)

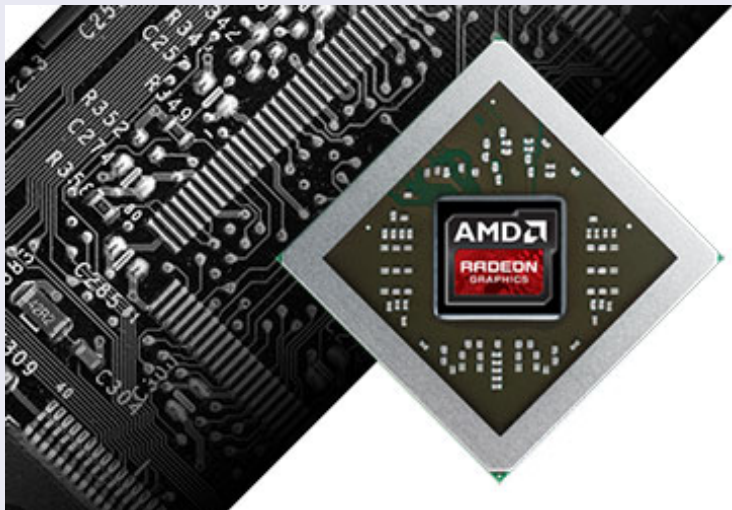


Figura 5: AMD Radeon R9 m200 (notebook), fonte:

<http://www.amd.com/pt-br/products/graphics/notebook/r9-m200>

GPGPU

General Purpose Graphics Processing Unit ou Unidade de Processamento Gráfico de Propósito Geral trata-se da do uso, através de tecnologias para programação de propósito geral, como OpenCL e CUDA, de placas de vídeo para outros fins além de renderização gráfica. Ou seja, é a utilização da GPU para realizar a computação em aplicações que antes eram tratadas pela CPU.

Exemplos

Exemplos da utilização das GPUs para outros fins podem ser identificados nas áreas de processamento de imagem, visão computacional, inteligência artificial, cálculo numérico e pesquisa operacional.

fonte

<https://pt.wikipedia.org/wiki/GPGPU>

Como funciona a aceleração por GPU

Como funciona a aceleração com GPU

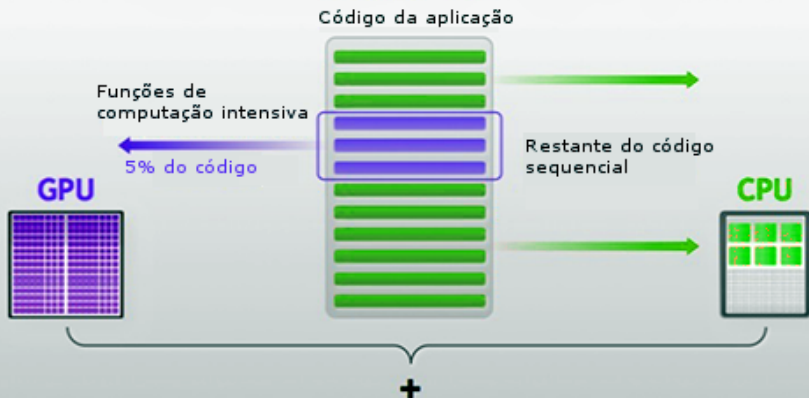


Figura 6: fonte:

<http://www.nvidia.com.br/object/what-is-gpu-computing-br.html>

CUDA

CUDA™ (Compute Unified Device Architecture) é uma plataforma de computação paralela e um modelo de programação inventados pela NVIDIA. Ela permite aumentos significativos de performance computacional ao aproveitar a potência da unidade de processamento gráfico (GPU) para o processar dados.

O modelo de programação CUDA utiliza as linguagens C, C++ ou Fortran.

http://www.nvidia.com.br/object/cuda_home_new_br.html

Fluxo de processamento em CUDA

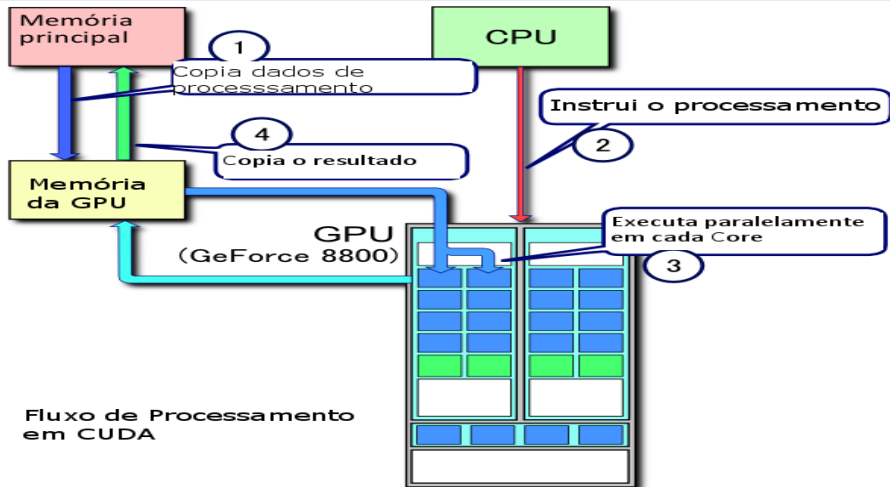


Figura 7: Adaptado de <https://en.wikipedia.org/wiki/CUDA>

GPUs são dispositivos que oferecem poder computacional bastante superior aos processadores comuns CPUs, quando trata-se de problemas massivamente paralelos e que apresentam um espaço solução considerável. Uma das ferramentas mais estáveis e utilizadas na literatura para resolução de problemas diversos em GPU trata-se da plataforma CUDA [3].

Conteúdo

- 1 Introdução
 - Apresentação
 - Conceitos
- 2 Objetivos**
- 3 Metodologia
- 4 Cronograma
- 5 Referências

Objetivo Geral

Simular as funcionalidades de um escalonador de processos de um Sistema Operacional em GPU.

Objetivos Específicos

- Desenvolver um método heurístico baseado nas diferentes abordagens de *scheduling* presentes na literatura que garanta uma solução razoável para o problema em questão;
- Avaliar o desempenho computacional da solução desenvolvida, comparando com outras já existentes;
- Documentar as conclusões obtidas através da escrita de um artigo.

Conteúdo

- 1 Introdução
 - Apresentação
 - Conceitos
- 2 Objetivos
- 3 Metodologia**
- 4 Cronograma
- 5 Referências

O algoritmo será construído baseado na metodologia *multistart*, que prevê que diversas soluções iniciais podem ser paralelamente refinadas, através de uma técnica como SWAP e SHIFT, e suas interações podem ser avaliadas através de um critério de parada ou metodologia de corte (*branch & bound*).

Pretende-se gerar instâncias de *jobs*, através de uma ferramenta como o IBM CPLEX, que apresentarão quantidade e tempos de processamento variável. A solução construída deve ser capaz de processar estes dados em GPU, gerar as sequências e tempos finais, que serão comparados com algoritmos tradicionais.

Conteúdo

- 1 Introdução
 - Apresentação
 - Conceitos
- 2 Objetivos
- 3 Metodologia
- 4 Cronograma**
- 5 Referências





Cronograma de Atividades do Projeto



Tabela 1: Cronograma do Projeto em Meses

Atividade	Abril	Maio	Junho
Estudo da ferramenta e Revisão Bibliográfica	●		
Implementação	●	●	●
Testes e Validação		●	●
Escrita do Relatório Final (artigo)			●

Conteúdo

- 1 Introdução
 - Apresentação
 - Conceitos
- 2 Objetivos
- 3 Metodologia
- 4 Cronograma
- 5 Referências

-  Andrew S Tanenbaum and Herbert Bos.
Modern operating systems.
Prentice Hall Press, 2014.
-  Michael L Pinedo.
Scheduling: theory, algorithms, and systems.
Springer Science & Business Media, 2008.
-  Mathias Bourgoïn, Emmanuel Chailloux, and Jean-Luc Lamotte.
Efficient abstractions for gpgpu programming.
International Journal of Parallel Programming, 42(4):583–600, 2014.
-  Ashok Dwarakinath.
A fair-share scheduler for the graphics processing unit.
PhD thesis, STONY BROOK UNIVERSITY, 2008.

-  N Melab, I Chakroun, M Mezmaç, and D Tuyttens.
A GPU-accelerated Branch-and-Bound Algorithm for the Flow-Shop Scheduling Problem.
14th IEEE International Conference on Cluster Computing, Cluster'12, 2012.
-  Sparsh Mittal and Jeffrey S. Vetter.
A survey of cpu-gpu heterogeneous computing techniques.
ACM Comput. Surv., 47(4):69:1–69:35, July 2015.