U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Técnicas de Computação de Alto Desempenho para o Processamento e Análise de Imagens Complexas da Cavidade Pélvica Feminina

Carlos Alex S. J. Gulo

Orientador: Prof. Dr. João Manuel R. S. Tavares

Co-Orientador: Prof. Dr. Antonio Carlos Sementille

Relatório: Planeamento de Investigação
Programa Doutoral em Engenharia Informática

Julho, 2014

© Carlos Alex S. J. Gulo: Julho, 2014

Faculdade de Engenharia da Universidade do Porto

Técnicas de Computação de Alto Desempenho para o Processamento e Análise de Imagens Complexas da Cavidade Pélvica Feminina

Carlos Alex S. J. Gulo

Dissertação submetida à Faculdade de Engenharia da Universidade do Porto
como requisito parcial para obtenção do grau de

Doutor em Engenharia Informática

Julho, 2014

Abstract

Supported by processors evolution, high performance computing have contributed to development in several scientific research areas which require advanced computations, such as image processing, augmented reality, and others. To fully exploit high performance computing available in these resources and to decrease processing time, is necessary apply parallel computing. However, those resources are expensive, which implies the search for alternatives ways to use it. The multicore and manycore processors architecture become a low cost options, as they were designed to provide infrastructure for high performance computing and attend real-time applications. With the improvements gained in technologies related to multicomputer, multiprocessor and, more recently, to GPGPUs, the parallelization of computational image processing and image analysis techniques has gained extraordinary prominence by enabling the reduction of the processing time of complex methods applied in complex images. The main objective of this doctoral project is to develop new techniques to massively parallel programming applied to the processing and analysis, efficiently, particularly in real-time, complex images of female pelvic cavity.

Keywords: High Performance Computing. Image Processing. Medical Image.

Resumo

A evolução constante na velocidade de cálculos dos processadores tem sido uma grande aliada no desenvolvimento de áreas da Ciência que exigem processamento de alto desempenho. Associados aos recursos computacionais faz-se necessário o emprego de técnicas de computação paralela no intuito de explorar ao máximo a capacidade de processamento da arquitetura escolhida, bem como, reduzir o tempo de espera no processamento. No entanto, o custo financeiro para aquisição deste tipo de hardware não é muito baixo, implicando na busca de alternativas para sua utilização. As arquiteturas de processadores multicore e General Purpose Computing on Graphics Processing Unit (GPGPU), tornam-se opções de baixo custo, pois são projetadas para oferecer infraestrutura para o processamento de alto desempenho e atender aplicações que requerem resultados de forma muito rápida, em particular em tempo real. Com o aperfeiçoamento das tecnologias multicomputador, multiprocessador e GPGPU, a paralelização de técnicas de processamento e análise de imagem tem obtido destaque por viabilizar a redução do tempo de processamento de métodos complexos aplicados em imagens complexas. O objetivo principal deste projeto de doutoramento é desenvolver novas técnicas de programação massivamente paralela aplicadas ao processamento e análise, de forma eficiente, em particular em tempo real, de imagens complexas da cavidade pélvica feminina.

Keywords: Processamento de Alto Desempenho. Processamento e Análise de Imagem. Imagem Médica.

Índice

1 Introdução **1**

I Fundamentos Teóricos **3**

2 Arquiteturas Paralelas **5**
2.1 Introdução 5
2.2 Histórico dos *multicores* e *manycores* 11
2.3 Considerações Finais 13

3 Técnicas de Computação Paralela **15**
3.1 Introdução 15
3.1.1 Decomposição 15
3.1.2 Sincronização 17
3.2 Métodos de Paralelização em GPGPU 18
3.3 Aspectos de Programação Paralela 22
3.4 Considerações Finais 23

4 Revisão Sistemática de Literatura **25**
4.1 Introdução 25
4.2 Aplicação da Revisão Sistemática 26
4.2.1 Planeamento 26
4.2.2 Execução 28
4.2.3 Resultados 29
4.3 Considerações Finais 32

II Proposta de Tese **33**

5 Proposta de Tese **35**
5.1 Motivação 35
5.2 Objetivos 36
5.3 Questão de Pesquisa 37

5.4 Hipótese de Pesquisa 37
5.5 Metodologia Proposta 38
5.5.1 Tarefa de Pesquisa 1 38
5.5.2 Tarefa de Pesquisa 2 38
5.6 Plano de Trabalho 39

Referências **43**

A Periódicos de Interesse **49**

B Conferências de Interesse **51**

C Grupos de Pesquisa **53**

Index **49**

Lista de Figuras

2.1 Taxonomia de Flynn (adaptado de [1]). 6
2.2 Modelo de arquiteturas para fluxos de dados múltiplos (adaptado de [2]). . 7
2.3 Modelo de arquiteturas para fluxos de instruções múltiplas (adaptado de [2]). 7
2.4 Lei de Amdahl (adaptado de [3]). 9
2.5 Diferenças entre: CPU (*multicores*) e GPGPU *manycores* (adaptado de [3]). 10
2.6 Arquitetura CUDA (adaptado de [4]). 13

3.1 Paralelização utilizando modelo *pipeline* (adaptado de [5]). 16
3.2 Conceitos de processos, *threads* e prioridades (adaptado de [5]). 17
3.3 Evolução de Operações em Ponto Flutuante em CPUs e GPUs (adaptado de [4]). 19
3.4 *Single Instruction, Multiple Thread* (adaptado de [4]). 20
3.5 Visão simplificada da GPGPU Fermi (Nvidia) (adaptado de [6]). 21

Lista de Tabelas

4.1 Número de publicações Identificadas (Ident.), Selecionadas (Sel.) e Avaliadas (Aval.) na etapa de Execução. 28

5.1 Cronograma de Atividades. 41

A.1 Periódicos relacionados aos temas de pesquisa. 49

B.1 Conferências de Interesse. 51

Lista de Abreviaturas

RSL	Revisão Sistemática de Literatura
ALU	*Arithmetic and Logic Unit*
API	*Application Programming Interface*
CPU	*Central Processing Unit*
CT	*Computed Tomography*
CUDA	*Compute Unified Device Architecture*
FPGA	*Field-Programmable Gate Array*
GPGPU	*General Purpose Computing on Graphics Processing Unit*
GPU	*Graphics Processing Unit*
IBM	*International Business Machines Corporation*
IEEE	*Institute of Electrical and Electronics Engineers*
LD/ST	*load/store*
MFLOPS	*Millions of Floating Point Operations Per Second*
MIMD	*Multiple Instruction, Multiple Data Stream*
MIPS	*Million of Instructions per Second*
MISD	*Multiple Instruction, Single Data Stream*
MPI	*Message Passing Interface*
MRI	*Magnetic Resonance Imaging*
NoC	*Network-on-Chip*
OpenCL	*Open Computing Language*

OpenCV *Open Source Computer Vision*

OpenGL *Open Graphics Library*

OpenMP *Open Multi-Processing*

PC *Personal Computer*

POSIX *Portable Operating System Interface*

SDDAC *Sistema de Detecção de Doença Auxiliado por Computador*

SIMD *Single Instruction, Multiple Data Stream*

SIMT *Single Instruction, Multiple Thread*

SISD *Single Instruction, Single Data Stream*

SM *Stream Multiprocessor*

SMP *Symmetric Multiprocessor*

SP *Streaming Processors*

SPM *Stream Programming Model*

SPMD *Single Program Multiple Data Stream*

SFU *Special Function Units*

UP *Unidade de Processamento*

VTK *Visualization ToolKit*

2D *bidimensional*

3D *tridimensional*

Capítulo 1

Introdução

Ao longo da história da Computação, o avanço da capacidade de processamento esteve ligado diretamente ao aumento na velocidade de cálculo dos processadores. Tradicionalmente, a indústria tem aumentado a velocidade de cálculo inserindo um maior número de transístores nos processadores [3]. Esta abordagem tem enfrentado dificuldades devido às limitações físicas do silício, principalmente, pelo excesso de consumo de energia e aquecimento destes processadores [5].

Nos últimos anos, no entanto, os avanços nesta área seguiram outra direção, sendo populares atualmente as arquiteturas *multicore* de uso geral e as *General Purpose Computing on Graphics Processing Unit* (GPGPU). A indústria de processadores passou a oferecer chips com vários núcleos compondo o processador. Estes processadores baseados na arquitetura multicore de uso geral, oferecem recursos de paralelismo, proporcionando ganho de desempenho e processamento mais rápido [5, 7].

A procura por processamento de alto desempenho, em geral, tem sido atendida por equipamentos ou sistemas computacionais de custo elevado. Diante da popularização das GPGPUs e a aplicação de técnicas consolidadas de programação paralela, diversas áreas de pesquisa como a computação científica [8], o processamento e a análise de imagem médica [9, 10, 11], imagens de satélite [12, 13], imagens adquiridas e usadas na navegação de robôs [14], realidade aumentada [14, 15], e muitas outras, podem conquistar avanços ainda mais significativos, sem a necessidade de grandes investimentos financeiros.

Dentre as áreas mencionadas anteriormente, a área de processamento e análise de imagens médicas [9, 10, 11] tem contribuído bastante com o avanço da medicina, incorporando técnicas e metodologias para auxílio médico em diagnósticos clínicos, no seguimento de patologias e na definição de melhores planos de reabilitação. O diagnóstico, na maioria das vezes, é realizado a partir de imagens capturadas por aparelhos de *Computed Tomography* (CT), Ultrassonografia, *Magnetic Resonance Imaging* (MRI), dentre outros.

Contudo, estas imagens possuem planos de cortes transversais, oclusão causada por outros órgãos ou posicionamento do paciente no momento do exame, além de ruídos, resultando na falta de continuidade das informações extraídas por estes dispositivos. Assim, para obter diagnósticos mais consistentes e precisos, bem como realizar o processamento em tempo real, é necessário combinar técnicas computacionais de processamento e análise de imagens [16, 9] e de computação de alto desempenho [17, 18, 19, 20].

A presente proposta visa explorar e desenvolver novas técnicas voltadas ao processamento massivamente paralelo que possam ser aplicada, de forma eficiente, nomeadamente em tempo real, à análise e diagnóstico baseado em imagens médicas.

A presente Proposta de Tese está organizada em quatro capítulos, incluindo o presente Capítulo de introdução:

- **Capítulo 2 - Arquiteturas Paralelas**, trata conceitos e terminologias da área de computação paralela e aspectos de evolução histórica, sobre arquiteturas GPGPUs e *multicores*. Descreve, de maneira introdutória, aspectos da computação de alto desempenho baseado na taxonomia de Flynn, métrica para avaliação de desempenho, bem como, apresenta o modelo de comunicação utilizado na arquitetura GPGPU;

- **Capítulo 3 - Técnicas de Computação Paralela**, apresenta os métodos de paralelização de maneira geral, descreve a necessidade de decompor problemas em tarefas menores e também, apresenta o funcionamento dos mecanismos de sincronização, utilizados na comunicação entre as unidades de processamento. O capítulo concentra-se em introduzir os Métodos de Paralelização em GPGPU, além de apresentar a arquitetura *Compute Unified Device Architecture* (CUDA) e uma breve comparação entre as linguagens de programação paralela: CUDA C/C++, *Open Multi-Processing* (OpenMP) e *Message Passing Interface* (MPI);

- **Capítulo 4 - Revisão Sistemática**, apresenta a descrição dos procedimentos realizados na revisão de literatura, e a discussão acerca das principais pesquisas relacionadas com a presente proposta.

- **Capítulo 5 - Projeto de Tese**, são apresentados os objetivos, questões de pesquisa, hipótese e metodologia de pesquisa, além da descrição detalhada do plano de trabalho previsto para a realização do doutoramento.

- Nos **Apêndices A** e B, estão elencadas as revistas e conferências de interesse, respectivamente. No **Apêndice C**, encontra-se uma tabela listando os grupos de pesquisa com interesse em técnicas de processamento e análise de imagem médica combinadas com estratégias de computação de alto desempenho.

Parte I

Fundamentos Teóricos

Capítulo 2

Arquiteturas Paralelas

A maior procura por desempenho computacional, em geral, é na realização de cálculos, a fim de atender aplicações científicas, médicas, militares, entretenimento, dentre outras. Na maioria das vezes, estas aplicações exigem o desenvolvimento de técnicas para processamento de alto desempenho. O presente capítulo apresenta conceitos e terminologias inerentes à computação paralela, além de um breve histórico sobre a evolução das arquiteturas paralelas.

2.1 Introdução

A necessidade de processamento mais veloz é cada vez mais exigida em diversas áreas de pesquisa e da indústria em geral. A resolução de problemas em áreas como a computação científica, a computação gráfica, o processamento e análise de imagem, a compressão de vídeo, manipulação de enormes quantidade de dados, previsão do tempo, dentre muitos outros, requer alto poder de processamento [21].

A computação paralela tem sido utilizada para lidar com problemas desta magnitude. Por meio de técnicas de paralelização, utilizadas em processamento de alto desempenho, as aplicações são reescritas com objetivo de decompor o algoritmo ou os dados em partes menores. Estas partes menores, também chamadas de tarefas, são distribuídas para serem executadas em processadores ou núcleos simultaneamente [22, 23, 24, 25].

Durante estas etapas realiza-se o processo de comunicação e coordenação geral das tarefas entre os processadores envolvidos no processamento. Ao utilizar a programação paralela, dois fatores devem ser levados em consideração: o tipo de arquitetura paralela e o tipo de comunicação dos processadores. Dessa maneira, apresenta-se uma taxonomia baseada no fluxo de instruções e fluxo de dados de uma aplicação, conhecida como

taxonomia de Flynn [1, 26, 21]. Esta classificação amplamente utilizada, é ilustrada na Figura 2.1.

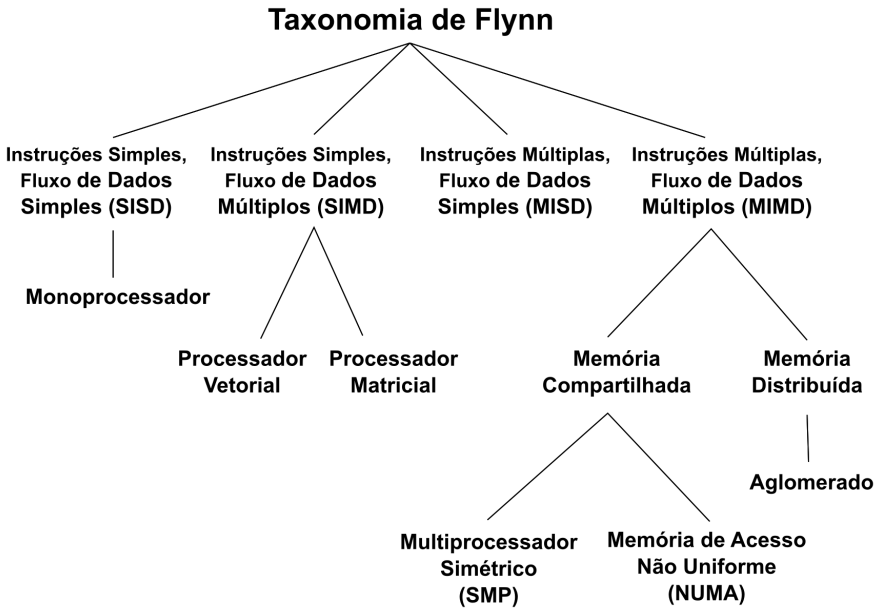
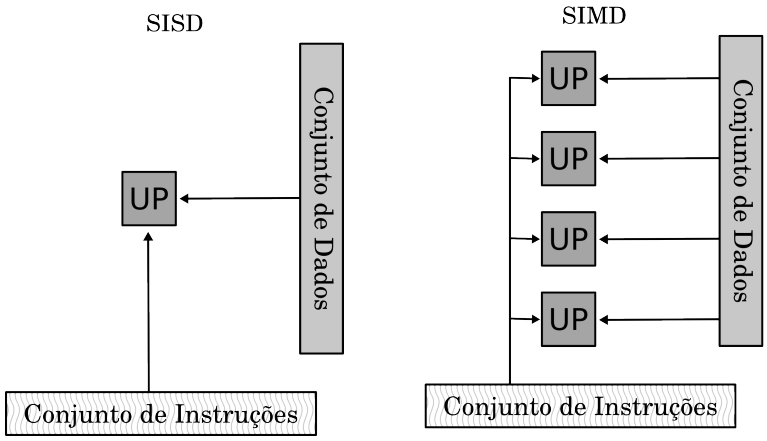


Figura 2.1: *Taxonomia de Flynn (adaptado de [1]).*

- *Single Instruction, Single Data Stream* (SISD): este modelo, representado na Figura 2.2a, utiliza uma *Unidade de Processamento* (UP) para executar uma instrução por vez. Nos computadores e *mainframes* antigos, apesar de alguns possuírem múltiplas unidades e processamento, os dados e os fluxos de instruções não tinham qualquer relação ou dependência. Em outras palavras, é um computador que realiza processamento sequencial, baseado no modelo de von Neumann ou ainda, na máquina universal de Turing [3]. O desempenho dos computadores que utilizam esta arquitetura, geralmente, é medido em *Million of Instructions per Second* (MIPS), e o número de instruções enviadas para processamento é limitado no tempo [21, 22];

- *Single Instruction, Multiple Data Stream* (SIMD): supercomputadores vetoriais são baseados no modelo SIMD e permitem a paralelização da instrução, otimizando o tempo de processamento de um conjunto diferente de dados, conforme ilustrado na Figura 2.2b. O desempenho é medido em *Millions of Floating Point Operations Per Second* (MFLOPS);

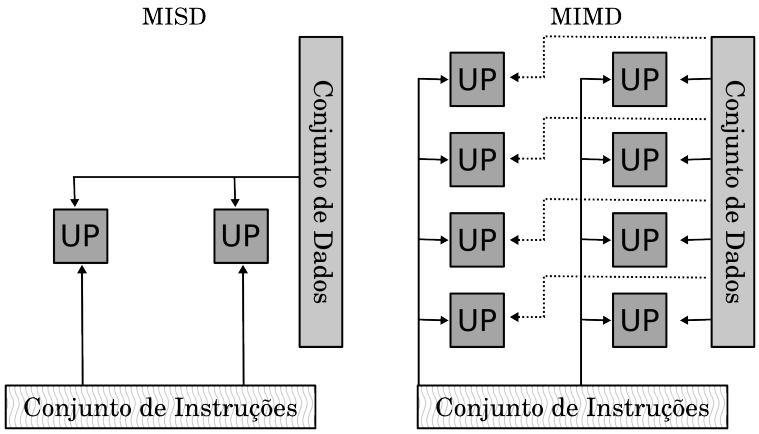


(a) *SISD* **(b)** *SIMD*

Figura 2.2: *Modelo de arquiteturas para fluxos de dados múltiplos (adaptado de [2]).*

- *Multiple Instruction, Single Data Stream* (MISD): conforme apresentado na Figura 2.3a, pode ser aplicado na resolução de problemas específicos, para garantir o processamento de cálculos matemáticos sólidos, sendo que os dados e as instruções são executados em unidades de processamento, necessariamente, diferentes [21]. Como exemplo conceitual pode-se citar, o processamento de vários algoritmos de criptografia tentando quebrar uma única mensagem codificada; e

- *Multiple Instruction, Multiple Data Stream* (MIMD): neste modelo apresentado na Figura 2.3b é possível realizar o paralelismo de instruções independentes para cada processador, com a vantagem de executar múltiplas tarefas simultaneamente. No entanto, há a necessidade de sincronizar em tempo real a execução das tarefas paralelas entre os processadores no final de uma aplicação simples [21].



(a) *MISD* **(b)** *MIMD*

Figura 2.3: *Modelo de arquiteturas para fluxos de instruções múltiplas (adaptado de [2]).*

O sincronismo entre os processadores é uma forma de comunicação, que pode ser obtida através do barramento de memória compartilhada ou em barramento de rede local. A maioria dos computadores modernos enquadram-se nesta categoria, os quais utilizam a programação paralela [27].

A comunicação realizada entre os processadores, de acordo com o processamento por evento de comunicação, é classificada em "granularidade". Granularidade fina, para pequenas quantidades de tarefas executadas ou sincronizadas com menor frequência; e grossa, para grandes quantidades de tarefas executadas ou sincronizadas com maior frequência [3, 27].

O modelo MIMD divide-se em multiprocessamento fortemente e fracamente acoplados. Na presente pesquisa foram levados em consideração apenas os sistemas fortemente acoplados, sendo esta a categoria para a arquitetura de GPGPU. Utilizar mais processadores, paralelizando a resolução dos problemas já citados, é uma prática que tem conquistado bons resultados. O desempenho da paralelização de um algoritmo pode ser medido através da eficiência e do *speedup* [28, 29].

O *speedup* é o ganho de desempenho na execução, em número de vezes, de um processo em n processadores em relação ao tempo de execução do processo, T_p, correspondente sequencialmente, em 1 processador, calculado como:

$$S(n) = \frac{T_p(1)}{T_p(n)} \tag{2.1}$$

A eficiência do algoritmo paralelizado pode ser calculado pela relação entre o *speedup* e o número de processadores efetivamente utilizados. Esta medida indica quando um processador esté sendo subutilizado quando o valor resultante for abaixo de 1, ou se o problema precisa ser decomposto de uma maneira mais eficiente para utilizar mais processadores para um valor resultante acima de 1. Segundo a Lei de Amdahl e como ilustrado na Figura 2.4, aumentar arbitrariamente o número de processadores num sistema paralelo não garante o aumento constante do *speedup* [26, 27].

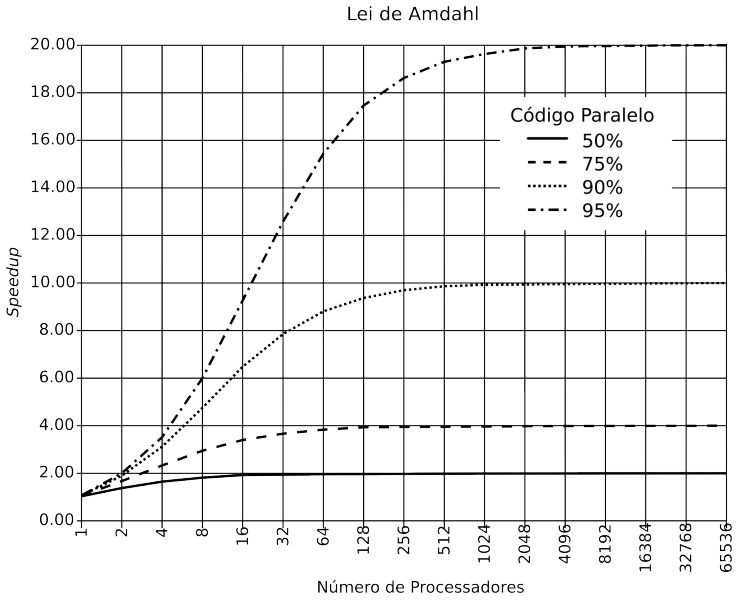


Figura 2.4: *Lei de Amdahl (adaptado de [3]).*

Ao aumento arbitrário no número de processadores atribui-se o termo Escalabilidade, definido para representar o crescimento proporcional da capacidade de processamento paralelo, com a inclusão de novos processadores no sistema [26].

Baseado na Lei de Amdahl [30], de acordo com a porção de código otimizada f por um *speedup* S, o ganho de desempenho máximo é definido como:

$$Speedup_{enhanced}(f,S) = \frac{1}{(1-f)+\frac{f}{S}} \tag{2.2}$$

Na procura de maior desempenho, sistemas com *multicores* e *manycores*, tem sido produzidos pela indústria de processadores. Na Figura 2.5, são apresentadas diferenças estruturais que influenciam, não só o desempenho, mas também o consumo de energia [4].

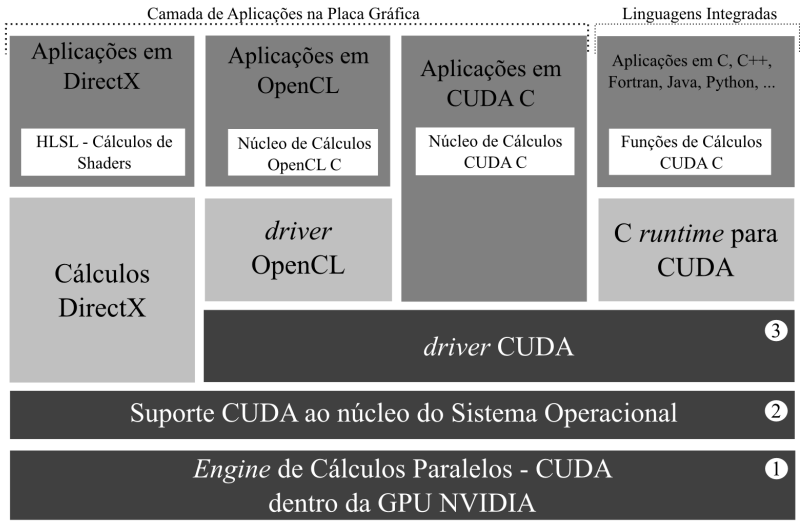


Figura 2.5: *Diferenças entre: CPU (multicores) e GPGPU manycores (adaptado de [3]).*

A *Central Processing Unit* (CPU) utiliza grande parte dos seus transistores para controle e cache, enquanto a GPGPU utiliza para cálculos. Os sistemas *multicores* possuem um bom desempenho e pouco consumo de energia, apesar dos *cores* serem projetados para baixo desempenho. Os seus processadores são interconectados por um único barramento ou um *Network-on-Chip* (NoC) [3, 21].

No caso dos multiprocessadores, os *chips* são separados e interconectados por barramento, permitindo que cada *chip* multiprocessado seja um *chip multicore*. Os processadores que compõem estes *chips* são de alto desempenho, com isso, influencia diretamente o poder computacional, entretanto, possuem o consumo de energia mais elevado em relação ao sistema *multicore* [22].

As GPGPUs são baseadas no modelo conhecido como *Single Program Multiple Data Stream* (SPMD) e permitem manipular um número muito grande de *threads* ao mesmo tempo [21, 22]. *Threads* são fluxos de instruções que permitem compartilhar espaço de endereçamento, temporizadores, arquivos, e podem ser executados simultâneamente, ou ainda, concorrentemente [31, 1, 32].

A GPGPU, escopo da presente pesquisa, pode ser classificada num modelo do tipo SIMD e MIMD, em razão de ser capaz de executar conjuntos de instruções funcionais num fluxo de dados por meio do *kernel*, conhecido como Modelo de Programação em Fluxo ou *Stream Programming Model* (SPM). *Kernel*, neste contexto, são funções aplicadas em cada elemento do conjunto de registros que exigem processamento similiar, definidos como fluxo ou *stream* [4].

As GPGPUs, por meio do *Stream Multiprocessor* (SM), vêm sendo cada vez mais

aplicadas à computação de propósito geral e intensivo, tornando-se desta maneira verdadeiras GPGPUs. As principais características das GPGPUs descritas por Gebali [22], tais como intensidade de cálculos, paralelismo de dados e localidade temporal e espacial, são apresentadas em detalhe no Capítulo 3. As GPGPUs passaram por um processo de evolução até atingir a atual capacidade de processamento. Esta evolução é descrita na próxima seção.

2.2 Histórico dos *multicores* e *manycores*

A partir do surgimento dos primeiros computadores pessoais na década de 80, denominados *Personal Computer* (PC), a indústria de computadores tem aumentado o desempenho destes equipamentos, principalmente, incrementando a velocidade da CPU.

A velocidade de processamento dos primeiros PCs era na frequência de 1MHz, já atualmente, é cerca de 1000 (mil) vezes mais rápido, alcançando *clocks* entre 1GHz e 4GHz. O crescimento na velocidade de processamento, conhecido desde 1965 como a Lei de Moore [27], enfrenta dificuldades como limitações físicas no tamanho de transistores, consumo de energia e dissipação de calor.

Uma das alternativas encontradas pela indústria de processadores, a partir de 2005, foi oferecer soluções com o processamento de várias CPUs num mesmo *chip*, conhecidos como *multicores* [3]. Juntamente com os *multicores*, o conceito de computação paralela tem sido disseminado ao consumidor doméstico como a solução para realizar várias tarefas simultaneamente, sem perder desempenho [33]. De qualquer maneira, é importante salientar a necessidade de reformulação do código da aplicação para obter aproveitamento da arquitetura *multicore*.

As primeiras *Graphics Processing Unit* (GPUs) foram lançadas, quase que juntamente, com os primeiros PCs. Entre as décadas de 80 e 90, diferentes fabricantes de placas gráficas surgiram no mercado. A primeira geração de processador gráfico, ainda nos anos 80, era capaz de mostrar textos e gráficos, tinha a capacidade de armazenamento em memória de apenas 16KBytes, além de representar cerca de 16 cores. Neste período todo o processamento era realizado diretamente na CPU.

Com o avanço dos sistemas operacionais no começo da década de 90, apresentando interfaces gráficas em 2D, surgiu uma procura para um novo mercado de processadores gráficos. Neste contexto, a Silicon Graphics criou a biblioteca *Open Graphics Library* (OpenGL) para permitir interface com o *hardware* de aceleração gráfica. Durante este período, começaram a ser desenvolvidas aplicações gráficas em 3D, especialmente jogos como Doom, Duke Nukem 3D e Quake [34].

Outras empresas iniciaram as suas atividades na produção de placas gráficas aceleradoras para aplicações 3D, como por exemplo a *International Business Machines Corporation* (IBM), primeira fabricante a oferecer aceleração gráfica *bidimensional* (2D) e *tridimensional* (3D), a S3, a NVIDIA a qual se tornou uma das principais fabricantes do setor e disputa a liderança de mercado com a ATI Technologies, 3dfx Interactive, dentre outras, contribuindo para a consolidação destas aplicações.

O termo GPU é empregado para o primeiro *chip* com funções integradas de realizar transformação, iluminação, recorte a ajuste de triângulos, texturização e renderização de imagens. A única maneira do programador interagir com a GPU era por meio das bibliotecas programáveis como OpenGL, desenvolvida pela Silicon Graphics, e DirectX, desenvolvida pela Microsoft.

Com o propósito de permitir maior controle de desenvolvedores sobre o processamento realizado na GPU, a NVIDIA incorporou o DirectX 8.0 nos seus *chips* [33, 34]. A segunda geração de GPU oferecia funções de transformação geométrica, aplicação de iluminação, e principalmente o processamento paralelo, inovado com a empresa 3dfx.

Todas estas funções eram realizadas diretamente no *hardware* da GPU, reduzindo a carga de processamento da CPU [34]. Continuando o processo de evolução das GPUs, surge o suporte para transformação de informações enviadas pela CPU, chamada de *vertex shaders*. Com isso, operações como transformações em vértices, iluminação, geração de coordenadas de textura, dentre outras, poderiam ser modificadas, marcando assim a terceira geração das GPUs.

A quarta geração é complementada com o recurso de *fragment shader*, o qual permite criar efeitos e transformações diretamente nos pixels antes de serem desenhados na ecrã do monitor. O *fragment shader*, também conhecido como *pixel shader*, atua em segmentos rasterizados permitindo o cálculo de cores, acesso e aplicação de texturas, cálculo de coordenadas de textura por pixel, além de outros efeitos. Pesquisadores passaram a realizar operações utilizando a unidade de processamento gráfico por meio da *Application Programming Interface* (API), conhecida como *shading language* [28].

Conhecer bem a linguagem *shading*, dominar computação gráfica, além da dificuldade de manipulação de dados em ponto flutuante e monitorar se os cálculos eram finalizados sem falhas, foram algumas das principais limitações de uso destes processadores para uso geral [33].

Diante das limitações, a NVIDIA desenvolveu a *Compute Unified Device Architecture* (CUDA), seguindo padrões de requisitos do *Institute of Electrical and Electronics Engineers* (IEEE) na construção da *Arithmetic and Logic Unit* (ALU). Dessa maneira, a unidade aritmética de ponto flutuante de precisão única foi projetada para utilizar um conjunto de instruções adequado para uso geral e não apenas processamento gráfico [33].

A Figura 2.6, ilustra a arquitetura modular CUDA, apresentando no módulo 1 a GPGPU com suporte CUDA, no módulo 2, o suporte de inicialização e configuração CUDA para o sistema operativo, e módulo 3 o *driver* CUDA.

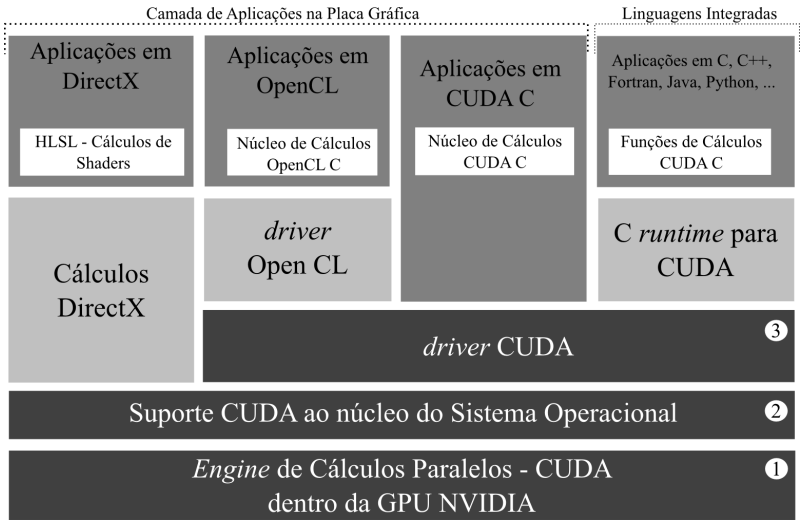


Figura 2.6: *Arquitetura CUDA (adaptado de [4]).*

O desenvolvimento de aplicações para executar nestes processadores gráficos, apesar de utilizar a arquitetura CUDA, exige conhecimentos avançados em programação OpenGL ou DirectX. No intuito de estender o recurso para um número maior de desenvolvedores, a NVIDIA incorporou uma pequena quantidade de *keywords* na linguagem C padrão adicionando funcionalidades específicas da arquitetura CUDA, criando desta maneira um compilador para esta linguagem, denominado CUDA C [33, 3, 34].

2.3 Considerações Finais

Os computadores multiprocessados ou *multicores*, na maioria das vezes não executam aplicações completamente paralelas em todas as suas unidades de processamento, em razão de implementações sequenciais destas aplicações. Há décadas grandes empresas e instituições de pesquisas utilizam computadores de grande porte paralelizando aplicações que exigem alto desempenho computacional. No entanto, a aquisição deste tipo de equipamento exigia investimento de alto custo.

Desta maneira, com a popularização dos computadores *multicores* e a possibilidade de melhorar o desempenho das aplicações, o desenvolvimento de aplicações paralelas têm aumentado significativamente [3]. Assim, as GPGPUs tê vindo a ser utilizadas em diversas pesquisas, e em alguns casos como infraestrutura para computação de alto

desempenho. No próximo capítulo são apresentadas as técnicas de programação paralela mais significativas destinadas às GPGPUs.

Capítulo 3

Técnicas de Computação Paralela

Diversas áreas da Ciência utilizam a computação de alto desempenho para resolver problemas cada vez mais complexos. Muitos destes problemas envolvem cálculos que exigem alta precisão e rapidez no processamento dos dados, o que pode ser alcançado por meio da paralelização dos algoritmos e execução em vários processadores. Neste capítulo são apresentadas algumas das técnicas e bibliotecas de programação paralela mais significativas para esta pesquisa, enfatizando a arquitetura CUDA.

3.1 Introdução

A computação paralela, definida na Seção 2.1, tem como objetivo minimizar a carga de operações por processador e maximizar o *speedup* do processamento como um todo.

O processo de paralelização de um algoritmo, inicia-se com a decomposição de problemas maiores em tarefas. Em seguida as tarefas são distribuídas entre os processadores exigindo do programador, dentre outras ações, codificar a aplicação para garantir independência de dados e realizar o controle de sincronização ou de comunicação entre as UP (Unidades de Processamento), em busca de maior eficiência na paralelização das tarefas [27].

3.1.1 Decomposição

De acordo com Vadja [5], a técnica de decomposição pode ser classificada em funcional, baseada em dados e baseada em tarefas. Na decomposição funcional, é possível uma UP continuar realizando as suas operações sem a necessidade de aguardar o resultado de processamento em outras UPs. Ao utilizar esta técnica, a aplicação pode ser implementada determinando, especificamente, as partes do código a serem paralelizadas.

Dessa maneira, a decomposição funcional pode ser subdivida em estática e dinâmica. Na paralelização de código, quando é definido um número específico de UP, a técnica é conhecida como decomposição funcional estática, sendo a sua principal característica a falta de escalabilidade. No caso da decomposição funcional dinâmica, a aplicação é independente da quantidade de processadores, permitindo escalabilidade e *speedup* ao sistema computacional [5].

Para a decomposição baseada em dados, a paralelização em tempo de execução, inicia-se com a distribuição de processos, ou *threads*, entre as UPs disponíveis para o processamento de conjuntos de dados diferentes e não dependentes [24]. No modelo *pipeline*, os dados são decompostos em etapas diferentes e cada UP realiza o processamento especializado destas etapas. No entanto, pode ser que algumas das UPs executem o processamento da mesma etapa. Na Figura 3.1, as etapas são representadas como *Estágio* e as UPs, como *núcleos*.

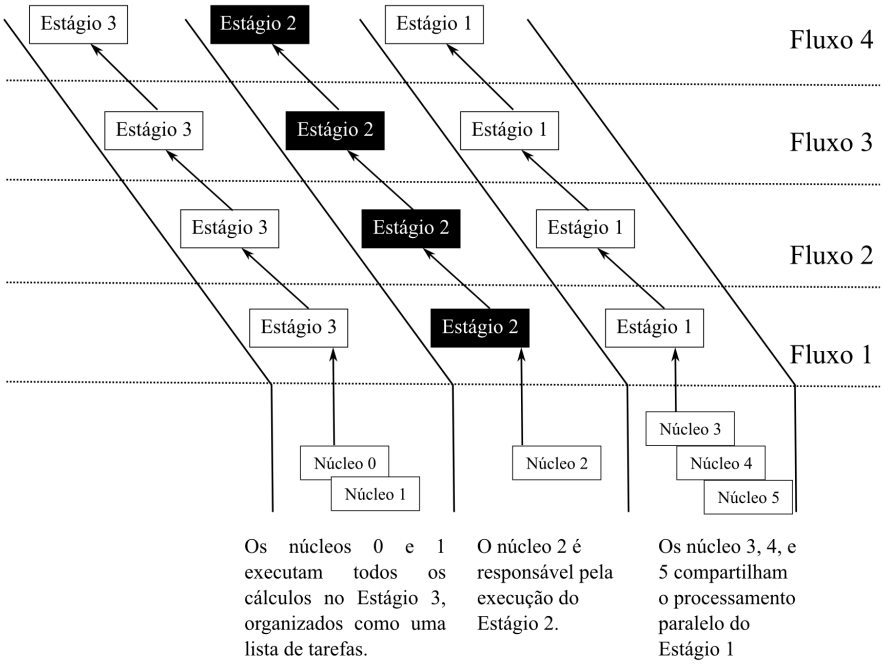


Figura 3.1: *Paralelização utilizando modelo pipeline (adaptado de [5]).*

As categorias do modelo apresentado anteriormente, podem ser inadequadas para problemas que exigem dependência de dados e processos. Para permitir o paralelismo eficiente nesta categoria de problemas, Vadja [5] considera promissora a utilização de técnicas de previsão na busca e processamento de dados.

A implementação de técnicas de decomposição de tarefas, deve levar em consideração o funcionamento básico de processos e *threads*, ao nível de sistema operativo. Conceitualmente, processos podem conter *threads*, desta maneira, caracteriza-se a dependência

hierárquica entre eles. Os *threads* compartilham recursos como espaço de endereçamento, variáveis globais, arquivos, dentre outros. Por outro lado, os processos são completamente isolados, sendo considerados pelo sistema operacional como unidades de isolamento permitindo gerenciamento de falhas.

O sistema operacional gerencia o acesso aos recursos computacionais e utiliza o conceito de prioridade, baseado na associação entre processos e *threads*, como pode ser observado na Figura 3.2.

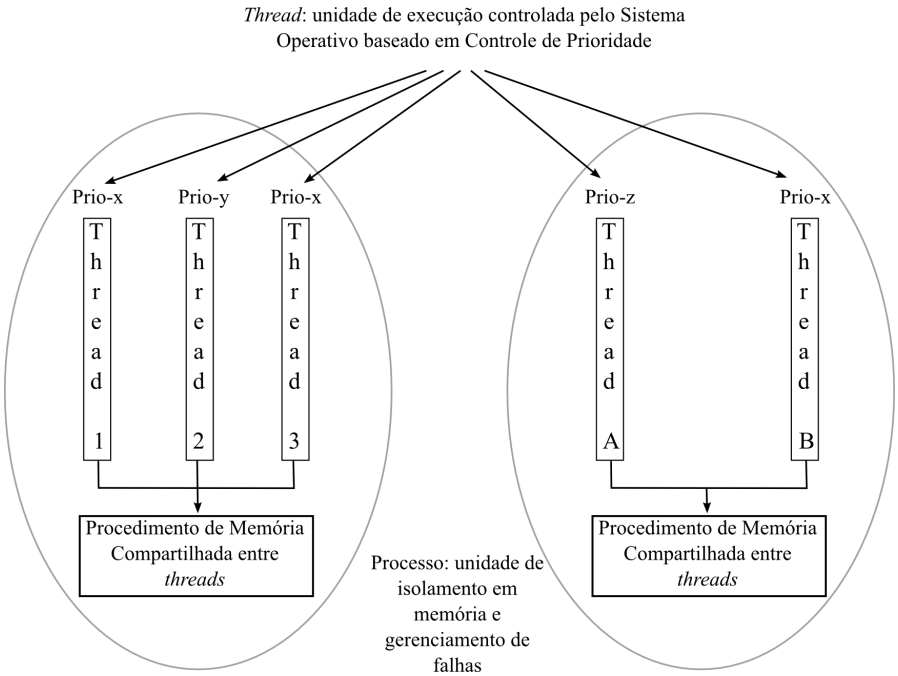


Figura 3.2: *Conceitos de processos, threads e prioridades (adaptado de [5]).*

3.1.2 Sincronização

Aplicações paralelas, em geral, exigem interações na troca de informação, acesso aos mesmos dados, espera no término de processamento, dentre outras formas. O uso destas interações é facilitado por meio de mecanismos de sincronização como *locks*, semáforos, regiões críticas, dentre outros.

Durante a execução de uma aplicação paralela, uma UP pode precisar de resultados em processamento noutra UP. Esta dependência de dados, ou dependência funcional, exige que a aplicação aguarde o processamento das UPs envolvidas e então, posteriormente, realize o sincronismo para concluir a tarefa. Numa outra situação, partes da aplicação paralela podem realizar acessos frequentes a recursos do sistema, como *hardware*, acesso de memória compartilhada, dentre outros. Em ambos os casos, a decomposição exige a necessidade de sincronizar as diferentes tarefas paralelizadas.

3.2 Métodos de Paralelização em GPGPU

A maneira como pode ser realizada a comunicação entre as UPs pode ser dividida em sistemas fortemente e fracamente acoplados. A ênfase nesta pesquisa, como mencionado anteriormente, é em sistemas fortemente acoplados. Dessa maneira, o *Symmetric Multiprocessor* (SMP), consiste num conjunto de processadores idênticos que podem executar tarefas em cooperação para resolver problemas científicos utilizando o compartilhamento de memória [31].

Sistemas computacionais compostos por SMP, também são conhecidos como computação distribuída, grade computacional ou computação massivamente paralela. As UPs envolvidas na execução das tarefas comunicam-se para trocar informações acerca do processamento, além de realizar operações, como por exemplo, atualização dos dados armazenados em memória.

As técnicas de paralelização mais comuns aos tipos de arquiteturas *multicores* e GPGPU (*manycores*) [4, 3], são: paralelismo de tarefas, o qual realiza a execução simultânea de múltiplas tarefas, que podem ser *threads* ou processos, em dados diferentes. O paralelismo de dados, onde múltiplos dados recebem a execução de uma tarefa e o paralelismo de instruções, quando há a execução de múltiplas instruções por processador.

A aplicação destas técnicas de paralelização exige o suporte específico do *hardware*. Kirk and Hwu [3], levantaram as diferenças entre os projetos da CPU e GPGPU. O projeto da CPU é otimizado para desempenho de código sequencial, utilizando sofisticadas lógicas de controle para permitir que instruções de uma única *thread* executem em paralelo; a velocidade de acesso à memória não aumenta muito em razão da utilização do processador para propósito geral, satisfazendo dispositivos de entrada e saída, sistemas operativos e aplicativos.

O projeto da GPGPU prioriza a realização de uma enorme quantidade de cálculos em pontos flutuantes, disponibilizando pequenas unidades de memória *cache* para facilitar o controle de requisitos de largura de banda. Permitem ainda, que múltiplos *threads* acessem os mesmos dados na memória, dispensando a necessidade de recuperação constante de dados na memória principal, sendo que a velocidade de acesso à memória é, em geral, 10 vezes mais rápida em processadores gráficos [3].

Diante das diferenças de *hardware* apresentadas anteriormente, a Figura 3.3 ilustra a evolução na capacidade de cálculos de pontos flutuantes entre CPUs e GPGPUs, um dos principais requisitos na exigência por processamento de alto desempenho.

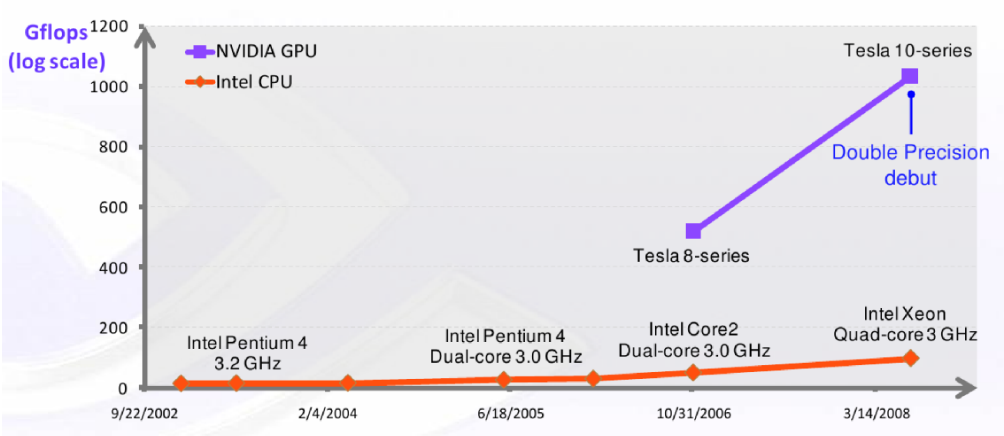


Figura 3.3: *Evolução de Operações em Ponto Flutuante em CPUs e GPUs (adaptado de [4]).*

A partir destas considerações acerca das GPGPUs, ressaltam-se as principais características sobre o modelo SPM, denominado pela NVIDIA como *Single Instruction, Multiple Thread* (SIMT). Neste modelo, os dados são representados como um *stream*, ou seja, um fluxo de dados é classificado como um conjunto, e estruturado num *array*. A execução de uma instrução é aplicada em um conjunto de dados.

Os *kernels* executam operações paralelamente em todo o *stream*, utilizando-o como dado de entrada e saída. As operações de gestão podem ser divididas em cópia, indexação e decomposição de subconjuntos, ou podem ser calculados paralelamente utilizando *kernels* [3, 4].

No modelo SIMT, a chamada de uma variedade de *kernels* é organizada como uma hierarquia de grupo de *threads*. O recurso de dividir *kernels* em blocos independentes, bem como o suporte eficiente aos *threads* em GPGPU garante escalabilidade transparente e portável, permitindo um programa em CUDA executar em qualquer número de *cores*. Os *threads* são utilizados para paralelismo de granularidade fina e são agrupados formando blocos. Os blocos são utilizados para paralelismo de granularidade grossa e são agrupados formando uma grade, a qual representa uma chamada de *kernel* na placa gráfica. Como ilustrado na Figura 3.4, esta hierarquia permite que cada *thread* dentro de um bloco, assim como, cada bloco dentro de uma grade, tenham um índice de identificação único [35].

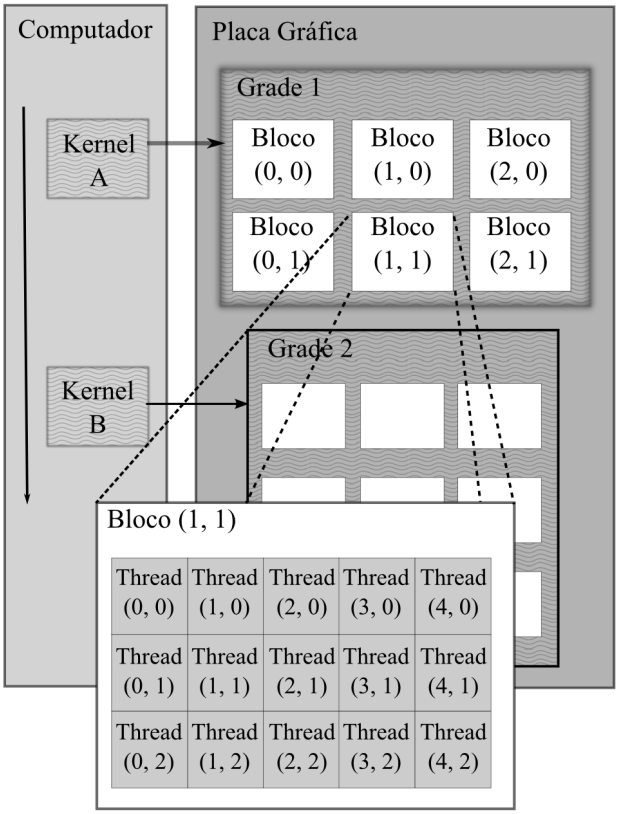


Figura 3.4: *Single Instruction, Multiple Thread (adaptado de [4]).*

Neste modelo, os *threads* de um bloco podem ser organizados em três dimensões para executar num único multiprocessador na GPGPU. Também podem ser sincronizados e compartilhar dados com outros *threads* do mesmo bloco por meio de memória compartilhada, já que este tipo de comunicação é mais rápida do que através da memória global da placa gráfica utilizada na comunicação entre os blocos [3].

Na Figura 3.5, é apresentado um diagrama sobre a composição da arquitetura GPGPU desenvolvida pela NVIDIA, conhecida como *Compute Unified Device Architecture* (CUDA). A arquitetura CUDA é adequada para realizar as operações descritas no modelo anterior, e conforme ilustrado nesta figura, a arquitetura consiste de *Stream Multiprocessor* (SM), podendo acoplar até dezenas destes multiprocessadores. Para cada SM, ilustrado na Figura 3.5 (b), é possível observar o agrupamento de *Streaming Processors* (SP) ou CUDA *cores*. As unidades de *load/store* (LD/ST) permitem que os endereços "origem" e "destino" sejam calculados para 16 *treads* por *clock*. Instruções do tipo *seno, cosseno e raiz quadrada* são executadas em unidades de funções especiais, identificadas na Figura por *Special Function Units* (SFU). Cada SM possui 4 unidades SFU e organiza as *threads* em grupos de 32 unidades paralelas, que são definidos como *warps*. Cada SM

pode executar até 2 *warps* concorrentemente, através de 2 controladores *warp* e 2 unidades de envio. A Figura 3.5 (c) ilustra o CUDA *core*, o qual é composto por uma unidade lógica aritmética (INT) e uma unidade de ponto flutuante (FP).

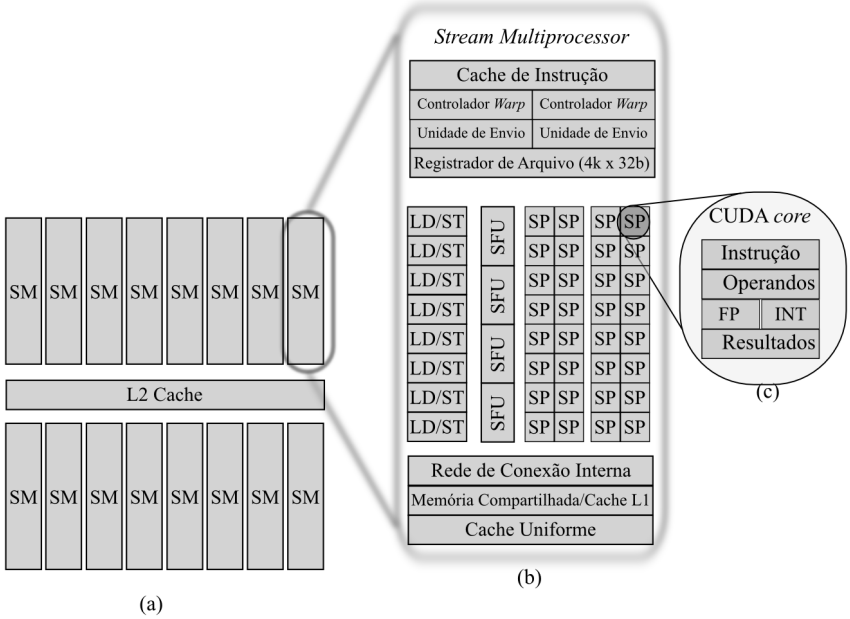


Figura 3.5: *Visão simplificada da GPGPU Fermi (Nvidia) (adaptado de [6]).*
(a) Diagrama de bloco do *stream multiprocessor*. b) Diagrama de bloco do Processador de *thread/stream*. c) Diagrama de bloco do Processador CUDA *core*.

O suporte em linguagens de alto nível para desenvolvimento em CUDA utilizam o estilo tradicional da linguagem C/C++, dessa maneira utilizam fluxos de controle como *if-then-else*, *for* e *while*. A implementação destes controles são diferenciadas de acordo com a arquitetura do processador, GPGPU ou CPU.

Como visto anteriormente, uma parte do processador executa operações baseadas no modelo SIMD durante várias partes do tempo, dessa maneira caso porções de código em execução utilizem *branch* diferentes, ambos serão processados. Esta é uma das situações de redução de desempenho para programas com implementações de *branchs*. *Branch* são ramificações no código de implementação fazendo com que dois caminhos de execução sejam sempre verificados, como por exemplo em controles do tipo *if-then-else*.

Os mecanismos de controle utilizados em CUDA, de maneira geral, implementam MIMD *branching*, SIMD *branching* e previsão de desvios. MIMD *branching* permite que diferentes processadores executem diferentes ramificações dependentes de dados sem penalidades, similar ao que acontece em CPU. SIMD *branching* exige que todos

processadores executem instruções idênticas, por outro lado, ramificações divergentes podem ocasionar redução de desempenho [4].

O mecanismo de previsão de desvios é empregado na avaliação de ramificações, e de acordo com a condição identificada, é atribuído um marcador de condição. Este marcador será verificado antes do processador percorrer a próxima ramificação. O custo de processamento de todas as ramificações será o mesmo para cada parte da ramificação, somado ao custo de avaliação do ramo [27].

3.3 Aspectos de Programação Paralela

Baseando-se em arquiteturas paralelas e técnicas de computação paralela, linguagens de programação paralela foram desenvolvidas a fim de permitir interação na utilização destes recursos e aplicação em áreas de interesse comercial ou científico.

Kirk and Hwu [3] classificam os modelos de programação paralela em interação de processos e decomposição de problemas. De acordo com os mecanismos de comunicação entre os processos paralelos, o modelo chamado interação de processos pode ser dividido em: memória compartilhada, troca de mensagens e implícita.

O modelo de decomposição de problemas caracteriza a maneira como os processos são formulados para o processamento paralelo, sendo este dividido em paralelismo de tarefas, de dados, e implícito. Modelos de computação paralela que exigem troca de mensagens entre processadores e sistemas multiprocessados de memória distribuída, podem ser melhores explorados por meio da linguagem de programação *Message Passing Interface* (MPI), e *Open Multi-Processing* (OpenMP) ou PThreads para sistemas multiprocessados de memória compartilhada. Seguem algumas das principais características destas linguagens [3]:

- Aplicações implementadas em MPI, podem executar em mais de 100.000 processadores de um sistema em *cluster*, no entanto, pode ser extremamente custoso implementar modificações para executar a aplicação em sistema de memória compartilhada.

- A linguagem OpenMP, é considerada ideal para sistemas de memória compartilhada, porém, dificuldades na gestão de *overhead* limitam a sua escalabilidade para algumas centenas de processadores. O OpenMP foi originalmente desenvolvido em Fortran, sendo modificado para C/C++ posteriormente.

- O POSIX Threads, normalmente identificado apenas como Pthreads atende aos padrões definidos pelo *Portable Operating System Interface* - (POSIX) para progra-

mação com *threads*. A biblioteca Pthreads é um conjunto de procedimentos desenvolvidos em C que fornece um mecanismo para codificação *multithread*.

A linguagem CUDA C, desenvolvida especialmente para GPGPUs, não possui os problemas e limitações apresentadas pelas linguagens descritas anteriormente. Como vantagem em relação ao MPI, permite a comunicação entre GPGPU e CPU utilizando troca de mensagens unilateral. Em comparação com o OpenMP, possui alta escalabilidade e baixo *overhead* no gerenciamento de *threads*. No entanto, não oferece suporte à variedade de aplicações as quais o OpenMP suporta com sucesso.

Com o objetivo de padronizar o modelo de programação para GPGPU, alguns fabricantes como Apple, Intel, AMD, e a própria NVIDIA, que desenvolveu a arquitetura CUDA, uniram-se no desenvolvimento do modelo de programação denominado *Open Computing Language* (OpenCL). Com a padronização, pretende-se permitir que as aplicações desenvolvidas utilizando este modelo, possam executar em quaisquer processadores com suporte às extensões e API da linguagem, além de garantir ao programador a gestão do paralelismo em processadores massivamente paralelos.

As linguagens CUDA e OpenCL possuem várias similaridades, diante disso, Kirk and Hwu [3] afirmam haver uma certa vantagem para a linguagem CUDA, em velocidade de execução quando o *hardware* oferece suporte para ambas as linguagens, e garantem também que a linguagem OpenCL exige desenvolvimento de aplicações em mais baixo nível em relação ao CUDA.

3.4 Considerações Finais

Diante da necessidade por processamento de alto desempenho, no processo de paralelização dos algoritmos deve-se levar em consideração a arquitetura de processadores para, posteriormente, definir-se a melhor estratégia de paralelização e linguagem de programação.

O objetivo é explorar ao máximo a capacidade de processamento da arquitetura selecionada, obtendo resultados satisfatórios. O emprego de técnicas de paralelismo, na maioria das vezes, visa alcançar resultados tais como a redução no tempo de processamento.

Capítulo 4

Revisão Sistemática de Literatura

O desenvolvimento de soluções computadorizadas capazes de realizar o processamento e a análise de imagem médica tem contribuído no avanço da medicina, de maneira geral. o número de pesquisas nesta área cresce em uma velocidade muito grande, inclusive em razão da disponibilidade de recursos computacionais mais rápidos e de baixo custo, como é o caso dos processadores *manycores* e *multicores*, tema abordado no Capítulo 2. O presente capítulo descreve em detalhes os procedimentos realizados durante a revisão de literatura, com o objetivo identificar sistemas processados por computação de alto desempenho capazes de realizar diagnóstico auxiliado por computador.

4.1 Introdução

O processo de revisão de literatura consiste na descrição de literaturas consideradas importantes para uma determinada área ou assunto de interesse. Existem diferentes formas de realizar a revisão de literatura [36], como por exemplo a revisão de literatura tradicional e a revisão sistemática de literatura. Uma revisão de literatura, é considerada tradicional quando o objetivo desta revisão é realizar um levantamento sobre pesquisadores, teorias e hipóteses, bem como as metodologias utilizadas na resolução de problemas [37]. Normalmente, estas informações são concentradas na forma de relatórios.

A Revisão Sistemática de Literatura (RSL), consiste em um conjunto de procedimentos que permite examinar sistematicamente a literatura de interesse, e assim *identificar*, *avaliar* e *interpretar* o estado da arte de pesquisas relevantes para a área de interesse. Estes procedimentos permitem que outros possam reproduzir os métodos e determinar objetivamente se aceitam o resultado de sua revisão. A RSL é composta pelas etapas de *a)* Planeamento, *b)* Execução e *c)* Apresentação de Resultados. Este tipo de revisão não requer que sua execução seja realizada em apenas uma vez, desta maneira, as etapas po-

dem ser revisadas, refinadas e ajustadas, mesmo com a revisão em andamento. Cada uma das etapas contidas no processo de RSL, é descrita na seção seguinte.

4.2 Aplicação da Revisão Sistemática

A presente RSL tem como objetivo identificar na literatura científica especializada, *Sistema de Detecção de Doença Auxiliado por Computador* (SDDAC) processados por computação de alto desempenho e sejam aplicados a imagens da cavidade pélvica feminina.

4.2.1 Planeamento

A partir da definição do objetivo de realizar a RSL, é necessário definir as questões de pesquisa, pois implicam na principal atividade do processo de revisão como um todo. São as questões de pesquisa que direcionam os procedimentos de extração e síntese de informação da literatura identificada, bem como possui grande relevância na delimitação do escopo de pesquisa [37]. Inicialmente, a questão de pesquisa definida foi a seguinte: **Quais estratégias e plataformas de computação de alto desempenho podem ser aplicadas no processamento e análise de imagens complexas da cavidade pélvica feminina?** A partir deste questionamento, surgiram outras questões para complementar a qualidade final da revisão:

1. Qual a modalidade de imagiologia utilizada?
2. Qual o tipo de patologia o sistema detecta?
3. Qual arquitetura computacional exigida?
4. Qual estratégia de computação de alto desempenho adotada/desenvolvida?
5. O sistema realiza detecção em tempo real?
6. Qual o tipo de licença do sistema?

A fase seguinte consiste em definir o Protocolo que estabelece os critérios de seleção, inclusão e exclusão de literatura no processo de RSL, bem como palavras-chave, motores de busca, e até definir o idioma de interesse. O Protocolo é utilizado no restante processo de revisão. A primeira estratégia definida foi considerar apenas publicações escritas em língua inglesa, visto que abrange o maior número de produções científicas na área da computação e engenharia. Em seguida, definiu-se realizar a busca por publicações nos

motores de busca: CiteSeerX [1], ScienceDirect [2], IEEE Xplore [3], ACM Portal [4] e Medline [5]. A identificação da publicação iniciou-se com a busca das palavras-chave: *medical image, high performance computing, image processing* nos motores de busca.

A análise das publicações identificadas pelos motores de busca, de maneira geral, leva em consideração a ocorrência das palavras-chave indicadas, no campo *keywords*, título e *abstract*. Contudo, as publicações devem ser analisadas individualmente para identificar se atendem ao critério de seleção, inclusão e exclusão da literatura. As publicações identificadas pelos itens relacionados a seguir, atendem ao critério de seleção:

1. Domínio de Aplicação
   (a) Imagens médicas; ou
   (b) Cavidade pélvica feminina.
2. Métodos e Técnicas
   (a) Processamento e análise de imagens médicas;
   (b) Computação de alto desempenho.
3. Métricas
   (a) Precisão desempenho das técnicas/métodos de imagens adotados/desenvolvidos;
   (b) Desempenho em tempo de processamento (*speedup*).

A partir da definição dos critérios de seleção de publicações, foram estabelecidos os critérios para excluir do processo de revisão, listados a seguir:

1. publicações com o mesmo título, selecionadas por motores de busca distintos;
2. falta de informação sobre as técnicas de processamento e análise de imagens empregadas na pesquisa;
3. falta de informação sobre o desempenho em tempo de processamento obtido pela solução proposta;
4. publicações com 1 página e pôster;

[1] http://citeseerx.ist.psu.edu
[2] http://www.sciencedirect.com
[3] http://ieeexplore.ieee.org
[4] http://dl.acm.org/
[5] http://www.ncbi.nlm.nih.gov/pubmed/

5. publicações que não descrevem a avaliação da solução proposta.

Não foram definidos os critérios de inclusão, pois foi considerado que as publicações que não atendem aos critérios de exclusão, automaticamente devem ser inseridas para a fase de extração de dados.

4.2.2 Execução

O protocolo descrito na seção 4.2.1, foi aplicado no mês de junho de 2014. Durante a etapa de Execução, os critérios de seleção, inclusão e exclusão, devem ser levados em consideração, contudo, sendo aplicados sistematicamente na introdução e conclusão, e, se necessário no restante da publicação, caracterizando a fase avaliação da publicação. Na Tabela 4.1, indica o número total de publicações obtidas de acordo com a busca nos respectivos motores de busca, indicando exatamente a expressão de caracteres utilizada:

Tabela 4.1: *Número de publicações Identificadas (Ident.), Selecionadas (Sel.) e Avaliadas (Aval.) na etapa de Execução.*

Motores de Busca	Publicações			
	Expressão	Ident.	Sel.	Aval.
CiteSeerX	"medical image"AND "image processing"AND "high performance computing"	174	11	07
ScienceDirect	"medical image"AND "image processing"AND "high performance computing"[Journals(Computer Science)]	124	12	02
IEEE Xplore	((("medical image") AND "image processing") AND "high performance computing")	108	11	03
ACM Portal	("medical image"and "image processing"and "high performance computing") and (PublishedAs:journal OR PublishedAs:proceeding) and (FtFlag:yes) and (AbstractFlag:yes)	51	10	02
Medline	"medical image"AND "high performance computing"	11	04	
Total		480	48	14

É importante destacar que somente os parâmetros de busca no motor de busca IEEE Xplore foram ajustados, de acordo com as seguintes informações:

- Publisher: IEEE
- Content Type: Conference Publications, Journals & Magazines Publication Title: Computer Graphics and Applications, IEEE, High Performance Computing and Simulation (HPCS), 2012 International Conference on, Information Technology and Applications in Biomedicine, 2009. ITAB 2009. 9th International Conference on, Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on, Computing in Science & Engineering, Computer, Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, Computational Biology and Bioinformatics, IEEE/ACM Transactions on, High Performance Computing, Networking, Storage and

Analysis (SCC), 2012 SC Companion, Image Processing, IEEE Transactions on, Medical Imaging, IEEE Transactions on, Information Technology in Biomedicine, IEEE Transactions on, Biomedical Engineering, IEEE Transactions on, Biomedical Imaging: From Nano to Macro, 2009. ISBI '09. IEEE International Symposium on, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Visualization and Computer Graphics, IEEE Transactions on, Parallel and Distributed Systems, IEEE Transactions on, High Performance Computing on the Information Superhighway, 1997. HPC Asia '97

Por fim, após a conclusão da fase de avaliação da publicação, a literatura deve ser analisada por completo. Somente após a análise completa, inicia-se a fase de extração dos dados de interesse. A fase de extração dos dados, consiste em oferecer suporte para a análise e síntese da informação coletada das publicações, para responder às questões de pesquisa.

A presente RSL encontra-se em andamento, desta maneira, destaca-se que até o momento foram avaliados um total de **14** publicações. **10** publicação foram removidas da seleção de acordo com os critérios de exclusão definidos anteriormente. Contudo, serão apresentados na próxima secão, a informação coletada das publicações analisadas até o momento.

4.2.3 Resultados

A Tabela 4.2.3 apresenta os dados extraídos das publicações avaliadas nesta revisão.

	modalidade das imagens médicas	**domínio de aplicação das técnicas**	**técnica de processamento e análise de imagem**	**plataforma/técnica(s) de alto desempenho**	**SDDAC**	**tempo real**	***speedup***	**licença**
Kikinis et al. [38]	MRI, CT	esclerose múltipla, tumor cerebral, análise óssea	classificação (k-NN, EM), registro linear (inter/intra-sujeito), registro não linear, geração do modelo de superfície, renderização de volume	*multi thread*, *clusters*, SIMD e NUMA	sim	sim	linear	não informado
Zhang et al. [39]	CT	cáries dentárias	Algoritmos tradicionais: histograma, operações morfológicas e construção 3D	MapReduce	sim	não	linear	não informado (baseado em Java)
Dasika et al. [40]	MRI, CT	Qualquer aplicação médica	Segmentação: Grafos, Laplaciana, Gaussiana	*Hardware* em FPGA	não	não	2 vezes	não informado
Stone et al. [41]	MRI	Qualquer aplicação médica	Transformada de Fourier	SIMD (CUDA em GPU)	não	não	13 vezes	não informado

O uso de imagens na medicina permite a extração de informação que pode ser aplicada no monitoramento, planeamento e tratamento de patologias. A extração de informação de imagens médicas, trata-se de um procedimento complexo e requer sistemas avançados, capazes de processar e obter diagnósticos precisos e com tempo de processamento reduzido. A solução proposta foi criar uma rede de comunicação interna baseada em *cluster*, utilizando as técnicas de paralelização consolidadas de memória distribuída, com MPI para processamento de algoritmos que requer maior poder computacional, e paralelização *multithread* em algoritmos considerados de menor exigência de poder computacional. Os componentes da segmentação de imagens abordados foram: a aquisição de imagens por CT e MRI; filtros adaptativos; classificação estatística (k-NN e EM); e modelagem anatômica explícita (*inter-subject*, *intra-subject* e *non-linear*); e a visualização com os métodos de geração de superfícies *marching cubes*, e renderização com o método *shear-warp*. Apenas os métodos *marching cubes*, EM e *non-linear* foram paralelizados utilizando a abordagem *multithread*, sendo os demais, paralelizados em MPI. De maneira geral, o tempo de processamento utilizando imagens de esclerose múltipla, tumor cerebral e análise óssea, obteve *speedup* linear. A contribuição da pesquisa foi demonstrar a possibilidade de combinar técnicas "convencionais"de processamento e análise de imagens médicas e computação de alto desempenho em tarefas rotineiras de um laboratório de planeamento cirúrgico [38].

O diagóstico de cárie dentária é um procedimento de avaliação rotineira que emprega, tradicionalmente, medidas subjetivas e incorpora inspeções visuais e táteis. Os avanços alcançados recentemente em processamento e análise de imagens e tomografia computadorizada de raio X microfoco, vem contribuindo para a detecção e tratamento das cáries. As imagens longitudinais, ou seja de cortes transversais, obtidas por tomografia são utilizadas para gerar modelos tridimensionais da arcada dentária do paciente, e posteriormente são analisados e comparados com exames anteriores para diagnosticar precisamente o grau da lesão. Este procedimento é muito caro computacionalmente, desta maneira, foram aplicadas técnicas de paralelização com o *framework* MapReduce, além de técnica para compressão de dados. A solução proposta foi dividida em duas partes: a paralelização compressão de dados; e a construção do modelo 3D. O método proposto foi capaz de comprimir os dados em 23% em relação aos espaço ocupado pelos dados originais, enquanto a construção tridimensional obteve um *speedup* linear até um total de 64 nós no *cluster* [39].

Os dispositivos utilizados na reconstrução de imagens médicas, comumente dotados de arquiteturas de processamento de alto desempenho, são considerados responsáveis por um alto consumo de energia elétrica. O objetivo é reduzir o consumo de energia elétrica durante o processo de reconstrução de imagens obtidas a partir de tomografia computado-

rizada e ressonância magnética. A solução proposta trata-se de um *hardware*, utilizando *Field-Programmable Gate Array* (FPGA), e é utilizado em comparações de desempenho/consumo de energia com GPUs da NVIDIA. Não foram realizados experimentos com imagens médicas de domínio específico. A arquitetura, explicando de maneira resumida, é composta de aceleradores programáveis de *loop*, que foram adicionados em *hardware* para cálculo de propósito geral, e realizam a distribuição da carga de processamento utilizando a metodologia de blocos, característico em multiplicação de matrizes. Os métodos de segmentação de imagens adotados e implementados na arquitetura desenvolvida, foram: segmentação baseada em grafos, filtro Laplaciano e convolução Gaussiana. A solução foi considerada capaz de realizar a reconstrução de imagens com melhor taxa de desempenho por consumo. O ganho de desempenho foi de até 2 vezes, em relação a GPUs modernas (2009) da NVIDIA. Quando levado em consideração a relação desempenho por consumo de energia, a diferença chega a 54 vezes [40].

Exames de ressonância magnética são comumente utilizados em medicina como um maneira não intrusiva de observar a estrutura e o funcionamento de órgãos em qualquer região do corpo. No primeiro momento, as amostras são capturadas no domínio do espaço e em uma trajetória pré-definida, por fim, as amostras são transformadas para reconstruir a imagem desejada. Este procedimento requer tempo de processamento na obtenção das imagens, além de ser altamente suscetível a ruídos. A solução proposta foi baseada na técnica de paralelização *Single Instruction, Multiple Data Stream* e na arquitetura CUDA, utilizando a capacidade de processamento da GPGPU. Apesar da arquitetura G80, disponível na época do desenvolvimento da pesquisa, não possuir suporte a números em ponto flutuante, o desempenho obtido nos experimentos permitiram reduzir o tempo de processamento da reconstrução de imagem de MRI em 20%, em comparação ao processamento realizado em CPU *quad-core* [41].

4.3 Considerações Finais

O uso de FPGAs pode permitir a customização do projeto de *hardware*, entretanto, não fornece flexibilidade em implementações baseadas em *software*, tornando sua contribuição questionável quando comparado com as modernas GPUs.

Técnicas de computação paralela são empregadas no intuito de explorar ao máximo a capacidade de processamento de arquiteturas *multicore*, no entanto, o custo financeiro para aquisição de *hardware* para computação de alto desempenho não é muito baixo, implicando na busca de alternativas para sua utilização. Desta maneira, a GPGPU torna-se uma opção de baixo custo sem comprometer o poder de desempenho.

Parte II

Proposta de Tese

Capítulo 5

Proposta de Tese

Neste capítulo apresenta-se a Proposta Tese, abordando o estudo sobre técnicas de paralelização aplicadas ao processamento de imagens médicas. Também são apresentados os objetivos, questões de pesquisa, hipóteses e metodologia, bem como o plano de trabalho para realização do presente projeto.

5.1 Motivação

Um dos maiores problemas em mulheres acima de 50 anos é o prolapso vaginal, o qual é caracterizado pelo deslocamento dos órgãos da cavidade pélvica feminina (intestino, bexiga, útero, ovários, vagina, reto e uretra) para o canal vaginal em razão do enfraquecimento dos músculos e ligamentos desta parte do corpo. Como consequência deste problema, são comuns disfunções na bexiga e intestino como, incontinência e/ou retenção urinária, obstipação, e em casos extremos, chegando à exposição dos órgãos pelo canal vaginal [42, 43]. A perda do tônus muscular e o relaxamento dos músculos são considerados normais, podem ser associados ao envelhecimento, procedimentos cirúrgicos de retirada de útero, múltiplos partos naturais, cirurgias abdominais ou vaginais, e até mesmo em razão da menopausa. Com o objetivo de contribuir na identificação precoce deste tipo de problema, Chowdhury et al. [42], Onal et al. [43], Ma et al. [44, 45], desenvolveram métodos para visualizar, medir e analisar estruturas pélvicas femininas, utilizando técnicas de processamento e análise de imagem.

O uso de análise computadorizada de imagens médicas aplicado com a finalidade de auxiliar no diagnóstico precoce de doenças, estabelece o conceito de sistemas de diagnóstico auxiliado por computador [16, 9]. Esta categoria de sistemas, geralmente, requer alto índice de precisão na identificação da patologia para a qual foi projetado,

bem como deve ser capaz de realizar o processamento computacional, de preferência, em tempo real [46, 18]. Este tipo de sistema requer, no mínimo, a combinação de técnicas de:

1. Processamento e análise de Imagem: aplicadas para realizar redução de ruídos e imperfeições que dificultem a identificação da informação de interesse, realçar dados importantes, permitir a extração de características de interesse na identificação da patologia em estudo, reconhecimento e interpretação de variáveis de interesse [42, 43, 44, 45];

2. Computação de Alto Desempenho: obter métodos computacionais mais robustos, efetivos e de execução rápida e eficiente, explorando técnicas de computação paralela para utilizar o máximo desempenho disponível na arquitetura empregada [40, 38].

5.2 Objetivos

A presente proposta de Doutoramento, tem como objetivo principal, desenvolver, aplicar e testar novas técnicas voltadas ao processamento massivamente paralelo aplicadas ao contexto do diagnóstico baseado em imagens médicas, levando em consideração os elevados requisitos computacionais e restrições temporais normalmente existentes neste domínio. Outros objetivos deste trabalho, são:

- Elaborar uma revisão e classificar os algoritmos empregados em processamento e análise de imagens médicas, bem como identificar as principais características destes algoritmos para uso em aplicações que possam permitir diagnóstico clínico eficiente, em particular em tempo real;

- Levantar e classificar os algoritmos, métodos e técnicas de programação em alto desempenho;

- Propor, implementar e validar novas técnicas robustas e eficientes para o processamento e análise de imagens médicas em arquitetura híbrida de alto desempenho, utilizando estudos de casos sintéticos e reais;

- Desenvolver uma plataforma computacional de alto desempenho para o processamento e análise de imagens médicas da cavidade pélvica feminina que permita ajuda clínica de forma eficiente.

5.3 Questão de Pesquisa

"Quais estratégias e plataformas de computação de alto desempenho podem ser aplicadas no processamento e análise de imagens complexas da cavidade pélvica feminina?" Contudo, outras questões são de grande relevância para a presente pesquisa existem, como:

a) Quais as modalidades de imagem utilizadas?

b) Quais os tipos de patologias os sistemas detectam?

c) Quais arquiteturas computacionais foram exigidas?

d) Quais estratégias de computação de alto desempenho foram adotadas/desenvolvidas?

e) Os sistemas realizam detecção em tempo real?

f) Quais os tipos de licenças dos sistemas?

5.4 Hipótese de Pesquisa

De acordo com o estudo sobre SDDAC acelerados por computação de alto desempenho, apresentado no Capítulo 4, as principais estratégias de paralelização empregadas na medicina, de maneira geral, adotaram *cluster* de computadores [38], FPGA [40] ou *Graphics Processing Unit* (GPU)s [41]. Considerando os avanços recentes da arquitetura *manycore* (*General Purpose Computing on Graphics Processing Unit* (GPGPU)), o baixo custo financeiro e a alta capacidade de processamento disponível, a técnica de programação paralela *Single Instruction, Multiple Thread* (SIMT) para GPUs, juntamente com a arquitetura *multicore*, tornam-se uma solução viável para incorporar a arquitetura de processamento de alto desempenho para *Sistema de Detecção de Doença Auxiliado por Computador*. A arquitetura proposta não deve ser interpretada "apenas" como a paralelização de algoritmos sequenciais de processamento e análise de imagens, mas sim como a otimização dos métodos de pré-processamento (suavização) e segmentação, seguimento, alinhamento e emparelhamento de imagens dos órgãos da cavidade pélvica feminina [45] e elementos arquiteturais (decomposição de dados).

5.5 Metodologia Proposta

Pretende-se, neste trabalho, adotar uma metodologia composta pelas etapas descritas a seguir.

5.5.1 Tarefa de Pesquisa 1

Primeiramente, concluir a revisão sistemática - em andamento - acerca de estudos e análises sobre sistema de detecção de doença auxiliado por computador, com foco em tomografia computadorizada, ressonância magnética e ultrassonografia, especialmente aplicados à cavidade pélvica feminina. O objetivo desta tarefa é mapear e identificar as técnicas de processamento e análise de imagem desenvolvidas para sistemas de diagnósticos de patologias de tempo real. Até a conclusão do presente projeto, não foi identificado na literatura científica nenhum sistema de diagóstico de patologias da cavidade pélvica feminina em tempo real.

1. **Atividades**

   (a) identificar as principais modalidades de imageamento adotadas em SDDAC;

   (b) realizar uma revisão da literatura acerca de pesquisas envolvendo SDDAC;

   (c) identificar e selecionar as técnicas de computação de alto desempenho adotadas em SDDAC;

   (d) identificar e selecionar parâmetros para comparação de trabalhos relacionados;

   (e) identificar e selecionar as técnicas de processamento e análise de imagem da cavidade pélvica feminina.

2. **Resultados Esperados**

   (a) relatório técnico preliminar descrevendo as principais técnicas, arquiteturas e plataformas de alto desempenho adotadas em SDDAC;

   (b) artigo com o "estado da arte" sobre sistemas de diagóstico médico baseado em imagem médica com processamento em tempo real.

5.5.2 Tarefa de Pesquisa 2

A segunda tarefa de pesquisa do presente projeto, é explorar e validar estratégias de computação de alto desempenho. Tem como objetivo, oferecer suporte computacional efici-

ente para atender SDDAC em tempo real. Nesta tarefa, devem ser consideradas arquiteturas paralelas, modelos e técnicas de programação paralela, direcionando a pesquisa para arquiteturas massivamente paralelas em GPGPU, multicores, e modelos de programação paralela SIMT (*Single Instruction, Multiple Thread*). A métrica utilizada para avaliar o desempenho da solução paralelizada será o *speedup*.

1. **Atividades**

   (a) revisar a documentação sobre arquitetura CUDA em GPU;

   (b) explorar computação em GPU como suporte a processamento e análise de imagem;

   (c) identificar a arquitetura e plataforma disponível para realização dos experimentos.

2. **Resultados Esperados**

   (a) relatório técnico final com as principais técnicas, arquiteturas e plataformas de alto desempenho adotadas em SDDAC aplicadas à cavidade pélvica feminina;

   (b) artigo com o modelo proposto da arquitetura de alto desempenho para o processamento em tempo real do SDDAC aplicado à cavidade pélvica feminina;

   (c) finalizar projeto de tese e solicitar inscrição definitiva.

5.6 Plano de Trabalho

Em seguida são descritas as tarefas principais envolvidas nas etapas de desenvolvimento desta pesquisa, bem como o cronograma previsto para realização de cada tarefa. A duração prevista para realização deste doutoramento é de 48 meses.

1. Unidades Curriculares (30 créditos): Cumprir os créditos nas unidades curriculares do PRODEI: Planeamento de Investigação, Computação Paralela, Paradigmas de Programação.

2. Unidades Curriculares (24 créditos): Cumprir os créditos nas unidades curriculares do PRODEI: Metodologias de Investigação Científica, Interação e Ambientes de Simulação Visual, Extração de Conhecimento e Aprendizagem Computacional.

3. Cenário para Testes: Posteriormente, preparar a infraestrutura inicial para a fase de desenvolvimento e implementação da plataforma computacional para processamento de alto desempenho, a qual estará interligada com uma fase de testes e

validação de novas técnicas, metodologias, bem como modelos computacionais, buscando identificar a necessidade de eventuais correções no planeamento inicial.

4. Implementação e Validação: Estruturar e implementar as soluções consideradas mais adequadas ao problema em questão. A seguir o projeto entra na fase de testes finais para validação das soluções desenvolvidas. Nessa fase, serão realizados testes em laboratório e em ambiente de produção com o intuito de identificar a necessidade de possíveis ajustes de implementação. A plataforma computacional desenvolvida na presente proposta deverá possuir arquitetura aberta, modular e interface gráfica com ajustes de configuração, bem como ser composta por bibliotecas computacionais de domínio público quando possível. A intenção é adotar: *Open Source Computer Vision* (OpenCV) (processamento e análise de imagem: http://sourceforge.net/projects/opencv), *Visualization ToolKit* (VTK) (visualização e processamento de objetos gráficos: http://www.vtk.org), OpenMP, MPI, além de OpenCL/CUDA.

5. Produção Científica: publicação dos resultados em periódicos de interesse e participação em conferências da área, identificados nas Tabelas A e B, respectivamente.

6. Escrita da Tese.

Tabela 5.1: *Cronograma de Atividades.*

Cronograma de Atividades								
	2014		2015		2016		2017	
Atividades	01	02	03	04	05	06	07	08
Unidades Curriculares (30 créditos)								
Unidades Curriculares (24 créditos)								
Revisão da literatura envolvendo SDDAC								
Levantamento de técnicas de computação de alto desempenho adotadas em SDDAC								
Levantamento de técnicas de processamento e análise de imagem da cavidade pélvica feminina								
Relatório técnico preliminar acerca de técnicas, arquiteturas e plataformas de alto desempenho adotadas em SDDAC								
Produção Científica: o "estado da arte" sobre SDDAC com processamento em tempo real								
Revisão da documentação acerca da arquitetura CUDA em GPU								
Levantamento/utilização de GPU como suporte a processamento e análise de imagem								
Relatório técnico final acerca de técnicas, arquiteturas e plataformas de alto desempenho adotadas em SDDAC								
Produção Científica: modelo da arquitetura de alto desempenho para o processamento em tempo real do SDDAC aplicado à cavidade pélvica feminina								
Projeto de Tese final e Inscrição definitiva								
Cenário para Testes								
Implementação e Validação								
Produção Científica								
Escrita da Tese								

Referências

[1] W. Stallings, *Computer Organization and Architecture: Designing for Performance*, 6th ed. Pearson Education International, 2003, iSBN 0-13-049307-4.

[2] M. Schmeisser, B. C. Heisen, M. Luettich, B. Busche, F. Hauer, T. Koske, K.-H. Knauber, and H. Stark, "Parallel, distributed and GPU computing technologies in single-particle electron microscopy," *Acta Crystallographica Section D*, vol. 65, no. 7, pp. 659–671, Jul 2009. [Online]. Available: http://dx.doi.org/10.1107/S0907444909011433

[3] D. Kirk and W.-M. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. Elsevier, 2010.

[4] NVIDIA, *GPU Tutorial: Build environment, Debugging/Profiling, Fermi, Optimization/CUDA 3.1 and Fermi advice*, NVIDIA, 2010.

[5] A. Vadja, *Programming Many-Core Chips*, A. Vadja, Ed. Springer, 2011, iSBN: 978-1-4419-9738-8.

[6] W. mei Hwu, *GPU Computing GEMS*, emerald edition ed., W. mei Hwu, Ed. Morgan Kaufmann and NVIDIA, 2011.

[7] B. Chapman, F. Desprez, G. R. Joubert, A. Lichnewsky, F. Peters, and T. Priol, *Parallel Computing: From Multicores and GPUs to Petascale*, G. R. Joubert, Ed. IOS Press BV, 2010. [Online]. Available: http://www.booksonline.iospress.nl/

[8] J. Kurzak, D. A. Bader, and J. Dongarra, *Scientific Computing with Multicore and Accelerators*, ser. Computational Science, H. Simon, Ed. CRC Press, 2010, no. ISBN 978-1-4398-2536-5.

[9] S. Olabarriaga, J. Snel, C. Botha, and R. Belleman, "Integrated support for medical image analysis methods: From development to clinical application," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 11, no. 1, pp. 47–57, Jan 2007.

[10] L. Shi, W. Liu, H. Zhang, Y. Xie, and D. Wang, "A survey of GPU-based medical image computing techniques." *Quant Imaging Med*

Surg, vol. 2, no. 3, pp. 188–206, Sep 2012. [Online]. Available: http://dx.doi.org/10.3978/j.issn.2223-4292.2012.08.02

[11] P. Lenkiewicz, M. Pereira, M. Freire, and J. Fernandes, "Accelerating 3D medical image segmentation with high performance computing," in *IEEE International Workshops on Image Processing Theory, Tools and Applications*. IEEE, 2008, pp. 1–8.

[12] Y. Ouma, R. Tateishi, and J. Sri-Sumantyo, "Urban features recognition and extraction from very-high resolution multi-spectral satellite imagery: a micro–macro texture determination and integration framework," *IET Image Processing*, vol. 4, no. 4, pp. 235–254, 2010.

[13] X. Jin and C. H. Davis, "Vehicle detection from high-resolution satellite imagery using morphological shared-weight neural networks," *Image and Vision Computing*, vol. 25, no. 9, pp. 1422 – 1431, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885606003544

[14] G. Schonung, "Visual geometric environment modeling for augmented reality and robotics," *Advances in Media Technology*, 2011.

[15] T. Tawara and K. Ono, "A framework for volume segmentation and visualization using augmented reality," in *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, march 2010, pp. 121 –122.

[16] P. Jannin, E. Krupinski, and S. K. Warfield, "Validation in medical image processing," *Medical Imaging, IEEE Transactions on*, vol. 25, no. 11, pp. 1405–1409, Nov 2006.

[17] T. Kulhánek and M. Šárek, "Processing of medical images in virtual distributed environment," in *Proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to Increase Digital Citizenship*, ser. EATIS '09. New York, NY, USA: ACM, 2009, pp. 10:1–10:3. [Online]. Available: http://doi.acm.org/10.1145/1551722.1551732

[18] N. Chrisochoides, A. Fedorov, A. Kot, N. Archip, P. Black, O. Clatz, A. Golby, R. Kikinis, and S. K. Warfield, "Toward real-time image guided neurosurgery using distributed and grid computing," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, ser. SC '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1188455.1188536

[19] G. Teodoro, T. D. R. Hartley, U. Catalyurek, and R. Ferreira, "Run-time optimizations for replicated dataflows on heterogeneous environments," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 13–24. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851479

[20] M. Nolle, G. Schreiber, H. Schulz-mirbach, and T. I. I, "PIPS - a general purpose parallel image processing system," in *DAGM - Symposium Mustererkennung*, 1994, pp. 609–623.

[21] D. Page, *A Practical Introduction to Computer Architecture*, D. Gries and F. B. Schneider, Eds. Springer, 2009.

[22] F. Gebali, *Algorithms and Parallel Computing*. John Wiley & Sons, 2011.

[23] R. Duncan, "A survey of parallel computer architectures," *Computer*, vol. 23, no. 2, pp. 5 –16, feb 1990.

[24] J. Gurd, "A taxonomy of parallel computer architectures," in *Design and Application of Parallel Digital Processors, 1988., International Specialist Seminar on the*, apr 1988, pp. 57 –61.

[25] K. Hwang, "Advanced parallel processing with supercomputer architectures," *Proceedings of the IEEE*, vol. 75, no. 10, pp. 1348 – 1379, oct. 1987.

[26] H. El-Rewini and M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*, ser. Wiley Series on Parallel and Distributed Computing, A. Y. Zomaya, Ed. Wiley-Interscience, 2005, vol. 19, no. ISBN 0-471-46740-5.

[27] B. Parhami, *Introduction to Parallel Processing: Algorithms and Architectures*, ser. Plenun Series in Computer Science, R. G. Melhem, Ed. Kluwer Academic Publishers, 2002, no. ISBN: 0-306-45970-1.

[28] H. Jang, A. Park, and K. Jung, "Neural network implementation using CUDA and OpenMP," in *Proceedings of the 2008 Digital Image Computing: Techniques and Applications*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 155–161. [Online]. Available: http://portal.acm.org/citation.cfm?id=1469127.1470322

[29] A. Merigot and A. Petrosino, "Parallel processing for image and video processing: Issues and challenges," in *Parallel Computing*. Else, September 2008, pp. 694–699.

[30] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33 –38, july 2008.

[31] J. L. Hennessy and D. A. Patterson, *Computer Architecture A Quantitative Approach*, 4th ed. Elsevier, 2007, no. ISBN 978-0-12-370490-0.

[32] A. S. Tanenbaum, *Sistemas Operacionais Modernos*. Pearson Education do Brasil, 2010, iSBN 9788576052371.

[33] J. Sanders and E. Kandrot, *CUDA by example: An introduction to General-Purpose GPU programming*. Addison-Wesley, 2011.

[34] R. Borgo and K. Brodlie, "State of the art report on GPUs," The University of Leeds, School of Computing, Tech. Rep., February 2009, visualization and Virtual Reality Research Group.

[35] M. Hofmann and T. Binna, "Massive parallel image processing," University of Applied Science Rapperswil, Tech. Rep., 2010, monografia de Graduação.

[36] C. Okoli and K. Schabram, "A guide to conducting a systematic literature review of information systems research," *Sprouts: Working Papers on Information Systems*, vol. 10, no. 26, 2010. [Online]. Available: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1954824

[37] H. M. Cooper, *The structure of knowledge synthesis*, ser. Knowledge in Society, 1988, vol. 1.

[38] R. Kikinis, S. Warfield, and C.-F. Westin, "High performance computing (HPC) in medical image analysis (MIA) at the surgical planning laboratory (SPL)." High Performance Computing Asia Conference and Exhibition, 09 1998.

[39] H. Zhang, M. J. Boyles, G. Ruan, H. Li, H. Shen, and M. Ando, "XSEDE-enabled high-throughput lesion activity assessment," in *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, ser. XSEDE '13. New York, NY, USA: ACM, 2013, pp. 10:1–10:8. [Online]. Available: http://doi.acm.org/10.1145/2484762.2484783

[40] G. Dasika, K. Fan, and S. Mahlke, "Power-efficient medical image processing using puma," in *IEEE 7th Symposium on Application Specific Processors*. San Francisco, CA: IEEE, July 2009, pp. 29–34. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5226332&isnumber=5226326

[41] S. S. Stone, J. P. Haldar, S. C. Tsao, W.-m. W. Hwu, Z.-P. Liang, and B. P. Sutton, "Accelerating advanced MRI reconstructions on GPUs," in *Proceedings of the 5th Conference on Computing Frontiers*, ser. CF '08. New York, NY, USA: ACM, 2008, pp. 261–272. [Online]. Available: http://doi.acm.org/10.1145/1366230.1366276

[42] A. Chowdhury, J. Burns, A. Mukherjee, B. Sen, J. Yao, and R. Summers, "Automated detection of pelvic fractures from volumetric CT images," in *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*, May 2012, pp. 1687–1690.

[43] S. Onal, S. Lai-Yuen, P. Bao, A. Weitzenfeld, and S. Hart, "Image based measurements for evaluation of pelvic organ prolapse," *Journal of Biomedical Science and Engineering*, vol. 6, pp. 45–55, 2013.

[44] Z. Ma, J. M. R. Tavares, R. N. Jorge, and T. Mascarenhas, "A review of algorithms for medical image segmentation and their applications to the female pelvic cavity," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 13, no. 2, pp. 235–246, 2010, pMID: 19657801. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10255840903131878

[45] Z. Ma, R. M. N. Jorge, T. Mascarenhas, and J. M. R. Tavares, "Segmentation of female pelvic organs in axial magnetic resonance images using coupled geometric deformable models," *Computers in Biology and Medicine*, vol. 43, no. 4, pp. 248 – 258, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010482513000152

[46] A. Eklund, M. Andersson, and H. Knutsson, "fMRI analysis on the GPU possibilities and challenges," *Computer Methods and Programs in Biomedicine*, vol. 105, no. 2, pp. 145 – 161, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169260711001957

Apêndice A

Periódicos de Interesse

Os principais periódicos de interesse, relacionados com os temas de Processamento e Análise de Imagens Médicas, Visão Computacional, Computação de Alto Desempenho, estão listados na Tabela A.1. Os índices indicados na coluna **Fator de Impacto** foram extraídos no *website* de cada *journal*, em 30/06/2014.

Tabela A.1: *Periódicos relacionados aos temas de pesquisa.*

ISSN	Título	Fator de Impacto
1939-8018	Journal of Signal Processing Systems for Signal, Image, and Video Technology	0.551
1524-0703	Graphical Models and Image Processing	0.697
0097-8493	Computers & Graphics	0.794
1432-2315	The Visual Computer	0.909
0897-1889	Journal of Digital Imaging	1.100
1432-1769	Machine Vision and Applications	1.103
1861-8219	Journal of Real-Time Image Processing	1.156
1047-3203	Journal of Visual Communication and Image Representation	1.195
1077-3142	Computer Vision and Image Understanding	1.232
0010-4825	Computers in biology and medicine	1.359
0169-2607	Computer Methods and Programs in Biomedicine	1.555
1467-8659	Computer Graphics Forum	1.638
0895-6111	Computerized medical imaging and graphics	1.664

Continua na página seguinte

Tabela A.1 – continuando da página anterior

Título	Qualis	Deadline
1573-689X	Journal of Medical Systems	1.783
0262-8856	Image and Vision Computing	1.959
1089-7771	IEEE Transactions on Information Technology in Biomedicine	1.978
1386-5056	International Journal of Medical Informatics	2.061
1532-0464	Journal of Biomedical Informatics	2.434
0739-5175	IEEE Engineering in Medicine and Biology Magazine	2.727
1057-7149	IEEE Transactions on Image Processing	3.199
0920-5691	International Journal of Computer Vision	3.623
0278-0062	IEEE Transactions on Medical Imaging	4.027
1361-8415	Medical Image Analysis	4.087

Apêndice B

Conferências de Interesse

As principais conferências de interesse, relacionadas com o presente tema do projeto estão indicadas na Tabela B.1, que apesar de listar algumas datas limites ultrapassadas será atualizada frequentemente. Os índices indicados na coluna **Qualis** estão atualizados e foram obtidos no repositório http://qualis.capes.gov.br/webqualis.

Tabela B.1: *Conferências de Interesse.*

Título	**Qualis**	**Deadline**
HPCA - High-Performance Computer Architecture	A1	05/09/2014
HPDC - IEEE International Symposium on High Performance Distributed Computing	A1	20/01/2015
IPDPS - IEEE International Parallel and Distributed Processing Symposium	A1	10/10/2014
ICIP - IEEE International Conference on Image Processing	A1	15/01/2015
MICCAI - Medical Image Computing and Computer Assisted Intervention	A1	28/02/2014
ICPP - International Conference on Parallel Processing	A2	05/03/2014
ICIAP - International Conference on Image Analysis and Processing	B1	11/11/2014
SIBGRAPI - Brazilian Symposium on Computer Graphics and Image Processing	B1	25/04/2014
SBAC-PAD - International Symposium on Computer Architecture and High Performance	B2	17/06/2014
		Continua na página seguinte

Tabela B.1 – continuando da página anterior

Título	Qualis	Deadline
PDCAT - International Conference on Parallel and Distributed Computing, Applications and Technologies	B2	01/08/2014
CAIP - International Conference on Computer Analysis of Images and Patterns	B2	indefinido
PDPTA - International Conference on Parallel and Distributed Processing Techniques and Applications	B2	04/06/2014
IST - IEEE International Conference on Imaging Systems & Techniques	B4	10/07/2014

Apêndice C

Grupos de Pesquisa

Em seguida, foram identificados alguns grupos de pesquisa com interesse na área de processamento e análise de imagens médicas acelerados por estratégias de computação de alto desempenho.

- Surgical Planning Laboratory[1] - a teaching affiliate of Harvard Medical School, USA.
- Medical Image High Performance Computing (HPC) & Informatics Laboratory[2] - University of Iowa, Iowa, USA.
- Image and Video Processing and Communications Laboratory[3] - University of New Mexico, Albuquerque, USA.
- Heterogeneous and Parallel Computing Lab[4] - Norwegian University of Science and Technology, Trondheim, Norway.
- Scalable Computing Laboratory[5] - Pennsylvania State University, Pennsylvania, USA.
- GAPIS - Grupo de Arquitetura e Processamento de Imagens e Sinais, Universidade Federal de São Carlos, São Carlos, Brasil.

[1] https://www.spl.harvard.edu/index.html
[2] http://www.uiowa.edu/mihpclab/
[3] http://www.ivpcl.org/
[4] http://research.idi.ntnu.no/hpc-lab/
[5] http://www.cse.psu.edu/research/scl