LEAPaD: Laboratório de Estudos Avançados em Sistemas Paralelos e Distribuídos

Projeto Procad: USP-UnB-UFPel

Dezembro de 2013

Sumário

1 Identificação do Projeto

Apresentamos aqui o projeto LEAPaD para o edital Procad de 2013, para facilitar a leitura seguimos a mesma estrutura de tópicos sugerida pela CAPES.

2 Dados da IES e do PPG - Proponente, Associadas I e II

2.1 Proponente

Programa de Pós-Graduação em Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

2.2 Associada I

Programa de Pos-Graduação em Informática
Universidade de Brasilia (UnB)

2.3 Associada II

Programa de Pós-Graduação em Computação
Centro de Desenvolvimento Tecnológico
Universidade Federal de Pelotas

3 Identificação do Coordenador - Proponente, Associadas I e II

3.1 Proponente

Alfredo Goldman - IME - USP

3.2 Associada I

Alba Cristina Magalhães Alves de Melo - CIC - UnB

3.3 Associada II

Gerson Geraldo Homrich Cavalheiro - CDTec - UFPel

4 Resumo

O processamento paralelo e distribuído é uma realidade nos sistemas computacionais há muitos anos. No entanto, observa-se no histórico mais recente, o crescimento dos horizontes de sua aplicação em função de uma nova realidade de mercado, a qual oferece preços muito competitivos para os mais diversos tipos de plataformas de hardware, implicando no aumento das ofertas de soluções e, consequentemente, de demandas de aplicações da sociedade em geral. Logo, se no passado não muito distante, problemas e questões relacionadas ao processamento paralelo e distribuído surgiam em função do desenvolvimento de aplicações caracterizadas por necessitarem de uma grande quantidade de recursos computacionais, hoje surge uma nova classe de problemas em função da pluralidade de dimensões que as novas plataformas de processamento oferecem. Esta pluralidade reflete a incorporação nas plataformas de execução de diferentes tecnologias de hardware com suporte ao processamento intensivo e paralelo, incluindo não apenas os multiprocessadores e aglomerados de computadores, mas também FPGAs, GPUs, grades e nuvens computacionais.

Dentre os efeitos observados, buscam-se novos modelos computacionais para solucionar as diferentes questões operacionais que se apresentam nesta nova realidade. Neste contexto, o presente projeto se insere apresentando a criação do LEAPaD, um laboratório, virtualmente distribuído entre as instituições parceiras, vocacionado em explorar questões ligadas ao gerenciamento das ações que envolvem o processamento paralelo e distribuído. O LEAPaD, acrônimo para Laboratório de Estudos Avançados em Sistemas Paralelos e Distribuídos, se institui de forma a consolidar linhas de atuação ligadas aos programas associados a este projeto, buscando explorar soluções para construções de aplicações e sistemas de gerenciamento e exploração de processamento de alto desempenho em arquiteturas paralelas e distribuídas, tais como FPGAs, ambientes multiprocessados e com GPUs, aglomerados de computadores e de grades e nuvens computacionais.

5 Detalhamento do Projeto

O avanço tecnológico dos últimos anos proporcionou a diferentes classes de consumidores, representados desde pessoas físicas até grandes instituições, o acesso a uma vasta gama de opções de plataforma de hardware para servir de suporte à execução de suas aplicações. Como consequência, tornam-se mais evidentes as necessidades de aprimoramento nas soluções que melhor

explorem os recursos de processamento disponíveis, em particular quando as questões encontram-se relacionadas à exploração de recursos que ofereçam alguma forma de processamento paralelo ou distribuído, ou mesmo solução em hardware para algoritmos especializados.

Considerando que não existem estratégias gerais que ofereçam uma abstração de como explorar esta grande variedade de recursos, seja em nível de particionamento da concorrência das aplicações [Asanovic, 2006] ou de modelos ou linguagens de desenvolvimento [Skilicorn, 1998], o enfoque deste projeto está em promover a pesquisa sobre ferramentas para gestão e exploração eficiente de diferentes plataformas de hardware paralelo, como FPGA, GPU, multiprocessadores e grades computacionais. Isso, em direção a uma realidade onde as plataformas de processamento terão configurações heterogêneas e, não raro, moldadas de acordo com a necessidade das aplicações [ABC$^+$06].

Temos como objetivo o desenvolvimento de novos modelos que ofereçam ao menos níveis moderados de abstração de uso e facilidade de portabilidade entre diferentes plataformas. Considera-se assim que respostas adequadas às questões que envolvem o uso racional e efetivo das plataformas de hardware consistem em aspectos chaves para oferta de aplicações que realmente atendam aos anseios dos usuários, explorando de forma eficiente a plataforma de execução disponível.

No contexto apresentado, as instituições envolvidas possuem em curso pesquisas relacionadas às questões apresentadas, vocacionando o LEAPaD nesta área, tendo como principal enfoque a gestão de recursos para o processamento paralelo e distribuído. As linhas de atuação encontram-se delimitadas nos seguintes eixos de atuação.

- Aplicações sobre hardware configurável a Aceleradores Manycore;
- Computação em Grade e em Nuvem;
- PAD em Arquiteturas Paralelas.

Apesar das linhas de atuação acima estarem listadas de forma independentes, espera-se que no decorrer do projeto uma estreita cooperação entre os diversos eixos que compõem o presente projeto. Segue abaixo uma descrição mais completa dos temas específicos, ligados a cada eixo:

5.1 Aplicações sobre Hardware Configurável e Aceleradores Manycore

FPGAs são circuitos integrados projetados para serem configurados de acordo com as exigências da aplicação. Eles oferecem um alto grau de paralelismo especialmente para computações de precisão simples. Em muitas aplicações dedicadas a PADs, os tipos de dados são adequados a FPGAs que abrem a possibilidade de implementação em plataformas de prototipação. Neste projeto são abordadas 3 áreas de aplicação que permitem o emprego de hardware configurável para aceleração do processamento, assim sendo (i) Bioinformática, (ii) Computação Quântica e (iii) Criptoanálise. Em Bioinformática,comparações de sequências biológicas podem ser aceleradas com o uso de FPGA. Já em computação quântica, os FPGAs são usados para acelerar as simulações de algoritmos de computação quântica. Enquanto que em criptoanálises, os FPGAs podem ser usados para reduzir o tempo de computação para revelar informações secretas protegidas por algoritmos criptográficos. Os problemas inerentes a cada aplicação bem como as propostas de soluções a serem abordadas neste projeto são detalhadas nesta Seção.

5.1.1 Aplicação de Bioinformática

Problema de pesquisa

Nas últimas décadas, houve um crescimento sem precedentes na área de Genômica Comparativa e um volume enorme de dados foi produzido. Esses dados precisam agora ser analisados, para a determinação de propriedades que podem ajudar a melhorar os processos envolvidos na confecção de medicamentos, contribuir para o melhor entendimento dos mecanismos existentes em diversas doenças, desenvolver teorias que expliquem melhor aspectos evolucionários e correlações entre as espécies, dentre outros. Para abordar problemas tão importantes, algoritmos sofisticados foram propostos e diversas ferramentas de Bioinformática foram desenvolvidas para a utilização desses algoritmos [Mou04]. No entanto, devido a quantidade enorme de dados biológicos a serem analisados, a maioria dessas ferramentas demora horas ou mesmo dias para executar. Esse é um problema que os Projetos Genoma estão enfrentando atualmente, de tal maneira que acredita-se que os projetos bem sucedidos nessa area serão aqueles que conseguirem produzir resultados mais rápido.

Atualmente, arquiteturas de processamento de alto desempenho compostas de centenas de milhares de multicores são uma realidade. Além disso,

arquiteturas many-core tais como GPUs (Graphic Processing Units), conseguem um desempenho de mais de 3 TFLOPS (Trilhão de Operações de Ponto Flutuante por Segundo) a um custo muito baixo. Outra alternativa que possui altíssimo desempenho são hardwares programáveis como os FPGAs (Field-Programmable Gate Arrays), que foram integrados a Desktops, com manipulação relativamente simples.

Estado da arte

O método exato mais usado atualmente para obter o alinhamento de pares de sequências biológicas é o proposto por Smith e Waterman (SW) [SW81]. O processamento paralelo e distribuído foi usado para reduzir o tempo de execução do algoritmo SW, em plataformas compostas por FPGAs [BCdMJ10], GPUs (Graphics Processing Units) [SdM11] [dOSdM13] [LSM10] e Cell/BEs [SA09]. Além disso, existem versões paralelas e distribuídas do SW para clusters [Batista08] [RA04], grids [CS05] e clouds [LM12]. Em [SdM11], a mesma comparação foi executada cerca de 700x mais rápido em uma GPU, quando comparado com um core. Com essa versão paralela, o alinhamento ótimo entre os cromossomos 21 humano (46 Milhões de Pares de Bases - MBP) e do chimpanzé (34 MBP) foi obtido em 18.5 horas. Essa foi a primeira vez que o resultado ótimo foi produzido para sequencias maiores do que 10 MBP. No entanto, o tempo de execução ainda continua alto e precisa ser reduzido.

BLAST [ea95] é um método heurístico para comparação de pares de sequências que é intensamente utilizado por projetos de genômica comparativa. Apesar de uma comparação BLAST ser executada de maneira muito rápida, normalmente uma sequência é comparada com centenas de milhares de outras sequências, pertencentes a um banco de dados genômico, Nesse cenário, execuções BLAST demoram também muito tempo e esse tempo precisa ser reduzido. Existem esforços de implementação do BLAST em GPU [VS11] e em Grids [SMB10].

Outras aplicações importantes de Bioinformática são classificação de RNAs não-codificadores, alinhamento múltiplo de sequencias [SdM13] e anotação genômica. Para essas aplicações, existem muito poucas versões paralelas e distribuídas.

Objetivos

Na sub-área de Aplicação em Bioinformática, temos os seguintes objetivos:

- Investigar as características de aplicações de Bioinformática e determinar quais delas devem ser consideradas na escolha da plataforma de alto desempenho (hardware reconfigurável ou aceleradores manycore) e do ambiente de programação (OpenMP, VHDL, OpenCL, CUDA, entre outros) mais adequados.

- Investigar e propor novas ferramentas de bioinformática que usem somente hardware reconfigurevel, somente aceleradores manycore ou ambos.

- Dadas as características da aplicação de Bioinformática e a plataforma de alto desempenho sobre a qual a mesma será executada, propor um modelo de previsão de desempenho que seja capaz de prever o tempo de execução e o throughput da aplicação.

5.1.2 Aplicação na Computação Quântica

Problema de pesquisa

A Computação Quântica (*CQ*) segue atingindo novos marcos rumo a construção de computadores quânticos [NC00]. Apesar de todos os esforços, vários desafios técnicos limitam os sistemas atuais à apenas alguns bits quânticos [SH04]. Devido à indisponibilidade de *hardware* quântico, o estudo e desenvolvimento de aplicações na *CQ* usualmente é feito estritamente pela especificação matemática das computações ou por meio de ferramentas de simulação. Este último caracteriza a abordagem mais prática, entretando a complexidade computacional associada a simulação de sistemas quânticos a partir de computadores clássicos limita o tamanho dos sistemas que podem ser simulados.

Estado da Arte

Várias abordagens para otimização da simulação quântica já foram estudadas. Em [RMR+06] e [NMI08], *clusters* para acelerar as computações são utilizados e em [GRTZ10], busca-se a aceleração das computações através de *GPUs* (*Graphic Processing Units*).

Abordagens voltadas à simulação sequencial, representadas por [BW03] e [Via07], visam a redução da complexidade das computações através de otimizações algorítmicas. Todas essas pesquisas possuem grandes méritos por proporem grandes avanços em suas respectivas áreas.

A exploração das abstraçõess do Modelo qGM [RA10] para fundamentação, modelagem e simulação paralela/distribuíds da CQ busca a consolidação do

ambiente computacional D-GM para desenvolvimento de aplicações fuzzy via registradores quânticos. Considera-se a gerencia de execução pela VirD-GM [AMR+14] e a modelagem via interface gráfica do VPE-qGM [MRP13].

O framework de simulação D-GM visa a simulação quântica em arquiteturas híbridas (clusters, CPUs multicore e GPUs), habilitando o controle e execução da simulação a partir de arquiteturas multicore. Essa proposta se mostra relevante por prover o gerenciamento da simulação em CPUs multicore, capacidade esta que será posteriormente combinada com implementações voltadas à GPU. Da sinergia entre essas implementações, busca-se estabelecer uma plataforma de simulação híbrida, explorando diferentes arquiteturas usualmente encontradas em clusters e expandindo as capacidades de simulação do ambiente VPE-qGM [SMRP12, AMRP12, MRP13, AMR+14].

Na incerteza modelada por sistemas fuzzy, consideram-se relevantes as propriedades dos conectivos da lógica fuzzy (LF), como negações [Bed07], aggregadores [RBB13] e implicações [BDSR10]. Este conectivos podem ser modelados por registrados, permitindo que além de diferentes emoções, também a intensidade associada a estas sejam modeladas em relacionamentos entre dois ou mais agentes [RAG10].

Em uma aboragem científica investigamos a integração da CQ com a LF para modelagem e simulação via software do relacionamento emocional entre agentes, como apresentado em [RAG10]. Em uma abordagem tecnológica investigaremos as possibilidades da utilização de simulação via hardware, com foco na construção de bibliotecas de otimização de códigos com descrições em VHDL e validações em FPGA com extensão e possibilidade de prototipação em circuitos integrados.

O projeto considera colaborar em duas abordagens da simulação quântica, descritas logo a seguir.

Objetivos

Os objetivos a serem atingidos neste projeto no contexto da Computação Quântica explora a potencialidade do uso, em software, de recursos de processamento paralelo e/ou distribuído via software de simulação, como a possibilidade de descrever algorítmos em hardware dedicado. Desta forma, são dois os objetivos principais a serem atingidos:

- Modelagem da incerteza no relacionamento emocional entre agentes, integrando a Lógica Fuzzy e a Computação Quântica.

- Colaborar com métodos específicos para descrição de circuitos quânticos através da linguagem VHDL, incluindo a correspondente execução em dispositivos lógicos reconfiguráveis como FPGAs, para simulação paralela (via *hardware*) de algoritmos básicos da computação quântica.

Os objetivos secundários a serem alcançados são:

- *Potencializar a dinâmica de execução do ambiente D-GM para suporte à execução concorrente de sistemas fuzzy simulados por registradores quânticos, baseadas nas abstrações do modelo qGM e na simulação de propriedades inerentes, como o paralelismo e o emaranhamento de estados quânticos.*

- Promover a expansão de novos construtores e extensão dos ambientes de desenvolvimento VPE-qGM e VirD-GM em diferentes arquiteturas: aglomerados de computadores, arquiteturas dedicadas e arquiteturas multiprocessadores, incluindo a potencialidade das placas de vídeo (GPUs);

- Disponibilizar, na forma de *software* livre, a prototipação das ferramentes VPE-qGM e VirD-GM, integrando funcionalidades para geração de processos, configuração de memória e gerenciamento da execução quando de aplicações científicas e/ou tecnológicas de nanoestruturas;

- Investigar a simulação de conectivos da lógica fuzzy via registradores quânticos, modelados no VPE-qGM sob gerenciamento da VirD-GM, fundamentar o desenvolvimento de sistema fuzzy para simulação de relacionamento emocial entre agentes;

- Buscar a otimização de circuitos em termos de área e frequência à nível RT (Transferência de Registradores) e viabilizar a exploração arquitetural pela aplicação de técnicas para o aumento de desempenho;

- Extender a metodologia `qExVHDL` para especificação de algoritmos que envolvam paralelismo e emaranhamento de estados quânticos;

5.1.3 Aplicação em Criptoanálise

Problema de pesquisa

As últimas décadas presenciam uma necessidade crescente por sistemas computacionais que garantam o sigilo de informações, seja durante o processamento ou armazenamento [Sta07]. Com o advento das transações de compra

e venda via internet, estas ações dependem essencialmente de algoritmos de criptografia e protocolos específicos para a garantia de sigilo de informações e legitimidade das transações. Apesar de os algoritmos criptográficos modernos serem a priori seguros, eles não podem consumir um tempo excessivo de processamento a fim de evitar transtornos nas transações.

Em paralelo com a criptografia, a criptoanálise, ciência dedicada a violar textos cifrados, também vem se desenvolvendo, principalmente para avaliar a segurança de algoritmos criptográficos. As criptoanálises são também conhecidas como ataques criptográficos. Apesar de os algoritmos de criptografia serem públicos, a chave criptográfica deve ser mantida secreta a fim de garantir a segurança das informações cifradas. A chave criptográfica é um conjunto de símbolos conhecido apenas pelos entes envolvidos na comunicação e descobri-la é o objetivo principal da criptoanálise. Uma chave criptográfica é representada tipicamente por uma sequência de 64 a 1024 bits o que garante um número grande de combinações possíveis e dificulta a ação dos ataques criptográficos.

Em 1996, porém, Paul Kocher [Koc96] apresentou à comunidade acadêmica um modo alternativo de realizar ataques, estabelecendo uma correlação entre os dados envolvidos na operação de encriptação/decriptação e as propriedades físicas inerentes ao funcionamento do circuito de criptografia, como por exemplo tempo de execução, consumo de potência ou emissão eletromagnética. Estes ataques são conhecidos como ataques a canais laterais (do inglês Side Channel Attacks - SCA). Tais ataques mostram-se bem sucedidos, podendo revelar a chave criptográfica em um tempo de processamento relativamente curto quando comparado as abordagens clássicas de criptoanálise. Em [MOP09], experimentos feitos com um processador Z80 executando o algoritmo DES (Data Encryption Standard), mostram que é possível revelar a chave com a monitoração do sistema durante a encriptação de apenas 63 mensagens diferentes de entrada e seus correspondentes traços de consumo de potência obtidos durante o processamento.

O processo de análise dos traços obtidos, porém, exige uma capacidade de processamento relativamente alta quando realizados em um circuito criptográfico equipado com algum método de proteção a SCA onde é necessário analisar centenas de milhares de traços de consumo de potência, com tamanhos que dependem da taxa de amostragem e tempo de execução do algoritmo. Ataques desta natureza podem levar 30 dias de processamento ou mais em um desktop de propósito geral.

Estado da Arte

Na literatura existem algumas propostas para acelerar a computação de programas científicos, em geral utilizando circuitos dedicados para a computação. Desta forma alguns trabalhos relacionados são revisados. Em [CLS+08] é comparada a utilização de FPGAs e GPUs como aceleradores de processamento, e em [Tah09] é analisada apenas a utilização de GPUs como elementos de aceleração de processamento. Em ambos os trabalhos, porém, é feita uma caracterização das classes de programas existentes, para melhor comparar a utilização de tais aceleradores de execução de software. [CLS+08] baseia-se em uma classificação em 7 classes de programas de alta performance, chamadas Anões (do inglês Dwarves), caracterizando-os para encontrar semelhanças capazes de facilitar a programação de tais programas. Esse trabalho foi estendido por [ABD+09], visando encontrar outros Dwarves capazes de abranger essa visão de programas para outras aplicações, tais como software embarcados e algoritmos de criptografia.

Em [NNH+12] os autores propõem um framework baseado em FPGA para aceleração de simulação de fenômenos físico-temporais complexos. Este tipo de simulação se caracteriza por apresentar acessos a endereços de memória não sequenciais causando um grande impacto no desempenho computacional. Neste trabalho a localidade de dados é melhorada por usar uma técnica de renumeração de nodos resultando em padrões de acesso sequenciais. Além disso, o acesso a memória embarcada no FPGA aumenta o reuso de dados e reduz a largura de banda de acesso à memória.

Em [VPB13] os autores propõem uma aceleração da execução da transformada rápida de Fourier 3D (3D-FFT) usando FPGA. Segundo os autores, nem todos os FPGAs são indicados para esta aplicação, pois existem FPGAs que operam a frequências mais baixas que um processador de propósito geral. FPGAs heterogênios, ou seja, com recursos internos tais como processadores DSP, multiplicadores e memórias são indicados para este tipo de aplicação. Os autores obtém ganhos de desempenho de até 1900x em relação a uma execução em software em um processador de propósito geral para uma FFT 3D aplicada sobre 2048 pontos.

Objetivos

Objetivo principal deste trabalho é obter uma arquitetura de hardware capaz de acelerar o processamento de ataques criptográficos a fim de permitir que criptoanálises possam ser realizadas em dispositivos eletrônicos equipados com algum tipo de proteção a SCA tais como smart cards, os quais

demandam uma grande quantidade de traços de consumo de potência e consequentemente exigem um grande esforço computacional.

Objetivos Estratégicos

- Contribuir para a pesquisa de segurança a Sistemas Embarcados
- Contribuir para o desenvolvimento de arquiteturas de alto desempenho de processamento
- Dominar o algoritmo de análise diferencial de potência visando encontrar os pontos críticos de sua execução
- Dominar e avaliar métodos para paralelizar o algoritmo
- Propor uma arquitetura de hardware baseadas em FPGAs para o processamento de alto desempenho.

Objetivos Específicos

- Paralelizar o algoritmo de criptoanálise
- Identificar as funções que demandam processamento intensivo e o tipo de operação executada
- Propor uma arquitetura dedicada capaz de executar tais operações críticas em hardware
- Validar a arquitetura proposta e avaliar seu desempenho para a dada aplicação
- Avaliar o canal de comunicação entre o PC e FPGA de modo que este tenha o menor impacto possível no desempenho do sistema proposto.

5.2 Computação em Grade e em Nuvens

A Computação em Nuvem ou *Cloud Computing*, é um paradigma de computação distribuída que se propõe a fornecer infraestrutura computacional, plataforma de desenvolvimento e *software* aplicativo na forma de serviços altamente escaláveis, dinâmicos e entregues sob demanda aos usuários finais [FG+09a].

A ideia central desse paradigma é promover a redução de custos operacionais através da otimização no uso do recursos computacionais [FZRL08].

Essa relação de consumo é regulada através dos Acordos de Nível de Serviço (SLA), o qual estabelece os níveis aceitáveis de qualidade de um serviço (QoS) entregue a um consumidor por um prestador de serviço.

Outra característica central da Computação em Nuvem é o uso de virtualização. A virtualização oferece a abstração necessária para que os recursos computacionais físicos (processamento, armazenamento, recursos de rede) possam ser vistos como um conjunto de recursos mais abstratos, permitindo que serviços possam ser construidos sobre estes de forma independente [FZRL08].

Como uso da virtualização é possível compartilhar a mesma máquina física com múltiplos sistemas operacionais e/ou aplicações de usuários finais, promovendo o isolamento entre os mesmos [AAPZ10]. Essa propriedade garante um nível de encapsulamento e flexibilidade que permite que tais máquinas possam ser manipuladas e tenham suas aplicações migradas entre *servidores*; com maior facilidade do que se isto ocorresse usando-se somente máquinas físicas. Essas características fornecem uma melhor segurança, gerenciamento e isolamento, e tem sido centrais para a disseminação do uso da Computação em Nuvem [APTZ12].

Segundo Koomey [Koo11], no período de 2005 a 2010 a energia consumida na manutenção dos data centers aumentou em 56%. No final desse período, a energia utilizada para manter essas estruturas, correspondia a aproximadamente 1,5% do total da energia consumida globalmente. Esse consumo não contribui somente para causar impacto na economia, aumentando os custos operacionais, mas também influenciam muito no impacto ambiental, principalmente no aumento da emissão de dióxido carbono resultante das fontes de geração de energia. Dessa forma, existe uma preocupação em otimizar o uso de recursos, observando a eficiência energética e sem comprometer o desempenho dos serviços fornecidos [BAB12].

Com o uso mais eficiente da infraestrutura computacional, a Computação em Nuvem pode também contribuir com a redução nas emissões de dióxido carbono, colaborando com a diminuição do impacto causado na natureza [The12, BAB12].

Considerando este cenário, as frentes de pesquisa desta seção do projeto tem como premissa comum contribuir para a redução do consumo e para uso mais eficiente dos recurso energéticos utilizados pelas infraestruturas que suportam o ambientes em nuvem, colaborando assim com a preservação das fontes de energia não-renováveis, e do meio ambiente como um todo.

5.2.1 Mecanismos para Economia de Energia no Escalonamento de Aplicações

Problema de pesquisa

Recentemente, reduzir o consumo de energia tornou-se um fator de grande importância para a indústria por motivos econômicos, ambientais e de imagem pública. A questão financeira vem do fato de que a energia elétrica é um serviço pago pela quantidade utilizada. Logo, reduzir o consumo de energia significa economizar dinheiro. As questões ambientais e de imagem pública vem do fato de que produzir energia no mundo produz CO_2. De acordo com [PS07], uma diminuição de pelo menos 15% nas emissões de CO_2 até 2020 é necessária para manter o aumento de temperatura global menor que 2 °C. Por outro lado, [Nor91] sugere que uma redução de $1/3$ nas emissões de gases estufa pode gerar uma economia maior do que o investimento necessário para alcançar a meta.

Os *datacenters* que provém os serviços de computação na nuvem precisam de máquinas e de infraestrutura de rede de alto desempenho, na maioria das vezes com recursos redundantes. Para manter o funcionamento dessas máquinas, grande quantidade de refrigeração é necessária. Além disso, essa infraestrutura costuma ser projetada para funcionar nos momentos de pico[BCRR12]. Logo, grande quantidade de energia é consumida mesmo quando os recursos não são utilizados. Portanto essa energia é desperdiçada.

Em [Pla07] afirma-se que aproximadamente 2% de toda a emissão humana de CO_2 é causada pelo setor de tecnologia de informação. Logo, a diminuição no consumo de energia desse setor terá impactos fundamentais, tanto economicamente como ambientalmente. Dada a tendência de empresas migrarem seus serviços de TI para nuvens, reduzindo o consumo de energia das suas próprias instalações, deve-se concentrar em garantir redução do consumo de energia nos datacenters que abrigam os recursos de nuvens já que esses ambientes tenderão a ser os vilões em termos de consumo de energia nos próximos anos.

Desenvolver mecanismos, como escalonadores energeticamente eficientes, não só tem importância econômica e ambiental, mas também representa um ponto de pesquisa em aberto na área de computação em nuvem, como pode ser observado na citação a seguir encontrada na última edição da revista IEEE Communications Magazine:

> *"Another aspect that should be considered while provisioning resources is energy consumption. This aspect is getting more attention from industrial and government parties. Calls for the sup-*

port of green clouds are gaining momentum. With that in mind, resource allocation algorithms aim to accomplish the task of scheduling virtual machines on the servers residing in data centers and consequently scheduling network resources while complying with the problem constraints." [SJSO13]

Essa revista está na terceira posição em termos de fator de impacto de revistas da área de telecomunicações segundo o Thomson Reuters Journal Citation Reports (JCR) [Reu].

Diferente de outros mecanismos voltados para escalonar aplicações em nuvens com o objetivo de reduzir o consumo energético, esta proposta tem como foco a preparação das tarefas *antes* do escalonamento e o desenvolvimento de novos algoritmos de escalonamento que utilizem avanços recentes na área de infraestrutura de rede.

Estado da Arte

Diversos algoritmos de escalonamento para nuvens já foram propostos [HsCtH13]. Mais recentemente, escalonadores que levam em conta energia elétrica também foram propostos [KCS12, LDRC12, WCC13, CtJ12, HSL$^+$12].

Em Kim *et al.* [KCS12] é proposta uma metodologia de avaliação do consumo de energia de uma máquina virtual (MV) MV i baseada na quantidade de acessos à memória, instruções e núcleos virtuais ativos em algum instante t. Os autores propõem a Equação 1 para estimar o consumo de energia.

$$E_{MV_i} = C_1 N_i + C_2 M_i - C_3 S_t \tag{1}$$

Na Equação 1, N_i corresponde ao número de instruções, M_i aos acessos à memória e S_t é o número de núcleos ativos em t, e C_1, C_2 e C_3 são coeficientes determinados por regressão.

Usando esse modelo, podemos estimar o gasto de energia de uma MV durante a execução de uma aplicação e utilizar essa informação durante o processo de escalonamento. Contudo, esse modelo não leva em consideração a energia consumida na transferência de dados entre as MVs. Portanto, o consumo total de energia de uma aplicação distribuída não seria simplesmente a soma dos E_{MV_i}. Conforme observado por Baliga et al. [BAS$^+$08], o consumo de energia dos elementos de interconexão de redes não pode ser ignorado principalmente em datacenters onde as comunicações são realizadas em taxas da ordem de Gigabits por segundo, taxas essas que exigem dispositivos (placas de rede, amplificadores, comutadores, roteadores) que

consomem mais energia do que a geração anterior de equipamentos que operavam à taxas de centenas de Megabits por segundo.

Usando esse modelo, [KCS12] propõe um algoritmo de escalonamento que atribui a cada MV instanciada um consumo máximo de energia por período. Se esse limite é ultrapassado, a MV fica ociosa. Entretanto, ela ainda consome energia e não contribui em nada à tarefa.

Em [LDRC12], Liu *et al.* formulam um modelo de programação inteira com o objetivo de minimizar o consumo de energia das tarefas. Contudo, o *makespan* das tarefas não é levado em conta no problema; é apenas calculado um número mínimo de servidores que devem ser conectados para que as aplicações sejam executadas dentro de um tempo máximo.

A técnica de DVFS (*dynamic voltage and frequency scaling* – Mudança Dinâmica de Tensão e Frequência) é frequentemente utilizada em equipamentos eletrônicos para reduzir seu consumo de energia. Sabe-se que o consumo de energia de um circuito integrado é dado pela Equação 2 [WCC13].

$$E \sim fcv^2 \tag{2}$$

Na Equação 2, f é a frequência, c é a capacitância e v a tensão. Logo, variando-se a frequência e a tensão, alteramos o consumo de energia do circuito. O DVFS permite que variemos esses dois parâmetros de forma a satisfazer as restrições do problema a ser resolvido consumindo o mínimo de energia necessária.

Wu *et al.* [WCC13] propõe um algoritmo que integra DVFS ao escalonamento. Nele, uma tarefa é atribuída a um servidor que satisfaça duas desigualdades. A primeira afirma que o servidor tem um processador com frequência superior a um mínimo requerido e a segunda que o servidor escolhido dentre todos os possíveis seja o que satisfaça com mais folga esses requisitos. Entretanto, o consumo de energia das próprias MVs não é levado em consideração e como várias MVs podem ser executadas em um mesmo servidor concorrentemente, é irreal medir os gastos de energia de uma aplicação diretamente por quantas instruções os servidores executam, sendo preferível uma abordagem como a mencionada em [KCS12], que apesar de indireta é individual para cada aplicação.

Já Chang-tian *et al.*[CtJ12] utilizam algoritmos genéticos e DVFS para a otimização conjunta de economia de energia e *makespan* das aplicações. Os algoritmos propostos usam um parâmetro de conversão entre energia e desempenho.. Algoritmos genéticos, contudo, são notadamente lentos e heurísticas mais eficientes podem ser utilizadas em conjunto das técnicas propostas para atingir melhores resultados.

O algoritmo proposto por Huang *et al.* em [HSL+12] chega a decisões de forma global sobre a atribuição de tarefas não críticas, em oposição a decisões gulosas. O trabalho obtém economia ao redor de 30%, em média, em comparação com os gastos de energia do algoritmo HEFT, um algoritmo bastante utilizado para escalonamento de tarefas em sistemas distribuídos. Contudo, o algoritmo não leva em conta o consumo de energia de nenhum componente dos PCs da rede além do processador. Logo, gastos da memória e discos rígidos não são contabilizados.

Diferente de grades que tem como objetivo principal a execução de aplicações que façam uso intenso da capacidade de processamento dos nós da grade, em nuvens, os objetivos são diversos. Usuários de nuvens podem por exemplo alocar recursos para servir conteúdo HTTP via servidores web com balanceamento de carga, uma tarefa que não necessariamente vai consumir muito poder de processamento de um equipamento. Por conta disso, a execução de aplicações em nuvens tem uma fase inicial que é a fase de alocação de máquinas virtuais. Cada tarefa, com suas particularidades, costuma ser associada a uma máquina virtual, que é uma imagem de um sistema operacional contendo tudo que a tarefa precisa para executar. Antes dessa tarefa ser executada essa imagem precisa ser instanciada em uma máquina física. Essa fase inicial impede que algoritmos de escalonamento como o HEFT sejam utilizados. Entretanto, em [BCdF11], é proposto um algoritmo que altera o DAG de uma aplicação de forma que ela possa ser escalonada pelo HEFT ou por outros algoritmos para computação distribuída existentes sem necessidade de incluir particularidades de nuvens nos algoritmos de escalonamento. A ideia por trás desse algoritmo é incluir tarefas iniciais no DAG que representem as instanciações de máquinas virtuais, que devem ser realizadas antes das tarefas serem executadas. Assim, um escalonador vai realizar o escalonamento como se o DAG fosse um DAG convencional de uma aplicação distribuída.

Nesse algoritmo, tarefas em um mesmo caminho no grafo que dependam da mesma MV tem apenas uma MV instanciada. Isso favorece o paralelismo da aplicação, diminuindo o tempo esperado para que ela execute, como foi verificado em [Vir].

Equipamentos de interconexão de redes, assim como os computadores que processam as aplicações em sistemas distribuídos, também consomem energia. Algumas pesquisas recentes propõem mecanismos a serem implementados nas camadas de transporte e enlace da arquitetura Internet com o objetivo de aumentar a eficiência energética durante o processo de transferência de dados via rede. Por exemplo, enlaces com taxa adaptativa podem mudar a capacidade de transmissão de acordo com a sua demanda. Como

o consumo de energia de um enlace *ethernet* é diretamente proporcional à sua capacidade e independente da sua utilização [BCRR12], modificar a capacidade em função da demanda levará a um consumo energético mais eficiente. Há duas formas possíveis de realizar essa variação: desligando os enlaces temporariamente, método conhecido como *sleeping mode*, ou reduzindo a latência (*rate switching*). Comparações entre os dois são feitas em [MGW09] e [WAT12]. Ambas ressaltam que o *sleeping mode* é menos complexo e obtém economias semelhantes ao *rate switching*.

Na camada de transporte há a proposta do TCP "verde" [IC98]. Nele, uma opção TCP_SLEEP é adicionada ao cabeçalho do TCP. Quando um cliente envia essa opção para um servidor que a compreenda, o cliente notifica que irá "dormir". O servidor, então, não envia mais nenhum pacote para o cliente, apesar de manter a conexão aberta. Além disso, o socket associado à essa conexão é fechado de forma que não ocorra um acúmulo de informação excessivo. Quando o cliente "acorda", ele notifica o servidor e o fluxo de informação é reiniciado.

De acordo com o padrão do TCP [Ste94], quando uma opção não é reconhecida por um host ela é ignorada. Assim, o TCP verde é compatível com as variações de TCP encontrados nos sistemas operacionais modernos e pode ser usado em redes heterogêneas.

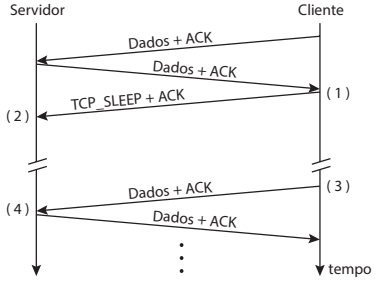

Figura 1: Diagrama de estado demonstrando o estado "adormecido" do TCP verde (Extraído de [IC98])

A Figura 1 mostra, em (1), um cliente enviando a um servidor um pacote com a opção TCP_SLEEP. Em (2), o servidor bloqueia o socket e a conexão entra em modo "adormecido". Entre (2) e (3), o cliente pode ser desligado, já que não ocorre envio de informação. Em (3), o cliente acorda e envia um

pacote ao servidor, que a recebe em (4) e retoma a conexão.

Diferentemente dos algoritmos propostos nessa sessão, propomos duas formas de economizar energia sem sacrificar QoS em nuvens. A primeira envolve a modificação das representações das tarefas e a segunda utiliza avanços recentes nas camadas de transporte e de enlace.

Objetivos

O objetivo deste tópico do projeto é propor algoritmos para a alocação de recursos em nuvens com foco em economia de energia. Esses algoritmos serão de dois tipos: (i) modificadores de representação das tarefas e (ii) escalonadores que utilizam desenvolvimentos recentes na área de infraestrutura de redes. Esses algoritmos avançarão o estado da arte e diferirão dos existentes da literatura por tornar a própria aplicação energeticamente eficiente antes de ser escalonada, modificando sua representação, e também por integrar ao processo de escalonamento decisões sobre a participação ou não de computadores em operações na nuvem através das camadas inferiores da arquitetura Internet.

Para a primeira abordagem o nosso objetivo é inserir no DAG que representa a aplicação informações sobre o consumo de energia de cada tarefa e usar tais dados para decidir se a instanciação de uma nova máquina virtual deve ser feita ou não. A abordagem inicial será adaptar diretamente o algoritmo de [BCdF11]. Contudo, toda vez que uma máquina virtual deve ser instanciada e ela é redundante (ou seja, já foi instanciada pelo menos uma vez), verificamos se o caminho ao qual será conectada terá um número mínimo de instruções. Caso contrário, ela não é instanciada. A ideia por trás de verificar um número mínimo de instruções será feito pela MV para justificar o consumo de energia de mais uma instanciação.

Para a segunda abordagem o nosso objetivo é que o escalonador tome decisões que levem ao desligamento ou não dos enlaces interligando computadores que compartilhem recursos de uma nuvem dependendo dos requisitos da aplicação. O escalonador também terá incluído um módulo para permitir o ajuste da capacidade dos enlaces de forma a reduzir o consumo total de energia.

Ainda na segunda abordagem pretendemos desenvolver um escalonador que tire proveito da opção TCP_SLEEP do recente "TCP verde" para desativar máquinas supérfluas em um datacenter, economizando energia.

As metas são:

- Propor e implementar novos algoritmos para modificação dos DAGs,

incorporando as instanciações de máquinas virtuais;

- Propor um escalonador que tire proveito do TCP verde e da adaptação de enlaces para desativar máquinas superfluas em um data center e reduzir as taxas de transferências de dados, economizando energia.

Resultados esperados

Os algoritmos de alterações de DAGs terão seu desempenho medido em ambiente de simulação. Os DAGs modificados serão escalonados pelo HEFT. O conjunto de *scripts* que serão utilizados já foram escritos e estão disponíveis em [VB].

O novo escalonador energeticamente eficiente será integrado ao simulador de rede ns-3 [ns3]. Optamos pelo uso do ns-3 ao ns-2 devido a ele ter um desempenho melhor [WvLW09].

Usaremos ambas as abordagens para comparar e unir as soluções e realizar análises sobre quando cada método ou uma união deles deve ser utilizada.

Em todos os experimentos de simulação serão usadas informações extraídas de aplicações executadas em nuvens reais ao redor do mundo como os traces do Google [Joh11]. Informações sobre o consumo de energia de computadores modernos serão extraídas do projeto SPEC [SPE13]. Informações sobre adaptação de enlace serão extraídas de sites de fabricantes de equipamentos de rede.

Após experimentos em ambientes simulados, usaremos imagens modificadas de GNU/Linux [Sta] virtualizadas na Amazon Web Services [Ama]. Solicitaremos o uso sem custo para propósitos educacionais e, caso o pedido seja recusado, usaremos fundos de outros projetos.

Todos os resultados obtidos nesse tópico do projeto serão formatados em artigos a serem submetidos para conferências e revistas com nível Qualis A. Todos os códigos produzidos serão disponibilizados publicamente no website do projeto a ser desenvolvido.

5.2.2 Alocação Energeticamente Eficiente de Máquinas Virtuais

Problema de pesquisa

A relação entre consumo de energia e desempenho dos computadores tem aumentado constantemente a cada ano. Se esta tendência continuar, o custo da energia consumida por computadores durante a sua vida útil vai ultrapassar o custo de aquisição desses computadores. O problema é ainda pior

para infraestruturas de clusters, centros de processamento de dados e computação em larga escala. Nesses cenários tem se observado um crescimento de 16 a 20% ao ano no consumo de energia, o que corresponde a dobrar o consumo de energia a cada 4 a 5 anos [BBLZ10].

Neste tópico do projeto nós propomos duas linhas de pesquisa. A primeira é voltada para a alocação eficiente de máquinas virtuais em máquinas físicas e a segunda é voltada para o estudo da confiabilidade de nuvens computacionais que venham a usar os mecanismos de alocação propostos na primeira linha de pesquisa. Nesse tópico consideramos que as máquinas físicas possuem diversos recursos como CPU, memória e placas de rede.

Para a alocação eficiente de máquinas virtuais vamos considerar as conclusões apresentadas em diversos trabalhos como [FWB07] que mostram que computadores equipados com diversos núcleos e processadores são mais eficientes energeticamente quando operam próximos às suas capacidades máximas. Levando isso em consideração nós trabalharemos em duas abordagens. A primeira é baseada no algoritmo da mochila e a segunda é baseada em computação evolutiva. Ambas as abordagens serão comparadas com uma estratégia que não leve em consideração a eficiência energética a fim de se comprovar as vantagens de realizar alocação de máquinas virtuais que mantenha os computadores em utilização próxima às suas capacidades máximas. É importante observar que esse tipo de alocação pode aumentar a quantidade de computadores livres para atender às novas requisições por alocação de máquinas virtuais. Dessa forma, a taxa de bloqueio de novas requisições também será avaliada.

Utilizar os computadores próximos às suas capacidades pode trazer como consequência a redução do tempo de vida útil do hardware. Assim, um modelo matemático similar àquele proposto em [CMT+13] será construído com o objetivo de estimar métricas de confiabilidade de nuvens que venham a utilizar os algoritmos de alocação de máquinas virtuais propostos.

Estado da Arte

É possível encontrar na literatura alguns estudos sobre a otimização do consumo de energia em ambientes de computação em nuvem com virtualização baseada em hipervisor. O trabalho apresentado em [BB13] propõe uma abordagem para resolver o problema de detecção de sobrecarga em computadores. O objetivo é prever os eventos de sobrecarga, manter o nível de QoS requisitado pelos clientes mas sem gerar ineficiência energética. Sem a utilização da proposta dos autores, uma abordagem trivial seria migrar uma máquina virtual aleatória de um computador sobrecarregado para um

computador que tenha capacidade de receber essa máquina virtual, o que dificilmente levaria a uma alocação ótima. A proposta dos autores, baseada em um modelo de cadeias de Markov, consegue alcançar um resultado próximo do ótimo, mantendo a eficiência energética e sem descumprir os requisitos de QoS.

O OpenStack Neat [BB12] é um framework de software livre voltado para a consolidação dinâmica de máquinas virtuais em nuvens baseadas na plataforma OpenStack. O framework invoca a API do OpenStack de modo a definir alocações de máquinas virtuais que mantenham as máquinas físicas operando em um nível de alta eficiência energética.

A maioria das propostas para alocação de máquinas virtuais observam a utilização da CPU das máquinas físicas e tomam decisões baseadas nessa observação. Isso faz sentido, uma vez que existe uma relação entre o consumo total de energia de um computador e sua utilização de CPU, como observado em [FWB07] que propõe um modelo de consumo de energia que relaciona esse consumo linearmente com a utilização da CPU. É interessante notar que em centros de processamento de dados, como observado em [BH07], o valor médio da utilização das CPU fica em torno de 35%.

Modelos para predição do consumo de energia são capazes de fornecer previsões próximas do real para aplicações que fazem uso intensivo de CPU. No entanto, esses modelos tendem a ser imprecisos quando dispositivos de E/S e memória são levados em consideração como concluiu [DMR10]. Por esta razão, nas nossas propostas pretendemos levar em consideração diversos dispositivos além da CPU a fim de aumentar a precisão do mecanismo de alocação de máquinas virtuais de modo que as alocações propostas sejam de fato eficientes energeticamente.

Diversos simuladores de computação em nuvem disponíveis, como o CloudSim [RB13] fazem simulações de algoritmos de alocação de máquinas virtuais apenas com base na observação da CPU. Por isso pretendemos desenvolver um framework de simulação para avaliar os nossos mecanismos que leve em consideração outros dispositivos como memória e placas de rede. A ideia é que esse framework seja fácil de ser expandido de modo a permitir a utilização por outros pesquisadores que trabalhem com mecanismos para computação em nuvem.

Objetivos

A alocação de máquinas virtuais visando a eficiência energética baseia-se no fato de que computadores ociosos podem ser suspensos para modos de operação com baixo consumo de energia. Além disso também leva-se em

consideração o fato de que esses computadores podem ser reativados remotamente utilizando a tecnologia Wake-on-Lan. Entretanto é importante levar em consideração os requisitos de Qualidade de Serviço na hora de alocar as máquinas virtuais. Fazer uma alocação considerando apenas a economia de energia pode levar a altas taxas de bloqueio e insatisfação dos usuários que estejam requisitando alocações de máquinas virtuais em uma nuvem.

A alocação de máquinas virtuais considerada no nosso trabalho baseia-se nos passos propostos em [BB12]: é realizada uma alocação de máquinas virtuais preliminar e periodicamente algoritmos são executados a fim de verificar se há ganhos, em termos de consumo de energia, caso máquinas virtuais alocadas em diferentes máquinas físicas sejam consolidadas.

O objetivo desse tópico do projeto é apresentar propostas para alocação de máquinas virtuais que minimizem a quantidade de máquinas físicas utilizadas. Em outras palavras, queremos maximizar o número de máquinas virtuais por máquina física considerando restrições de utilização de recursos e de cumprimento de requisitos de QoS, a fim de garantir economia de energia sem um aumento significativo na taxa de bloqueio das requisições de alocações de máquinas virtuais.

Além disso pretendemos desenvolver um modelo matemático que permita prever a disponibilidade e a confiabilidade de nuvens que utilizem as propostas de alocação de máquinas virtuais a serem desenvolvidas, tomando como base o trabalho desenvolvido em [CMT+13].

As metas são:

- Propor algoritmos para alocação de máquinas virtuais baseados em duas abordagens: (i) abordagem baseada no problema da mochila multidimensional com múltiplas escolhas, e (ii) abordagem baseada na computação evolutiva.
- Avaliar os algoritmos propostos.
- Propor um modelo matemático que permita prever a confiabilidade e a disponibilidade de nuvens que utilizem os algoritmos propostos.

Resultados esperados

Os novos algoritmos para alocação de máquinas virtuais serão avaliados em um framework de simulação a ser desenvolvido. As bibliotecas `OpenOpt` [Kro13] e `Inspyred` [Gar13] serão utilizadas no desenvolvimento do framework para facilitar as implementações dos algoritmos baseados no pro-

blema da mochila e dos algoritmos baseados em computação evolutiva respectivamente.

Experimentos serão realizados usando traces de sistemas reais. Serão analisados traces com registros de mais de 11.776×24 horas disponibilizados pelo projeto CoMon, uma infraestrutura de monitoramento do PlanetLab. Alguns dos traces coletados entre Março e Abril de 2011[1] serão escolhidos aleatoriamente. Esses traces incluem dados coletados a cada 5 minutos de mais de 1.000 máquinas virtuais alocadas em máquinas físicas espalhadas por mais de 500 localidades ao redor do mundo.

Todos os resultados obtidos nesse tópico do projeto serão formatados em artigos a serem submetidos para conferências e revistas com nível Qualis A. Todos os códigos produzidos serão disponibilizados publicamente no website do projeto a ser desenvolvido.

5.2.3 Composições de Serviços em Nuvens Orientadas ao Contexto de Economia de Energia

Problema de pesquisa

Análises orientadas a objetivos tem sido propostas na literatura de engenharia de requisitos para captar a intenção por trás de requisitos de software [YM98]. Os objetivos são uma abstração útil para representar as necessidades e expectativas das partes interessadas, e oferecem uma maneira muito intuitiva para extrair e analisar requisitos. *Goal-Models* (GM) são uma ferramenta de engenharia de requisitos bem definida e aceita na comunidade de engenharia de requisitos. Ela é utilizada para descrever sistemas como um conjunto de metas e tarefas de baixo nível. Dessa forma busca-se facilitar a compreensão do sistema como um todo.

No contexto de composição orientada a serviços, mais especificamente em coreografias de serviços web em que uma aplicação é representada por papeis que podem ser realizados por diversos serviços, há a proposta de tratar a composição como um conjunto de agentes. Cada agente é responsável por agir como um proxy para os serviços que possam realizar um dado papel. Essa visão de composição de serviços pode ser mapeada para ser realizada sobre nuvens computacionais [CHO13].

Em um cenário de composição orientada a serviços com agentes que utilize GM, o comportamento esperado e os objetivos que um agente deve atender não podem ser estáticos já que ambos dependem do contexto. Por exemplo, se n serviços que consumam muita CPU estão sendo executados em

[1] `http://github.com/beloglazov/planetlab-workload-traces`

máquinas virtuais de um mesmo datacenter e um agente detecte um aumento no consumo de energia do sistema de refrigeração do datacenter, ele pode passar parâmetros para esses serviços de modo a reduzir o consumo de CPU e consequentemente reduzir o consumo de energia do sistema de refrigeração. Tais variações no comportamento dos agentes/serviços não conseguem ser capturadas diretamente pelo GM. Para isso é necessário utilizar a abordagem de *Context Goal-Models* (CGM) proposta em [ADG10].

Nesse tópico do projeto nós desenvolveremos um CGM que modele situações que devam levar a ações com o objetivo de economizar energia em nuvens computacionais utilizadas para a execução de composições de serviço. Além disso um algoritmo para decidir se um objetivo pode ser alcançado (por exemplo, economizar a energia em $x\%$) considerando o contexto atual do ambiente e as suas restrições, também será desenvolvido.

Estado da Arte

O conceito de cumprimento de um objetivo já foi abordado em trabalhos anteriores. Por exemplo, Baresi [BP10] define o que ele chama de objetivos ao vivo. Nesse trabalho os objetivos tornam-se entidades atuantes que, com base em pré-condições, reagem para eliminar/reduzir/evitar obstáculos que sejam identificados. Entretanto, os objetivos são tratados individualmente e suas ações não dependem de qualquer outro objetivo, dificultando a cooperação entre diferentes objetivos e agentes para atender níveis globais de qualidade para a aplicação como um todo.

Dalpiaz [DGM11] estuda o fato de que em alguns casos atender um sub-objetivo não é suficiente para atender um objetivo. De forma similar, a falha em atender um sub-objetivo muitas vezes não significa falhar em atender o objetivo. Apesar dos seus estudos, Dalpiaz não considera cenários dependentes de contexto.

Em [SM11], o processo de adaptação de um sistema é modelado para um sistema de controle. Os requisitos do sistema servem de entrada para um framework de adaptação que é integrado com as informações de contexto. Indicadores do ambientes realimentam o sistema de controle com as convergência de requisitos observadas. Dessa forma ações podem ser tomadas de modo a levar o sistema para o que os autores chamam de estado consistente.

Não é do nosso conhecimento que haja alguma proposta de utilizar CGM para reduzir o consumo de energia em nuvens.

Objetivos

Nesse tópico do projeto nós desenvolveremos um CGM que modele situações que devam levar a ações com o objetivo de economizar energia em nuvens computacionais utilizadas para a execução de composições de serviço. Além disso um algoritmo para decidir se um objetivo pode ser alcançado (por exemplo, economizar a energia em $x\%$) considerando o contexto atual do ambiente e as suas restrições, também será desenvolvido.

As metas são:

- Desenvolver um Context Goal-Model (CGM) que oriente a escolha dos serviços de uma composição com o objetivo de reduzir o consumo de energia de datacenters.
- Desenvolver um algoritmo que forneça a possibilidade, ou não, de atender um dado objetivo relacionado com o consumo de energia.

Resultados esperados

Espera-se que através de medições possa-se confirmar a eficácia da utilização do CGM para controlar o consumo de energia de datacenters utilizados para hospedar serviços web a serem usados por composições de serviços.

Todos os resultados obtidos nesse tópico do projeto serão formatados em artigos a serem submetidos para conferências e revistas com nível Qualis A. Todos os códigos produzidos serão disponibilizados publicamente no website do projeto a ser desenvolvido.

5.2.4 Uso do Modelo BSP em Nuvens

Problema de pesquisa

Modelos de infraestrutura e conceitos que abordam Computação em Nuvem estão se tornando padrões para várias aplicações. A escalabilidade é real e há vantagens econômicas em utilizar cenários de virtualização. Nesse ambiente o dimensionamento dos recursos deve atender os requisitos das aplicações, como processadores, memória e rede de comunicação. Os recursos instanciados podem ser monitorados para evitar congestionamentos e sobrecargas [KM05][BGLA13]. Um cenário ideal é compreendido como aquele que permite a busca da solução em menor tempo e utilizando o mínimo de recursos, desde sua inicialização. Uma solução para predição do cenário ideal de virtualização consiste em observar o custo assintótico das aplicações e o tamanho da entrada.

Um modelo de programação paralela candidato para predição em nuvem é o BSP (*Bulk Synchronous Parallel*). Fora da nuvem, o BSP é conhecido pela predição eficiente. Dentro da nuvem, o BSP é tema de pesquisas recentes, pois nesse contexto existem outras variáveis que tem efeito na latência e envolvem o uso compartilhado de recursos de processamento e comunicação em rede. Com base nesses conceitos, surge o questionamento:

Neste subprojeto, queremos verificar a possibilidade de usar a predição por meio da notação assintótica do modelo BSP, assim como verificar a sua eficiência para a antecipação de um cenário ideal de virtualização.

Estado da arte

A comunidade de Tecnologia da Informação adotou o modelo de Computação em Nuvem [AFG+10] como solução para sua escalabilidade. Provedores comerciais de Computação em Nuvem permitem o fácil acesso a um grande conjunto de recursos computacionais. Computação em Nuvem é um tipo de modelo de utilitários [Par66] onde a aplicação pode escolher, sob demanda, o tipo e a quantidade de recursos necessários em qualquer momento. Nesse ambiente são cobrados apenas os recursos efetivamente utilizados (como um serviço medido). As comunidades acadêmicas e científicas também foram atraídas por muitos dos benefícios trazidos pelas plataformas de Computação em Nuvem (virtualização, elasticidade de recursos, eliminação de configuração de hardware e custos de manutenção, etc.). No entanto, como Gupta e Milojicic [GM11] apontam, problemas como a má performance da rede, a variação de desempenho e a sobrecarga causada pelos sistemas operacionais virtualizados configuram novos desafios para a execução de Aplicações de Alto Desempenho sobre esse tipo de plataforma.

Pesquisas recentes [GFC+12][GM11][CJL+08][BHBE10] propuseram formas sofisticadas para conseguir um melhor desempenho em plataformas de Computação em Nuvem. Um melhor desempenho conquistado também está relacionado com a quantidade ajustada de recursos alocados em nuvem. Quando muitos recursos são alocados, podem ocorrer desperdícios. Quando poucos recursos são alocados, o tempo de execução pode ser mais lento e dessa forma gerar mais custos. O modelo de programação paralela *Bulk Synchronous Parallel* (BSP) é bem conhecido pela sua predição eficiente e pode potencialmente ser transportado para Computação em Nuvem, o que contribui para uma alocação mais precisa de recursos na plataforma.

O modelo BSP foi introduzido por Valiant [Val90]. O cálculo é organizado como uma sequência de super-passos. Cada super-passo é composto de

múltiplas fases computacionais paralelas, onde cada processador (em todos os nós) executa somente cálculos locais, seguido por uma fase de comunicação, em que as comunicações de dados envolvem todos os processadores. A fase de comunicação é conhecida como h-relação. Uma h-relação representa o envio e o recebimento de h dados em cada processador [G$\ddot{9}$7]. Há diversos trabalhos explorando como a h-relação pode ser efetivamente feita, provendo implementações eficiente em muitos cenários [Deh06]. Entre dois super-passos existe uma barreira de sincronização, o que garante o início simultâneo de todos processadores para a próxima etapa.

Trabalhos utilizando BSP como um modelo de programação para Computação em Nuvem apareceram pela primeira vez no contexto de aplicações de processamento de grafos de grande porte. Essas aplicações aparecem naturalmente em problemas de diferentes áreas do conhecimento (Ciência da Computação, Biologia, Engenharia, etc) e usam grafos para representar relações entre diferentes entidades, tais como enlaces em páginas da web, relações de usuários em redes sociais, ordem de fragmentos de DNA, etc.

Kechid e Myoupo[KM05] analisaram a predição de algoritmos BSP em cluster reais, não virtualizados em nuvem. O custo de uma aplicação BSP é dado pela soma dos super-passos. Isto significa que são somados os custos da computação local, da comunicação e da sincronização. O tempo da computação local é $w = máx\{w_i\}$ onde w_i é o número de operações aritméticas de cada processador i durante o super-passo. O tempo da operação aritmética é dado pelo resultado de uma operação de ponto flutuante, como multiplicação, ou divisão, ou adição, ou subtração. Portanto, o custo da computação depende também da arquitetura física.

O custo da comunicação é dado pela h-relação. O h é o número de mensagens comunicadas. Assim, $h = máx\{h_i\}$, onde $i \in \{1..p\}$ e p são os processadores. Há duas maneiras de se calcular o h:

1. quando é transmitido simultaneamente $h = máx\{h_i send, h_i receive\}$; e

2. quando há via única $h = máx\{h_i send + h_i receive\}$.

O total da comunicação é $g * h$, onde g é o tempo de transferência de uma palavra. O tempo da sincronização é dado por L, que possui a mesma unidade de medida de g.

Muitos autores tem debatido o custo do h. Bisseling[Bis04] apresentou que alguns autores consideram erroneamente o h de forma generalizada. Por exemplo, em dois cenários com 3 processadores. Ambos os cenários são 2-relações, mas no segundo há muito mais trocas e o custo deveria ser diferente,

pois há efeitos no cálculo do tempo. Kechid e Myoupo[KM05] reforçam que o g da comunicação é pior quando existem muitas mensagens pequenas. Eles também declaram que o tempo da comunicação também depende do padrão utilizado (*broadcast*, *gather*, *scatter*, etc) e propõem um Benchmark.

O custo de um super-passo BSP em cenários não virtualizados em nuvem é comumente conhecido por $S = w + g * h + L$. Em nuvem devem ser considerados outros tipos de latências, pois esse ambiente envolve o uso compartilhado de processamento, de memória e de rede de comunicação por máquinas virtuais distribuídas em diferentes nós de nuvem. A descoberta dos detalhes do custo assintótico e a alocação otimizada de recursos na nuvem configuram-se como um novo desafio.

Metas

O objetivo geral consiste em elaborar uma arquitetura para antecipar a alocação de recursos em nuvem usando o custo assintótico de aplicações BSP e o tamanho da entrada. Para tanto, as seguintes etapas serão realizadas:

- descrever o modelo BSP e sua evolução histórica;
- analisar diversas publicações científicas que tratam do custo assintótico de aplicações BSP;
- descrever as aplicações BSP e seus custos assintóticos;
- avaliar a eficiência da predição de algumas aplicações BSP;
- descrever arquiteturas de plataformas de nuvem;
- avaliar aplicações BSP em nuvem e a eficiência da predição;
- analisar ambientes de nuvem procurando detalhar custos que tem impacto no desempenho;
- desenvolver uma arquitetura que subsidie a predição de aplicações BSP em nuvem como forma de antecipar a alocação de recursos que afetam no desempenho; e
- avaliar a arquitetura desenvolvida executando aplicações BSP.

5.3 PAD em Arquiteturas Paralelas

Arquiteturas paralelas, dotadas de memória compartilhada ou distribuída, têm servido a Sociedade como alternativa ao processamento intensivo (de alto desempenho ou de larga escala) há muitos anos. Encontramos em TOP 500 (`www.top500.org`) uma lista atualizada com 500 das máquinas paralelas com maior capacidade de processamento instalada. No entanto, deve ser observado que a pesquisa em Processamento de Alto Desempenho para arquiteturas paralelas não deve ser restrito a estas grandes plataformas, mas sim a miríade de configurações disponíveis no mercado. Ainda assim, muitos dados podem ser obtidos a partir da TOP 500 a medida em que mostram tendências a serem incorporadas no futuro próximo em computadores produzidos em grande escala.

Neste sentido, é possível observar que as arquiteturas básicas (arquiteturas com memória compartilhada ou distribuída) são utilizadas de forma combinada. Também verifica-se que, no tempo, uma vez que esta lista registra dados desde 1993, diferentes tecnologias tiveram maior participação no incremento do poder computacional. Na sua última atualização contabiliza-se que 94% das configurações possuem processadores com 6 ou mais cores e 53 máquinas contam com co-processadores/aceleradores. Entendendo que não existem linguagens ou ambientes de programação que ofereçam recursos para explorar estes diferentes níveis de paralelismo, e mesmo o fato de inexistir um modelo universal para descreve-las, faz-se necessário o desenvolvimento de um novo conjunto de ferramentas apresentando um novo conjunto de abstrações para este contexto de programação paralela.

5.3.1 Uso do Modelo BSP em Ambientes com GPUs

Problema de pesquisa

Há mais de 50 anos nasceu a computação paralela com a finalidade de executar mais rápido soluções computacionais para problemas complexos. Com isto foram criados modelos de abstrações de máquinas paralelas, que buscavam oportunidades de ajudar aos desenvolvedores paralelos, assim como parametrizar o desempenho dos algoritmos em ambientes paralelos. Um dos modelos mais aceitos pela comunidade cientifica foi criado por Valiant em 1990, o modelo BSP (Bulk Synchronous Parallel).

Em 2007, a empresa NVIDIA criou o CUDA (Compute Unified Device Architecture), que é utilizado como uma linguagem de programação para utilizar as placas gráficas em computação de propósito geral, abrindo uma enorme gama de possibilidades. Em seguida, as empresas de fabricação de

GPU focaram suas pesquisas no hardware e aumento de número de núcleos em suas GPUs. Hoje existem GPUs com mais de 3000 núcleos de processamento. A principal objetivo neste trabalho é adaptar o modelo BSP para GPUs, com a finalidade de parametrizar o desempenho de algoritmos paralelos que rodem em Placas Gráficas e em outras arquiteturas massivamente paralelas.

Estado da arte

A computação de alto desempenho (HPC em inglês) e alguns modelos de computação paralela surgiraram por volta de 50 anos atrás, com o objetivo principal de encontrar soluções computacionais para problemas grandes e complexos [Fos95], tais como, simulação de partículas de alta energia, mecânica quântica, dinâmica molecular e bioinformática.

Novos paradigmas, como a computação em grade [FBA+03, Cas06, GKS+] ou computação em nuvem foram introduzidos para empregar os recursos de hardware heterogêneos distribuídos como um sistema virtual unificado [Fos01]. Hoje em dia, os cientistas podem acessar a muitos recursos (por exemplo, aplicações, armazenamento, CPUs, GPUs, sensores remotos, supercomputadores, entre outros) interligados como um sistema distribuído por toda a Internet [GKGF04]. A tendência dos sistemas distribuídos é que os recursos serão heterogêneos, estes não necessariamente tem que ser dedicado e eles devem pertencer a um sistema com alguns componentes essenciais de segurança, administração [eSKG+10, SKG+10], a heterogeneidade [CSS11], escalonamento de processos [GQ03, FG09b], tolerância a falhas [PGK11], balanceamento de carga [CGS07], etc.

As placas gráficas, ou unidades de processamento gráfico, como seu nome mesmo diz, foram inicialmente concebidas com o objetivo de acelerar as tarefas de gráficos 2D e 3D. No entanto, com a evolução da arquitetura, as GPUs passaram de um único núcleo a um conjunto de núcleos altamente paralelos para cálculos de propósito geral. Além disso, as GPUs podem estar em parte de uma grande quantidade de dispositivos eletrônicos, só para citar alguns: desktops, laptops, telefones celulares [NVI] e supercomputadores.

Há alguns anos, com o desenvolvimento de interfaces de programação de aplicativos e linguagens de programação para GPUs surgiu o conceito de Propósito Geral em GPU (GPGPU). Existem muitas evidências que GPGPUs tem excelentes resultados em muitas áreas científicas, no entanto, as ferramentas para aproveitar ainda mais este dispositivo paralelo e implementações mais refinadas ainda estão surgindo [NVI12, HZG08]. Pesquisadores nesta área podem criar aplicativos com um conceito de paralelismo,

que é capaz de executar tanto em arquiteturas de CPU e como em GPUs. No entanto, a implementação de um algoritmo, para execução paralela simultânea, não garante que a execução com a mesma eficiência nas duas arquiteturas. Em particular, algoritmos com paralelismo em grandes blocos de dados, podem obter grandes benefícios quando são implementados em GPU [ZRL12].

O modelo BSP (Bulk Synchronous Parallel) foi criado em 1990 por Valiant [Val90], o modelo BSP oferece uma abstração simples e precisa de uma máquina paralela. Também, oferece um modelo síncrono de programação paralela. Um algoritmo BSP consiste de um número arbitrário de supersteps. Todos os processadores são sincronizados entre supersteps, os processadores esperam até que cada processador termine seus cálculos locais em um superstep, antes de começar o próximo superstep.

A expressão mais comum para calcular o tempo de cada superstep é mostrada na equação abaixo

$$t = w + hg + l, \tag{3}$$

onde W é a computação máxima local em um passo, h o número máximo de mensagens enviadas ou recebidas por qualquer processador, g e l são os parâmetros relacionados com a largura de banda da rede e a latência. Portanto, um algoritmo BSP é completamente modelado pelos parâmetros (w, h, g, l).

Recentemente, o modelo BSP foi implementado em arquiteturas multicore [Val11]. A implementação do modelo BSP para multicore permitiu incorporar o tamanho da memória compartilhada como mais um parâmetro na computação. O modelo BSP é uma boa alternativa para resolver os problemas paralelos em arquiteturas massivamente paralelas.

Objetivos

Objetivo Geral

O objetivo geral deste projeto focará na adaptação do modelo BSP sobre placas gráficas e arquiteturas massivamente paralelas

Objetivos Específicos

- Testar diferentes sistemas de GPU com algoritmos e aplicações que apresentem o melhor desempenho na paralelização com o modelo BSP;
- Estudar e caracterizar o comportamento de entidades computacionais sobre GPUs massivamente paralelas;

- Criar um modelo algorítmico o modelo BSP para o estado de arte atual de GPUs e Supercomputadores;
- Propor opções para balanceamento de cargas, tolerância a falhas ou Check-pointing de processos em ambientes heterogêneos.

5.3.2 Uso de Formas de Paralelização Automática

Problema de pesquisa

É crescente a evolução das plataformas de processamento paralelo e onde cada vez mais estão presentes elementos heterogêneos. No cenário atual tem-se de um lado plataformas multicore com coprocessadores [Int13] e GPUs com grande poder de processamento, e de outro aplicações científicas com código legado que precisam usufruir do poder de processamento disponível com essas novas plataformas. O grande desafio para essas plataformas é a coexistência com as tecnologias e outras plataformas já consolidadas e amplamente utilizadas.

O desenvolvimento tem como base a utilização dos SDKs (*Sofware Development Kits*) fornecidos em conjunto com essas plataformas, utilizando como linguagem de entrada a linguagem C/C++. Entre as GPUs, as que a NVIDIA fabrica utilizam CUDA e as da AMD/ATI utilizam OpenCL [Khr13], que também tem sido utilizado pelos co-processadores da Intel. Frente a essas novas plataformas e modelos de programação não se tem a garantia e nem se pode exigir que aplicações existentes sejam reescritas na linguagem utilizada por essas novas plataformas. A coexistência com aplicações de código legado leva tanto o hardware quanto o software à manutenção de compatibilidade.

São necessárias soluções que deem suporte a execução de código de novas e de aplicações existentes em outros modelos, pois reescrever código pode custar caro em termos de tempo e muitos vezes é inviável. No intuito de facilitar o desenvolvimento e utilização de aplicações de código legado, tem-se proposto abordagens que utilizam diretivas de compilação para anotar o código e guiar o compilador na paralelização do código ou que utilizam ferramentas que fazem a paralelização automática do código sem alteração do código original.

Estado da arte

Para essas novas plataformas que tem com principais linguagens de entrada C/C++, além das bibliotecas dos kits de desenvolvimento, existem trabalhos

e ferramentas que geram código, como o compilador PGI e o OpenHMPP, que fazem a tradução *source-to-source* para código CUDA ou OpenCL.

Programar para essas novas plataformas não é uma tarefa trivial, isso faz com que a busca por ferramentas que facilitem o desenvolvimento de aplicações paralelas, diminuindo a complexidade envolvida, ganhe importância. Projetos tem movido esforços para prover mecanismos que facilitem o desenvolvimento.

Neste contexto duas abordagens tem se destacado, uma delas usa diretivas de compilação para guiar o compilador na tradução e paralelização do código e a outra aplica técnicas e modelos para detectar automaticamente quais regiões de código são paralelizáveis e então gera o código para a plataforma alvo.

Diretivas de compilação são utilizadas para a inserção de anotações e dicas no código original que possam guiar o compilador no processo de otimização de regiões consideradas paralelizáveis e na geração de código paralelo com o uso de bibliotecas fornecidas pelo ambiente de execução (*runtime*). Normalmente, são implementadas com o uso de diretivas de pré-processamento `#pragma` da linguagem `C/C++`. No pré-processamento, os *tokens* que compõem a diretiva serão tratados e estarão sujeitos a substituições feita por macros, por exemplo.

Desde o surgimento da plataforma CUDA, algumas ferramentas tem sido desenvolvidas com esse objetivo, entre elas estão o hiCUDA [HA09], [HA11] e [hP12], CGCM e mais recentemente atingiram grande destaque com a proposta do padrão OpenACC [Ope12a, Ope11] por um consórcio de empresas, entre elas a NVIDIA que é uma das principais fabricantes de GPUs. Outras ferramentas trabalham com a mesma ideia com objetivos similares, por exemplo a accULL [RLRFdS12] é uma implementação do padrão OpenACC com suporte a CUDA e a OpenCL. Para multicore, o OpenMP já utilizava um conjunto de diretivas de compilação para arquiteturas multicore de memória compartilhada [Ope12b] [CJvdP07]. O OpenMC [LME09, LE10] é uma extensão do OpenMP para suporte a GPU, utiliza o compilador Cetus em conjunto com o analisador OpenMP e o componente tradutor OpenMP-to-GPU (Omp2gpu ou O2G translator) para gerar código para GPU.

Do uso de diretivas de compilação é possível notar a transferência da complexidade de um cenário para outro, pois mesmo que o usuário seja poupado de aprender uma nova linguagem de programação ou de reescrever seu código, ainda é necessário que aprenda o conjunto de diretivas da ferramenta que utiliza e em alguns casos as anotações tornam-se tão ou mais complexas que reprogramar para a nova plataforma, pois há a necessidade de se especificar muitas restrições para o que o código final seja adequado.

A abordagem de paralelização automática de código está ligada à tradução de código para a arquitetura alvo sem modificações no código original. O compilador deve ser capaz de identificar e paralelizar regiões de código sem a intervenção do programador. Pode tanto considerar o processo partindo-se da tradução de código fonte para fonte, quanto da tradução de binários em tempo de execução. As ferramentas Par4All [ACE+12] e KernelGen [MLZ13] são exemplos dessas categorias, pois detectam regiões de código paralelizáveis e geram código automaticamente.

Objetivos

Esta linha de pesquisa tem como objetivo o estudo e a aplicação de técnicas de paralelização de código para uso eficiente de recursos em sistemas heterogêneos. Considerando o tema abordado e os resultados de experimentos preliminares concluí-se que as ferramentas apresentadas atendem em parte a geração automática de código, pois geram código para cada plataforma isoladamente, existindo portanto a necessidade de uma ferramenta que trabalhe com a paralelização automática para arquiteturas modernas, mas que considere cenários *multicore* e *multigpu*. Além disso, a ferramenta deve fazer uso eficiente dos recursos e do potencial de computação obtido pela cooperação dos elementos heterogêneos de processamento presentes nessas plataformas.

As ferramentas existentes não exploram totalmente os recursos disponíveis, escolhem durante a geração de código a GPU que o usuário especificar ou a que tem o maior número de *cores*. Mesmo que modelos de execução sejam distintos dos convencionais as ferramentas atuais não consideram uma solução híbrida unindo *multicore* e *multigpu* em um cenário misto.

Nesta linha tem-se investigado técnicas, ferramentas e suas aplicações para o uso de elementos heterogêneos em computação de maneira eficiente, considerando cenários mistos com a aplicação técnicas de detecção de regiões paralelizáveis e de paralelização automática para a melhoria do uso de recursos em todos os níveis sistêmicos.

Técnicas de Paralelismo de Granularidade Grossa podem ser aplicadas para o escalonamento de tarefas entre processadores e GPUs. Já as técnicas de granularidade mais fina podem ser empregadas para que *threads* explorem características específicas das arquiteturas tais como hierarquia de memória (caches), multiplicidade de *cores*, alocação de registradores e situações locais. Já o estudo técnicas de compilação e otimização de código é importante, principalmente para *loops*, pois suas regiões de código dominam o tempo de execução. O estudo dessas técnicas faz-se necessário para determinar se um dado laço ou região é paralelizável e se o código pode ser transformado

em *kernels* a serem lançados para execução em um dispositivo acelerador. Além disso, é necessário determinar se realmente ocorrerá uma aceleração da execução do trecho de código proporcionando melhoria no tempo de execução ou se o código deve ser executado pela CPU. O modelo de programação pode gerar código *multithread* com OpenMP, e esses *workers* em tempo de execução determinarão se fazem ou não uso do dispositivo acelerador disparando a execução de kernels na GPU.

5.3.3 Trabalhando com Atores em Ambientes NUMA

Problema de pesquisa

O modelo de atores é uma abstração de alto nível cujo objetivo é facilitar o desenvolvimento de aplicações paralelas. Para isto emprega estratégias como o isolamento do desenvolvedor da plataforma de hardware sendo utilizada. Neste modelo, a execução da aplicação se baseia em um ambiente de execução que é o responsável por garantir o uso eficiente da máquina em questão.

Os processadores mais recentes apresentam uma arquitetura hierárquica de memória. Além disto, algumas máquinas que contam com diversos processadores possuem arquiteturas onde a comunicação entre os processadores não é uniforme (Non-Uniform Memory Access - NUMA). É o papel do ambiente de execução fazer com que as aplicações escritas neste modelo de desenvolvimento possuam um bom desempenho nessas distintas arquiteturas.

Muitos dos sistemas hoje desenvolvidos no modelo de atores demandam um grande poder computacional, e esse poder computacional é frequentemente fornecido por máquinas NUMA. No entanto, nenhum dos ambientes de execução utilizados atualmente leva em consideração os diferentes aspectos dessas novas máquinas. Aqui descrevemos o projeto que visa adaptar e desenvolver técnicas para o desenvolvimento de ambientes de execução para o modelo de atores que com o conhecimento da arquitetura da máquina sobre a qual a aplicação será executada, seja capaz de tomar melhores decisões sobre o escalonamento e o equilíbrio de carga desses atores. Nossos resultados preliminares mostram que é possível obter melhoras de até 150% de desempenho ao mesmo tempo que limitamos o desempenho no pior caso a uma perda de 9%.

Estado da arte

Ambientes de execução para o modelo de atores estão disponíveis tanto como bibliotecas acessíveis as mais diferentes linguagens de programação assim como partes integrantes da própria linguagem. Os dois exemplos mais conhecidos são Erlang [Arm10] e Scala [OAC+06].

A maneira pela qual essas linguagens implementam o modelo de atores pode ser normalmente classificada em duas categorias, o baseadas em eventos ou baseadas em *thread*s. No modelo baseado em *thread*s, cada ator é representado no ambiente de execução como um *thread* ou processo do sistema operacional. Ou seja, existe uma relação direta entre cada uma das linhas de execução do sistema operacional e dos atores da aplicação. Já no modelo baseado em eventos, cada ator é representado por apenas uma simples estrutura de dados. O ambiente de execução, então, utiliza um pool de *thread*s para executar os atores. Ambas as abordagens tem as suas vantagens. A implementação baseada em *thread*s é bem mais próxima do sistema operacional e portanto é capaz de aproveitar de uma maneira mais direta das ferramentas e técnicas de otimização disponíveis para aplicações gerais. Por outro lado, no modelo baseado em eventos, o ambiente de execução é o responsável pelo escalonamento e balanceamento de carga dos atores da aplicação, assim o ambiente de execução é mais complexo do que aquele baseado em *thread*s. Por outro lado, como no modelo baseado em eventos cada um dos atores é representado por uma pequena estrutura de dados, é normalmente possível ter aplicações que utilizam um número muito superior de atores, sendo comum aplicações que têm centenas ou até mesmo milhares de atores executando de forma concorrente. Além disto, o ambiente de execução de atores, por ser uma camada intermediária entre o sistema operacional e a aplicação, tem mais informações sobre o comportamento da aplicação desta forma podendo tomar decisões de escalonamento e balanceamento de carga mais acertadas. Por este motivo, a implementação do modelo de atores baseada em eventos tem sido a mais comum entre os ambiente de execução mais difundidos.

Mas mesmo no modelo baseado em eventos, há várias maneiras pela qual o pool de *thread*s pode ser implementado. A maneira mais simples consiste em uma fila de execução única que mantém todos os atores prontos a serem executados. Tão logo um *thread* esteja disponível, ele vai pegando os atores desta fila. Uma outra maneira consiste na utilização de uma fila de atores independente para cada um dos *thread*s. Essa maneira alternativa tem a vantagem de que não há contenção na fila dos atores, já que cada *thread* possui a sua. Isso permite que o sistema escale e seja capaz de trabalhar

com dezenas de *thread*s. No entanto, essa maneira de trabalhar faz com que haja um desbalanceamento de carga entre as filas, o que não acontecia com a fila unificada. Assim, é preciso que um sistema de balanceamento de carga seja empregado. Os ambientes de execução normalmente possuem dois tipos de balanceamento de carga, um baseado em roubo de trabalho e outro em balanceamento periódico. No primeiro caso, *thread*s com a fila vazia passam a roubar atores dos *thread*s com excesso de trabalho, enquanto no segundo uma rotina periódica tenta fazer com que o tamanho das filas de cada um dos *thread*s esteja equilibrada, ou seja, com aproximadamente o mesmo número de atores cada um. O objetivo de tal rotina de balanceamento de carga é assegurar também que cada um dos atores terá um tempo equivalente de uso do processador, pois já que atores que estivessem numa fila reduzida teriam proporcionalmente mais tempo de processador do que atores em uma fila com muitos atores. Há no entanto uma outra vantagem que o uso de múltiplas filas oferece e que não é possível no caso de uma fila única: *soft-affinity*. *Soft-affinity* é a propriedade de um escalonador que diz que um ator não vai ser migrado de processador a não ser que haja a necessidade. Com uma fila única, um mesmo ator pode ser escolhido para a execução a cada momento por um *thread* diferente o que não ocorre com múltiplas filas. Isso traz diversas vantagens do ponto de vista de execução sendo que a mais importante delas é o aproveitamento das caches dos processadores assim como a preservação da proximidade dos dados no caso de uma máquina NUMA.

No entanto, nenhum dos ambientes de execução disponíveis no mercado hoje levam em consideração a hierarquia de memória da máquina para as tomadas de decisão de criação de atores, balanceamento de carga ou roubo de trabalho. Isso pode causar uma diminuição considerável no desempenho das aplicações.

Objetivos

A nossa proposta envolve a avaliação de aplicações baseadas no modelo de atores para, através da utilização de suas características, poder tomar decisões mais apropriadas em máquinas NUMA. algumas dessas melhorias já foram avaliadas e nós pudemos constatar que o seu uso pode trazer melhoras de até 150% no tempo de execução total e ao mesmo tempo limitar a perda de desempenho a 9% no pior caso [FGM13]. Essas melhorias serão aplicadas em duas categorias: colocação inicial dos atores e balanceamento de carga e roubo de trabalho hierárquico.

A colocação inicial dos atores pode ser feita de diversas maneiras depen-

dendo do seu comportamento inicial. Existem alguns atores que são muito mais demandados pelo sistema, por serem responsáveis por atividades centrais das aplicações. Esses atores são normalmente envolvidos na maioria das trocas de mensagens do sistema como estão entre os maiores criadores de atores da aplicação. Esses atores são denominados *hub*s. Desta maneira faz sentido espalhar os *hub*s entre os diferentes processadores e eventualmente nós da máquina NUMA em questão. Por outro lado, os atores criados por um mesmo *hub* tem a tendência de se comunicar entre si e com o seu *hub* criador. Logo, esses atores, chamados de conjunto de afinidades do *hub*, devem ser colocados mais próximos dos seus *hub*s assim aprimorando a comunicação entre eles. Nós propomos o uso de duas estratégias distintas para cara tipo de atores, uma para os *hub*s e outra para os demais. Assim, os *hub*s são espalhados pela máquina e os demais são colocados próximos de seus grupos de afinidade.

Durante a execução da aplicação, desequilíbrios vão acabar ocorrendo. Filas de atores de cada um dos *thread*s vão apresentar tamanhos diferentes e o ambiente de execução precisará equilibrar a carga, ainda que a colocação inicial tenha sido feita de maneira a minimizá-los. Esse é o motivo pelo qual o ambiente de execução precisa de um balanceador de carga periódico. Além disto, durante a execução da aplicação se uma fila ficar sem atores, o seu *thread* empregará um algoritmo de roubo de trabalho até que a próxima execução do balanceador de carga possa corrigir o problema. Em ambos os casos, os atores serão migrados de um *thread* a outro (e portanto de um processador a outro), no entanto em uma arquitetura NUMA a maneira pela qual os atores serão migrados influencia o desempenho da aplicação. Nós propomos que a migração de atores seja feita de uma maneira que minimize a distância entre os atores e os seus grupos de afinidade, em seguida migrar atores entre *thread*s executando no mesmo nó NUMA e só então considerar a migração de atores entre os nós NUMA.

5.4 Justificativa

Há alguns anos, a realidade do mercado de sistemas computacionais deixa claro que as novas demandas dos usuários são por aplicações que necessitam de soluções que incluam paralelismo e/ou distribuição conforme sua natureza. Além disso, a pesquisa na área ainda está aberta em diferentes tópicos [ABC+06]. Para aplicações que apresentam altas demandas de processamento, soluções baseadas em multiprocessadores, aglomerados de computadores e grades computacionais são propostas. Para outra classe de aplicações, como as que requerem a mobilidade de usuários e/ou equipa-

mentos, apresenta-se soluções de nuvens computacionais. Não raramente observamos que estas diferentes plataformas tem sido combinadas para oferecer uma solução única. O estado da arte na área de processamento paralelo e distribuído apresenta suporte ao paralelismo em múltiplos níveis, havendo distribuição de processos com granularidade grossa entre nós de uma arquitetura distribuída e *threads* sobre recursos de processamento internos aos nós que respondem de forma mais satisfatória a exploração de um paralelismo mais fino [asu13].

No entanto, mais recentemente, inseriu-se neste contexto a possibilidade de incluir hardware configurável, como FPGAs, que permitem construir um hardware especializado na execução de uma determinada tarefa. Encontramos diversos trabalhos, como [KD09, CMHM10, MJK12, THL09, CLS+08, LKC+10] que apontam esta tendência de uso combinado de diferentes tecnologias. Entre as questões que se colocam encontram-se as que ponderam sobre o uso eficiente dos recursos disponíveis e as que tratam do nível de abstração oferecido aos programadores visando explorar estes recursos de forma confiável e com produtividade.

Nesta perspectiva, a proposta do presente projeto se apoia na necessidade de se compreender e aperfeiçoar o uso das diferentes plataformas de execução disponíveis, tendo como horizonte sistemas computacionais heterogêneos dando suporte à diversidade das demandas de uma nova geração de aplicações. A principal **justificativa** deste projeto está em estabelecer no país um grupo de pesquisadores sensíveis a estas questões, na medida em que busca fortalecer os diferentes eixos de ação por meio do trabalho colaborativo as pesquisas que encontram-se em curso, e ao mesmo tempo em que oferece a computação heterogênea como um horizonte de trabalho comum. Para tanto, alguns desafios necessitam ser vencidos, sendo necessário investimento nos seguintes tópicos:

- Explorar o potencial de recursos de hardware configurável, em relação ao custo e a capacidade de implementar hardware especializado na execução de algoritmos específicos executados de forma intensiva;
- Obter ferramentas que ofereçam níveis de abstração suficientes para que os programadores possam explorar de forma efetiva e eficiente os recursos de hardware disponíveis;
- Tratar questões relacionadas com a distribuição da carga computacional de forma a realizar o aproveitamento dos recursos disponíveis com menor grau de desperdício de investimento.

5.5 Objetivo Geral

Os Programas de Pós-Graduação envolvidos na presente proposta possuem diversos tópicos de pesquisa relacionados mas que, dada às diferentes vocações observadas, diferentes aspectos e temas são desenvolvidos em cada um. O presente projeto visa potencializar os resultados de pesquisa dos programas envolvidos pelo estimulo às ações de intercâmbio de seus membros, docentes e discentes. Para tanto, é proposta a criação de um laboratório de pesquisa, fisicamente distribuído entre as instituições, para estimular a execução de atividades de forma colaborativa. O **Objetivo Geral** deste projeto é, portanto, estabelecer e consolidar o LEAPaD, Laboratório de Estudos Avançados em Sistemas Paralelos e Distribuídos, como referência a toda comunidade dos programas envolvidos nos tópicos de pesquisa relacionadas. O LEAPaD tera como vocação desenvolver pesquisa em áreas associadas à computação em plataformas heterogêneas Como consequência espera-se que o trabalho cooperativo promova a troca de experiências, resultando em um crescimento qualitativo dos resultados de pesquisa alcançados em cada grupo pela agregação dos conhecimentos e vocações desenvolvidos em cada programa parceiro.

5.6 Objetivos Específicos

Os programas envolvidos neste projeto possuem diferentes históricos. Enquanto o programa da UFPel é o mais jovem, tendo sido criado em 2011, o programa da USP é o mais antigo, tendo sido criado em 1987, sendo que dois anos após, em 1989, o programa da UnB foi instituído. Os programas da USP e da UnB possuem doutorado, sendo que a USP possui curso neste nível desde a instalação do programa e o credenciamento para doutorado na UnB foi concedido em 2010. Na última avaliação trianual, divulgada pela CAPES em dezembro de 2013, o único programa envolvido neste projeto que teve incremento no seu conceito foi o da USP, tendo agora o Conceito 6. Os demais permaneceram com Conceito 3 e 4, UFPel e UnB, respectivamente. Assim, um dos **Objetivos Específicos** deste projeto é buscar a consolidação destes programas visando:

- Consolidar o Conceito 6 obtido pelo PPGCC da USP, reforçando a produção científica em direção ao conceito máximo;
- Incrementar a produção científica e consolidar linhas temáticas de pesquisa visando obtenção de Conceito 5 ao PPGINFO da UnB;

- Alavancar a produção científica e promover o amadurecimento dos pesquisadores do PPGC da UFPel para permitir obtenção de Conceito 4 e a possibilidade de criação de um curso de doutorado no programa.

Outro Objetivo Específico é fomentar, por meio do LEAPaD, o desenvolvimento de projetos de pesquisa em colaboração, viabilizando troca de experiências pela mobilidade de pesquisadores das instituições parceiras.

5.7 Metodologia

Para as pesquisas teóricas, faremos revisões sistemáticas dos trabalhos relacionados de forma a identificar claramente os problemas de pesquisa a serem enfrentados. Para garantir a interação entre os participantes do projeto, resultados parciais da pesquisa serão disponibilizados regularmente no sítio web do projeto.

Neste sítio serão organizadas as informações públicas do projeto, como atividades em execução, oportunidades de trabalho e resultados alcançados. Relatórios técnicos produzidos também terão espaço nesta página bem como uma seção *Wiki* para compartilhamento de dúvidas e registro dos avanços obtidos entre os membros do grupo.

Com relação ao desenvolvimento de software: a medida que o software for desenvolvido, realizaremos experimentos a fim de validar a sua eficácia e analisar a sua eficiência. Esta última será avaliada por meio de medições pontuais (microbenchmarks) e por meio de medições globais do desempenho do sistema a fim de analisar sua escalabilidade, tolerância a falhas e a flutuações na disponibilidade, latência e largura de banda da rede.

A equipe conta com acesso a duas grades para realizar experimentos. Uma deles, a GRID-RS (gridrs.lad.pucrs.br), mantida por uma federação de instituições no estado do Rio Grande do Sul, na qual a UFPel oferece um conjunto de recursos. A segunda grade é a Grid 5000 (www.grid5000.fr). Os coordenadores da instituição proponente e das instituições associadas possuem um histórico de colaborações com instituições francesas que participam desta grade. Estas parcerias serão exploradas para podermos realizar experimentos em ambientes controlados de grande extensão. A equipe de trabalho foi cuidadosamente escolhida de forma a incluir pesquisadores com experiência prévia em cada um dos aspectos fundamentais e tópicos de pesquisa descritos nesta proposta. A nossa estratégia será investigar todos estes tópicos e incluir os resultados parciais das pesquisas realizadas pelos vários integrantes do grupo em um único corpo integrado de software mantido em um repositório de acesso público na Internet.

Como forma de potencializar o trabalho conjunto, serão realizados três seminários, do tipo workshop, reunindo as equipes. O primeiro workshop será realizado nos primeiros seis meses de desenvolvimento do projeto. O segundo encontro entre o 24^o e o 30^o mês. O workshop de encerramento se dará nos últimos seis meses de trabalho. Estes encontros terão como motivação especial, respectivamente, (a) motivar a execução de atividades conjuntas, oferecendo espaço para discussão e compartilhamento de experiências; (b) avaliar o desenvolvimento dos trabalhos e balizar os esforços em direção aos objetivos e metas obtidos; e, (c) avaliar os resultados obtidos.

Outra forma que será buscada para promover a interação dos grupos nas diferentes instituições é oportunizar a co-orientação de alunos de mestrado e doutorado. Consideramos que a co-orientação de dissertações e teses consistem em uma das formas mais efetivas de compartilhar conhecimento, equalizando as diferenças de saberes das diferentes instituições e ampliando os horizontes e a vivência dos estudantes envolvidos. Espera-se também que novos doutores, formados na UnB ou na USP durante a execução do projeto, possam agregar experiência em um período de pós-doutoramento nas outras instituições parceiras para promover ainda mais o fortalecimento dos vínculos entre os grupos.

Iremos também utilizar princípios dos chamados Métodos Ágeis de Desenvolvimento de Software [Coc02] que prevêem a construção de software de forma incremental, iterativa e adaptativa. Além do repositório com o código compartilhado, adotaremos a prática de testes automatizados a fim de garantir a integridade do código em todos os momentos, à medida em que ele cresce. Sempre que possível, o desenvolvimento incremental do software se baseará em técnicas de refatoração [Fow99], mas, se necessário, partes do código serão descartadas e re-escritas.

5.8 Resultados Esperados

O desenvolvimento deste projeto resultará nos próximos quatro anos em (1) formação de recursos humanos nos níveis de graduação, mestrado, doutorado e pós-doutorado, (2) desenvolvimento de modelos teóricos e protocolos para processamento de alto desempenho, (3) implementação de software funcional distribuído com uma licença livre, e (4) publicações de artigos em congressos e revistas científicas nacionais e internacionais.

Os produtos científico-tecnológicos esperados são:

- Algoritmos para alocação de máquinas virtuais e em nuvens com o objetivo de reduzir o consumo de energia sem aumentar significativa-

mente a taxa de bloqueio de requisições de novas alocações;

- Algoritmo baseado em CGM que devolva como saída a possibilidade ou não de atender um dado objetivo relacionado com consumo de energia (por exemplo, reduzir o custo com energia em x%);
- elaborar uma arquitetura para antecipar a alocação de recursos em nuvem usando o custo assintótico de aplicações BSP;
- Adaptar o modelo BSP para ambientes heterogêneos compostos por diversas GPUs e arquiteturas massivamente paralelas;
- Criar e/ou extender ferramentas para paralelização automática de código em ambientes heterogêneos;
- Propor ferramentas e técnicas para a execução eficiente de máquinas virtuais para linguagens de atores em ambientes NUMA;
- Propor e avaliar versões paralelas de aplicações de bioinformática a serem executadas de maneira eficiente em plataformas de alto desempenho compostas por FPGAs e/ou aceleradores manycore;
- Consolidação do ambiente D-GM e seus componentes de software (VPE-qGM e VirD-GM) para suporte às simulações paralelas e/ou distribuídas via software da computação quântica. Consolidação de métodos específicos para descrição de circuitos quânticos através da linguagem VHDL, incluindo a correspondente execução em dispositivos lógicos reconfiguráveis como FPGAs;
- Obter uma arquitetura de hardware dedicada para acelerar o processamento de ataques criptográficos.

Em termos quantitativos esperamos obter os seguintes resultados nos próximos 4 anos:

- 2 pós doutorados;
- 7 doutorados;
- 12 mestrados;
- 10 projetos de IC;
- 4 artigos em periódicos de primeira linha;

- 12 artigos em conferências nacionais e internacionais de primeira linha;
- 12 artigos em workshops nacionais e internacionais.

5.9 Estratégias de Disseminação dos Resultados da Pesquisa

Para disseminar os resultados do projeto, além das publicações já listadas acima, também contaremos com a publicação de relatórios técnicos. Entre as atribuições dos bolsistas e pesquisadores ligados ao projeto, estará também a atualização de um sítio web do projeto, que será possivelmente hospedado na plataforma STOA atualmente disponível na USP.

5.10 Estratégias de Acompanhamento dos Egressos

Para o acompanhamento dos egressos, também vamos usar um sítio web, onde para cada egresso será indicado qual o seu papel no projeto, assim como o seu vínculo atual. Para cada participante do projeto teremos um responsável que deverá atualizar os dados em conjunto com o egresso.

5.11 Estratégias de Seleção da Equipe - Bolsistas e Colaboradores

Inicialmente, para a seleção dos pesquisadores, procuramos encontrar professores com áreas de interesse comuns nas instituições participantes, de forma a permitir uma cooperação fluída na equipe. Boa parte dos pesquisadores envolvidos no projeto já possui alunos de iniciação científica, mestrado e/ou doutorado em temas ligados ao projeto. Logo, a equipe inicial foi formada de forma natural.

Para a seleção de novos bolsistas, faremos chamadas abertas dentro de cada instituição e serão selecionados novos alunos pela de análise de currículo e entrevistas.

5.12 Disponibilidade de Infraestrutura e de Apoio-Técnico para o desenvolvimento

O IME-USP dispõe de uma rede local de alto desempenho, com 700 microcomputadores e vários servidores com alto poder de processamento e capacidade de armazenamento, para seus alunos, docentes e funcionários. Essa rede está conectada ao backbone do campus da Cidade Universitária, o qual está conectado à Internet por linhas de alta velocidade, e provê várias formas

de acesso remoto, como VPN. O IME-USP agrega laboratórios compartilhados com mais de 200 microcomputadores, usados por alunos de graduação e pós-graduação. Além disso, vários grupos de pesquisa mantêm laboratórios para o desenvolvimento de projetos de pesquisa com a participação de seus alunos. Além disso, temos todo um corpo de funcionários altamente qualificados para apoio à pesquisa.

Membros do PPGInf contam com os diversos laboratórios do Departamento de Ciência da Computação e do Instituto de Ciências Exatas. Durante o ano de 2011, houve uma reestruturação da distribuição de espaço físico para atender grupos emergentes do Departamento de Computação, com forte participação de docentes e discentes do PPGInf. Esses novos espaços foram alocados no novo prédio para o qual se mudou o Departamento de Ciência da Computação em novembro de 2012. Três novos laboratórios foram criados, totalizando no momento 10 laboratórios: LAICO, LARA, LISA, TransLAB, COMNET, LaFORCE e LaBid, LES, LabPOS, LabIE.

Ao todo, a infraestrutura do Departamento de Ciência da Computação da UnB, conta com mais de 300 microcomputadores e vários servidores com alto poder computacional para alunos, docentes e funcionários. Em particular, para computação de alto desempenho, contamos com um cluster de 12 GPUs (5 GTX 680, 5 GTX 580, 1 GTX 560 Ti e 1 GTX 285), uma plataforma de Computação Reconfigurável (XD 2000i), além do Instituto de Exatas ao qual o CIC é vinculado constar com um cluster Sun com 80 processadores: 40 nodos Opteron Dual 2.4GHz, 2GB RAM, interconectados por Gigabit Ethernet.

O PPGC da UFPel está situado no Campus Porto, com acesso ao backbone nacional da RNP, estando sendo consolidada a conexão via rede COMEP. Toda área física deste Programa está coberta por rede sem fio e os laboratórios e salas de aula possuem acesso à rede cabeada. Nos laboratórios da Computação, atendendo tanto a demanda de pesquisa como a demanda de ensino, na graduação e pós-graduação, são disponibilizados cerca de 150 computadores. A manutenção destes equipamentos, bem como instalação e suporte, é mantida por um funcionário e quatro estagiários. O apoio administrativo é realizado por três funcionários em tempo integral.

Para pesquisa o PPGC ainda conta com diferentes plataformas de trabalho.

- GPUs para processamento matricial intensivo de diferentes capacidades, dentre estas uma Tesla Nvidia k20
- Placas de prototipação de hardware

- Servidores de processamento multiprocessados (arquiteturas UMA e NUMA com 8, 16 e 64 cores)
- Um dos clusters possui cinco nós de 6-cores
- Um cluster com 12 nós ligado à GRADE-RS, uma grade na qual outras três instituições gaúchas (PUC/RS, UFSM e UFRGS) compartilham capacidade de processamento.
- Diversos servidores de aplicação para uso em pesquisa, com recursos Web, Wiki, Git.

5.13 Formação/Aperfeiçoamento de Docentes e/ou Pesquisadores

Nestes projeto, que envolve tantos pesquisadores de áreas correlatas, tantos os alunos como os próprios pesquisadores terão a possibilidade de interagir facilmente com outras pessoas. Isto por si só já deve proporcionar grandes oportunidades de aperfeiçoamento.

Durante estágios de pós doutorado os docentes também terão oportunidades claras de se aperfeiçoar, sendo que os estudantes também terão oportunidades semelhantes pelas atividades ligadas à co-orientação de um pesquisador de uma instituição parceira. O Projeto envolve bolsistas de iniciação científica, mestrandos e profissionais de nível téccnico, contratados para realização tarefas específicas, contribuindo assim para formação de recursos humanos na área tecnológica. Espera-se que a visão ampliada da área de pesquisa, pela vivência de suas atividades no LEAPaD, estes estudantes amadureçam a forma de pensar e executar a pesquisa.

5.14 Melhoria dos Programas de Pós-Graduação Participantes

Com relação à escolha das equipes, utilizamos os seguintes critérios:

- Buscamos instituições com diferentes avaliações segundo os critérios da CAPES, logo temos instituições com notas 3, 4 e 6;
- Além disto, os problemas de pesquisa a serem abordados estão todos relacionados com a computação paralela e distribuída, logo, os participantes terão a oportunidade de trabalhar em suas áreas de pesquisa e em áreas próximas;
- Boa parte dos pesquisadores envolvidos no projeto já se conhece.

Considerando os critérios acima, o nosso objetivo foi de criar uma sinergia que permita uma colaboração fluída entre os diversos pesquisadores. Esta colaboração entre os pesquisadores, deve gerar pesquisa original, que servirá a melhorar os programas participantes. Também estamos prevendo diversos visitas de pesquisa entre os estudantes envolvidos, permitindo uma colaboração ainda maior entre as instituições.

5.15 Publicações Conjuntas

Até o momento os coordenadores não possuem um número significativo de publicações conjuntas, mas eles veem trabalhando juntos, desde o doutoramento, realizado no mesmo instituto. Inclusive, já houve a participação dos coordenadores como membros de bancas de outros coordenadores.

Além disto, observando o currículo dos coordenadores é fácil notar que cada um está bem inserido no seu grupo de pesquisa. Cada um com várias publicações conjuntas. Logo, esperasse que esta sinergia possa ser transportada para toda a equipe.

Daniel Macêdo Batista do IME-USP trabalha em conjunto com Genaína Nunes Rodrigues e Alba Cristina Magalhaes Alves de Melo desde 2012. Neste ano eles trabalharam no artigo [GCB+13], que foi aceito para publicação no periódico *Journal of Grid Computing*. Além disso Daniel Batista e Genaína Nunes respectivamente orientam e co-orientam o aluno Felipe Pontes Guimarães, doutorando no IME-USP e que também será integrante do LEAPaD.

6 Cronograma de Atividades

Primeiro ano

- Algoritmos para modificação de DAGs incorporando as instanciações de máquinas virtuais com o objetivo de reduzir o consumo de energia
- Algoritmo para alocação de máquinas virtuais baseado na abordagem do problema da mochila multidimensional com múltiplas escolhas (MMKP)
- Fazer uma revisão sistemática de todo o trabalho já feito sobre o modelo BSP;
- Testar diferentes sistemas de GPU com algoritmos e aplicações que apresentem o melhor desempenho na paralelização com o modelo BSP;

- Estudo aprofundado de ferramentas de paralelização automática de código;
- Continuar os estudos sobre as modificações de máquina virtual Erlang em ambientes NUMA de memória compartilhada;
- Estudo das Aplicações de Bioinformática, determinando o seu potencial de paralelização. Escolha das Aplicações;
- Prototipação da biblioteca qFuzzAnalyzer como extensão da qGMAnalyzer - Inovação na Modelagem Lógica de Sistemas Fuzzy via Computação Quântica;
- revisão de algoritmos de criptoanálise visando explorar seu paralelismo de processamento. Revisão do estado da arte em arquiteturas paralelas para processamento massivo em FPGA. Realizar o treinamento com o fluxo de projeto de hardware. Realizar o treinamento de alunos com as ferramentas de desenvolvimento, validação e simulações de sistemas em hardware;

Segundo ano

- Algoritmo de escalonamento para nuvens que considere a utilização do TCP "verde" para suspender e reiniciar máquinas virtuais
- Algoritmo para alocação de máquinas virtuais baseado em abordagens de computação evolutiva
- Análise do desempenho de algoritmos de escalonamento para nuvens em cenários onde hajam máquinas físicas pré-reservadas para a alocação de máquinas virtuais com objetivos específicos: uso intensivo de CPU, uso intensivo de memória, uso intensivo de rede, etc...
- Avaliar a eficiência da predição de aplicações BSP na nuvem;
- Estudar e caracterizar o comportamento de entidades computacionais sobre GPUs massivamente paralelas;
- Investigação de técnicas de como adaptar ferramentas de paralelização automática para ambientes com várias GPU e ambientes com uso simultâneo de GPU e CPUs (possivelmente multi-core);
- Estudar as possibilidades de modificações da máquina virtual para ambientes sem memória compartilhada;

- Projeto de Aplicações Paralelas de bioinformática para FPGAs e/ou Aceleradores Manycore. Escrita de modelo de previsão de desempenho para estas aplicações;
- Desenvolvimento de estratégias de gerenciamento de processos computacionais para suporte às aplicações móveis/distribuídas no Projeto FL+QC;
- Realizar o treinamento de alunos para uso de dispositivos FPGAs usados para prototipação de sistemas digitais de diferentes aplicações. Avaliar e definir os requisitos necessários para o projeto do hardware dedicado. Propor arquiteturas para processamento massivamente paralelo dedicadas a aplicação, e validar arquiteturas propostas.

Terceiro ano

- Algoritmo de escalonamento para nuvens que considere a existência de enlaces com capacidade adaptativa interligando *data centers*;
- CGM que leve a mudanças nas escolhas de serviços web com o objetivo de reduzir e evitar aumentos desnecessários no consumo de energia;
- Algoritmo que devolva como saída a possibilidade ou não de atender um dado objetivo relacionado com consumo de energia (por exemplo, reduzir o custo com energia em x%);
- Desenvolver uma arquitetura que permita a predição de desempenho de aplicações BSP em nuvens;
- Criar um modelo algorítmico o modelo BSP para o estado de arte atual de GPUs e Supercomputadores;
- Implementação incial de técnicas de paralelização automática em ambientes heterogêneos;
- Estudo de técnicas de caracterização de *hubs* em programas, inicialmente, usando a linguagem Erlang;
- Implementacao e testes de aplicações de bioinformática em FPGAs e aceleradores. Verificação da acurácia do modelo proposto no primeiro ano para aplicações de bioinformática;
- Prototipação de estratégias de gerenciamento de processos computacionais para suporte às aplicações móveis/distribuídas no Projeto FL+QC sobre sistemas heterogêneos com FPGAs e GPUs;

- Realizar a integração hardware-software da arquitetura proposta e o sistema computacional o qual será adaptado. Validar a integração do hardware dedicado no sistema computacional. Avaliar o desempenho de execução comparado a arquiteturas de propósito geral, avaliar, também, diferentes dispositivos FPGA a fim de verificar o impacto da tecnologia empregada em cada FPGA sobre o desempenho da computação.

Quarto ano

- Modelo matemático que permita prever a confiabilidade e a disponibilidade de nuvens que utilizem os algoritmos de escalonamento propostos nos anos anteriores do projeto;
- Avaliar a arquitetura de predição de desempenho para aplicações BSP executadas em nuvens computacionais;
- Propor opções para balanceamento de cargas, tolerância a falhas ou Check-pointing de processos em ambientes heterogêneos;
- Criação de um protótipo para paralelização automática em ambientes heterogêneos;
- Mudanças na uma máquina virtual Erlang para considerar tanto o ambiente NUMA, como para encontrar os *hubs* de forma automática nos programas;
- Coleta e Análise de Resultados Experimentais, utilizando sequências genômicas reais;
- Integração dos protótipos em uma biblioteca para simulação eficiente do Projeto FL+QC;
- Avaliar o desempenho da arquitetura e propor possíveis ajustes nas arquiteturas. Propor possíveis modificações no algoritmo de criptoanálise visando evitar gargalos remanescentes no sistema proposto.

7 Equipes do Projeto - Proponente, Associadas I e II

7.1 Proponente

professores:

- Alfredo Goldman - Coordenador;
- Daniel Batista;
- Marco Aurélio Gerosa;
- Marco Dimas Gubitoso;
- Fabio Kon.

pos-doc Daniel de Angelis Cordeiro;

Doutorandos:

- Alessandro Kraemer;
- Marcos Amaris Gonzalez;
- Rogério Aparecido Gonçalves;
- Emílio Francesquini;
- Lucas Virgili Leiro;
- Rafael Roque de Souza;
- Felipe Pontes Guimarães;
- Gustavo Ansaldi Oliva;
- Paulo Bittencourt Moura;
- Ivanilton Polato;
- Carlos Eduardo Moreira Santos.

Mestrandos:

- Albert Philippe de la Fuente Vigliotti;
- Leonardo Alexandre Ferreira Leite;
- Guilherme M. Nogueira.

7.2 Associada I

professores:

- Alba Melo Cristina Magalhães - Coordenadora;
- Ricardo Jacobi;
- Maria Emilia Walter;
- Genaína Nunes Rodrigues;

- George Luiz Medeiros Teodoro;
- Maristela Terto Holanda;
- Vander Ramos Alves;
- Eduardo Adilio Pelinson Alchieri;
- Aletéia Patrícia Favacho de Araújo.

doutorandos:

- Daniel Sundfeld Lima;
- Edans Flávius de Oliveira Sandes;
- Emerson de Araújo Macedo;
- Ricardo Chaves;
- Idarlan Martins Machado;
- André Luiz Peron Martins Lanna;
- Rodrigo Pinheiro de Almeida;
- Wander Queiroz.

mestrandos:

- Danilo Filgueira Mendonça;
- José Carlos Guimarães Jr;
- Marco Antônio Caldas de Figueiredo Jr.

Graduandos:

- Felipe Rodopoulos.

7.3 Associada II

professores:

- Gerson Geraldo Homrich Cavalheiro - Coordenador;
- Paulo R. Ferreira Jr;
- Luciano Volcan Agostini;
- Renata Hax Sander Reiser;
- Maurício Lima Pilla;
- Adenauer Correa Yamin;

- André Rauber du Bois;
- Bruno Zatt;
- Marcelo Schiavon Porto;
- Rafael Iankowski Soares;
- Vander Ramos Alves;
- Simone André da Costa Cavalheiro;
- Ana Marilza Pernas Fleischmann;
- Luciana Foss.

Mestrandos:

- Alan Schlindvein de Araújo;
- Luciano Ludwig Loder;
- Rodrigo Costa de Moura;
- Vilnei Marins de Freitas das Neves;
- Rodolfo Migon Favaretto.

Graduandos:

- Anderson Braga de Avila;
- Leandro Bresciane das Neves;
- Bruno Giacobo Pinto;
- Israel Silva Barbará;
- Rodrigo Medeiros Duarte;
- Guilherme Porto Britto Cousin;
- Murilo Figueiredo Schmalfuss.

8 Orçamento

Dada a grande quantidade de pesquisadores e estudantes envolvidos no projeto, estamos solicitando o número máximo de bolsas de iniciação científica, de estágios de pós-doutorado no Brasil e de auxílios moradia. Isto é:

- 36 bolsas de Iniciação científica de até 12 meses;
- 48 mensalidades de para o pós-doutorado, sendo cada uma limitada até 12 meses;

- 96 mensalidades de auxílio moradia, sendo que as de mestrado são limitadas a 6 meses, de doutorado a 12 meses e de iniciação científica a 2 meses.

Além disso, nos comprometemos a respeitar a quantidade anual de bolsas e auxílios moradia. Considerando o número de pesquisadores e estudantes envolvidos no projeto, o número de mensalidades solicitado pode ser considerado relativamente pequeno.

As bolsas de iniciação científica serão usadas prioritariamente na UnB e UFPel.

Com relação a custeio, solicitamos 90 mil reais, por ano, pelos quatro anos do projeto, o que perfaz 360 mil reais.

A seguir citamos alguns veículos onde almejamos publicar:

- IEEE Global Communications Conference (Globecom);
- ACM Symposium On Applied Computing (SAC);
- IEEE International Conference on Communications (ICC);
- IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID);
- SBC Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC);
- Simpósio Brasileiro de Arquitetura de Computadores (SBAC-PAD);
- International Conference on Parallel Processing (ICPP);
- IEEE International Parallel & Distributed Processing Symposium (IPDPS);
- EuroPar;
- International Meeting on High Performance Computing for Computational Science (VecPar);
- International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar);
- IEEE International Symposium on Network Computing and Applications (IEEE NCA);
- IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE);

- IEEE International Conference on Cloud Computing (CGC);
- IEEE International Conference on Cloud Computing Technology and Science (CLOUDCOM);
- Symposium on Integrated Circuits and Systems (SBCCI);
- International Symposium on Circuits and Systems (ICSAS);
- International Conference on Field Programmable Logic and Applications (FPL);
- Design, Automation &Test in Europe (DATE).

Além destas, pode ser interessante a participação de alunos de IC em:

- Escola de Processamento de Alto Desempenho (Rio Grande do Sul) - ERAD-RS;
- Escola de Processamento de Alto Desempenho (São Paulo) - ERAD-SP;
- Escola Regional de Redes de Computadores (ERRC);
- Escola de Microeletrônica (EMICRO);
- Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD);
- Latin American Symposium on High Performance Computing (HPCLatin).

9 Bibliografia

Referências

[AAPZ10] B. Addis, D. Ardagna, B. Panicucci, and Li Zhang. Autonomic management of cloud service centers with availability guarantees. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 220 –227, july 2010.

[ABC+06] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[ABD+09] Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Sen, John Wawrzynek, David Wessel, and Katherine Yelick. A view of the parallel computing landscape. *Communications of the ACM*, 52(10):56–67, October 2009.

[ACE+12] Mehdi Amini, Béatrice Creusillet, Stéphanie Even, Ronan Keryell, Onig Goubier, Serge Guelton, Janice Onanian McMahon, François-Xavier Pasquier, Grégoire Péan, and Pierre Villalon. Par4all: From convex array regions to heterogeneous computing. In *2nd International Workshop on Polyhedral Compilation Techniques, Impact (Jan 2012)*, 2012.

[ADG10] Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering*, 15(4):439–458, 2010.

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[Ama] Amazon. Amazon web services. `http://aws.amazon.com/`. Último acesso em: 25/10/2013.

[AMR+14] A. Ávila, A. Maron, R. Reiser, M. Pilla, and A. Yamin. Gpu-aware distributed quantum simulation. In *Proceedings of SAC 2014*, pages 1–6, 2014. (to be published).

[AMRP12] A. Avila, A. Maron, R. Reiser, and M. Pilla. Extending the VirD-GM environment for the distributed execution of quantum processes. In *Proceedings of the XIII WSCAD-WIC*, pages 1–4, 2012.

[APTZ12] D. Ardagna, B. Panicucci, M. Trubian, and Li Zhang. Energy-aware autonomic resource allocation in multitier virtualized environments. *Services Computing, IEEE Transactions on*, 5(1):2 –19, jan.-march 2012.

[Arm10] Joe Armstrong. Erlang. *Commun. ACM*, 53:68–75, September 2010.

[asu13] A survey on amdahal's law extension in multicore architectures. *New Computer Architectures and their Applications*, 3(3):20–46, 2013.

[BAB12] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012.

[BAS+08] J. Baliga, R.W.A. Ayre, W.V. Sorin, K. Hinton, and RodneyS. Tucker. Energy consumption in access networks. In *Optical Fiber communication/National Fiber Optic Engineers Conference, 2008. OFC/NFOEC 2008. Conference on*, pages 1–3, 2008.

[BB12] Anton Beloglazov and Rajkumar Buyya. OpenStack neat: A framework for dynamic consolidation of virtual machines in OpenStack clouds-A blueprint. Technical report, Technical Report CLOUDS-TR-2012-4, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, 2012.

[BB13] A. Beloglazov and R. Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379, 2013.

[BBLZ10] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. arXiv e-print 1007.0066, July 2010.

[BCdF11] D.M. Batista, C.G. Chaves, and N.L.S. da Fonseca. Embedding software requirements in grid scheduling. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6, 2011.

[BCdMJ10] Azzedine Boukerche, Jan Mendonca Correa, Alba Cristina M.A. de Melo, and Ricardo P. Jacobi. A hardware accelerator for the fast retrieval of dialign biological sequence alignments in linear space. *IEEE Transactions on Computers*, 59(6):808–821, 2010.

[BCRR12] Aruna Prem Bianzino, Claude Chaudet, Dario Rossi, and Jean-Louis Rougier. A survey of green networking research. *IEEE Communications Surveys and Tutorials*, 14(1):3–20, 2012.

[BDSR10] B. Bedregal, G. Dimuro, R. Santiago, and R. Reiser. On interval fuzzy S-implications. *Information Science*, 180(8):1373–1389, 2010.

[Bed07] B.C. Bedregal. A normal form which preserves tautologies and contradictions in a class of fuzzy logics. *J. Algorithms*, 62:135–147, 2007.

[BGLA13] Héctor Blanco, Fernando Guirado, Josep Lluís Lérida, and V. M. Albornoz. Mip model scheduling for multi-clusters. In *Proceedings of the 18th international conference on Parallel processing workshops*, Euro-Par'12, pages 196–206, Berlin, Heidelberg, 2013. Springer-Verlag.

[BH07] L.A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[BHBE10] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. Haloop: efficient iterative data processing on large clusters. *Proc. VLDB Endow.*, 3(1-2):285–296, September 2010.

[Bis04] Rob H. Bisseling. *Parallel Scientific Computation: A Structured Approach using BSP and MPI.* Oxford University Press, USA, 5 2004.

[BP10] Luciano Baresi and Liliana Pasquale. Live goals for adaptive service compositions. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10*, pages 114–123, 2010.

[BW03] B. Butscher and H. Weimer. Simulation eines quantencomputers, 2003.

[Cas06] Harold Castro. Grid computing: promesa de los sistemas distribuidos. *Revista Sistemas ACIS*, (98):45–57, Oct-Dic 2006.

[CGS07] Daniel Cordeiro, Alfredo Goldman, and Dilma Da Silva. Load Balancing on an Interactive Multiplayer Game Server. In *Euro-Par*, pages 184–194, 2007.

[CHO13] CHOReOS Project, 2013. `http://www.choreos.eu/bin/view/Main/`. Acesso em 13 de Dezembro de 2013.

[CJL+08] Ronnie Chaiken, Bob Jenkins, Peraake Larson, Bill Ramsey, Darren Shakib, Simon Weaver, and Jingren Zhou. Scope: easy and efficient parallel processing of massive data sets. *Proc. VLDB Endow.*, 1(2):1265–1276, August 2008.

[CJvdP07] Barbara Chapman, Gabriele Jost, and Ruud van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007.

[CLS+08] Shuai Che, Jie Li, Jeremy W. Sheaffer, Kevin Skadron, and John Lach. Accelerating compute-intensive applications with gpus and fpgas. In *Proceedings of the 2008 Symposium on Application Specific Processors*, SASP '08, pages 101–107, Washington, DC, USA, 2008. IEEE Computer Society.

[CMHM10] Eric S Chung, Peter A Milder, James C Hoe, and Ken Mai. Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and gpgpus? In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 225–236. IEEE Computer Society, 2010.

[CMT+13] Gustavo Callou, Paulo Maciel, Dietmar Tutsch, João Ferreira, Julian Araújo, and Rafael Souza. Estimating sustainability impact of high dependable data centers: a comparative study between Brazilian and US energy mixes. *Computing*, 95(12):1137–1170, 2013.

[Coc02] Alistair Cockburn. *Agile Software Development*. Addison-Wesley Longman, 2002.

[CS05] Chunxi Chen and Bertil Schmidt. An adaptive grid implementation of DNA sequence alignment. *Future Generation Comp. Syst*, 21(7):988–1003, 2005.

[CSS11] Shuai Che, Jeremy W. Sheaffer, and Kevin Skadron. Dymaxion: optimizing memory access patterns for heterogeneous systems. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 13:1–13:11, New York, NY, USA, 2011. ACM.

[CtJ12] Ying Chang-tian and Yu Jiong. Energy-aware genetic algorithms for task scheduling in cloud computing. In *ChinaGrid Annual Conference (ChinaGrid), 2012 Seventh*, pages 43–48, 2012.

[Deh06] Frank Dehne. Guest editor's introduction. *Algorithmica*, 45(3):263–267, July 2006.

[DGM11] Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Adaptive socio-technical systems: a requirements-based approach. *Requirements Engineering*, 18(1):1–24, September 2011.

[DMR10] G. Dhiman, K. Mihic, and T. Rosing. A system for online power prediction in virtualized environments using gaussian mixture models. In *2010 47th ACM/IEEE Design Automation Conference (DAC)*, pages 807–812, 2010.

[dOSdM13] Edans Flavius de O. Sandes and Alba Cristina M.A. de Melo. Retrieving smith-waterman alignments with optimizations for megabase biological sequences using gpu. *IEEE Transactions on Parallel and Distributed Systems*, 24(5):1009–1021, 2013.

[ea95] S. F. Altschul et al. Basic local alignment tool. *Journal of Molecular Biology*, 215(2):410–413, 1995.

[eSKG+10] Francisco José da Silva e Silva, Fabio Kon, Alfredo Goldman, Marcelo Finger, Raphael Y. de Camargo, Fernando Castor Filho, and Fábio M. Costa. Application execution management on the InteGrade opportunistic grid middleware. *J. Parallel Distrib. Comput.*, 70(5):573–583, 2010.

[FBA+03] Luis Ferreira, Victor Berstis, Amnstrong Armstrong, et al. *Introduction to Grid Computing with Globus*. IBM RedBooks, 2003.

[FG+09a] A. Fox, R. Griffith, et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Tech. Rep. UCB/EECS*, 28, 2009.

[FG09b] Emilio Francesquini and Alfredo Goldman. Scheduling on multi-core clusters. In *Workshop on Algorithms and Techniques for Scheduling on Clusters and Grids*, Les Plantiers, 2009. Proc. of ASTEC 2009.

[FGM13] Emilio Francesquini, Alfredo Goldman, and Jean-François Mehaut. A NUMA-Aware Runtime Environment for the Actor Model. In *Proceedings of the 42nd International Conference on Parallel Processing, ICPP 2013*, Lyon, France, oct 2013. Accepted.

[Fos95] Ian Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Addison-Wesley Longman Publishing Co. Inc, Boston, MA, USA, 1995.

[Fos01] I. Foster. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, pages 1–4, London, UK, 2001. Springer-Verlag.

[Fow99] Martin Fowler. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.

[FWB07] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM.

[FZRL08] I. Foster, Yong Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1 –10, nov. 2008.

[G̈97] Silvia Götz. Algorithms in CGM, BSP and BSP* model: A survey. Technical report, Carleton University, 1997.

[Gar13] Aaron Garrett. Inspyred inspired intelligence initiative, 2013. Accessed: 2013-11-13.

[GCB+13] Felipe Pontes Guimaraes, Pedro Célestin, Daniel Macêdo Batista, Genaína Nunes Rodrigues, and Alba Cristina Magalhaes Alves de Melo. A Framework for Adaptive Fault-Tolerant Execution of Workflows in the Grid: Empirical and Theoretical Analysis. *Journal of Grid Computing*, 2013. Aceito para publicação. Já disponível em `http://dx.doi.org/10.1007/s10723-013-9281-4`.

[GFC+12] Zhenyu Guo, Xuepeng Fan, Rishan Chen, Hucheng Zhou, Jiaxing Zhang, Chang Liu, Lidong Zhou, and Wei Lin. Nemo: Program analysis and optimization for distributed data-parallel computation. Technical Report MSR-TR-2012-53, Microsoft Research, May 2012.

[GKGF04] Andrei Goldchleger, Fabio Kon, Alfredo Goldman, and Marcelo Finger. InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines. 16:16–449, 2004.

[GKS+] A. Goldchleger, F. Kon, SW Song, A. Goldman, M. Finger, JR Braga Jr, R. Camargo, E. Guerra, V.E.M. Rocha, T.M. Jorge, et al. The InteGrade Project: Status Report. *III Workshop on Grid Computing and Applications III Workshop de Computação em Grid e Aplicações*, page 49.

[GM11] Abhishek Gupta and Dejan Milojicic. Evaluation of hpc applications on cloud. Technical Report HPL-2011-132, Hewlett Packard Development Company, August 2011.

[GQ03] Alfredo Goldman and Carlos Queiroz. A Model for Parallel Job Scheduling on Dynamic Computer Grids. In *Middleware Workshops*, pages 264–266, 2003.

[GRTZ10] E. Gutierrez, S. Romero, M. Trenas, and E. Zapata. Quantum computer simulation using the cuda programming model. *Computer Physics Communications*, pages 283–300, 2010.

[HA09] Tianyi David Han and Tarek S. Abdelrahman. hicuda: a high-level directive-based language for gpu programming. In *GPGPU-2: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pages 52–61, New York, NY, USA, 2009. ACM.

[HA11] Tianyi David Han and Tarek S. Abdelrahman. hicuda: High-level gpgpu programming. *IEEE Transactions on Parallel and Distributed Systems*, 22:78–90, 2011.

[hP12] THE hiCUDA PROJECT. The hicuda project site @ONLINE, 10 2012.

[HsCtH13] Lu Huang, Hai shan Chen, and Ting ting Hu. Survey on resource allocation policy and job scheduling algorithms of cloud computing. *Journal of Software*, 8(2), 2013.

[HSL+12] Qingjia Huang, Sen Su, Jian Li, Peng Xu, Kai Shuang, and Xiao Huang. Enhanced energy-efficient scheduling for parallel applications in cloud. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, CCGRID '12, pages 781–786, Washington, DC, USA, 2012. IEEE Computer Society.

[HZG08] Qiming Hou, Kun Zhou, and Baining Guo. BSGP: bulk-synchronous GPU programming. *ACM Trans. Graph*, page 12, 2008.

[IC98] L. Irish and K. Christensen. A 'green tcp/ip' to reduce electricity consumed by computers. *Proceedings of IEEE Southeastcon*, pages 302–305, apr 1998.

[Int13] Intel. O coprocessador intel(r) xeon phi(tm) 5110p @ONLINE, January 2013.

[Joh11] John Wilkes. Second format of cluster-usage traces, 2011. `http://code.google.com/p/googleclusterdata/wiki/TraceVersion2`. Último acesso: 13/11/2013.

[KCS12] Nakku Kim, Jungwook Cho, and Euiseong Seo. Energy-credit scheduler: An energy-aware virtual machine scheduler for cloud systems. *Future Generation Computer Systems*, 2012.

[KD09] Nachiket Kapre and André DeHon. Accelerating spice model-evaluation using fpgas. In *Field Programmable Custom Computing Machines, 2009. FCCM'09. 17th IEEE Symposium on*, pages 37–44. IEEE, 2009.

[Khr13] Khronos. Opencl - the open standard for parallel programming of heterogeneous systems @ONLINE, January 2013.

[KM05] Mounir Kechid and Jean Frederic Myoupo. Towards a more realistic bsp cost model. In *Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, HPCASIA '05, pages 3–, Washington, DC, USA, 2005. IEEE Computer Society.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, UK, 1996. Springer-Verlag.

[Koo11] Jonathan Koomey. Growth in data center electricity use 2005 to 2010. *Oakland, CA: Analytics Press. August*, 1:2010, 2011.

[Kro13] Dmitrey L. Kroshko. OpenOpt website, 2013. Accessed: 2013-11-13.

[LDRC12] Ning Liu, Ziqian Dong, and R. Rojas-Cessa. Task and server assignment for reduction of energy consumption in datacenters. In *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*, pages 171–174, 2012.

[LE10] Seyong Lee and Rudolf Eigenmann. Openmpc: Extended openmp programming and tuning for gpus. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society.

[LKC+10] Victor W Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, et al. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. 38(3):451–460, 2010.

[LM12] Alessandro F. Leite and Alba C. M. A. Melo. Executing a biological sequence comparison application on a federated cloud environment. In *Int. Conf. on High Performance Computing (HiPC)*, 2012.

[LME09] Seyong Lee, Seung-Jai Min, and Rudolf Eigenmann. Openmp to gpgpu: A compiler framework for automatic translation and

optimization. In *PPoPP '09: Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 101–110. ACM, 2009.

[LSM10] Y. Liu, B. Schmidt, and D. Maskell. CUDASW++2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions. *BMC Research Notes*, 3(1):93, 2010.

[MGW09] David Meisner, Brian T. Gold, and Thomas F. Wenisch. Powernap: eliminating server idle power. *SIGPLAN Not.*, 44(3):205–216, March 2009.

[MJK12] Pingfan Meng, Matthew Jacobsen, and Ryan Kastner. Fpga-gpu-cpu heterogenous architecture for real-time cardiac physiological optical mapping. In *Field-Programmable Technology (FPT), 2012 International Conference on*, pages 37–42. IEEE, 2012.

[MLZ13] Dmitry Mikushin, Nikolay Likhogrud, and Eddy Z. Zhang. Kernelgen - the design and implementation of a next generation compiler platform for accelerating numerical models on gpus. *Rutgers Technical Reports*, 2013.

[MOP09] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2009.

[Mou04] D. W. Mount. *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press, 2004.

[MRP13] A. Maron, R. Reiser, and M. Pilla. High-performance quantum computing simulation for the quantum geometric machine model. In *Proceedings of CCGRID 2013*, pages 1–8, 2013.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[NMI08] Jumpei Niwa, Keiji Matsumoto, and Hiroshi Imai. General-purpose parallel simulator for quantum computing. *Lecture Notes in Computer Science*, 2509:230–249, 2008.

[NNH+12] Zoltán Nagy, Csaba Nemes, Antal Hiba, András Kiss, Árpád Csík, and Péter Szolgay. Fpga based acceleration of computational fluid flow simulation on unstructured mesh geometry. In Dirk Koch, Satnam Singh, and Jim Tørresen, editors, *FPL*, pages 128–135. IEEE, 2012.

[Nor91] William D. Nordhaus. To slow or not to slow: The economics of the greenhouse effect. *The Economic Journal*, 101:920–937, 1991.

[ns3] ns-3. `http://www.nsnam.org/`. Último acesso em: 25/10/2013.

[NVI] NVIDIA. From Super Phones to Super Cars. `http://www.nvidia.co.uk/object/who-are-nvidia-uk.html` Visited Oct 15 2012.

[NVI12] NVIDIA. *CUDA C: Programming Guide, Version 4.2.*, April 4 2012.

[OAC+06] Martin Odersky, Philippe Altherr, Vincent Cremet, Iulian Dragos, Gilles Dubochet, Burak Emir, Sean McDirmid, Stéphane Micheloud, Nikolay Mihaylov, Michel Schinz, Lex Spoon, Erik Stenman, and Matthias Zenger. An Overview of the Scala Programming Language (2. edition). Technical report, EPFL, 2006.

[Ope11] OpenACC. Openacc application programming interface. version 1.0. @ONLINE, November 2011.

[Ope12a] OpenACC. Openacc directives for accelerators site @ONLINE, 10 2012.

[Ope12b] Site OpenMP. Openmp site @ONLINE, January 2012.

[Par66] D.F. Parkhill. *The Challenge of the Computer Utility*. Number p. 246 in The Challenge of the Computer Utility. Addison-Wesley Publishing Company, 1966.

[PGK11] V. G. Pinheiro, Alfredo Goldman, and Fabio Kon. Adaptive fault tolerance mechanisms for opportunistic environments: a mobile agent approach. *Concurrency and Computation: Practice and Experience*, 23(17):2078–2091, 2011.

[Pla07] Global Action Plan. An inefficient truth. *Global Action Plan Report*, 2007.

[PS07] Dennis Pamlin and Katalin Szomolányi. Saving the climate @ the speed of light - first roadmap for reduced co2 emissions in the eu and beyond. *World Wildlife Fund and European Telecommunications Network Operatorr' Association*, april 2007.

[RA04] Stjepan Rajko and Srinivas Aluru. Space and time optimal parallel sequence alignments. *IEEE Transactions on Parallel and Distributed Systems*, 15, 2004.

[RA10] R. Reiser and R. Amaral. The quantum states space in the qGM model. In *Proc. of the III Workshop-School on Quantum Computation and Quantum Information*, pages 92–101, Petrópolis/RJ, 2010. LNCC Publisher.

[RAG10] M. RAGHUVANSHI, A.; PERKOWSKI. Fuzzy quantum circuits to model emotional behaviors of humanoid robots. In *Proceedings of EVOLUTIONARY COMPUTATION (CEC)*, 2010.

[RB13] Nikolay Grozev Rajkumar Buyya, Rodrigo N. Calheiros. CloudSim website, 2013. Accessed: 2013-12-04.

[RBB13] R.H.S. Reiser, B. Bedregal, and M. Baczyński. Aggregating fuzzy implications. *Information Science*, 253:126–146, 2013.

[Reu] Thomson Reuters. Journal citation reports (JCR). http://www.ieee.org/publications_standards/publications/journmag/journalcitations.htm Accessed: 11/11/2013.

[RLRFdS12] Ruymán Reyes, Iván López-Rodríguez, Juan J. Fumero, and Francisco de Sande. accull: an openacc implementation with cuda and opencl support. In *Proceedings of the 18th international conference on Parallel Processing*, Euro-Par'12, pages 871–882, Berlin, Heidelberg, 2012. Springer-Verlag.

[RMR+06] K. De Raedt, K. Michielsen, H. De Raedt, B. Trieu, G. Arnold, M. Richter, Th. Lippert, H. Watanabe, and N. Ito. Massive parallel quantum computer simulator. http://arxiv.org/abs/quant-ph/0608239, 2006.

[SA09] A. Sarje and S. Aluru. Parallel genomic alignments on the cell broadband engine. *IEEE Transactions on Parallel and Distributed Systems*, 20(11):1600–1610, November 2009.

[SdM11] E. F. de O. Sandes and A. C. M. A. de Melo. Smith-Waterman Alignment of Huge Sequences with GPU in Linear Space. In *IPDPS*, pages 1199–1211, 2011.

[SdM13] E. F. de O. Sandes and A. C. M. A. de Melo. MSA-GPU: Exact Multiple Sequence Alignment Using GPU. In *Brazilian Symposium on Bioinformatics (BSB)*, pages 47–58, 2013.

[SH04] W. H. Steeb and Y. Hardy. *Problems and solutions in Quantum Computing and Quantum Information.* World Scientific, New Jersey, 2004.

[SJSO13] Mohamed Abu Sharkh, Manar Jammal, Abdallah Shami, and Abdelkader Ouda. Resource allocation in a network-based cloud computing environment: Design challenges. *IEEE Communications Magazine*, 2013.

[SKG+10] F.J.S. Silva, F. Kon, A. Goldman, M. Finger, R.Y. Camargo, F. Castor Filho, and F.M. Costae. Application Execution Management on the InteGrade. 2010.

[SM11] Vitor E. Silva Souza and John Mylopoulos. From awareness requirements to adaptive systems: A control-theoretic approach. *2011 2nd International Workshop on Requirements@Run.Time*, pages 9–15, August 2011.

[SMB10] M. S. Sousa, A. C. M. A. Melo, and A. Boukerche. An adaptive multi-policy grid service for biological sequence comparison. *Journal of Parallel and Distributed Computing*, 70(2):160–172, February 2010.

[SMRP12] M. Schmalfuss, A. Maron, R. Reiser, and M. Pilla. q GM_c-analyzer: Biblioteca para suporte à simulação quântica em C++. In *Proceedings of the XIII WSCAD-WIC*, 2012.

[SPE13] SPEC. Standard performance evaluation corporation, 2013. `http://www.spec.org/benchmarks.html#power`. Último acesso: 13/11/2013.

[Sta] Richard Stallman. Linux and the gnu system. `https://www.gnu.org/gnu/linux-and-gnu.html`. Último acesso em: 25/10/2013.

[Sta07] Willian Stallings. *Criptografia e Segurança de Redes - Princípios e práticas*. Pearson, Brasil, 2007.

[Ste94] W Stevens. *TCP/IP illustrated*. Addison-Wesley Pub. Co, Reading, Mass, 1994.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, March 1981.

[Tah09] M. Taher. Accelerating scientific applications using gpus. In *4th International Design and Test Workshop*, pages 1–6, 2009.

[The12] TheGreenGrid. Página do projeto. `http://www.thegreengrid.org`, 2012.

[THL09] David Barrie Thomas, Lee Howes, and Wayne Luk. A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 63–72. ACM, 2009.

[Val90] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990.

[Val11] Leslie G. Valiant. A bridging model for multi-core computing. *J. Comput. Syst. Sci.*, 77(1):154–166, January 2011.

[VB] Lucas Virgili and Daniel Macêdo Batista. Modificação de DAGs de aplicações para nuvens computacionais - código. `https://github.com/lvirgili/dag_modification`. Último acesso em: 25/10/2013.

[Via07] G.F. Viamontes. *Efficient Quantum Circuit Simulation*. Phd thesis, The University of Michigan, 2007.

[Vir] Lucas Virgili. Modificação de DAGs de aplicações para nuvens computacionais. `http://www.cecm.usp.br/~lvirgili/Relatorio.pdf`. Trabalho de Conclusão de Curso.

[VPB13] M. TaherB. Sharat Chandra Varma, Kolin Paul, and M. Balakrishnan. Accelerating 3d-fft using hard embedded blocks in fpgas. In *VLSI Design*, pages 92–97, 2013.

[VS11] P. D. Vouzis and N. V. Sahinidis. Gpu-blast: Using graphics processors to accelerate protein sequence alignment. *Bioinformatics*, pages 182–188, 2011.

[WAT12] Adam Wierman, Lachlan L. H. Andrew, and Ao Tang. Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Perform. Eval.*, 69(12):601–622, December 2012.

[WCC13] Chia-Ming Wu, Ruay-Shiung Chang, and Hsin-Yu Chan. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Generation Computer Systems*, 2013.

[WvLW09] E. Weingartner, H. vom Lehn, and K. Wehrle. A performance comparison of recent network simulators. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–5, 2009.

[YM98] Eric Yu and John Mylopoulos. Why goal-oriented requirements engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, pages 15–22, 1998.

[ZRL12] Z. Zhong, V. Rychkov, and A. Lastovetsky. Data Partitioning on Heterogeneous Multicore and Multi-GPU Systems Using Functional Performance Models of Data-Parallel Applications. In *2012 IEEE International Conference on Cluster Computing (Cluster 2012)*, 2012.