Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Escalonador GPU Aware para a Plataforma de Nuvens Federadas BioNimbuZ

Francisco Anderson Bezerra Rodrigues

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof.a Dr.a Aletéia Patrícia Favacho de Araújo

Brasília
2018

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Escalonador GPU Aware para a Plataforma de Nuvens Federadas BioNimbuZ

Francisco Anderson Bezerra Rodrigues

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo (Orientador)
CIC/UnB

Prof. Dr. Goku Dr. Bruce Wayne
Planeta Veggeta Wayne Entreprises

Prof. Dr. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 32 de Onzembro de 2018

Dedicatória

Na *dedicatória* o autor presta homenagem a alguma pessoa (ou grupo de pessoas) que têm significado especial na vida pessoal ou profissional. Por exemplo (e citando o poeta): *Eu dedico essa música a primeira garota que tá sentada ali na fila. Brigado!*

Agradecimentos

Nos *agradecimentos*, o autor se dirige a pessoas ou instituições que contribuíram para elaboração do trabalho apresentado. Por exemplo: *Agradeço aos gigantes cujos ombros me permitiram enxergar mais longe. E a Google e Wikipédia.*

Resumo

O *resumo* é um texto inaugural para quem quer conhecer o trabalho, deve conter uma breve descrição de todo o trabalho (apenas um parágrafo). Portanto, só deve ser escrito após o texto estar pronto. Não é uma coletânea de frases recortadas do trabalho, mas uma apresentação concisa dos pontos relevantes, de modo que o leitor tenha uma ideia completa do que lhe espera. Uma sugestão é que seja composto por quatro pontos: 1) o que está sendo proposto, 2) qual o mérito da proposta, 3) como a proposta foi avaliada/validada, 4) quais as possibilidades para trabalhos futuros. É seguido de (geralmente) três palavras-chave que devem indicar claramente a que se refere o seu trabalho. Por exemplo: *Este trabalho apresenta informações úteis a produção de trabalhos científicos para descrever e exemplificar como utililzar a classe LaTeX do Departamento de Ciência da Computação da Universidade de Brasília para gerar documentos. A classe `UnB-CIC` define um padrão de formato para textos do CIC, facilitando a geração de textos e permitindo que os autores foquem apenas no conteúdo. O formato foi aprovado pelos professores do Departamento e utilizado para gerar este documento. Melhorias futuras incluem manutenção contínua da classe e aprimoramento do texto explicativo.*

Palavras-chave: Computação em nuvem, Federação de nuvens, escalonamento, GPGPU, BioNimbuZ

Abstract

O *abstract* é o resumo feito na língua Inglesa. Embora o conteúdo apresentado deva ser o mesmo, este texto não deve ser a tradução literal de cada palavra ou frase do resumo, muito menos feito em um tradutor automático. É uma língua diferente e o texto deveria ser escrito de acordo com suas nuances (aproveite para ler `http://dx.doi.org/10.6061%2Fclinics%2F2014(03)01`). Por exemplo: *This work presents useful information on how to create a scientific text to describe and provide examples of how to use the Computer Science Department's LaTeX class. The `UnB-CIC` class defines a standard format for texts, simplifying the process of generating CIC documents and enabling authors to focus only on content. The standard was approved by the Department's professors and used to create this document. Future work includes continued support for the class and improvements on the explanatory text.*

Keywords: Cloud Computing, Cloud Federation, schedulling, GPGPU, BioNimbuZ

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

AWS *Amazon Web Services*.

CFS *Completely Fair Scheduler*.

CPU Unidade de Processamento Central, do inglês *Central Processing Unit*.

GCP *Google Cloud PLataform*.

GPGPU Unidade de Processamento Gráfico de Propósito Geral, do inglês *General Purpose Graphics Processing Unit*.

GPU Unidade de Processamento Gráfico, do inglês *Graphics Processing Unit*.

LABID Laboratório de Informática e Dados.

RAM *Random Access Memory*.

UnB Universidade de Brasília.

Capítulo 1

Introdução

A tecnologia tem se tornado cada vez mais ubíqua na sociedade. Com o advento da Internet, a interação dos seres humanos com a tecnologia explodiu, gerando um tráfego imenso de dados, e com isso a necessidade de processamento em larga escala. Nesse cenário, surgiu o conceito de nuvem computacional, um paradigma que permite processamento em larga escala sem ser necessário que o usuário tenha em mãos hardware com tamanha capacidade computacional.

Assim, grandes empresas da área de Tecnologia da Informação, como o *Google* e a *Microsoft*, possuem vários *datacenters* com uma imensa quantidade de computadores interligados via rede, os quais disponibilizam esses recursos de forma virtualizada a usuários que necessitem de processamento e armazenamento em larga escala. A disponibilidade dessa capacidade de computação tem gerado uma revolução na forma como os serviços computacionais são disponibilizados na Internet. Dessa forma, pequenas empresas agora conseguem prover serviços em larga escala sem necessitarem de um grande investimento em infraestrutura computacional e grandes empresas conseguem reduzir custos com equipamentos.

Como existem vários provedores de nuvem e cada um deles tem seus pontos fortes e fracos, surgiu então a ideia de criar um plataforma que utilize serviços de vários provedores de nuvem, podendo explorar o ponto forte de cada um deles. Assim, emergiu o conceito de Federação de Nuvens, que são plataformas nas quais os usuários conseguem o máximo de flexibilidade provido pela combinação de funcionalidades que os distintos provedores de nuvem disponibilizam a seus usuários.

Todavia, como toda nova tecnologia, ela possui seus próprios desafios, pois desenvolver uma plataforma com tamanha flexibilidade requer uma arquitetura muito bem projetada, implementada e capaz de ser eficiente concomitantemente em que sua interface seja agradável ao usuário. Uma plataforma de Federação de Nuvens que tem sido continuamente desenvolvida é o BioNimbuZ, desenvolvido no Laboratório de Informática e Dados (LA-

BID) da Universidade de Brasília (UnB)[1] [2] [3] [4] [5] [6] [7] [8] por alunos de graduação e pós-graduação.

1.1 Objetivos

Este trabalho tem como objetivos principal desenvolver um escalonador para o BioNimbuZ que seja capaz de escalonar tarefas para arquiteturas heterogêneas. Para cumprir esses objetivos os seguintes objetivos parciais deve ser alcançados:

- Estudar algoritmos de escalonamento.
-
- Implementar um escalonador capaz de escalonar tarefas para arquiteturas heterogêneas.
- Integrar o escalonador para funcionar no BioNimbuZ, adaptando o BioNimbuZ se necessário.
- Testar o ganho de desempenho obtido ao se utilizar o escalonador desenvolvido na nuvem.

1.2 Estrutura do Trabalho

Esse projeto contém mais três capítulos e um apêndice. O segundo capítulo aborda a atividade de escalonamento e propõe o algoritmo que será implementado. O capítulo três descreve a plataforma de nuvens Federadas BioNimbuZ. E o quarto e último capítulo fala como foi o processo de implementação do escalonador.

Capítulo 2

Nuvem Computacional

Este capítulo apresenta, inicialmente a definição de nuvem computacional, suas vantagens e desvantagens. Em seguida, busca-se classificar os tipos de nuvem, tanto em termos de quais serviços são providos, quanto em termos de como a infraestrutura computacional é implementada. Por fim, é apŕesentado o conceito de federação de nuvens e como essas federações podem combinar provedores de nuvens distintos em prol da performance e da relação custo-benefício.

2.1 Definição de Computação em Nuvem

Por alguns anos a computação em nuvem não possuía uma definição bem definida principalmente por ser uma tecnologia nova. Muitas vezes sendo confundida com computação em *Grid*. Atualmente sua definição já está bem definida, com duas principais definições[9][10], que explicitam as seguintes características de uma nuvem computacional:

- *On-demand*;
- *self-service*;
- Fácil acesso via rede;
- Recursos virtualizados;
- Alta escalabilidade, com capacidade de redimensionamento durante a execução; e,
- Serviço mensurado.

Analisando o conjunto de característcas supracitadas é possível visualizar as vantagens dessa tecnologia: reduz o investimento inicial de empresas em infraestrutura de TI; a escalabilidade remove o custo de reinvestimento na infraestrutura, além de não desperdiçar investimentos feitos em servidores em momentos de pouca carga, se paga por aquilo que

se consome. Entretanto, um dos maiores desafios da computação em nuvem é segurança e privacidade, pois é necessário enviar dados da empresa para a nuvem, que pode não estar sob controle da empresa. Mas há tipos de nuvem que apresentam menos riscos em termos de segurança e privacidade, por exemplo nuvens privadas, que serão descritas a seguir..

2.2 Tipos de Nuvens

A computação em nuvem é uma evolução natural da computação em *Grid*, no qual a principal característica que o distingue da computação em nuvem é que na última os recursos computacionais são virtualizados, além do fácil acesso via Internet. Existem três tipos de serviços principais oferecidos por provedores de nuvem, que são [9]:

- ***Infrastructure as a Service:*** Também chamado de *Metal as a Service:*, o provedor disponibiliza recursos computacionais virtualizados diretamente, em termos de máquinas virtuais. Capazes de serem reconfigurados dinamicamente pelo serviço de elasticidade, por exemplo a *Oracle Cloud* [11];
- ***Plataform as a Service:*** Uma plataforma sobre a qual os usuários podem implantar softwares ou serviços é disponibilizada. Os programas implanatados podem fazer uso de algumas funcionalidade fornecidas em forma de bibliotecas, linguagens de programação, entre outros. Um exemplo básico é o serviço de hospedagem de sites, que são disponibilizados, por exemplo, no *Amazon Web Services* (AWS) e no *Google Cloud PLataform* (GCP);
- ***Software as a Service:*** O usuário tem acesso a softwares que estão sendo executados na nuvem computacional. O que permite migrar requisito computacional do equipamento onde o serviço executa para a nuvem. Um exemplo bastante conhecido é o *Google Docs* [12].

A categorização de nuvens computacionais por meio dos serviços providos não é a única forma de classificar nuvens. Elas também são categorizadas de acordo com a forma em que são implantadas. Podendo ser públicas, privadas, híbridas e comunitárias, descritas abaixo[9]:

- **Nuvem pública**: Criada para uso pelo público em geral. Podendo ser disponibilizada e/ou mantida por uma organização com ou sem fins lucrativos,
- **Nuvem privada**: Quando a infraestrutura da nuvem é provida por uma única organização, para uso interno. Ela pode ser gerenciada e/ou operada por um terceiro. Nuvens privadas são uma solução para o problema de privacidade dos dados, pois ela pode estar disponível apenas internamente na organização.

- **Nuvem comunitária**: Nuvem com infraestrutura desenvolvida com o objetivo de ser utilizada por um grupo de pessoas e/ou organizações para fins comuns. Ela pode ser gerenciada por uma ou mais partes dos interessadas na nuvem.
- **Nuvem híbrida**: Combinação de duas ou mais das opções anteriores. Geralmente unificadas por meio de padronizações de protocolos de comunicação ou uso de tecnologia proprietária.

2.3 Federações de Nuvens

Como cada provedor de nuvem possui seu próprio método de precificação, uma interface própria para comunicação com os usuários de seus serviços e funcionalidades próprias, surgiu federações de nuvens computacionais com o objetivo de minimizar dependência entre os provedores do serviço e também reduzir custos, aproveitando o melhor de cada provedor de nuvem.

Existem duas formas nas quais plataformas de federação podem combinar os serviços das nuvens: horizontalmente e verticalmente[13]. No modelo vertical instâncias de nuvem de um provedor A são criadas a partir de instâncias de nuvem de um provedor B, que pode solicitar serviços de um provedor de nuvem C, e assim por diante. No modelo horizontal as nuvens trabalham sem existir uma hierarquia entre si, assemelhando-se bastante com sistemas de rede *peer to peer*. É possível combinar mais de um estilo em uma mesma federação.

Capítulo 3

Escalonamento

O escalonamento é um problema clássico da computação. Que surgiu junto com os sistemas operacionais multitarefas. Sua complexidade é considerada NP-difícil, porém isso não impediu a evolução dos sistemas operacionais, cujos escalonadores são desenvolvidos com objetivos e metodologias diferentes para alcançarem esses objetivos. De qualquer forma um problema que todos os sistemas operacionais multitarefa buscam resolver é o *starvation*, que é quando um processo/*thread* permanece em execução por muito tempo, impedindo que outros sejam executados.

3.1 Tipos de escalonadores

Escalonadores podem ser categorizados de várias formas, com base nas funcionalidades que possui, eis alguns exemplos:

- **O que é escalonado**: Podem ser:
  - processos, *threads* para serem executados;
  - jobs para serem excecutados na nuvem;
  - páginas da memória *Random Access Memory* (RAM) para *swapping*;
  - pacotes para serem transmitidos pela rede;
  - requisições de leitura/escrita em memória secundária.
- **Suporte para preempção**: Se permite interrupção de uma tarefa em execução para atender outras;
- **Suporte para prioridade**: Se permite que priorização entre as tarefas que serão escalonadas;

- **Suporte para para tempo-real**: Se permite que requerem execução constante tenham esse requisito atendido.

O problema do escalonamento é o método pelo qual trabalho, definido por algum conjunto de características(como duração e requisitos), é atribuído à recursos que capazes de completá-lo. Por mais que exista a dificuldade teórica[14], isso não impediu a evolução dos Sistemas Operacionais, os quais foram capazes de fazer escalonamento de vários processos mesmo na época de Unidade de Processamento Central, do inglês *Central Processing Unit* (CPU)s tinham apenas um núcleo. Popularizando dessa forma os computadores pessoais na década de 80. Atualmente, as CPUs possuem vários núcleos, e são capazes de ter mais de um contexto carregado por núcleo, vide Ryzen™ Threadripper™[15]. O que faz com que seja necessário que o processo de escalonamento leve em consideração como o mesmo será distribuído(ou não) entre os núcleos.

A atividade de escalonamento pode ser otimizada para vários objetivos, entre os quais podemos citar:

- Maximinizar quantidade de trabalho realizada por unidade de tempo;
- Minimizar tempo no qual trabalhos ficam esperando para serem executados;
- Minimizar tempo entre um conjunto de trabalhos estarem prontos para serem executados até o fim da execução do conjunto(latência ou tempo de resposta);
- Distribuir de forma justa o tempo que cada um dos trabalhos terão de uso de um recurso escasso.

Esses objetivos são, às vezes, contraditórios. Na prática prioriza-se um conjunto de métricas como base para otimização. Por exemplo o GNU/Linux utiliza o *Completely Fair Scheduler* (CFS), que se baseia no algoritmo *Fair queuing*. Como o nome já diz, o foco desse escalonador está em ser justo. Internamente utiliza-se uma árvore rubro-negra indexados pelo tempo gasto no processador. Para ser justo, o tempo máximo de cada processo fica em execução interrupta é o quociente do tempo que o processo ficou aguardando para ser executado pelo número total de processos.

3.2 Escalonamento em Nuvens Computacionais

Este trabalho focará no escalonamento em federações de nuvens computacionais. O qual recentemente presenciou a ascensão do uso de unidades de processamento gráfico(Unidade de Processamento Gráfico, do inglês *Graphics Processing Unit* (GPU)) para processamento de propósito geral (Unidade de Processamento Gráfico de Propósito Geral,

do inglês *General Purpose Graphics Processing Unit* (GPGPU)[16][17]. Nesse contexto, o problema do escalonamento pode ser formalmente definido da seguinte forma:
Dado:

- Conjunto T de tarefas;
- Conjunto M de máquinas virtuais;
- Conjunto $C, |C| = |M|$ de CPUs das máquinas virtuais;
- Conjunto $G, |G| \leq |M|$ de GPUs das máquinas virtuais;
- Função $F : T \times (C \cup G) \rightarrow \mathbb{R}$, o qual estima o tempo de execução da tarefa T_i no recurso designado.

Encontrar uma função injetora $A : T \rightarrow C \cup G$, que minimize $\sum_{t \in T} F(t, A(t))$.

Por mais que o problema do escalonamento seja um clássico na área da Ciência da Computação, são poucos que lidam com escalonamento em nuvens computacionais, e ainda menos que lidam com nuvens compostas por arquiteturas heterogêneas.

/*colocar aqui revisão do estado da arte de computação em nuvem*/

3.3 Algoritmo proposto

O escalonador proposto para implementação segue a ideia básica de escalonamento de listas. Haverão três listas: lista de tarefas a serem feitas, lista de CPUs disponíveis e lista de GPUs disponíveis. A lista de tarefas é ordenada por tempo previsto de execução, que é dado por um estimativa a partir do programa a ser rodado e do arquivo de entrada. Essa ordenação será em ordem decrescente de tempo previsto. A lista de CPUs disponíveis é ordenada com base na frequência e no número de núcleos. A lista de GPUs tem sua ordem determinada pela quantidade de operações em pontos flutuante consegue realizar por segundo.

O escalonamento ocorre da seguinte forma, como ilustrado na figura 3.1 :

1. Existem tarefas que podem ser executadas nos recursos disponíveis?
2. Se sim, obtenha o processo que está no topo da lista.
   (a) É capaz de rodar em GPU?
   (b) Se sim:
      i. O tempo prevista para rodá-lo na melhor GPU disponível é melhor que o tempo previsto para rodá-lo na melhor CPU disponível?

ii. Se sim, escalone-o para nessa GPU.

iii. Se não, escalone-o para na melhor CPU disponível.

(c) Se não, escalone-o na melhor CPU disponível.

(d) Remova a tarefa e o recurso alocado de suas respectivas listas.

3. Se não, encerre o escalonamento.

4. Volte para a regra 1.

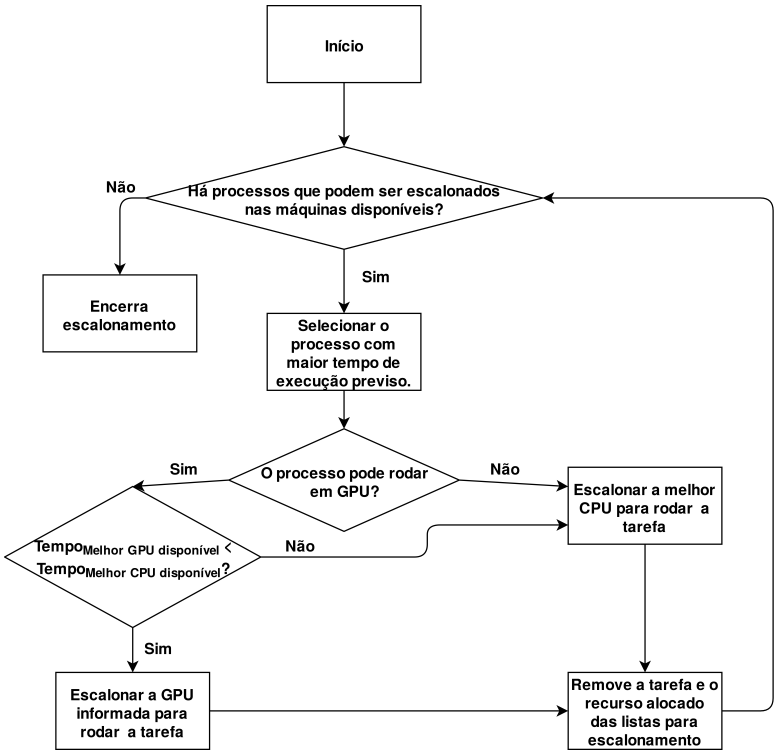

Figura 3.1: Diagrama de Funcionamento do Algoritmo de Escalonamento.

Observa-se que, para o algoritmo supracitado seja válido, pressupõe-se, que toda tarefa do algoritmo é capaz de rodar em CPU. O que é correto, pois atualmente todas as tarefas do BioNimbuZ rodam em CPU. O pressuposto simplifica o primeiro passo do algoritmo, pois se a lista de CPUs não estiver vazia, essa condição é automaticamente satisfeita. Futuramente pode ser necessário adaptá-lo para ser capaz de lidar com tarefas que só são capazes de serem executadas em GPU.

Como trabalho futuro, se o BioNimbuZ obtiver suporte executação de uma mesma tarefa de forma distribuída, uma pequena modificação que pode ser feita no algoritmo supracitado é: ao invés de remover as tarefas escalonadas da lista de tarefas, colocá-las no fim dessa mesma lista. Fará com que o escalonamento seja interrompido apenas quando todos os recursos disponíveis forem alocados, pois sempre haverá tarefas para serem alocadas.

Referências

[1] Bacelar, Breno Rodrigues Moura; Deric Lima: *Política para armazenamento de arquivos no zoonimbus*. Biblioteca de monografias da UnB, 2013. Monografia (Licenciatura em Ciência da Computação). 2

[2] Hugo Saldanha, Edward de Oliveira Ribeiro, Maristela Holanda Aleteia PF Araujo Genaína Nunes Rodrigues Maria Emilia Telles Walter Jo
btxfnamespacelong ao Carlos Setubal Alberto MR Dávila: *A cloud architecture for bioinformatics workflows.* CLOSER, 11:477, 2011. 2

[3] Lima, D., B. Moura, G. Oliveira, E. Ribeiro, A. Araujo, M. Holanda, R. Togawa e M. E. Walter: *A storage policy for a hybrid federated cloud platform: A case study for bioinformatics*. Em *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, páginas 738–747, May 2014. 2

[4] Saldanha, Hugo, Edward Ribeiro, Carlos Borges, Aletéia Araújo, Ricardo Gallon, Maristela Holanda, Maria Emília Walter, Roberto Togawa e Jo
btxfnamespacelong ao Carlos Setubal: *Towards a hybrid federated cloud platform to efficiently execute bioinformatics workflows*. Em Pérez-Sánchez, Horacio (editor): *Bioinformatics*, capítulo 05. InTech, Rijeka, 2012. `http://dx.doi.org/10.5772/50289`. 2

[5] Oliveira, G. S. S. de, E. Ribeiro, D. A. Ferreira, A. P. F. Araújo, M. T. Holanda e M. E. M. T. Walter: *Acosched: A scheduling algorithm in a federated cloud infrastructure for bioinformatics applications*. Em *2013 IEEE International Conference on Bioinformatics and Biomedicine*, páginas 8–14, Dec 2013. 2

[6] Oliveira Barreiros Júnior, Willian de: *Escalonador de tarefas para o plataforma de nuvens federadas bionimbuz usando beam search iterativo multiobjetivo*. Biblioteca de monografias da UnB, 2016. Monografia (Bacahrelado em Engenharia da Computação). 2

[7] Borges, C. A. L., H. V. Saldanha, E. Ribeiro, M. T. Holanda, A. P. F. Araujo e M. E. M. T. Walter: *Task scheduling in a federated cloud infrastructure for bioinformatics applications*. Em *Proceedings of the 2nd International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, páginas 114–120. INSTICC, SciTePress, 2012, ISBN 978-989-8565-05-1. 2

[8] Saldanha, Hugo Vasconcelos: *Bionimbus: uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática*. Tese de

Mestrado, Universidade de Brasília, 2013. http://repositorio.unb.br/handle/10482/12046. 2

[9] Mell, Peter e Timothy Grance: *The nist definition of cloud computing*. Relatório Técnico 10.6028/NIST.SP.800-145, National Institute of Standards and Technology: U.S. Department of Commerce, Sep 2011. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. 3, 4

[10] Vaquero, Luis M., Luis Rodero-Merino, Juan Caceres e Maik Lindner: *A break in the clouds: Towards a cloud definition*. SIGCOMM Comput. Commun. Rev., 39(1):50–55, dezembro 2008, ISSN 0146-4833. http://doi.acm.org/10.1145/1496091.1496100. 3

[11] Oracle: *Infrastructure as a Service | Oracle Cloud*. https://cloud.oracle.com/iaas, 2018. [Online; acessado em 23 de Fevereiro de 2018]. 4

[12] Google: *Google Docs - create and edit documents online, for free.* https://www.google.com/docs/about/, 2018. [Online; acessado em 6 de Fevereiro de 2018]. 4

[13] Celesti, A., F. Tusa, M. Villari e A. Puliafito: *How to enhance cloud architectures to enable cross-federation*. Em *2010 IEEE 3rd International Conference on Cloud Computing*, páginas 337–345, July 2010. 5

[14] Ullman, J.D.: *Np-complete scheduling problems*. Journal of Computer and System Sciences, 10(3):384 – 393, 1975, ISSN 0022-0000. http://www.sciencedirect.com/science/article/pii/S0022000075800080. 7

[15] AMD: *Ryzen^{TM} Threadripper^{TM} Processors* . http://www.amd.com/en/products/ryzen-threadripper, 2017. [Online; acessado em 21 de Novembro de 2017]. 7

[16] Dimitrov, Martin, Mike Mantor e Huiyang Zhou: *Understanding software approaches for gpgpu reliability*. Em *Proceedings of 2Nd Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-2, páginas 94–104, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-517-8. http://doi.acm.org/10.1145/1513895.1513907. 8

[17] Yang, Yi, Ping Xiang, Jingfei Kong e Huiyang Zhou: *A gpgpu compiler for memory optimization and parallelism management*. SIGPLAN Not., 45(6):86–97, junho 2010, ISSN 0362-1340. http://doi.acm.org/10.1145/1809028.1806606. 8