



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



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 Vegeta Wayne Enterprises

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

Brasília, de de

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 utilizar a classe L^AT_EX 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](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 L^AT_EX 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

| | | |
|----------|---|----------|
| 1 | Introdução | 1 |
| 1.1 | Objetivos | 1 |
| 1.2 | Estrutura do Trabalho | 1 |
| 2 | Escalonamento | 2 |
| 2.1 | Algoritmo proposto | 2 |
| 3 | BioNimbuZ | 5 |
| 4 | Implementação | 8 |
| 4.1 | Sistema de Escalonamento do BioNimbuZ | 8 |
| | Referências | 9 |

Lista de Figuras

| | |
|--|---|
| 2.1 Diagrama de Funcionamento do Algoritmo de Escalonamento. | 3 |
|--|---|

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.

GNU GNU's Not Unix[1].

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

GPL GNU General Public License[2].

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

REST *REpresentational State Transfer*.

Capítulo 1

Introdução

1.1 Objetivos

Este trabalho tem como objetivo:

- Implementar um escalonador capaz de escalonar tarefas para arquiteturas heterogêneas no BioNimbuZ.
- 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. Também haverá um apêndice falando sobre software livre.

Capítulo 2

Escalonamento

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

- Maximizar 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 indexada pelo tempo gasto no processador. Para ser justo, o tempo máximo de cada processo fica em execução interrompida é o quociente do tempo que o processo ficou aguardando para ser executado pelo número total de processos.

2.1 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 uma 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 2.1 :

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.

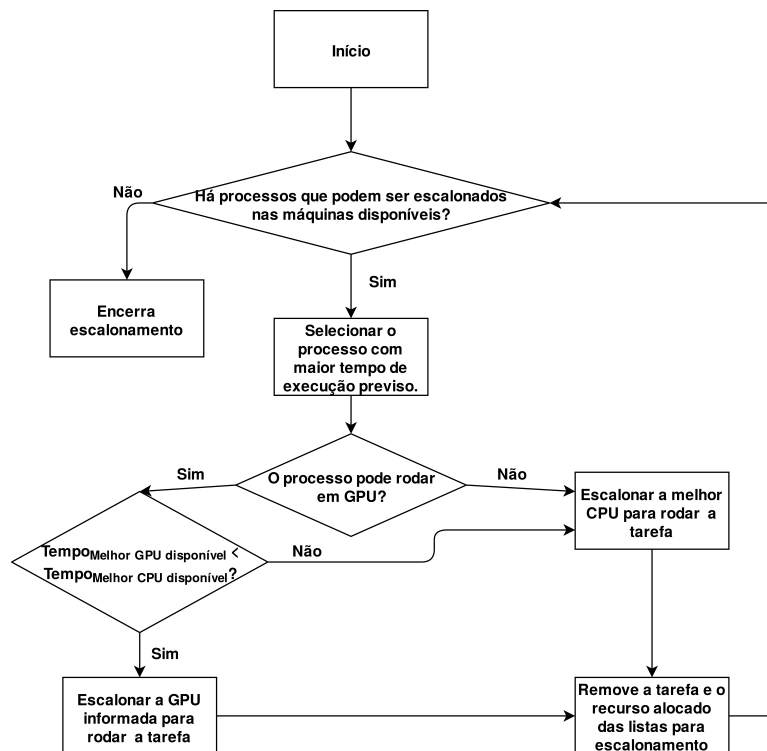


Figura 2.1: Diagrama de Funcionamento do Algoritmo de Escalonamento.

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 execução 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.

Capítulo 3

BioNimbuZ

O BioNimbuZ é uma plataforma livre de nuvens confederadas para execução de workflows de bioinformática desenvolvida no laboratório de Bioinformática e Dados(LABID) por alunos de graduação e pós-graduação. Originalmente proposta por Saldanha[16] e refinada por alunos de iniciação científica, graduação, mestrado e doutorado.[17][18][19][20][21][22]

Implementado utilizando uma arquitetura de camadas, o BioNimbuZ possui 4 camadas, descritas a seguir:

- Camada de Aplicação: Responsável por prover a interface de comunicação com o usuário, seja via uma interface gráfica(GUI), seja via *web*. Após fazer login, o usuário pode enviar *workflows* para serem executados e fazer *upload* do arquivos necessários. Além de poder acompanhar o andamento de seus *workflows* e pode obter, caso queira, o resultados parciais que já tiverem sido produzidos.
- Camada de Integração: Tem como objetivo de integrar as Camadas de Aplicação e de Núcleo, fazendo uso do *framework* REST para prover de forma prática essa funcionalidade, utilizando operações definidas no protocolo *HTTP*, como *GET*, *DELETE* e *PUT*. Existem três tipos de mensagens trocadas entre o Núcleo e a camada de Aplicação:
 - *Request*: Requisições da camada de Aplicação que contém todos os dados necessários para o seu processamento;
 - *Response*: Respostas que definem as mensagens enviadas da camada de Núcleo do BioNimbuZ; e
 - *Action*: Comandos a serem executados pelo núcleo, que são uma requisição enviada ao núcleo para se obter dados na resposta.

- Camada de Núcleo: Realiza toda a gerência da federação, provendo vários serviços. Entre eles:

- Serviço de predição: Objetiva orientar o usuário do BioNimbuZ a escolher as melhores combinações de máquinas virtuais/provedores a partir da especificação do *workflow* a ser executado e custo pretendido;
- Serviço de tarifação: Responsável por calcular o valor que os usuários devem pagar pelos serviços providndos do BioNimbuZ. Para tal, comunica-se com o serviço de monitoramento para obter informações como tempo de execução e quantidade de máquinas virtuais alocadas. É função desse serviço garantir o cumprimento das métricas de tarifação das nuvens integradas à federação, e repassar o valor ao usuário;
- Serviço de segurança: Realiza principalmente a autenticação de usuário, além de verificar as autorizações do mesmo. Contudo muitos outros aspectos de segurança computacional podem ser implementados por esse serviço, como criptografia na troca de mensagens;
- Serviço de Tolerância a Falhas: Como o nome diz, esse serviço é responsável em certificar que todos os serviços do BioNimbuZ estejam disponíveis o máximo de tempo possível. Além de ter a responsabilidade de tratar quaisquer falahas que venham a ocorrer. Tira vantagem da arquitetura distribuída do BioNimbuZ para prover redundância de dados;
- Serviço de Armazenamento: Possui a responsabilidade de gerenciar arquivos utilizados como entrada e/ou saída de cada estágio de um *workflow*. Deve desempenhar seu papel de forma eficiente, do ponto de vista de custos de armazenamento e transmissão desses dados entre o local que está armazenado e o em que serão processados;
- Serviço de Escalonamento: Responsável por fazer o escalonamento de curto prazo de *jobs*. Com curto prazo quer-se dizer que é tão somente o escalonamento de *jobs* que estão prontas para serem executadas, à máquinas virtuais que os processarão. Não é responsabilidade do serviço de escalonamento lidar com dependências, pois essa responsabilidade é do controlador de *jobs*, que será explicado em breve. No momento do início deste projeto, o BioNimbuZ possuía 5 políticas de escalonadonamento, são eles:
 - * AcoSched: Baseado em *Load Balancing Ant Colony Scheduling*, desenvolvido por Oliveira. Baseado em heurística[19]
 - * AHP: Baseado em *Analytic Hierarchy Process*[19]

- * *BasicSched*: Política *First In First Out*, implementada no início da plataforma;
 - * C99: Baseia-se no *Beam Search* interativo multiobjetivo. [22] Esse é o escalonador em uso atualmente.
 - * *RoundRobin*: O clássico escalonamento *Round Robin*;
- Camada de Infraestrutura: Disponibiliza uma interface de comunicação do BioNimbuZ com os provedores de nuvem. Utilizando *plugins* para mapear requisições provenientes da Camada de Núcleo para comandos específicos de cada provedor.

O BioNimbuZ é capaz de ser integrado tanto à nuvens públicas quanto privadas, utilizando *plugins* para permitir conexão com vários provedores de nuvem, cada qual com sua própria interface. Os *plugins* não existem apenas na camada de integração: vários serviços da camada de núcleos também são disponibilizados como tal, provendo grande flexibilidade à plataforma.

Internamente, o BioNimbuZ utiliza o Apache Zookeeper[23] para prover serviços de coordenação de ambientes distribuídos. Desenvolvido pela fundação Apache[28], tem como objetivo ser e fácil manuseio. Possui um modelo de dados semelhante a uma estrutura de diretórios. Uma outra tecnologia que também é utilizada no BioNimbuZ é o Apache Avro[24], para serialização de dados para transmissão pela rede.

Capítulo 4

Implementação

4.1 Sistema de Escalonamento do BioNimbuZ

Referências

- [1] GNU: *The GNU Operating System and the Free Software Movement*. <https://www.gnu.org/>, 2017. [Online; acessado em 23 de Novembro de 2017]. ix
- [2] GNU: *The GNU General Public License v3.0 GNU Project - Free Software Foundation*. <https://www.gnu.org/licenses/gpl-3.0.en.html>, 2017. [Online; acessado em 23 de Novembro de 2017]. ix
- [3] 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>.
- [4] AMD: *Ryzen™ Threadripper™ Processors* . <http://www.amd.com/en/products/ryzen-threadripper>, 2017. [Online; acessado em 21 de Novembro de 2017].
- [5] 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>.
- [6] 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>.
- [7] Oguchi, T., M. Higuchi, T. Uno, M. Kamaya e M. Suzuki: *A single-chip graphic display controller*. Em *1981 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, volume XXIV, páginas 170–171, Feb 1981.
- [8] Dalton, Steven, Luke Olson e Nathan Bell: *Optimizing sparse matrix-matrix multiplication for the gpu*. ACM Trans. Math. Softw., 41(4):25:1–25:20, outubro 2015, ISSN 0098-3500. <http://doi.acm.org/10.1145/2699470>.
- [9] Mulligan, Jeffrey B.: *A gpu-accelerated software eye tracking system*. Em *Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA '12*, páginas 265–268, New York, NY, USA, 2012. ACM, ISBN 978-1-4503-1221-9. <http://doi.acm.org/10.1145/2168556.2168612>.
- [10] Services, Amazon Web: *Amazon EC2 Instance Types - Amazon Web Services(AWS)*. <https://aws.amazon.com/ec2/instance-types/>, 2017. [Online; acessado em 21 de Novembro de 2017].

- [11] Plataforma, Google Cloud: *Google Compute Engine Pricing / Compute Engine Documentation / Google Cloud Platform*. <https://cloud.google.com/compute/pricing>, 2017. [Online; acessado em 21 de Novembro de 2017].
- [12] Azure, Microsoft: *Azure Linux VM sizes - GPU*. <https://docs.microsoft.com/en-us/azure/virtual-machines/linux/sizes-gpu>, 2017. [Online; acessado em 21 de Novembro de 2017].
- [13] Cloud, IBM: *GPU Computing - IBM Cloud*. <https://www.ibm.com/cloud/bare-metal-servers/gpu>, 2017. [Online; acessado em 21 de Novembro de 2017].
- [14] 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>.
- [15] 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>.
- [16] 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>. 5
- [17] 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. 5
- [18] Lima, D., B. Moura, G. Oliveira, E. Ribeiro, A. Araujo, M. Holanda, R. Togawa e M. E. M. T. 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. 5
- [19] 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. 5, 6
- [20] 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). 5
- [21] Vergara, Guilherme Fay: *Arquitetura de um controlador de elasticidade para nuvens federadas*. 2017. 5

- [22] 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 (Bacharelado em Engenharia da Computação). 5, 7
- [23] Foundation, The Apache Software: *Apache Zookeeper - Home*. <https://zookeeper.apache.org/>, 2017. [Online; acessado em 24 de Novembro de 2017]. 7
- [24] Foundation, The Apache Software: *Welcome to Apache Avro!* <https://avro.apache.org/>, 2017. [Online; acessado em 24 de Novembro de 2017]. 7
- [25] Project, GNU: *What is Free Software? - GNU Project - Free Software Foundation*. <https://www.gnu.org/philosophy/free-sw.html>, 2017. [Online; acessado em 24 de Novembro de 2017].
- [26] Project, GNU: *Hurd*. <https://www.gnu.org/software/hurd/index.html>, 2017. [Online; acessado em 24 de Novembro de 2017].
- [27] Project, Debian: *Debian – Debian GNU/kFreeBSD*. <https://www.debian.org/ports/kfreebsd-gnu/>, 2017. [Online; acessado em 24 de Novembro de 2017].
- [28] Foundation, The Apache Software: *Welcome to Apache Software Foundation!* <https://apache.org/>, 2017. [Online; acessado em 15 de Dezembro de 2017]. 7