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 Veggeta Wayne Entreprises

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 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

Capítulo 1

Introdução

1.1 O problema do escalonamento

O escalonamento é o método pelo qual trabalho, definido por algum conjunto de características, é atribuído à recursos que capazes de completá-lo. O Problema do Escalonamento, que é a busca do escalonamento no qual o tempo de execução do conjunto de trabalhos é mínimo, é NP-completo[**?**]. De acordo com Sipser[**?**], um problema A é NP-completo se:

1. A pertence à classe de problemas NP, e
2. Todo problema B $\in$ NP é redutível em tempo polimonial a A.

Um problema pertence à NP se a for possível verificar a validade de uma solução em tempo polimonial.[**?**][**?**].

Por mais que exista a dificuldade teórica supracitada, 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^{TM} Threadripper^{TM}[**?**]. 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.

Além do escalonamento de processos para serem executados na CPU, o problema do escalonamento também aparece em várias outras situações na computação:

- Escalonamento de processos que farão swapping.

- Escalonamento de requisições que farão acesso ao disco.
- Escalonamento de pacotes que serão enviados pela rede.
- Escalonamento de tarefas para serem executadas em máquinas virtuais numa nuvem.

Esta monografia focará no último tópico citado. O qual recentemente presencio ua ascenção 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)[**?**][**?**].

1.2 Unidade de Processamento gráfico

As GPUs surgiram inicialmente como hardware dedicado embutido para acelerar a renderização de imagens em jogos eletrônicos de arcades por volta dos anos 80[**?**].

Com o passar das décadas e a evolução da tecnologia, as GPUs ficaram cada vez mais poderosas e complexas, principalmente para atender à rápida evolução dos jogos eletrônicos. Quando o computador pessoal popularizou, surgiu as GPUs como as conhecemos, como placas acopláveis à placa mãe focadas em processamento gráfico.

Na última década, as unidades de processamento gráfico(GPUs) ascenderam como opção para processamento paralelo. Em especial operações que envolvam matrizes conseguem ter um ganho de performance considerável[**?**], como processamento de imagens e vídeos[**?**].

Atualmente provedores de nuvem estão provendo máquinas virtuais com GPUs, como a *Amazon Web Services* (AWS)[**?**], o *Google Cloud PLataform* (GCP)[**?**], *Microsoft Azure*[**?**] e a *IBM Cloud*[**?**].

1.3 Computação em Nuvem

A Computação em Nuvem é um sistema distribuído que disponibiliza serviços de computação ao usuário, abstraindo informações sobre como esse serviço é provido. Existem muitas características que ajudam a definir a Computação em Nuvem, mas as consideradas principais são[**?**]:

- *On-demand self-service*;
- Fácil acesso via rede;
- Recursos virtualizados.

- Alta escalabilidade, com capacidade de redimensionamento durante a execução; e,
- Serviço mensurado.

Computação em nuvem é considerado um novo paradigma para provisionamento de infraestrutura de computação, o qual transfere o local da infraestrutura para a rede. Porém parte dos conceitos sobre os quais esse conceito se baseia não são novos.[?]CITAR CITAÇÕES DO ARTIGO BASE. Na Computação em Nuvem, existem dois atores principais: os usuários de um serviço que está na Nuvem e o provedor da Nuvem. Existem três tipos de serviços pricipais que são[?]:

- ***Infrastructure As A Service:*** O provedor disponibiliza recursos computacionais virtualizados. Capazes de serem reconfigurados dinamicamente;
- ***Plataform As A Service:*** Uma plataforma sobre a qual os usuários podem implantar softwares ou serviços é disponibilizada. Os programas implanatado podem fazer uso de algumas funcionalidade fornecidas em forma de bibliotecas, linguagens de programação, entre outros;
- ***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.

Como cada provedor de nuvem possui seu próprio método de precificação, e uma interface própria para comunicação com os usuários de seus serviços, existe um movimento de federação de nuvens computacionais com o objetivo de minimizar dependência para com os provedores do serviço e também reduzir custos.

1.4 Nuvens Federadas: BioNimbuZ

O BioNimbuZ é uma plataforma livre de nuvens federadas para execução de workflows. Desenvolvido no laboratório de Bioinformática e Dados(LABID) por alunos de graduação e pós-graduação. Originalmente proposta por Saldanha[?] e refinada por alunos de iniciação científica, graduação, metrado e doutorado.[?][?] [?][?][?][?]

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*. A partir dessa camada o usuário pode enviar *workflows* para serem executados e fazer *upload* do arquivos necessários. Além de poder acompanhar o andamento de seus *workflows* e poder 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.
- Camada de Núcleo: Realiza toda a gerência da federação, incluindo: escalonamento de tarefas, controle de acesso de usuários, descobrimento de recursos e provedores, prover serviços de elasticidade, monitoramento, armazenamento entre outros.
- 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.

Desenvolvido em Java, utiliza o Apache Zookeeper[**?**] para coordenação do sistema distribuído e o Apache Avro[**?**] para serialização de dados. É capaz de utilizar os serviços de nuvem da *Microsoft Azure*, *AWS* e a *GCP*.

1.4.1 Software Livre

O BioNimbuZ está disponível sob os termos da GNU General Public License[**?**] (GPL), a licença do Projeto GNU's Not Unix[**?**] (GNU). O Projeto GNU foi criado por Richard Stallman em 1983, com o objetivo de fazer um sistema operacional que respeitasse a liberdade dos usuários de utilizar os software que possuem da forma que lhe for mais conveniente. Essas liberdades são[**?**]:

- Liberade de rodar o programa da forma que quiser, e para qualquer propósito.
- Liberdade de estudar como o programa funciona, e poder modificar ele para o mesmo faça a computação da forma que o usuário quiser.
- Liberdade para redistribuir cópias.
- Liberdade de redistribuir cópias das versões modificadas para outros.

Para que essas liberdades sejam exercidas é necessário acesso ao código fonte. Atualmente o projeto GNU conseguiu colocar em produção o sistema operacional GNU. As distribuições mais populares do GNU utilizam o kernel Linux desenvolvido por Linux Towards, por mais que seja possível usar outros kernels como o Hurd[**?**] e o do FreeBSD[**?**].

Hoje em dia, softwares livre são robustos o bastante para opções válidas perante softwares proprietários. Principalmente em ambiente acadêmico, pois o compartilhamento de conhecimento é fundamental para o desenvolvimento da ciência e tecnologia.

Capítulo 2

Escalonamento

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 utilizada 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.

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 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 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.

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.

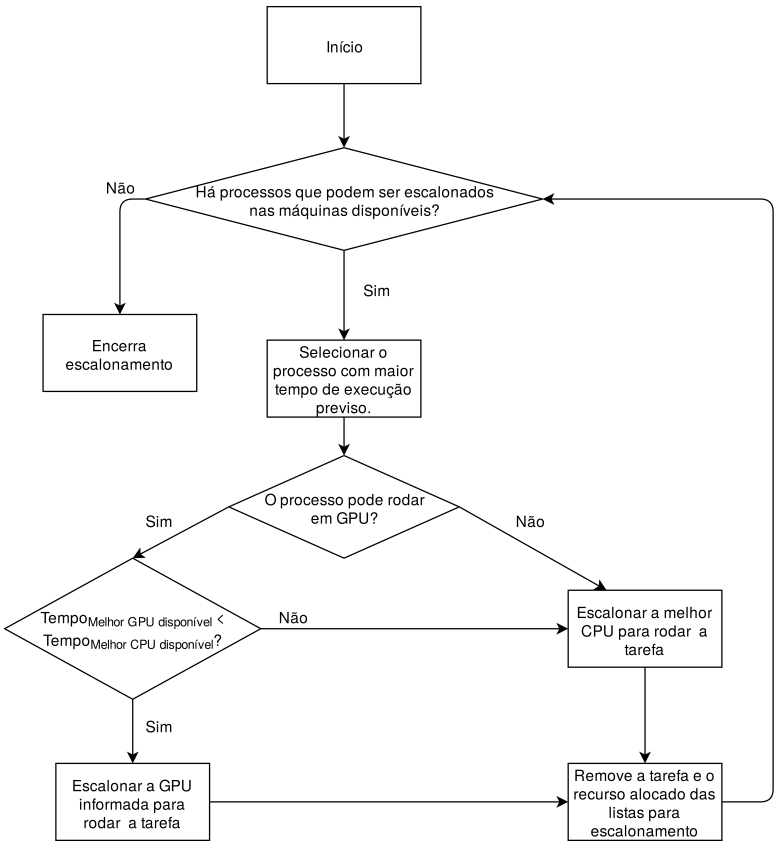

Figura 2.1: Diagrama de Funcionamento do Algoritmo de Escalonamento.