# Proposta de um Escalonador Multicritério Acelerado por GPU para Nuvens OpenStack

**Leonardo R. Rodrigues**[1], **Omir C. A. Jr.**[1], **Marcelo Pasin**[2], **Guilherme P. Koslovski**[1]

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA) - UDESC

[2]University of Neuchâtel (UniNE) - Switzerland

***Resumo.*** *Hospedar Infraestruturas Virtuais (IVs) em Data Centers (DCs) é uma tarefa NP-Difícil. A maioria das abordagens propostas na literatura para encontrar uma solução perante as funções objetivos, executam somente em CPU. O processamento em Graphics Processing Units (GPUs) removeu barreiras de escalabilidade de algoritmos. Este trabalho propõe desenvolver métodos multicritérios em GPU para alocar IVs em DC gerenciados por OpenStack.*

## 1. Introdução e Motivação

Na computação em nuvem, um provedor Infrastructure-as-a-Service (IaaS) pode atender múltiplos clientes disponibilizando IVs para seus clientes. O gerenciamento e escalonamento das IVs apresentam um impacto direto na qualidade dos serviços ofertados pelos provedores. Especificamente, o escalonador é responsável por selecionar e provisionar os recursos físicos para hospedar Máquinas Virtuais (MVs), contêineres e redes virtuais. Sobretudo, realizar o escalonamento é um problema *NP-Difícil*, devido a diversos fatores, como: *(i)* as requisições dos clientes e os recursos do DC são heterogêneos; *(ii)* os recursos do DC são limitados, necessitando o gerenciamento da admissão das requisições, recusando ou adiando as requisições que não possam ser prontamente atendidas; *(iii)* o escalonador deve trabalhar com diferentes topologias de DC e com variados tipos de Infraestrutura Virtual (IV). Em nuvens baseadas no *Cloud Operating System (COS)* OpenStack, o gerenciamento dos recursos virtuais é distribuído entre diversos módulos (ou projetos). Embora um escalonamento combinado de recursos de processamento (MVs e contêineres) e comunicação (*switches* e enlaces) seja benéfico para provedores e clientes [d. Souza et al. 2017], o aspecto gerencial distribuído utilizado no OpenStack dificulta o processo.

A tomada de decisão efetuada pelo escalonador seleciona uma alternativa dentre um conjunto de alternativas viáveis, possuindo o objetivo de maximizar os resultados através de um conjunto de critérios. Em resumo, um método Análise de Decisão Multicritério (ADM) consiste em [Saaty and Vargas 2012]: *(i)* definir o conjunto de alternativas que possam solucionar o problema; *(ii)* definir o conjunto de critérios para análise; *(iii)* a avaliação de cada alternativa relacionado aos critérios modelados; e *(iv)* avaliar o impacto do resultado final sobre cada alternativa [Fischer et al. 2013]. Entretanto, devido a necessidade de respostas instantâneas, os escalonadores utilizam algoritmos simples para solucionar o problema (*e.g., best fit*,*worst fit*, *round-robin*). Esta restrição torna incabível a aplicação de algoritmos complexos em provedores reais.

Os desafios tecnológicos e científicos são evidentes no escalonamento combinado de recursos em nuvens OpenStack. Diante do exposto, o objetivo deste trabalho é propor um escalonador multicritério acelerado por GPU, integrado com o OpenStack, agregando

a alocação de uma IV em um único módulo, possuindo três diretivas principais: *(i)* otimizar a função objetivo do provedor; *(ii)* ser escalável, tornando possível a aplicação do escalonador em provedores reais; e *(iii)* reduzir a sobrecarga de comunicação intermódulos. Para tanto, o restante do trabalho está organizado como segue. A Seção 2 discute os trabalhos relacionados, enquanto a Seção 3 descreve a proposta de escalonador otimizado em GPU. A Seção 4 apresenta as considerações finais do artigo e perspectivas.

## 2. Trabalhos Relacionados

A literatura apresenta soluções de escalonadores que utilizam um amplo conjunto de métodos (*e.g.,* algoritmos genéticos, redes neurais e programação linear) para fornecer soluções próximas do ótimo. A Tabela 1 apresenta uma síntese dos principais trabalhos elencados. Inicialmente, é possível observar que a maioria das abordagens são desenvolvidas para executar somente em CPU. Quanto ao método de virtualização (MV ou contêiner), não foram encontrados artigos que suportem ambos simultaneamente. Inclusive, o escalonamento de recursos virtuais de comunicação não é abordado por todos os trabalhos. Um destaque é apontado para Nesi e seus colegas que propuseram *framework* acelerado em GPU para realizar a alocação de IVs demonstrando a escalabilidade do problema através do uso de programação paralela, obtendo *speedups* de até $6234$x em algoritmos específicos [Nesi et al. 2018a]. O *framework* suportará o escalonador proposto.

| Artigo | Método Utilizado | Virtualização | Rede Virtual | Arq. | Servidores |
|---|---|---|---|---|---|
| [d. Souza et al. 2017] | Programação Linear Mista | MVs | Sim | CPU | $64$ |
| [Zhou et al. 2016] | Analytic Hierarchy Process (AHP) e Redes Neurais | MVs | Sim | CPU | $69$ |
| [Guerrero et al. 2018] | Algoritmos Genéticos | Contêiner sobre MVs | Não | CPU | $50$ |
| [Nesi et al. 2018b] | Métodos de busca local e de agrupamento | MVs | Sim | GPU | $24334$ |
| O autor (proposta) | Métodos multicritério e de agrupamento | Contêineres e MVs | Sim | GPU | $\geq 24334$ |

**Tabela 1. Síntese dos trabalhos correlatos.**

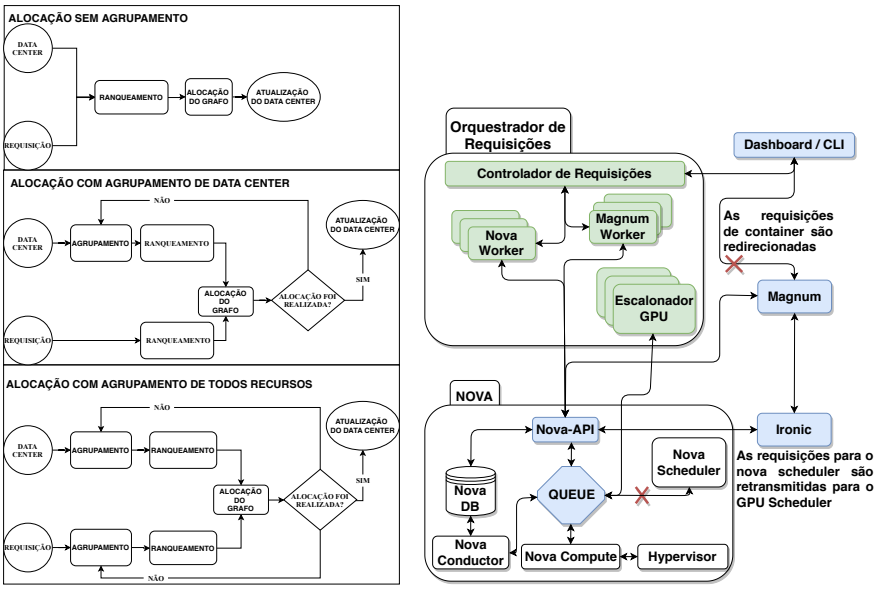
## 3. Proposta do Escalonador para OpenStack

A proposta consiste no escalonamento combinado de MVs, contêineres e recursos de comunicação em um ambiente OpenStack. Cada recurso virtual será analisado por métodos multicritério, otimizando a função objetivo definida pelo gerenciador da nuvem. Ademais, os algoritmos de escalonamento serão processado em GPU. A seguir, cada item da proposta é individualmente detalhado.

### 3.1. Estratégias de Escalonamento

Para ser adaptável a vários provedores com diferentes configurações de DC, três estratégias para escalonamento das requisições são propostas. A seleção da estratégia apropriada ocorrerá dinamicamente (*i.e.,* o escalonador obterá o tamanho do DC e selecionará a estratégia através de um conjunto de políticas), conforme apresentado pela Figura 1(a).

A primeira estratégia, denominada de **Alocação Sem Agrupamento** é aplicável a pequenos DCs. Nesta abordagem, o DC o método de seleção multicritério é aplicado para todos os candidatos, e após a obtenção do ranqueamento dos servidores, a alocação é realizada. A segunda estratégia denominada **Alocação Com Agrupamento de DC** é utilizada para agrupar servidores e, em seguida, aplicar o método multicritério para obtenção




(a) Estratégias de Alocação. (b) Arquitetura do OpenStack.

**Figura 1. Proposta de escalonador.**

do grupo mais adequado. A abordagem será recursivamente aplicada até a identificação de um limiar (número de servidores) aceitável para o tamanho dos grupos. Se o limiar for atingido, o método multicritério expandirá o grupo em questão ranqueando os servidores e identificando os recursos hospedeiros. Por fim, caso a requisição ultrapasse um determinado número de recursos (contêineres, MVs e enlaces), é selecionada então a terceira estratégia, denominada de **Alocação Com Agrupamento de Todos os Recursos**. A estratégia é utilizada para formar agrupamentos dos recursos do DC e da requisição do usuário, possuindo o princípio similar ao descrito na segunda estratégia.

### 3.2. Arquitetura do Escalonador GPU

Para realizar o escalonamento combinado de MVs e contêineres, alterações no fluxo de dados da arquitetura do OpenStack são propostas (Figura 1(b)). Ainda, a sobrecarga de comunicação entre os módulos do OpenStack será reduzida, através da inclusão de 4 módulos na arquitetura do OpenStack (módulos verdes) e na alteração das chamadas de *Remote Procedure Call* (RPC) dos módulos existentes (módulos azuis).

O módulo **Controlador de Requisições** concentra o processamento das requisições de MVs, contêineres e rede, atuando como uma orquestrador dos processos de Nova e Magnum. Para o seu funcionamento é necessário alterar as chamadas RPC dos módulos do Nova e Magnum para que as novas requisições sejam enviadas primeiramente para o controlador. O **Nova Worker** é um módulo que recebe as requisições de MVs do Contador de Requisições, enquanto o **Magnum Worker** orquestra as requisições de contêineres, verificando se é uma requisição sobre Máquina Virtual (MV) ou sobre *bare metal*. A distinção é essencial para direcionar as chamadas RPCs. Ainda, o escalonador verifica a quantidade de instâncias de MVs e *bare metal* são necessárias para suportar a requisição de contêineres, realizando um escalonamento combinado e otimizado.

O módulo do **Escalonador GPU** combina métodos multicritérios e de agrupa-

mento, implementado os algoritmos da Figura 1(a). O *kernel*[1] do escalonador é adaptável as configurações especificadas pelo gerenciador. A arquitetura do escalonador é composta por 5 submódulos. O **Controlador de *Buffer*** é responsável pela orquestração do fluxo de requisições, organizando-as em filas de prioridade. O **Seletor Multicritério** alterna o método multicritério (*e.g.*, AHP, TOPSIS) a ser executado, de acordo com a política do provedor. Por sua vez, o **Seletor de Agrupamento** altera o algoritmo de acordo com a topologia do DC. O módulo possui os métodos de *Markov Cluster Algorithm* (MCL), *Multiple Correlation Clustering* (MCC) e K-Means para reduzir o espaço de busca. O **Seletor de Funções Objetivo** adequa o escalonador com as necessidades do provedor. Por fim, o **Analisador de Requisições** recebe os servidores ranqueados e realiza chamadas para o controlador de *buffer*. Após, o analisador retorna o servidor mais adequado, obtido pelo seletor multicritérios, para a requisição em questão.

## 4. Considerações

Os escalonadores atuais sofrem com problemas de escalabilidade, havendo uma dicotomia entre o tempo de execução e a qualidade da resposta fornecida. O escalonador proposto utiliza programação paralela em GPU para reduzir o tempo de computação necessário para selecionar o servidor adequado a atender a requisição. Ainda, a proposta busca o escalonamento combinado de MVs e contêineres para otimizar o DC.

**Agradecimentos** Os autores agradecem ao LabP2D, FAPESC e UDESC.

## Referências

d. Souza, F. R., Miers, C. C., Fiorese, A., and Koslovski, G. P. (2017). Qos-aware virtual infrastructures allocation on sdn-based clouds. In *17th IEEE/ACM CCGrid*.

Fischer, A., Botero, J. F., Beck, M. T., De Meer, H., and Hesselbach, X. (2013). Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*.

Guerrero, C., Lera, I., and Juiz, C. (2018). Genetic algorithm for multi-objective optimization of container allocation in cloud architecture. *J GRID COMPUT*, 16(1):113–135.

Nesi, L. L., Pillon, M. A., de Assunção, M. D., and Koslovski, G. P. (2018a). GPU-accelerated algorithms for allocating virtual infrastructure in cloud data centers. In *18th IEEE/ACM CCGrid*.

Nesi, L. L., Pillon, M. A., de Assunção, M. D., Miers, C. C., and Koslovski, G. P. (2018b). Tackling virtual infrastructure allocation in cloud data centers: a gpu-accelerated framework. In *14th CNSM 2018*.

Saaty, T. L. and Vargas, L. G. (2012). *Models, methods, concepts & applications of the analytic hierarchy process*, volume 175. Springer Science & Business Media.

Zhou, X., Lin, F., Yang, L., Nie, J., Tan, Q., Zeng, W., and Zhang, N. (2016). Load balancing prediction method of cloud storage based on analytic hierarchy process and hybrid hierarchical genetic algorithm. *SpringerPlus*, 5(1):1989.

[1]Nomenclatura usada para identificar funções executadas em GPU.