Alocação de Ambientes Virtuais com base na Afinidade entre Perfis de Aplicações Massivamente Paralelas e Distribuídas

Victor Oliveira[1 2], **Jonathan Barbosa**[2], **Matheus Bandini**[2], **Bruno Schulze**[2], **Raquel Pinto**[1]

[1]Instituto Militar de Engenharia (IME)
Rio de Janeiro – RJ – Brasil

[2]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis – RJ – Brasil

{victord,jonathanpb,mbandini,schulze}@lncc.br, raquel@ime.eb.br

Abstract. *This paper presents a virtual machine scheduling algorithm to run Massively Parallel and Distributed Computing applications with intensive usage of CPU, memory and I/O. The scheduling algorithm is proposed in order to address the allocation of virtual machines in Cloud Computing environments based on the concept of "Affinity" between applications. For the method's implementation, the virtual machines resource consumption was monitored, in order to obtain historical data that allows to determine application profiles. As a result, a virtual machine scheduler for physical shared resources was created. Its objectives are to avoid combinations of applications that may cause performance degradation and to improve the usage rate of computing resources.*

Resumo. *Este artigo apresenta um algoritmo de escalonamento de máquinas virtuais que executam aplicações de Computação Massivamente Paralela e Distribuída (CMPD), com uso intensivo de CPU, memória e I/O. O método de escalonamento tem por finalidade tratar a alocação das máquinas virtuais em ambientes de nuvens computacionais com base no conceito de "Afinidade" entre aplicações. Para a implementação do método, foi realizado o monitoramento do consumo de recursos das máquinas virtuais para obter dados históricos de execução que permitem determinar os perfis das aplicações. Como resultado, foi criado um escalonador para alocar máquinas virtuais, cujos objetivos são evitar combinações de aplicações que causem degradação do desempenho entre si e melhorar a taxa de utilização dos recursos computacionais.*

1. Introdução

Na tentativa de melhor aproveitar o uso dos recursos computacionais de modo a reduzir custos, novas técnicas, tecnologias e arquiteturas desenvolvidas estão conquistando grande aceitação no mercado e no meio acadêmico. Uma dessas tecnologias é a Computação em Nuvem, que tenta resolver problemas como consumo energético e alocação de espaço físico em grandes centros de processamento de dados e de CMPD (Computação Massivamente Paralela e Distribuída).

A preocupação crescente com a qualidade dos serviços prestados por provedores de nuvens computacionais motiva pesquisas focadas em desenvolver mecanismos e

metodologias para promover melhorias na forma de alocar aplicações nesses recursos [Zheng et al. 2013]. Neste sentido, conhecer o perfil do consumo de recursos das aplicações, os ambientes virtualizados e os efeitos causados pela concorrência contribui para esses esforços, no sentido de minimizar as perdas de desempenho.

Com o objetivo de otimizar o uso da infraestrutura disponível, um dos fatores que caracterizam a Computação em Nuvem é a possibilidade de haver a competição por um mesmo recurso físico, devido a dois ou mais ambientes virtualizados compartilharem tais recursos. A concorrência, entretanto, pode resultar na degradação das aplicações de nuvem. De acordo com [Emani and O'Boyle 2015], a alocação inadequada de aplicações concorrentes pode causar a degradação do desempenho. Caso os limites especificados por contratos de Qualidade de Serviço sejam extrapolados, a proposta de nuvem pode ser invalidada. Por essa razão, torna-se necessário desenvolver métodos de escalonamento de aplicações de nuvem que permitam alocar aplicações que possuam características diferentes e, por conseguinte, reduzem o impacto da concorrência entre si. Esse princípio é conhecido na literatura como Afinidade entre aplicações [Licht 2014].

Dado que as aplicações em uma máquina virtual podem alterar o seu perfil de uso de recursos computacionais ao longo da execução, é necessário analisar essa mudança no caso em que for gerada degradação em outros ambientes virtuais. Por exemplo, apenas classificar a aplicação como intensiva em processamento (CPU-*Bound*), não permite afirmar que esta máquina virtual vai ocupar 100% de CPU durante todo o tempo de execução. Existe a possibilidade de que, em um dado momento, a aplicação troque seu perfil de consumo e comece a utilizar outro recurso intensamente [Schad et al. 2010]. É neste momento que se apresenta uma das motivações deste trabalho, onde a alteração no perfil da aplicação pode acarretar sobrecarga e degradação na execução das outras máquinas virtuais que estão alocadas em um mesmo hospedeiro.

O uso do conceito de Afinidade entre aplicações [Mury et al. 2014] visa contribuir na alocação do ambiente virtual que se baseia nas características de consumo e afinidade entre aplicações. Sendo assim, é necessário monitorar e analisar os diversos perfis das aplicações, estabelecidos através do histórico do consumo de recursos. Por meio deste estudo, foi definido um grau de afinidade que é utilizado pelo escalonador para otimizar o processo de alocação e migração dos ambientes virtuais em uma nuvem, afim de evitar o impactado da concorrência dos recursos computacionais.

Este trabalho propõe duas técnicas de escalonamento, o estático e o dinâmico. Estas duas técnicas referem-se ao momento em que as decisões são tomadas. No escalonamento estático, os perfis das aplicações são previamente conhecidos e, uma vez escalonadas, as máquinas virtuais são mantidas na mesma máquina física até o fim da execução. Entretanto, no escalonamento dinâmico, pode-se não ter conhecimento inicial sobre as características da aplicação, de forma que o perfil de uso dos recursos pode mudar ao longo da execução. As aplicações chegam ao escalonador em momentos distintos de tempo. Quando o escalonador detecta alteração no perfil da aplicação, ele pode decidir migrar as máquinas virtuais de forma a evitar a queda do desempenho daquelas que compartilham o mesmo ambiente físico [Alam and Varshney 2016].

A seção 2 deste artigo discute alguns trabalhos relacionados aos principais tópicos abordados durante o desenvolvimento do método de escalonamento proposto. A seção 3

apresenta a metodologia utilizada para determinar a Afinidade entre um conjunto de aplicações e entre recursos computacionais. A seção 4 apresenta o método de escalonamento proposto e descreve a arquitetura do escalonador desenvolvido. A seção 5 apresenta os resultados dos experimentos que buscaram validar o algoritmo de escalonamento criado. Por fim, a seção 6 conclui o artigo e apresenta propostas de trabalhos futuros.

2. Revisão da Literatura

Em [Mury et al. 2014], é avaliado o aumento do uso de ambientes virtualizados. Entretanto, a maioria desses estudos são limitados a um nível de análise de desempenho, não aprofundando o estudo sobre os efeitos da concorrência entre os vários ambientes virtuais, nem como mitigar esses efeitos. Esse trabalho apresentou o conceito de Afinidade, que define o grau de coexistência entre as classes de aplicações. As classes de algoritmos associadas aos tipos de bibliotecas paralelas utilizadas na implementação dessas aplicações influenciam nas suas combinações. Os resultados demonstraram que os efeitos dessas combinações têm valores diversos, o que torna necessário detalhar o estudo para classificá-los. Sendo assim, justifica-se definir e analisar o conceito de "Afinidade", buscando melhorar o uso dos recursos, sobretudo no que tange a Computação Massivamente Paralela e Distribuída. Por fim, a principal contribuição foi a criação de tabelas comparativas de desempenho concorrente que permitem a análise visual dos resultados. Dessa forma, é possível avaliar os impactos causados pela concorrência, o que possibilita a rápida avaliação das melhores combinações, bem como daquelas que devem ser evitadas.

O trabalho de [Yokoyama 2015] apresentou uma plataforma de nuvem privada voltada à criação e gerência de *clusters* computacionais para a aplicação na resolução de tarefas de computação de alto desempenho. Foi adotada como base uma arquitetura paralela de memória distribuída. Além de desenvolver um sistema para criação de *clusters* computacionais em nuvem, o trabalho ainda apresentou um modelo de escalonamento *offline* de máquinas virtuais baseado na Afinidade entre as aplicações de *Benchmark* em execução nos hospedeiros. Tal modelo de alocação busca melhorar o aproveitamento dos recursos disponíveis na infraestrutura e a vazão de tarefas executadas.

A abordagem deste trabalho difere da apresentada por [Yokoyama 2015] em três aspectos principais: **i)** o *ProSched* realiza o escalonamento *online* e *offline* das aplicações virtuais; **ii)** o escalonamento proposto foi aplicado tanto para aplicações reais como para *Benchmarks*; e **iii)** possui a capacidade de migrar as aplicações de acordo com a afinidade entre elas, o que não é contemplado por [Yokoyama 2015].

[Bernado 2014] apresenta uma arquitetura capaz de suportar os efeitos de sobrecargas momentâneas em servidores físicos e virtuais hospedados em ambientes de nuvem. A arquitetura, denominada *Phoenix*, tem o objetivo de automatizar a gerência de máquinas virtuais hospedadas em uma nuvem. Além disso, é proposto um esquema de balanceamento de carga nos momentos em que não há sobrecarga dos recursos, associada a uma estratégia reativa, que é acionada caso tais sobrecargas sejam identificadas.

[Calheiros et al. 2011], por sua vez, apresenta o *CloudSim*, cujo objetivo é fornecer um sistema de simulação que permite a modelagem, a simulação e a experimentação de infraestruturas de nuvem e serviços de aplicativos. Dentre as conclusões obtidas, está a identificação da necessidade de monitorar as aplicações, com a finalidade de otimizar o uso da nuvem computacional e verificar os efeitos da concorrência na infraestrutura.

Entretanto não é apresentado um estudo que defina quais tipos de aplicações poderiam coexistir nestes ambientes virtuais, sem que haja a degradação em função da concorrência por recursos computacionais.

No trabalho de [Simmons et al. 2007], é proposta a criação de um sistema de tomada de decisões baseado em SLAs para a agregação de recursos de forma otimizada. Na proposta, há um controle de uso de recursos que pontua os gastos e compara com os níveis de serviços propostos, penalizando a carga excessiva. A proposta dos autores baseia-se no cálculo de consumo, mas refere-se à plataforma como um serviço, sem monitorar a carga em toda a infraestrutura, nem avaliam os diferentes tipos de aplicações que podem concorrer por um mesmo recurso.

[Nanos et al. 2010] apresenta uma análise sobre o impacto das aplicações científicas que são executadas em um *cluster* virtualizado, baseando-se no impacto causado pelo uso intensivo de rede e de I/O. Os resultados e as conclusões dos autores apontam a necessidade de definir o perfil do comportamento das aplicações para melhor escaloná-las em ambientes de HPC virtualizados, afim de evitar a sobrecarga dos recursos computacionais.

O trabalho desenvolvido por [Juliani 2014] apresenta um método de escalonamento de máquinas virtuais em nuvens computacionais focadas em HPC. O escalonador desenvolvido pelo autor leva em consideração o consumo energético e o tipo de carga de trabalho que as máquinas virtuais irão executar para decidir quando e em qual servidor as mesmas serão alocadas. A avaliação do algoritmo foi feita através da utilização do *CloudSim*. Os resultados obtidos indicam a necessidade de analisar detalhes específicos das infraestruturas e aplicações para contribuir na otimização dos recursos e, consequentemente, aumentar os níveis de oferta de serviço e reduzir os problemas causados pelo uso dos recursos de forma concorrente.

Todos os trabalhos anteriormente apresentados apontam para a lacuna ainda existente quanto à necessidade de aprofundar os estudos dos efeitos da concorrência pelo compartilhamento de um ambiente real por vários ambientes virtualizados lá hospedados. Mesmo assim, eles não citam a necessidade de fazer este estudo com a utilização do conceito de Afinidade entre as aplicações.

3. Avaliação Experimental da Afinidade entre Aplicações

As avaliações sobre os efeitos das concorrência são importantes para definir o modelo de escalonamento para uso em ambientes distribuídos. A análise do comportamento das aplicações reais e sintéticas em máquinas virtuais, executando em um ambiente controlado e com equipamentos homogêneos, permitiu verificar o efeito da concorrência e sobretudo a importância do conceito de afinidade. Isso possibilita estender esse modelo para ambientes reais de nuvens. A afinidade é caracterizada por um grau normalizado que define o nível de influência entre aplicações que compartilham recursos. E o perfil é uma análise dinâmica do comportamento de consumo de recursos das aplicações durante o seu ciclo de vida, tais como suas intensividades e seu histórico de consumo.

Foram utilizadas *benchmarks* e aplicações reais científicas, afim de simular um ambiente real e assim avaliar os diversos perfis de aplicações. No decorrer das execuções das aplicações, um histórico do perfil de uso de recursos foi obtido por meio do monitoramento. São utilizados como parâmetros para esta estratégia: uso de CPU, memória, I/0

de disco. Estas informações são utilizadas para obter o perfil das aplicações científicas em ambientes dedicados e compartilhados. Vale frisar que classes de aplicações e perfil são conceitos distintos. Uma classe se difere de perfil uma vez que aplicações de uma mesma classe podem possuir perfis/comportamentos diferentes.

As aplicações utilizadas neste trabalho foram escolhidas por apresentarem perfis distintos de uso de recursos computacionais: HPL (*High-Performance Linpack Benchmark*) é uma aplicação sintética intensiva em CPU e que pode ser intensiva em memória, dependendo do tamanho da matriz de entrada; IOzone é uma aplicação sintética intensiva em I/O de disco que realiza operações sobre um sistema de arquivos; BLAST (*Basic Local Alignment Search Tool*) é uma aplicação real utilizada na área da Biologia e que apresenta uso intensivo de memória; e Montage (*Image Mosaic Software for Astronomers*) é uma aplicação científica utilizada na área da Astronomia cujo perfil de uso de recursos apresenta variações na intensidade de uso ao longo do tempo, o que valida a hipótese de que as aplicações podem sofrer alterações no decorrer da execução.

Para realização dos experimentos, foi utilizado KVM como camada de virtualização, e as seguintes configurações de infraestrutura: 3 servidores reais com Processador Intel(R) CPU X5650 2.67 GHz (12 núcleos), 16 GB de memória RAM, HD de 1TB (7200RPM), Sistema Operacional *Ubuntu Server* 14.04 LTS, rede Gigabit Ethernet. Em cada servidor real são alocados no máximo 2 ambientes virtuais. Isso tem por finalidade avaliar a afinidade entre duas máquinas virtuais concorrendo por recursos reais. Essas máquinas virtuais foram configurados da seguinte maneira: 4 núcleos QEMU Virtual, 6GB de memória RAM, HD Virtual de 20GB e Sistema Operacional *Ubuntu Server* 14.04 LTS. E os experimentos foram divididos em 2 grupos: experimentos executados em ambientes dedicado e ambientes compartilhados. Para cada experimento, foram realizados 30 execuções.

Nos experimentos em ambientes dedicados, todas as aplicações propostas foram executadas de modo isolado, sem qualquer outra aplicação sendo executada na mesma máquina física.

De acordo com [Patterson and Hennessy 2016], "Um computador que processa uma mesma quantidade de carga de trabalho em menos tempo é o mais rápido", definindo assim, tempo de execução como a melhor medida para definir o desempenho computacional. Diante de tal assertiva, o tempo é a medida escolhida para definir o grau da afinidade entre as aplicações executadas em ambientes virtuais, calculado pela Equação 1.

$$A_{i,j} = \frac{Tb_i}{Tc_{i,j}} \tag{1}$$

Nos experimentos em ambientes compartilhados, foram executadas todas as combinações de pares de aplicações, gerando a matriz de afinidades apresentada na tabela 1. Essa matriz é utilizada no escalonamento estático, que utiliza o conhecimento prévio para melhor alocar as aplicações a partir de uma fila de execução. A equação apresentada a cima é usada para obter o grau de afinidade entre as aplicações, a partir dos tempos de execução. A afinidade *A* entre as aplicações *i* e *j* é definida pela razão entre o tempo base (*Tb*) da aplicação *i* e o seu tempo de execução em concorrência (*Tc*) com a aplicação *j*.

Tabela 1. Matriz de afinidade entre as aplicações (quanto maior melhor)

	HPL	BLAST	IOZONE	MONTAGE
HPL	0,83	0,90	0,91	0,95
BLAST	0,85	0,91	0,86	0,91
IOZONE	0,93	0,96	0,51	0,83
MONTAGE	0,85	0,92	0,43	0,57

As aplicações utilizadas neste trabalho foram escolhidas por cada uma usar de forma intensiva um recurso computacional específico. Isso permitiu a simplificação da matriz de afinidade das aplicações para a criação de um nova matriz, mais genérica, baseando-se nos recursos computacionais CPU, Memória e I/O de disco, como pode ser visto na tabela 2. Essa matriz genérica é utilizada no escalonamento dinâmico, onde o perfil de uma aplicação pode não ser conhecido, fazendo com que o escalonador tenha que tomar decisões em tempo de execução.

Tabela 2. Matriz de afinidade baseado nos recursos computacionais

	CPU	Memória	I/O
CPU	0,83	0,90	0,91
Memória	0,85	0,91	0,86
I/O	0,93	0,96	0,51

Os valores obtidos estão normalizados de forma que $0 \leq a_{i,j} \leq 1$, tal que *a* representa um valor da matriz de afinidade de *i* com *j*. Quanto mais próximo *a* é de 1, melhor é o grau da afinidade. Em contrapartida, quanto menor e mais próximo do zero, pior é a afinidade de *i* com *j*.

4. Escalonador ProSched

O método de escalonamento desenvolvido é constituído por quatro serviços independentes. Cada serviço é responsável por desempenhar uma função específica no método de escalonamento (Figura 1). É importante ressaltar que, no contexto deste trabalho, uma tarefa é caracterizada por uma máquina virtual que executa uma ou mais aplicações.

O serviço `ProSched Web` é a interface através da qual a submissão de aplicações, gerenciamento e o acompanhamento, em tempo real, da alocação e utilização de recursos e tarefas são realizados pelos administradores.

O `Deployer` tem o objetivo de enviar as aplicações para a infraestrutura, atuando diretamente no gerenciamento das máquinas virtuais. Ele trabalha diretamente em conjunto com o escalonador, requisitando a máquina física mais apropriada para a execução de cada aplicação. Após essa comunicação, o `Deployer` inicia instâncias virtuais nos recursos reais ou realiza a migração em tempo real de máquinas virtuais cujos graus de afinidade sejam baixos (inferiores a 0.60). Este serviço também é responsável pela ativação dos monitores de cada máquina virtual da infraestrutura. Vale enfatizar que o patamar de afinidade 0.6 foi obtido de modo empírico através do histórico de execução das aplicações sendo possível ajustá-lo.

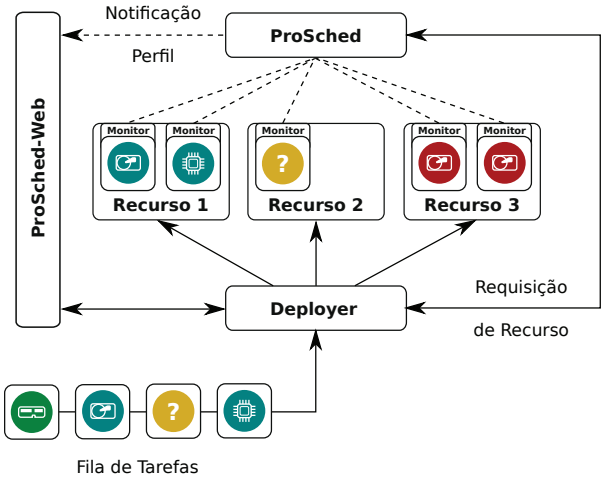


Figura 1. Arquitetura do Escalonador Baseado no Perfil das Aplicações

O serviço `Monitor` tem o objetivo de coletar e analisar dados sobre as tarefas durante a sua execução em uma máquina virtual. Para cada ambiente virtual, um agente monitor é alocado para coletar as informações de uso de recursos e, ao final, armazenar seu histórico em arquivo para ser utilizado posteriormente. Esta coleta é feita de forma não intrusiva, sem a necessidade de modificar o código das aplicações (Figura 2).

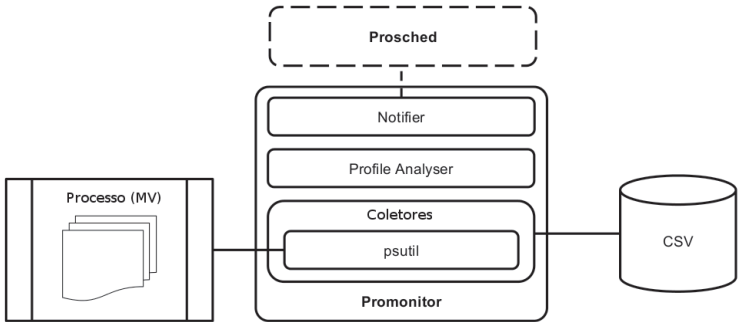
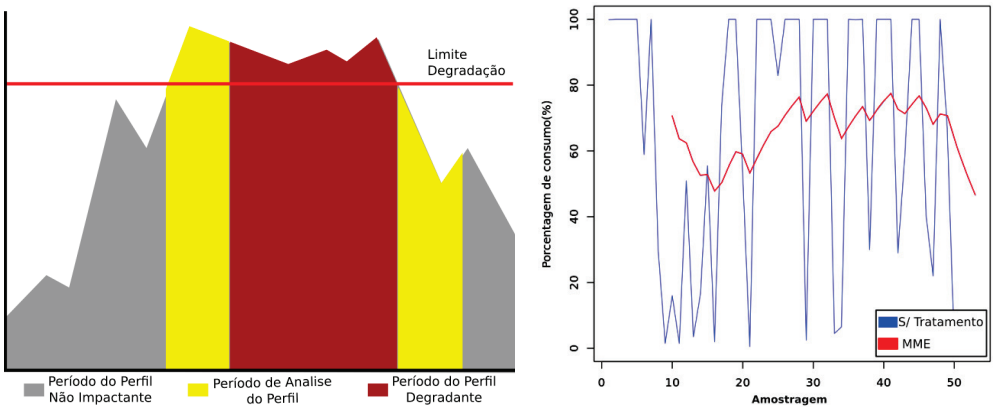


Figura 2. Arquitetura do Serviço Monitor

O perfil de execução das aplicações é obtido pelo monitoramento da tarefa. Uma vez que o valor coletado ultrapasse o limite de degradação do sistema, situação esta identificada pelo `Profile Analyser`, o monitor envia uma mensagem, através do módulo `Notifier`, sinalizando ao escalonador sobre a troca de perfil de consumo de recursos desta tarefa (Figura 3(a)). Durante todo o monitoramento da tarefa, os dados de interesse são coletados e armazenados em arquivos que serão utilizados pelo escalonador como conhecimento base em futuras execuções.

Quando analisado o gráfico de um histórico de consumo de CPU de uma tarefa, é possível verificar imediatamente dois aspectos: a variação do uso dos recursos formam picos e vales, e a existência de tendências ao longo do tempo. Uma solução encontrada para este problema foi a aplicação da Média Móvel Exponencial (MME) nos valores de consumo de recursos obtidos dos ambientes virtuais. Deste modo, os movimentos de curva são suavizados, permitindo uma representação real do comportamento das aplica-

ções. Isso permite que mudanças repentinas, e não constantes, não sejam erroneamente classificadas como o perfil atual da tarefa (Figura 3(b)).



(a) Processo de análise do perfil (b) Coleta com MME e sem tratamento

Figura 3. Método de análise de dados do Monitor

O próximo serviço a ser tratado é o `Prosched`. Sua principal contribuição é reduzir o *makespan*[1] de uma fila de tarefas. Tal contribuição é alcançada por meio do aprendizado de seus perfis dinâmicos com base em execuções anteriores. Em casos de mudança de perfil, o escalonador é capaz de alocar ou migrar a tarefa para outra máquina real na infraestrutura. Para isso, é feita uma análise para encontrar tarefas mais afins, garantindo a manutenção da capacidade de execução desse ambiente.

O comportamento do método de escalonamento desenvolvido une técnicas do algorítimo *Round-Robin* (RR), com afinidade de aplicações e perfil dinâmico de execução. Desta forma, a primeira etapa na alocação é encontrar o recurso com menor quantidade de tarefas. Caso seja encontrado, a tarefa é alocada no recurso disponível. Se os recursos possuírem ao menos uma tarefa, utiliza-se a afinidade de aplicações dada pela tabela 2.

Após uma tarefa ser inserida na fila de execução, o escalonador inicia o processo de eleição de recursos para a sua alocação. Os recursos são organizados de forma a simplificar uma alocação RR, ordenando-os de forma crescente pela quantidade de tarefas. Em paralelo, o escalonador procura em sua tabela de afinidade tarefas iguais e as agrega de forma a obter o perfil médio de execução. Para o conhecimento de execução desse perfil, o escalonador utiliza apenas execuções nas quais a tarefa não concorreu por recursos com outras aplicações. Para isso, os seguintes casos são analisados:

1. A tarefa possui afinidade e não está em execução: o recurso é alocado e o escalonador informa ao `Deployer` em qual *host* iniciar a tarefa. O `Deployer`, por sua vez, inicia a máquina virtual e o monitor para aquela tarefa;
2. A tarefa possui afinidade e está em execução: neste caso, o escalonador apenas registra o perfil de execução da tarefa, sem atuar sobre o sistema;
3. A tarefa não possui afinidade, mas está em execução: o escalonador avalia a tarefa e, com base no perfil médio de execução e se o tempo de execução seja maior do que o tempo de migração, o escalonador requisita ao `Deployer` que realize

[1]Intervalo de tempo entre a alocação da primeira tarefa até o fim da execução da última [Pinedo 2008]

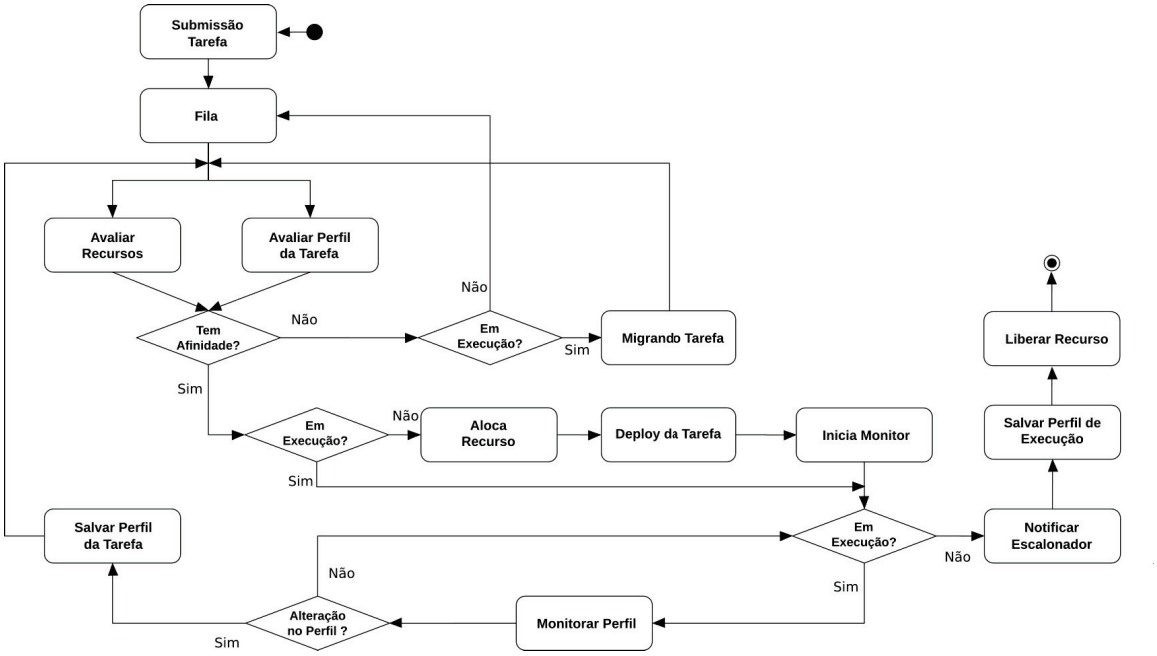


Figura 4. Fluxograma do Algoritmo de Escalonamento Prosched

a *Live-Migration* da máquina virtual para um recurso que a tarefa possua maior afinidade. Caso contrário, não é feita a migração;

4. A tarefa não possui afinidade e não está em execução: O escalonador a coloca na fila para que seja reavaliada durante o processo de notificação dos monitores.

Durante o ciclo de vida da tarefa, o `Monitor` coleta as informações e notifica o escalonador caso uma mudança de perfil seja detectada. Quando a tarefa termina sua execução, o escalonador armazena o perfil na tabela de conhecimento e encerra sua execução, informando que o recurso foi liberado. Na Figura 4 demonstra as etapas do funcionamento do algoritmo de escalonamento desenvolvido.

5. Validação e Resultados

Esta seção tem por objetivo apresentar os resultados do algoritmo de escalonamento estático e dinâmico, bem como validar a estratégia de alocação baseada na afinidade entre as aplicações e entre os recursos computacionais. Para avaliar o desempenho do escalonamento estático, este foi comparado com as estratégias FCFS (*First-Come First-Served*), o *Random*, na forma de uma alocação aleatória, e a estratégia proposta por este trabalho, denominada *Affinity*. Para validação do algoritmo dinâmico, será aplicada uma comparação entre os algoritmo *Round-Robin Affinity*, com e sem conhecimento dos perfis das aplicações, e aplicando sobre uma fila, um conhecimento híbrido, mesclando essas duas possibilidades quanto ao conhecimento dos perfis. As abordagens estática e dinâmica foram adotadas em função dos tipos de escalonamento online e offline.

Com o intuito de validar a hipótese de que a alocação baseada na afinidade pode minimizar o *makespan* e, consequentemente, otimizar o uso dos recursos computacionais, quando comparada com demais abordagens, utilizou-se no experimento estático apenas um servidor real para execução de uma fila de aplicações. Essa configuração teve por finalidade demostrar que, no pior caso, é possível a redução do tempo de uma fila de

execução. O ganho utilizando apenas 1 servidor indica que é possível obter ganhos com mais de um. É importante lembrar que o estudo da afinidade aqui proposto visa avaliar o impacto entre duas máquinas virtuais concorrendo por recursos computacionais em um mesmo hospedeiro. Por isso em ambos os experimentos, são sempre alocadas duas máquinas virtuais, que estarão executando uma ou mais aplicações.

A ordem da fila de execução (Figura 5) é definida pelas seguintes aplicações: BLAST, HPL, IOzone, IOzone, HPL, HPL, Montage, HPL, Montage e IOzone. Esta fila foi criada com o objetivo de analisar os resultados do escalonamento estático envolvendo aplicações com baixo grau de afinidade, de acordo com a Tabela 1.

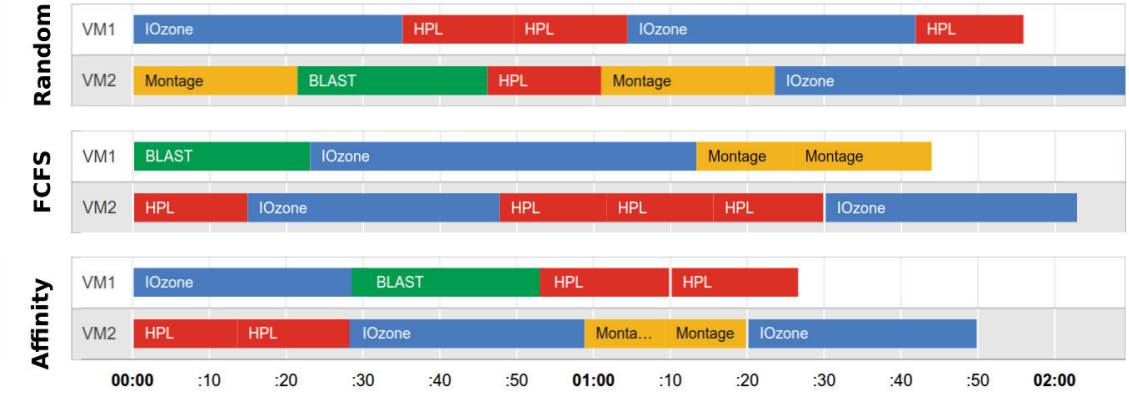


Figura 5. *Timeline* da fila, em horas, da execução de cada estratégia: FCFS, Affinity e Random

A estratégia *Random* foi aquela que resultou no pior desempenho, com um *makespan* médio de 129,5 minutos. O aumento do tempo de execução está relacionado ao fato de que as primeiras aplicações a serem alocadas concorrentemente foram o IOzone e o Montage. A matriz de afinidade mostra um baixo grau de afinidade entre essas aplicações. Com a análise do *timeline* do algoritmo *Random* (Figura 5), é possível verificar que, durante a execução do IOzone, um outro IOzone é escalonado para concorrer pelos mesmo recursos, enquanto que a aplicação seguinte ao IOzone é o HPL. Se o conhecimento do comportamento dessas aplicações tivesse sido utilizado, o HPL poderia ter sido executado antes do IOzone, o que garantiria um melhor uso dos recursos computacionais, alcançando-se também a redução do tempo de execução.

Ao analisar o tempo de cada abordagem (Figura 5), a estratégia baseada na afinidade entre aplicações obteve um *makespan* médio de 110,9 minutos. A redução de tempo está relacionada ao uso do conhecimento prévio para evitar alocações de aplicações com baixo grau de afinidade, o que não ocorreu nos outros algoritmos. Isso foi possível ao evitar que aplicações intensivas em I/O fossem alocadas concorrentemente.

Em resumo, a alocação baseado na afinidade conseguiu reduzir o tempo do *makespan* entre as abordagens FCFS e Random em, aproximadamente, 13,3 e 18,6 minutos, respectivamente. Os resultados constatam que a estratégia proposta obteve um ganho de desempenho de até 16,7%, o que comprova a eficiência do algoritmo.

O experimento cujo resultado é ilustrado pela Figura 6 comprova que o escalonador dinâmico tem a capacidade de evitar a alocação de aplicações com baixo grau de afinidade em situações nas quais os perfis sejam definidos. O experimento visa também

mostrar como o escalonador atua em conjunto com o `Monitor`, quando não há nenhum conhecimento sobre as aplicações, sendo necessário migrá-las através do *live-migration*.

Para os experimentos, foram elaborados perfis de aplicações a serem executados, criados a partir da combinação das aplicações estudas neste trabalho. O objetivo da elaboração desses perfis é validar o emprego, em conjunto, do monitoramento e do escalonamento, visto que as aplicações possuem perfis distintos e, com isso, é mais flexível verificar seu comportamento ao longo das execuções. Isso permite validar a política de escalonamento proposta. Os experimentos foram organizados da seguinte maneira: são enviadas 6 conjuntos de perfis de aplicações para o escalonador, chegando em momentos distintos de tempo, uma após a outra, como acontece em um cenário comercial real. O conjunto de perfis de aplicações elaboradas para o escalonamento são: A {`HPL, IOZONE`}, B {`BLAST, HPL`}, C {`HPL`}, D {`IOZONE`}, E {`HPL, IOZONE`} e F {`BLAST, HPL`};

A fila a ser executada dinamicamente pelo escalonador segue a seguinte ordem de chegada dos Perfis: A, B, D, E, C, F. A formação dessa fila de execução tem por objetivo explorar o pior caso para o algoritmo de escalonamento proposto, por coincidir a alocação concorrente de duas aplicações conflitantes, identificadas nos resultados da análise do efeito da concorrência.

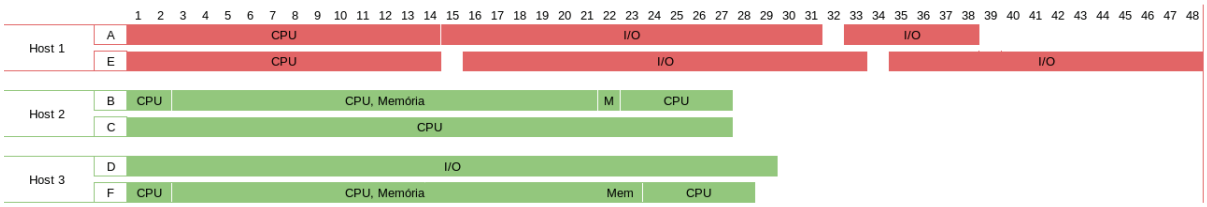


Figura 6. Perfil de execução das tarefas utilizando o método *Round-Robin*(RR)

No primeiro experimento, denominado RR Base, é empregado a estratégia de alocação *Round-Robin*. Essa estratégia de alocação é comumente utilizada por sistemas de nuvens computacionais, como por exemplo, o *OpenStack* [Openstack 2016]. O objetivo deste experimento é comparar essa estratégia com a desenvolvida neste trabalho, que otimiza as alocações posteriores com base no perfil de consumo das aplicações, bem como na migração do ambiente quando o grau de afinidade for baixo.

Através da análise da Figura 6, é possível perceber que o maior impacto foi ocasionado no *Host 1*. Inicialmente, as aplicações eram intensivas em CPU. Entretanto, após aproximadamente 14 minutos, ambas as aplicações trocaram seu perfil de consumo de recursos e passaram a ser intensivas em I/O. Devido a isso, as aplicações acabam competindo pelo mesmo recurso que demonstrou ser o mais crítico em relação à concorrência. O compartilhamento de I/O pelas aplicações causa uma degradação de aproximadamente 50% na execução deste perfil, obtendo assim, um *makespan* de 48 minutos.

A Figura 7 ilustra o experimento no qual são descobertos os perfis das aplicações em tempo real, através do monitoramento e da identificação pelo Monitor. O problema identificado no experimento base (Figura 6) é solucionado pelo escalonador proposto, através da migração dos ambientes virtuais conflitantes. Na Figura 7, é possível perceber que o momento de impacto foi identificado quando o processo de balanceamento de carga é iniciado. Esta abordagem obteve um *makespan* de 41 minutos, 7 minutos a menos do

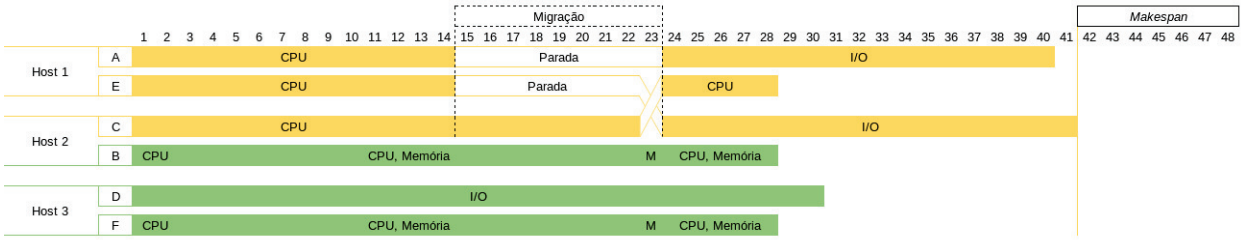


Figura 7. Perfil de execução das tarefas utilizando o método proposto, sem o conhecimento das aplicações (ASC)

que para o RR Base.

O experimento ilustrado pela Figura 8 aloca as aplicações de acordo com os perfis obtidos previamente. Isso permite que a utilização dos recursos seja otimizada nos recursos computacionais, respeitando a matriz de afinidade como base(Tabela 2).

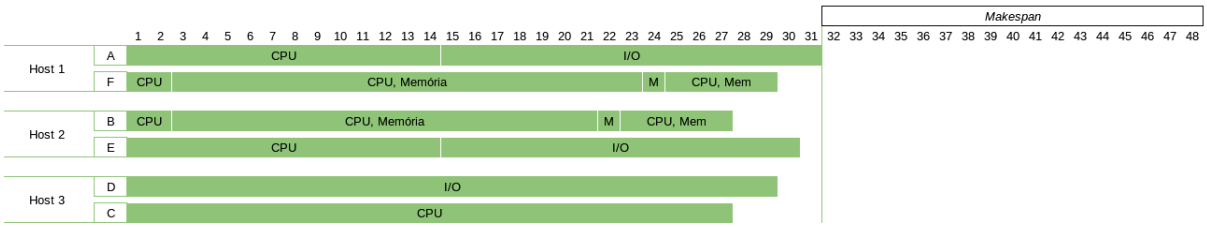


Figura 8. Perfil de execução das tarefas utilizando o método proposto com o conhecimento das aplicações (ACC)

Na figura 8, é possível verificar que, a partir da alocação inicial das 3 primeiras tarefas, inicia-se o uso da matriz de afinidade. É nesse momento que o escalonador verifica o perfil e o histórico de consumo de recursos de cada aplicação. Por exemplo, quando o `Perfil E` é recebido pelo escalonador, verifica-se a existência de uma aplicação com consumo intensivo em I/O. Por esse motivo, os *Hosts* 1 e 3 foram considerados inelegíveis para recebimento de tal tarefa, o que fez com que o escalonador optasse pelo *Host* 2. Em seguida, a próxima aplicação recebida é aquela classificada com o `Perfil C`, que apresenta consumo intensivo de CPU. De acordo com a matriz de afinidade, CPU tem grau de afinidade com I/O de 0,91, razão pela qual o escalonador aloca esta aplicação concorrendo com o `Perfil D` no *Host* 3. As últimas aplicações a serem escalonadas (`Perfil F`) possuem intensividade de CPU e de Memória. Com isso, ela é alocada concorrendo com I/O, devido ao grau de afinidade com aplicações deste tipo ser de 0,86.

O algoritmo proposto permitiu aliar o estudo do impacto da concorrência entre aplicações, em conjunto com o conhecimento sobre os seus perfis. A adoção do algoritmo resultou na redução do *makespan* de aproximadamente 31 minutos, sendo, em média, 54% mais rápido do que o Experimento 1(RR), e 38% quando comparado ao algoritmo com uso da afinidade sem conhecimento das aplicações, apresentado no Experimento 2 (ASC).

Os resultados destes experimentos permitiram demonstrar o ganho de tempo quando utiliza-se perfis de aplicações. Além disso, conforme as aplicações são repetidamente executadas, mais refinado fica o seu perfil, o que permite aumentar a qualidade

do escalonamento em alocações futuras.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi desenvolvido e apresentado um algoritmo de escalonamento de máquinas virtuais, com o objetivo de utilizar o conceito de afinidades entre as aplicações, a fim de aumentar a taxa de uso dos recursos computacionais. Para tanto, foi desenvolvido um sistema de monitoramento de máquinas virtuais, não intrusivo, que analisa e define os diversos perfis das aplicações. O monitor utiliza um analisador de média móvel exponencial que evita a notificação errônea da variação dos perfis da aplicação.

Uma vez que três das aplicações estudadas apresentam uso intensivo de recursos distintos, foi possível criar uma generalização da matriz de afinidades entre aplicações para uma matriz de afinidade entre recursos que, por sua vez, é adotada pela estratégia de escalonamento dinâmico desenvolvida.

Os resultados comprovam os benefícios de utilizar a estratégia de escalonamento *Affinity*, que explora a concorrência entre as aplicações que possuem maior grau de afinidade. Os resultados do uso do modelo associado ao escalonamento estático, proporcionaram um ganho de até 16,7% de desempenho em relação às outras implementações, equivalente à redução de aproximadamente uma hora no *makespan*.

Os resultados obtidos no experimento de alocação dinâmica comprovam a eficiência do escalonador desenvolvido. Foi possível agregar o estudo sobre afinidade das aplicações ao sistema de monitoramento, para identificar uma tarefa que altera o seu consumo de recursos computacionais e impacta negativamente sobre a infraestrutura. Além disso, permitiu a identificação dos perfis em tempo real, além da analise do histórico de consumo, otimizando as alocações e consequentemente, o uso dos recursos computacionais. Um outro ponto que merece destaque é o sistema que identifica e migra os ambientes virtuais quando uma aplicação com baixo grau de afinidade é identificada. Esse fator, aliado à alocação eficiente, permitiram ao escalonador proposto obter melhor uso dos recursos, além de reduzir o tempo que uma aplicação espera na fila até ser executada. Foi possível constatar uma redução do *makespan* de, respectivamente, 17% (7 minutos) do ASC e de 54,8% (17 minutos) do ACC, respectivamente, em relação ao escalonamento RR Base.

Com o desenvolvimento apresentado neste artigo, foram identificados tópicos que podem ser explorados na forma de trabalhos futuros. São eles: i) aperfeiçoar o método de escalonamento para que sua política de alocação trabalhe com atribuição de prioridades, de modo que uma aplicação crítica seja executada com antecedência; ii) avaliar o grau de afinidade entre três ou mais máquinas virtuais em um mesmo servidor físico; iii) analisar o perfil das aplicações e o impacto sobre a afinidade entre aplicações em ambientes com arquiteturas heterogêneas; iv) avaliar o impacto da concorrência por recursos de rede sobre o modelo de escalonamento proposto, em especial, no que se refere à migração das máquinas virtuais.

Referências

Alam, M. and Varshney, A. K. (2016). A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System. *International Journal of Applied Evolutionary Computation (IJAEC)*, 7(2):61–75.

Bernado, E., P. W. P. R. (2014). Arquitetura para Suportar Sobrecargas Momentâneas em Ambientes de Computação em Nuvem. (dissertação de mestrado), Instituto Militar de Engenharia (IME).

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience*, 41(1):23–50.

Emani, M. K. and O'Boyle, M. (2015). Celebrating Diversity: A Mixture of Experts Approach for Runtime Mapping in Dynamic Environments. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2015, pages 499–508, New York, NY, USA. ACM.

Juliani, F. (2014). Um Método de Escalonamento Baseado no Comportamento de Aplicações HPC para Nuvens Computacionais Balanceando Desempenho e Eficiência Energética. (dissertação de mestrado), Instituto Militar de Engenharia.

Licht, F. L. (2014). *Afinidade de Tipos de Aplicações em Nuvens Computacionais*. PhD thesis, Universidade Federal do Parana, Departamento de Informatica, Curitiba, PR.

Mury, A. R., Schulze, B., Licht, F. L., de Bona, L. C., and Ferro, M. (2014). A Concurrency Mitigation Proposal for Sharing Environments: An Affinity Approach Based on Applications Classes. In *Intelligent Cloud Computing*, pages 26–45. Springer.

Nanos, A., Goumas, G., and Koziris, N. (2010). Exploring I/O Virtualization Data Paths for MPI Applications in a Cluster of VMs: a Networking Perspective. In *European Conference on Parallel Processing*, pages 665–671. Springer.

Openstack (2016). Openstack Documentation Review Associate VM Placement. `http://docs.openstack.org/icehouse/training-guides/content/operator-computer-node.html`. Accessed: 2016-11-28.

Patterson, D. and Hennessy, J. (2016). *Computer Organization and Design: The Hardware Software Interface: ARM Edition*. The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition.

Schad, J., Dittrich, J., and Quiané-Ruiz, J.-A. (2010). Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *Proceedings of the VLDB Endowment*, 3(1-2):460–471.

Simmons, B., McCloskey, A., and Lutfiyya, H. (2007). Dynamic Provisioning of Resources in Data Centers. In *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*, pages 40–40. IEEE.

Yokoyama, D. (2015). Modelo para o Escalonamento de Aplicações Científicas em Ambientes de Nuvens Baseado em Afinidades. Dissertação de Mestrado, Laboratório Nacional de Computação Científica.

Zheng, Z., Wu, X., Zhang, Y., Lyu, M. R., and Wang, J. (2013). QoS Ranking Prediction for Cloud Services. *IEEE Trans. Parallel Distrib. Syst.*, 24(6):1213–1222.