Uma Abordagem Hierárquica para Escalonamento de *Workflows* Científicos Executados em Nuvens *

Igor Barreto Rodrigues[1], **Daniel de Oliveira**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (IC/UFF)

{ibarreto, danielcmo}@ic.uff.br

Resumo. *Os Workflows científicos são utilizados para modelar experimentos baseados em simulação onde cada simulação é composta por diversos programas que possuem dependência de dados entre si. Muitas dessas simulações necessitam executar em ambientes de alto desempenho para terminar em tempo hábil. Executar workflows nesses ambientes envolve escalonar atividades dos workflows em diferentes recursos. Muitos escalonadores tradicionais buscam otimizar apenas o tempo de execução, porém em cenários mais reais existem vários fatores a serem considerados, e.g. consumo de energia e custo financeiro. Algumas abordagens recentes já levam em consideração múltiplos objetivos, porém muitas delas utilizam funções objetivo ponderadas que podem se tornar um problema quando muitos objetivos são considerados. Esse artigo propõe uma abordagem que usa múltiplas funções objetivo de forma hierárquica, onde os múltiplos objetivos são levados em consideração em uma determinada ordem estabelecida pelo cientista. Foram executados experimentos em workflows sintéticos e reais na nuvem da Amazon EC2. Os resultados obtidos apresentaram melhoras no objetivo escolhido para otimização.*

1. Introdução

Experimentos *in silico* são aqueles em que modelamos fenômenos reais por meio de simulações computacionais [Deelman et al. 2009]. Esses experimentos são comumente compostos por diversos programas que possuem uma dependência de dados entre si. Esse conjunto de programas e as restrições de precedência são conhecidos como *workflows* científicos e são um paradigma popular usado por cientistas para a modelagem de simulações computacionais. Cada atividade do *workflow* está associada a um programa de computador, e cada execução de uma atividade é chamada de ativação. Cada ativação processa um conjunto de dados de entrada e produz um conjunto de dados de saída [Ogasawara et al. 2011]. Os *workflows* são gerenciados por sistemas complexos chamados de Sistemas de Gerência de *Workflows* Científicos (*i.e.* SGWfC). Muitos *workflows* consomem e produzem um grande volume de dados e necessitam de ambientes de processamento de alto desempenho (PAD), *e.g.* nuvens de computadores [Vaquero et al. 2008].

A nuvem é capaz de fornecer um ambiente favorável para a execução de *workflows*, pois além da disponibilidade, os cientistas podem explorar a elasticidade, *i.e.* a capacidade de aumentar e/ou diminuir a quantidade de máquinas virtuais (VMs)

*Este artigo foi financiado parcialmente pelo CNPq e FAPERJ

criadas para executar o *workflow*. Na nuvem, o cientista paga apenas pelos recursos que consumir *(pay-per-use)* [Vaquero et al. 2008]. Para que a nuvem possa ser utilizada em sua capacidade máxima, os SGWfCs existentes devem ser capazes de distribuir as execuções dos *workflows* nestes ambientes. Existem diversos SGWfCs que apoiam a execução em nuvens como o Pegasus [Deelman et al. 2007], o Swift/T [Zhao et al. 2007] e o SciCumulus [Oliveira et al. 2012], que foi o SGWfC utilizado neste artigo, pois realiza o escalonamento de *workflows* baseado em múltiplos objetivos.

O SciCumulus [de Oliveira 2012] é um SGWfC adaptativo em relação ao estado dos recursos computacionais na nuvem. Ele é capaz de verificar a capacidade computacional e ajustar dinamicamente a distribuição das ativações e a capacidade computacional do ambiente a fim de alcançar um melhor desempenho. Seu escalonador atual considera 3 objetivos: o tempo de execução, o custo financeiro e a confiabilidade. Embora o SciCumulus seja capaz de otimizar o escalonamento baseado em um modelo de custo com três objetivos, a prioridade dos mesmos é definida pelo cientista de forma ponderada, o que pode gerar 2 problemas: (i) ao se executar um *workflow* pela primeira vez, o cientista não tem ideia alguma de como o mesmo se comportará baseado nos pesos de entrada, somente após várias execuções e ajustes é que o cientista consegue obter um resultado mais próximo do que deseja e (ii) a medida que adicionamos mais objetivos ao modelo (*e.g.* o consumo de energia), mais difícil se torna a função objetivo definir o melhor escalonamento [Oliveira and Saramago 2010].

Neste artigo, ao invés de utilizarmos uma função multi-objetivo ponderada, propomos o uso de múltiplas funções objetivo de forma hierárquica para otimizar múltiplos critérios do cientista, fazendo um balanceamento entre as restrições de custo financeiro, tempo de execução, tolerância a falhas e o custo energético. A abordagem proposta não utiliza mais pesos associados aos objetivos, e o cientista precisa somente definir a ordem de prioridade dos objetivos, montando portanto uma hierarquia. A avaliação experimental foi realizada comparando a versão atual do escalonador do SciCumulus, onde a função objetivo é ponderada e abordagem proposta. Os resultados apresentaram uma melhora considerável de desempenho no escalonador do SGWfC, maximizando os ganhos definidos pela ordem de prioridade do cientista. Foram realizados avaliações com um *workflow* sintético e com um *workflow* real da área de bioinformática.

O restante deste artigo está estruturado da seguinte maneira. A Seção 2 apresenta trabalhos relacionados. A Seção 3 apresenta o formalismo utilizado. A Seção 4 apresenta a abordagem proposta. A Seção 5 aborda a avaliação experimental, e, finalmente, a Seção 6 conclui o artigo e discute sobre trabalhos futuros.

2. Trabalhos Relacionados

O escalonamento de atividades de *workflows* é um conhecido problema NP-difícil. Apesar disso, a maioria dos trabalhos busca soluções para otimizar apenas 1 ou 2 objetivos, porém na maioria dos casos são utilizados formalismos que agregam todos os objetivos em uma única função analítica, utilizando pesos para restringir alguns objetivos e maximizar outros. Garg *et al.* [Garg et al. 2009] utilizam um escalonamento que otimiza o *makespan* e a confiabilidade, por meio de um vetor de pesos. Assayad *et al.* [Assayad et al. 2004] e Hakem *et al.* [Hakem and Butelle 2007] utilizam abordagens similares para otimizar confiabilidade e tempo. A principal desvantagem dessas

abordagens é que a solução depende de uma combinação de pesos entre os objetivos para tentar atender as necessidades do cientistas, que podem não conhecer como o *workflow* se comporta com determinados dados de entrada.

Sakellariou *et al.* [Sakellariou et al. 2007] que otimizam somente 2 objetivos e Fard *et al.* [Fard et al. 2012] que otimizam 4 objetivos (*makespan*, consumo de energia, custo monetário, confiabilidade) calculam a solução otimizando um objetivo e tentando não violar as restrições dos outros baseados nos requisitos do cientista. Essas abordagens foram propostas baseadas em *grids* computacionais e não em nuvens. Yu *et al.* [Yu et al. 2007] utilizam algoritmos genéticos para otimizar o *makespan* e o custo financeiro.

Pandey *et al.* [Pandey et al. 2010] utilizam uma heurística baseada em otimização de enxame de partículas para escalonamento em nuvens levando em consideração custo financeiro. A principal desvantagem é que sua abordagem demora muito a convergir, gerando um alto custo computacional. Além disso, não visa outros objetivos como, por exemplo, o tempo de execução. Duan *et al.* [Duan et al. 2014] geram o escalonamento otimizando tempo de execução e custo financeiro, porém com opções de restrição de largura de banda e armazenamento, utilizando um algoritmo baseado em teoria de jogos.

Diferentemente dos trabalhos relacionados, a abordagem hierárquica proposta nesse artigo se utiliza de múltiplas funções objetivo e atende a várias restrições escolhidas pelo usuário. Porém, não atribuímos pesos a cada objetivo a ser otimizado [Oliveira et al. 2012], ou tentamos otimizar determinada função e não violar as restrições das demais [Sakellariou et al. 2007]. A abordagem aqui proposta estende a proposta de Durillo *et al.* [Durillo et al. 2015], que busca encontrar um conjunto de soluções para cada VM que está envolvida na execução do *workflow*.

3. Formalismo Utilizado

A execução de um *workflow* pode ser longa, e, apesar do cientista geralmente desejar obter os resultados no menor tempo possível (o que o levaria a executar o *workflow* em máquinas mais robustas e mais caras), nem sempre ele possuirá valores financeiros suficientes para isso. Logo, existem métricas que podem ser escolhidas no momento da execução de forma a balancear os objetivos. Tais métricas podem ser usadas para definir um modelo de custo a ser utilizado para definir quais as melhores ativações para determinadas máquinas virtuais. Esse tipo de modelo de custo já é utilizado por SGWfCs como o SciCumulus [Oliveira et al. 2012]. No entanto, existe um objetivo que é adicionado ao modelo de custo nesse artigo que o SciCumulus não atendia até o momento: o custo energético. O formalismo aqui exposto, bem como cada função objetivo, é uma extensão do formalismo apresentado por de Oliveira [Oliveira et al. 2012].

Consideremos $VM = \{VM_0, VM_1, ..., VM_{n-1}\}$ como o conjunto de *n* VMs criadas na nuvem e disponíveis para execução do *workflow*. Para cada VM, existe um valor associado a ela chamado de índice de retardo de computação *(csi)* (*computational slowdown index*) [de Oliveira 2012]. Neste artigo o valor de *csi* é obtido com base nos dados de proveniência das execuções anteriores do *workflow* [Freire et al. 2008], ou seja, é calculada uma média baseada nos tempos de execuções passadas de ativações dos *workflows*. Desta forma, a uma VM com desempenho mínimo será atribuído o valor 1 e as restantes com valores relativos *(e.g. 1,72)*. Este índice também pode ser obtido

através da função *csi(*VM_j*)*.

Consideremos *P(*ac_i, VM_j*)* como o tempo de execução previsto de uma ativação ac_i em uma VM_j na nuvem. Da mesma forma, *P(*ac_i, VM_j*)* $= tempo(ac_i) \times csi(VM_j)$, em que $tempo(ac_i)$ é uma função que retorna o tempo médio normalizado da execução de uma ativação, ou seja, calcula a média do tempo das execuções anteriores dividido pelo valor de *csi* das VMs onde as ativações foram executadas. Considere também $\varphi(W, VM)$ como sendo o escalonamento total das ativações pertencentes ao conjunto *AC* do *workflow W* em *VM*. Dado um *workflow W* que possui uma série de atividades $A = \{a_1, a_2, ..., a_n\}$ e um conjunto de ativações $AC = \{ac_1, ac_2, ..., ac_k\}$, seja $\varphi(W, VM) = \{sched_1, sched_2, ..., sched_k\}$ o conjunto dos escalonamentos para execução paralela do conjunto de ativações *AC*. O escalonamento de uma ativação é definido como *sched(*$ac_i, VM_j, inicio, fim$*)*, em que *início* é o tempo de início de execução de uma ativação ac_i em uma dada VM_j e *fim* o tempo fim. Conforme definido por [Oliveira et al. 2012], são definidos diversos escalonamentos *(sched)* ao longo da execução do *workflow*. Portanto é definido que *ord(*$sched_i$*)* é a posição do $sched_i$ na ordem de sequência de todos os escalonamentos realizados, logo, temos $sched_i < sched_j \leftrightarrow ord(sched_i) < ord(sched_j)$.

A transferência de dados também é um custo importante e que deve ser contabilizado no custo total da execução do *workflow*. Como o SciCumulus lê e escreve em um *storage* compartilhado no Amazon S3 [Oliveira et al. 2012], todos os escalonamentos possíveis em $\varphi(W, VM)$ leem e escrevem o mesmo volume de dados, logo, o custo sempre será equivalente independente do escalonamento. Considere ac_i como uma ativação associada a uma atividade $a_i(Act(ac_i) = a_i)$ de um *workflow* W com tempo de execução definido em $tempo(ac_i)$ e com $sched(ac_i, VM_k)$ já definido. Agora considere uma ativação ac_j associada a atividade $a_j(Act(ac_j) = a_j)$ com dependência $DepAc(ac_i, ac_j, volume(ac_i, ac_j))$ e $sched(ac_j, VM_x)$, onde $volume(ac_i, ac_j)$ é a quantidade de dados transferidos entre ac_i e ac_j. Como $VM_k \neq VM_x$, é necessário incluir uma nova dependência de dados e uma ativação artificial em *W* para representar a transferência de dados entre ac_i e ac_j. Portanto, vamos substituir $DepAc(ac_i, ac_j, volume(ac_i, ac_j))$ por $DepAc(ac_i, ac_w, volume(ac_i, ac_j))$ e $DepAc(ac_w, ac_j, volume(ac_i, ac_j))$, em que ac_w é uma ativação artificial criada para representar a transferência. Logo teremos o tempo de execução da ativação ac_w:

$$tempo(ac_w) = \frac{volume(ac_i, ac_j)}{\min(network(VM_k), network(VM_x))} \quad (1)$$

Podemos então calcular o tempo total de transferência de dados *(TT)* para todas as ativações de *W*, assumindo que existam *k* atividades e portanto *k-1* transferências:

$$TT = \sum_{w=0}^{k-1} time(ac_w) \quad (2)$$

A confiabilidade é um critério muito importante a ser considerado pelo cientista, uma vez que a nuvem pode apresentar flutuações de desempenho e até mesmo falhas *(i.e máquina reinicia, desliga, muda de servidor)*. Dessa forma, definimos o chamado *custo de confiabilidade*. Esse custo define o impacto que uma falha pode gerar na execução

do *workflow*, *i.e.* se o custo for alto, há chance da execução ser altamente impactada por falhas. A taxa de falha de uma VM pode ser obtido como $F(VM_j), \forall\ VM_j \in VM$. Esse valor pode ser obtido com base nos dados de proveniência armazenados sobre um tipo de VM_j. O custo de confiabilidade é inversamente proporcional a confiabilidade da execução do *workflow* na VM. Esse custo representa o número médio de falhas que ocorrem durante um período de execução da ativação. Se o custo de confiabilidade for minimizado, aumentamos a confiabilidade da execução como um todo, dessa forma, o custo de confiabilidade de uma determinada ativação executar em uma máquina é representado por $R(ac_i, VM_j)$ e pode ser definido como:

$$R(ac_i, VM_j) = F(VM_j) \ \times \ P(ac_i, VM_j) \tag{3}$$

O consumo energético é uma preocupação recente que vem sendo agregada aos ***SGWfC***, buscando atender não só as necessidades do cientista, mas também a questões ambientais [Durillo et al. 2014, Fard et al. 2012]. Assim, a métrica de consumo energético foi acoplada ao escalonador do SciCumulus. Considere $CPUCons(VM_j)$ como uma função que retorna o consumo da CPU por hora *(watts/hora)* de uma determinada VM_j, analogamente a função $RAMCons(VM_j)$ retorna o consumo da memória RAM de uma VM_j em *(watts/hora)*. As informações sobre a ***CPU*** e ***RAM*** de cada máquina são fornecidas pelo próprio sítio da Amazon EC2, e com a referência de cada item é possível verificar o consumo energético no sítio do fabricante, que no caso é a Intel[1]. Consideremos então que:

$$isSched(ac_i, VM_j) = \begin{cases} 1\ , se\ \ sched(ac_i, VM_j) \\ 0\ , caso\ \ contrário \end{cases}$$

Com isso, podemos calcular o consumo energético total a partir da soma de todos os tempos das ativações pertencentes a todas as atividades escalonadas nas VMs do *workflow W*. Assim, a Equação 5 calcula o consumo energético total $E(\varphi)$ de cada VM_j durante o processamento de todas as tarefas escalonadas para a mesma:

$$E(\varphi) = \sum_{i=0} \sum_{j=0} CPUCons(VM_j) \times P(ac_i, VM_j) \times isSched(ac_i, VM_j) + RAMCons(VM_j) \times P(ac_i, VM_j) \times isSched(ac_i, VM_j) \tag{4}$$

Assim como a confiabilidade, o custo financeiro deve ser considerado em execuções no ambiente de nuvem, pois o cientista "aluga" máquinas por um período de tempo, pagando somente pelos recursos que consumir [Vaquero et al. 2008], mas deve-se usar de forma consciente, do contrário a execução pode se tornar bastante custosa. Como seguimos o modelo de cobrança da Amazon EC2, e a mesma possui um esquema de precificação por janela de tempo utilizada (1 hora), independente se a máquina está ociosa ou processando. Consideremos o custo monetário de execução como $M(\varphi)$. O tempo de execução total deve ser a soma das várias janelas de tempo de tamanho ***TQ*** o que nos leva ao conjunto $TQ = \{tq_1, tq_2, ..., tq_x\}$ de janelas de tempo, onde denominamos o valor da janela de tempo a ser pago como ***Vq***(VM_j). Considere $Exec(tq_z, VM_j)$ uma função que retorne se a VM_j está executando uma ativação na janela de tempo δ_i, desta forma:

[1]http://www.intel.com.br/content/www/br/pt/homepage.html

$$Exec(tq_z, VM_j) = \begin{cases} 1\,, se\ VM_j\ roda\ ac_i/ac_i \in Ac\ em\ tq_z \\ 0\,, caso\ contrário \end{cases}$$

Como a função $Exec(tq_z, VM_j)$ só calcula o custo de uso da VM enquanto a mesma está processando, precisamos considerar também o tempo em que a mesma se encontra ociosa aguardando a transferência de dados. Portanto, o cálculo do custo monetário fica divido em duas partes: a primeira calcula o custo enquanto as máquinas estão processando as ativações, e a segunda parte calcula o tempo em que as máquinas aguardam o tempo de transferência dos dados:

$$M(\varphi) = (Vq(VM_j) \times \sum_{j=1}^{|VM|} \sum_{i=1}^{\frac{makespan}{|tq|}} Exec(tq_i, VM_j) + Vq(VM_j) \times \sum_{i=0}^{k} time(ac_j)) \quad (5)$$

Com isso obtemos o custo financeiro de cada VM, porém, ainda há outro fator a ser considerado, que é a unidade de informação transferida pela rede, pois os provedores de nuvem também cobram por isso. Dessa forma, temos o custo financeiro de transferência como $TR(\varphi)$. Considere como *Vt* o valor que o usuário tem de pagar por cada unidade de dados transferidos. Portanto o valor total a ser pago pela transferência de dados é:

$$TR(\varphi) = (Vt \times \sum_{j=0}^{k-1} \sum_{i=0}^{k-1} volume(ac_i, ac_j)) \quad (6)$$

4. Algoritmo de Escalonamento Proposto

O algoritmo de escalonamento proposto nesse artigo é uma variação do algoritmo de escalonamento guloso proposto por de Oliveira *et al.* [Oliveira et al. 2012] e que se encontra disponível para utilização no SGWfC SciCumulus [2]. No algoritmo original proposto por de Oliveira *et al.*, uma única função objetivo é utilizada para escolher a VM mais adequada para se executar uma determinada ativação, e o cientista define a importância de cada objetivo por meio de pesos. Entretanto, quanto mais objetivos são adicionados a função, menos eficiente a função se torna [Oliveira and Saramago 2010].

O algoritmo proposto nesse artigo procura gerar um escalonamento que represente um *trade-off* entre múltiplos critérios (custo financeiro, tempo de execução, confiabilidade e custo energético) e que utilize múltiplas funções objetivo hierarquizadas ao invés de se utilizar uma única função objetivo ponderada. No algoritmo proposto, sempre que uma VM se torna disponível para executar uma ativação, é escolhida a ativação liberada (que não depende de mais nenhuma ativação para executar) mais adequada para sua execução. Como entrada, o algoritmo recebe um conjunto de VMs, um conjunto de ativações e suas dependências de dados e a hierarquia de critérios. Inicialmente são definidas todas as ativações disponíveis, *i.e.* que não possuem nenhuma dependência de dados associada a outra ativação.

Suponhamos que o cientista escolha a seguinte lista de prioridades para os critérios: consumo de energia, custo financeiro, confiabilidade e tempo de execução,

[2]https://github.com/UFFeScience/SciCumulus

conforme apresentado na Figura 1. Seguindo essa ordem, será(ão) escolhida(s) uma ou mais ativações para uma determinada VM que minimizam o critério de consumo de energia, primeiro objetivo e calculado utilizando a Equação 5. A partir do melhor valor retornado pela função objetivo, são selecionados todos os escalonamentos cujo valor difira de no máximo *x%* do melhor valor retornado (*threshold*). Esses escalonamentos são considerados como ”empates” e, a partir daí, é utilizado o segundo critério para reduzir o tamanho da lista de escalonamentos. Essa ideia é similar a ordenação imposta por Sistemas de Gerência de Banco de Dados em consultas SQL que utilizam a cláusula *ORDER BY*, onde uma série de critérios de ordenação são definidos de maneira hierárquica. No caso do exemplo da Figura 1, o menor custo energético identificado foi 2.1. Para o *threshold* utilizado de 10%, duas ativações foram consideradas ”empatadas”: tanto a de custo 2.1 quanto a de custo 2.2. Assim as duas ativações são selecionadas e o algoritmo transita para o próximo critério que é custo financeiro. No critério de custo financeiro uma ativação tem custo 0.9 e a outra 2.7. Como o custo da segunda ativação supera o da primeira em mais de 10%, a primeira ativação é a escolhida para executar.

Uma restrição imposta no algoritmo proposto é que caso o mesmo alcance o último nível da hierarquia e ainda necessite transitar para outro critério pois os escalonamentos ainda se encontram ”empatados”, seria então escolhida a melhor ativação definida no primeiro critério, desconsiderando assim o *threshold*, *i.e.* o algoritmo tenta minimizar o máximo de critérios possíveis, porém caso não seja possível, o primeiro critério da lista é o escolhido. Assim, no pior caso, o algoritmo pode ”degenerar” para uma otimização de um único objetivo.

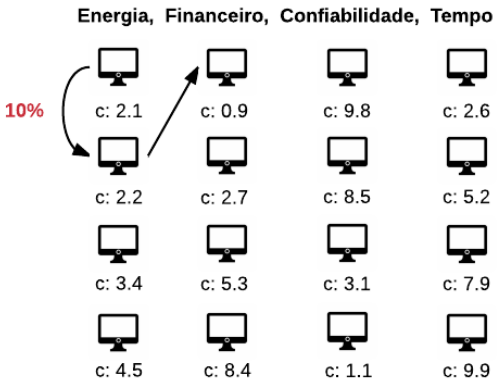

Figure 1. Exemplo de escalonamento hierárquico

Na próxima seção apresentamos como o algoritmo proposto foi avaliado utilizando-se *workflows* sintéticos e reais da área de bioinformática.

5. Avaliação Experimental

Esta seção apresenta a avaliação da abordagem proposta, explicando as ferramentas adotadas para execução, e as métricas utilizadas, bem como os parâmetros e ambientes onde foram executados os experimentos. Para os experimentos utilizamos o SGWfC SciCumulus [Oliveira et al. 2012] para gerenciar a execução do SciPhy [Ocaña et al. 2011], que é um *workflow* de análise filogenética, e que será explicado na subseção 5.1. O ambiente para execução dos testes foi a nuvem da Amazon EC2,

onde o mesmo foi executado em paralelo, tratado na subseção 5.3. Também foram feitos experimentos com um *workflow* sintético em um ambiente simulado baseado no WorkflowSim [Yang et al. 2014], tratado na subseção 5.2.

5.1. O *Workflow* SciPhy

O SciPhy é um *workflow* desenvolvido para gerar árvores filogenéticas, cujo objetivo é representar a semelhança entre os organismos, com base em sequências de aminoácidos, RNA ou DNA embutidos em arquivos de entrada no formato multi-fasta. Desta forma, o SciPhy auxilia a explorar a filogenia e determinar a vida evolutiva de genes ou genomas destes organismos [Ocaña et al. 2011]. Este *workflow* é composto por cinco atividades principais em sua versão 2.0 [Ocana et al. 2015]: (i) *DataSelection* (Seleção de dados e formatação); (ii) *Mafft* (Construção do alinhamento genético); (iii) *ReadSeq* (Conversão do formato de alinhamento); (iv) *ModelGenerator* (Eleição do modelo evolutivo) e (v) *RaXML* (Geração da árvore filogenética).

A primeira atividade (*dataselection*), que é implementada por um *script*, verifica a qualidade e o formato dos arquivos de entrada multi-fasta, sendo a operação mais rápida do *workflow*. A segunda atividade (*mafft*) faz a construção do alinhamento genético (MSA), recebendo como entrada os arquivos multi-fasta para produção do MSA. A próxima atividade é o *readseq*, que converte os arquivos de formato FASTA para o formato PHYLIP, que é o esperado nas próximas atividades. Na atividade *modelgenerator*, que é uma das mais demoradas no *workflow*, é realizado o teste em cada um dos arquivos de entrada produzidos pela atividade anterior, com o objetivo de encontrar o melhor modelo evolutivo a ser utilizado. Por fim, tanto o melhor modelo evolutivo quanto o MSA associado a ele são utilizados na tarefa de geração de árvore filogenética executada pelo *raxml*. Como o cientista ainda não sabe qual método de alinhamento irá produzir a melhor árvore, ele tem de executar o *workflow* diversas vezes, fazendo uma varredura de parâmetros e explorando diversos algoritmos de alinhamento possíveis. Mais informações a respeito do SciPhy podem ser obtidas em Ocaña *et al.* [Ocana et al. 2015].

5.2. Avaliação com o *Workflow* Sintético

Como forma de avaliar a abordagem proposta neste artigo, foram realizados dois experimentos. O primeiro deles utilizou um *workflow* sintético cujas informações de ativações foram inspiradas em perfis de aplicações reais de dinâmica de fluidos computacional. Além disso, foi necessário especificar os tipos de máquinas virtuais que processariam as ativações, e então foram atribuídas as mesmas configurações de um *cluster* com 4 máquinas da Amazon EC2[3]. O *cluster* simulado foi montado conforme mostra a Tabela 1.

O *workflow* sintético é composto por 595 ativações (uma para cada arquivo de entrada), que possuem dependência entre si, divididos em 5 fases de processamento (simulando 5 programas de um *workflow* real de dinâmica de fluidos computacional). Foram realizadas execuções cobrindo 4 cenários utilizando tanto a abordagem hierárquica proposta quanto a abordagem ponderada do SciCumulus, onde os objetivos estavam ordenados conforme a seguir:

[3]https://aws.amazon.com/pt/ec2/instance-types/

Quantidade	Tipo	Descrição
1	*M3.Medium*	Processador de 1 núcleo, com 3,75 GB de RAM
1	*M3.Large*	Processador com 2 núcleos e 7,5 GB de RAM
1	*M3.xLarge*	Processador com 4 núcleos e 15 GB de RAM
1	*M3.2xLarge*	Processador com 8 núcleos e 30 GB de RAM

Table 1. Lista de Máquinas *Cluster* Sintético

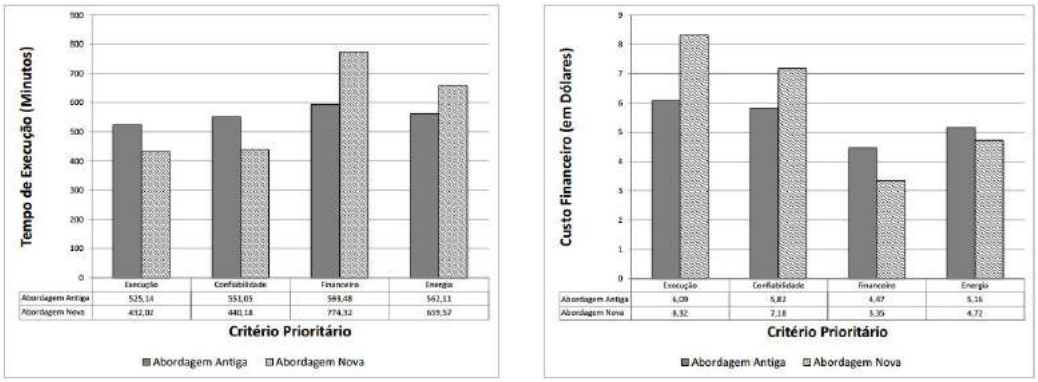

Figure 2. (a) Tempo de Execução (b) Custo Financeiro

1. **Cenário 1** (Priorizando o Tempo de Execução): Modelo multi-objetivo (Peso 0.7 para objetivo de tempo de execução e 0.1 para demais objetivos) x Lista Hierárquica: Tempo de execução, Confiabilidade, Custo financeiro, Consumo de energia.
2. **Cenário 2** (Priorizando a Confiabilidade): Modelo multi-objetivo (Peso 0.7 para objetivo confiabilidade e 0.1 para demais objetivos) x Lista Hierárquica: Confiabilidade, Tempo de execução, Consumo de energia, Custo financeiro.
3. **Cenário 3** (Priorizando o Custo financeiro): Modelo multi-objetivo (Peso 0.7 para objetivo custo financeiro e 0.1 para demais objetivos) x Lista Hierárquica: Custo monetário, Consumo de energia, Confiabilidade, Tempo de processamento.
4. **Cenário 4** (Priorizando o Custo energético): Modelo multi-objetivo (Peso 0.7 para objetivo consumo de energia e 0.1 para demais objetivos) x Lista Hierárquica: Consumo de energia, Custo monetário, Confiabilidade, Tempo de processamento.

A Figura 2(a) apresenta o tempo de execução de cada um dos 4 cenários explicitados anteriormente (cada cenário tem um critério prioritário). A Abordagem antiga representa a função ponderada e Abordagem Nova representa múltiplas funções hierarquizadas. Para o Cenário 1 é fácil observar que a abordagem hierárquica terminou sua execução em um tempo menor, sendo portanto superior a abordagem ponderada. O mesmo aconteceu no Cenário 2, onde novamente a abordagem hierárquica completou a execução do *workflow* mais rapidamente que a abordagem com uso de pesos. Isso acontece devido ao fato de as melhores máquinas virtuais (mais pontentes) são também as mais confiáveis (menos propícias a erros). No caso dos Cenários 3 e 4, o tempo de execução da abordagem hierárquica foi maior, uma vez que máquinas que consomem menos e que custam menos foram priorizadas. Essas máquinas são comumente menos potentes, logo o *workflow* executou em um tempo maior.

A Figura 2(b) representa o consumo financeiro em dólares dos 4 cenários executados. No Cenário 3, que prioriza o custo financeiro, a abordagem hierárquica foi superior a ponderada, com custo financeiro inferior, em que o cientista precisaria investir apenas 3,35 dólares enquanto que na abordagem ponderada seriam necessários 4,47 dólares. Os valores calculados para consumo financeiro foram baseados nas funções definidas nas Equações 6 e 7. O Cenário 4 priorizou o consumo de energia gasto e a abordagem hierárquica apresentou melhor desempenho. A Figura 3 mostra o consumo de energia para cada um dos 4 cenários, e no quarto conjunto de barras (relativo ao critério prioritário de energia) é possível notar que o escalonamento gerado pela abordagem hierárquica consumiu menos energia que a abordagem ponderada.

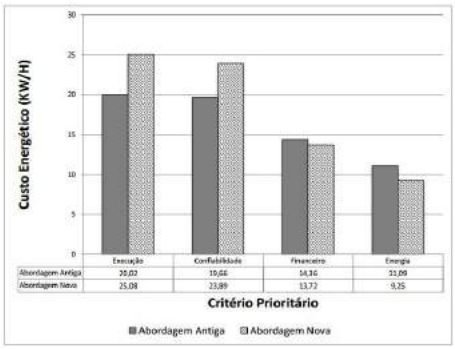

Figure 3. Custo Energético *Workflow* Sintético

Considerando todos os cenários executados com o *workflow* sintético, pode-se perceber então que um melhor escalonamento e consequente uso de recursos disponíveis é produzido pela abordagem hierárquica proposta.

5.3. Avaliação com o *Workflow* Real

Similarmente a avaliação com o *workflow* sintético apresentado na subseção 5.2, foram realizados testes com o *workflow* real SciPhy, descrito na subseção 5.1, com 250 arquivos multi-fasta de entrada. O SciPhy foi executado no ambiente de nuvem da Amazon EC2, porém nesse experimento utilizamos um *cluster* com 9 máquinas totalizando 32 núcleos virtuais, conforme apresentado na Tabela 2.

Quantidade	Tipo	Descrição
2	*M3.Medium*	Processador de 1 núcleo, com 3,75 GB de RAM
3	*M3.Large*	Processador com 2 núcleos e 7,5 GB de RAM
2	*M3.xLarge*	Processador com 4 núcleos e 15 GB de RAM
2	*M3.2xLarge*	Processador com 8 núcleos e 30 GB de RAM

Table 2. Lista de Máquinas *Cluster* Real

De forma a avaliar a abordagem proposta, o SciPhy foi executado no SciCumulus seguindo cada um dos 4 cenários apresentados anteriormente e para cada um deles o *workflow* foi executado com as duas versões do escalonador (hierárquico e ponderado). A primeira análise realizada foi em relação ao tempo de execução de cada uma das abordagens em cada um dos cenários propostos (*i.e.* prioridade dos critérios). A Figura 4(a) apresenta os tempos de execuções dos quatro cenários, em que o eixo vertical é

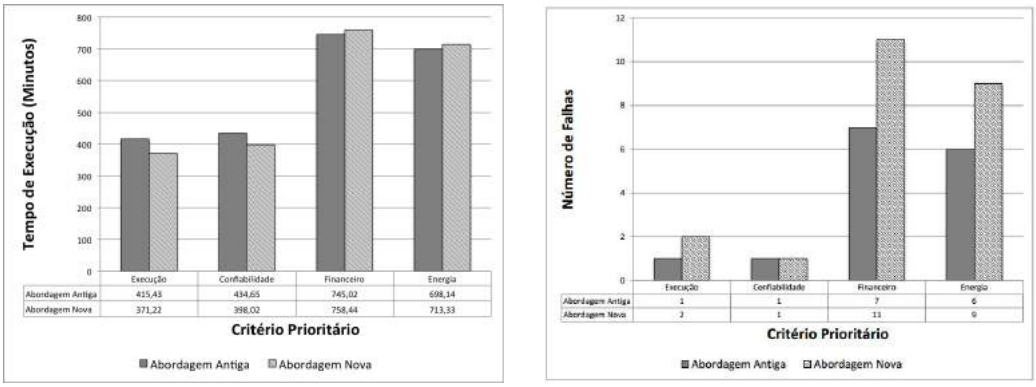

Figure 4. (a) Tempo de Execução (b) Confiabilidade

representado por tempo em minutos e o eixo horizontal os critérios que foram priorizados. Segundo o primeiro conjunto de barras (tempo de execução como critério prioritário), nota-se que a abordagem hierárquica levou apenas 371 minutos para concluir a execução do *workflow* enquanto ao utilizar o escalonador com pesos levou-se 415 minutos para concluir. Portanto, a abordagem hierárquica apresentou melhor desempenho. Podemos perceber também que mesmo quando o critério principal é a confiabilidade, também foi possível reduzir o tempo de execução, uma vez que as máquinas mais confiáveis são as mais rápidas. Já para os cenários onde os critérios de custo financeiro e custo de energia foram prioritários, o tempo de execução aumentou, uma vez que as máquinas com menor custo e menor consumo de energia são as mais lentas (*i.e* M3.medium) como pode ser observado na Figura 4(a).

Além do tempo de execução foram avaliadas as quantidades de falhas ocorridas em cada um dos cenários. A quantidade de falhas por execução do *workflow* real pode ser conferida com base no Gráfico apresentado na Figura 4(b). O cenário em que a confiabilidade era o critério prioritário apresentou a mesma taxa de falha na abordagem hierárquica e na abordagem com pesos. Porém, como o tempo de execução foi reduzido (Figura 4(a)) na abordagem hierárquica, podemos peceber que a mesma se torna mais interessante, pois manteve a taxa de falhas e reduziu o tempo de execução. Em todos os outros cenários, a taxa de falha aumentou com a utilização da abordagem hierárquica.

A terceira análise é relativa ao custo financeiro de cada cenário. Basicamente quando o cientista opta por minimizar prioritariamente o custo financeiro, ele está abrindo mão de desempenho e confiabilidade. Assim, a execução do *workflow* se caracteriza como uma execução mais longa e com mais falhas, pois máquinas mais baratas serão usadas prioritariamente. Logo, se analisarmos somente o cenário em que o custo financeiro é o prioritário pela Figura 5(a) nota-se no terceiro grupo que sua execução foi mais demorada, e para levar mais tempo, máquinas inferiores foram as mais utilizadas e por isso a ocorrência de falhas é mais presente (conforme já apresentado na 4(b)). Entretanto, como podemos observar na Figura 5(a), o custo financeiro desse cenário foi o menor (nas abordagens hierárquica e com pesos) e foi efetivamente reduzido utilizando-se a abordagem hierárquica. Pode-se peceber que mesmo quando o critério de consumo energético era o prioritário, a abordagem hierárquica ainda foi capaz de reduzir o custo financeiro associado.

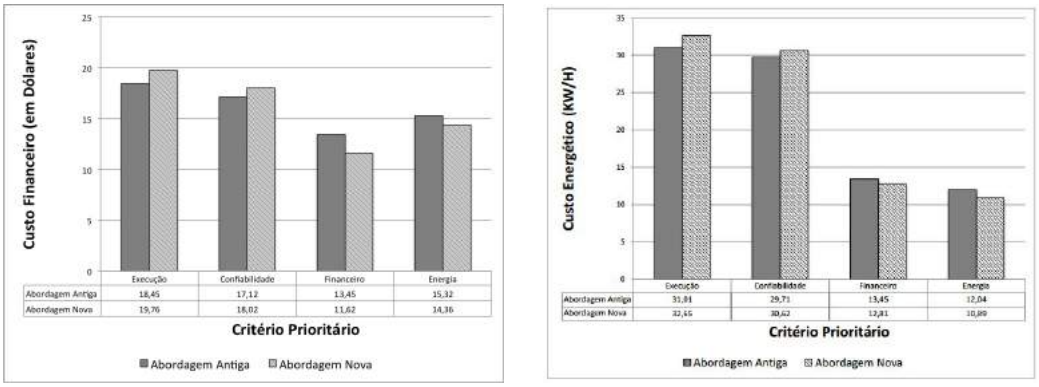

Figure 5. (a) Custo Financeiro (b) Consumo Energético

A quarta e última análise tem como objetivo verificar o consumo de energia do *cluster* durante a execução do *workflow*. Os valores de consumo energético podem ser obtidos na Figura 5(b). As máquinas que consomem menos energia (e para onde as maiores ativações são escalonadas quando se quer minimizar o custo de energia) são as com menor poder de processamento, consequentemente essas execuções tem seu tempo total como o segundo mais demorado, e a segunda maior ocorrência de falhas. Porém, ao analisarmos o consumo de energia somente, como apresentado na Figura 5(b) podemos peceber que quando o critério prioritário era o consumo energético, a abordagem hierárquica consumiu menos *kw/h* que a abordagem com pesos. De acordo com os resultados obtidos, a abordagem hierárquica obteve melhores resultados em todas as execuções do *workflow* real, comprovando que este tipo de escalonamento consegue obter melhores resultados do que uma abordagem ponderada em funções multiobjetivos, atendendo melhor as necessidades do usuário e proporcionando um ambiente mais real, com uma execução do *workflow* mais eficiente.

6. Conclusão e Trabalhos Futuros

Muitos dos escalonadores de SGWfCs existentes utilizam funções multi-objetivo ponderadas para escalonar as ativações do *workflow* em ambientes distribuídos como nuvens de computadores. Nesses sistemas, o cientista atribui pesos para cada objetivo, definindo assim o tipo de execução que atenda as suas necessidades. Esse tipo de abordagem acaba sendo complexa de ser utilizada por dois motivos principais: (i) o cientista pode não ter conhecimento suficiente para atribuir os pesos e (ii) quanto maior a quantidade de objetivos na função, pior pode se tornar o escalonamento, uma vez que os pesos dados aos objetivos se tornam ”pulverizados”.

Dados esses problemas e com o objetivo de evoluir ainda mais os escalonadores dos SGWfCs, baseado nas ideias de Durillo *et al* [Durillo et al. 2015] que utilizam a abordagem Pareto, em que é construída uma lista de escalonamentos como solução para atender a todos os objetivos do *workflow*, foi desenvolvido um novo escalonador hierárquico multiobjetivo nesse artigo. Esse escalonador utiliza múltiplas funções objetivo hierarquizadas. Os resultados experimentais mostraram que é factível o uso da abordagem hierárquica, uma vez que os critérios puderam ser minimizados como na abordagem com pesos. Como trabalhos futuros pretendemos evoluir a abordagem proposta para considerar ambientes de nuvens federadas, onde o usuário e o SGWfC

têm mais opções de VMs fornecidas pelos provedores de nuvens, conforme discutido por Durillo *et al.* [Durillo et al. 2015].

References

Assayad, I., Girault, A., and Kalla, H. (2004). A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints. In *Dependable Systems and Networks, 2004 International Conference on*, pages 347–356. IEEE.

de Oliveira, D. C. M. (2012). *Uma Abordagem de Apoio à Execução Paralela de Workflows Científicos em Nuvens de Computadores*. PhD thesis, Universidade Federal do Rio de Janeiro.

Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540.

Deelman, E., Mehta, G., Singh, G., Su, M.-H., and Vahi, K. (2007). Pegasus: mapping large-scale workflows to distributed resources. In *Workflows for e-Science*, pages 376–394. Springer.

Duan, R., Prodan, R., and Li, X. (2014). Multi-objective game theoretic schedulingof bag-of-tasks workflows on hybrid clouds. *Cloud Computing, IEEE Transactions on*, 2(1):29–42.

Durillo, J. J., Nae, V., and Prodan, R. (2014). Multi-objective energy-efficient workflow scheduling using list-based heuristics. *Future Generation Computer Systems*, 36:221–236.

Durillo, J. J., Prodan, R., and Barbosa, J. G. (2015). Pareto tradeoff scheduling of workflows on federated commercial clouds. *Simulation Modelling Practice and Theory*, 58:95–111.

Fard, H. M., Prodan, R., Barrionuevo, J. J. D., and Fahringer, T. (2012). A multi-objective approach for workflow scheduling in heterogeneous environments. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 300–309. IEEE Computer Society.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21.

Garg, S. K., Buyya, R., and Siegel, H. J. (2009). Scheduling parallel applications on utility grids: time and cost trade-off management. In *Proceedings of the Thirty-Second Australasian Conference on Computer Science-Volume 91*, pages 151–160. Australian Computer Society, Inc.

Hakem, M. and Butelle, F. (2007). Reliability and scheduling on systems subject to failures. In *Parallel Processing, 2007. ICPP 2007. International Conference on*, pages 38–38. IEEE.

Ocaña, K. A., de Oliveira, D., Ogasawara, E., Dávila, A. M., Lima, A. A., and Mattoso, M. (2011). Sciphy: A cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In *Advances in Bioinformatics and Computational Biology*, pages 66–70. Springer.

Ocana, K. A., Silva, V., De Oliveira, D., and Mattoso, M. (2015). Data analytics in bioinformatics: Data science in practice for genomics analysis workflows. In *e-Science (e-Science), 2015 IEEE 11th International Conference on*, pages 322–331. IEEE.

Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., and Mattoso, M. (2011). An algebraic approach for data-centric scientific workflows. *Proc. of VLDB Endowment*, 4(12):1328–1339.

Oliveira, D., Ogasawara, E., Ocaña, K., Baião, F., and Mattoso, M. (2012). An adaptive parallel execution strategy for cloud-based scientific workflows. *Concurrency and Computation: Practice and Experience*, 24(13):1531–1550.

Oliveira, L. S. d. and Saramago, S. A. F. P. (2010). Multiobjective optimization techniques applied to engineering problems. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 32:94 – 105.

Pandey, S., Wu, L., Guru, S. M., and Buyya, R. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*, pages 400–407. IEEE.

Sakellariou, R., Zhao, H., Tsiakkouri, E., and Dikaiakos, M. D. (2007). Scheduling workflows with budget constraints. In *Integrated research in GRID computing*, pages 189–202. Springer.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55.

Yang, M., Rutherfoord, B., and Qian, K. (2014). Learning cloud computing and security through cloudsim simulation. In *Proceedings of the 2014 Information Security Curriculum Development Conference*, InfoSec '14, pages 6:1–6:1, New York, NY, USA. ACM.

Yu, J., Kirley, M., and Buyya, R. (2007). Multi-objective planning for workflow execution on grids. In *Proceedings of the 8th IEEE/ACM International conference on Grid Computing*, pages 10–17. IEEE Computer Society.

Zhao, Y., Hategan, M., Clifford, B., Foster, I., Von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T., and Wilde, M. (2007). Swift: Fast, reliable, loosely coupled parallel computation. In *Services, 2007 IEEE Congress on*, pages 199–206. IEEE.