HEURÍSTICA BASEADA EM GERAÇÃO DE COLUNAS PARA OTIMIZAÇÃO ENERGÉTICA EM *CLUSTERS* DE SERVIDORES DE GRANDE ESCALA

Hugo Harry Kramer, Eduardo Uchoa
Departamento de Engenharia de Produção - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 4º andar, São Domingos, 24210-240, Niterói, RJ
hugoharry@gmail.com, uchoa@producao.uff.br

Vinicius Petrucci
Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 3º andar, São Domingos, 24210-240, Niterói, RJ
vpetrucci@ic.uff.br

Anand Subramanian
Departamento de Engenharia de Produção - Universidade Federal da Paraíba
Centro de Tecnologia, Campus I - Bloco G, Cidade Universitária, 58051-970, João Pessoa, PB
anand@ct.ufpb.br

RESUMO

Plataformas emergentes conhecidas como computação em nuvem disponibilizam diversos serviços de internet que são hospedados e compartilhados em *clusters* de servidores físicos e consolidados através de técnicas de virtualização. O grande consumo energético destes *clusters* se tornou uma importante questão econômica e ecológica, criando a necessidade de se investigar técnicas de otimização capazes de melhorar a eficiência energética de tais infraestruturas computacionais. Neste trabalho são apresentados modelos e algoritmos para minimizar o consumo de energia em um *cluster*, não apenas ligando/desligando servidores, mas também ajustando as frequências de operação de suas CPUs. Nossa abordagem difere de outras encontradas na literatura por ser mais realista, considerando inclusive a heterogeneidade dos servidores. Os algoritmos de otimização propostos são baseados em técnicas de geração de colunas e os experimentos realizados indicam que eles são capazes de obter soluções de alta qualidade em baixo tempo computacional, mesmo para instâncias de grande porte.

PALAVRAS CHAVE. Geração de colunas, Gerenciamento de energia, Virtualização de *clusters* de servidores.

ABSTRACT

Platforms known as utility/cloud computing make available a sort of network services which, in turn, are hosted on a shared cluster infrastructure of physical servers,and are consolidated using server virtualization techniques. The energy consumption of these clusters became a important economic and ecological concern, and thus requires major investigation of optimization techniques to improve the energy efficiency of such computing infrastructures. This work deals with models and algorithms whose the goal is to minimize the energy consumption in a server cluster, not only by the use of on/off mechanisms, but also adjusting its CPUs operating frequencies. Our approach differs from those found in literature by its more realistic assumptions, which include the servers heterogeneity. The proposed optimization algorithms are based in column generation techniques, and the experiments suggest that they are suitable to obtain consistently high quality solutions in a short processing time, even in large instances.

KEYWORDS. Column generation, Energy management, Server cluster virtualization.

1 Introdução

Um número crescente de *clusters* de servidores tem sido empregado em *data centers* de grande escala no apoio ao desenvolvimento e implementação de diferentes serviços e aplicações em ambientes computacionais escaláveis e eficientes, especialmente focados em serviços baseados na Internet. Um *cluster* de servidores é um sistema distribuído que consiste de centenas ou milhares de máquinas interligadas por uma rede rápida (Cardellini *et al.*, 2002). Tais arquiteturas de *clusters* vêm se tornando comuns em plataformas conhecidas como computação em nuvem (Hayes, 2008), a exemplo da Amazon EC2 e da Google App Engine. Nessas plataformas, os serviços são hospedados em sua maioria em servidores físicos compartilhados e suas cargas de processamento podem ser distintas, além de estarem sujeitas a variações ao longo do tempo.

Os *clusters* de servidores que viabilizam estas plataformas têm, em geral, um grande porte de maneira que seja possível atender a grandes demandas de processamento e implicam em custos elevados com energia elétrica, além de contribuir indiretamente com o aumento da geração e emissão de gases de efeito estufa e a consequente deterioração do meio ambiente (Kaplan *et al.*, 2008). Isto mostra que, de fato, o tema deve ser visto como uma questão de interesse quanto à investigação do uso de técnicas de otimização que contemplem o aprimoramento da eficiência energética dos *clusters* e da infraestrutura envolvida (Bianchini e Rajamony, 2004; Fan *et al.*, 2007; Ranganathan, 2010).

A melhor utilização dos recursos de *hardware* de um *cluster* (e, com isso, o aumento da eficiência energética destes ambientes computacionais), onde diversos serviços independentes e com demandas por processamento distintas devem ser executados, se faz possível por meio da aplicação de técnicas de virtualização. Segundo Kiyanclar (2005), virtualização é a reprodução fiel em *software* de uma arquitetura completa dando a ilusão de uma máquina real às aplicações executadas sobre a máquina virtual.

O emprego de técnicas de virtualização possibilita a utilização de uma ou mais máquinas virtuais em um único servidor físico e tornam possível a consolidação de vários serviços por servidor. Além disso, é possível distribuir dinamicamente os serviços aos servidores de um ambiente virtualizado permitindo que alguns servidores sejam desligados em períodos de baixa demanda e ligados novamente quando for exigido. Isto também pode ser combinado com o que se denomina em inglês por *Dynamic Voltage and Frequency Scaling* (DVFS), uma técnica que consiste em variar a frequência e a tensão do processador em tempo de execução de acordo com as necessidades de processamento com o intuito de utilizar a energia de maneira mais eficiente.

Neste contexto, o problema a ser tratado neste trabalho é o de encontrar a melhor configuração para um *cluster* heterogêneo virtualizado de grande escala com o objetivo de minimizar os custos relacionados ao consumo de energia elétrica de forma que a demanda por processamento e a execução dos serviços no *cluster* seja garantida. Para tanto, considera-se que mais de um serviço pode ser executado por um servidor físico por meio de técnicas de virtualização, um determinado servidor físico pode ser ligado e desligado dependendo da situação e a capacidade de processamento de um servidor pode ser ajustada por meio de DVFS. Daqui em diante, este problema será chamado de Problema de Configuração de *Clusters* Virtualizados Heterogêneos (PCCVH).

O restante do artigo está organizado da maneira que segue. A Seção 2 apresenta os trabalhos relacionados ao problema tratado neste artigo. A Seção 3 descreve formalmente e apresenta uma formulação matemática para o PCCVH. A Seção 4 apresenta a abordagem de GC proposta. Na Seção 5 os resultados obtidos são apresentados e discutidos. Por fim, a Seção 6 contém as conclusões acerca da abordagem de solução proposta para o problema.

2 Trabalhos relacionados

Algumas abordagens de otimização, baseadas no Problema de Empacotamento em Caixas (PEC), são encontradas na literatura para tratar da configuração de *clusters* de servidores

virtualizados. Bichler *et al*. (2006) tratam da atribuição de máquinas virtuais com foco no planejamento da capacidade, enquanto Khanna *et al*. (2006) buscam minimizar os custos de migração de serviços entre os servidores de forma que a performance seja garantida. No entanto, estes trabalhos não abordam a questão do consumo energético.

Wang *et al*. (2008) apresentam uma arquitetura de controle para garantia de eficiência energética em tempo real para ambientes virtualizados, mas não consideram mecanismos de liga/desliga nem a alocação das máquinas virtuais. Um procedimento heurístico de solução é apresentado por Srikantaiah *et al*. (2008) para consolidação de serviços em servidores virtualizados buscando economia de energia. Entretanto, a técnica de DVFS não é considerada e o algoritmo não apresenta garantias de que as soluções obtidas sejam ao menos próximas do ótimo. Um algoritmo recente para otimizar a consolidação de máquinas virtuais é apresentado por Wang e Wang (2010) considerando DVFS. No entanto, a política de DVFS é aplicada local e independentemente a cada servidor, o que pode não resultar em uma configuração ótima global para o *cluster*.

Abordagens que consideram tanto DVFS quanto mecanismos de liga/desliga podem ser encontradas em Elnozahy *et al*. (2003), Rusu *et al*. (2006) e Bertini *et al*. (2007) com o objetivo de minimizar o consumo energético, mas não são aplicáveis a *clusters* de servidores virtualizados e, portanto, não tratam da consolidação de serviços nos servidores. Uma revisão de trabalhos que abordam a racionalização do consumo de energia em ambientes de computação em nuvem é apresentada em Berl *et al*. (2010).

Não é do conhecimento dos autores a existência de trabalhos que considerem simultaneamente (1) o emprego de virtualização, (2) a técnica de DVFS combinada com mecanismos de ligar/desligar servidores, (3) o caso da heterogeneidade dos servidores e (4) uma abordagem eficiente de otimização baseada em Geração de Colunas (GC). Neste trabalho são propostas abordagens de GC para tratar o PCCVH de forma a encontrar soluções de boa qualidade em tempo computacional razoável, tendo em vista que em situações práticas os serviços e suas cargas são dinâmicos e o *cluster* deve ser reconfigurado periodicamente para garantir o atendimento das demandas a cada mudança.

3 Formulação matemática

Matematicamente, o PCCVH pode ser definido da seguinte forma: sejam o conjunto $N = \{1, 2, ..., i, ..., n\}$ dos tipos de servidores no *cluster*, $S_i = \{1, 2, ..., s, ..., q_i\}$ o conjunto dos servidores do tipo $i \in N$, o conjunto $F_i = \{1, 2, ..., j, ..., r_i\}$ das frequências de CPU em que cada servidor do tipo $i \in N$ pode operar, e $K = \{1, 2, ..., k, ..., m\}$ o conjunto dos serviços que devem ser executados no *cluster*. Cada servidor do tipo $i \in N$ operando a uma frequência de CPU $j \in F_i$ tem capacidade D_{ij} (por exemplo, em requisições/s). O consumo energético (por exemplo, em Watts) para manter um servidor do tipo $i \in N$ operando na frequência $j \in F_i$ é dado por C_{ij}. A demanda (na mesma unidade da capacidade D_{ij}) de um serviço $k \in K$ é representada por d_k.

O PCCVH pode ser modelado como um problema de Programação Linear Inteira (PLI) como descrito na formulação $\mathcal{F}_1$ a seguir. As variáveis de decisão são definidas como segue: $y_{isj} \in \{0, 1\}$ é uma variável binária que indica se o servidor $s \in S_i$ do tipo $i \in N$ está ligado e operando na frequência $j \in F_i$, e $x_{isjk} \in \{0, 1\}$ é uma variável binária que indica se o serviço $k \in K$ é executado no servidor $s \in S_i$ do tipo $i \in N$ que está ligado e operando na frequência de CPU $j \in F_i$.

$$(\mathcal{F}_1)\ z = Min \sum_{i \in N} \sum_{s \in S_i} \sum_{j \in F_i} C_{ij} y_{isj} \tag{1}$$

Sujeito a

$$\sum_{k \in K} d_k x_{isjk} \leq D_{ij} y_{isj} \qquad \forall i \in N, s \in S_i, j \in F_i \tag{2}$$

$$\sum_{i \in N} \sum_{s \in S_i} \sum_{j \in F_i} x_{isjk} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{j \in F_i} y_{isj} \leq 1 \qquad \forall i \in N, s \in S_i \tag{4}$$

$$x_{isjk} \in \{0,1\} \qquad \forall i \in N, s \in S_i, j \in F_i, k \in K \tag{5}$$

$$y_{isj} \in \{0,1\} \qquad \forall i \in N, s \in S_i, j \in F_i. \tag{6}$$

A função objetivo dada por (1) busca minimizar o consumo energético do *cluster*. As restrições (2) garantem que a capacidade D_{ij} de um servidor $s \in S_i$ do tipo $i \in N$ operando na frequência $j \in F_i$ não seja ultrapassada. As restrições (3) impõem que cada serviço $k \in K$ seja executado no *cluster*, enquanto as restrições (4) garantem que cada servidor $s \in S_i$ do tipo $i \in N$ pode operar em, no máximo, uma frequência $j \in F_i$. As expressões (5) e (6) definem a natureza das variáveis x_{isjk} e y_{isj}.

O PCCVH é um problema NP-difícil, desde que pode ser visto como uma generalização do Problema de Empacotamento em Caixas de Tamanho Variável (PECTV), conhecido em inglês por *Variable-sized Bin Packing Problem*, que é NP-difícil (Friesen e Langston, 1986). No PCCVH, os serviços seriam os itens que devem ser empacotados, as demandas seriam os tamanhos dos itens e as caixas seriam os servidores com sua capacidade e seu custo determinados pela configuração. Além disso, cada tipo de caixa (servidor) pode assumir um entre r_i, $i \in N$ possíveis tamanhos (capacidades, determinadas pela frequência), o que também vai determinar seu custo. Outra diferença se dá pelo fato de o número de caixas de cada tipo ser limitado.

O uso direto de $\mathcal{F}_1$ para determinar a configuração de um *cluster* de grande escala é impraticável, uma vez que o número de variáveis e restrições se tornará grande demais para ser tratado por um resolvedor de problemas de PLI. Em experimentos preliminares, esta formulação se mostrou ineficiente em termos de tempo computacional, inclusive, para obter *lower bounds* para as instâncias utilizadas através da resolução de sua relaxação linear. Para superar esta limitação, propomos uma abordagem eficiente baseada em GC que será descrita na seção seguinte.

4 Abordagem de geração de colunas

Nesta seção, a abordagem proposta para o problema é descrita e consiste em obter soluções inteiras viáveis através de um algoritmo baseado em GC. Como o uso direto da formulação $\mathcal{F}_1$ para obter soluções inteiras viáveis se torna proibitivo quando o número de servidores e serviços cresce, e consequentemente o número de variáveis, podemos reformular o problema através da Decomposição de Dantzig e Wolfe (1960). Isto permite resolver a relaxação linear da formulação resultante por meio de um algoritmo de GC.

Na reformulação do problema, é possível agregar as linhas que correspodem a serviços com demandas idênticas ou similares, isto é, os serviços cujas demandas pertencem a um intervalo definido em torno de uma média podem ser agrupados. Segundo Alves e Valério de Carvalho (2007), em modelos com tais características existe uma solução ótima com variáveis duais iguais para, a exemplo do PEC, itens com o mesmo tamanho. Gilmore e Gomory (1963) também relataram este fato em várias iterações do algoritmo de GC para o Problema de Corte de Materiais (PCM). Este esquema de agregação é razoável para o PCCVH dado que há uma certa tolerância na estimativa das demandas dos serviços que devem ser executados no *cluster*.

Isto posto, seja $K' = \{1, 2, ..., k, ..., m'\}$ o conjunto dos índices dos grupos de serviços agregados e n_k o número de serviços com demanda d_k. Além disso, assume-se que o conjunto K' está ordenado em ordem decrescente de acordo com os valores das demandas d_k, $k \in K'$.

Um *padrão* é definido como o vetor de incidência em $\mathbb{Z}^{m'}$ associado a um subconjunto de K'. Para um padrão p, $p_k \leq n_k$ é uma constante inteira que indica quantos serviços com demanda d_k, $k \in K'$ são atendidos pelo padrão $p \in P(i)$. Seja $P(i)$ o conjunto de todos os padrões viáveis para um servidor do tipo $i \in N$, ou seja, aqueles tal que

$$\sum_{k \in K'} p_k d_k \leq D_{ir_i} \qquad \forall i \in N, \tag{7}$$

onde D_{ir_i} é a capacidade máxima de um servidor do tipo $i \in N$ e está associada à sua frequência mais alta. Uma variável inteira e não negativa λ_{ip} define quantos serviços de um padrão $p \in P(i)$ são atribuídos a um servidor do tipo $i \in N$ ligado na frequência $f(p,i)$, definida como a menor frequência tal que $\sum_{k \in K'} p_k d_k \leq D_{if(p,i)}$.

Além do esquema de agregação de linhas, a convergência do algoritmo de GC pode ser acelerada se um pequeno número de *variáveis de troca* for adicionado. As variáveis adicionais $t_k \in \mathbb{Z}_+$ representam a substituição de um certo número de serviços agregados com demanda d_k pela mesma quantidade de serviços agregados com demanda d_{k-1}, $k \in K'$. Estas variáveis sempre representam a troca entre um serviço agregado e outro com demanda imediatamente menor que, por sua vez, pode ser trocado da mesma forma por outro e assim por diante. A inclusão das variáveis de troca é equivalente a adicionar desigualdades ao problema dual, restringindo seu espaço de solução (Valério de Carvalho, 2005). A adição de desigualdades ao problema dual é utilizada por Alves e Valério de Carvalho (2007) como uma das diferentes estratégias de estabilização e aceleração de um algoritmo de GC proposto para o PECTV.

De acordo com o que foi exposto acima, a reformulação $\mathcal{MP}$ do problema via Decomposição de Dantzig-Wolfe é definida como segue:

$$(\mathcal{MP})\ Min \sum_{i \in N} \sum_{p \in P(i)} C_{if(p,i)} \lambda_{ip} \tag{8}$$

Sujeito a

$$\sum_{i \in N} \sum_{p \in P(i)} p_k \lambda_{ip} - t_k + t_{k-1} \geq n_k \qquad \forall k \in K' \tag{9}$$

$$\sum_{p \in P(i)} \lambda_{ip} \leq q_i \qquad \forall i \in N \tag{10}$$

$$\lambda_{ip} \geq 0 \qquad \forall i \in N, p \in P(i) \tag{11}$$

$$t_k \geq 0 \qquad \forall k \in K'. \tag{12}$$

Nesta formulação, chamada de problema mestre, as restrições de integralidade das variáveis λ e t são relaxadas e, desta forma, a solução de $\mathcal{MP}$ por GC resulta em um limite inferior para o valor da solução do problema. A função objetivo (8) minimiza o consumo energético do *cluster*. As restrições (9) garantem que todas as demandas sejam atendidas. As restrições (10) impedem que mais do que q_i servidores do tipo $i \in N$ operem numa mesma frequência. Por fim, as expressões (11) e (12) definem o domínio das variáveis λ_{ip} e t_k.

4.1 Subproblema de *pricing*

Sejam π_k, $k \in K'$, as variáveis duais associadas às restrições (9) e sejam α_i as variáveis duais associadas às restrições (10). Conhecidos os valores de tais variáveis duais, para cada par (i,j), $i \in N$, $j \in F_i$, o subproblema de *pricing* SP_{ij} é formulado da forma a seguir.

$$(\text{SP}_{ij})\ Min\ C_{ij} - \alpha_i - \sum_{k \in K'} \pi_k p_k \tag{13}$$

Sujeito a

$$\sum_{k \in K'} d_k p_k \leq D_{ij} \tag{14}$$

$$p_k \in \mathbb{Z}_+ \qquad \forall k \in K'. \tag{15}$$

Este problema pode ser visto como um Problema da Mochila Inteira e, mesmo sendo NP-difícil, é bem resolvido na prática através de algoritmos pseudo-polinomiais. Uma solução obtida por SP_{ij} em termos de p_k, $k \in K'$, representa um padrão válido para servidores do tipo $i \in N$ operando a uma frequência $j \in F_i$. No algoritmo de GC, as colunas são geradas pelo subproblema de *pricing* e adicionadas ao Problema Mestre Restrito (PMR) se o valor da função objetivo (13), que representa o custo reduzido de uma coluna, for negativo.

4.2 Base inicial para geração de colunas

Para gerar uma base inicial para o algoritmo de GC, foi desenvolvida uma heurística gulosa simples definida a seguir. Sejam LS uma lista que contém todos os servidores do *cluster* e LA uma lista composta por todos os serviços. Primeiramente, LS é ordenada de acordo com as capacidades máximas dos servidores e LA de acordo com as demandas, ambas em ordem decrescente. Em seguida, enquanto LA não estiver vazia, deve-se tentar atribuir os serviços restantes à máxima frequência do primeiro elemento corrente de LS, sempre dando prioridade àqueles serviços com maior demanda. Por fim, as variáveis λ associadas à solução completa do problema gerada pela heurística e as variáveis de troca t são adicionadas ao PMR. Caso esta heurística não seja capaz de encontrar uma solução inteira viável, uma coluna artificial deve ser adicionada ao PMR para fornecer uma base inicial viável para a inicialização do algoritmo de GC. Esta coluna tem custo igual a $\sum_{i \in N} C_{ir_i} q_i + 1$ e coeficientes iguais ao lado direito das restrições (9) de $\mathcal{MP}$.

4.3 Algoritmo de geração de colunas

O pseudocódigo do algoritmo de GC é apresentado em Algoritmo 1. Assume-se que os resolvedores de PL (ResolvedorPL) e de Problemas da Mochila (ResolvedorSP) são caixas-pretas e que uma base inicial é fornecida ao PMR pela heurística descrita na Subseção 4.2. A variável booleana $colsAdicionadas$ é definida como verdadeiro se alguma coluna tiver sido adicionada ao PMR em uma iteração e como falso, caso contrário. A primeira iteração do algoritmo se dá pela solução do PMR (linha 3). Em seguida (linhas 4-14), enquanto $colsAdicionadas$ for verdadeiro, o subproblema SP_{ij} correspondente a cada par (i, j), $i \in N$, $j \in F_i$, é resolvido e a coluna é adicionada ao PMR se seu custo reduzido for menor que zero. Após todas as colunas correspondentes ao par (i, j) terem sido adicionadas, o PMR é resolvido novamente.

4.4 Heurística primal

Uma Heurística de Arredondamento (HA) baseada em GC foi desenvolvida para obter soluções inteiras viáveis por meio do arredondamento para cima dos *lower bounds* das variáveis de uma solução fracionária obtida pela GC. O pseudocódigo mostrado no Algoritmo 2 contém um esboço de HA com suas três etapas: (i) geração de uma base inicial (linha 2); (ii) construção de uma solução inteira viável (linha 4); e (iii) refinamento da solução construída (linha 8). Caso não tenha sido possível obter uma solução inteira viável na etapa de construção, o método retorna a solução obtida pela heurística gulosa.

O pseudocódigo do procedimento construtivo é apresentado no Algoritmo 3, onde $i \in \{1, ..., n\}$ é o índice associado a um tipo de servidor e, além disso, assume-se que quanto maior

Algoritmo 1 GeracaoDeColunas(PMR)

1: $sol_{SP} \leftarrow \emptyset$
2: $colsAdicionadas \leftarrow$ verdadeiro
3: $sol \leftarrow$ ResolvedorPL(PMR)
4: **enquanto** $colsAdicionadas$ **faça**
5: $\quad colsAdicionadas \leftarrow$ falso
6: $\quad$ **para** cada tipo de servidor $i \in N$ **faça**
7: $\quad\quad$ **para** cada frequência $j \in F_i$ **faça**
8: $\quad\quad\quad$ Atualiza SP_{ij} com os valores de α_i e $\pi_k, \forall k \in K'$
9: $\quad\quad\quad sol_{SP} \leftarrow$ ResolvedorSP(SP_{ij})
10: $\quad\quad\quad$ **se** $f(sol_{SP}) < 0$ **então**
11: $\quad\quad\quad\quad$ Adiciona a coluna associada à variável λ ao PMR
12: $\quad\quad\quad\quad colsAdicionadas \leftarrow$ verdadeiro
13: $\quad$ **se** $colsAdicionadas$ **então**
14: $\quad\quad sol \leftarrow$ ResolvedorPL(PMR)
15: **retorna** sol

Algoritmo 2 HeuristicaDeArredondamento()

1: $sol \leftarrow \emptyset$
2: PMR $\leftarrow$ GeraBaseInicial()
3: $solHeuristicaGulosa \leftarrow$ solução inteira associada à base inicial
4: $sol \leftarrow$ Construcao(PMR)
5: **se** sol for fracionária **então**
6: $\quad$ **retorna** $solHeuristicaGulosa$
7: **senão**
8: $\quad sol \leftarrow$ Refinamento(PMR, sol)
9: **retorna** sol

for o valor de i, menor será a capacidade de um servidor deste tipo em sua frequência mais alta. Inicialmente, é definido $i = n$ (linha 1). O número máximo de colunas permitido durante a fase de construção é denotado por $maxCols$ e $maxTentativas$ é o número máximo de tentativas consecutivas de arredondamento de uma variável. Com base em experimentos preliminares, adotou-se que $maxCols = 1000$ e $maxTentativas = 10n$ (linha 2). Em seguida, uma solução inicial, possivelmente fracionária, é obtida através do algoritmo de GC (linha 4). Neste ponto do algoritmo, primeiramente é utilizado um resolvedor de Problemas da Mochila 0-1 até que não seja mais possível adicionar uma coluna binária ao PMR. Em seguida, um resolvedor de Problemas da Mochila Inteira é usado até a convergência do algoritmo de GC. Enquanto a solução for fracionária e $tentativas \leq maxTentativas$, busca-se gerar uma solução inteira viável modificando a solução encontrada pela GC (linhas 5-19). Caso a solução obtida pela GC seja inviável ou não for mais possível arredondar o valor de pelo menos uma variável λ de forma que o atendimento às restrições do problema seja mantido, deve-se desfazer a fixação das variáveis associadas ao servidor i (linhas 6 - 7) e incrementar o valor de $tentativas$, ou seja, tais variáveis λ terão seus *bounds* restaurados para $\lambda \geq 0$. Feito isso, o valor de i deve ser decrementado ou, caso $i = 0$, definido como $i = n$ (linhas 8-13). Se for encontrada uma solução fracionária viável, reinicia-se o valor de $tentativas$ e arredonda-se para cima o *lower bound* (LB) de uma variável λ selecionada de acordo com o critério a seguir. Seja $\mathcal{P}$ o conjunto das variáveis que possuem valores fracionários em uma solução obtida por GC. Uma variável λ_l, $l \in \mathcal{P}$, onde $l = \text{argmin}_{p \in \mathcal{P}}(\lceil \lambda_p \rceil - \lambda_p)$ (linha 16) é escolhida para ter seu LB arredondado para cima. Se o número de colunas do PMR for maior que $maxCols$, aquelas colunas associadas às variáveis λ que estiverem fora da base são removidas (linha 18). Por fim, o algoritmo de GC é resolvido sobre o PMR atual (linha 19). Aqui, diferentemente do que ocorre na linha 4, apenas o resolvedor de Problemas da Mochila 0-1 é utilizado.

Uma rotina de refinamento, cujo pseudocódigo está descrito no Algoritmo 4, é aplicada à solução obtida pelo procedimento construtivo de forma a melhorá-la e funciona como segue. Seja

Algoritmo 3 Construcao(PMR)

1: $i \leftarrow n$
2: $maxCols \leftarrow 1000$, $maxTentativas \leftarrow 10n$
3: $tentativas \leftarrow 0$
4: $sol \leftarrow$ GeracaoDeColunas(PMR)
5: **enquanto** sol for fracionária **e** $tentativas \leq maxTentativas$ **faça**
6: **se** sol for inviável **ou** não houver mais variáveis λ que possam ser arredondadas para cima **então**
7: Atualiza PMR restaurando os LBs originais destas variáveis, i.e., $\lambda \geq 0$, associadas ao tipo de servidor i
8: $i \leftarrow i - 1$
9: $tentativas \leftarrow tentativas + 1$
10: **se** $tentativas > maxTentativas$ **então**
11: **retorna** sol
12: **se** $i = 0$ **então**
13: $i = n$
14: **senão**
15: $tentativas \leftarrow 0$
16: Atualiza PMR definindo o LB da variável fracionária λ_l, onde $l = \text{argmin}_{p \in \mathcal{P}}(\lceil \lambda_p \rceil - \lambda_p)$, como $\lceil \lambda_p \rceil$
17: **se** número de colunas de PMR $> maxCols$ **então**
18: Remove as colunas associadas às variáveis λ que estão fora da base
19: $sol \leftarrow$ GeracaoDeColunas(PMR)
20: **retorna** sol

PMR$'$ uma cópia da base de PMR (linhas 2-3). Para cada tipo de servidor $i \in N$, a fixação dos LBs das variáveis λ associadas a i é desfeita e o procedimento construtivo é chamado para tentar encontrar uma nova solução inteira viável (linhas 4-9). Caso a solução seja melhorada, a melhor solução é atualizada (linhas 7-8). Ao final de cada iteração, PMR é atualizado com a cópia guardada em PMR$'$ (linha 9). Durante a rotina de refinamento, apenas o resolvedor de Problemas da Mochila 0-1 é utilizado.

Algoritmo 4 Refinamento(PMR, sol)

1: $sol^* \leftarrow sol$
2: Atualiza PMR removendo as colunas associadas às variáveis λ que estão fora da base
3: PMR$'$ $\leftarrow$ PMR
4: **para** cada tipo de servidor $i \in N$ **faça**
5: Atualiza PMR restaurando os LBs originais destas variáveis, i.e., $\lambda \geq 0$, associadas ao tipo de servidor i
6: $sol \leftarrow$ Construcao(PMR)
7: **se** $f(sol) < f(sol^*)$ **então**
8: $sol^* \leftarrow sol$
9: PMR $\leftarrow$ PMR$'$
10: **retorna** sol^*

A preferência pela Mochila 0-1, ao invés da Mochila Inteira, como subproblema de *pricing* se deve ao fato da primeira ser capaz de gerar colunas mais adequadas à estratégia de arredondamento empregada. Isto porque o arredondamento para cima do limite inferior de uma variável associada a uma coluna gerada a partir da resolução da Mochila Inteira tem maior chance de levar a uma "sobrecobertura"das demandas, quando comparada a uma variável associada a uma coluna gerada a partir da resolução da Mochila 0-1.

5 Resultados computacionais

Nesta seção apresentamos os resultados computacionais obtidos para avaliar a abordagem de otimização proposta. A abordagem proposta foi implementada em C++ e os testes foram executados em um PC Intel® Core™ i7 com 2,66 GHz e 4 GB RAM e sistema operacional Linux 64 bits sendo utilizada uma única *thread*. O resolvedor de PL utilizado foi o ILOG CPLEX 12.1. Para a etapa de *pricing* foram utilizados o algoritmo Minknap[1] (Pisinger, 1997) para Problemas

[1]Código fonte disponível em `http://www.diku.dk/~pisinger/codes.html`

da Mochila 0-1 e o algoritmo Bouknap[1] (Pisinger, 2000) para Problemas da Mochila Inteira. Para todas as instâncias testadas, a heurística gulosa sempre foi capaz de fornecer uma solução inteira viável ao PMR inicial sem a necessidade de adicionar colunas artificiais. Além disso, em nenhuma instância o número máximo de tentativas consecutivas de construir uma solução inteira viável foi excedido.

As instâncias de teste utilizadas foram derivadas de medições reais do consumo de potência e capacidade de processamento para quatro diferentes tipos de servidor em um *cluster*. Maiores informações sobre o *cluster* e configuração das máquinas usadas podem ser encontradas em Petrucci *et al*. (2011).

Uma instância de teste é composta pelos seguintes elementos: número de tipos de servidores; número de servidores de cada tipo; número de frequências disponíveis em cada tipo; número de serviços que serão executados no *cluster*; demandas de cada serviço; capacidade em cada frequência de cada tipo de servidor; e, por fim, consumo energético em cada frequência de cada tipo de servidor. Cada instância de teste obedece a seguinte notação: ct-G-X-Y-$Z w$. O parâmetro G especifica um determinado grupo de instâncias. O parâmetro X representa o número de tipos de servidor no *cluster*. O número de serviços a serem executados é dado por $Y \in \{1000, 2000, 4000\}$. O parâmetro $Z \in \{a, b, c\}$ determina o intervalo discreto em que as demandas dos serviços foram geradas, onde $a = \{1, ..., 100\}$, $b = \{20, ..., 100\}$ e $c = \{50, ..., 100\}$. De fato, desde que geralmente há uma limitação de precisão ao estimar uma demanda de serviço, na maioria dos casos, é razoável considerar que mesmo quando há milhares de serviços, o número de valores de demandas distintos não é maior que algumas centenas. Por último, o parâmetro de w é usado para fazer uma distinção entre as instâncias com valores iguais para os parâmetros X, Y e Z.

Dois grupos de instâncias foram gerados, onde cada um contém 18 problemas-teste. No primeiro grupo ($G = 1$), os valores de demanda de cada serviço foram selecionados aleatoriamente dentro do intervalo especificado. Quanto ao segundo grupo ($G = 2$), os valores das capacidades de cada tipo de servidor foram divididas por um valor constante, no caso, igual a 5, e o número de servidores disponíveis para cada tipo foi multiplicado também por essa constante. Assim, a diferença entre os dois grupos de instâncias está na relação entre as demandas dos serviços e as capacidades dos servidores, de modo que essa proporção é maior nas instâncias do segundo grupo.

A Tabela 1 sumariza os resultados obtidos pela abordagem proposta para os dois grupos de instâncias. Nesta tabela, a coluna **Estratégia 1** mostra os resultados obtidos utilizando a estratégia de solução dos subproblemas de *pricing* descrita na Subseção 4.4 para os Algoritmos 3 e 4. A coluna **Estratégia 2** contém os resultados encontrados quando apenas o resolvedor de Problemas da Mochila Inteira é utilizado. Além disso, **Instância** denota o problema-teste, **LB** é o *lower bound* obtido por $\mathcal{MP}$ via GC, **Tempo** indica o tempo de CPU em segundos, **Iter** e **Cols** são, respectivamente, o número de iterações do algoritmo de GC e o número de colunas geradas até obter o *lower bound*, **UB** é o *upper bound* obtido por uma dada estratégia e **Gap (%)** representa a diferença percentual entre o *upper bound* e o *lower bound* obtido.

A partir dos resultados das relaxações lineares obtidos ao resolver $\mathcal{MP}$ via GC, observa-se que os números de iterações e de colunas geradas foram menores ao se utilizar a primeira estratégia de solução de subproblemas de *pricing* na grande maioria das instâncias do primeiro grupo e, além disso, com médias consideravelmente inferiores. A segunda estratégia obteve um resultado melhor com relação ao número de iterações apenas na instância ct-1-4-1000-b1. No segundo grupo de instâncias, a primeira estratégia produziu limites inferiores com menos iterações e colunas geradas na maioria das instâncias. Como ocorreu no primeiro grupo, a segunda estratégia efetuou, para uma instância (ct-2-4-1000-b2), menos iterações que a primeira. Em comparação aos resultados do primeiro grupo de instâncias, a diferença entre as duas estratégias no que se refere aos valores médios de iterações e colunas geradas é menos expressiva.

Quanto aos limites superiores obtidos, é possível notar que, como mencionado na Subseção 4.4, o uso da primeira estratégia tende a levar a heurística HA a melhores resultados em

relação aos obtidos pela segunda estratégia. Este fato ocorreu para 17 das 18 instâncias do primeiro grupo. Já no segundo grupo, a primeira estratégia encontrou os melhores limites superiores em 11 das 18 instâncias. Em 5 instâncias, ambas as estratégias obtiveram os mesmos limites superiores e, em 2 instâncias, os melhores limites superiores foram obtidos pela segunda estratégia. Além disso, os *gap* médios foram menores para ambos os grupos de instâncias quando a primeira estratégia foi utilizada.

Tabela 1: Resultados das relaxações lineares e da heurística de arredondamento para os dois grupos de instâncias

		Estratégia 1						Estratégia 2					
		Relaxação linear			Heurística			Relaxação Linear			Heurística		
Instância	LB	Tempo	Iter	Cols	UB	Tempo	Gap(%)	Tempo	Iter	Cols	UB	Tempo	Gap(%)
Primeiro grupo de instâncias													
ct-1-4-1000-a1	8742	0,45	**118**	**494**	**8828**	6,58	0,98	0,56	183	819	8913	1,41	1,96
ct-1-4-1000-a2	8231	0,39	**103**	**448**	**8262**	6,33	0,38	0,58	191	835	8492	1,43	3,17
ct-1-4-1000-b1	10103	0,14	60	**295**	**10139**	2,73	0,36	0,07	**53**	298	10356	0,78	2,50
ct-1-4-1000-b2	10123	0,14	**53**	**265**	**10224**	4,09	1,00	0,11	62	352	10271	0,86	1,46
ct-1-4-1000-c1	12955	0,02	**30**	**190**	**12979**	1,06	0,19	0,04	32	231	13128	0,55	1,34
ct-1-4-1000-c2	12889	0,02	**27**	**177**	**12911**	1,33	0,17	0,04	32	226	12996	0,74	0,83
ct-1-4-2000-a1	17104	0,12	**50**	**282**	**17233**	5,77	0,75	0,16	95	521	17382	1,32	1,63
ct-1-4-2000-a2	17188	0,14	**56**	**303**	**17250**	12,12	0,36	0,16	91	504	17382	1,09	1,13
ct-1-4-2000-b1	20535	0,05	**29**	**218**	**20588**	4,50	0,26	0,05	35	259	20682	1,50	0,72
ct-1-4-2000-b2	20295	0,04	**30**	**222**	**20418**	3,58	0,61	0,08	40	276	20682	1,75	1,91
ct-1-4-2000-c1	30579	0,02	**14**	**267**	**30607**	1,35	0,09	0,03	25	351	30691	1,74	0,37
ct-1-4-2000-c2	30895	0,03	**15**	**275**	30951	1,76	0,18	0,02	23	360	**30947**	3,01	0,17
ct-1-4-4000-a1	33907	0,06	**29**	**244**	**33965**	9,66	0,17	0,10	61	385	34163	2,49	0,76
ct-1-4-4000-a2	34359	0,07	**28**	**248**	**34510**	9,75	0,44	0,10	53	372	34644	2,51	0,83
ct-1-4-4000-b1	40944	0,06	**25**	**302**	**41065**	5,09	0,30	0,06	38	370	41197	2,46	0,62
ct-1-4-4000-b2	40594	0,04	**19**	**273**	**40735**	7,53	0,35	0,05	34	345	40801	3,21	0,51
ct-1-4-4000-c1	51805	0,03	**22**	**380**	**51889**	1,08	0,16	0,05	28	435	51987	4,02	0,35
ct-1-4-4000-c2	51395	0,04	**24**	**385**	**51455**	1,23	0,12	0,05	30	441	51493	2,40	0,19
Média		**0,10**	**41**	**293**		4,75	**0,38**	0,13	61	410		**1,85**	1,13
Segundo grupo de instâncias													
ct-2-4-1000-a1	43826	0,14	**77**	**544**	**43905**	1,11	0,18	0,14	89	602	44131	3,54	0,70
ct-2-4-1000-a2	41268	0,12	**68**	**500**	**41310**	1,17	0,10	0,12	82	566	41485	0,71	0,53
ct-2-4-1000-b1	50641	0,07	**37**	**488**	**50682**	0,92	0,08	0,06	39	519	50715	2,10	0,15
ct-2-4-1000-b2	50741	0,07	35	**475**	50826	0,67	0,17	0,04	**34**	493	**50818**	3,12	0,15
ct-2-4-1000-c1	64922	0,03	**28**	**577**	**64938**	0,23	0,02	0,05	33	646	**64938**	0,45	0,02
ct-2-4-1000-c2	64594	0,02	**29**	**583**	**64597**	0,24	<0,01	0,03	35	668	**64597**	0,27	<0,01
ct-2-4-2000-a1	85734	0,06	**30**	**630**	**85760**	1,23	0,03	0,08	45	700	85936	3,38	0,24
ct-2-4-2000-a2	86155	0,08	**32**	**648**	**86250**	0,93	0,11	0,09	51	737	**86250**	13,23	0,11
ct-2-4-2000-b1	102913	0,06	**32**	**792**	**102929**	0,62	0,02	0,07	37	842	**102929**	0,95	0,02
ct-2-4-2000-b2	101712	0,06	**32**	**788**	**101741**	0,64	0,03	0,06	33	816	101791	4,30	0,08
ct-2-4-2000-c1	154519	0,02	**15**	**1136**	**154582**	0,28	0,04	0,03	19	1212	154604	0,56	0,06
ct-2-4-2000-c2	156179	0,04	**17**	**1150**	**156198**	0,24	0,01	0,03	18	1168	156254	0,33	0,05
ct-2-4-4000-a1	169968	0,09	**33**	**985**	**170079**	1,07	0,07	0,07	35	1005	170146	6,98	0,10
ct-2-4-4000-a2	172232	0,08	**32**	**985**	**172436**	1,06	0,12	0,11	46	1054	172466	23,32	0,14
ct-2-4-4000-b1	205208	0,07	**32**	**1290**	**205325**	0,69	0,06	0,10	35	1326	**205325**	11,11	0,06
ct-2-4-4000-b2	203455	0,07	**31**	**1272**	**203477**	0,73	0,01	0,07	32	1302	203606	1,14	0,07
ct-2-4-4000-c1	259590	0,08	**29**	**1845**	259672	0,26	0,03	0,08	33	1913	**259599**	0,47	<0,01
ct-2-4-4000-c2	257535	0,08	**29**	**1841**	**257555**	0,22	0,01	0,09	35	1933	257565	0,74	0,01
Média		**0,07**	**34**	**918**		**0,68**	**0,06**	**0,07**	41	972		4,26	0,14

6 Conclusões e trabalhos futuros

Neste trabalho foi proposta uma nova abordagem para um problema de otimização energética de *clusters* heterogêneos virtualizados. A partir dos resultados obtidos, observa-se que a proposta de solução para o problema se mostrou eficiente ao encontrar soluções de qualidade,

próximas das ótimas, e em baixo tempo computacional, para as instâncias de grande porte geradas. Observou-se também que uma estratégia diferente para a solução dos subproblemas, em média, levou a melhores soluções e, para as instâncias onde a proporção entre as demandas dos serviços e as capacidades dos servidores é maior, reduziu o tempo computacional para a heurística proposta.

Estes resultados estão de acordo com o que se deseja ao tratar do gerenciamento energético e da configuração de *clusters* de grande escala, tendo em vista que em tais infraestruturas computacionais há uma grande quantidade de serviços que deve ser executada. O número de serviços, bem como suas demandas, pode variar ao longo do tempo e, por isso, o *cluster* deve ser reconfigurado periodicamente, onde o tempo necessário para obter uma boa solução correspondente à nova configuração deve ser de apenas alguns segundos.

Como trabalhos futuros sugere-se a realização de simulações com eventos de variação dinâmica das demandas dos serviços em intervalos de tempo de forma que seja necessário reconfigurar o *cluster* periodicamente. Desta forma, o algoritmo pode ser avaliado em situações mais próximas da realidade.

Sugere-se ainda a extensão da abordagem de otimização para tratar outros tipos de demandas para os serviços, o que incluiria uma nova dimensão ao problema. Por exemplo, no caso específico de serviços com maior intensidade de operações de E/S, seria interessante considerar as decisões de armazenamento e alocação de memória RAM, que também variam ao longo do tempo. Além disso, há uma tendência recente para que se permita tratar a eficiência energética em servidores com arquiteturas *multi-core*, considerando a aplicação de DVFS independentemente a cada núcleo do processador (Kim *et al.*, 2008), implicando em uma maior gama de configurações possíveis para melhor otimização energética dos servidores.

Agradecimentos

Os autores agradecem aos revisores pelos comentários e sugestões que ajudaram a melhorar a qualidade do trabalho. Agradecem também à CAPES, ao CNPq e à FAPERJ (processo E-26/110.550/2010) pelo apoio financeiro.

Referências

Alves, C. e Valério de Carvalho, J. M. (2007), Accelerating column generation for variable sized bin-packing problems. *European Journal of Operational Research*, v. 183, n. 3, p. 1333–1352.

Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M. e Pentikousis, K. (2010), Energy-efficient cloud computing. *The Computer Journal*, v. 53, n. 7, p. 1045.

Bertini, L., Leite, J. e Mossé, D. (2007), Statistical QoS guarantee and energy-efficiency in web server clusters. *Euromicro Conference on Real-Time Systems*, p. 83–92.

Bianchini, R. e Rajamony, R. (2004), Power and energy management for server systems. *Computer*, v. 37, n. 11, p. 68–76.

Bichler, M., Setzer, T. e Speitkamp, B. (2006), Capacity planning for virtualized servers. *Workshop on Information Technologies and Systems (WITS), Milwaukee, Wisconsin, USA*.

Cardellini, V., Casalicchio, E., Colajanni, M. e Yu, P. S. (2002), The state of the art in locally distributed web-server systems. *ACM Computing Surveys*, v. 34, n. 2, p. 263–311.

Dantzig, G. B. e Wolfe, P. (1960), Decomposition principle for linear programs. *Operations research*, v. 8, n. 1, p. 101–111.

Elnozahy, E. N., Kistler, M. e Rajamony, R. Energy-efficient server clusters. *Power-Aware Computer Systems*, volume 2325 of *Lecture Notes in Computer Science*, p. 179–197, 2003.

Fan, X., Weber, W.-D. e Barroso, L. A. Power provisioning for a warehouse-sized computer. *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, p. 13–23, New York. ACM, 2007.

Friesen, D. K. e Langston, M. A. (1986), Variable sized bin packing. *SIAM journal on computing*, v. 15, p. 222.

Gilmore, P. C. e Gomory, R. E. (1963), A linear programming approach to the cutting stock problem-Part II. *Operations research*, v. 11, n. 6, p. 863–888.

Hayes, B. (2008), Cloud computing. *Communications of the ACM*, v. 51, n. 7, p. 9–11.

Kaplan, J., Forrest, W. e Kindler, N. (2008), Revolutionizing data center energy efficiency. *McKinsey & Company, Relatório Técnico*.

Khanna, G., Beaty, K., Kar, G. e Kochut, A. (2006), Application performance management in virtualized server environments. *10th IEEE/IFIP Network Operations and Management Symposium*, p. 373–381.

Kim, W., Gupta, M. S., Wei, G.-Y. e Brooks, D. System level analysis of fast, per-core DVFS using on-chip switching regulators. *International Symposium on High-Performance Computer Architecture*, p. 123–134. IEEE Computer Society, 2008.

Kiyanclar, N. (2005), A survey of virtualization techniques focusing on secure on-demand cluster computing. *Arxiv preprint cs.OS/0511010*.

Petrucci, V., Carrera, E. V., Loques, O., Leite, J. e Mossé, D. Optimized management of power and performance for virtualized heterogeneous server clusters. *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid (CCGrid'11)*, 2011.

Pisinger, D. (1997), A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, v. 45, n. 5, p. 758–767.

Pisinger, D. (2000), A minimal algorithm for the bounded knapsack problem. *INFORMS Journal on Computing*, v. 12, n. 1, p. 75.

Ranganathan, P. (2010), Recipe for efficiency: principles of power-aware computing. *Communications of the ACM*, v. 53, n. 4, p. 60–67.

Rusu, C., Ferreira, A., Scordino, C., Watson, A. e Mossé, D. Energy-efficient real-time heterogeneous server clusters. *Real-Time and Embedded Technology and Applications Symposium, 2006. Proceedings of the 12th IEEE*, p. 418–428. IEEE, 2006.

Srikantaiah, S., Kansal, A. e Zhao, F. Energy aware consolidation for cloud computing. *Proceedings of the 2008 conference on power aware computing and systems*, HotPower'08, p. 10–10, Berkeley. USENIX Association, 2008.

Valério de Carvalho, J. M. (2005), Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, v. 17, n. 2, p. 175.

Wang, Y., Wang, X., Chen, M. e Zhu, X. Power-efficient response time guarantees for virtualized enterprise servers. *Proceedings of the 2008 Real-Time Systems Symposium*, p. 303–312, Washington, DC, USA. IEEE Computer Society, 2008.

Wang, Y. e Wang, X. Power optimization with performance assurance for multi-tier applications in virtualized data centers. *Proceedings of the 2010 International Conference on Parallel Processing Workshops*, p. 512–519, Los Alamitos, CA, USA. IEEE Computer Society, 2010.