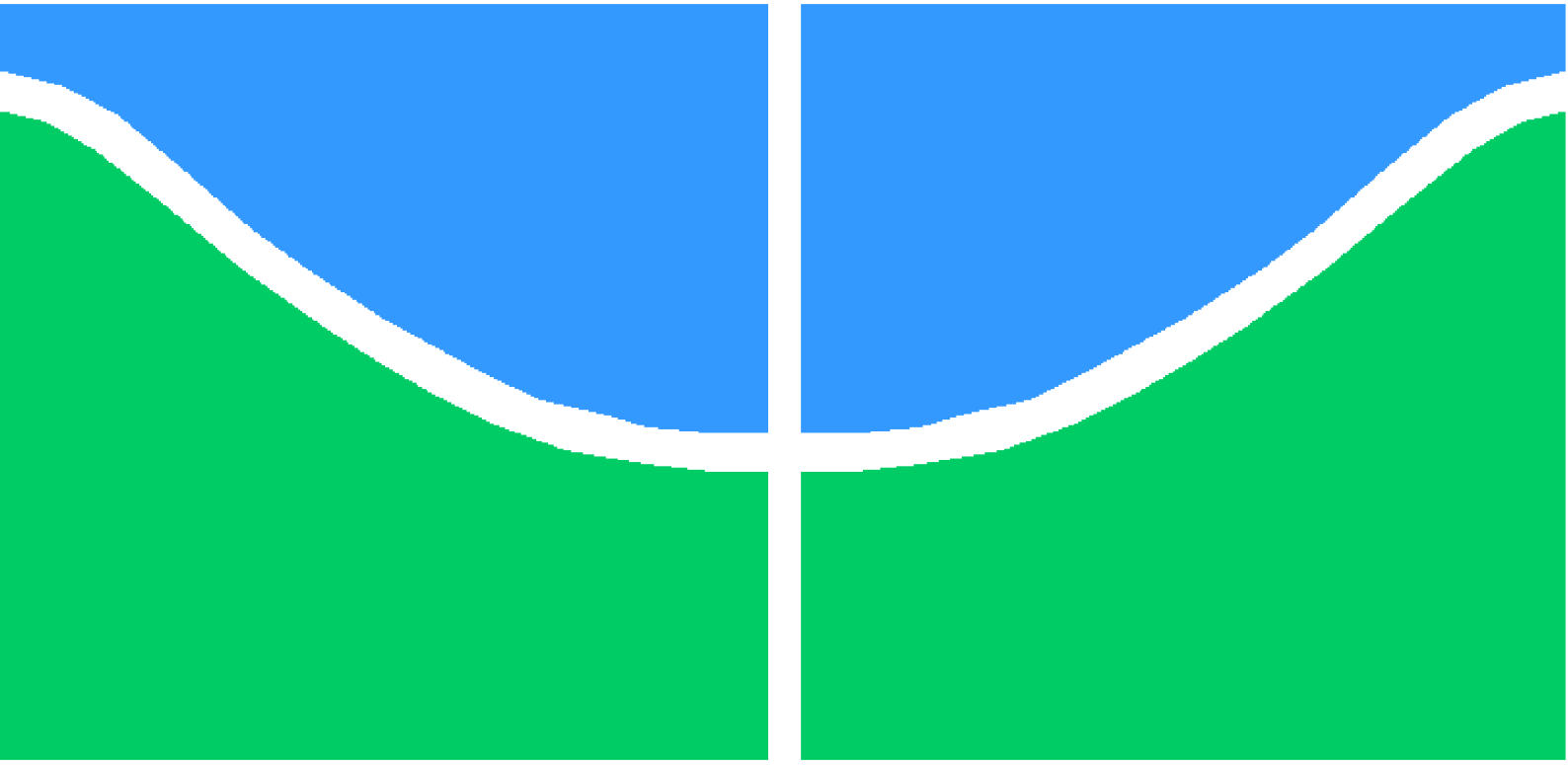

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Uma abordagem pseudo-Booleana para consolidação de máquinas virtuais com restrições de afinidade

Autor: Ulysses Bernard Mendes Lara

Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF

2020

Ulysses Bernard Mendes Lara

Uma abordagem pseudo-Booleana para consolidação de máquinas virtuais com restrições de afinidade

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF

2020

Resumo

Consolidação de máquinas virtuais é um problema que tenta alocar de maneira eficiente máquinas virtuais em um conjunto de hardwares, com o objetivo de minimizar a quantidade de hardware ligado. Por essa razão, esse tipo de pesquisa é bastante necessária para empresas que provêm computação em nuvem.

Uma questão que é interessante de ser levada em conta nas abordagens de consolidação de máquinas virtuais é a de como se comportam máquinas virtuais que estão em mesmo hardware, pois há a possibilidade de o agrupamento delas competir pelos recursos do hardware, prejudicando a qualidade do serviço prestado por elas.

O objetivo desse trabalho é estender uma abordagem já existente que usa uma estratégia pseudo-Booleana para consolidação de máquinas virtuais, para que sejam contempladas regras adicionais relacionadas a conflitos de máquinas virtuais, que ocorrem quando dois ou mais tipos de máquinas disputam pelos mesmos recursos quando alocadas em mesmo hardware.

Palavras-chaves: Consolidação de máquinas virtuais, SAT, pseudo-Boolean, lógica proposicional, afinidade de aplicações.

Abstract

Virtual machine consolidation is a problem that tries to place virtual machines in a more energy-efficient way, reducing the amount of required hardware. For that reason, techniques for virtual machine consolidation are widely required for data center services.

This problem of Virtual machine consolidation doesn't have a trivial solution, and when we try to expand this problem with conflicts inside the workloads of the virtual machines, we have a more complex problem.

The approach of this paper is to use a well-defined work of Virtual machine consolidation that uses a pseudo-Boolean formulation and extends it with new rules to avoid conflicts that can occur when two or more types of virtual machine applications compete for the same hardware resource.

Key-words: pseudo boolean, boolean satisfiability, virtual machines consolidation.

Lista de ilustrações

Figura 1 – Exemplo 1 VMs em hardware 9
Figura 2 – Exemplo 2 VMs em hardware 10
Figura 3 – Gráficos de número de restrições por abordagem 31
Figura 4 – Gráficos de número de restrições por abordagem 31

Lista de tabelas

Tabela 1 – Hardware e Máquinas Virtuais 16
Tabela 2 – Hardware 16
Tabela 3 – Máquinas Virtuais 16
Tabela 4 – Hardware 19
Tabela 5 – Máquinas Virtuais 19
Tabela 6 – Máquinas Virtuais com Classes 24
Tabela 7 – Tabelas com resultados dos experimentos 32
Tabela 8 – Tabelas com número de variáveis e restrições geradas 33

Sumário

1 INTRODUÇÃO 8

Introdução 8
1.1 Problemática 8
1.1.1 Recursos Físicos 9
1.1.2 Afinidade Entre Aplicações 9
1.2 Considerações 11

2 REFERENCIAL TEÓRICO 12
2.1 Lógica Proposicional 12
2.2 Satisfatibilidade 12
2.3 Pseudo-Boolean 13

3 TRABALHOS CORRELATOS 14
3.1 Primeira Formulação Pseudo-Booleana 14
3.1.1 Fórmula Objetivo 15
3.1.2 Recurso Necessário Disponível 15
3.1.3 Limitar Recurso Por Hardware 15
3.1.4 Uma Máquina Virtual Por Hardware 16
3.1.5 Exemplo 16
3.2 PBFVMC 17
3.2.1 Exemplo 19
3.3 Afinidade De Aplicações 20

4 PBFVMC - AFFINITY 22
4.1 Abordagem Totalmente Restritiva 22
4.1.1 Exemplo 24
4.2 Abordagem Parcialmente Restritiva 25
4.2.1 Exemplo 26
4.3 Minimização de Conflitos 27
4.4 Considerações 28

5 EXPERIMENTOS 29
5.1 Resultados 29

6 CONCLUSÃO 34

REFERÊNCIAS 35

1 Introdução

O conceito de computação em nuvem está difundido como um novo paradigma para o desenvolvimento de sistemas. Esse paradigma propõe a execução e o armazenamento por meio da internet em vez de computadores locais (MA, 2012).

Uma das abordagens adotadas pela programação em nuvem é a *Infrastructure-as-a-service* ou Infraestrutura como um serviço (IaaS). Essa designação é feita para definir um serviço executado na nuvem, feito através de máquinas virtuais que rodam em *clusters*, nos quais uma máquina pode possuir várias máquinas virtuais (VMs) (TERRA-NEVES; LYNCE; MANQUINHO, 2018).

Um dos problemas enfrentados pelos provedores da computação em nuvem é o alto impacto financeiro e ambiental acarretado por deixar as máquinas ligadas e os seus respectivos sistemas de refrigeração, já que esse é um dos principais custos dos *datacenters* responsáveis por esse serviço. (TERRA-NEVES; LYNCE; MANQUINHO, 2018).

Nesta perspectiva, uma das grandes áreas de pesquisa é a melhor alocação de máquinas virtuais (ZHENG et al., 2013), que visa minimizar o número de máquinas ligadas. Para essa finalidade, há várias limitações físicas que devem ser levadas em consideração, como a quantidade de memória RAM, processamento, barramento da internet, dentre outras que se apresentam em determinados contextos.

Uma das abordagens utilizadas para a alocação de máquinas virtuais é a consolidação de máquinas virtuais que visa a realocação de máquinas virtuais distribuídas em vários servidores para um menor número de hardware possível (RIBAS et al., 2013). Por meio de uma alocação eficiente é possível ter uma economia de até 75% em alguns casos.

No entanto, para os clientes do serviço, nem sempre é vantajoso para uma VM estar em execução juntamente com alguns tipos de VMs. Litch(2014) fez um trabalho sobre a afinidade entre diversos tipos de aplicações e identificou que algumas classes podem interferir no desempenho de outra aplicação, mesmo quando isoladas em virtualização, causando a degradação de desempenho dessas máquinas.

1.1 Problemática

Para abranger a problemática envolvida na consolidação de máquinas virtuais deve-se levar em conta os limites físicos e algumas regras arbitrárias no quesito de qualidade do serviço que serão elucidadas no decorrer do capítulo.

Para alcançar a consolidação de máquinas virtuais foi feita a análise do problema

em duas partes, uma visando a melhor alocação dos recursos físicos e outra corrigindo conflitos que essa alocação possa acarretar.

1.1.1 Recursos Físicos

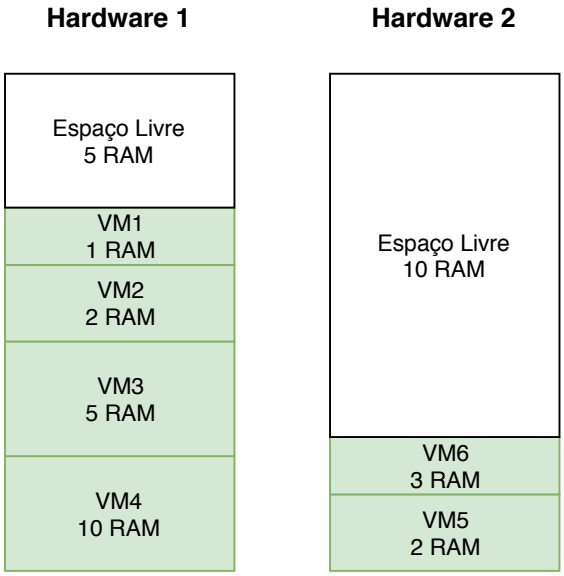

Figura 1 – Exemplo 1 VMs em hardware

O problema do mal gerenciamento dos recursos físicos acarreta custos energéticos em um provedor nuvem, pois esse mau gerenciamento torna máquinas que deveriam estar inativas em máquinas ativas.

Na distribuição de máquinas virtuais em hardwares deve ser levado em conta capacidades físicas das máquinas, como a memória RAM, processamento, capacidade de armazenamento entre outros, porém cada uma das VMs alocadas possuem necessidades diferentes para serem executadas. A distribuição dessas VMs, caso feita de forma não otimizada, acarreta em um maior número de hardware ligado.

Como pode ser visto na Figura 1, que mostra uma visão simplificada contemplando apenas memória RAM, seria possível desligar o hardware 2 passando as VM5 e VM6 para o hardware 1, desse modo economizando a energia consumida.

Com a redução de hardware ativo o consumo energético por sua vez tende a cair, tornando a hospedagem mais sustentável tanto financeiramente.

1.1.2 Afinidade Entre Aplicações

O problema de afinidade entre aplicações consiste em aplicações que disputam algum recurso da máquina física, o que provoca danos para aplicação, de velocidade ou

até mesmo de indisponibilidade da mesma(LICHT, 2014), afetando diretamente a solução de gerenciar os recursos físicos.

Por exemplo uma aplicação com intensidade de processamento como uma aplicações cpu-bound, onde se depende muito do processador e pouco das entradas e saídas, ao serem colocadas juntas em um mesmo hardware podem competir pelo uso do processador, causando perda na qualidade das aplicações.

Entretanto nesses conflitos é importante ressaltar que o consumo de recurso de uma aplicação não se mantém padrão, por exemplo uma máquina classificada como cpu-bound, não significa que ela tem intensidade do uso do processamento em toda sua execução, há a possibilidade de o comportamento dela variar em determinado período do seu fluxo de trabalho(SCHAD; DITTRICH; QUIANé-RUIZ, 2010).

A variância de comportamento de aplicações gera um obstáculo vigente à classificação das aplicações que irão rodar na máquina virtual, pois é necessário determinar o potencial conflitante entre aplicações.

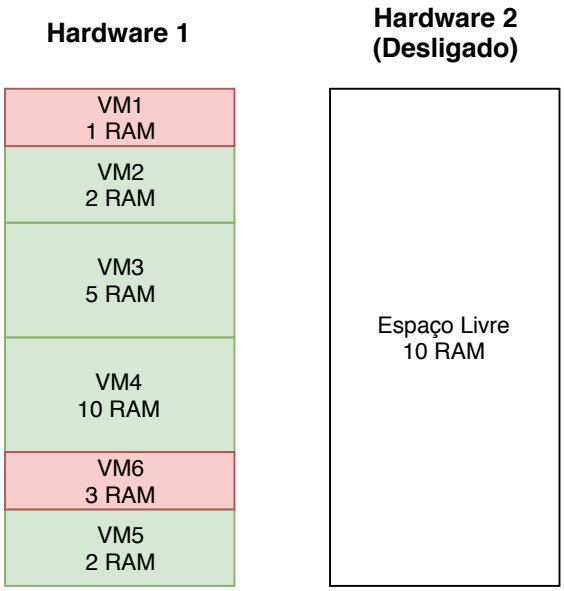

Figura 2 – Exemplo 2 VMs em hardware

No caso do escopo da Subseção 1.1.1 o problema se intensifica, pois uma solução seria colocar o número máximo de máquinas virtuais em menos recursos físicos ativos, no entanto isso aumenta a probabilidade de que ocorra uma distribuição em que máquinas disputam os recursos.

A Figura 2 mostra uma solução de consolidação de máquinas virtuais, porém, com as VMs 1 e 6 sendo conflitantes, mostrando que a solução pode acarretar perda na qualidade das aplicações que rodam nessas VMs.

1.2 Considerações

O objetivo deste trabalho está em criar uma extensão para uma formulação já conhecida de consolidação de máquinas virtuais, a PBFVMC (RIBAS et al., 2013), a fim de codificar restrições que visem considerar a afinidade entre diversas máquinas virtuais.

Este trabalho está dividido da seguinte forma: No Capítulo 2 é apresentado o referencial teórico básico necessário para o trabalho. No Capítulo 3 os trabalhos correlatos, com a apresentação da formulação PBFVMC e o problema de afinidade, são apresentados. No Capítulo 4 formulamos a extensão da PBFVMC, seguida do Capítulo 5 apontando os resultados obtidos com a nova formulação. Por fim, no Capítulo 6, finalizamos o trabalho com a conclusão e trabalhos futuros.

2 Referencial Teórico

Esse capítulo é destinado ao levantamento teórico da solução, essa teoria será usada como ferramenta para a proposta do trabalho. Primeiramente uma breve revisão sobre lógica proposicional, em seguida introduzindo o conceito de satisfatibilidade (FRANCO; MARTIN, 2009) e de pseudo-Boolean (ROUSSEL; MANQUINHO, 2009).

2.1 Lógica Proposicional

A lógica proposicional é um sistema formal destinado para a representação de proposições e sentenças declarativas, que podem possuir valores verdadeiros ou falsos.

Uma lógica proposicional possui regras estabelecidas para sua formulação mediante utilização de conectivos lógicos para seu desenvolvimento. Os conectivos permitidos são $\wedge$ (e), $\vee$ (ou), $\neg$ (negação), $\Rightarrow$ (se...então), $\iff$ (se e somente se). Por meio destes podemos construir sentenças compostas, elaboradas a partir da junção de sentenças com os conectores mencionados, dando possibilidade de representação das expressões lógicas mais complexas.

2.2 Satisfatibilidade

A satisfatibilidade Booleana (SAT) é um problema NP-completo. Onde dada uma expressão lógica com variáveis Booleanas, o problema tenta decidir se há uma valoração verdadeira para as variáveis que torna a expressão verdadeira.

Nesses quesitos, quando a formulação é dita como satisfeita é quando ocorre que o conjunto da formulação é dado como verdadeiro, e não satisfeita quando é falso.

Exemplo:

Dadas as variáveis x_1, x_2 na expressão lógica $y = x_1 \wedge x_2$

x_1	x_2	y	tipo
0	0	0	solução não satisfeita
0	1	0	solução não satisfeita
1	0	0	solução não satisfeita
1	1	1	solução satisfeita

Para a resolução SAT, foram desenvolvidos programas especializados que decidem soluções para uma lógica proposicional, utilizando métodos avançados para resolução

rápida do problema. Isso é visto com grande valor na computação por serem problemas NP-completos.

2.3 Pseudo-Boolean

As funções pseudo-Booleanas têm o intuito de definir uma expressão de desigualdade com constantes e variáveis booleanas para representar uma expressão lógica. Por mesclar o uso de números inteiros e variáveis booleanas ela se difere de uma restrição estritamente booleana, por essa razão é dada a nomenclatura pseudo-Booleana.

A correspondência dos valores Booleanos com valores reais ocorre pelo uso de variáveis Booleanas junto a restrições de desigualdade linear e coeficientes inteiros agindo como um peso nas variáveis Booleanas.

O formato de uma restrição Booleana linear tem o formato:

$$\sum_{j=1}^{N} c_j \cdot x_j \triangleright k$$

Nessa representação temos c_j e k como constantes inteiras, x_j como uma variável booleana e o $\triangleright$ representando operadores relacionais $(<, \leq, >, \geq, =)$, dessa forma quando o resultado da disjunção é verdadeiro a restrição é dita como satisfeita.

Exemplo. *Uma restrição plausível é:*

$$3 \cdot x_1 + 11 \cdot x_2 + 20 \cdot x_3 + 12 \cdot x_4 < 12$$

É vista como satisfeita quando $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0$ *ou* $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 0$ *ou* $x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0$ *já que com esses valores a disjunção se torna verdadeira.*

Uma das vantagens do uso das funções pseudo-Booleanas está na representação mais expressiva e natural para a modelagem de problemas reais, além de poder também se beneficiar da otimização trazidas pelos resolvedores SAT.

3 Trabalhos Correlatos

Este capítulo visa compreender os estudos que servem como base para o desenvolvimento deste trabalho.

Pela proposta de resolução do problema se basear em técnicas pseudo-Booleanas e otimização por restrições, o capítulo trabalhará com dois trabalhos de consolidação de máquinas virtuais, esse trabalhos são divididos em duas formulações que visam encontrar uma decisão satisfatória para o problema em menor tempo hábil. Por esse motivo a estrutura deste capítulo contemplará uma primeira formulação (RIBAS et al., 2012) e após isso será analisada a evolução da formulação que se encontra no trabalho. (RIBAS et al., 2013).

Outro trabalho que será abordado nesse capítulo diz respeito a conflitos entre aplicações de máquinas virtuais (LICHT, 2014). As ideias expressas nesses trabalhos serão usados como base para o desenvolvimento da consolidação de máquinas virtuais com afinidade de classes.

3.1 Primeira Formulação Pseudo-Booleana

Na solução proposta no artigo (RIBAS et al., 2012) é definido seis conjuntos de restrições para minimizar o uso de máquinas ligadas, a obtenção dessas fórmulas foram feitas através da abstração do problema.

Como explicado na Seção 2.3 sobre lógica pseudo-Booleana, para representar um problema através dessa lógica é necessário abstrair o problema em variáveis booleanas e disjunções de grandeza. No caso dessa primeira formulação da consolidação de máquinas virtuais queremos colocar k $\{vm_1...vm_k\}$ máquinas virtuais em n $\{vm_1...vm_k\}$ hardwares, tentando minimizar o número de hardwares ligados.

Para iniciar a formulação primeiramente é necessário definir as variáveis a serem utilizadas, essas variáveis têm a intenção de abstrair alguns objetos do problema, no caso dessa formulação tem a necessidade da representação dos hardwares e das máquinas virtuais. Para os hardwares foi adotado uma representação como hw_i^p e hw_i^r onde essas variáveis representam o processamento e a RAM respectivamente do i-ésimo hardware, gerando assim $(2 \cdot n)$ variáveis para representar os hardware.

Para as máquinas virtuais o processo é similar como o que ocorre nos hardware, as variáveis são representadas como $vm_j^{p \cdot hw_i}$ e $vm_j^{r \cdot hw_i}$, representando o CPU e RAM respectivamente, necessário para a j-ésima VM alocada no i-ésimo hardware. Sendo assim o números de variáveis para as máquinas virtuais é $(2 \cdot n \cdot k)$.

Portanto a quantidade de variáveis total do problema é $(2 \cdot n + 2 \cdot n \cdot k)$ sendo n a quantidade de hardware e k a quantidade de máquinas virtuais.

Já as constantes que serão usadas nessa abordagem são RAM_{hwi} $PROC_{hwi}$, que são o número de RAM e cpu que do i-ésimo hardware, e RAM_{vm_j} $PROC_{vm_j}$, que são as quantidades necessárias de RAM e cpu necessária para a j-ésima VM.

3.1.1 Fórmula Objetivo

A primeira fórmula definida diz respeito ao objetivo do problema que é minimizar o número de máquinas físicas ligadas e matematicamente é expressa por:

$$min : \sum_{i=1}^{n} hw_i \tag{3.1}$$

3.1.2 Recurso Necessário Disponível

A segunda e terceira fórmula proposta dizem respeito a verificação se a quantidade de RAM e processamento das máquinas somadas equivale ao somatório da quantidade respectiva das máquinas virtuais.

$$\sum_{i=1}^{n} RAM_{hw_i}.hw_i^{ram} \geq \sum_{j=1}^{n} RAM_{vm_j} \cdot vm_j^{ram} \tag{3.2}$$

$$\sum_{i=1}^{n} PROC_{hw_i}.hw_i^{proc} \geq \sum_{j=1}^{n} PROC_{vm_j} \cdot vm_j^{proc} \tag{3.3}$$

3.1.3 Limitar Recurso Por Hardware

As duas restrições definidas aqui dizem respeito à limitação do uso do equipamento físico pelas VMs, ela faz com que um hardware não estoure sua capacidade de alocar máquinas virtuais nele.

$$\forall i \in 1..n(\sum_{j=1}^{k} RAM_{vm_j} \cdot vm_j^{ram \cdot hw_i} \leq RAM_{hw_i}) \tag{3.4}$$

$$\forall i \in 1..n(\sum_{j=1}^{k} PROC_{vm_j} \cdot vm_j^{proc \cdot hw_i} \leq PROC_{hw_i}) \tag{3.5}$$

3.1.4 Uma Máquina Virtual Por Hardware

A última restrição proposta é elaborada com o intuito de restringir o uso de uma máquina virtual por hardware.

$$\forall i \in 1..n(\sum_{j=1}^{n} vm_i^{proc \cdot hwj} \cdot vm_i^{ram \cdot hwj} \cdot hw_j^{proc} \cdot hw_j^{ram} = 1) \tag{3.6}$$

Analisando as restrições apresentadas podemos perceber que a quantidade gerada é $(2 + 2 \cdot n + k)$ restrições.

3.1.5 Exemplo

Nessa sub-seção será mostrado um exemplo da construção das fórmulas da primeira formulação dado um conjunto de hardwares e máquinas virtuais.

Sendo estes:

Tabela 1 – Hardware e Máquinas Virtuais

Tabela 2 – Hardware

Nome	RAM	Processamento
hw_1	100	20
hw_2	60	10

Tabela 3 – Máquinas Virtuais

Nome	RAM	Processamento
vm_1	50	5
vm_2	30	20
vm_3	10	5

A partir desses dados geramos as restrições citadas anteriormente, onde para Equação 3.1 temos:

$$min: \quad hw_1 + hw_2$$

para 3.2 e 3.3:

$$100 \cdot hw_1^{ram} + 60 \cdot hw_2^{ram} \geq 90$$
$$20 \cdot hw_1^{proc} + 10 \cdot hw_2^{proc} \geq 30$$

Equação 3.4:

$$50 \cdot vm_1^{ram \cdot hw_1} + 30 \cdot vm_2^{ram \cdot hw_1} + 10 \cdot vm_3^{ram \cdot hw_1} \leq 100 \tag{3.7a}$$
$$50 \cdot vm_1^{ram \cdot hw_2} + 30 \cdot vm_2^{ram \cdot hw_2} + 10 \cdot vm_3^{ram \cdot hw_2} \leq 60 \tag{3.7b}$$

Equação 3.5:

$$5 \cdot vm_1^{proc \cdot hw_1} + 20 \cdot vm_2^{proc \cdot hw_1} + 5 \cdot vm_3^{proc \cdot hw_1} \leq 20 \tag{3.8a}$$
$$5 \cdot vm_1^{proc \cdot hw_2} + 20 \cdot vm_2^{proc \cdot hw_2} + 5 \cdot vm_3^{proc \cdot hw_2} \leq 10 \tag{3.8b}$$

Equação 3.6:

$$vm_1^{proc \cdot hw_1} \cdot vm_1^{ram \cdot hw_1} \cdot hw_1^{proc} \cdot hw_1^{ram} + vm_1^{proc \cdot hw_2} \cdot vm_1^{ram \cdot hw_2} \cdot hw_2^{proc} \cdot hw_2^{ram} = 1 \tag{3.9a}$$
$$vm_2^{proc \cdot hw_1} \cdot vm_2^{ram \cdot hw_1} \cdot hw_1^{proc} \cdot hw_1^{ram} + vm_2^{proc \cdot hw_2} \cdot vm_2^{ram \cdot hw_2} \cdot hw_2^{proc} \cdot hw_2^{ram} = 1 \tag{3.9b}$$
$$vm_3^{proc \cdot hw_1} \cdot vm_3^{ram \cdot hw_1} \cdot hw_1^{proc} \cdot hw_1^{ram} + vm_3^{proc \cdot hw_2} \cdot vm_3^{ram \cdot hw_2} \cdot hw_2^{proc} \cdot hw_2^{ram} = 1 \tag{3.9c}$$

Para que essas fórmulas sejam satisfeitas é necessário que as variáveis $vm_1^{proc*hw_2}$ $vm_2^{proc*hw_1}$, $vm_3^{proc*hw_2}$, $vm_1^{ram*hw_2}$, $vm_2^{ram*hw_1}$, $vm_3^{ram*hw_2}$, hw_1, hw_2 sejam verdadeiras, mostrando que a minimização maior encontrada é de 2, tendo que para esse conjunto, todos os hardwares devem ficar ligados para uma solução satisfatória.

3.2 PBFVMC

O modelo PBFVMC (RIBAS et al., 2013; RIBAS, 2015) propõe a consolidação de máquinas virtuais por meio de um modelo pseudo-Boolenao levando em consideração a capacidade de memória RAM e CPU dos *hardware* e das VMs a serem alocadas.

Para representar os aspectos do problema de consolidação de máquinas virtuais em um método de resolução pseudo-Booleana, é necessário que seja, de algum modo, representados em formas de variáveis os elementos do problema, o modo encontrado de representação nessa solução foi:

- n - Número total de *hardware*.
- k - Número total de Máquinas virtuais.
- hw_i - i-ésimo *hardware* i, onde i é o índice.
- $vm_j^{hw_i}$ - j-ésimo VM j rodando no *hardware* i, onde i e j são o índice dos *hardware* e das VMs respectivamente.

O número gerado de variáveis dessa formulação é $(n + n \cdot k)$ variáveis, isso ocorre pois é destinado uma variável para cada hardware $(n$ $)$e cada máquina virtual tem uma variável para representá-la em cada hardware $(n \cdot k)$.

Para a representação do PBFVMC também é necessário a definição de algumas constantes, sendo elas:

- Ram_{hwi} - memória RAM do i-ésimo *hardware* i, onde i é o iterador.
- $Proc_{hwi}$ - CPU do i-ésimo *hardware* i, onde i é o iterador.
- Ram_{vm_j} - memória RAM da j-ésima VM.
- $Proc_{vm_j}$ - CPU da j-ésima VM.

O número gerado de variáveis dessa formulação é $(n + n \cdot k)$ variáveis, isso ocorre pois há uma variável para cada hardware (n) e cada máquina virtual tem uma variável para representá-la em cada hardware $(n \cdot k)$.

A primeira restrição a ser estruturada na formulação é a função objetivo que visa a redução dos números de *hardware* ligados.

$$min : \sum_{i=1}^{n} hw_i \tag{3.10}$$

As duas fórmulas a seguir restringem que o somatório dos recursos dos *hardware* ligados acomodam os recursos utilizados pela máquinas virtuais.

$$\sum_{i=1}^{n} Ram_{hw_i} \cdot hw_i \geq \sum_{j=1}^{k} Ram_{vm_j} \tag{3.11}$$

$$\sum_{i=1}^{n} Proc_{hw_i} \cdot hw_i \geq \sum_{j=1}^{k} Proc_{vm_j} \tag{3.12}$$

As restrições a seguir dizem respeito ao quanto de recurso as máquinas podem prover às máquinas virtuais.

$$\bigvee i \in 1..n \ (\sum_{j=1}^{k} (Ram_{vm_j} \cdot \neg vm_j^{hw_i}) + Ram_{hw_i} \cdot hw_i \geq \sum_{j=1}^{k} Ram_{vm_j}) \tag{3.13}$$

$$\bigvee i \in 1..n \ (\sum_{j=1}^{k} (Proc_{vm_j} \cdot \neg vm_j^{hw_i}) + Proc_{hw_i} \cdot hw_i \geq \sum_{j=1}^{k} Proc_{vm_j}) \tag{3.14}$$

A restrição abaixo define que cada máquina virtual deve estar alocada a um *hardware*.

$$\forall i \in 1..k \ \sum_{j=1}^{n} (vm_j^{hw_i} \geq 1) \tag{3.15}$$

E, por fim, a última restrição garante que uma máquina virtual possua alocação em apenas um *hardware*, não podendo dividir os recursos da VM para distribuição.

$$\forall j \in 1..k \ (\sum_{i=1}^{N} \neg vm_j^{hw_i} \geq n - 1) \tag{3.16}$$

Analisando essas restrições vemos que é gerado $2 + 2 \cdot n + 2 \cdot k$ restrições na formulação PBFVMC.

3.2.1 Exemplo

Nesta seção, codificamos uma fórmula utilizando as informações de *hardware* descritas na tabela 4 e com as informações de virtuais na Tabela 5.

Tabela 4 – Hardware

Nome	RAM	Processamento
hw_1	100	20
hw_2	60	10

Tabela 5 – Máquinas Virtuais

Nome	RAM	Processamento
vm_1	50	5
vm_2	30	20
vm_3	10	5

Para a restrição 3.10:

$$min: \quad hw_1 + hw_2 \tag{3.17a}$$

Para as restrições 3.11 e 3.12, temos:

$$100 \cdot hw_1 + 60 \cdot hw_2 \geq 90 \tag{3.18a}$$

$$20 \cdot hw_1 + 10 \cdot hw_2 \geq 30 \tag{3.18b}$$

Para as restrições 3.13 e 3.14 o seguinte conjunto é gerado:

$$(50 \cdot \neg vm_1^{hw1}) + (30 \cdot \neg vm_2^{hw1}) + (10 \cdot \neg vm_3^{hw1}) + 100 \cdot hw_1 \geq 90 \tag{3.19a}$$
$$(50 \cdot \neg vm_1^{hw2}) + (30 \cdot \neg vm_2^{hw2}) + (10 \cdot \neg vm_3^{hw2}) + 60 \cdot hw_2 \geq 90 \tag{3.19b}$$
$$(5 \cdot \neg vm_1^{hw1}) + (20 \cdot \neg vm_2^{hw1}) + (5 \cdot \neg vm_3^{hw1}) + 20 \cdot hw_1 \geq 30 \tag{3.19c}$$
$$(5 \cdot \neg vm_1^{hw2}) + (20 \cdot \neg vm_2^{hw2}) + (5 \cdot \neg vm_3^{hw2}) + 10 \cdot hw_2 \geq 30 \tag{3.19d}$$

Finalmente codificamos a restrição 3.15 como:

$$vm_1^{hw_1} + vm_2^{hw_1} + vm_3^{hw_1} \geq 1 \tag{3.20a}$$
$$vm_1^{hw_2} + vm_2^{hw_2} + vm_3^{hw_2} \geq 1 \tag{3.20b}$$

Para a restrição 3.16:

$$\neg vm_1^{hw_1} + \neg vm_1^{hw_2} \geq 1 \tag{3.21a}$$
$$\neg vm_2^{hw_1} + \neg vm_2^{hw_2} \geq 1 \tag{3.21b}$$
$$\neg vm_3^{hw_1} + \neg vm_3^{hw_2} \geq 1 \tag{3.21c}$$

Para essa formulação ser satisfeita, as variáveis a serem atribuídas com verdadeiro são as vm_1^{hw2}, vm_2^{hw1}, vm_3^{hw2}, hw_1, hw_2, o que faz com que a vm_1 e a vm_3 sejam alocadas em hw_2, e a vm_2 em hw_1.

3.3 Afinidade De Aplicações

Um estudo interessante que contempla afinidade de aplicações é o *Afinidades de tipos de aplicações em nuvens computacionais* (LICHT, 2014) nesse artigo o autor utiliza divisões de classe existentes para realizar o estudo de como diferentes aplicações deterioram suas performances quando colocada em mesmo *hardware* com outra aplicação de característica conflitante.

As deteriorações constatadas no estudo possuem impacto em todas aplicações hospedadas em determinado *hardware*, fato que ocorre pela concorrência entre recursos como CPU memória RAM, que dependendo dos conflitos acarretam em deteriorações brandas ou severas.

Nesse sentido o trabalho criou o conceito de classes para separar qual recurso computacional é o mais saturado no uso da aplicação, por exemplo, temos as classes: *cpu-bound*; *io-bound*, e; *memory-bound*, que representam, respectivamente, saturação no uso de processamento, acesso a dispositivos de entrada/saída e de memória.

Nesse estudo é mostrado que os conflitos constatados têm a possibilidade de ocorrer com aplicações de classes diferentes ou aplicações com classes iguais, o que nos leva a concluir que uma solução que envolva consolidação máquinas virtuais com conflitos entre aplicações deva diferenciar essas características.

4 PBFVMC - Affinity

Na modelagem apresentada na Seção 3.2 a formulação faz a otimização considerando apenas alocação das máquinas por quantidade de CPU e memória RAM, não obstante, essa abordagem desconsidera uma possibilidade que ocorre no mundo real: a concorrência entre máquinas virtuais, que pode acarretar prejuízos na performance das máquinas virtuais como apresentado na Seção 3.3.

Portanto, aproveitando a efetividade do modelo PBFVMC, podemos ampliá-lo para contemplar o problema de afinidades. Desse modo pode-se ter uma maior aplicação sem que as máquinas virtuais tenham o desempenho degradado. Para isso, novas restrições devem ser adicionadas, para que transmitam os requisitos de afinidade entre as classes de máquinas virtuais.

Para adicionar essa nova restrição não há a necessidade da adição de novas variáveis Booleanas, para isso nós devemos criar restrições que transmitam os comportamentos encontrados no problema de afinidade.

No entanto é necessário que as classes das virtuais sejam definidas logo de início por um especialista, foge do nosso escopo definir as classes às quais pertencem cada VM, e também delegamos ao usuário da fórmula a decisão sobre as afinidades entre as classes.

Portanto, para essa extensão proposta, uma classe é como um rótulo dado a uma máquina virtual, que diz com quem ela pode compartilhar recursos de um mesmo hardware.

4.1 Abordagem Totalmente Restritiva

A abordagem que será adotada aqui é uma abordagem em que não há a possibilidade de duas máquinas conflitantes ficarem juntas. Considerando esse aspecto, existem, dois casos a serem abordados, o primeiro dos casos a ser analisado é quando uma classe é conflitante com outra classe, denota-se como classe uma determinada aplicação com características que junto a outra, diferente ou de igual classe, em um mesmo *hardware*, pode prejudicar ou não a performance da aplicação.

Para a abstração dos conflitos, de modo que simplifique a restrição pseudo-Booleana é adotado o pressuposto de que os conflitos ocorrem entre dois tipos de classes, em caso de um conflito expresso no mundo real com três ou mais conflitos, a restrição é reestruturada para contemplar todas as duplas pertencentes nesse conjunto.

Exemplo. *Dado o conjunto de conflitos* $c = \{(A, B, C), (D, E, F)\}$ *este con-*

junto seria transformado para o seguinte conjunto $conflitos = \{(A,B),(B,C),(A,C),(D,E)(D,F),(E,F)\}$

Uma restrição que proíba essa coexistência de uma determinada classe A com uma classe B em um mesmo *hardware*, pode ser expressa como mostrado na restrição 4.1, as variáveis $vm_j^{hw_i \cdot clsA}$ e $vm_j^{hw_i \cdot clsB}$ são as mesmas mostradas no PBFVMC no Seção 3.2 mas filtrando as classes A e B respectivamente, já na parte de constantes, são adicionadas h e z que são os números de VMs da classe A e da classe B respectivamente.

Para representar os tipos de conflitos as restrições escritas nessa seção adotou a notação $\forall c \in (a,b)$ para definir a retirada de um par de conflito de diferente classe do conjunto de conflitos c e $\forall c \in (a,b)$ para retirada de conflitos iguais co conjunto c.

$$\forall i,k,c \in 1..n,1..z,(a,b)\ (\sum_{j=1}^{h} vm_j^{hw_i \cdot clsA}) + h \cdot vm_k^{hw_i \cdot clsB} \leq h \tag{4.1}$$

A restrição 4.1 funciona nesse caso de classes diferentes, pois faz com que o somatório das VM da classe A não possam ser ligados caso a $vm_k^{hw_i \cdot clsB}$ estiver ligada na mesma inequação, pois geraria um resultado maior que h.

Contudo, para deixar de acordo com o padrão pseudo-Booleano explicado na Subseção 2.3, mudamos a inequação de menor igual ($\leq$) para maior igual($\geq$), multiplicando tudo por -1, desse modo a restrição fica como mostrado na equação 4.2.

$$\forall i,k,c \in 1..n,1..z,(a,b)\ (\sum_{j=1}^{h} -vm_j^{hw_i \cdot clsA}) - h \cdot vm_k^{hw_i \cdot clsB} \geq -h \tag{4.2}$$

O segundo caso a ser estabelecido é quando uma classe se auto conflita, não podendo coexistir consigo mesma, nesse caso a restrição que contempla a solução pode ser expressa através da equação 4.4.

$$\forall i,c \in 1..n,(a,a)\ (\sum_{j=1}^{h} vm_j^{hw_i \cdot clsA}) \leq 1 \tag{4.3}$$

Com essa restrição vemos que o somatório das máquinas virtuais ligadas em determinado *hardware* não pode passar de 1 fazendo com que nenhuma classe fique junta com outra de igual classe.

Com essas duas fórmulas, há a possibilidade de otimizar as máquinas com as afinidades conflitantes, porém de uma forma inflexível, retornando não satisfatível caso não haja espaço suficiente para distribuir os conflitos. Todavia, como na restrição anterior é necessário padronizar multiplicando por -1 em ambos os lados da inequação.

$$\forall i \in 1..n \sum_{j=1}^{h} -vm_j^{hw_i \cdot clsA} \geq -1 \tag{4.4}$$

Com essa mudança na abordagem PBFVMC não há crescimento de variáveis porem ocorre o crescimento das restrições, o crescimento das restrições aumenta para $(n \cdot z)$ para conjunto de conflitos diferentes e (n) restrições para cada conjunto de conflito iguais, esse valores são acrescidas as quantidades de restrições do PBFVMC $(2n + 2k + 2)$.

4.1.1 Exemplo

Utilizando o mesmo exemplo utilizado da Seção 3.2.1, mas agora adicionando a categorização das classes.

Tabela 6 – Máquinas Virtuais com Classes

Nome	RAM	CPU	Classe
vm_1	50	5	A
vm_2	30	20	A
vm_3	10	5	B

As restrições relativas ao PBFVMC continuam as mesmas do exemplo 3.2.1, no entanto, com algumas regras adicionais sendo estas: Uma VM da classe A é conflitante com uma VM de classe A no mesmo hardware; Uma VM de Classe A conflita com uma VM de classe B no mesmo hardware;

conflita com outra VM de mesma classe ou com uma VM da classe B.

Assim usando as novas restrições temos:

$$Constantes: \qquad n = 2 \quad z = 1 \quad h = 1 \quad c = \{(A, B), (A, A)\}$$

Classe AxB

$$-1 \cdot vm_1^{hw_1} - 1 \cdot vm_2^{hw_1} - 1 \cdot vm_3^{hw_1} \geq -1 \tag{4.5a}$$
$$-1 \cdot vm_1^{hw_2} - 1 \cdot vm_2^{hw_2} - 1 \cdot vm_3^{hw_1} \geq -1 \tag{4.5b}$$

Classe AxA

Agora para a vm_1 e vm_2 que são da mesma classe (auto-conflito), nesse caso usamos a restrição 4.4:

$$-1 \cdot vm_1^{hw_1} - 1 \cdot vm_2^{hw_1} \geq -1 \tag{4.6a}$$
$$-1 \cdot vm_1^{hw_2} - 1 \cdot vm_2^{hw_2} \geq -1 \tag{4.6b}$$

No exemplo acima vemos que a solução não satisfaz a inequação, uma vez que não existe recurso físico para separar as VMs em outras máquinas.

4.2 Abordagem Parcialmente Restritiva

A solução anterior mostrada na Seção 4.1 tem como finalidade restringir VMs com características opostas de alocar no mesmo hardware, porém impossibilita que máquinas conflitantes fiquem juntas, o que pode levar, em alguns casos, a escassez de hardware, tornando a solução insatisfatível. Portanto, surge a necessidade de apenas minimizar os conflitos encontrados.

Uma solução possível seria aumentar a tolerância com uma variável T, de a inequação se mantém a mesma da Seção 4.1, de modo que apenas adicionamos T a inequação como mostrado na equação 4.7.

$$\forall i, k \in 1..n, 1..z(-\sum_{j=1}^{h} vm_j^{hw_i \cdot clsA}) - h \cdot vm_k^{hw_i \cdot clsB} + T \geq -h \tag{4.7}$$

Desse modo, temos, que dada uma classe A conflitante com uma classe B, podemos ter T máquinas da classe A a mais quando houver a presença de pelo menos uma VM da classe B em mesmo hardware. Outro aspecto que essa restrição trás é a da adição da inversão no conjunto dos conflitos de modo que determinados conflitos $c = \{(A, B), (B, C), (A, C)\}$ terão que ser transformado em $c = \{(A, B), (B, C), (A, C), (B, A), (C, B), (C, A)\}$, gerando duas restrições para cada conflito.

Já para os casos em que temos classes auto conflitantes a restrição fica como mostrado na equação 4.8. Nesse caso a variável T foi colocada no lugar da variável 1 da inequação 4.4, pois na formulação anterior o 1 representava a tolerância que uma classe podia ficar com ela mesmo.

$$\forall i \in 1..n(-\sum_{j=1}^{h} vm_j^{hw_i \cdot clsA}) \geq -T \tag{4.8}$$

Com essas duas restrições estabelecidas temos que T vira uma constante de tolerância. Apesar disso, o uso dessa constante é arbitrário e não leva em consideração o quão ruim é o conflito e em que proporções ele afeta o desempenho. Para contemplar isso é necessário definir um T que possibilita maior controle da tolerância. Esse T é dado pela equação 4.9, sendo E uma escala 1:E, de modo que para cada VM da classe B é permitido alocar E VM da classe A.

$$T = (\sum_{j=1}^{h} vm_j^{hw_i \cdot clsB}) \cdot E \tag{4.9}$$

A equação 4.10 mostra T para classes iguais, nessa equação de T para classes iguais em caso de números fracionários, o valor é arredondado para baixo aceitando apenas valores inteiros. Essa restrição possui um formato 1:E pela proposta de conflitos em classes iguais, tornando-se porcentagem para esses casos.

$$T = h \cdot \frac{porcentagem}{100} \tag{4.10}$$

Com essas novas restrições o número de restrições aumenta para $(n \cdot h + n \cdot z)$ para conjunto de conflitos diferentes e (n) restrições para cada conjunto de conflito iguais, esse valores são acrescidas as quantidades de restrições do PBFVMC $(2n + 2k + 2)$.

4.2.1 Exemplo

Continuando com o exemplo da Seção 4.1 com o uso da tabela 6, agora aplicando as restrições apresentadas na Seção 4.2

Sendo a classe A conflitante com a B, e onde para cada classe A eu aceite uma VM da classe B no mesmo hardware, temos através da restrição 4.1:

$$Constantes: \qquad n = 2 \quad z = 1 \quad h = 1 \quad E = 1$$

$$-1 \cdot vm_1^{hw_1} - 1 \cdot vm_3^{hw_1} + 1 \cdot vm_3^{hw_1} \geq -1 \tag{4.11a}$$
$$-1 \cdot vm_1^{hw_2} - 1 \cdot vm_3^{hw_2} + 1 \cdot vm_3^{hw_2} \geq -1 \tag{4.11b}$$
$$-1 \cdot vm_3^{hw_1} - 1 \cdot vm_1^{hw_1} + 1 \cdot vm_1^{hw_1} 1 \geq -1 \tag{4.11c}$$
$$-1 \cdot vm_3^{hw_2} - 1 \cdot vm_1^{hw_2} + 1 \cdot vm_1^{hw_2} 1 \geq -1 \tag{4.11d}$$

Agora para a vm_1 e vm_2 que são da mesma classe, uma classe auto conflitante, para isso usamos a restrição 4.4:

$$-vm_1^{hw_1} - vm_2^{hw_1} \geq 1 \tag{4.12a}$$
$$-vm_1^{hw_2} - vm_2^{hw_2} \geq 1 \tag{4.12b}$$

No caso da última restrição a constante T não se mostra tão evidente, pois o conjunto de máquinas virtuais é pequeno.

4.3 Minimização de Conflitos

Apesar da Subseção 4.2 tentar contemplar limitações da tolerância a abordagem ainda traz algumas dificuldades proibitivas, que não retorna solução quando suas restrições de conflito não são satisfeitas, ou seja, quando a tolerância estipulada é excedida.

Entretanto, em casos que o número de hardware seja limitado às abordagens anteriores são contraprodutivas, pois, seria necessário um grande número de hardware ou continuar a abordagem padrão sem contabilizar as máquinas virtuais conflitantes.

Portanto, visando uma maior adaptabilidade foi elaborada uma terceira modificação na formulação. Nessa abordagem é visado a minimização dos conflitos existentes, dessa maneira a formulação retornará a melhor alocação encontrada com o menor número de conflitos existentes. Para isso foi adicionado à formulação original do PBFVMC duas novas variáveis, sendo elas M_{dif} representando um conflito entre classes diferentes e M_{iguais} representando conflitos de classes iguais. Essas novas variáveis têm o objetivo de contabilizar a quantidade de conflitos existentes. Desse modo a quantidade de variáveis adicionadas nesta nova formulação é uma para cada par de conflito de classes diferentes somado a uma por cada tipo de conflito de classes iguais.

Para ocorrer essa contagem de conflitos há a necessidade de adicionar as restrições:

$$\forall i, k, c \in 1..n, 1..z, (a, b); -vm_j^{hw_i \cdot clsA} - vm_j^{hw_i \cdot clsB} - \neg M_{dif} \geq -2 \tag{4.13}$$

$$\forall i \in 1..n \sum_{j=1}^{h} -vm_j^{hw_i \cdot clsA} - \sum_{j=1}^{h} \neg M_{igual} \geq -h \tag{4.14}$$

Com as novas variáveis definidas surge a possibilidade de minimizar esses conflitos através da restrição de minimização, atualizando a restrição 3.10 para 4.15, onde os

hardwares continuam sendo minimizados com o acréscimo da minimização dos conflitos.

$$min : \sum_{i=1}^{n} hw_i + 2 \cdot \sum_{ci=1}^{QC_{iguais}} M_{igual} + 2 \cdot \sum_{cd=1}^{QC_{diferentes}} M_{dif} \tag{4.15}$$

Nessa nova minimização temos novas variáveis QC_{iguais} e $QC_{diferentes}$, que representam as quantidades total de variáveis minimizadoras iguais e diferentes respectivamente, para calcular o valor de QC_{iguais} pode-se usar a fórmula $(n \cdot h)$, já para calcular $QC_{diferentes}$ pode ser usado a formula $(k \cdot z \cdot n)$. A quantidade de novas restrições podem ser escritas como n para cada conjunto de classes iguais e $(k \cdot z \cdot n)$ para cada conjunto de classe diferente.

4.4 Considerações

Com a adição das restrições explicitadas neste capítulo é possível realizar a proposta de manejar conflitos, dentro dessas propostas feitas são contemplados escopos de diferentes casos de conflitos entre classes de máquinas virtuais sendo eles: a totalmente restritiva para quando não há a possibilidade de máquinas virtuais conflitantes ficarem juntas, a tolerante a conflitos que possui uma tolerância para as máquinas virtuais conflitantes e por último a de minimização de conflitos.

5 Experimentos

Para realizar os experimentos da extensão feita no modelo PBFVMC, foi desenvolvido um programa utilizando a linguagem Rust para gerar as restrições que faz a leitura de dois arquivos, um contendo o *workload* de *hardware* e máquinas virtuais e outro contendo os conflitos. A estrutura do primeiro arquivo é dada pelos *hardware* e máquinas virtuais que são representados pela sua quantidade de RAM e CPU e no casos das máquinas virtuais um campo numérico representando a classe da máquina virtual, já no segundo arquivo foi colocado os pares de conflitos existentes. Os arquivos utilizados para os experimentos estão disponíveis no github [1].

Para a realização dos testes foi utilizado um conjunto derivado Google Cluster Data (WILKES, 2011; REISS; WILKES; HELLERSTEIN, 2011) com alterações de tamanho e adicionando dados fictícios de classes. Os testes foram realizados utilizando um conjunto de 4, 8 e 16 *hardware* com máquinas virtuais ocupando 25%, 50%, 75% e 90% da capacidade em cada teste. Essas porcentagens representam respectivamente (11,18,37,42) máquinas virtuais para 4 *hardware*, (18,45,62,79) máquinas virtuais para 8 *hardware* e (46,98,125,158) máquinas virtuais para 16 *hardware*.

A distribuição das classes conflitantes foi feita de maneira sintética, representando aspectos interessantes para avaliação do modelo, sendo esses: *workload* composto por duas classes conflitantes na proporção 20% classe A 80% classe B, três classes conflitantes com cada classe sendo um terço do total de máquinas virtuais, duas classes igualmente distribuidas (50% cada) e por último 20% das VMs com autoconflito.

No caso dos experimentos que contém tolerância, a proporção definida para essas tolerâncias foram para aceitar uma proporção de 1:1 para as classes diferentes e 20% de conflito aceito para classes iguais.

Para executar os testes foi utilizado o solver clasp (GEBSER; KAUFMANN; SCHAUB, 2012) em três máquinas equipadas com um processador Core i7-8700 3.2 GHz e com 16GB de memória RAM, com um tempo limite de execução de 7200 segundos.

5.1 Resultados

Os resultados dos experimentos podem ser interpretados através da tabela 7, que é composta por 4 matrizes onde as linhas representam a quantidade de *hardware* testadas para cada adaptação feita na formulação, que são descritas através das siglas TR (totalmente restritiva), TC(tolerância de classes) MIN(minimização de conflitos). Já as

[1] https://github.com/UnB-SAT/experimentos-PBFVMC-affinity

colunas representam o *workload* aplicado. Nas células se encontra o resultado, o primeiro número representa o tempo que o resolvedor usou de processamento, seguido pelo melhor resultado encontrado para a função de minimização, quando estiver negrito é porque o resolvedor conseguiu provar que a solução é ótima, quando estiver *UNSAT*, significa que é impossível alocar as virtuais com a configuração atual, e por fim, *TLE* representa que o resolvedor usou todo o tempo de processamento alocado a ele e nenhuma alocação foi encontrada.

Dos resultados obtidos na tabela 7, pode ser percebido que em todas as abordagens os workload menores (25% e 50%) tiveram melhor desempenho, outra percepção é o impacto significativo nos workload de 75% que todas as abordagens tiveram certa dificuldade.

Analisando os casos exclusivos de cada abordagens, começando pela abordagem totalmente restritiva, foi constatado a separação das classes conforme o proposto na seção 4.1, no entanto, como se pode ver na tabela os resultados quando a quantidade de hardware foi 16 hardwares a solução poucas vezes conseguiu retornar um resultado ótimo, característica que mostra que com o aumento da quantidade de hardware essa proposta levaria mais de 7200s para encontrar uma solução.

Já os experimentos com a abordagem parcialmente restritiva com tolerância teve resultados melhores que o primeiro em relação ao tempo como mostrado na tabela, porém o experimento realizado teve uma tolerância fixa, isso faz com que o resultado do experimento apenas mostre a viabilidade da formulação sendo necessários mais testes com uma gama mais elaborada de experimentos.

Os experimentos de minimização tiveram uma dificuldade maior para achar soluções ótimas, isso deve ter ocorrido pelo fato de ser uma abordagem mais pesada que as demais, vemos isso quando analisando a carga de restrições e variáveis que essa formulação tem quando comparada com as outras como podemos ver na Figura 3.

A Figura 3 é um gráfico que mostra a quantidade de restrições geradas por cada abordagem no experimento de duas classes com distribuição de 50% cada. Analisando esse gráfico podemos ver que a abordagem de minimização de fato é a mais discrepante no aumento do número de restrições, não obstante, a abordagem totalmente restritiva e a de tolerância de conflitos também tem um aumento no número de restrições comparadas com a PBFVMC.

Os experimentos realizados neste trabalho tiveram dois fatores limitantes que fizeram ser testados conjunto de dados relativamente pequenos quando comparados com o PBFVMC, esses fatores foram o número de restrições e o desempenho das abordagens, do qual os fatores podem estar interligados, pois, a quantidade grande de restrições da formulação acarreta arquivos maiores que dificulta a leitura por parte do solver pseudo-

Booleano.

A tabela 8 tenta mostrar o problema da quantidade de restrições problema, nela é feita a divisão dos experimentos conforme o explicado anteriormente na tabela 6 com a representação feita na tabela com o formato número de variáveis / número de restrições, a partir dessa tabela vemos como o número cresce abruptamente conforme o crescimento dos dados.

Portanto, analisando todos os resultados apresentados vemos que apesar de bem estruturada as restrições estabelecidas no Capítulo 4, temos que na prática, a geração de fórmulas de forma abrupta satura nossa capacidade de processamento, limitando a aplicabilidade da proposta.

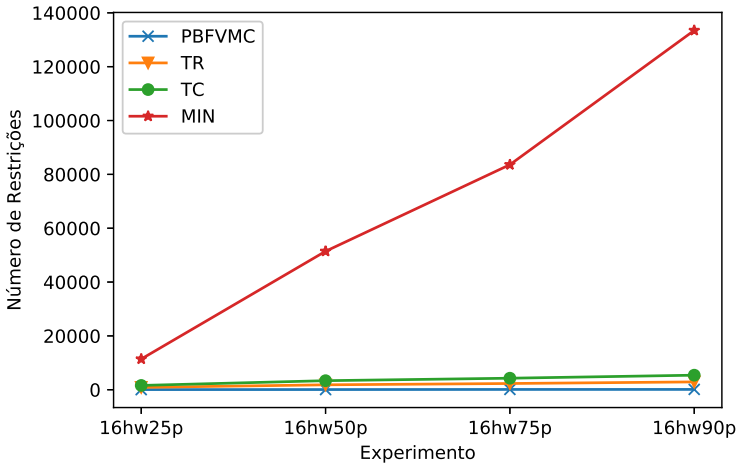

Figura 3 – Gráficos de número de restrições por abordagem

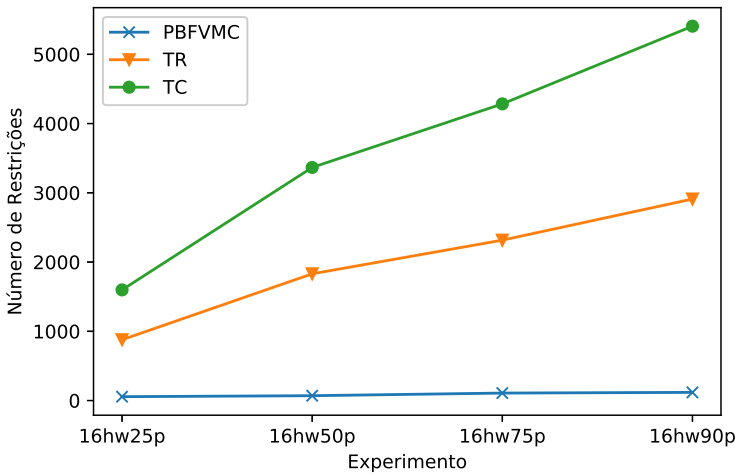

Figura 4 – Gráficos de número de restrições por abordagem

Tabela 7 – Tabelas com resultados dos experimentos

Matriz do tempo *hardware* por *workload* com 3 classes conflitantes				
Hardware/workload	25%	50%	75%	90%
4 hardware TR	**0.001/3**	**0.001/4**	**0.002/UNSAT**	**0.002/UNSAT**
8 hardware TR	**0.004/4**	**1.456/6**	**1997.580/8**	TLE
16 hardware TR	**0.010/4**	7200/9	7200/9	7200/16
4 hardware TC	**0.001/3**	**0.014/3**	7200/4	**0.358/4**
8 hardware TC	**0.152/3**	7200/5	7200/7	**1.476/8**
16 hardware TC	**0.694/4**	**18.864/8**	**487.607/12**	**27.372/15**
4 hardware MIN	**0.003/3**	**0.001/4**	7200/4	7200/4
8 hardware MIN	**0.238/4**	**10.431s/6**	7200/8	7200/8
16 hardware MIN	**0.689/4**	7200/9	7200/14	7200/16
Matriz do tempo hardware por *workload* com 2 classes conflitantes(20% - 80%)				
Hardware/Workload	25%	50%	75%	90%
4 hardware TR	**0.000/2**	**0.000/3**	**0.056/4**	**0.004/4**
8 hardware TR	**0.001/3**	7200/5	7200/7	**0.533/8**
16 hardware TR	**0.004/4**	7200/9	7200/13	**0.178/15**
4 hardware TC	**0.001/2**	**0.045/3**	7200/4	**0.026/4**
8 hardware TC	**0.169/3**	7200/5	7200/7	**0.156/8**
16 hardware TC	**2.252/4**	**1.507/8**	**5.828/12**	**5.820/15**
4 hardware MIN	**0.016/3**	**0.001/3**	**1.133s/4**	**0.087/4**
8 hardware MIN	**0.001/3**	7200/5	7200/7	**5.347/8**
16 hardware MIN	**0.242/4**	7200/9	7200/13	**2.339/15**
Matriz do tempo *hardware* por *workload* com 1 classes autoconflito(20%)				
Hardware/Workload	25%	50%	75%	90%
4 hardware TR	**0.001/2**	**0.000/3**	**0.001/UNSAT**	**0.001/UNSAT**
8 hardware TR	**0.001/3**	**0.095/UNSAT**	**0.065/UNSAT**	**0.109/UNSAT**
16 hardware TR	7200/9	TLE	TLE	TLE
4 hardware TC	**0.000/2**	**0.001/3**	7200/4	**0.001/4**
8 hardware TC	**0.002/3**	7200/5	7200/7	**0.176/8**
16 hardware TC	**0.008/4**	**0.381/8**	**0.545/12**	**0.235/15**
4 hardware MIN	**0.001/2**	**0.001/3**	**0.001/4**	**0.001/4**
8 hardware MIN	**0.004/3**	**258.562/5**	7200/7	**20.695/8**
16 hardware MIN	7200/5	7200/8	7200/13	7200/16
Matriz do tempo *hardware* por *workload* com 2 classes conflitantes(50% - 50%)				
Hardware/Workload	25%	50%	75%	90%
4 hardware TR	**0.001/2**	**0.001/3**	**0.175/UNSAT**	**0.095/UNSAT**
8 hardware TR	**0.0001/3**	**630.022/6**	7200/7	TLE
16 hardware TR	**0.004/4**	7200/9	7200/13	7200/16
4 hardware TC	**0.000/1**	**0.003/2**	7200/4	**0.001/4**
8 hardware TC	**0.001/2**	7200/5	7200/7	**0.002/8**
16 hardware TC	**0.003/4**	**0.569s/8**	0.190/12	**0.205/15**
4 hardware MIN	**0.001/2**	**0.007/4**	7200/4	**7200/4**
8 hardware MIN	**0.0031/3**	7200/5	4945.646/7	**7200/8**
16 hardware MIN	**0.305/5**	7200/9	7200/13	7200/16

Tabela 8 – Tabelas com número de variáveis e restrições geradas

Matriz variáveis/restrições *hardware* por *workload* com 3 classes conflitantes				
Hardware/workload	25%	50%	75%	90%
4 hardware TR	48 /84	76 /130	152 /236	172 /274
8 hardware TR	152 /222	368 /492	504 /654	640 /816
16 hardware TR	752 /878	1584 /1830	2016 /2316	2544 /2910
4 hardware TC	48 /120	76 /190	152 /380	172 /430
8 hardware TC	152 /342	368 /828	504 /1134	640 /1440
16 hardware TC	752 /1598	1584 /3366	2016 /4284	2544 /5406
4 hardware MIN	204 /188	496 /466	1976 /1908	2512 /2434
8 hardware MIN	992 /894	5744 /5484	10744 /10382	17280 /16816
16 hardware MIN	12032 /11406	52784 /51430	85328 /83596	135664/133470
Matriz variáveis/restrições *hardware* por *workload* com 2 classes conflitantes(20% - 80%)				
Hardware/Workload	25%	50%	75%	90%
4 hardware TR	48 /68	76 /106	152 /204	172 /230
8 hardware TR	152 /174	368 /396	504 /542	640 /688
16 hardware TR	752 /718	1584 /1494	2016 /1884	2544 /2382
4 hardware TC	48 /76	76 /118	152 /232	172 /262
8 hardware TC	152 /198	368 /468	504 /638	640 /808
16 hardware TC	752 /862	1584 /1798	2016 /2284	2544 /2878
4 hardware MIN	120 /104	256 /226	992 /924	1260 /1182
8 hardware MIN	512 /414	2960 /2700	5304 /4942	8320 /7856
16 hardware MIN	6080 /5454	25600 /24246	42016 /40284	65536 /63342
Matriz variáveis/restrições *hardware* por *workload* com 1 classes auto-conflito(20%)				
Hardware/Workload	25%	50%	75%	90%
4 hardware TR	48 /36	76 /50	152 /88	172 /98
8 hardware TR	152 /62	368 /116	504 /150	640 /184
16 hardware TR	752 /142	1584 /246	2016 /300	2544 /366
4 hardware TC	48 /36	76 /50	152 /88	172 /98
8 hardware TC	152 /62	368 /116	504 /150	640 /184
16 hardware TC	752 /142	1584 /246	2016 /300	2544 /366
4 hardware MIN	56 /36	88 /50	180 /88	204 /98
8 hardware MIN	176 /62	440 /116	600 /150	760 /184
16 hardware MIN	896 /142	1888 /246	2416 /300	3040 /366
Matriz variáveis/restrições *hardware* por *workload* com 2 classes conflitantes(50% - 50%)				
Hardware/Workload	25%	50%	75%	90%
4 hardware TR	48 /56	76 /82	152 /160	172 /178
8 hardware TR	152 /126	368 /292	504 /390	640 /496
16 hardware TR	752 /494	1584 /1014	2016 /1292	2544 /1614
4 hardware TC	48 /76	76 /118	152 /232	172 /262
8 hardware TC	152 /198	368 /468	504 /638	640 /808
16 hardware TC	752 /862	1584 /1798	2016 /2284	2544 /2878
4 hardware MIN	168 /152	400 /370	1520 /1452	1936 /1858
8 hardware MIN	800 /702	4416 /4156	8192 /7830	13120 /12656
16 hardware MIN	9216 /8590	40000 /38646	64512 /62780	102400/100206

6 Conclusão

Esse trabalho abordou três extensões de um método pseudo-Booleano existente para a consolidação de máquinas virtuais, com o objetivo de tratar os conflitos ocorridos em conjuntos de máquinas virtuais conflitantes. Cada extensão tentou contemplar uma diferente abordagem do problema, a saber: (i) máquinas conflitantes não podem ocupar uma mesma máquina, (ii) máquinas conflitantes podem ocupar uma mesma máquina sob uma certa tolerância fixa e (iii) máquinas conflitantes podem ocupar uma mesma máquina, com o objetivo de minimizar o conflito.

Através dos experimentos realizados foi visto que as extensões propostas neste trabalho tiveram resultados para o problema de separar classes, porém com dificuldade em relação ao tempo de execução quando se eleva as instâncias de hardware e as máquinas virtuais do conjunto testado. Essa dificuldade ocorreu pela forma que foi estruturada a formulação, resultando um crescimento muito abrupto das restrições.

Apesar dos resultados terem uma aplicação apenas para conjunto de dados pequenos, a formulação proposta nesse artigo abre uma possibilidade de uma futura evolução. Uma proposta interessante para trabalhos futuros seria evoluir a formulação de modo que diminua o impacto ocorrido no tempo que o solver leva, nesse aspecto seria interessante a modificação das restrições para que reduzisse a quantidade de variáveis e restrições geradas.

Referências

FRANCO, J.; MARTIN, J. A history of satisfiability. In: _____. [S.l.: s.n.], 2009. v. 185. Citado na página 12.

GEBSER, M.; KAUFMANN, B.; SCHAUB, T. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.*, v. 187, p. 52–89, 2012. Citado na página 29.

LICHT, F. L. Afinidades de tipos de aplicações em nuvens computacionais. 2014. Citado 3 vezes nas páginas 10, 14 e 20.

MA, S. A review on cloud computing development. *Journal of Networks*, v. 7, 02 2012. Citado na página 8.

REISS, C.; WILKES, J.; HELLERSTEIN, J. L. *Google cluster-usage traces: format + schema*. Mountain View, CA, USA, 2011. Revised 2014-11-17 for version 2.1. Posted at <https://github.com/google/cluster-data>. Citado na página 29.

RIBAS, B. *Um método de pré-processamento de fórmulas SAT e pseudo-Boolean baseado em técnicas de programação linear inteira mista*. Tese (Doutorado), 09 2015. Citado na página 17.

RIBAS, B. et al. On modelling virtual machine consolidation to pseudo-boolean constraints. In: . [S.l.: s.n.], 2012. v. 7637, p. 361–370. ISBN 9783642346538. Citado na página 14.

RIBAS, B. et al. Pbfvmc: A new pseudo-boolean formulation to virtual-machine consolidation. In: . [S.l.: s.n.], 2013. Citado 4 vezes nas páginas 8, 11, 14 e 17.

ROUSSEL, O.; MANQUINHO, V. Pseudo-boolean and cardinality constraints. In: _____. [S.l.: s.n.], 2009. v. 185. Citado na página 12.

SCHAD, J.; DITTRICH, J.; QUIANé-RUIZ, J.-A. Runtime measurements in the cloud: Observing, analyzing, and reducing variance. *PVLDB*, v. 3, p. 460–471, 10 2010. Citado na página 10.

TERRA-NEVES, M.; LYNCE, I.; MANQUINHO, V. Virtual machine consolidation using constraint-based multi-objective optimization. *Journal of Heuristics*, v. 25, 11 2018. Citado na página 8.

WILKES, J. *More Google cluster data*. Mountain View, CA, USA: [s.n.], 2011. Google research blog. Posted at <http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html>. Citado na página 29.

ZHENG, Z. et al. Qos ranking prediction for cloud services. *Parallel and Distributed Systems, IEEE Transactions on*, v. 24, p. 1213–1222, 06 2013. Citado na página 8.