Uso de Modelos Preditivos e SLAs para Reconfiguração de Ambientes Virtualizados

Elder M. Rodrigues, Ana T. Winck, Avelino F. Zorzo, Duncan D. Ruiz, Fábio D. Rossi

[1]Programa de Pós Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre - RS - Brasil

{elder.macedo,ana.winck,avelino.zorzo,duncan,fabio.rossi}@pucrs.br

Abstract. *This work proposes a strategy to dynamically allocate resources on virtualized environments based on SLA requirements. This allocation uses data mining algorithms executed over a set of results generated by benchmarks. The data mining algorithms produce a predictive model that indicates, from the current virtual machines configuration, what is the best set of parameters to change in order to improve the performance of the virtualized system. This paper also presents a new subsystems that uses the predictive model and the SLAs requirements to reconfigure the virtualized environment.*[1]

Resumo. *Este trabalho propõe a realocação dinâmica de recursos em ambientes virtualizados a partir de requisitos derivados de SLAs. A realocação utiliza algoritmos de mineração de dados gerados a partir da execução de benchmarks. Os algoritmos de mineração produzem modelos preditivos que sugerem, a partir de uma determinada configuração das máquinas virtuais, qual o melhor conjunto de parâmetros deve ser modificado para melhorar o desempenho de todo o ambiente virtualizado. Esses modelos preditivos são utilizados por um subsistema de reconfiguração que combina os mesmos com as políticas estabelecidas nos SLAs.*

1. Introdução

A virtualização de sistemas operacionais (SO) é uma prática que tem sido explorada para permitir a execução simultânea de múltiplos SOs em uma única máquina física [Mergen et al. 2006]. *Xen* [Barham et al. 2003] é um paravirtualizador que oferece tal recurso. A arquitetura deste é representada como uma camada de abstração sobre o *hardware*, a qual permite a execução de diversas máquinas virtuais (*VM*) paralelas, cada uma com seu próprio SO. Sua empregabilidade, motivada pela economia de recursos físicos, tem ganho destaque em *data centers* que, em função da grande demanda no fornecimento de serviços, podem compartilhar, para vários clientes, uma mesma infra-estrutura computacional [Cunha et al. 2007]. Esses ambientes, por serem altamente dinâmicos, precisam que seus recursos sejam constantemente ajustados, em busca de uma melhor utilização dos mesmos. Para tanto, uma boa prática é estabelecer acordos de níveis de serviço (*Service Level Agreement - SLA)*.

[1]Este trabalho foi desenvolvido em colaboração com a HP Brasil P&D.

Um *SLA* é a declaração de expectativas e obrigações que existem no relacionamento de negócio entre duas organizações: o provedor do serviço e seu consumidor [Gupta, D. et al 2006]. Esta declaração especifica os níveis de qualidade de serviço que o fornecedor se compromete em disponibilizar, bem como as cláusulas legais e as conseqüências para cada parte se houver descumprimento destes deveres. Dessa forma, o *SLA* estabelece o nível de serviço requisitado, sendo projetado para criar uma compreensão comum sobre serviços, prioridades e responsabilidades entre clientes e provedores.

Para propor melhorias no que se refere à performance do *Xen*, respeitando políticas de *SLA*, a seguinte questão pode ser explorada: qual a melhor alocação de recursos para uma máquina *Xen* quando várias *VMs* estão sendo executadas?

A Figura 1 exemplifica um ambiente virtualizado composto por quatro *VMs* (1, 2, 3, 4), sendo que cada uma dessas *VMs* tem necessidades diferentes de consumo (ex: *CPU* e memória), ilustrado pela altura de cada *VM*. As linhas tracejadas representam a disponibilidade de recursos para as *VMs*. A área hachurada equivale aos recursos disponíveis para o ambiente. A Figura 1a representa uma alocação de recursos sem a utilização de SLAs, a qual não preocupa-se em definir tais recursos às necessidade de cada *VM*. Analisando-se a Figura 1a percebe-se que a *VM* 2 está subutilizando os recursos a ela disponíveis, enquanto que a necessidade de consumo das *VMs* 1 e 3 é maior do que está à sua disposição, não sendo possível atender a toda sua demanda. Para melhorar o desempenho global do *Xen*, espera-se que os recursos possam ser adequadamente redistribuídos, respeitando políticas de *SLA* previamente definidas, e ajustando os recursos de acordo com as mesmas. Este modelo ideal está representado na Figura 1b, onde é possível notar, pela linha tracejada, que os recursos computacionais estão distribuídos proporcionalmente à demanda de cada *VM*, sem excesso ou escassez, respeitando a disponibilidade global do ambiente.

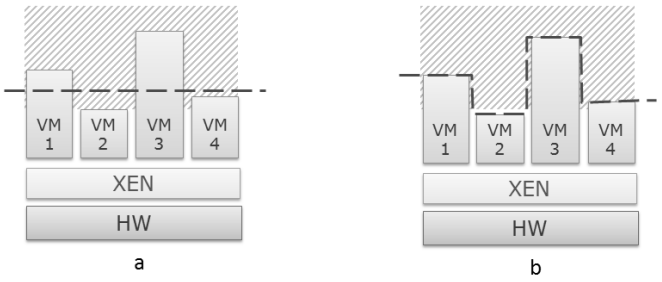

Figura 1. Realocação de recursos no Xen

Para que essa distribuição seja efetuada corretamente, é importante ter conhecimento sobre a demanda de recursos de cada *VM* e a disponibilidade do ambiente como um todo. Uma possível estratégia é fazer uso de resultados de testes de desempenho, obtidos através da execução de *benchmarks* sobre cada sistema operacional virtualizado. Por *benchmarks* é possível obter dados sobre a real capacidade computacional à disposição em cada *VM*. Os *benchmarks* utilizados nesta pesquisa geram resultados na forma de um relatório tabular, contendo métricas de desempenho dos sistemas operacionais. Seus resultados servem como base para que sejam mapeados os níveis de recursos consumidos por uma aplicação sob determinada carga de trabalho. Contudo, devido à diversidade de configurações vigentes, bem como ao grande volume de métricas retornadas por cada *benchmark*, torna-se difícil examinar, interpretar e aferir seus resultados manualmente.

Para solucionar essa questão, são empregados algoritmos de mineração de dados.

Este artigo está estruturado conforme segue. A Seção 2 apresenta o modelo do processo de realocação de recursos proposto, detalhando suas etapas. Na Seção 3 é mostrado como o processo proposto é validado, apresentando a sua estrutura experimental. A Seção 4 apresenta os resultados obtidos com o modelo proposto. Na Seção 5 são listados os trabalhos relacionados ao tema apresentado. A Seção 6 relata as conclusões obtidas e os trabalhos futuros a serem desenvolvidos.

2. Modelo do Processo de Realocação de Recursos Proposto

A Figura 2 ilustra o processo proposto para realocação de recursos, o qual abrange um total de 9 etapas. Primeiramente são executados *benchmarks* sobre distintas configurações do ambiente virtualizado pelo *Xen* (a). Com essas execuções é possível definir *SLAs*, através da decomposição de métricas de alto nível em métricas de baixo nível (b), e também é possível obter dados de configuração e de desempenho (c) desse ambiente. Sobre esses últimos dados são aplicados algoritmos de mineração de dados (d), os quais buscam identificar padrões e apontar, através de modelos preditivos (e), características quanto ao desempenho computacional do ambiente virtualizado. Esses modelos, que interagem com um subsistema de reconfiguração (f), apresentam, a partir de uma configuração vigente de uma dada máquina *Xen* (g), qual a melhor maneira de reconfigurar os parâmetros desta (i), baseando-se em políticas de *SLA* definidas (h).

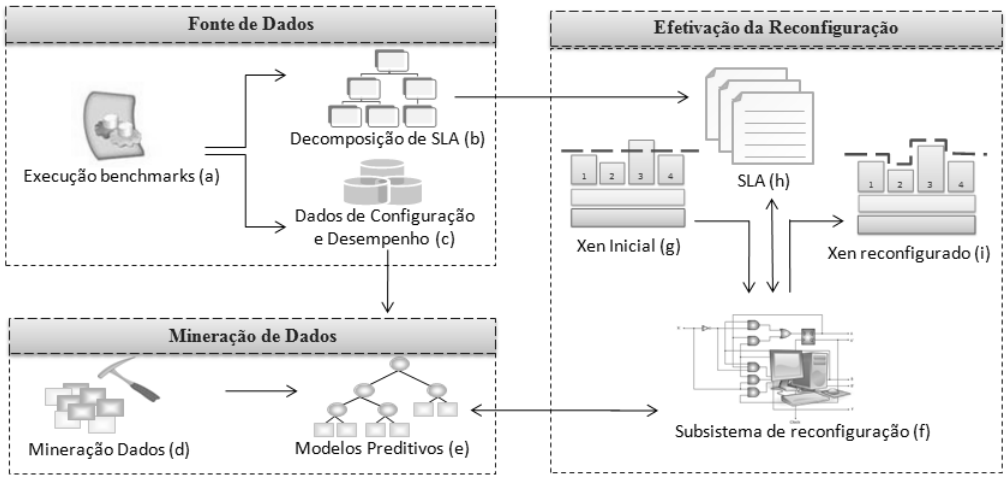

Figura 2. Processo proposto para a reconfiguração de máquinas Xen

2.1. Fonte de Dados

A execução de *benchmarks* é o ponto incial para a estratégia de reconfiguração dinâmica de ambientes virtualizados proposta neste artigo. Os *benchmarks* são executados sobre cada *VM* que compõe os distintos ambientes virtualizados. Foram utilizados dois *benchmarks*: *TPC-W* [TPC-W 2008] e *Unixbench* [Unixbench 2007]. Os resultados do primeiro são utilizados para a decomposição de *SLAs*, e do segundo para compor um banco de dados de configuração e desempenho dos ambientes sobre os quais esse *benchmark* foi executado. Dessa forma, é possível utilizá-los durante o processo de predição de parâmetros para reconfiguração do ambiente.

2.1.1. Decomposição de SLA

Considerando que um *SLA* estabelece o nível de qualidade dos serviços que o fornecedor se compromete em disponibilizar, caso não esteja sendo monitorado o desempenho e a utilização dos recursos por cada uma das máquinas virtuais que compartilham um mesmo *hardware*, pode uma *VM* estar com 60% dos recursos alocados e estar utilizando apenas 10%, e outra *VM* possuir 40% dos recursos e estar utilizando 100% destes. Por este motivo, utilizamos *SLAs* para definir quais as necessidades de recursos para uma *VM* rodando um tipo específico de aplicação, e para monitorar a utilização destes recursos por cada uma destas *VMs* com o objetivo de avaliar a distribuição dos recursos com base nas regras declaradas no *SLA*.

Um *SLA* pode especificar várias necessidades de serviço, que inclui métricas como disponibilidade, tempo de resposta e processamento [Udupi et al. 2007]. Estes valores de níveis de serviços, quando expressos em um *SLA*, tomam a forma de *Service Level Objectives (SLOs)*, e são aplicados nas métricas de desempenho, conhecidas como *Service Level Indicators (SLIs)*. "Disponibilidade" e "tempo médio de resposta do serviço" são exemplos de *SLIs* utilizados em um *SLA*, enquanto que os valores 95% de disponibilidade ou 1,3 segundos de tempo de resposta são exemplos de valores de um *SLOs* [Sauve et al. 2005]. Apesar de *SLO* e *SLI* serem os mais importantes parâmetros, um *SLA* pode ainda possuir diversos outros, tais como penalidades a serem pagas pelo provedor do serviço quando este falhar em cumprir o nível de serviço prometido e recompensas que serão pagas pelo cliente do serviço quando o provedor do serviço superar as expectativas do nível de serviço acordado [Marques 2006].

Para uma definição precisa dos níveis de recursos que uma aplicação necessita para atender as especificações de um *SLA*, alguns trabalhos [Gupta, D. et al 2006], [Cunha et al. 2007] propõem a decomposição das métricas de alto nível em métricas de baixo nível. Desse modo, uma métrica como tempo de resposta de uma aplicação é decomposta em métricas de baixo nível, com uso, por exemplo, dos seguintes atributos: largura de banda (lb), utilização do processador (up) e memória disponível (md). Para cada um desses são definidos valores mínimos a serem atingidos, onde $vmlb$, $vmup$ e $vmmd$ representam, respectivamente, tais valores para os atributos citados.

Para atingirmos um indicador de nível de serviço (*SLI*) é necessário que, por exemplo, $lb > vmlb$, $up > vmup$ e $md > vmmd$. Para conhecermos estes níveis, precisamos submeter a aplicação a um conjunto de testes através da execução de *benchmarks* sob diversas configurações, com o objetivo de definir a quantidade de recursos do sistema que cada componente monitorado pelas métricas de baixo nível consome. Deste modo, é possível determinar o volume de recursos consumidos por cada componente do sistema para suprir de maneira precisa cada *SLI* de um *SLA*. No entanto, trabalhos anteriores [Gupta, D. et al 2006] [Cunha et al. 2007] abordam apenas a definição estática dos recursos necessários para que uma aplicação cumpra um *SLA*. Caso uma aplicação necessite mais recursos do que aqueles definidos em seu *SLA*, mesmo existindo recursos não utilizados por outras *VMs*, a aplicação poderá apresentará uma degradação na qualidade do serviço oferecido (*QoS*).

Com o objetivo de superar esta limitação, foi implementado um subsistema de reconfiguração responsável pela realocação dos recursos (ver Seção 2.3), de modo a não

quebrar um *SLA* entre *VMs* que compartilhem uma mesma estrutura física. Para tanto, torna-se fundamental saber com precisão o desempenho dessas *VMs*, onde a estratégia é fazer uso das métricas de desempenho das *VMs*, extraídas de execuções de *benchmarks*. Com essas métricas, espera-se predizer características de desempenho e, a partir delas, sugerir novos parâmetros de configuração, de modo a alcançar um ganho de desempenho.

2.1.2. Dados de Configuração e Desempenho

Os dados extraídos das execuções de *benchmarks* servem para apoiar a tomada de decisão para a reconfiguração. Para tanto, busca-se ter uma grande quantidade de dados. Assim, é feito um planejamento de execução de *benchmarks* que, conforme apresentado em [Winck, A. T., Ruiz, D. D. 2008], definem:

- configurações de ambientes, como: paravirtualizador, sistema operacional e hardware (destacando-se a memória disponível);
- limites de consumo de *CPU* (em percentual) disponível para o ambiente. Por conveniência, definiu-se limites que variam de 70% a 100%, com intervalos de 5 em 5;
- número total de *VMs* utilizadas. Os testes foram realizados com 4 VMs, mas esse número pode ser alterado de acordo com as necessidades do usuário e disponibilidade do ambiente;
- limites de consumo para cada *VM* do ambiente, onde essas recebem uma fatia do percentual de *CPU* disponível. A essa fatia é atribuido o nome de *CAP*. O critério empregado define que o valor mínimo de CAP é 10 e suas variações são múltiplas de 5;
- para cada combinação de *CAP* são atribuidas diferentes alocações de memória, definidas em *MB*, para cada *VM*. A soma dos valores de memória das *VMs* resulta no total de memória disponível para o ambiente. Os valores de memória para cada VM são definidos inteiros, e o valor mínimo alocado é de 40 MB.

Para fins de ilustração, a Figura 3 mostra duas situações com 7 configurações cada (representadas de 239 a 245 e de 281 a 287), onde cada uma é composta por: (1) ambiente, contendo configurações de memória, escalonador, *hardware*, entre outros; (2) *CPU*, o qual representa o limite de consumo de *CPU* disponível para o ambiente; (3) *VMs* utilizadas; (4) quantidade de *CAP* disponível para cada *VM*; (5) memória (em *MB*) alocada para cada *VM*; e (6) configuração, que atribui um número de configuração para cada conjunto das configurações anteriores.

Portanto, para cada célula de configuração vs. memória há uma execução de *benchmark* que retorna métricas e seus respectivos valores. Esse conjunto de características é armazenado em um banco de dados que comporta tanto as configurações definidas, como os resultados de *benchmarks* obtidos de cada configuração. Foram planejadas 539 configurações distintas, com 4 *VMs* cada uma, o que resultou em 2.156 execuções de *benchmarks*.

2.2. Mineração de Dados

A mineração de dados busca converter dados brutos em informação. Neste trabalho ela é empregada para identificar se é possível melhorar a performance de uma dada máquina

Ambiente		Escalonador: Credit / Xen: 3.0.4 / Maquina: Xeon Kernel: 2.6.16.33 / Memória Disponível: 280MB			
CPU (%)		85			
VMs		VM 1	VM 2	VM 3	VM 4
CAP		10	10	20	45
		Memória (MB)			
Configuração	239	70	70	70	70
	240	80	70	70	60
	241	90	70	70	50
	242	100	70	70	40
	243	110	70	60	40
	244	120	70	50	40
	245	130	70	40	40

Ambiente		Escalonador: Credit / Xen: 3.0.4 / Maquina: Xeon Kernel: 2.6.16.33 / Memória Disponível: 280MB			
CPU (%)		100			
VMs		VM 1	VM 2	VM 3	VM 4
CAP		25	15	25	35
		Memória (MB)			
Configuração	281	70	70	70	70
	282	80	70	70	60
	283	90	70	70	50
	284	100	70	70	40
	285	110	70	60	40
	286	120	70	50	40
	287	130	70	40	40

Figura 3. Planejamento de execuções de benchmarks

Xen. Nesse sentido, optou-se em utilizar tarefas preditivas de mineração de dados, focalizando em algoritmos de classificação [Tan et al. 2006]. Essas técnicas buscam construir modelos preditivos que apresentem a melhor combinação entre um conjunto de atributos, denominados atributos explanatórios, e um dado atributo de interesse, denominado atributo preditivo. A classificação [Han, J., Kamber, M. 2001] descreve e distingue o atributo preditivo, tal que os modelos resultantes possam ser utilizados para predizer a classe cujos atributos explanatórios pertencem. Vale ressaltar que, como sistema de apoio, optou-se em utilizar o algoritmo J48 (implementação de árvore de decisão C4.5) [Quinlan 1996] do ambiente Weka [Weka 2008].

Os modelos preditivos produzidos indicam, a partir da configuração vigente de uma máquina *Xen*, quais conjuntos de configurações, se alterados, podem fornecer ganho no desempenho. Para que esses resultados pudessem ser produzidos satisfatoriamente, os dados de configuração e desempenho, mencionados na Seção 2.4, foram adequadamente pré-processados, conforme descrito em [Winck, A. T., Ruiz, D. D. 2008]. Esta preparação é feita em duas etapas: a primera para definir o atributo preditivo, e a segunda para definir os atributos explanatórios.

O atributo preditivo foi planejado, convenientemente, como binário, e sua definição é dada como segue. Primeiramente foram coletados os resultados de desempenho de cada *VM* ($rdvm$), em cada configuração. Tais resultados foram definidos por $rdvm_i$, onde i é o número da *VM* em questão. Para obter o desempenho da configuração (dc) como um todo, foi efetuado o somatório do desempenho de cada *VM* em cada configuração, chegando-se a $dc_n = \sum_{i=1}^{t} rdvm_i$, onde n é o número da configuração e t é o número total de *VMs* para uma dada configuração.

Para verificar se há benefício em reconfigurar, é definido um valor para o custo de reconfiguração de tal ambiente. De posse desse valor é verificado o resultado da diferença entre o desempenho de uma configuração inicial e o de uma configuração alvo. O resultado da diferença determinará o atributo preditivo. Se a diferença for maior do que o custo, então significa que a configuração é benéfica e o atributo preditivo é 1; se a diferença for menor ou igual ao custo, então não há benefício em reconfigurar, e o atributo preditivo é 0. De posse desses resultados, é elaborada uma matriz quadrada, cujo tamanho corresponde ao total de configurações analisadas (no caso, 539), contendo os valores do atributo preditivo. O valor de cada célula da matriz indica se mudança da configuração inicial, para a configuração alvo é benéfica. Como efeito, essa matriz deve ser capaz de produzir um grafo dirigido, necessariamente acíclico, com as mudanças de configurações benéficas.

Como a definição do atributo preditivo é elaborada a partir da comparação da configuração inicial com as configurações alvo, é apropriado que o pré-processamento dos atributos explanatórios siga o mesmo princípio. Esses atributos, que combinados com o atributo preditivo correspondente, são utilizados pela mineração de dados. Para a seleção desses, foi obedecido o critério de que devem ser empregados apenas aqueles que correspondem aos parâmetros de configuração que se deseja alterar. Nesse sentido, optou-se por utilizar os atributos referentes ao percentual de *CPU*, *CAP* e memória. Os modelos preditivos produzidos sugerem reconfigurações para esses parâmetros. Vale ressaltar que esses modelos não são gerados dinamicamente.

2.3. Monitoração e Efetivação da Reconfiguração

O subsistema de realocação de recursos proposto busca otimizar a utilização do *hardware* e possibilitar a utilização de *SLA* em ambientes virtualizados. Esse processo é feito através da realocação de recursos entre *VMs*, tendo como base os modelos preditivos. A partir desses modelos, o subsistema de realocação de recursos utiliza a nova configuração para as máquinas virtuais dependendo de sua configuração e de suas necessidades atuais. Estas necessidades são determinadas pelo sistema de monitoramento que analisa o nível de recursos utilizados por cada uma das máquinas virtuais que compartilham a estrutura. No entanto, caso uma máquina virtual necessite de um nível de processamento maior do que os limites já definidos e exista a disponibilidade de recursos, é de responsabilidade do subsistema de realocação distribuir os recursos adequadamente.

A realocação consiste em mover os recursos não utilizados pelas demais *VMs* para a *VM* que esteja utilizando todos os recursos atribuídos a ela no *SLA* e, que ainda assim, necessite de mais recursos (área escura da *VM* 3 da Figura 4a). No entanto a quantidade de recursos que o subsistema pode realocar para uma *VM* nunca pode provocar a quebra do *SLA* das outras *VMs*. Deste modo, mesmo que existam recursos não utilizados dentro dos limites definidos no *SLA* da *VM*, estes não serão realocados (áreas quadriculada da *VM* 1, Figura 4). Ou seja, o subsistema executa a realocação apenas dos recursos que se encontram disponíveis (área hachurada Figura 4) além daqueles definidos nos *SLAs* (linha pontilhada Figura 4). Após a realocação (Figura 4b), o subsistema continua monitorando os níveis de recursos utilizado pelas *VMs*. Quando o processamento da *VM* que recebeu os recursos retorna ao nível descrito em seu *SLA*, os recursos tornam-se novamente disponíveis para serem utilizados pelas demais *VMs*. A descrição completa do subsistema está relatada em [Rossi 2005].

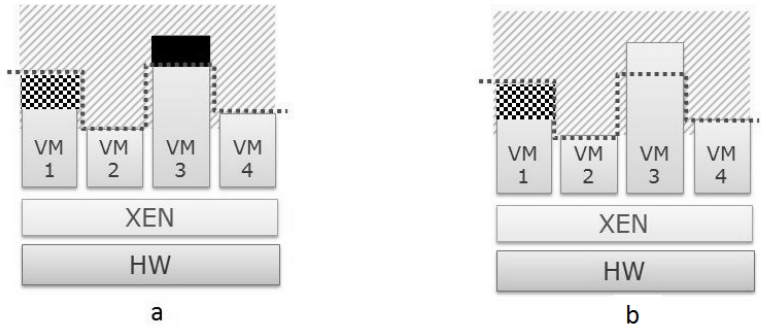

Figura 4. Realocação de recursos

3. Ambiente de Teste

Para verificar o ganho de performance obtido com a proposta descrita neste artigo, foram executadas três aplicações de *software* livre rodando em um ambiente virtualizado. Este ambiente é composto por quatro máquinas virtuais idênticas. Cada *VM* hospeda um servidor *web Apache* [Apache 2008], *Tomcat* 5.5 [Tomcat 2008] e um servidor de banco de dados *MySQL* 5.0 [MySQL 2008]. Para realizar as cargas de trabalho utilizamos a ferramenta *TPC-W* [TPC-W 2008]. O *TPC-W* é um *benchmark* padrão da indústria para aplicações de comércio eletrônico. O tamanho do banco de dados foi configurado em 10.000 itens e 200.000 usuários. Cada conjunto de teste é executado através de uma aplicação cliente que emula o acesso concorrente dos usuários ao servidor. O intervalo que cada cliente emulado espera antes de iniciar a interação seguinte (*think time*) foi configurado como um valor randômico entre 1 e 7 segundos. Definimos 100 usuários como número máximo de conexões simultâneas no *Tomcat* e no *MySQL*.

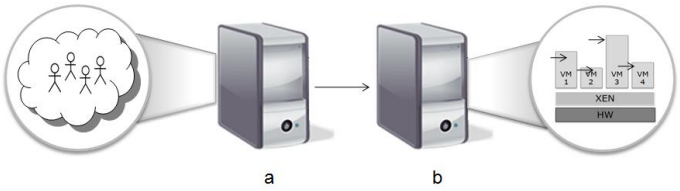

Figura 5. Estrutura utilizada

O ambiente de teste busca simular um *data center* virtualizado, com múltiplas aplicações compartilhando um conjunto comum de recursos. Este ambiente é composto de duas máquinas físicas. Uma é utilizada como hospedeiro das *VMs* com os serviços, sendo que esta estrutura hospeda quatro Máquinas Virtuais *Xen* (Figura 5 (b)). A outra máquina (Figura 5 (a)) é utilizada como gerador de *workload* dos clientes (*RBE - Remote Browsers Emulator*). A ligação entre os clientes (*RBEs*) e o servidor é realizada através de uma rede rápida *Gigabit Ethernet*. A estrutura virtualizada descrita é hospedada em um servidor com um processador *Athlon(tm) 64 X2 Dual Core de* 3.8 *GHz* com 2.0 *GB* de memória *RAM* e com um disco rígido *IDE* com capacidade de 160 *GB*. O sistema operacional utilizado foi o *Ubuntu 7.04 server i386, Kernel 2.6.19-4-server*. Já a estrutura que hospeda os *RBEs* consiste em um servidor com um processador *Athlon(tm) 64 X2 Dual Core de* 3.8 *GHz* com 2.0 *GB* de memória *RAM*.

4. Resultados

Para avaliar as funcionalidades e o desempenho do subsistema desenvolvido, foram aplicadas cargas de trabalhos sintéticas sobre a estrutura supracitada. Em um primeiro momento foi definido um *SLA* que foi aplicado a todas as *VMs* da estrutura. Este *SLA* possui um *SLO* que especifica que com uma carga de até 25 usuários simultâneos no sistema o tempo de resposta da aplicação deve ser inferior a 1 segundo. A partir deste ponto, foi realizada a decomposição e configuração dos recursos, que foram disponibilizados de maneira uniforme para todas as *VMs* da estrutura. Através da decomposição foram definidas os limites de recursos alocados para cada *VM*, que foram de 20% do processador (*CAP*) e 400 *MB* de memória *RAM*.

Na execução dos testes, busca-se retratar dois diferentes cenários. No primeiro, todas as *VMs* são submetidas a uma mesma carga de trabalho, com exceção de uma *VM*, que recebe uma carga maior de usuários do que a definida em seu *SLA*. No segundo, duas *VMs* recebem a carga de usuários definidas em seus *SLAs* e as duas restantes recebem uma carga de usuários maior.

No primeiro cenário, as aplicações hospedadas em cada uma das *VMs* foi submetida a uma carga de teste que simula o acesso simultâneo de 25 usuários. No entanto, a *VM* 1 teve seu número de usuários simultâneos alterado para 40. Este acréscimo de usuários busca retratar um cenário onde o *SLA* seria quebrado temporariamente por parte do cliente, motivado por um pico de acesso de usuários ao sistema. Os testes foram realizados em dois ambientes idênticos. Em um ambiente utilizamos apenas a decomposição de *SLA* para definir o limite de recurso utilizado por cada *VM*, e no outro utilizamos a decomposição de *SLA* e o subsistema de realocação de recursos proposto.

Como pode-se observar na Figura 6, o tempo de resposta das *VMs* 2, 3 e 4 se encontra dentro dos limites definidos em seus *SLAs*. A *VM* 1, por sua vez, apresenta um tempo de resposta maior que o estipulado. Este resultado se deve ao número de usuários excedentes aos definidos no *SLA*. Embora a estrutura que hospeda as *VMs* possua 20 % dos seus recursos livres, o escalonador do *Xen* não é capaz de realizar a realocação dos recursos não utilizados em ambientes com *SLAs*. Com a utilização do subsistema de realocação estes recursos tornam-se disponíveis para qualquer uma das *VMs* da estrutura, possibilitando suprir uma demanda maior por recursos pelas *VMs*. Comparando o tempo de resposta da aplicação, quando executada apenas com o escalonador do *Xen*, com o tempo de resposta de quando se utiliza o subsistema, nota-se uma acentuada redução do mesmo. Outro resultado importante, mostra que a utilização do subsistema possibilitou um ganho de desempenho global para as *VMs* da estrutura. Este ganho é proporcionado pelo fato do subsistema realizar a realocação para a *VM* que necessitar primeiro, neste caso a *VM* 1. No entanto, se o processamento da *VM* 1 retornar aos níveis descritos em seu *SLA*, o subsistema retira os recursos adicionais e realoca os mesmos para próxima *VM* que apresentar um pico de processamento. Deste modo, os recursos não são monopolizados pela *VM* com sobrecarga e tornam-se disponíveis para todas as *VMs* da estrutura.

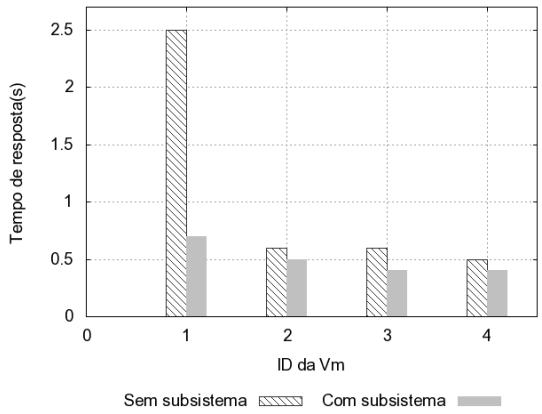

Figura 6. Sobrecarga em 1 VM

No segundo cenário de teste, de modo análogo ao anterior, as aplicações

hospedadas na *VM*3 e na *VM*4 foram submetidas a uma carga que simula o acesso simultâneo de 25 usuários. No entanto, as *VMs* 1 e 2 tiveram o número de usuários simultâneos alterado para 35. O acréscimo de usuários busca retratar um cenário onde o *SLA* das duas *VMs* é quebrado temporariamente por parte do cliente e as *VMs* disputam os recursos.

Como resultado (Figura 7), podemos observar que as *VM*s 1 e 2 apresentaram um tempo de resposta maior do que o valor definido nos *SLOs* dos respectivos *SLAs*. Este resultado é motivado pelo maior número de usuários no sistema, do que aquele utilizado para decomposição do *SLA* e conseqüente definição dos níveis de recursos da *VM*. Ao realizar os mesmos testes, mas utilizando o subsistema de realocação, obtivemos uma consideravel redução do tempo de resposta das *VMs* 1 e 2. Entretanto, neste cenário, houve uma redução no ganho das outras *VMs* (*VM* 4, Figura 7) ou até mesmo não apresentando ganho (*VM* 3, Figura 7). Estes resultados devem-se às necessidades de processamento das *VM* 1 e 2 ser maior. Deste modo, utilizaram por uma fatia de tempo maior os recursos realocados, em detrimento das outras *VMs* que possuíam uma carga menor.

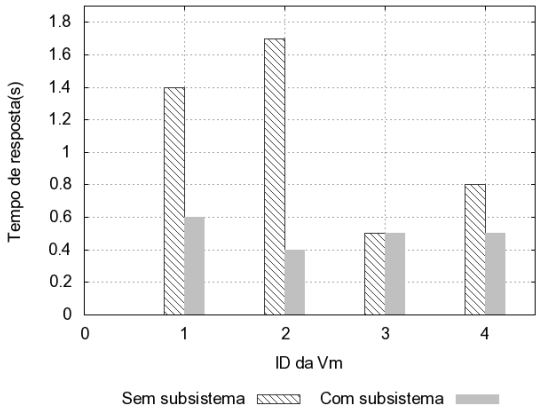

Figura 7. Sobrecarga em 2 VMs

Conforme os resultados apresentados, a utilização do subsistema de realocação permite atender uma maior demanda por recursos do que aqueles definidos através da decomposição de *SLA* e apresentando um desempenho melhor do que o obtido com o escalonador do *Xen*. Também podemos observar que a utilização do subsistema propiciou uma melhora no desempenho de todas as *VMs* da estrutura, e não apenas naquelas que possuíam uma sobrecarga de usuários.

5. Trabalhos Relacionados

Trabalhos recentes têm aplicado técnicas de mineração de dados para análise de desempenho em aplicações multicamadas. Estes trabalhos utilizam o [TPC-W 2008]. Com mineração de dados, [Jung et al. 2006] e [Parekh et al. 2006] buscam apontar gargalos e identificar, em que situações e com quais configurações pode ocorrer queda de performance. [Udupi et al. 2007] faz uso de mineração de dados para definir políticas de *SLA*. [Cunha et al. 2007] apresenta um estudo semelhante, propondo a identificação de *SLA* a partir de *SLOs* já definidos. Neste último trabalho, é citada a utilização do *Xen*, em um *data center*, como cenário. Apesar dos trabalhos supracitados proporem métodos

para garantir determinada qualidade do serviço em ambientes virtualizados, nenhum deles possibilita manter este nível em situações de sobrecarga das *VMs* da estrutura. Outro diferencial do nosso trabalho é a possibilidade de utilizar recursos ociosos, de modo a melhorar o desempenho das *VMs*, estando elas sobrecarregadas ou não.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou um modelo para realocação de recursos em ambientes virtualizados. Tal modelo conta com duas principais estratégias. A primeira está em utilizar mineração de dados para verificar se determinada máquina *Xen* merece ser reconfigurada, e assim indicar um melhor conjunto de parâmetros a ser modificado. A segunda contribuição está em estabelecer acordos de níveis de serviço (*SLA*) para que seja possível obter uma melhor utilização desses recursos. Esta última beneficia-se dos resultados obtidos com a mineracão de dados, empregando apenas as reconfiguracões que estejam de acordo com os *SLAs*. Ambas as estratégias utilizam os resultados obtidos com as execuções de *benchmarks*. A mineração utiliza esses como fonte de dados para, a partir do desempenho apresentado em cada execução, construir os modelos preditivos que indicam a reconfiguração. Já para definir os *SLAs*, os resultados de *benchmarks* são utilizados para que seja possível fazer a decomposição das métricas de alto nível, em métricas de baixo nível.

Para efetuar a reconfiguração, foi implementado um subsistema que tem por objetivo otimizar a utilização do *hardware* e efetuar a utilização dos *SLAs* definidos. Para realocar os recursos, esse subsistema faz uso dos modelos preditivos gerados pela mineração e das políticas de *SLA* definidas. Os resultados apresentados mostraram que, com a utilização do subsistema implementado, o provedor consegue manter o serviço com os níveis definidos no *SLA*, em algumas situações de sobrecarga da *VM*. Deste modo ele garante recompensas da parte do cliente, pelo fato do serviço superar as expectativas do nível de serviço acordado. É importante ressaltar que os modelos preditivos utilizados para a reconfiguração dizem respeito a um dado ambiente. Caso o ambiente seja alterado, um novo conjunto de *benchmarks* deve ser executado para que novos modelos preditivos sejam gerados.

Para aprimorar o subsistema desenvolvido, propomos como trabalhos futuros automatizar a execução dos casos de testes, de modo a simplificar o processo de decomposição através da utilização de *benchmarks*. Um outro ponto a ser trabalhado, é possibilitar a realocação dos recursos não utilizados pelas *VMs*, mas que se encontram dentro do nível de recursos definidos no *SLA*. Entretanto, para ser possível a realocação destes recursos, se faz necessária a utilização de técnicas de predição que possibilitem avaliar quais as necessidades futuras de recursos pelas *VMs*, de modo a não quebrar o *SLA* existente. Tais predições poderiam ser definidas já na etapa de mineração. Quanto à produção de modelos preditivos, uma abordagem seria utilizar resultados de benchmarks em tempo de execução para tornar o sistema mais preciso. Entretanto, a utilização dessa abordagem deve ser tratada com cautela para que a mesma não impacte de forma negativa no desempenho do sistema.

7. Agradecimentos

Agradecemos a Leandro T. Costa e Juliano Potrich pela coleta dos dados e execução dos benchmarks. Avelino F. Zorzo possui bolsa de produtividade CNPQ/Brasil.

Referências

Apache (2008). Apache http server project. *http://httpd.apache.org/*. Acesso fev. 2008.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *SOSP*, pages 164–177.

Cunha, I., Almeida, J., Almeida, V., and Santos, M. (2007). Self-adaptive capacity management for multi-tier virtualized environments. In *IFIP*, pages 129–138.

Gupta, D. et al (2006). Enforcing performance isolation across virtual machines in Xen. Technical report, HP Laboratories Palo Alto.

Han, J., Kamber, M. (2001). *Data mining: concepts and techniques.* Morgan Kaufmann.

Jung, G., Parekh, J., Pu, C., and Sahai, A. (2006). Detecting Bottleneck in n-tier IT Applications Through Analysis. In *DSOM*, pages 149–160.

Marques, F. T. (2006). Projeto de infra-estrutura de TI pela perspectiva de negócio. Master's thesis, Universidade Federal de Campina Grande, PB, Brasil.

Mergen, F. M., Uhlig, V., Krieger, O., and Xenidis, J. (2006). Virtualization for high-performance computing. *SIGOPS Oper. Syst. Rev.*, 40(2):8–11.

MySQL (2008). The world's most popular open source database. *http://www.mysql.com/*. Acesso 10 de fev. 2008.

Parekh, J., Jung, G., Swint, G., Pu, C., and Sahai, A. (2006). Comparison of performance analysis approaches for bottleneck detection in multi-tier enterprise applications. In *DSOM*, pages 149–160.

Quinlan, R. J. (1996). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rossi, F. D. (2005). Alocação dinâmica de recursos no Xen. Master's thesis, Dissertação (Mestrado em Ciência da Computação) - Faculdade de Informática - PUCRS, Porto Alegre.

Sauve, J., Marques, F., Moura, J., Sampaio, M., Jornada, J., and Radziuk, E. (2005). Optimal choice of service level objectives from a business perspective. In *Workshop of HP openview university association*, pages 15–27.

Tan, N., Steinback, M., and Kumar, V. (2006). *Introduction to data mining*. Addison Wesley.

Tomcat (2008). Apache tomcat. *http://tomcat.apache.org*. Acesso em 12 de jan. 2008.

TPC-W (2008). Transactional processing performance council. *http://www.tpc.org/tpcw/*. Acesso em 10 de fev. 2008.

Udupi, B., Sahai, A., and Singhal, S. (2007). A classification-based approach to policy refinement. In *IFIP/IM*, pages 758–788.

Unixbench (2007). Linux benchmark suite homepage. *http://lbs.sourceforge.net/*. Acesso em 06 de nov. 2007.

Weka (2008). Weka: Waikato environment for knowledge analysis. *www.cs.waikato.ac.nz/ml/weka/*. Acesso em 22 out. 2007.

Winck, A. T., Ruiz, D. D. (2008). Processo de KDD para auxílio à reconfiguração de ambientes virtualizados. In *SBSI*, pages 211–222. SBC.