Processo de KDD para Auxílio à Reconfiguração de Ambientes Virtualizados

Ana T. Winck, Duncan D. Ruiz

Programa de Pós Graduação em Ciência da Computação – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Porto Alegre – RS – Brasil

{awinck,duncan}@pucrs.br

Abstract. *This work presents a KDD process to reconfigure Xen virtual machines (VM). Xen allows several VMs running on a single hardware, simultaneously. At first, we obtain benchmarked performance data from each VM. After stored these data in a data warehouse, they can be properly prepared to be used by data mining techniques. Thus, the predictive models generated by data mining algorithms are enriched with reconfiguration instructions. Such models intend to suggest, from a current configuration, the best set of reconfiguration parameters to modify the environment and reach a better global performance.*

Resumo. *Este trabalho propõe um processo de KDD para auxílio à reconfiguração de máquinas virtualizadas Xen. Xen permite a execução simultânea de diversas máquinas virtuais (VM) sobre um mesmo hardware. Inicialmente, são obtidos dados de desempenho de cada VM, compilados de execuções de benchmarks sobre cada uma. Esses dados, após armazenados no data warehouse, podem ser preparados para serem utilizados por técnicas de mineração. Os modelos preditivos gerados podem, então, ser enriquecidos com instruções de reconfiguração. Tais modelos buscam sugerir, dada uma configuração vigente, qual o melhor conjunto de parâmetros de configuração para modificar o ambiente, e alcançar um ganho global de desempenho.*

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 [MER06]. Xen [BAR03] 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* [CHE07] que, em função de sua demanda no fornecimento de serviços, podem compartilhar, para vários clientes, uma mesma infra-estrutura computacional. Para propor melhorias no que se refere à performance do Xen, 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 apresenta uma típica atuação do Xen, onde se tem, no exemplo, quatro VMs (1, 2, 3, 4), sendo que cada uma dessas VMs tem níveis diferentes de consumo (ex: CPU e memória), ilustrado pela altura de cada VM. As linhas tracejadas

representam a disponibilidade de recursos para o ambiente. Partimos do pressuposto que, ao inicializar uma máquina Xen, os recursos disponíveis sejam igualmente alocados para todas as VMs. Essa alocação está representada pela Figura 1a, onde percebe-se que a VM 2 está subutilizando os recursos a ela disponíveis, enquanto que a necessidade de consumo da VM 3 é maior do que o limite à sua disposição, não sendo possível atender a toda sua demanda. Para melhorar a *performance* global do Xen, espera-se que os recursos possam ser adequadamente redistribuídos. Este modelo ideal está representado na Figura 1b, onde é possível notar, pela linha tracejada, que os recursos computacionais foram distribuídos proporcionalmente à demanda de cada VM, sem excesso ou escassez, respeitando a disponibilidade global do ambiente.

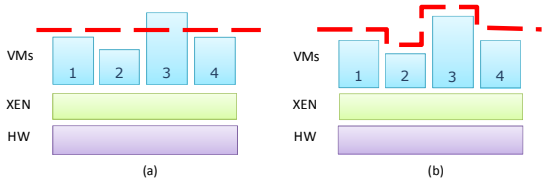


Figura 1. Distribuição de recursos no Xen

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. Somando os valores encontrados para cada VM, é possível medir o desempenho global do ambiente. Os *benchmarks* utilizados nesta pesquisa geram resultados na forma de um relatório tabular, contendo métricas de desempenho dos sistemas operacionais. 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. Assim, é conveniente que seja empregado algum suporte computacional que trate esta questão.

O objetivo desta pesquisa é propor um processo de apoio à reconfiguração de máquinas virtualizadas Xen, baseado em um processo de KDD (*Knowledge Discovery in Databases*). Ela está inserida em um projeto de parceria, onde outra pesquisa cobre os esforços de reconfiguração. Está sendo empregada a experiência adquirida pelo PPGCC/PUCRS, no desenvolvimento de um ambiente de *data warehousing* para métricas de processo de desenvolvimento software, relatadas em [BEC06]. O processo de KDD [FAY96][HAN01][LIR07] abrange um conjunto de etapas onde, a partir de uma coleção distinta de dados, pode ser constituído um *data warehouse* (DW) para que, com os dados dele, sejam utilizadas técnicas de mineração de dados para produzir conhecimento útil ao objetivo para o qual este processo esteja sendo aplicado. Para tanto, é preciso (1) coletar adequadamente dados que sejam relevantes ao problema, (2) efetuar uma limpeza nos mesmos, (3) prepará-los para que estes possam ser especialmente úteis para atender ao problema endereçado, (4) selecionar e executar um algoritmo de mineração adequado e (5) interpretar e apresentar os resultados obtidos.

Este artigo está organizado conforme segue. A Seção 2 apresenta o processo de KDD proposto, detalhando suas etapas. Na Seção 3 são detalhadas as etapas referentes à

fonte e dados, mostrando como os dados de benchmarks são planejados e coletados. A Seção 4 apresenta o DW modelado e como os dados são carregados. Na Seção 5 mostra-se a mineração de dados, detalhando como foi efetuada a preparação para alcançar resultados satisfatórios. A Seção 6 apresenta as etapas de apresentação dos resultados e de efetivação da reconfiguração. Não Seção 7 apresenta-se os testes do ambiente proposto. Por fim são apresentadas as conclusões e referências.

2. Processo de KDD Proposto

A Figura 2 ilustra o processo de KDD construído, o qual abrange um total de 12 etapas, conforme segue. Primeiramente são executados *benchmarks* sobre distintas configurações do ambiente virtualizado pelo Xen (a), de onde se obtém dados de configuração e desempenho (b) desse ambiente. Após passarem por um processo de extração, transformação e carga (ETC) (c), estes dados são armazenados em um DW especificamente projetado para este contexto (d). A partir desse DW, os dados precisam ser preparados (e) para serem utilizados por algoritmos de mineração de dados (f). Estes algoritmos buscam identificar padrões e apontar, através de modelos preditivos (g), características quanto ao desempenho computacional do ambiente virtualizado. Com efeito, esses modelos são enriquecidos (h) para que, ao serem lidos por um subsistema de reconfiguração (i) apresentem, dada uma configuração vigente de uma máquina Xen (j), e seguindo possíveis políticas de SLA (*service level objective)* (k), qual a melhor maneira de reconfigurar os seus parâmetros (l). Para melhor explicar esse processo de KDD, estas etapas são agrupadas nos seguintes contextos: fonte de dados (etapas a e b); *data warehouse* (etapas c e d); mineração de dados (etapas de e à g); apresentação (etapa h); e efetivação da reconfiguração (etapas de i à l).

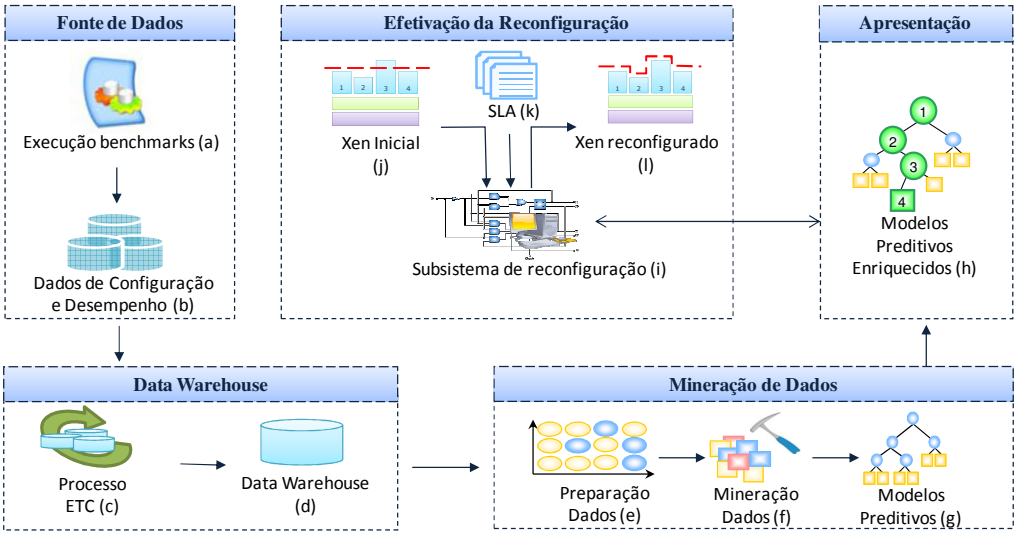


Figura 2. Processo de KDD construído para a reconfiguração de máquinas Xen

3. Fonte de Dados

As etapas que compõem o processo de KDD aqui proposto atuam sobre os dados obtidos das execuções de *benchmarks* e dos cenários sobre os quais estes foram executados. Os resultados das execuções dos *benchmarks* e os dados de configuração do

ambiente formam a fonte de dados deste processo de KDD. Os *benchmarks* utilizados foram Lmbench [MCV96], Iozone [IOZ06] e Unixbench [UNI07]. Por opção do projeto de parceria, os esforços de medições são focalizados nas execuções do Unixbench.

Segundo Han [HAN01] e Tan [TAN06] a mineração de dados é capaz de extrair conhecimento a partir de grandes volumes de dados. Assim, é importante obter um número satisfatório e consistente de resultados de *benchmarks*, os quais devem ser executados em larga escala e sobre diferentes cenários. Para tanto, foram realizadas execuções que englobam um conjunto de configurações disponíveis para o ambiente analisado. Além da configuração do ambiente (ex: hardware e sistema operacional), são definidas configurações próprias de cada execução.

As execuções de *benchmarks* buscam simular uma situação real. Logo, é importante mapear situações que possam ocorrer quando o sistema estiver em operação, executando aplicações nas VMs. Nesse sentido, foram definidos limites de consumo de CPU (em percentual), cujo objetivo foi o de simular um ambiente cujas aplicações estejam consumindo um determinado percentual do total de CPU disponível para o ambiente. Para tentar retratar essas situações, foram definidos os seguintes percentuais de CPU: 70%, 85%, 90% e 100%. O percentual de CPU utilizado é válido para o ambiente como um todo. Dessa forma, cada VM recebe uma fatia desse total. A essa fatia é atribuído o nome de CAP. Combinações de CAP simulam diferentes necessidades de consumo de aplicações que estejam executando em cada VM. Por fim, para cada combinação de CAP, são simuladas diferentes alocações de memória, definidas em MB, para cada VM. A soma dos valores de memória das VM resulta no total de memória disponível para o ambiente como um todo.

Foram definidas 539 configurações. A Tabela 1 mostra duas situações com 7 configurações cada (representadas de 239 a 245 e de 281 a 287), onde cada uma é representada 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 configuração vs. memória há uma execução de *benchmark* que retorna métricas e seus respectivos valores, totalizando 2.156 execuções de *benchmarks*.

Tabela 1. Planejamento de execuções de *benchmarks*

Ambiente		Escalonador: Credit / Xen: 3.0.4 / Maquina: Xeon Kernel: 2.6.16.33 / Memória Disponível: 280MB				**Ambiente**		Escalonador: Credit / Xen: 3.0.4 / Maquina: Xeon Kernel: 2.6.16.33 / Memória Disponível: 280MB			
CPU (%)		85				**CPU (%)**		100			
VMs		VM 1	VM 2	VM 3	VM 4	**VMs**		VM 1	VM 2	VM 3	VM 4
CAP		10	10	20	45	**CAP**		25	15	25	35
		Memória (MB)						**Memória (MB)**			
Configuração	**239**	70	70	70	70	**Configuração**	**281**	70	70	70	70
	240	80	70	70	60		**282**	80	70	70	60
	241	90	70	70	50		**283**	90	70	70	50
	242	100	70	70	40		**284**	100	70	70	40
	243	110	70	60	40		**285**	110	70	60	40
	244	120	70	50	40		**286**	120	70	50	40
	245	130	70	40	40		**287**	130	70	40	40

4. Data Warehouse

Para que os dados das execuções e das configurações sejam apropriadamente inseridos no DW, é importante que estes passem por um processo de extração, transformação e carga (ETC) [KIM02]. A etapa de extração busca identificar e extrair, a partir de diferentes origens, apenas os dados que se relacionam com o DW. A etapa de transformação busca eliminar problemas de ruídos ou inconsistência dos dados, busca como preencher valores nulos e busca padronizar os dados provenientes de fontes heterogêneas. Por fim, após os dados terem sido devidamente extraídos e transformados, eles são carregados no DW. Entretanto, para que esse processo seja corretamente efetuado, é importante que o DW tenha sido convenientemente modelado.

Um DW é comumente construído a partir de uma modelagem multidimensional [KIM02], o qual é composto por tabelas fato e dimensões. As dimensões denotam as características do problema a ser modelado. O fato contém os atributos chave das dimensões e atributos que representem valores relevantes ao produto. O modelo analítico desenvolvido e o processo de ETC correspondente são discutidos na seqüência.

4.1. Modelo Analítico

A Figura 3 apresenta o modelo analítico desenvolvido, no esquema floco de neve [KIM02], focalizado em um cenário para captura de métricas de *benchmarks*. Esse modelo possui dimensões que denotam (a) ambiente; (b) configuração; e (c) métricas (por limitações de espaço, estão sendo apresentados apenas os atributos das dimensões Configuracao e MaquinaVirtual, Figura 3b, uma vez que estes são referenciados no decorrer deste artigo). Entendeu-se ser o esquema floco de neve o mais conveniente pois o perfil típico do usuário desse modelo domina a estruturação proposta [KIM02], principalmente, para configuração do ambiente. Nos experimentos realizados, o modelo apresentou-se suficientemente abrangente, sendo capaz de armazenar uma série de execuções para diferentes configurações e *benchmarks*.

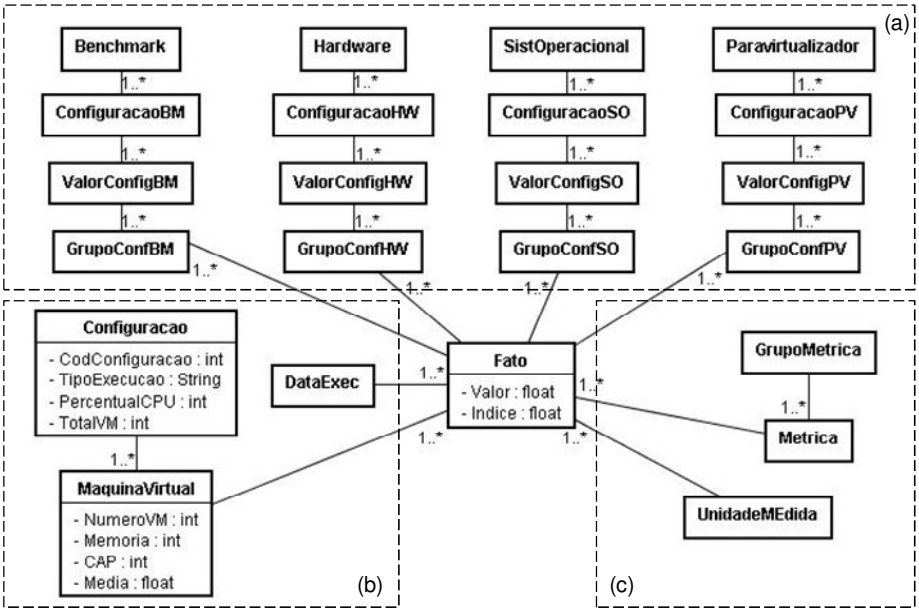


Figura 3. Modelo analítico para execuções de *benchmarks*, em UML

As dimensões de ambiente (Figura 3a) são modeladas hierarquicamente para atender à diversidade de configurações que possam ocorrer. Cada conjunto dessas dimensões está organizado em quatro níveis, onde os três primeiros especificam a configuração de cada característica, tendo: propriedade (ex: Paravirtualizador), configuração da propriedade (ex: ConfiguracaoPV) e valores de configuração de cada propriedade (ex: ValorConfigPV). Esta solução permite acomodar quaisquer detalhamentos que as propriedades necessitem, para melhor caracterizarem os fatos. Entretanto, para que seja possível associar um fato a uma combinação de configurações, o quarto nível é uma tabela ponte, na terminologia de Kimball [KIM02], que agrupa diferentes conjuntos de configurações (ex: GrupoConfXen). Exemplos de conteúdo: Paravirtualizador = {Xen}; ConfiguracaoPV = {Versão, Escalonador}; ValorConfigPV = {3.0.2, 3.0.4, Credit, Sdef}. Os valores "3.0.2, 3.0.4" correspondem à "Versão", e "Credit, Sdef" a "Escalonador". A finalidade da utilização de grupos torna-se evidente quando há necessidade de agrupar essas características entre si, de modo que um fato esteja associado a um Xen com uma versão e um escalonador, que correspondem a linhas distintas de ValorConfigPV.

As dimensões de configuração (Figura 3b) comportam características referentes ao planejamento efetuado para cada configuração, conforme exemplos da Tabela 1. Para tanto, são estabelecidas três dimensões (Configuracao, MaquinaVirtual e DataExec). Duas delas estão hierarquizadas para representar as características de cada configuração (dimensão Configuracao), bem como as características das VMs desta configuração (dimensão MaquinaVirtual). Nessa última dimensão, o atributo média indica o valor referente à *performance* de cada VM, onde esse valor é calculado e fornecido pelo próprio benchmark. A terceira dimensão (DataExec) identifica a estampa de tempo (*timestamp*) em que as execuções de *benchmarks* foram efetuadas para configuração.

As dimensões de métricas (Figura 3c) armazenam características de cada métrica, reportada por cada benchmark. Alguns *benchmarks* utilizam critérios próprios de agrupamento de métricas, reportando o mesmo nome de métrica, em diferentes grupos. Entendeu-se ser conveniente reproduzir esses agrupamentos no modelo. Assim, para cada grupo de métricas (dimensão GrupoMetrica), são agrupadas métricas em diferentes contextos (dimensão Metrica). Por fim, são identificadas quais são as unidades de medidas (dimensão UnidadeMedida) utilizadas para cada métrica, em cada *benchmark*.

A tabela fato representa os valores medidos para cada métrica em seu respectivo contexto. Este é composto pelos atributos chave das dimensões, caracterizando o cenário em que cada métrica foi capturada, e por dois atributos numéricos que representam o valor medido para cada métrica e seu índice correspondente (estes valores são calculados e fornecidos pelo *benchmark*).

4.2. Processo de ETC

Não há uma uniformização na apresentação dos resultados de *benchmarks*, nem um formato padrão para intercâmbio de resultados. Para que o DW seja propriamente alimentado, é necessário complementar os resultados das execuções de *benchmark* com os dados de configuração. Para tanto, os dados referentes ao ambiente e às

configurações estão organizados seguindo o mesmo formato dos exemplos de planejamento de execuções apresentados na Tabela 1.

Para garantir consistência nos dados que alimentam o DW, é feita uma análise cuidadosa nos dados resultantes das execuções de *benchmarks* e nos dados contidos no planejamento de execuções, de forma com que os dados dessas duas fontes sejam adequadamente relacionados. Kimball [KIM02] defende a criação de um DSA (*data staging área*), uma área de trabalho para dados que estão passando pelo processo de transformação. Optou-se, por ora, em organizar esse DSA no formato de uma planilha eletrônica, onde são identificadas e tratadas possíveis inconsistências ou ruídos.

Após a etapa de ETC, os dados estão adequadamente inseridos no DW construído que, portanto, já admite a manipulação por ferramentas OLAP para pivoteamento de dados, *roll-up*, *drill-down* e *slice&dice*, operações típicas de ambientes de processamento analítico de dados.

5. Mineração de Dados

A mineração de dados é a etapa do processo de KDD que converte dados brutos em informação. Busca-se identificar, nesta etapa, se é possível melhorar a *performance* de uma dada máquina Xen, e como isso deve ser feito, indicando uma reconfiguração adequada para seus parâmetros. Nesse sentido, optou-se em utilizar tarefas preditivas de mineração de dados, focalizando em algoritmos de classificação, que indicam, dada uma configuração vigente, quais conjuntos de configurações podem fornecer melhora de desempenho. Como sistema de apoio, optou-se em utilizar o algoritmo J48 (implementação de árvore de decisão C4.5 [QUI96]) do ambiente Weka [WEK07].

5.1. Preparação de Dados

Para que um algoritmo de mineração retorne resultados mais satisfatórios e condizentes com o propósito do problema, é sugerido [TAN06] [HAN01] [FAY96] [LIR07] que os dados a serem minerados passem por uma etapa de preparação para a mineração (a qual não deve ser confundida com o processo de ETC). Esta busca trabalhar com os dados que já estão contidos no DW e adequá-los para a mineração. Optou-se por conduzir a preparação em duas etapas. Na primeira, os dados são coletados para definir o atributo preditivo. Na segunda, os demais atributos são adequadamente organizados em um arquivo a ser lido pelo algoritmo de mineração.

5.1.1. Preparação do Atributo Preditivo

Para definir o atributo preditivo, após vários testes preliminares, considerou-se conveniente efetuar um mapeamento entre todas as execuções, de modo a identificar se a alteração de uma dada configuração para outra traz vantagens, ou não, na reconfiguração.

Primeiramente são coletados os atributos média da dimensão MaquinaVirtual do DW (ver seção 4.1) e, para cada configuração, é efetuado o somatório do atributo média de cada VM. Esse somatório representa o desempenho global de cada configuração. Em seguida, cada configuração é comparada com as demais, onde se tem a configuração inicial e a configuração alvo (ex: tendo a configuração 1 como inicial, têm-se as configurações 2, 3, ..., 539 como alvo), onde é feita a diferença entre o somatório da configuração alvo e o somatório da configuração inicial.

De posse desses resultados, é elaborada uma matriz quadrada (no exemplo, de tamanho 539), cujo objetivo é mapear o efeito da mudança de configurações. As células dessa matriz contêm valores 0 ou 1. Caso a diferença seja maior do que zero, então significa que houve melhora de desempenho, e a célula correspondente da matriz é alimentada com 1; caso a diferença seja menor ou igual a zero, significa que a configuração inicial apresenta um desempenho melhor ou equivalente à configuração alvo, e a célula correspondente da matriz é alimentada com 0. O valor de cada célula vai dizer se mudança da configuração inicial (eixo x), para a alvo (eixo y) vale a pena, e esse será o atributo preditivo. Essa matriz é capaz de produzir um grafo dirigido, necessariamente acíclico.

5.1.2. Preparação dos Atributos para a Mineração

Por questões de simplicidade e portabilidade, optou-se por preparar os dados em um arquivo do tipo "arff" (*attribute-relation file format,* formato preferencial aceito pelo Weka), mesmo a ferramenta Weka sendo capaz de se comunicar diretamente com um DW. Como a definição do atributo preditivo foi efetuada considerando-se uma configuração inicial comparada com as configurações alvo, é apropriado que o arquivo para a mineração seja elaborado seguindo o mesmo princípio. Esse arquivo contém dados que representam a configuração inicial, dados que representam a configuração alvo, e o valor do atributo preditivo correspondente. Os dados que fazem parte do arquivo correspondem a parâmetros de configuração que podem ser alterados. Foram escolhidos, como parâmetros, os seguintes: CAP e Memória.

5.1.3. Situações Especiais Tratadas na Preparação

Com uma série de experimentos efetuados, detectaram-se algumas anomalias. Estas mostraram que os atributos, cujos tipos de valores são numéricos, estavam sendo tratados, pelo algoritmo de mineração, como valores ordenáveis. Contudo, constatou-se que os mesmos não apresentam relações de ordem, necessariamente, na maioria dos casos. Assim, optou-se por torná-los todos categóricos.

A soma dos valores de CAP resultam no total de percentual de CPU. Estes são armazenados individualmente para cada VM. Entretanto, já que o arquivo para a mineração está trabalhando com os parâmetros das configurações inicial e alvo, é interessante que o CAP seja trabalhado como a combinação do valor de cada uma delas, e não tratados individualmente. Assim, foi definido um único atributo categórico para CAP, o qual representa a sua combinação de valores. Para os diferentes percentuais de CPU definidos, há um total de 77 combinações de CAP. De maneira análoga, os valores de memória para cada VM são tratados como um conjunto, e não individualmente.

6. Apresentação e Efetivação da Reconfiguração

A pesquisa relatada nesse artigo está inserida em um projeto de parceria, onde outra pesquisa, de forma complementar, cobre os esforços correspondentes às etapas de (i) à (l) da Figura 2, e efetiva a reconfiguração. O enriquecimento do modelo preditivo, etapa (h) da Figura 2, é o ponto de integração entre essas duas pesquisas, e representa seu protocolo de comunicação. Este enriquecimento constitui-se de instruções, em alto nível, para a reconfiguração do ambiente.

Cada heurística definida segue o modelo Evento-Condição-Ação. Os eventos descrevem situações que alertam o subsistema de reconfiguração para examinar se é conveniente efetivar uma reconfiguração. As condições servem para selecionar o modelo preditivo adequado, já existente, e explorá-lo. As ações definem quais parâmetros de configuração devem ser modificados, e como isso deve ser feito. Os modelos preditivos enriquecidos apenas sugerem reconfigurações. O subsistema de reconfiguração é que avalia se tais configurações são convenientes, considerando políticas de SLA que possam estar previamente definidas. O formato do protocolo de comunicação está definido em XML e, por limitações de espaço, não é apresentado nesse artigo.

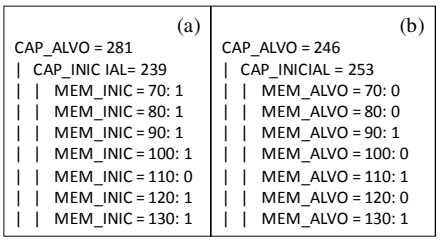
7. Teste do Ambiente Proposto

Buscando retratar situações reais, foram planejadas 539 configurações distintas para as execuções de *benchmarks*, cada uma composta por diferentes combinações de CPU, CAP e memória. Cada configuração contou com 4 VMs. Cada VM de cada configuração teve execução de um *benchmark* (Unixbench), totalizando em 2.156 execuções. Cada resultado do Unixbench produziu um total de 27 métricas e seus respectivos valores. Esses dados foram devidamente organizados no DW, resultando em um total de 58.212 registros na tabela fato. Para o teste feito, as execuções de benchmarks foram efetuadas em um único ambiente computacional. Por essa razão, a etapa de preparação dos dados para a mineração fez uso apenas dos atributos contidos nas dimensões Configuracao e MaquinaVirtual do DW (seção 4.1). Essas, entretanto, mostraram-se suficientes para realizar a mineração e produzir os resultados esperados.

Para os experimentos, o arquivo "arff" foi preparado obedecendo à política de mapear, para cada configuração inicial, todas as demais combinações para configuração alvo. Optou-se por separar distintos arffs para cada percentual de CPU. Foram criados 4 arquivos onde os atributos contidos nos mesmos foram: CAP inicial, CAP alvo, Memória inicial, Memória alvo, e o atributo preditivo. Optou-se, também, por representar os valores atribuídos aos atributos CAP de acordo com o primeiro número de configuração para um determinado conjunto de CAP. Para o exemplo da Tabela 1 (seção 3), o CAP do lado esquerdo (cuja combinação de valores é 10, 10, 20 e 45) foi mapeado para o valor 239; e o do lado direito (cuja combinação de valores é 10, 15, 25, 35) para o valor 281. Para os atributos Memória, mapeou-se o valor correspondente à memória da primeira VM para representar a combinação de memórias. Logo, os valores utilizados para memória foram: 70, 80, 90, 100, 110, 120 e 130. Por fim, o atributo preditivo é composto de valores 0 ou 1, onde 0 indica que a mudança de uma configuração para outra não apresenta melhora de desempenho, e 1 indica que tal mudança é recomendada.

Cada arquivo arff preparado é carregado no ambiente Weka e, sobre eles, é executado o algoritmo J48, utilizando o método de validação cruzada. Os modelos preditivos produzidos classificam situações que possam demandar reconfiguração. Para os experimentos efetuados, os modelos fizeram uso do atributo CAP alvo como raiz, derivando um ou mais atributos até atingir o atributo preditivo. A Figura 4 ilustra, em parte, duas diferentes situações encontradas no modelo preditivo gerado para o arquivo de 85% de CPU. Em (a) verifica-se que reconfiguração do CAP inicial 239 para o CAP alvo 281, com quaisquer configurações de memória, é recomendada, exceto quando a configuração de memória inicial for 110. Em (b) é possível constatar que, partindo-se do

CAP inicial 253, é recomendado reconfigurar para o CAP alvo 246, com quaisquer configurações de memória inicial, quando a memória alvo for 90, 110 e 130.



(a)

```
CAP_ALVO = 281
|  CAP_INIC IAL= 239
|  |  MEM_INIC = 70: 1
|  |  MEM_INIC = 80: 1
|  |  MEM_INIC = 90: 1
|  |  MEM_INIC = 100: 1
|  |  MEM_INIC = 110: 0
|  |  MEM_INIC = 120: 1
|  |  MEM_INIC = 130: 1
```

(b)

```
CAP_ALVO = 246
|  CAP_INICIAL = 253
|  |  MEM_ALVO = 70: 0
|  |  MEM_ALVO = 80: 0
|  |  MEM_ALVO = 90: 1
|  |  MEM_ALVO = 100: 0
|  |  MEM_ALVO = 110: 1
|  |  MEM_ALVO = 120: 0
|  |  MEM_ALVO = 130: 1
```

Figura 4. Exemplo de modelos preditivos

O formato apresentado na Figura 4 não é confortável para que o subsistema de reconfiguração o interprete, pois o mesmo precisa de uma estrutura de dados suficientemente simples e eficiente para não onerar sua execução. Para tanto, os resultados foram reformatados, de modo que o subsistema de reconfiguração possa identificar, rapidamente, uma situação de uma dada máquina Xen e configurá-la de acordo com os parâmetros sugeridos. A Tabela 2 mostra um exemplo de modelos preditivos enriquecidos. Esta contém parte dos resultados apresentados na Figura 4 e é estruturada da seguinte maneira: para uma dada configuração inicial são apresentadas apenas as configurações alvo benéficas para a reconfiguração. Esses resultados de configuração alvo são enriquecidos de modo que apenas os conteúdos de CAP e memória que se diferem da configuração inicial são indicados para a reconfiguração. Vale ressaltar que essa tabela não apresenta de forma exaustiva todas as possibilidades de configuração de memória inicial cuja reconfiguração é benéfica, sugeridas pelo modelo.

A primeira ocorrência da Tabela 2 diz respeito ao resultado do modelo apresentado na Figura 4a. Essa mostra CAP inicial 239 (10, 10, 20, 45) e, para fins de exemplificação, memória inicial 90. Com essa configuração inicial, o modelo indica que há benefício em configurar para o CAP 281 (10, 15, 25, 35) para quaisquer configurações de memória. As transições para estas reconfigurações estão apresentadas na configuração alvo e os parâmetros que devem ser alterados, nas instruções enriquecidas. A segunda ocorrência diz respeito ao resultado do modelo apresentado na Figura 4b. Essa mostra o CAP inicial 253 (10, 15, 20, 40) e, para fins de exemplificação, memória inicial 130. Com essa configuração inicial, o modelo indica que há benefício em reconfigurar para o CAP 246 (10, 10, 25, 40) quando a memória alvo for 90, 110 e 130. Assim, em configuração alvo, é apresentada apenas as transições que satisfaçam essa regra e, em instruções enriquecidas, os parâmetros que devem ser alterados.

O subsistema de reconfiguração avalia as instruções sugeridas e, dentre as que não violam regras de SLA, elege a que considerar mais apropriada. É importante ressaltar que as combinações de configurações de CAP e memória não são permutadas entre as VMs, de modo que as diferentes distribuições não estão reportadas nos modelos preditivos. Como conseqüência, o subsistema de reconfiguração torna-se responsável por avaliar a distribuição dessas combinações e aplicar a configuração adequada, pois o custo computacional necessário é menor do que o custo que o subsistema teria em percorrer todas as combinações de instruções, e selecionar a mais adequada.

Tabela 2. Exemplo de instruções enriquecidas

Configuração Inicial					Confiuguração Alvo								Instruções Enriquecidas							
					CAP				MEM				VM1		VM2		VM3		VM4	
	VM 1	VM 2	VM 3	VM 4	VM 1	VM 2	VM 3	VM 4	VM 1	VM 2	VM 3	VM 4	CAP	MEM	CAP	MEM	CAP	MEM	CAP	MEM
CAP	10	10	20	45	10	15	25	35	70	70	70	70		70	15		25		35	70
									80	70	70	60		80	15		25		35	60
									90	70	70	50			15		25		35	
MEM	90	70	70	50					100	70	70	40		100	15		25		35	40
									110	70	60	40		110	15		25	60	35	40
									120	70	50	40		120	15		25	50	35	40
									130	70	40	40		130	15		25	40	35	40
CAP	10	15	20	40	10	10	25	40	90	70	70	50		70	10		25	70		50
									110	70	60	40		110	10		25	60		
MEM	130	70	40	40					130	70	40	40			10		25	40		

8. Trabalhos Relacionados

Esforços recentes têm aplicado técnicas de mineração de dados para análise de desempenho em aplicações multicamadas. Estes trabalhos utilizam o Rubis[RUB07], um *benchmark* que simula uma aplicação multicamada de leilão, em ambientes de *data center*. Com mineração de dados, [JUN06] e [PAR06] buscam apontar gargalos e identificar, em quais situações e com quais configurações pode ocorrer queda de *performance*. [UDU07] faz uso de mineração de dados para definir políticas de SLA. [CHE07] apresenta um estudo semelhante, propondo a identificação de SLA a partir de SLO (*service level objectives*) já definidos. Neste último trabalho, é citada a utilização do Xen, em um *data center*, como cenário.

Não fica claro, nos trabalhos acima, se é adotado ou proposto um processo completo de KDD, em especial nas etapas de preparação de dados e seu apropriado armazenamento em um DW. Nós defendemos que organizar dados de *benchmarks* em um DW qualifica o processo como um todo por atender aos objetivos enunciados por Kimball [KIM02] de tornar a organização dos dados facilmente acessível, consistente e adaptável. Além disso, a organização em um DW já permite o uso de ferramentas OLAP. Por outro lado, os trabalhos acima também fazem uso do algoritmo J48 do Weka.

9. Conclusões e Trabalhos Futuros

Este trabalho apresentou um processo completo de KDD para apoio à reconfiguração de ambientes virtualizados, desde a execução de *benchmarks*, a compilação de seus resultados, alimentação no DW e a execução de algoritmos de mineração de dados, até a produção de um modelo preditivo enriquecido para uso por um subsistema de reconfiguração de máquinas virtualizadas Xen. O modelo analítico do DW construído mostrou-se adequado e convenientemente flexível para comportar os dados de execução de *benchmarks* em seus diferentes contextos. A organização desses dados tornou sua preparação para o algoritmo de mineração mais simples e confortável. Com a utilização da mineração de dados, foi possível gerar modelos e enriquecê-los para que possam, seletivamente, sugerir novos parâmetros de reconfiguração para ambientes virtualizados.

Como trabalhos futuros, as mesmas configurações endereçadas podem ser executadas para ambientes distintos. Os parâmetros de configuração desses ambientes também podem ser utilizados como atributos passíveis de alteração pela mineração de

dados. As políticas de SLA também podem ser agregadas ao modelo analítico e serem utilizadas pela mineração de dados, de modo a desonerar o subsistema de reconfiguração no momento de avaliar as possíveis estratégias de reconfiguração, e aplicá-las.

Agradecimentos: Este trabalho foi desenvolvido em colaboração com a HP Brasil P&D. Agradecemos ao professor Avelino Zorzo e aos acadêmicos Fabio Rossi, Juliano Potrich e Dionatan Korb pela colaboração no projeto de parceria. À Dra. Karin Becker agradecemos pelas valiosas contribuições nas versões preliminares deste artigo.

Referências

[BAR03] Barham et al. 2003. Xen and the art of virtualization. In: ACM Symposium on Operating Systems Principles. *Proceedings*… N.Y.: ACM Press, 2003, p. 164-177.

[BEC06] Becker K.; Ruiz, D.; Cunha, V.; Novello, T.; Souza, F. SPDW : a Software Development Process Performance Data Warehousing Environment. In: 30th Annual IEEE/NASA Software Engineering Workshop. *Anais* do 30th Annual IEEE/NASA SEW-06. Los Alamitos - CA : IEEE Press, 2006. v. 1. p. 107-118.

[CHE07] Chen, Y. et al. SLA decomposition: translating service level objectives (SLOs) to low-level system thresholds. In: International Conference on Autonimic Cmputing. *Proceedings*… Washington: IEEE Computer Society, Jun. 2007, p. 3.

[FAY96] Fayyad, U., Piatetsky-S G., Smyth P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, N.Y, v39, n11, p27-34, 1996.

[HAN01] Han, J.; Kamber, M. *Data mining*: Concepts and Techniques. San Francisco: Morgan Kaufmann, 2001.

[IOZ07] *Iozone*. < http://www.iozone.org>. Acesso out 2007.

[JUN06] Jung, G., et al. Detecting Bottleneck in n-Tier IT applications through analysis. In: International Workshop on Distributed System: operation and management. *Proceedings*… Dublin: Lecture Notes in Computer Science, Oct. 2006, p. 149-160.

[KIM02] Kimball, R., Ross, M. *The data warehouse toolkit*. São Paulo: Campus, 2002.

[LIR07] Li, T., Ruan, D. An extended process model of knowledge discovery in databases. *Enterprise Information Management*, v. 20, n. 2, p.169-177, 2007.

[MCV96] McVoy, L. W., Staelin, C. Lmbench: portable tools for performance analysis. In: Annual Technical Conference. *Proceedings*… San Diego: Usenix, 1996, p279-294.

[MER06] Mergen, M. F., Uhlig, V., Kreiger, O., Xendis, J. 2006. Virtualization for high-performance computing. SIGOPS. *Operating System. Review*, New York, v. 40, n. 2, p. 8-11, Apr. 2006.

[PAR06] Parekh, J., et al. Comparison of performance analysis approaches for bottleneck detection in multi-tier enterprise applications. In: International Workshop on Quality of Service. *Proceedings*… New Haven: IEEE Computer Science, Jun. 2006, p. 302-311.

[QUI96] Quinlan, J. R. *C4.5*: programs for machine learning. CA: Morgan Kaufman, 1996.

[RUB07] *RUBIS*: Rice University Bidding System. <rubis.objectweb.org/>. Acesso out 2007.

[TAN06] Tan, P-N., Steinbach, M., Kumar, V. *Introduction to data mining*. Boston: Addison Wesley, 2006.

[UDU07] Udupi, B., Sahai, A., Sinhal, S. A classification-based approach to policy refinement. In: Symposium on Integrated Management. *Proceedings*… IEEE Computer Science, May 2007, p. 785-788.

[UNI07] *Unixbench*. < www.tux.org/pub/tux/niemi/unixbench/>. Acesso out 2007.

[WEK07] *Weka*: Waikato Environment for Knowledge Analysis. <www.cs.waikato.ac.nz/ml/weka/>. Acesso out 2007.