SCIENTIA CUM INDUSTRIA(SCI. CUM IND.), V. 4, N. 1, PP. 15 — 20, 2016

Identificação de sistemas aplicado a máquinas virtuais: uma experiência com KVM, Xen e modelos ARIX

Guilherme Keiel*, Vinicius Binotti*, André Gustavo Adami*, Fernando Augusto Bender*

Resumo

A atual consolidação de servidores é possível graças a tecnologia da virtualização, garantindo isolamento, melhor aproveitamento dos recursos e reduzindo os seus custos operacionais. Este artigo analisa a aplicação de duas diferentes tecnologias de virtualização em servidores. Ambientes virtuais são criados com o *hypervisor* KVM e o Xen, que utiliza a paravirtualização. Através desses ambientes, experimentos de alocação dinâmica de memória para uma VM de teste são realizados. Uma vez conhecida a tecnologia que apresenta o melhor resultado, realiza-se uma avaliação entre as estruturas de modelos auto-regressivos utilizadas, sendo selecionada uma para representar o comportamento da máquina virtual. De posse do modelo escolhido, sua estabilidade é analisada e sugere-se a implementação de um esquema de controle utilizando o modelo preditor para a alocação dinâmica de memória a uma máquina virtual.

Palavras-chave
ARIX. KVM. Virtualização. Xen.

System Identification applied to virtual machines: an experiment with KVM, Xen and ARIX models

Guilherme Keiel*, Vinicius Binotti*, André Gustavo Adami*, Fernando Augusto Bender*

Abstract

The current consolidation of servers has been made possible due to the virtualization technology, assuring isolation, increasing the exploitation of the resources and reducing operating costs. This article analyses the application of two different virtualization technologies in servers. Virtual environments are created with the KVM hypervisor and with Xen, which uses paravirtualization. Through this virtual environments, experiments regarding dynamic memory allocation for a VM are performed. Knowing the technology with the best outcome, it was assessed which technique based on autoregressive modeling would best represent the virtual machine behavior. The stability of the selected model is analyzed and suggestions are made for applying control using the model for dynamic memory allocation in a virtual machine.

Keywords
ARIX. KVM, Virtualization. Xen.

I. Introdução

A técnica de virtualização está sendo apontada como uma das principais ferramentas para uma administração eficiente de centros de processamento de dados [1] [2]. Enquanto o uso de recursos em data centers tradicionais fica restrito em 5 a 20%, os servidores consolidados podem atingir um nível maior de utilização [1]. Essa racionalização do uso de servidores resulta no aumento da flexibilidade e redução da vulnerabilidade de um sistema computacional [1].

* Centro de Ciências Exatas e Tecnologia, Universidade de Caxias do Sul, RS, Brasil. email: gkeiel@ucs.br, vbinotti@ucs.br, agadami@ucs.br, fabender@ucs.br

Data de envio: 08/10/2015
Data de aceite: 12/12/2015

http://dx.doi.org/10.18226/23185279.v4iss1p15

São oferecidas diferentes soluções de virtualização no mercado e cada uma atende a requisitos específicos desejados em um ambiente virtualizado, seja a capacidade de VMs (*Virtual Machines*) instaladas, ou o suporte a diferentes sistemas operacionais. Nos ambientes virtualizados os *hypervisors* buscam obter um gerenciamento eficiente da memória entre as VMs em execução aplicando, basicamente, as técnicas do balonismo ou *memory hotplug* [3] [4]. Atualmente são propostos novos modelos de otimização, aplicando-se diferentes estratégias de gerenciamento diretamente no *hypervisor*. Autores propuseram diferentes algoritmos, como MEB (*Memory Balancer*) [5] para o monitoramento e predição de memória, métodos de alocação de memória sob diferentes critérios para cada VM [6] [7], e *overbooking* de recursos [8].

Este trabalho dá continuidade a um projeto de pesquisa iniciado por Krewer [9] e continuado por Barboza [10], os quais verificam, cada um, o comportamento de máquinas

virtuais geradas sob uma ferramenta de virtualização diferente. O objetivo geral do artigo é comparar o desempenho de um ambiente virtual com memória alocada dinamicamente entre duas diferentes tecnologias de virtualização, implementadas com o KVM e com o Xen, em uma mesma máquina física. Na etapa seguinte, com a estratégia que resultou no menor tempo de processamento, é obtido o modelo matemático do comportamento da VM. A obtenção desse modelo permitirá o conhecimento das características dinâmicas da máquina virtual em questão, o que é o primeiro passo para a criação de uma lei de controle visando o melhor direcionamento dos recursos de memória.

A. Virtualização

A virtualização possibilita que sistemas ou ambientes operacionais distintos compartilhem recursos do mesmo sistema físico [11]. As técnicas utilizadas da (i) virtualização total e (ii) paravirtualização se diferem pelo suporte dado ao sistema operacional, no desempenho e no nível de acesso aos recursos de hardware. Na virtualização total a camada de abstração roda juntamente com o sistema operacional, repassando instruções para o hardware real, e acima dela rodam as VMs [11] [12]. Na paravirtualização a camada virtualizada é o próprio sistema operacional modificado para a hospedagem das VMs [13] [14], o que torna a interação com o *hypervisor* mais eficiente, mas em contrapartida exige que sejam feitas modificações nos sistemas operacionais das VMs convidadas [13]. Entre as ferramentas que fornecem a virtualização serão analisados os *hypervisors* KVM e Xen, que implementam as técnicas distintas da virtualização total e da paravirtualização, respectivamente.

O KVM foi a primeira infraestrutura de virtualização a fazer parte do *kernel* nativo do Linux, utilizando *drivers* já existentes para obter o acesso ao hardware [15] [12]. É implementada em dois componentes [12]: o módulo KVM, o qual fornece o gerenciamento de hardware de virtualização e uma versão modificada do *driver* QEMU, que permite a emulação de plataformas virtuais (Figura 1). Para o gerenciamento da memória utiliza o compartilhamento de páginas de memória (KSM) e o balonismo [12]. O KSM substitui faixas idênticas de memória por uma única página, até que elas sejam solicitadas para modificação novamente. O balonismo possibilita um compartilhamento da memória total dimensionada para o *hypervisor* entre as suas VMs [3] [12]. Consiste em um *driver* especial instalado em cada sistema operacional convidado, cooperando com o *hypervisor*. Dá a ilusão de que se possui mais recurso de memória do que o que é disponível realmente, caracterizando o chamado *memory overcommitment*.

O XenServer é um software com a ferramenta da paravirtualização Xen voltada para a aplicação em servidores, não necessitando de um suporte nativo a virtualização pelo processador [16]. No Xen as requisições feitas pelas VMs em funcionamento, instaladas sob o domínio U, são repassadas ao sistema principal (Domínio 0) sem a necessidade de serem interpretadas, causando pouca interferência no desempenho do sistema [13] [14] (Figura 2). O Xen oferece o DMC (*Dynamic Memory Control*) que, através do balonismo, possibilita a realocação de uma quantidade de memória a uma VM em execução (a memória dinâmica) sem a necessidade da sua reinicialização [16] [13].

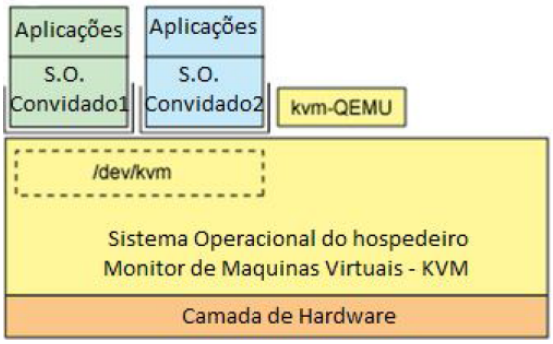

Fig. 1: Estrutura do *hypervisor* KVM [12].

O incoveniente do balonismo é que ele é restringido pela capacidade máxima de memória que é destinada na criação da VM [3]. A *memory hotplug* é uma técnica complementar ao balonismo, a qual permite essa alteração da capacidade máxima atribuída a VM enquanto ela estiver em execução [3].

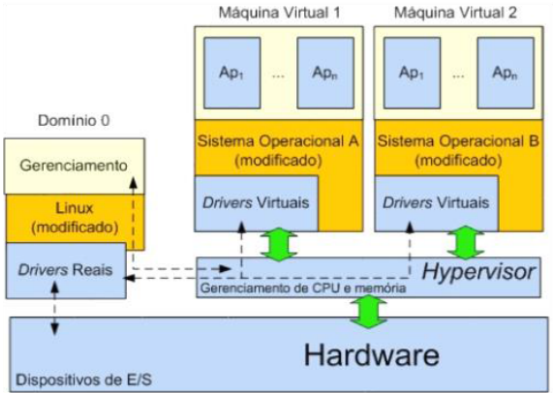

Fig. 2: Estrutura do *hypervisor* Xen [17].

B. Identificação de Sistemas

O levantamento do modelo é executado utilizando uma abordagem do tipo caixa preta [18], pois devido a complexidade do sistema, não é possível obter todos os modos que caracterizam o seu comportamento por um modelo físico exato.

A partir da estrutura de modelos geral utilizada na identificação de sistemas [19] propõe-se o modelo ARX (autoregressivo com entradas exógenas) [18], utilizado para a modelagem matemática de um sistema linear invariante no tempo (LIT) com adição de um sinal estocástico. A relação entre a entrada e a saída do sistema é definida através de (1):

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}v(k) \tag{1}$$

Os polinômios possuem n coeficientes. $B(q)/A(q)$ e $1/A(q)$ são os filtros da parte determinística e da parte estocástica do sistema, $u(k)$ a entrada exógena, $y(k)$ a saída e $v(k)$ a perturbação, originada de um processo estocástico [18] [20]. Com os resultados das entradas e saídas anteriores filtradas pelos parâmetros dos polinômios $A(q)$ e $B(q)$, a saída na iteração k pode ser estimada por (2).

$$\begin{aligned}\hat{y}(k) \quad = \quad & -a_1 y(k-1) \cdots - a_n y(k-n_a) + b_1 u(k-1) \\ & \cdots + b_n u(k-n_b)\end{aligned} \tag{2}$$

A estimação dos parâmetros é realizada pelo método dos mínimos quadrados, que por sua essência minimiza a norma euclideana do vetor de erros de predição [18] [19]. Este é o somatório dos quadrados dos erros entre as saídas medidas e as preditas pelo modelo [18].

No modelo ARMAX (auto-regressivo com média móvel e entradas exógenas), da equação (3), o erro de predição $e(k)$ é processado por um filtro de média móvel (MA) [20]. Portanto, a equação de diferenças utiliza também, além das entradas e saídas passadas, os erros de predição anteriores na estimação da próxima saída. A janela da média móvel é dada pelo número de coeficientes do polinômio $C(q)$ (4). Como o método de obtenção dos parâmetros é não linear, ali são utilizados algoritmos recursivos para a minimização numérica do custo, entre eles os métodos de Gauss-Newton, Levenberg-Marquardt e *descent gradient* [18].

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{C(q)}{A(q)}e(k) \tag{3}$$

$$\begin{aligned} \hat{y}(k) \quad = \quad & -a_1 y(k-1) \cdots - a_n y(k-n_a) \\ & +b_1 u(k-1) \cdots + b_n u(k-n_b) \\ & +e(k) + c_1 e(k-1) \cdots + c_n e(k-n_c) \end{aligned} \tag{4}$$

Quando o processo estocástico não é estacionário são usados os modelos ARIX (auto-regressivo integrador com entrada exógena) e ARIMAX (auto-regressivo integrador com média móvel e entrada exógena) [18]. São adaptações dos modelos anteriores, onde substitui-se $y(k)$ e $u(k)$ pelas suas diferenças, normalmente de primeira ordem, $\Delta y(k) = y(k) - y(k-1)$ e $\Delta u(k) = u(k) - u(k-1)$. O resultado, então, é usado na estimação dos parâmetros pelo método dos mínimos quadrados ou pelos métodos recursivos (se houver MA). Após obtidos os parâmetros, as saídas $\Delta\hat{y}(k)$ estimadas deverão ser integradas (somadas no tempo discreto) ao termo diferenciado anteriormente para a recuperação da saída $\hat{y}(k)$ [18] [20].

II. METODOLOGIA

O trabalho aborda a implementação de um ambiente virtualizado com ambas as tecnologias, da virtualização total e da paravirtualização, na mesma máquina física. Onde foi desenvolvido o algoritmo para a coleta dos dados da VM a cada iteração e o *script* para o gerenciamento do *hypervisor*.

A memória foi definida no intervalo de 256 a 1536 MB para a primeira realização e entre 256 e 768 MB, na segunda execução. O sistema possui uma dinâmica rápida, em poucas amostras (3 ou 4) ele já segue um valor de regime (para uma excitação do tipo impulso). Foi optado por uma realização com 200 amostras a fim de obter, com um número razoável de repetições, essa dinâmica. A mesma sequência aleatória originada foi utilizada em ambos os levantamentos, com o KVM e o Xen.

Criou-se um processo padrão, que consiste em inversões matriciais, e este foi executado em um laço contínuo dentro do ambiente virtualizado [9]. A partir daí são obtidos os tempos levados para cada execução desse processo, juntamente com a mémoria atribuída à VM, e estes são enviados ao sistema operacional hospedeiro através de uma conexão *socket*. A cada iteração é alterada a memória atribuída à VM em questão. A solução para a alteração automática de memória no Xen foi a utilização da pilha de ferramentas XAPI, disponibilizada pela própria desenvolvedora (Citrix), possibilitando o acesso direto a comandos de gerenciamento de recursos do *hypervisor* em uma linguagem de baixo nível. Realizado com a criação de um *script* no software PowerShell, apresentado no código seguinte:

```
1. Connect-XenServer -url https://192.168.
   0.102
2. $x=1
3. do{
4.     $memoria= .\ale $y
5.     Set-XenVM -UUID 9841895a-27ba-0da5-
   9ea6-81015946581a MemoryDynamicMax
   ($memoria)
6.     Start-Sleep -s 15
7.     $tempo= .\client 192.168.0.100
8.     .\escreve1 $memoria $tempo
9.     $x++
10.}while($x -le 200)
11.Disconnect-XenServer
```

Na etapa de identificação, a série obtida é avaliada quanto a sua estacionariedade e depois realizado o processo de identificação com as estruturas de modelos apresentadas, variando-se o número de parâmetros utilizados. A equação (5) exemplifica a álgebra para o caso do modelo ARX com 1 parâmetro para o polinômio A e 1 parâmetro em B. Esta deve ser expandida para mais coeficientes de acordo com a ordem da regressão a ser utilizada no cálculo [18]. Por fim, é realizada uma análise quantitativa dos erros de predição obtidos com todos os modelos auto-regressivos utilizados, a fim de determinar o melhor modelo [19].

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = A^{-1} \begin{bmatrix} -\sum_{n=1}^{N} y(k)y(k-1) \\ \sum_{n=1}^{N} y(k)u(k-1) \end{bmatrix} \tag{5}$$

onde

$$A = \begin{bmatrix} \sum_{n=1}^{N} y(k-1)^2 & \sum_{n=1}^{N} y(k-1)u(k-1) \\ \sum_{n=1}^{N} y(k-1)u(k-1) & \sum_{n=1}^{N} u(k-1)^2 \end{bmatrix}$$

Obteve-se cada modelo ARIX alterando-se a quantidade de coeficientes em A e B igualmente, em números ímpares, até o limite de 65 parâmetros. Para cada um desses modelos obtidos são estimadas as saídas seguintes, $\hat{y}(k)$, orientados pelos valores medidos passados da saída e da entrada. Assim, obtêm-se o vetor de erros de predição onde $e(k) = y(k) - \hat{y}(k)$. O comando utilizado na obtenção dos modelos ARIX e o posterior cálculo dos erros de predição foi escrito em Matlab na seguinte forma:

```
1. SYS= iddata(Y,U,1);
2. ORDERS= [na nb nk];
3. model= arx(SYS, ORDERS,
  'IntegrateNoise', true)
4. [SYS2]= compare(SYS,model,1);
5. y= get(SYS2,'OutputData');
6. mean_abs_error= mean(abs(Y-y))
7. max_error= max(abs(Y-y))
```

III. Resultados experimentais

A Figura 3 apresenta as duas séries, com 200 amostras cada, obtidas com as ferramentas KVM e Xen. A alocação dinâmica foi gerada em uma faixa definida entre 256 e 1536 MB.

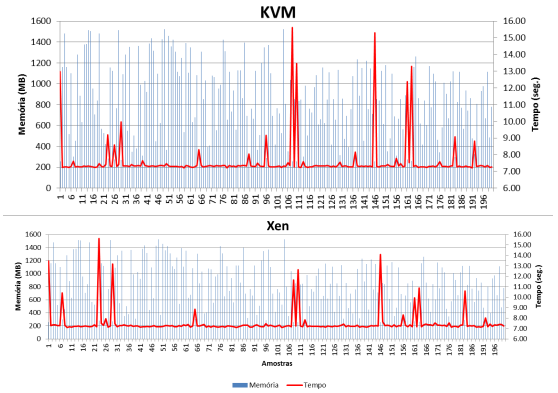

Fig. 3: Resposta à entrada aleatória de 256 à 1536 MB.

O resultado do segundo ensaio com ambas as ferramentas de virtualização pode ser observado na Figura 4. A faixa de memória utilizada, agora no intervalo menor, entre 256 e 768 MB.

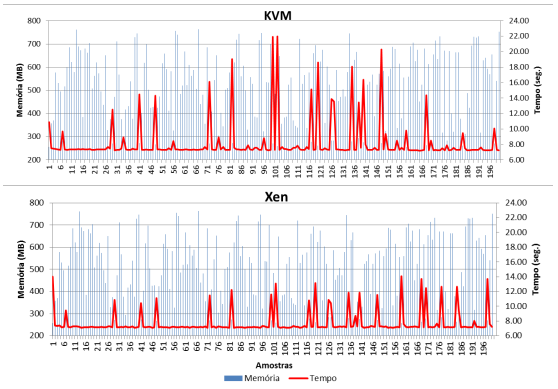

Fig. 4: Resposta à entrada aleatória de 256 à 768 MB.

Foi verificado que a estratégia de virtualização implementada pelo Xen apresentou tempos de processamento inferior aos da ferramenta KVM, dado as mesmas condições de ensaio. Quando comparadas com as amostras mais críticas das séries de dados, é visível os valores menores obtidos com o Xen. Este ensaio mostra uma maior rapidez da tecnologia da paravirtualização nas requisições ao hardware.

Foi obtida uma nova série utilizando a estratégia do Xen, de forma a usar uma entrada o suficientemente informativa, para excitar a maior quantidade possível de modos existentes no sistema. A Figura 5 apresenta a nova realização, usada na etapa de identificação do sistema.

A série obtida apresenta uma não estacionariedade, verificada pela tendência no decorrer da série. Isso levou a utilização dos modelos ARIX e ARIMAX, os quais trabalham considerando a não estacionariedade durante o processo de regressão. A Tabela 1 apresenta os erros de predição obtidos para cada modelo gerado, a fim de avaliar a qualidade dos modelos. São mostrados os erros absolutos médios e também o serros máximos com cada um dos modelos ARIX criados.

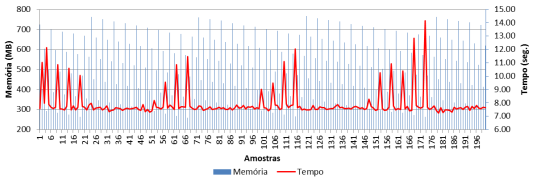

Fig. 5: Resposta à entrada aleatória no Xen (série 1).

Tabela 1: Erros obtidos com os modelos ARIX.

Parâmetros em A e B	Erro abs. médio	Erro máximo
5	0.3217	4.2466
11	0.3012	4.2688
15	0.3039	3.8833
21	0.2948	3.0988
25	0.2677	2.7481
31	0.2430	2.8706
35	0.2262	2.5385
41	0.2075	2.4996
45	0.1940	2.3347
51	0.1856	2.0719
55	0.1313	0.7522
61	0.1048	0.4206
65	0.0543	0.3384

A Tabela 2 avalia a qualidade dos modelos ARIMAX gerados. A primeira coluna informa a ordem dos modelos, expressa pelo número de parâmetros em A e B. Nas demais colunas, para cada ordem foram gerados outros modelos aumentando-se o número de parâmetros em C. Para cada um desses modelos, a primeira de suas linhas informa o erro médio e a segunda o erro máximo daquele modelo.

Tabela 2: Erros obtidos com os modelos ARIMAX.

	Quantidade de parâmetros em C							
	5	11	15	21	25	35	41	45
5	0.3248	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	4.2501	N/A	N/A	N/A	N/A	N/A	N/A	N/A
11	0.3253	0.3268	N/A	N/A	N/A	N/A	N/A	N/A
	3.7531	3.1690	N/A	N/A	N/A	N/A	N/A	N/A
15	0.3157	0.3335	0.3251	N/A	N/A	N/A	N/A	N/A
	3.0828	3.1142	3.1687	N/A	N/A	N/A	N/A	N/A
21	0,2991	0,2992	0,2924	0,8932	N/A	N/A	N/A	N/A
	2,9821	3,4797	3,1719	4,8747	N/A	N/A	N/A	N/A
25	0,2617	0,2743	0,2819	0,3140	0,2765	N/A	N/A	N/A
	2,7673	2,7552	2,7607	2,6831	2,5835	N/A	N/A	N/A
31	0,2386	0,2530	0,2518	0,2909	0,2528	N/A	N/A	N/A
	2,8462	2,6221	2,4121	2,8223	2,5401	N/A	N/A	N/A
35	0,2160	0,2347	0,2820	0,2874	0,2541	0,2532	N/A	N/A
	2,4864	2,4365	2,7582	3,0236	2,2812	2,0251	N/A	N/A
41	0,2189	0,2117	0,2305	0,2067	0,2545	0,5826	0,3264	N/A
	1,9225	2,1263	2,0117	1,7037	1,7870	2,2451	1,4698	N/A
43	0,2185	0,2016	0,2353	0,2697	0,2376	0,2573	0,2797	N/A
	2,2185	1,3510	1,7241	1,9175	2,1455	1,5759	1,6134	N/A
45	0,1963	0,2024	0,2135	0,2578	0,2858	2,6373	0,3377	0,5047
	1,8404	1,8275	1,8281	1,6578	1,7769	12,762	2,4610	2,3819
47	0,2223	0,2039	0,1953	0,3110	0,3199	0,4245	0,4272	0,3401
	2,3713	1,7990	1,4549	3,7768	1,4499	1,7555	1,5977	1,9936
49	0,1870	0,3351	0,2555	0,2433	0,3205	0,4014	0,8816	0,5452
	1,5670	1,4544	1,5160	2,1072	1,8840	2,5953	3,7128	2,3674
51	0,1628	0,1932	0,9053	0,5707	0,1960	0,3332	N/A	N/A
	1,4143	1,0271	3,5620	2,1353	2,0144	1,5994	N/A	N/A
55	0,1326	0,1878	0,1426	0,3462	0,2648	N/A	N/A	N/A
	0,8495	2,3596	1,5409	4,4340	1,4698	N/A	N/A	N/A
61	0,0941	0,3468	0,2827	N/A	N/A	N/A	N/A	N/A
	0,6207	7,8091	3,6455	N/A	N/A	N/A	N/A	N/A

A maior parte dos modelos ARIX apresentaram menor custo que os modelos ARIMAX. Entre as duas tabelas apresentadas o modelo ARIX com 65 parâmetros apresentou o menor resultado (Figura 6), com o erro absoluto médio de 0,0543 e erro máximo de 0,3384 em relação a série original. É notável que, nesse caso, a qualidade do modelo aumenta com

o incremento do número de parâmetros, porém na prática pode ser inviável implementar modelos muito extensos.

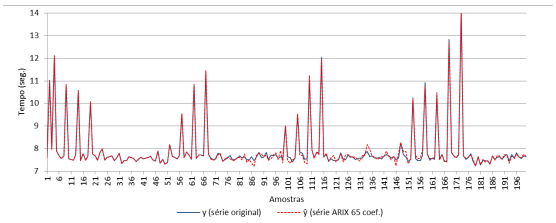

Fig. 6: Série obtida com modelo ARIX (65 coeficientes).

A validação do modelo é realizada avaliando a qualidade do modelo na predição de outra série de dados. Para isso obteve-se uma nova realização da série no Xen (Figura 7).

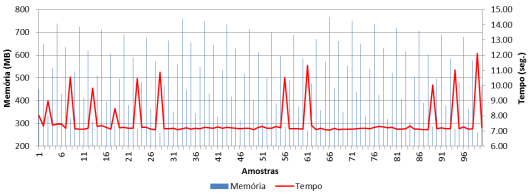

Fig. 7: Resposta à entrada aleatória no Xen (série 2).

Com cada um dos modelo ARIX já obtidos anteriormente foram estimadas as saídas da nova série e obtidos os erros associados (Tabela 3). Os modelos com muitos parâmetros apresentam, até o número de 17 coeficientes, o erro médio de predição decrescente. Porém tais modelos já começam a ficarem sobreparametrizados, o que é notado pelo cancelamento entre alguns de seus polos e zeros.

Tabela 3: Erros obtidos na validação.

Parâmetros em A e B	Erro abs. médio	Erro máximo
3	0,8090	4,2528
5	0,4446	3,0619
7	0,4172	3,1381
9	0,4047	3,1124
11	0,3820	3,0829
13	0,3606	3,0130
15	0,3468	2,5678
17	0,3178	2,7044
19	0,3338	2,6683
21	0,3689	2,6292
23	0,3506	2,7059
25	0.3808	2,6479
27	0.4413	2,6576
29	0.4557	2.7451
31	0.4736	2,8151

O modelo ARIX selecionado (6), com 15 parâmetros, apresentou o menor erro máximo e segundo menor custo associado. Analisando o modelo quanto a sua estabilidade: constatou-se a estabilidade entrada-saída do filtro, que atende a condição de estabilidade BIBO (*Bounded-Input Bounded-Output*) para sistemas de tempo discreto [21], devido a todos polos possuírem módulo menor que 1 e, portanto, estarem localizados no interior do círculo de raio unitário (Figura 8).

IV. Desafios e trabalhos futuros

Depois de validado o modelo parte-se para uma aplicação mais prática, inserindo uma lei de controle com base nas

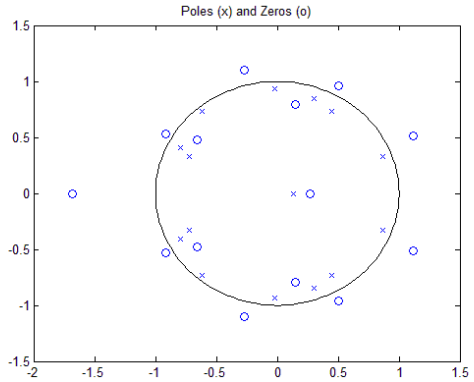

Fig. 8: Polos e zeros do modelo escolhido.

estimativas do modelo. Em que uma amostragem da utilização de memória deve ser efetuada em tempo real pelo algoritmo no domínio 0 do *hypervisor* Xen. Obtêm-se uma previsibilidade da utilização de recurso para cada iteração subsequente, através das predições geradas pelo modelo. Com isso é possível tomar ações de controle apropriadas, destinando mais ou menos memória para a máquina virtual em questão.

V. Conclusão

Partindo de esforços realizados em trabalhos anteriores [9], o primeiro passo deste artigo consistiu no desenvolvimento e implementação de um *script* para o acesso ao *hypervisor* Xen de forma automática. Foi criado um ambiente virtual com ambas as estratégias, KVM e Xen, e verificado os resultados em cada um. Constatou-se que o *hypervisor* Xen apresentou um desempenho melhor do que a estratégia de virtualização total, com o KVM. De posse dos dados, foram realizadas identificações com ambas as estruturas de modelo ARIX e ARIMAX para diferentes parâmetros, afim de avaliar a qualidade do modelos. Foi avaliada a estabilidade e realizada a validação do modelo escolhido.

Após a obtenção de um modelo significativo pode-se projetar uma lei de controle apropriada, onde é possível utilizar esse modelo preditivo para obter uma estimativa de quanto recurso deve ser destinando a VM em questão.

VI. Referências

[1] W. Vogels, "Beyond server consolidation," *ACM Queue*, 2008.

[2] Yichao Jin, Yonggang Wen, and Qinghua Chen, "Energy efficiency and server virtualization in data centers: An empirical investigation," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, March 2012, pp. 133–138.

[3] Haikun Liu, Hai Jin, Xiaofei Liao, Wei Deng, Bingsheng He, and Cheng-Zhong Xu, "Hotplug or ballooning: A comparative study on dynamic memory management techniques for virtual machines," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, no. 5, pp. 1350–1363, May 2015.

[4] S.S. Pinter, Y. Aridor, S. Shultz, and S. Guenender, "Improving machine virtualization with 'hotplug memory'," in *Computer Architecture and High Performance Computing, 2005. SBAC-PAD 2005. 17th International Symposium on*, Oct 2005, pp. 168–175.

[5] Luo Y. Zhao W., Wang Z., "Dynamic memory balancing for virtual machines," *ACM SIGOPS Operating Systems Review*, 2009.

[6] Weizhe Zhang, Tao Cheng, Hui He, and A.M.K. Cheng, "Lvmm: A lightweight virtual machine memory management architecture for virtual computing environment," in *Uncertainty Reasoning and Knowledge Engineering (URKE), 2011 International Conference on*, Aug 2011, vol. 1, pp. 235–238.

[7] Hongwu Lv Guangsheng Feng Zhanbo He Guilin Zhang, Huiqiang Wang, "A dynamic memory management model on xen virtual machine," *International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, 2013.

[8] J. Heo, Xiaoyun Zhu, P. Padala, and Zhikui Wang, "Memory overbooking and dynamic control of xen virtual machines in consolidated environments," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, June 2009, pp. 630–637.

[9] F. J. KREWER, "Identificação de sistemas aplicados a máquinas virtuais: Uma experiência com arx e kvm," Trabalho de Conclusão (Tecnologia em Automatização Industrial) – Universidade de Caxias do Sul, Caxias do Sul, 2013.

[10] F. F. V. BARBOZA, "Identificação de sistemas aplicados a máquinas virtuais 2: Uma experiência com arx e xenserver," Trabalho de Conclusão (Tecnologia em Automatização Industrial) – Universidade de Caxias do Sul, Caxias do Sul, 2014.

[11] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[12] T. JONES, "Anatomia de um hypervisor linux," IBM, 2009, Disponível em: <http://www.ibm.com/developerworks/br/library/l-hypervisor>. Acesso em: set. 2014.

[13] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Oct. 2003.

[14] F. D. ROSSI, "Alocação dinâmica de recursos no xen," Dissertação (Mestrado em Ciência da Computação) – Pontifica Universidade Católica do Rio Grande do Sul, 2008.

[15] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori, "kvm: the linux virtual machine monitor," in *Proceedings of the Linux Symposium*, Ottawa, Ontario, Canada, June 2007, vol. 1, pp. 225–230.

[16] D. Tosatto, "Citrix xenserver 6.2.0 administration guide," 2013, Disponível em: <http://rwmj.wordpress.com/20120/07/17/virtio-balloon>. Acesso em: set. 2014.

[17] F. N. N. Farias et al., "Pesquisa experimental para a internet do futuro: Uma proposta utilizando virtualização e o frame-work openflow," XXIX Simpósio de Redes de Computadores e Sistemas Distribuídos - SBRC, 2011.

[18] L. LJUNG, *System Identification: Theory for the User*, vol. 1, Prentice Hall, New Jersey, NJ, 1999.

[19] T.S. Soderstrom and P.G. Stoica, *System Identification*, Prentice Hall International Series In Systems And Control Engineering. Prentice Hall, 1989.

[20] L. A. Aguirre, *Introdução à identificação de sistemas: técnicas lineares e não-lineares aplicadas a sistemas reais*, vol. 1, Editora UFMG, Belo Horizonte, MG, 2007.

[21] Chi-Tsong Chen, *Linear System Theory and Design*, Oxford University Press, Inc., New York, NY, USA, 2nd edition, 1995.