A influência da virtualização no planejamento de capacidades para ambientes de computação em nuvem e aplicações distribuídas baseadas em SOA.

Luís Cézar Darienzo Alves[1], Marcos José Santana[1] ,
Regina Helena Carlucci Santana[1], Wilnice Tavares Reis Oliveira[2].

[1]Instituto de Ciências Mátematicas e de Computação.
Universidade de São Paulo (ICMC-USP).
Av. Trabalhador são-carlense, 400 - Centro CEP: 13566-590.
São Carlos – SP – Brasil.

[2]Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso (IFMT).
Cuiabá – MT – Brasil.

{luisc,msj,rcs}@icmc.usp.br, wilnice.oliveira@cba.ifmt.edu.br

Abstract. *This paper describes the impact of virtualization of processors and network interfaces in SOA-based applications performance, as well as the slowdown caused by SOAP envelope encoding and decoding libraries, when running in cloud computing environments, following the Infrastructure as a Service (IaaS) model. Such an influence is analyzed based on workloads mainly consisted of cpu-bound and memory-intensive applications and the results are fundamental for capacity planning of the platforms under consideration.*

Resumo. *Este artigo descreve o impacto gerado pela virtualização de processadores e interfaces de rede no desempenho de aplicações baseadas em SOA, bem como as perdas de desempenho sofridas pelas bibliotecas de codificação e decodificação de envelopes SOAP, quando tais aplicações são executadas em ambientes de computação em nuvem do tipo Infraestrutura como Serviço (IaaS). Tal influência é analisada com base em cargas de trabalho formadas, predominantemente, por aplicações CPU-bound e memory-intensive, sendo os resultados obtidos fundamentais para o planejamento de capacidade das plataformas consideradas.*

1. Introdução

Com o surgimento da computação em nuvem e a implantação de soluções baseadas em SOA (*Service-oriented architecture*) em ambientes do tipo IaaS (*Infrastructure as a Service*), tornou-se mais complexo o dimensionamento adequado da infraestrutura de TI (Tecnologia da Informação) a fim de cumprir SLAs (*Service-Level Agreement*) previamente estabelecidos, fornecendo, assim, serviços com níveis satisfatórios de qualidade.

Neste contexto, faz-se necessário o uso do planejamento de capacidade, que pode ser definido como o processo de prever os níveis futuros de carga de um sistema, permitindo, através da solução mais econômica, adiar ao máximo a sua saturação. Para isso, a previsão precisa considerar as características essenciais do sistema que definem o seu desempenho [Menasce and Almeida 2003].

Sendo assim, considerando que a virtualização é o principal fator na redução de desempenho em soluções de computação em nuvem, constituindo característica fundamental em plataformas do tipo Infraestrutura como serviço, este trabalho propõe analisar seu impacto na perda de desempenho de aplicações baseadas em SOA, implantadas em servidores de aplicação que, fundamentalmente, são executados sobre tais plataformas [Baun et al. 2011].

Desta forma, esta análise permitirá que os modelos de planejamento de capacidade comumente utilizados sejam atualizados, a fim de refletir essa nova realidade no setor da TI mundial, considerando os recursos virtualizados e as atividades inerentes às aplicações baseadas em SOA, tais como as operações de *marshalling* e *unmarshalling* de mensagens SOAP (*Simple Object Access Protocol*).

2. Trabalhos relacionados

Muitos trabalhos estão sendo desenvolvidos envolvendo planejamento de capacidade e avaliação de desempenho. Em [Praphamontripong et al. 2007], os autores, usando mecanismos assíncronos, propõem um modelo de análise de desempenho para servidores *Web* implementados com processamento concorrente. Em [Shimizu et al. 2009] é apresentada uma abordagem flexível para modelar aplicações, cujo objetivo é prever a necessidade de recursos de cada aplicação independente de uma plataforma de *hardware*.

Esses trabalhos, no entanto, não consideram as aplicações baseadas em SOA, que são abordadas em [Estrella et al. 2008], onde os autores apresentam diretrizes importantes para a avaliação de desempenho em *Web Services*, destacando os problemas com as especificações dos padrões utilizados, o que influencia, de forma negativa, na provisão da qualidade de serviço. Já em [Teixeira et al. 2011] é proposta uma abordagem, baseada em SLA e modelos estocásticos, para prever o consumo de recursos e a degradação do sistema ao utilizar vários tipos de carga de trabalho.

Ainda considerando o planejamento de capacidade, tendo como base as aplicações baseadas em SOA, os autores do trabalho [Zhang et al. 2007], de uma maneira mais específica, consideram as organizações em múltiplas camadas desses sistemas, apresentando metodologias para prever o impacto de cargas futuras em sistemas de múltiplas *threads* e múltiplos nós. Em [Lu et al. 2010], os autores apresentam uma nova política para o controle de admissão, baseada em sessões, para sistemas de *Web Services* estruturados em múltiplas camadas, denominada AWAIT, permitindo reduzir, para o mínimo, a quantidade de sessões abortadas. Em [Roy et al. 2011], os autores descrevem uma abordagem híbrida para o planejamento de capacidade de sistemas multicamadas baseados em componentes, que, através de heurísticas, determina a menor quantidade de recursos que devem ser alocados para um determinado componente.

Da maneira análoga, o planejamento de capacidade abordando computação em nuvem também tem sido objeto de estudos recentemente, como em [Iosup et al. 2011], onde os autores analisam o desempenho de sistemas, comerciais, de computação em nuvem para tratar cargas de trabalho científicas. Em [Chi et al. 2011], os autores apresentam um novo *framework*, denominado *SLA-Tree*, utilizado na tomada de decisão para cumprimento de SLAs. Em [Rao et al. 2011], os autores propõem um mecanismo de provisionamento auto-adaptativo de máquinas virtuais, comprovando a eficiência da solução através de um protótipo executado sobre a plataforma *Xen*.

Desta forma, apesar dos trabalhos presentes na literatura, este artigo se destaca pela abordagem inovadora ao considerar o impacto de sistemas de computação em nuvem no desempenho de aplicações baseadas em SOA, uma vez que tais aplicações são, frequentemente, implantadas em servidores de aplicação, elevando a quantidade de *softwares* cujos desempenhos são degradados pelo processo de virtualização. Tal análise permitirá que trabalhos futuros adaptem os modelos de planejamento de capacidade existentes para que reflitam, de forma realista, o comportamento das soluções empregadas.

3. Metodologia

Para avaliar a influência da virtualização no desempenho de aplicações baseadas em SOA, desenvolveu-se *Web Services* que representam aplicações *CPU-bound* e *memory-intensive*, sendo os mesmos implantados em servidores de aplicação executados em diferentes plataformas de virtualização.

Complementarmente, desenvolveu-se um *software* cliente, responsável pelo consumo dos *Web Services* e pelas aferições dos tempos de resposta, sendo executadas, em média, mil submissões para cada configuração de plataforma e serviço, ambas descritas posteriormente, com confiança de 95% e mantendo 10% dessas submissões como *warm-up* [Jain 1991].

Adicionalmente, é importante destacar que para todos os resultados apresentados, não houve sobreposições das médias e intervalos de confiança.

Assim, para uma completa compreensão da metodologia adotada, nas subseções a seguir serão descritas as plataformas e as cargas de trabalho utilizadas.

3.1. Plataforma

Ao total foram estabelecidas três plataformas computacionais nas quais as aplicações SOA foram implantadas. Todas são formadas pela mesma configuração de *hardware*: processador Intel Core I5® 2430M, com dois núcleos de 2.4GHz, 3 MB de *cache* e 4GB de memória DRAM (*Dynamic Random Access Memory*).

A primeira plataforma, denominada plataforma de referência, é caracterizada pela execução do sistema operacional GNU/Linux diretamente sobre o *hardware*, não havendo, assim, a presença de gerenciadores de máquinas virtuais. Essa plataforma, assim como as demais, executa o servidor de aplicação Oracle Glassfish®. A escolha de tal servidor foi motivada pelo fato dessa implementação ser a de referência para a arquitetura JEE (*Java Enterprise Edition*) [Marzullo 2009].

As demais plataformas, denominadas Plataforma 1 e Plataforma 2, executam, respectivamente, os gerenciadores de máquinas virtuais VMware ESXi® e Microsoft Hyper-v® sobre o *hardware* descrito anteriormente, sendo compostos de uma máquina virtual formada pelo sistema operacional GNU/Linux e os demais *softwares* presentes na plataforma de referência. É importante destacar que os *kits* de integração foram instalados nos sistemas operacionais convidados em ambas às plataformas.

Finalmente, deve-se observar que a escolha dos gerenciadores foi baseada na ampla utilização de ambos no desenvolvimento de soluções do tipo IaaS, tanto em ambientes corporativos quanto acadêmicos [Baun et al. 2011].

3.2. Carga de trabalho

A partir da metodologia utilizada em [Xiao et al. 2002, Shimizu et al. 2009, Ueno and Tatsubori 2006, Teixeira et al. 2011], onde algoritmos reais são utilizados para representar categorias de aplicações, optou-se por desenvolver três *Web Services* que, a partir de quatro operações, implementam os algoritmos selecionados nos trabalhos citados com o objetivo de representar as classes de aplicações *CPU-bound* e *memory-intensive* [Alves et al. 2009].

1. Ordenação de vetores: O primeiro *Web Service* é composto por uma única operação que realiza a ordenação de vetores pelo método *Quick Sort*. De acordo com [Xiao et al. 2002], os algoritmos de ordenação de vetores são importantes representantes das classes de aplicação *CPU-bound* e *memory-intensive*. A seleção desse algoritmo, especificamente, ocorreu devido à sua classificação como eficiente e por ser amplamente utilizado [Ziviani 2011]. Como configuração, utilizou-se $200.000 \leq N \leq 400.000$, onde N representa a quantidade de elementos a serem ordenados. Nesta configuração, à medida que o valor de N aumenta, a aplicação deixa de ser classificada unicamente como *CPU-bound* se tornando, também, *memory-intensive* de carga baixa.
2. Multiplicação de matrizes: Este *Web Service* é composto de uma operação de multiplicação de matrizes quadradas, cujas ordens variam entre 500 e 2000. Nesse tipo de aplicação, para matrizes de ordem pequena, a carga de trabalho é considerada *CPU-bound* e *memory-intensive* de carga moderada. No entanto, à medida que a ordem das matrizes é incrementada, a carga passa a ser considerada, além de *CPU-bound*, como *memory-intensive* de carga elevada.
3. Processamento de envelopes SOAP: Com o objetivo de aferir, separadamente, o impacto da virtualização no tempo de execução das aplicações e no tempo de codificação/decodificação de mensagens SOAP, este *Web Service* implementa duas operações que representam as tarefas de *marshalling* e *unmarshalling*, enviando e recebendo, respectivamente, envelopes SOAP cujos tamanhos variam entre $2MB$ e $10MB$.
   Adicionalmente, com o objetivo de avaliar o impacto da quantidade de elementos XML (*eXtensible Markup Language*) por mensagem SOAP, definiu-se que as mensagens possuiriam entre 10 e 1000 caracteres por elemento. Sendo assim, o tamanho do corpo do envelope SOAP permanece o mesmo, mas a quantidade de elementos XML a ser processada varia consideravelmente, uma vez que à medida que se aumenta a quantidade de caracteres por elemento, diminui-se a quantidade de elementos presentes na mensagem.

4. Influências da virtualização

O estudo do impacto da virtualização, ao executar aplicações baseadas em SOA em ambientes do tipo IaaS, foi dividido em três etapas:

1. Execução das aplicações: Esta etapa se refere à influência da virtualização e, principalmente, dos processadores virtualizados nos tempos de resposta das aplicações *CPU-bound* e *memory-intensive*;
2. Codificação/decodificação de envelopes SOAP: refere-se à influência da virtualização nos tempos de resposta ao processar mensagens SOAP. No entanto, essa etapa exclui a influência da virtualização das *interfaces* físicas

de rede. Essa estratégia permite separar a sobrecarga gerada nas bibliotecas de codificação/decodificação daquela gerada pela virtualização dos dispositivos físicos de rede, avaliada na terceira etapa;

3. Transmissão de dados: Esta etapa complementa a segunda ao analisar a influência da virtualização das *interfaces* físicas de rede.

Adicionalmente, como medida de comparação, utilizou-se a perda de desempenho obtida através da Equação 1, onde tr_v é o tempo médio de resposta obtido por uma plataforma virtualizada e tr_r é o tempo médio de resposta da plataforma de referência.

$$PD = \frac{tr_v}{tr_r} \tag{1}$$

4.1. Execução das aplicações

Conforme descrito anteriormente, esta primeira etapa possui como objetivo aferir a perda de desempenho, causada pela virtualização, nos tempos de resposta das aplicações baseadas em SOA. Desta forma, as submissões realizadas pelo *software* consumidor foram enviadas pela *interface loopback*, uma vez que essa *interface* é implementada totalmente em *software*, não realizando comunicação alguma com a *interface* física em qualquer uma das três plataformas avaliadas [Tanenbaum 2003].

Sendo assim, nas Figuras 1.a e 1.b é possível analisar, respectivamente, as perdas de desempenho sofridas pelas aplicações virtualizadas ao executarem as operações de ordenação de vetores, através do método Quick Sort, e multiplicação de matrizes.

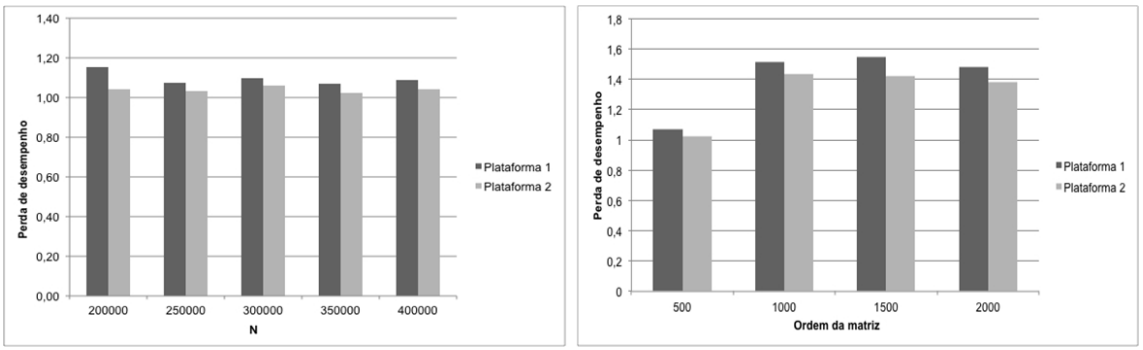

Figura 1. Perdas de desempenho ao executar aplicações *CPU-bound* e *memory-intensive* : a) Ordenação de vetores b) Multiplicação de matrizes.

Ao observar a Figura 1.a, é possível perceber que para aplicações *CPU-bound* e *memory-intensive* de carga baixa, as plataformas virtualizadas apresentam perdas de desempenho estáveis e de pequeno impacto, variando entre 7% e 10% na Plataforma 1 e entre 4% e 6% na Plataforma 2. Todavia, quando $N = 200.000$, as aplicações, executadas na Plataforma 1, apresentaram perdas de desempenho de, aproximadamente, 15%. Esse fato se deve à ineficiência do gerenciador de máquinas virtuais, da plataforma em questão, ao realizar a operação de *unmarshalling* de pacotes SOAP de tamanho reduzido, como será apresentado na Seção 4.2.

Esse comportamento estável se deve à evolução dos virtualizadores e, principalmente, ao curto espaço de tempo em que os processadores são utilizados. Nesse cenário,

as perdas de desempenho, sofridas pelas aplicações devido à virtualização, são minimizadas pelos servidores de aplicações que precisam alocar recursos para as execuções das operações solicitadas, o que, proporcionalmente, consome mais tempo que aquele dedicado à execução.

De maneira análoga, esse comportamento pode ser observado na Figura 1.b, em que para matrizes de pequena ordem, abaixo de 1000 elementos, as perdas de desempenho sofridas pelas aplicações, em relação à Plataforma de referência, são relativamente pequenas, sendo registrado, aproximadamente, 7% para a Plataforma 1 e 2% para a Plataforma 2.

Todavia, essa perda de desempenho reduzida deixa de ser observada à medida que o tempo de processamento aumenta e, consequentemente, a quantidade de memória manipulada. Nessa fase, o tempo de alocação de recursos pelos servidores de aplicações deixa de ser significativo e a influência da plataforma de virtualização se torna evidente, como pode ser observado na Figura 1.b quando a ordem das matrizes é superior a 500 e as aplicações deixam de ser classificadas, em relação ao consumo de memória, como *memory-intensive* de carga baixa e se tornam de carga moderada e elevada.

Apesar da elevação no índice de perda de desempenho da plataforma, a mesma se estabiliza após uma determinada ordem das matrizes, neste caso, quando a ordem das matrizes é superior a 1000, inclusive. No gráfico da Figura 1.b, essa perda oscilou, na Plataforma 1, entre 48% e 55% quando a ordem das matrizes era de 2000 e 1500, respectivamente. Já na Plataforma 2, a maior perda de desempenho registrada, após a estabilização, foi de 43% e a menor de 38%.

A partir dos dados anteriormente apresentados, é possível observar que, para as cargas de trabalho utilizadas, a Plataforma 1 apresentou perdas de desempenho superiores entre 5% e 12% em relação à Plataforma 2. O que indica que a arquitetura dos gerenciadores de máquinas virtuais, bem como sua implementação, possuem impactos diferentes nas soluções SOA avaliadas e que tal fato deve ser observado no planejamento de capacidade, assim como as fases estáveis de perda de desempenho.

4.2. Codificação/decodificação de envelopes SOAP

A análise do impacto da virtualização, no processamento de mensagens SOAP, é baseada nas perdas de desempenho sofridas pelas aplicações ao realizarem as tarefas de *marshalling* e *unmarshalling* das mensagens. De maneira análoga ao realizado na Seção 4.1, as submissões de consumo, efetuadas pelo *software* cliente, nesta segunda etapa de avaliação, foram realizadas pela *interface loopback*, permitindo que os resultados aferidos reflitam apenas as perdas de desempenho sofridas pelas bibliotecas relacionadas ao processamento de mensagens SOAP.

Desta forma, na Figura 2.a é possível observar as perdas de desempenho sofridas pelas aplicações, durante as operações de *unmarshalling*, onde cada envelope SOAP é composto por elementos XML de 10 caracteres. Complementarmente, na Figura 2.b, encontram-se as aferições obtidas nas operações de *marshalling*.

Como é possível observar na Figura 2.a, ambas as plataformas virtualizadas apresentaram perdas de desempenho em relação à plataforma de referência. A maior perda registrada para as Plataformas 1 e 2 foram de, aproximadamente, 21% e 15%, respectiva-

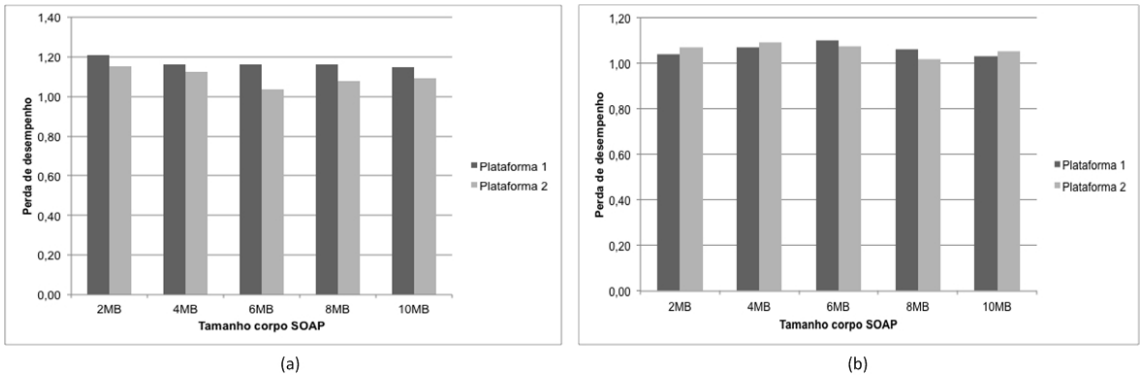

Figura 2. Perdas de desempenho sofridas pelas aplicações em plataformas virtualizadas ao processar mensagens SOAP com 10 caracteres por elemento XML: a) *Unmarshalling* b) *Marshalling*.

mente, ambas quando $S = 2MB$, onde S corresponde ao tamanho da mensagem SOAP processada.

Sendo assim, percebe-se que a maior perda de desempenho ocorreu com o menor valor de S. Esse fato indica que a perda de desempenho sofrida pelo servidor de aplicação, nas tarefas de alocação e manipulação inicial de objetos, é muito elevada devido à virtualização da plataforma. Pois, à medida que o valor de S aumenta consideravelmente, a perda de desempenho diminui, no entanto, ainda existe uma forte influência da virtualização na tarefa de *unmarshalling* dos pacotes SOAP.

Adicionalmente, é possível perceber que à medida que o tamanho da mensagem SOAP aumenta, as perdas de desempenho sofridas pelas aplicações nas plataformas virtualizadas tendem a estabilizar. Tal fato pode ser observado na Plataforma 1 com $S \geq 4MB$ e na Plataforma 2 com $S \geq 6MB$, cujas perdas de desempenho se estabilizaram em, aproximadamente, 16% e 8%, respectivamente.

Já na Figura 2.b, é possível observar que as perdas de desempenho sofridas pelas plataformas 1 e 2, ao executarem *marshalling* dos envelopes, são, significativamente, menores que as obtidas na tarefa de *unmarshalling*. Tal fato pode ser explicado pela maior complexidade na tarefa de *unmarshalling*, o que exige um maior tempo de processamento, uma vez que é necessário realizar o *parser* dos elementos XML, transformando-os em objetos alocados na memória DRAM, elevando as perdas de desempenho.

Durante as operações de *marshalling*, a Plataforma 1 obteve a menor perda de desempenho quando $S = 10MB$ e a maior quando $S = 6MB$, respectivamente, 3% e 10%. De maneira similar, a Plataforma 2 apresentou 2% e 9% como a menor e maior perda, quando $S = 8MB$ e $4MB$, respectivamente. Tal fato indica que, em média, as plataformas sofreram perdas de desempenho com diferenças entre 1% e 3%

No entanto, conforme pode ser observado na Figura 3, à medida que a quantidade de caracteres por elemento XML é incrementada, as diferenças, em porcentagem nas operações de *marshalling*, entre as perdas de desempenho registradas pelas Plataformas 1 e 2 também aumentam.

Ao analisar a Figura 3.a, é possível perceber que à medida que a quantidade de caracteres por elemento XML é ampliada para cem e mil, a Plataforma 1 permanece com

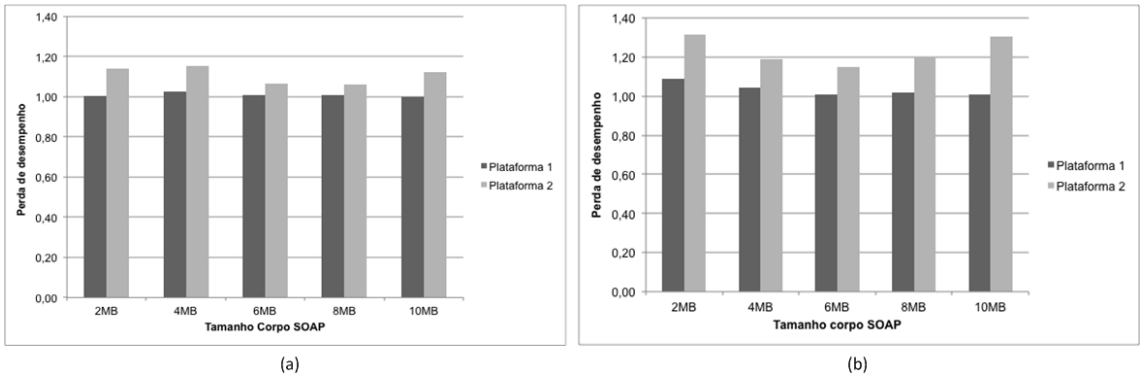

Figura 3. Perdas de desempenho sofridas pelas aplicações nas operações de *marshalling* com: a) 100 caracteres por elemento XML. b) 1000 caracteres por elemento XML.

o desempenho semelhante ao apresentado quando a quantidade de caracteres era menor. Já a Plataforma 2 apresenta uma variação relativamente elevada, sendo que, para mil caracteres, a maior perda registrada foi de 31% e a mínima de 15%, aproximadamente, quando $S = 2MB$ e $6MB$, respectivamente.

Tal fato indica que a Plataforma 2 é sensível ao tamanho da cadeia de caracteres e, consequentemente, a quantidade de elementos XML existentes em uma mensagem SOAP, diferentemente da Plataforma 1 que é sensível somente ao tamanho da mensagem.

Todavia, essa situação se inverte ao considerar a operação de *unmarshalling*, onde os resultados obtidos pela Plataforma 2 se apresentam estáveis quando a quantidade de caracteres por elemento SOAP é ampliado para cem e mil. Tal comportamento pode ser observado nas Figuras 4.a e 4.b, respectivamente.

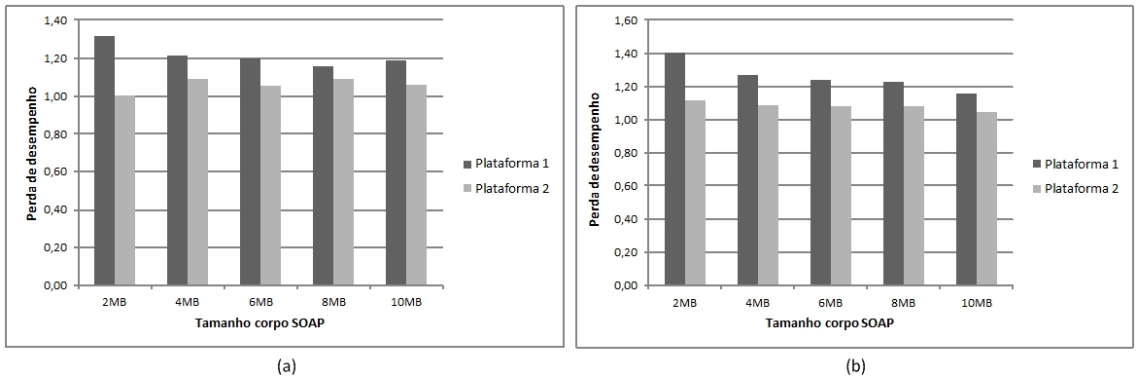

Figura 4. Perdas de desempenho sofridas pelas aplicações nas operações de *unmarshalling* com: a) 100 caracteres por elemento XML. b) 1000 caracteres por elemento XML.

Já a Plataforma 1 apresenta incremento na perda de desempenho quando a quantidade de caracteres por elemento XML é ampliada. Tal fato pode ser comprovado ao observar a Figura 4.a, em que a maior perda de desempenho sofrida foi de, aproximadamente, 32% quando $S = 2MB$ e para esse mesmo valor de S, a perda de desempenho em 4.b foi de, aproximadamente, 41%.

4.3. Transmissão de dados

Esta terceira etapa de avaliação foi realizada de maneira semelhante à etapa anterior, descrita na Seção 4.2. No entanto, as submissões realizadas pelo *software* cliente, para os *Web Services*, foram enviadas pela *interface* física de rede. Desta forma, o consumidor foi executado em uma máquina remota, conectada aos servidores das plataformas analisadas via uma rede de comunicação de dados de 100Mbps. Assim, as perdas de desempenho, analisadas nesta seção, foram calculadas pela razão entre os tempos de resposta obtidos pelas plataformas virtualizadas, ao receberem solicitações remotas, e os tempos de resposta obtidos na plataforma de referência, também ao receber requisições remotas.

A partir dessa informação, a figura 5.a apresenta as perdas de desempenho sofridas pelas aplicações distribuídas, executadas nas Plataformas 1 e 2, ao realizarem a operação de *unmarshalling* de mensagens SOAP, cujo corpo possui elementos XML de dez caracteres. Adicionalmente, a Figura 5.b apresenta as perdas de desempenho para as mesmas plataformas durante as operações de *marshalling*.

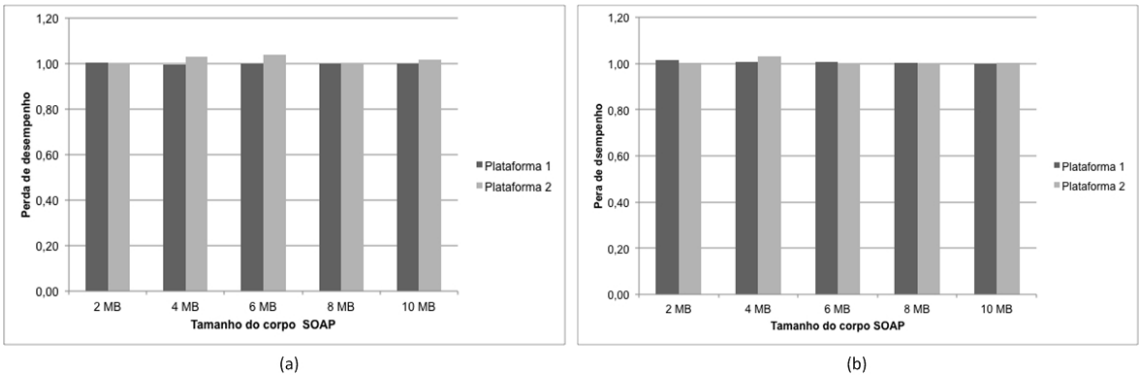

Figura 5. Perdas de desempenho sofridas pelas aplicações, em plataformas virtualizadas, ao processar mensagens SOAP, com dez caracteres por elemento XML, a partir de submissões remotas: a) *Unmarshalling*. b) *Marshalling*.

Ao observar os dados das Figuras 5.a e 5.b, é possível perceber que a virtualização das *interfaces* de rede não exerce grande impacto no desempenho das aplicações implantadas sobre servidores virtualizados, diferentemente do impacto registrado nas etapas anteriores, quando foram avaliados a influência no tempo de execução das aplicações e no tempo de codificação/decodificação de envelopes SOAP pelas bibliotecas dos servidores de aplicações.

Esse resultado indica que os *drivers* virtualizados, fornecidos com os *kits* de integração, juntamente com a técnica de *pass-through* de barramentos PCI (*Peripheral Component Interconnect*), permitem que o sistema operacional convidado acesse a *interface* de rede e os *buffers* com taxas de transferência semelhantes às obtidas pelos sistemas não virtualizados. De maneira similar, tal comportamento se repete ao ampliar a quantidade de caracteres por elemento XML para cem e mil.

5. Conclusões

Sendo a virtualização uma característica essencial em soluções de computação em nuvem do tipo IaaS, este trabalho analisou a sua influência na perda de desempenho das

aplicações baseadas em SOA, considerando, inclusive, o impacto gerado nas bibliotecas de manipulação de envelopes SOAP, durante as operações de *marshalling* e *unmarshalling*, e nas *interfaces* físicas de rede, mais especificamente nos módulos do núcleo do sistemas operacional GNU/Linux.

Para avaliar essa influência, foram desenvolvidos *Web Services* que, através de operações de ordenação de vetores, multiplicação de matrizes e troca de mensagens SOAP, de tamanhos previamente definidos, representaram aplicações *CPU-bound* e *memory-intensive* com carga de ocupação de memória variando de baixa à alta.

Com o objetivo de consumir tais *Web Services*, foi desenvolvido um *software* cliente, utilizado, também, para aferir os tempos de respostas das aplicações, cujas submissões de consumo foram realizadas através das *interfaces* de rede *loopback* e física. Como plataformas, optou-se pela instalação do sistema operacional GNU/Linux diretamente sobre o *hardware* e sobre as plataformas de virtualização total VMware® e Microsoft Hyper-v®, denominadas, respectivamente, Plataforma de referência, Plataforma 1 e Plataforma 2.

A partir das avaliações realizadas, foi possível concluir que para as aplicações *CPU-bound* e *memory-intensive* de carga baixa e com pequeno tempo de execução, as perdas de desempenho, ocasionadas pela virtualização nos tempos de processamento das aplicações, são reduzidas e constantes, uma vez que os tempos gastos pelos *contêineres Web* para estabelecer contextos são elevados se comparados aos tempos reais de processamento.

No entanto, à medida que o tempo de execução é incrementado, bem como a quantidade de memória DRAM consumida, a tecnologia de virtualização passa a influenciar consideravelmente nos resultados, impondo, assim, perdas de desempenho elevadas e constantes nos tempos de processamento das aplicações, sendo registrado perdas de até 54% na Plataforma 1 e 43% na Plataforma 2.

Ao considerar as perdas de desempenho pelas bibliotecas de codificação e decodificação de mensagens SOAP, foi possível concluir que, mesmo para pequenas mensagens, com tamanhos de aproximadamente 2 MB, a virtualização impacta negativamente no desempenho. Nas operações com 10 caracteres por elemento XML, as variações ficaram, nas operações de *unmarshalling*, entre 15% e 21% na Plataforma 1 e entre 4% e 15% na Plataforma 2. Nas operações de *marshalling*, as perdas de desempenho ficaram entre 3% e 10% na Plataforma 1 e 2% e 9% na Plataforma 2.

Essa influência também é observada à medida que a quantidade de caracteres por elementos XML é alterada. Nesse cenário, as plataformas virtualizadas apresentaram comportamentos distintos, já que a Plataforma 1 é sensível à variação da quantidade de caracteres nas operações de *unmarshalling*, diferentemente da Plataforma 2 que apresentou sensibilidade nas operações de *marshalling*

Finalmente, ao considerar a virtualização das *interfaces* de rede, foi possível concluir que o impacto ocasionado não é significativo, uma vez que o fator predominante é o tempo de tráfego e não o preenchimento dos *buffers* e acesso ao meio.

Sendo assim, faz-se necessário que os modelos de planejamento de capacidade para computação em nuvem do tipo IaaS, ao suportarem aplicações baseadas em SOA,

sejam modificados para considerar a separação dos tempos gastos na codificação e decodificação de envelopes SOAP, daquele consumido nas transmissões via *interface* física de rede, uma vez que estas *interfaces*, apesar de não imporem uma sobrecarga devido à virtualização, consomem tempos de transmissão proporcionais às quantidade de octetos trafegados no meio físico.

Faz-se necessário, ainda, que em tais modelos seja possível identificar para qual tipo de operação, *unmarshalling* ou *marshalling*, a plataforma é sensível, uma vez que comportamentos distintos foram detectados nesse contexto.

Tais modelos deverão, também, prever a separação dos processadores virtuais e físicos, a fim de representar as perdas de desempenho apresentadas na Seção 4.1. Adicionalmente, é necessário que seja introduzido índices, nesses modelos de processadores, a fim de representar a alternância de comportamento das perdas de desempenho destacadas nas Figuras 1.a e 1.b.

Finalmente, dando continuidade a este projeto, espera-se, em trabalhos futuros, expandir as análises de impacto realizadas, abordando sistemas de banco de dados e múltiplas instâncias de servidores de aplicação sobre uma mesma plataforma do tipo IaaS, o que permitirá, finalmente, modificar os modelos de planejamento de capacidades existentes, de forma que reflitam os resultados apresentados nessas pesquisas, permitindo que plataformas de computação em nuvem do tipo IaaS, na presença de cargas de trabalho formadas por aplicações baseadas em SOA, sejam realisticamente representadas.

Referências

Alves, L. C. D., Santana, M. J., Santana, R. H. C., and Oliveira, W. T. R. (2009). CMGS - obtendo eficiência no escalonamento de aplicações memory-intensive em clusters. In *XXXV Conferência Latino Americana de Informática*, Pelotas, RS, Brasil.

Baun, C., Kunze, M., Nimis, J., and Tai, S. (2011). *Cloud Computing: Web-Based Dynamic IT Services*. Springer Publishing Company, Incorporated, 1st edition.

Chi, Y., Moon, H. J., Hacigümüş, H., and Tatemura, J. (2011). SLA-tree: a framework for efficiently supporting SLA-based decisions in cloud computing. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 129–140, New York, NY, USA. ACM.

Estrella, J. C., Santana, R. C., Kuehne, B. T., Silva, J. C. F., and Paccanaro, L. C. (2008). Diretrizes para avaliação de desempenho de web services. *WPerformance*, pages 111–126.

Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D. (2011). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Trans. Parallel Distrib. Syst.*, 22(6):931–945.

Jain, R. K. (1991). *The Art of Computer Systems Performance Analysis*. Wiley, San Francisco, CA, USA, 1 edition.

Lu, L., Cherkasova, L., de Nitto Personè, V., Mi, N., and Smirni, E. (2010). AWAIT: Efficient overload management for busy multi-tier web services under bursty workloads. In *Proceedings of the 10th international conference on Web engineering*, ICWE'10, pages 81–97, Berlin, Heidelberg. Springer-Verlag.

Marzullo, F. P. (2009). *SOA na prática. Inovando seu negócio por meio de soluções orientadas a serviços*. Novatec, São Paulo, SP, Brasil.

Menasce, D. A. and Almeida, V. (2003). *Planejamento de Capacidade para Serviços na Web: Métricas, modelos e métodos*. Campus, Rio de Janeiro, RJ, Brasil.

Praphamontripong, U., Gokhale, S., Gokhale, A., and Gray, J. (2007). An analytical approach to performance analysis of an asynchronous web server. *Simulation*, 83(8):571–586.

Rao, J., Bu, X., Wang, K., and Xu, C.-Z. (2011). Self-adaptive provisioning of virtualized resources in cloud computing. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '11, pages 129–130, New York, NY, USA. ACM.

Roy, N., Dubey, A., Gokhale, A., and Dowdy, L. (2011). A capacity planning process for performance assurance of component-based distributed systems. In *Proceedings of the second joint WOSP/SIPEW international conference on Performance engineering*, ICPE '11, pages 259–270, New York, NY, USA. ACM.

Shimizu, S., Rangaswami, R., Duran-Limon, H. A., and Corona-Perez, M. (2009). Platform-independent modeling and prediction of application resource usage characteristics. *Journal of Systems and Software*, 82(12):2117 – 2127.

Tanenbaum, A. S. (2003). *Redes de Computadores*. Elsevier, Rio de Janeiro, trad. 4 ed. edition.

Teixeira, M., Massa, R., Oliveira, C., and Maciel, P. (2011). Planning service agreements in SOA-based systems through stochastic models. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 1576–1581, New York, NY, USA. ACM.

Ueno, K. and Tatsubori, M. (2006). Early capacity testing of an enterprise service bus. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '06, pages 709–716, Washington, DC, USA. IEEE Computer Society.

Xiao, L., Chen, S., and Zhang, X. (2002). Dynamic cluster resource allocations for jobs with known and unknown memory demands. *IEEE Trans. Parallel Distrib. Syst.*, 13(3):223–240.

Zhang, Q., Cherkasova, L., Mathews, G., Greene, W., and Smirni, E. (2007). A capacity planning framework for multi-tier enterprise services with real workloads. In *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pages 781–784.

Ziviani, N. (2011). *Projeto de algoritmos com implementações em Pascal e C*. Cengage Learning, 3st edition.