Uma Arquitetura para Auditoria de Nível de Serviço para Computação em Nuvem

Juliana Bachtold, Altair O. Santin, Maicon Stihler, Arlindo L. Marcon Jr, Eduardo Viegas

Pontifícia Universidade Católica do Paraná (PUCPR) – Escola Politécnica – Programa de Pós-Graduação em Informática (PPGIa) – Curitiba – PR – Brasil

{jbachtold, santin, sthiler, almjr, eduardo.veigas}@ppgia.pucpr.br

Abstract. *This paper's purpose is a multiparty Service Level Agreement (SLA) for cloud computing auditing. Audits are performed on the IaaS provider, IaaS contractor (SaaS provider) and SaaS client. The objective is to audit the cloud environment problems, internally and externally, in an unquestionable way by the parties. The proposal uses inspectors (auditing collector agents) and an independent auditor (third party), capable of identifying SLA deviations through information collected in the parties' environments. The results show that it is possible to audit and diagnose problems in the cloud by combining information from the parties with the independent auditing, in addition to avoiding conflicts of interest of the inspectors.*

Resumo: *Este artigo apresenta uma arquitetura para auditoria multipartes de acordo de serviço (SLA) em computação em nuvem. São auditados o provedor de IaaS, contratante de IaaS (provedor de SaaS) e cliente de SaaS. O objetivo é auditar problemas internos e externos ao ambiente de nuvem de modo incontestável pelas partes. A proposta emprega inspetores (agentes de coleta para auditoria) e auditores independentes (terceiros), capazes de identificar desvios de SLA através de informações coletadas nos ambientes das partes. Os resultados mostram que é possível auditar e diagnosticar problemas na nuvem combinando informações das partes, com a autoria independente, inclusive evitando-se conflito de interesse dos inspetores.*

1. Introdução

A utilização da computação em nuvem cresceu intensamente nos últimos anos, devido a sua adaptação às demandas dos negócios e ao encapsulamento da complexidade da tecnologia adjacente.

Basicamente, a nuvem computacional é formada por provedores que fornecem infraestrutura (*Infrastructure as a Service*, *IaaS*) para contratantes de *IaaS,* que desenvolvem aplicações (*Software as a Service*, *SaaS*) consumidas por seus clientes (usuário final ou consumidor de *SaaS*). Alguns provedores ofertam diretamente o ambiente para desenvolvimento e execução da aplicação (*Plataform as a Service*, *PaaS)* para que contratantes de *PaaS*, da mesma forma, desenvolvam aplicações para ofertar diretamente ao cliente (consumidor de SaaS).

As características que fazem o modelo de computação em nuvem ser atrativo para o contratante do serviço também restringem as ações de administração do contratante à interface de gerenciamento fornecida pelo provedor de *IaaS*.

Ao transferir seus sistemas para a nuvem computacional as organizações

Este trabalho é parcialmente financiado pelo Centro de Pesquisa e Desenvolvimento em Tecnologias Digitais para Informação e Comunicação (CTIC/MCTI), contrato 1313 e CNPq, processos: 310319/2009-9 e 478285/2011-6.

dividem com terceiros a responsabilidade pela tutela das informações e a execução de operações críticas aos seus negócios.

O estabelecimento de um Acordo de Nível de Serviço (*Service Level Agreement*, SLA) é uma forma de garantir que o serviço contratado venha a atender às necessidades da organização contratante. O SLA é um documento que contém os níveis de serviço esperados pelo contratante e garantidos pelo provedor, implementados como níveis de qualidade de serviço (*Quality of Service*, QoS) ou outros indicadores (métricas) definidos pelas partes.

Entretanto, o estabelecimento do SLA é apenas um dos componentes de governança para *cloud computing*. Considera-se que governança de *cloud computing* inclua, além do estabelecimento do SLA, a definição de políticas de uso e a implantação de rotinas de auditoria (Guo et. al., 2010).

A implantação de auditoria de SLA deve ser capaz de monitorar as ações de contratante e provedor dentro da nuvem, além de identificar responsabilidades externas sobre a degradação de serviço para qualquer uma das partes.

Em nuvem computacional o comportamento das partes (provedor, contratante e cliente), é responsável pelo desempenho do serviço computacional. Desta forma, a monitoração de SLA neste ambiente deve ser capaz de capturar informações sobre cada uma das partes para permitir a auditoria efetiva. Além de considerar o comportamento dos participantes, a auditoria efetiva deve ser capaz de detectar que fatores externos à nuvem alteraram a experiência (percepção) do cliente com relação ao serviço provido, como um problema com a conexão de rede entre as partes, por exemplo.

Se ocorrerem experiências negativas do cliente durante o consumo de serviços, deve ser possível para a auditoria identificar se o problema reside na aplicação do contratante, na conexão com o ambiente de nuvem computacional ou no provedor de infraestrutura, de maneira inquestionável para todas as partes. Ou seja, a auditoria multipartes necessita de pontos de monitoração com coleta continua de métricas sobre os serviços e recursos, considerados em cada uma das partes.

Entretanto, em *cloud computing,* o ambiente computacional das partes é segregado. Por este motivo, uma das partes não tem acesso às informações de auditoria das demais. Para contornar esta limitação e também para minimizar conflitos de interesse entre as partes, agentes de auditoria (inspetores) devem pertencer a uma entidade externa à nuvem para que esta tenha visibilidade sobre todas as partes. Os inspetores coletam e enviar as informações para um ambiente externo à nuvem. Então, um auditor independente e externo pode consolidar tais informações para identificar desvio de SLA e reportá-los à governança de TI do contratante. Este é o objetivo perseguido nesta proposta.

É importante frisar que o terceiro deve ser independente e confiável para todas as partes (provedor, contratante e cliente) e que as informações coletadas podem ser utilizadas no caso de questionamento quanto à aderência aos SLAs por qualquer e contra qualquer uma das partes.

Este trabalho apresenta uma arquitetura para implantação da auditoria multipartes em nível de IaaS e SaaS em nuvem computacional, com coleta de indicadores de forma anônima, implantados num protótipo que evidencia sua viabilidade.

O restante do trabalho está dividido da seguinte forma: a seção 2 apresenta a fundamentação; a seção 3 aborda os trabalhos relacionados; a seção 4 descrever a

arquitetura proposta; a seção 5 apresenta o protótipo desenvolvido e os testes realizados e a seção 6 desenvolve a conclusão do trabalho.

2. Fundamentação

Esta seção aborda os tipos de serviços em ambiente de nuvem computacional, conceitos aplicáveis a acordos em nível de serviço e fundamentos sobre virtualização.

2.1. Tipos de serviço em nuvem computacional

Os serviços de computação em nuvem podem ser disponibilizados de diferentes formas. O NIST define três modelos de arquitetura de nuvem de acordo com os recursos que são entregues para o cliente (Mell et. al., 2011):

- Software em Nuvem como um Serviço (*SaaS*). O serviço oferecido ao cliente consiste em aplicações disponibilizadas pelo provedor de *SaaS* executando em uma infraestrutura de nuvem. É o serviço que o cliente consome, e geralmente consiste numa aplicação.
- Plataforma em Nuvem como um Serviço (*PaaS*). Conjunto de ferramentas, ambiente de desenvolvimento e produção oferecida ao contratante para facilitar o desenvolvimento de sua aplicação.
- Infraestrutura em Nuvem como um Serviço (*IaaS*). São recursos computacionais (infraestrutura física) oferecidos ao contratante. Recurso computacional, em geral, consiste numa máquina virtual que oferece armazenamento, processamento e rede para que o contratante possa instalar e executar os softwares, desde o sistema operacional até as suas aplicações.

2.2. Acordos em nível de serviço

Um SLA pode especificar as necessidades e níveis a serem obtidos no fornecimento do serviço com alto nível de abstração. Os serviços são avaliados através de métricas de desempenho, conhecidas como Indicadores de Nível de Serviço (*Service Level Indicators*, SLIs). A integração da métrica de serviço (SLI) com o seu (valor) objetivo é denominado Objetivo de Nível de Serviço (*Service Level Objective*, SLO). "Disponibilidade" e "tempo médio de resposta do serviço" são exemplos de SLIs utilizados em um SLA enquanto que valores como 95% de disponibilidade ou 1,3 segundos de tempo de resposta, respectivamente, são exemplos de valores para SLOs (Sauvé et. al., 2005).

Para uma aplicação atender a um SLA, os recursos de infraestrutura que a suportam, por exemplo: processador, banda de rede e memória, devem ser adequadamente dimensionados. Cada recurso possui seus atributos como utilização do processador, largura de banda e memória disponível. As medições destes atributos são as métricas de recurso. São exemplos de métricas de recurso: percentual de memória disponível, percentual de consumo de rede, percentual de utilização de CPU etc.

Um conjunto de métricas de recurso reunidas para especificar requisitos de um serviço é denominado parâmetros de Qualidade de Serviço (*Quality of Service*, QoS). Métricas de recurso também são utilizadas para compor SLIs, quando estes SLIs compõem SLAs que regem o fornecimento de *IaaS*.

2.3. Virtualização

Virtualização de servidores implica em abstrair os recursos físicos de um servidor, de modo a parecerem diferentes logicamente do que são fisicamente, em geral

representando uma fração do hardware físico. Virtualização é utilizada por todas as nuvens computacionais e é considerada uma de suas características – característica esta não restrita apenas às nuvens que ofertam IaaS.

Num ambiente virtualizado a camada de software abstrai o hardware físico e instancia várias máquinas virtuais, cada uma com seus processos, sistema operacional e aplicações. Esta camada de abstração é controlada pelo *hypervisor* (hipervisor). É o *hypervisor* que disponibiliza a abstração de recursos de hardware para os sistemas operacionais.

Um sistema operacional executado numa instância de máquina virtual é chamado de SO hóspede (*guest*). As máquinas virtuais instanciadas no mesmo hardware compartilham os "mesmos recursos físicos". Esse compartilhamento de recursos pode significar um melhor aproveitamento do hardware, uma vez que os servidores oscilam quanto ao consumo de recursos de hardware, apresentando picos e ociosidade de uso em diferentes momentos.

3. Trabalhos relacionados

O trabalho de Skene (Skene et.al., 2010) modela o consumo de serviços na Internet (não especificamente em nuvem computacional), apresentando a influência dos fatores externos para contratante e provedor no consumo dos serviços. Problemas de conexão do cliente podem influenciar a percepção de tempo de resposta no lado servidor, por exemplo. Para determinar o cumprimento do SLA, torna-se importante que as influências externas sejam filtradas, utilizando-se o exemplo do SLI "tempo de resposta", a influencia da latência da conexão entre cliente-servidor pode ser obtida através do registro de *timestamps* entre o envio e o recebimento de pacotes.

No trabalho de Chen (Chen, Y. et. al., 2011) é apresentada uma cadeia de confiança estabelecida na utilização da nuvem: contratantes podem ter SLA estabelecido com provedores de *SaaS*, que por sua vez possuem outro SLA com provedores de *IaaS* – o que nem sempre é visível para o cliente. Porém, o comportamento do contratante pode influenciar o nível de serviço do provedor *IaaS*, em determinadas situações, por exemplo, quando seus clientes demandarem muito a rede. A interdependência entre as partes introduzida pela computação em nuvem requer auditabilidade mútua – provedores e contratantes devem ser auditados nos níveis de serviço que provem. Entretanto, em seu trabalho Chen não relaciona a terceira entidade auditável, o cliente. Ao considerá-lo estamos estendendo a proposta original de auditoria mútua para auditoria multipartes.

A inclusão do cliente como entidade auditável baseia-se no trabalho de Morsel (Morsel, 2001), onde são apresentados exemplos de métricas para obtenção da Qualidade da Experiência (*Quality of Experience*, QoE), como a utilização do parâmetro "tempo de resposta" a ser medido na estação do usuário. Segundo os autores, as métricas de Qualidade de Serviço (*Quality of Service*, QoS) não traduzem a percepção do usuário quanto ao nível de serviço consumido.

Na proposta de Barbosa (Barbosa et. al., 2006) são discutidas arquiteturas de auditoria de SLA em *grid computing*. As alternativas são avaliadas quanto à eficiência de cada arquitetura em diferentes cenários. As arquiteturas apresentadas no artigo são baseadas na existência de uma entidade terceira, que seja considerada confiável por ambos, para mediar a relação cliente e servidor. As arquiteturas apresentadas no trabalho não consideram a existência de ambientes computacionais segregados para provedor e contratante, uma característica do ambiente de nuvem computacional.

Haeberlen (Haeberlen, 2010) considera que para tornar a nuvem computacional *accountable* é necessário que seja definida uma taxa de amostragem adequada, todas as entidades monitoradas devem ter relógios sincronizados para captura e comparação adequada de *timestamps*. Os logs devem ser armazenados em outro ambiente, diferente daquele onde foram feitas as coletas. Além disto, os logs serem invioláveis, e.g. *write-once.* Apesar de definir os requisitos de auditabilidade, Haeberlen não define uma arquitetura para auditoria em nuvem.

Emeakaroha (Emeakaroha et. al., 2010) apresenta uma nuvem que utiliza a auditoria de SLA integrada ao funcionamento de um *broker* (negociador), utilizando nas medições da infraestrutura para redimensionar a nuvem computacional. Na arquitetura apresentada no artigo, não existe a figura do inspetor para o ambiente do cliente, fornecendo apenas informações sobre consumo de recursos de infraestrutura na visão do provedor.

Nenhum dos trabalhos relacionados oferece uma arquitetura para auditoria em nuvem que implemente a auditabilidade multipartes para computação em nuvem e para tal obtenha informações de cliente, contratante e provedor. Além disto, nenhuma proposta mostra-se capaz de identificar as causas de desvio de SLA dentro e fora da nuvem computacional ou preocupa-se com o possível conflito de interesses do auditor.

4. Proposta

A seguir são apresentadas as premissas e a arquitetura da proposta com seus principais componentes.

4.1. Premissas para auditoria de SLA

A arquitetura proposta para auditoria de SLA em computação em nuvem considerou os seguintes aspectos:

- As informações de auditoria devem ser obtidas por um terceiro; entidade independente em relação ao cliente, contratante e provedor; porém confiável a todos;
- Os inspetores coletam os dados do cliente, contratante e provedor continuamente e, os fornecem ao auditor que cria o SLI e o compara ao respectivo SLO;
- As informações obtidas no ambiente do cliente e contratante devem auxiliar na identificação de fatores externos, que influenciam negativamente a percepção da nuvem;
- As informações obtidas no ambiente do contratante e do cliente podem ser confrontadas com as informações do provedor para identificar não conformidade de SLA causado por fatores internos à nuvem (sejam de responsabilidade do contratante ou do provedor);
- O inspetor não deve ser capaz de associar o consumo de serviços ao cliente ou contratante dentro da nuvem, mas o rastreio das ações de clientes e contratantes dentro da nuvem deve ser possível para o pessoal de governança de TI do contratante;
- Através do comportamento de contratante e provedor é possível responsabilizar as partes por suas ações dentro do ambiente de nuvem;
- A coleta de informações do contratante permite que o nível de serviço seja auditado, considerando-se também a experiência de consumo do cliente.

4.2. Visão geral do cenário da proposta

O mecanismo de auditoria baseia-se em três componentes: inspetores, auditores e camada de governança. Os inspetores são posicionados dentro da nuvem para realização da coleta, sendo inseridos na infraestrutura do provedor (camada *IaaS*) e na camada de aplicação do contratante (camada *SaaS*). Cada inspetor é responsável por coletar dados que irão gerar um SLI ou parte desse.

Os auditores permanecem lógica e fisicamente fora da nuvem auditada e recebem os dados dos inspetores para calcular os SLIs. Pode haver um auditor para cada SLI ou família de SLIs correlacionados (i.e. calculados a partir dos mesmos dados).

Após calcular o SLI, o auditor confronta os resultados com o SLO a fim de identificar ocorrências de não-conformidade com o SLA.

A camada de governança obtém dos auditores os valores calculados para os SLIs e as sinalizações para os SLIs que apresentaram desvio. Por conhecer todos os SLIs calculados para cada cliente, a camada de governança, em caso de desvio de SLA, torna-se capaz de isolar causas de problemas a partir das medições obtidas. Na seção 4.3 será abordado a interpretação das medições.

Na Figura 1 é mostrado um exemplo de auditoria para um tipo de SLI para *IaaS* e um SLI medido em *SaaS*, mas é possível monitorar mais SLIs através da utilização em paralelo de outros inspetores e auditores (representado pela linha pontilhada na Figura 1). Por fim, uma camada de governança pode gerar dados estatísticos, a partir dos dados coletados, e avaliar o serviço de determinado provedor ou as causas do desvio de SLA para determinado cliente – informações que não são geradas pelos os auditores porque hierarquizamos a informação contextualizada para evitar o conflito de interesses.

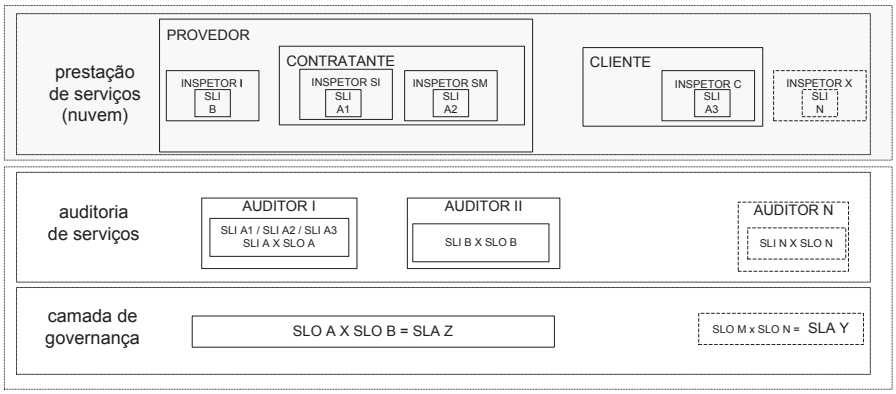

Figura 1. Visão geral da arquitetura da proposta

Na proposta um inspetor só lê o valor de um indicador relacionado a um pseudônimo. O mesmo acontece com o auditor que cria SLIs do mesmo tipo para o mesmo pseudônimo e os confronta com atributos de SLO para este pseudônimo, disponibilizando-os a camada de governança de modo consolidado. Porém, nem o inspetor, nem o auditor, conseguem relacionar o pseudônimo a uma identidade do mundo real – só a camada de governança tem esta habilidade. Assumimos que a homologação dos códigos que coletam os indicadores deve ser feita antes que os mesmos sejam colocados nos pontos de coleta do inspetor. Estes códigos podem ser inspecionados a qualquer momento pelo auditor através de geração de *hash code* criptográfico. Neste caso, é sugerido que o código do coletor seja desenvolvido usando a técnica de reentrância (Wloka et al., 2009).

Uma vez que os inspetores consultam as camadas de infraestrutura e aplicação, é necessário definir as premissas iniciais de ambiente para obtenção de métricas de auditoria. A infraestrutura da arquitetura presume a existência de virtualização. Para a camada de aplicação a arquitetura deve estar preparada para funcionar com aplicações disponibilizadas em servidores web – nos papéis de servidor de interface (para o cliente) e motor de aplicação.

Na camada de infraestrutura o inspetor (Inspetor I) obtém informações de virtualização sobre recursos disponibilizados e consumidos, controlados pelo *hypervisor*, i.e., o Inspetor I extrai as medições de SLIs da infraestrutura.

Tabela I – Descrição dos tempos coletados pelos inspetores

Inspetor	Onde	ID Registro	Função interna a operação
Inspetor C	Cliente (estação do usuário)	Tempo 1 (T1)	Registra o *timestamp* do envio de página web pelo cliente, i.e, no momento da requisição de operação a aplicação.
Inspetor SI	Servidor de *interface*	Tempo 2 (T2)	Registra o *timestamp* do recebimento do pedido de operação pelo servidor.
Inspetor SI	Servidor de *interface*	Tempo 3 (T3)	Registra o *timestamp* imediatamente anterior ao envio da solicitação para o servidor de *motor* da aplicação.
Inspetor SA	Servidor *de motor* da aplicação	Tempo 4 (T4)	Registra o *timestamp* imediatamente posterior ao recebimento da requisição de operação.
Inspetor SA	Servidor de *motor* da aplicação	Tempo 5 (T5)	Registra o *timestamp* imediatamente anterior ao envio do resultado da operação.
Inspetor SI	Servidor de *interface*	Tempo 6 (T6)	Registra o *timestamp* imediatamente posterior ao recebimento da resposta do motor da aplicação.
Inspetor SI	Servidor de *interface*	Tempo 7 (T7)	Registra o *timestamp* imediatamente anterior ao envio do resultado para o cliente.
Inspetor C	Cliente (estação do usuário)	Tempo 8 (T8)	Registra o *timestamp* imediatamente posterior ao recebimento do resultado da operação.

Na camada de aplicação os inspetores devem ser instalados no servidor de interface, no cliente e no servidor do motor de aplicação. As informações coletadas pelos inspetores são base para os auditores confeccionarem o SLI "tempo de resposta para o cliente (usuário) *por* operação".

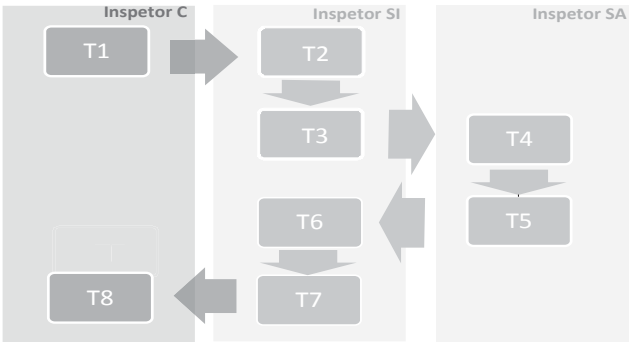

Figura 2 – Coleta de *timestamps* pelos inspetores

Os *timestamps* gerados por cada inspetor de software estão descritos na Tabela I e são apresentados abaixo:

- Um inspetor de aplicação é executado no cliente (Inspetor C). O objetivo do inspetor C é coletar *timestamps* das transações que ocorrem dentro da estação do cliente (T1 e T8, Figura 2).

- Os inspetores de aplicação que são executados no ambiente de aplicação do contratante estão assim alocados:
    a) O Inspetor SI é executado no servidor de interface e registra *timestamps* na interação com o cliente e com o servidor de motor da aplicação (T3, T6 e T7, Figura 2).
    b) O Inspetor SA é executado no servidor de motor de aplicação (Inspetor SA) e registra um *timestamp* (T4, Figura 2) ao receber a requisição para execução de operação e outro ao devolver a resposta para o solicitante (T5, Figura 2).

4.3. O papel do auditor e do módulo de governança

Além do SLI principal "tempo de resposta para o cliente por operação" (*Tru),* os auditores geram outros indicadores intermediários à partir dos *timestamps* coletados (Tabela II). A geração destes indicadores é necessária para localização das causas de ocorrências de desvios de SLA, interpretadas da seguinte maneira:

- Problemas de desempenho externos à nuvem, nos serviços de conexão do provedor e/ou do cliente à Internet. Identificam-se altas taxas de latência nas conexões do cliente, do servidor de interface ou do servidor de aplicação, através de SLIs calculados a partir de indicadores no lado do cliente ou servidor de interface. A identificação ocorre quando observados valores superiores aos limites definidos para T8 - T7 (Trd2) e T2 – T1 (Trd1), respectivamente.
- Problemas de desempenho dentro da nuvem são evidenciados por indicadores calculados com a diferença entre *timestamps* coletados dentro do mesmo servidor, caracterizando maior tempo para conclusão de determinada operação, nos seguintes casos:
    a) no servidor de interface, evidenciado por T3 – T2 (Tsi).
    b) no servidor de motor da aplicação, evidenciado por T5 – T4 (Tsa).

Tabela II – SLIs gerados pelo auditor

SLI	Operação representada pelo indicador	Cálculo	Análise do módulo de governança
Tru	Identifica o tempo de resposta para o usuário final (execução completa de uma operação).	Tru = T8– T1	Caso o SLO seja extrapolado, sugere a análise de mais indicadores, para identificar a operação gargalo.
Trd1	O indicador obtém o tempo para o envio de conteúdo do cliente para o servidor de *front-end* (aplicação).	Trd1= T2– T1	Tempo elevado entre o envio e o recebimento do conteúdo indica gargalos na comunicação em rede entre o cliente e o servidor.
Tsi	O indicador obtém o tempo de processamento do servidor de interface para executar a operação de recebimento e empacotamento de conteúdo.	Tsi= T3 - T2	Tempo elevado para receber e transmitir o conteúdo indica gargalo no servidor de interface, (não identificando se há problema na disponibilização do hardware contratado ou sobrecarga de recursos). No caso de medições mais altas são necessárias informações adicionais do Inspetor I.
Tsa	O indicador obtém o tempo de processamento do servidor de motor de aplicação para executar o envio de conteúdo.	Tsa= T5– T4	Tempo elevado para receber e transmitir o conteúdo indica gargalo no servidor de motor de aplicação, (não identificando se há problema na disponibilização do hardware contratado ou sobrecarga de recursos). No caso de medições mais altas são necessárias informações adicionais do Inspetor I.
Trd2	O indicador obtém o tempo para o envio do conteúdo do servidor de interface para o cliente.	Trd2= T8– T7	Tempo elevado entre o envio e o recebimento do conteúdo indica gargalos na comunicação em rede entre o servidor de interface e o cliente.

A identificação de problemas dentro da nuvem (evidenciada pelos indicadores Tsi e Tsa), em si, não caracteriza não-conformidade do provedor de acesso ou problemas no ambiente do cliente. Estas medições precisam ser combinadas com as informações obtidas pelo Inspetor I. Para as operações que levaram mais tempo para serem concluídas, o auditor busca evidências de maior consumo (ou até mesmo esgotamento) de recursos de hardware, como CPU e memória.

Anomalias em funções executadas dentro de uma operação podem ser reveladas pelas correlações entre *timestamps* obtidos pelos inspetores. A Tabela II sintetiza os indicadores e as respectivas análises que podem ser geradas pelo módulo de governança

4.4. Mapeamento entre contratante e cliente

O contratante do serviço de nuvem pode ser uma organização pública ou privada que oferece serviços desenvolvidos em nível *SaaS* a um conjunto de clientes; são os clientes que efetivamente realizam as operações nos serviços e aplicações oferecidas na camada *SaaS* da nuvem computacional. Assim, um provedor pode atender a vários contratantes (locatários) e este contratante pode, por sua vez, representar vários clientes.

Os *timestamps* obtidos pelos inspetores nas operações realizadas por clientes são transmitidos ao repositório do auditor. Naturalmente, cada cliente é identificado por um pseudônimo para evitar conflito de interesse de auditores e inspetores.

Um servidor de pseudônimos externo ao ambiente do mecanismo de monitoração e coleta (i.e., no domínio de governança) fornece os pseudônimos ao cliente. Ainda que o inspetor observe este pseudônimo a cada operação feita nos serviços da nuvem e o registre para fins de rastreamento, não deverá obter outras informações a ponto de identificar o cliente no mundo real.

O servidor de pseudônimos os renova com certa periodicidade para evitar o rastreamento de clientes que, por alguma razão – externa ao sistema de identificação, tenham seus pseudônimos reconhecidos. O registro da operação é armazenado associado ao pseudônimo e somente quando necessário poderá ser associado à verdadeira identidade do cliente. Somente com intervenções externas é fornecido o mapeamento entre pseudônimo e cliente, em geral por questões legais, que geralmente surgem quando há impasse quanto à conformidade de SLAs.

5. Protótipo

Para avaliar a viabilidade da proposta, a arquitetura para auditoria de SLA foi implementada em um protótipo e testada.

5.1. Desenvolvimento do Protótipo

O protótipo foi desenvolvido sobre uma aplicação Web, simples, baseada no envio de e-mails (mensagens e anexos). A aplicação está subdividida em camada de interface e motor de aplicação. A camada de interface é responsável por receber e enviar os formulários de interação (página web) com o cliente, tratando as respostas recebidas e encaminhando as solicitações de processamento ao motor de aplicação (um servidor SMTP – *Simple Mail Transfer Protocol*), que recebe o formulário de envio e seus anexos e realiza o envio das mensagens via SMTP.

Os servidores foram instanciados numa nuvem computacional e devem possuir configuração mínima de 01 núcleo virtual e 256 MB RAM. O servidor de interface se utiliza do Apache TomCat e acesso a um SGBD (Sistema de Gerenciamento de Banco de Dados) externo, nos testes adotou-se o MySQL. Todos os servidores precisam ter

seus relógios sincronizados a partir de um mesmo servidor NTP (*Network Time Protocol*).

As mensagens trocadas entre o servidor de interface e o servidor de motor de aplicação estão no formato JSON (formato de troca de mensagens alternativo ao XML – *Extensible Markup Language*). O serviço disponibilizado como motor de aplicação possui o estilo arquitetural REST (*REpresentational State Transfer*).

O Inspetor I conecta-se no *hypervisor* obtendo informações diretamente do *Xentrace*, que é um gerador de eventos disponível nativamente no virtualizador *XEN* (*bare-metal hypervisor*). As informações são enviadas para o auditor para registro da operação como descrito na seção 4.4. O Inspetor I, desenvolvido em 'C', salva as informações coletadas em arquivos texto para posterior consolidação no auditor.

Para fins de protótipo, o Inspetor I irá obter apenas o consumo de CPU por servidor para identificar esgotamento de recursos de processamento, entretanto, o Inspetor I é capaz de obter do *hypervisor* métricas de outros recursos de hardware. Como estas métricas não serão utilizadas nos testes a seguir, optamos por não mencioná-las.

Os inspetores da camada de aplicação, desenvolvidos em Java, foram nomeados de acordo com o ambiente onde são de executados, conforme apresentado na seção 4.2.

5.2. Definição dos cenários de testes

A escolha do ambiente de testes apenas em nuvem privada foi determinada pela impossibilidade de se utilizar nuvens públicas, uma vez que nestas o Inspetor I não teria acesso ao *hypervisor*. Desta forma, todos os cenários propostos exprimem variações de condições dentro de um mesmo ambiente computacional.

Tabela III – Planejamento dos cenários de testes da proposta

Objetivo do Cenário	Descrição	Implantação	Indicadores afetados
A – Definição de baseline, criação de indicadores de referência.	Recursos de processamento de ambos os servidores dedicados exclusivamente ao processamento da aplicação. Utiliza-se a banda de rede disponível para este cenário.	Execução de simulações no ambiente, sem implantação de restrição de banda ou adição de carga de processamento nas VMs dos servidores de interface e motor de aplicação.	Todos os indicadores são obtidos a fim de gerar as medições iniciais
B – Verificar a capacidade dos inspetores capturarem ocorrências de sobrecarga de processamento nos dois servidores e do auditor correlacionar as medições a gargalos causados pela infraestrutura.	Recursos de processamento de ambos os servidores comprometidos com carga de aplicações concorrentes. Utiliza-se a banda de rede disponível para este cenário.	Utilização de aplicação Java, com algoritmo de criptografia, chave de 168 bits e o algoritmo DES gerando consumo adicional de CPU, dentro das VMs dos servidores interface e motor. A execução desta aplicação ocorre em paralelo à execução da aplicação de interface e da aplicação principal (*core*).	Os indicadores que devem refletir alteração são Tsi e Tsa
C – Avaliar a eficiência do protótipo em identificar gargalos relacionados à conectividade.	Recursos de processamento de ambos os servidores dedicados exclusivamente ao processamento da aplicação. Restringe-se a banda de rede	Execução de simulações no ambiente, sem adição de carga de processamento nas VMs dos servidores interface e motor. Foi implantado a restrição de banda através do uso do CBQ (*Class Based Queue*), utilitário Linux para implantação de QoS. A velocidade medida entre o cliente e o servidor foi de 100 Kbps/s.	Os indicadores que devem refletir alteração são Trd1 e Trd2

Quando são executadas operações nos cenários B e C (Tabela III), obtêm-se resultados que desviam do SLA proposto (referência de SLA no protótipo é obtida em A). Como o

desvio de SLA em B e C é causado por intervenções em pontos distintos da nuvem, planejou-se medir o resultado através de indicadores relacionados a estes componentes.

A nuvem privada foi implantada na rede da PUCPR. Os servidores foram instanciados numa nuvem computacional *open source*, o Eucalyptus (Numi et. al., 2010) versão 2.0.3, utilizando para virtualização o XEN versão 4.1.2. O servidor de interface executou em sistema operacional Linux Ubuntu 10.04 e Apache TomCat versão 7.0.27, com 01 núcleo virtual e 256 MB RAM. O servidor de motor da aplicação também executou em sistema operacional Linux Ubuntu 10.04 com 01 núcleo virtual e 256 MB RAM.

O objetivo dos testes foi identificar a capacidade do protótipo para:

- Identificar interferências externas – não associadas ao desempenho da nuvem (no cliente);
- Identificar ocorrências de não-conformidade de SLA através das informações obtidas no ambiente do cliente, contratante e do provedor;
- Obter informações que permitissem identificar o responsável pela não-conformidade com o SLA, usando a análise dos SLIs.

5.3. Resultado dos testes e análises

Os resultados dos testes mostram a capacidade do auditor para identificar ocorrências de violação de SLA, através da formulação dos SLIs para cada função interna as operações executadas pelo cliente e sua comparação com os respectivos SLOs.

O módulo de governança pode identificar interferências externas ou internas à nuvem com sucesso e relacioná-las ao responsável pelo fato. Os indicadores extraídos em cada cenário são mostrados nas tabelas IV, V e VI. Para todos os casos o coeficiente de variabilidade foi inferior a 5%. Cada conjunto de testes foi repetido quarenta vezes por cenário e o tempo de uma operação é medido a partir do envio do formulário de mensagem pelo usuário final até o processamento da mensagem pelo servidor SMTP. Os indicadores gerados foram previamente apresentados na Tabela II.

No cenário A (Tabela III) o objetivo foi capturar os indicadores em cenário neutro (Tabela IV), sem intervenções internas ou externas à nuvem.

Tabela IV – Resultado dos testes para o cenário A

SLI	Tru	Trd1	Tsi	Tsa	Trd2
Média (em mSec)	4.354	641	427	1.130	519

Neste cenário (Tabela IV), os valores de referência foram estabelecidos para os SLIs medidos nos próximos cenários, compondo os SLOs para cada operação. Na simulação A os indicadores de consumo de CPU não indicaram esgotamento (ou consumo excessivo) de recursos, ficando o consumo médio de CPU em 12,6% para o servidor de interface e 7,5% para o servidor de motor de aplicação durante a execução de cada operação.

Tabela V - Resultado dos testes para o cenário B

SLI	Tru	Trd1	Tsi	Tsa	Trd2
Média (em mSec)	18.466	775	1.162	10.182	579

No cenário B (Tabela III), foram iniciadas outras aplicações para geração de carga de processamento adicional nos servidores de interface e motor de aplicação, durante os acessos a aplicação do experimento (Tabela V). Com a carga adicional,

esperou-se que fosse aumentado o tempo total das operações na aplicação e que os inspetores SI e SA fossem capazes de identificar quais as operações foram impactadas pela alteração no ambiente.

O auditor A, através das medições dos inspetores C, SI e SA identificou com sucesso o aumento nos tempos de funções (internas as operações) executadas no ambiente servidor, representadas pelos SLIs Tsi e Tsa. Comparados ao SLO estabelecido com as medições do Cenário A (Tabela IV), os indicadores desviaram-se em 270% e 900%, respectivamente. O maior tempo para execução das funções processadas nos servidores impactou em 420% o SLI Tru, que exprime o tempo total da operação.

Uma vez que as operações identificadas com a maior latência são realizadas dentro da nuvem – pois dependem exclusivamente de seu processamento pelo servidor, o módulo de governança requisita que o Auditor B (Figura 1) busque as medições do Inspetor I para obter o consumo de CPU dos servidores. Neste cenário foi identificado 86,5% de consumo de CPU no servidor de interface e 93,2% no servidor de motor de aplicação. Ficou evidenciado para o módulo de governança que a causa de desvio do SLO para esta operação foi causado pelo contratante por consumo intensivo de CPU.

No cenário C (Tabela III) a banda disponível para conexão do cliente ao servidor de interface foi reduzida, durante os acessos a aplicação do experimento (Tabela VI). Com esta alteração esperou-se que fosse aumentado o tempo total da operação e que os Inspetores C, SI e SA fossem capazes de identificar quais as funções que foram impactadas pela alteração no ambiente.

Tabela VI - Resultado dos testes para o cenário C

SLI	Tru	Trd1	Tsi	Tsa	Trd2
Média (em mSec)	15.557	12.009	189	1.185	654

Através dos dados dos inspetores C, SI e SA, o Auditor A identificou com sucesso o aumento nos tempos de operações que dependem da conexão de rede entre cliente e servidor de interface, representadas pelos SLIs Trd1 e Trd2. As medições obtidas neste cenário para estes indicadores desviaram-se em 1870% e 130% do SLO. O desvio foi expressivamente maior para o indicador Trd1 em relação ao Trd2 porque o arquivo (transmitido conjuntamente com o formulário de mensagem para o servidor de interface) utilizado nos experimentos foi de 3,5MB e a operação medida por Trd1 contempla todo o tempo da operação (incluindo a função de envio do formulário e de transmissão do arquivo). Já em Trd2, apenas um página de confirmação é enviado do servidor de interface para o cliente.

No cenário C o módulo de governança identificou que as operações com maior desvio do SLO são operações dependentes de conexão de rede e pode confirmar que a responsabilidade do descumprimento do SLA foi externa à nuvem.

Avaliando-se conjuntamente o resultado nos cenários B e C percebe-se que o módulo de governança pode identificar causas distintas para ocorrências de não conformidade com o SLA nas operações onde foi identificado maior tempo de resposta.

6. Conclusão

A definição de um SLA exprime a expectativa de clientes e contratantes e os compromissos dos provedores de serviços em nuvem. Entretanto, a qualidade dos serviços percebida pelo cliente não é afetada unicamente pelo provedor. Fatores

externos a nuvem e o próprio comportamento dos contratantes e clientes podem impactar em seu funcionamento.

A simples extração de informações de consumo de recursos infraestrutura por parte do provedor entrega uma visão parcial e não o isenta de conflitos de interesse sobre a entrega do serviço. As medições de SLIs extraídas pelo provedor podem não corresponder à forma como o usuário percebe e avalia o serviço. O usuário avalia o serviço através da sua experiência de uso, QoE, que é afetada pelo desempenho de uma cadeia de provedores, incluindo o próprio provedor da nuvem, e as operadoras que fornecem conexão do usuário à Internet.

A utilização de SLIs baseados em QoE podem auxiliar na identificação do componente onde há degradação de nível serviço e quem seja causador de descumprimento de SLA, mas, esta informação é visível parcialmente ao cliente e parcialmente ao contratante. Desta forma, informações, que eventualmente podem identificar gargalos em operações fora da nuvem, retirando a responsabilidade sobre as inconformidades de SLA do provedor, também não são visíveis a esse.

Através da geração dos indicadores para cada operação, torna-se possível combinar informações oriundas do cliente, contratante e provedor, obtendo-se uma única visão sobre a entrega de serviços e eventual causa sobre desvios de SLA.

O trabalho mostrou que para evitar conflitos de interesse entre cliente, contratante e provedor deve haver a participação de uma entidade neutra, confiável e independente, obtendo e relacionando as informações obtidas destes tornando todas as partes auditáveis, promovendo assim a auditabilidade multipartes. Para isso, a entidade de auditoria deve ter acesso às informações diretamente dos ambientes do provedor, do contratante e do cliente, consolidá-las em ambiente externo e livre de interferências.

A arquitetura de auditoria considerou que cliente, contratante e provedor estejam cientes de sua execução, uma vez que é necessária a adição de *plug-in* no lado do cliente e do contratante, e a disponibilização do código coletor no provedor, especificamente na camada de virtualização. Este foi o motivo que inviabiliza a execução de testes do protótipo em nuvem pública. Porém, entende-se que é interesse das partes auditarem-se entre si, quando isto acontecer recursos como os comentados neste trabalho deverão se tornar comuns e tais limitações não serão mais identificadas.

Os testes realizados em nuvem privada evidenciaram a capacidade do protótipo identificar o componente responsável pelo desvio, de acordo com a operação afetada, tornando possível atribuir responsabilidade sobre o descumprimento de SLA. Adicionalmente, a utilização dos auditores de infraestrutura e de aplicação, a arquitetura permite o desenvolvimento de novos inspetores e auditores, associados a outros SLIs.

Ainda que a tendência de utilização de computação em nuvem venha se intensificando no ambiente empresarial, a baixa confiança no nível de serviço obtido do provedor ainda é uma barreira de entrada para muitas empresas. Porém, a criação de serviços que sejam auditáveis em multipartes, somada a existência de entidades de auditoria de reconhecida boa reputação pode aumentar o nível de confiança dos futuros contratantes de computação em nuvem.

7. Referências

BARBOSA, AC; SAUVÉ, J.; CIRNE, W.; CARELLI, M. (2006). Evaluating architectures for independently auditing service level agreements, Elsevier FGCS, 22, 7, 721-731.

BUYYA, R.; YEO CS.; VENUGOPAL, S. (2008). Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, High Performance Computing and Communications. Proc. HPCC, pp. 25-27.

CHEN, Y. PAXSON, V., Katz R.H. (2011). New About Cloud Computing Security?, University of California at Berkeley, disponível em <eecs.berkeley.edu>. Acesso Fev. 2012.

CHOW, R.; GOLLE, P.; JAKOBSSON, M.; SHI, E.; STADDON, J.; MASUOKA, R.; MOLINA, J. (2009). Controlling data in the cloud: outsourcing computation without outsourcing control. Proceedings of ACM CCSW. ACM, pp. 85-90.

COMUZZI, M; KOTSOKALIS, C.; SPANOUDAKIS, G.; YAHYAPOUR, R. (2009). Establishing and Monitoring SLAs in complex Service Based Systems. ICWS, pp.783-790.

CSA GRC STACK RESEARCH GROUP. (2010). Cloud Security Alliance GRC Stack, disponível em: <cloudsecurityalliance.org/research/projects/grc-stack>. Acesso: Jan. 2012.

CSA RESEARCH GROUP. (2010). Guia de Segurança para Áreas Críticas Focado em Computação em Nuvem, disponível: <cloudsecurityalliance.org/guidance>. Acesso em Dez. 2012.

EMEAKAROHA, VC.; BRANDIC, I.; MAURER, M.; DUSTDAR, S. (2010). Low Level Metrics to High Level SLAs - LoM2HiS Framework: Bridging the GapBetween Monitored Metrics and SLA Parameters in Cloud Environments. Proc. HPCS, pp.48-54.

FIEDLER, M.; HOSSFELD, T.; PHUOC, TG. (2010). A generic quantitative relationship between quality of experience and quality of service, Network, IEEE , 24, 2, pp. 36-41.

GUO, Z.; SONG, M.; SONG, J. (2010). A Governance Model for Cloud Computing, Proc. MASS, pp.1-6.

HAEBERLEN, A. (2010). A Case for the Accountable Cloud, ACM SIGOPS Operating Systems Review, 44, 2, pp. 52-57.

HUBBARD, D; SUTTON, M. (2012). Top Threats to Cloud Computing, Cloud Security Alliance, disponível em: <cloudsecurityalliance.org/topthreats>. Acesso em Fev. 2012.

IDC. (2010). It cloud services user survey, pt.2: Top benefits & challenges, disponível em: <blogs.idc.com/ie/?p=210>. Acesso em Dez. 2011.

JANSEN, W.; GRANCE, T.(2011). Guidelines on Security and Privacy in Public Cloud Computing, disponível em: <nist.gov.br>. Acesso em Nov. 2011.

LANDWEHR, C.E. (2001). Computer Security**,** International Journal of Information Security, Springer Berlin / Heidelberg, pp. 3-13.

MELL, P; GRANCE T.(2011). The NIST Definition of Cloud Computing**,** disponível em <nist.gov>. Acesso em Out. 2011.

MOORSEL, AV. (2001). Metrics for the Internet Age: Quality of Experience and Quality of Business, disponível em: <www.hp.com>. Acesso em Mai 2012.

NURMI, D.; WOLSKI, R.; GRZEGORCZYK, C.; OBERTELLI, G.; SOMAN, S.; YOUSEFF, L.; ZAGORODNOV, D. (2009). The Eucalyptus Open-Source Cloud-Computing System. Proc. of IEEE/ACM CCGRID. IEEE Computer, pp. 124-131.

OPEN CLOUD STANDARDS INCUBATOR – DMTF. (2010). Architecture for Managing Clouds, disponível em: <www.dmtf.org >. Acesso em Jan. 2012.

SAUVÉ, J., MARQUES, F.; MOURA, A.; SAMPAIO, M.; JORNADA, J.; RADZIUK, E.; SCHÖNWÄLDER, J. (2005). SLA design from a business perspective, Ambient Networks - Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp. 72-83.

SKENE, J.; RAIMONDI, F.; EMMERICH, W. (2010). Service-Level Agreements for Electronic Services, IEEE Transactions on Software Engineering, 36, 2, pp. 288-304.

WLOKA J., SRIDHARAN, M., TIP, F. (2009). Refactoring for reentrancy. Proc. ACM SIGSOFT and ESEC/FSE. ACM, pp. 173-182.

