Um Middleware para Encenação Automatizada de Coreografias de Serviços Web em Ambientes de Computação em Nuvem

Leite Leonardo, Nelson Lago, Marco Aurelio Gerosa, Fabio Kon

Um Middleware para Encenação Automatizada de Coreografias de Serviços Web em Ambientes de Computação em Nuvem

Leonardo Leite, Nelson Lago, Marco A. Gerosa, Fabio Kon [1]

[1]Departamento de Ciência da Computação
IME - Universidade de São Paulo (USP)

{leofl,lago,gerosa,fabio.kon}@ime.usp.br

Abstract. *Automated deployment is an important issue in large-scale distributed systems, such as web services choreographies that implement business processes encompassing multiple organizations. In this paper, we present the CHOReOS Enactment Engine, a novel extensible open source middleware system that provides a platform for automation of the distributed deployment of web service choreographies in cloud computing environments. We evaluated the choreographies enactment time experimentally, varying the amount of services to be deployed and the amount of virtual machines available to the deployment process. The CHOReOS Enactment Engine enabled the automated deployment of service compositions on the cloud in an automated and reproducible way.*

Resumo. *A implantação automatizada é uma importante necessidade em sistemas distribuídos de grande escala, como é o caso de coreografias de serviços web, que implementam processos de negócios envolvendo várias organizações. Neste artigo, apresentamos o CHOReOS Enactment Engine, um novo sistema de middleware extensível que fornece uma plataforma para a implantação distribuída e automatizada de coreografias de serviços web em ambientes de computação em nuvem. Para avaliar a eficácia e escalabilidade do sistema, medimos experimentalmente o tempo de encenação de coreografias, tendo como carga a quantidade de serviços a serem implantados, e como recursos a quantidade de máquinas virtuais disponíveis para o processo de implantação. Nosso sistema de middleware está disponível como software livre e possibilita a implantação de complexas composições de serviços na nuvem de forma automatizada e reprodutível.*

1. Introdução

Profissionais da indústria já reconhecem a necessidade de processos totalmente automatizados para a implantação de sistemas [Humble and Farley 2011]. No entanto, muitas organizações ainda realizam a implantação de seus sistemas de forma manual, tornando esse processo moroso, propenso a erros e não-reprodutível [Dolstra et al. 2005]. O problema se agrava na implantação de sistemas distribuídos, pois o esforço de implantação cresce com a quantidade de nós do sistema.

Sistemas distribuídos estão migrando para ambientes de nuvem, nos quais são compostos e mantidos de forma descentralizada por várias organiza-

ções [Steen et al. 2012]. A computação em nuvem possibilita o acesso a um conjunto compartilhado de recursos computacionais que podem ser providos rapidamente [Mell and Grance 2011], favorecendo a criação de um processo de implantação totalmente automatizado. O modelo de Infraestrutura como Serviço (IaaS) fornece acesso a recursos virtualizados, como máquinas virtuais, de forma programática. Em contrapartida, no modelo de Plataforma como Serviço (PaaS), como o adotado pelo Google App Engine[1], os desenvolvedores da aplicação não se preocupam diretamente com a gerência dos recursos virtualizados ou com a configuração dos ambientes nos quais a aplicação será implantada. A nuvem também pode ser púlica, com clientes externos, como no caso da nuvem da Amazon[2], ou privada, com clientes internos, situação na qual a organização utilizam ambientes baseados em middleware como o OpenStack[3] [Zhang et al. 2010].

Serviços web são componentes que se comunicam pela rede por meio da utilização de protocolos baseados em padrões da Web [W3C 2004], e que podem ser compostos de forma a se obter um sistema mais complexo, mas com baixo acoplamento entre seus componentes. Em particular, coreografias são composições que implementam processos de negócios distribuídos, normalmente entre organizações, de maneira a diminuir o número de mensagens trocadas e distribuir a lógica do negócio entre as organizações envolvidas, dispensando o uso de coordenadores centralizados, uma vez que cada serviço "sabe" quando executar suas operações e com quais outros serviços interagir [Barker et al. 2009]. Um exemplo de notação para a especificação de coreografias é o BPMN2 [OMG 2011]. Para coreografias de serviços web, o processo de implantação deve ser distribuído. A *encenação* de uma coreografia deve ser coordenada, pois além de implantados no ambiente distribuído, os serviços da coreografia precisam conhecer a localização dos outros serviços, informação disponível apenas em tempo de implantação.

Neste artigo, apresentamos o CHOReOS Enactment Engine, um sistema de middleware que fornece uma plataforma como serviço para a implantação distribuída e automatizada de composições de serviços web, com ênfase em coreografias, em ambientes de computação em nuvem. O CHOReOS Enactment Engine é software livre[4] e desenvolvido no contexto do projeto CHOReOS, financiado pela Comissão Europeia com o objetivo de possibilitar a utilização de coreografias de serviços web em cenários de escala ultra grande. Nosso trabalho se diferencia dos trabalhos relacionados ao explorar como o ambiente de computação em nuvem traz benefícios ao processo de implantação, bem como ao considerar as restrições que esses ambientes impõem, como a falta de previsibilidade dos endereços das máquinas em tempo de configuração do serviço.

O CHOReOS Enactment Engine recebe uma especificação declarativa da coreografia, realiza a encenação da coreografia e devolve ao cliente informações sobre a localização de cada serviço implantado. A especificação da coreografia é uma descrição arquitetural, que deve ser criada pelo cliente da operação de encenação. Isso pode ser feito manualmente, mas também por processos automatizados por meio da conversão de modelos gráficos de mais alto nível, como o BPMN2[5]. Durante a implantação, o middleware

[1] `developers.google.com/appengine`

[2] `aws.amazon.com`

[3] `www.openstack.org`

[4] Código-fonte disponível em `github.com/choreos/choreos_middleware`.

[5] Essa é a abordagem de utilização do CHOReOS Enactment Engine implementada no contexto do projeto CHOReOS [Ben Hamida et al. 2012].

também realiza a troca de localizações dos nós da coreografia e configura dinamicamente os serviços para que a coreografia possa ser encenada.

Soluções comerciais de Plataforma como Serviço normalmente impõem restrições severas quanto às possíveis tecnologias a serem usadas por seus usuários no desenvolvimento de aplicações. Para amenizar esse problema, nosso sistema possui pontos de extensão que possibilitam a adequação do middleware a novas tecnologias. Esses pontos de extensão não dizem respeito apenas às tecnologias de desenvolvimento de serviços web, mas também à interação com serviços de infraestrutura, possibilitando que cada organização escolha sua plataforma de computação em nuvem.

Um aumento significativo na carga de um sistema normalmente tem grande impacto no seu tempo de execução. Consideramos um sistema escalável se for possível amenizar esse efeito por meio de um aumento proporcional nos recursos utilizados pelo sistema, ou seja, se um aumento nos recursos do sistema implicar em um aumento diretamente proporcional na sua capacidade de processamento [Law 1998]. Assim, para avaliar a eficácia e escalabilidade do CHOReOS Enactment Engine, medimos experimentalmente o tempo de implantação de coreografias variando a carga – quantidade de serviços – e os recursos disponíveis – quantidade de máquinas virtuais disponíveis para o processo de implantação.

Na próxima seção descrevemos alguns dos principais trabalhos relacionados à nossa abordagem. A Seção 3 apresenta a solução proposta e sua implementação. A Seção 4 apresenta resultados experimentais e, na Seção 5, descrevemos nossas conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Um dos aspectos fundamentais sobre processos de implantação automatizados é a linguagem para sua configuração. A linguagem pode ser procedimental [Dolstra et al. 2005] ou declarativa [Magee and Kramer 1996, Balter et al. 1998]. Outro aspecto relevante é o escalonamento de recursos computacionais para os serviços a serem implantados, o que pode ser feito, por exemplo, com o auxílio de sistemas de computação em grade [Watson et al. 2006].

Antes de instalar um serviço é preciso configurar adequadamente o sistema operacional e a plataforma na qual o serviço será implantado. No uso de ferramentas como Chef[6], Capistrano[7] e Nix [Dolstra et al. 2005], os usuários escrevem *scripts* que implementam o processo de configuração do ambiente e a implantação do serviço. No caso do Chef, um script configura a máquina na qual o serviço é instalado, enquanto que o Capistrano possibilita a coordenação da implantação de serviços em diferentes nós. Com as expressões Nix, é possível também unificar a especificação da implantação com o *build* da aplicação em um único script, possibilitando a edição parametrizada de arquivos de configuração da aplicação em função do local de implantação.

A abordagem procedimental possibilita ao usuário especificar a implantação de vários tipos de sistemas, mas normalmente requer especialização de seus usuários, pois vários detalhes do processo devem ser especificados. Esses *scripts* de implantação tam-

[6] `www.opscode.com/chef`
[7] `github.com/capistrano`

bém precisam ser desenvolvidos com o mesmo rigor do código da aplicação, inclusive com o uso de testes automatizados [Humble and Farley 2011]. Caso contrário, o processo de implantação torna-se pouco robusto e não confiável. Uma alternativa que evita essa sobrecarga no processo de desenvolvimento é o uso de sistemas especializados na implantação de determinados tipos de aplicações e que recebam, como entrada, uma simples especificação declarativa do sistema.

Um exemplo de abordagem declarativa é encontrado no uso de Linguagens de Descrição Arquitetural (ADLs), como a Darwin [Magee and Kramer 1996]. A linguagem Darwin se foca nos aspectos estruturais de sistemas distribuídos, descrevendo a conexão entre os módulos do sistema, mas sem descrever implementações ou sequências de interações entre os módulos. Em nosso trabalho, também descrevemos o sistema a ser implantado por meio de sua descrição estrutural, uma vez que é esse o aspecto necessário para que se possa automatizar o processo de implantação.

Regis, o ambiente de execução da linguagem Darwin [Magee et al. 1994], possui duas políticas de distribuição de programas por estações de trabalho. A primeira é o mapeamento definido pelo usuário de forma estática, abordagem não apropriada para ambientes de computação em nuvem. A segunda opção de política é a alocação automática em função da carga na CPU das estações de trabalho, não havendo flexibilidade para a consideração de outros recursos, como espaço em disco ou memória, por exemplo. Uma similaridade entre Regis e o CHOReOS Enactment Engine é o uso do middleware para o envio de mensagens contendo referências remotas dos componentes implantados para que eles possam estabelecer ligações dinâmicas entre si.

Olan [Balter et al. 1998] é um ambiente para a descrição, configuração e implantação de aplicações distribuídas em ambientes heterogêneos, e que também utiliza uma ADL própria. Baseando-se na entrada descrita na ADL, Olan gera scripts de Configuração de Máquina, que definem a execução do processo de implantação dos componentes no ambiente distribuído e do ajuste dos canais de comunicação entre esses componentes. A abordagem de gerar um script de configuração a partir de uma especificação declarativa é também implementada pelo CHOReOS Enactment Engine. A ADL de Olan também possibilita a especificação de restrições sobre a localização da implantação do componente, porém sem flexibilidade para a adoção de estratégias dinâmicas de alocação de nós.

O estudo de Quéma et al. [Quéma et al. 2004] realiza avaliações empíricas sobre desempenho e escalabilidade do processo de implantação proposto, que é executado de forma distribuída por agentes que se comunicam de forma assíncrona e hierárquica de acordo com a estrutura da composição sendo implantada, descrita por uma ADL. Essa estrutura hierárquica, no entanto, é apenas um caso particular das possibilidades na topologia de uma coreografia de serviços, o que impossibilita que essa solução seja diretamente adotada em nosso contexto. A avaliação de escalabilidade resulta em um aumento aproximadamente proporcional no tempo de implantação com o aumento proporcional da quantidade de componentes e da quantidade de máquinas disponíveis.

Os trabalhos descritos até aqui apresentam abordagens simples para o problema da distribuição dos componentes implantados pelas máquinas disponíveis. Já o trabalho de Watson et al., apresenta uma abordagem mais completa para esse problema com o uso de grades computacionais [Watson et al. 2006]. O foco da solução apresentada está

em escolher dinamicamente o provedor de infraestrutura e a máquina em que um serviço web deve ser implantado considerando os requisitos não-funcionais do serviço web. Isso é realizado não somente para a primeira implantação do serviço web, mas também para as replicações que ocorrem quando as instâncias existentes não conseguem mais atender aos requisitos não-funcionais. Uma desvantagem dessa abordagem é a carga adicional gerada pela análise dos requisitos não-funcionais a cada troca de mensagens efetuada pelos serviços implantados. Embora o trabalho de Watson et al. avance na problemática da distribuição dos serviços, nenhum dos trabalhos analisados considera as potencialidades e desafios dos ambientes de computação em nuvem, que oferecem serviços de infraestrutura para a gerência de recursos virtualizados.

3. CHOReOS Enactment Engine

Nesta seção descreveremos o funcionamento do CHOReOS Enactment Engine, abordando sua interface, arquitetura e pontos de extensão.

3.1. Interface

As funcionalidades do CHOReOS Enactment Engine são expostas como serviços REST, sendo a encenação de coreografias a principal operação disponível. Para executar essa operação, o middleware recebe uma descrição arquitetural da coreografia em formato XML, estruturado de acordo com o modelo de classes apresentado na Figura 1. A descrição de uma coreografia consiste numa lista de descrições de seus serviços, sendo cada serviço modelado de acordo com a classe ChorServiceSpec. Com base nessa descrição, o CHOReOS Enactment Engine implanta os artefatos em diversos nós nos ambientes de nuvem disponíveis. O cliente da operação de encenação recebe como resposta dados que modelam os serviços implantados de acordo com a classe Service, estrutura que informa em que nó cada serviço da coreografia foi implantado, assim como a URI de acesso ao serviço.

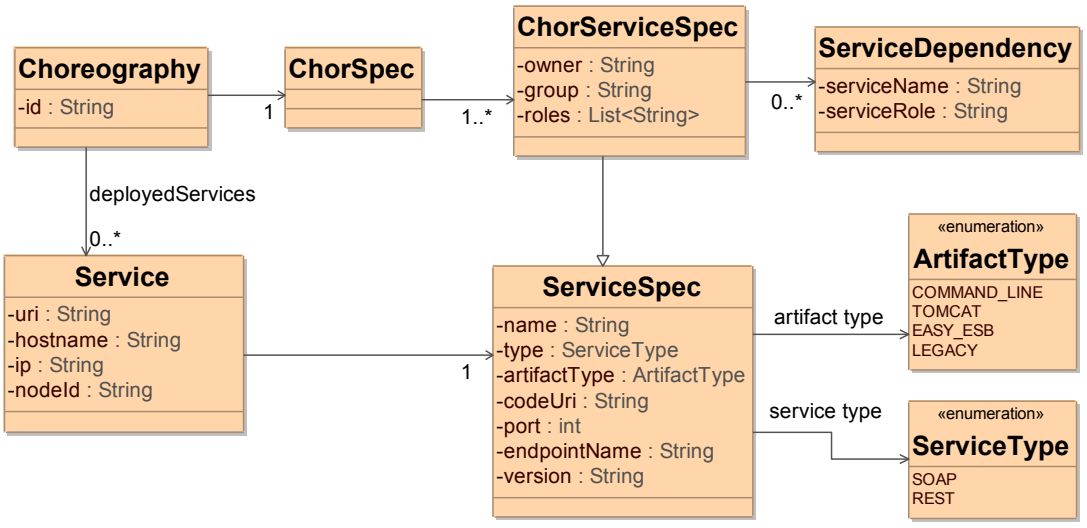

Figura 1. Estrutura da descrição arquitetural de uma coreografia.

O middleware implanta cada serviço acessando seu artefato implantável, identificado pelo atributo `codeUri`, e o instalando de acordo com procedimentos relacionados

ao tipo do artefato, identificado pelo atributo `artifactType`. Caso o tipo do artefato seja `COMMAND_LINE`, por exemplo, o artefato deve ser um arquivo jar que possa ser executado com o comando `java -jar`. Para que o middleware identifique a URI de acesso ao serviço, ainda é preciso informar na especificação do serviço os atributos `endpointName` e `port`, que são partes constituintes dessa URI.

Cada serviço na coreografia pode consumir operações de outros serviços da coreografia. Para isso, é preciso que cada serviço consumidor conheça a URI de acesso a cada serviço provedor do qual depende. Em uma implantação em um ambiente de nuvem, em que máquinas virtuais podem ser criadas no momento da encenação, esse endereço será conhecido apenas após a implantação de cada serviço. Assim, para possibilitar a ligação dinâmica entre os serviços da coreografia, cada serviço consumidor deve implementar uma operação denominada `setInvocationAddress`, que recebe a URI de um serviço provedor que pode desempenhar o papel de uma das dependências do serviço consumidor. A relação de quais serviços satisfazem as necessidades de outros serviços é definida na especificação da coreografia, utilizando a lista de elementos do tipo ServiceDepency, pertencentes a cada especificação de serviço.

Uma descrição completa da estrutura da especificação da coreografia, incluindo o *schema* XML correspondente e exemplos de descrições de coreografia em XML, está disponível em `http://ccsl.ime.usp.br/baile/files/ee_rest_api.pdf`. Esse documento contém também a descrição da interface REST do middleware.

3.2. Arquitetura

O CHOReOS Enactment Engine é composto por componentes que são instanciados nas infraestruturas das organizações participantes da coreografia ou em nuvens públicas. A relação de uso entre esses componentes é exibida na Figura 2. Os componentes Choreography Deployer e Deployment Manager são fornecidos pelo próprio middleware. Os componentes Chef Client e Chef Server são partes do Chef, uma ferramenta de código aberto para a gerência de configuração de software. Por fim, o componente Cloud Gateway é algum serviço de infraestrutura para a gerência de máquinas virtuais, como, por exemplo, o serviço Amazon EC2. A encenação de uma coreografia requer uma instância do componente Choreography Deployer e que cada organização possua, em sua infraestrutura, uma ou mais instâncias dos componentes Cloud Gateway, Chef Server e Deployment Manager. A seguir, descreveremos brevemente cada um desses componentes.

Cloud Gateway. Cria e destrói máquinas virtuais em uma infraestrutura de computação em nuvem. Neste contexto, essas máquinas virtuais são também chamadas *nós*. Esse componente é utilizado pelo Deployment Manager, que decide quando criar ou destruir os nós. Uma instância do Deployment Manager deve ser configurada de acordo com o Cloud Gateway a ser utilizado. Atualmente, podem ser utilizados como Cloud Gateway o serviço EC2 dos Web Services da Amazon, ou uma instalação do OpenStack. Outras tecnologias de nuvem ou virtualização podem ser acrescentadas mediante extensão do Deployment Manager.

Chef Server. Em um sistema gerenciado pelo Chef, as configurações de software de um nó são especificadas através de scripts denominados "receitas". As receitas podem especificar configurações do sistema operacional (p.ex., instalar o Java), instalação e configuração de middleware (p.ex., instalar o Tomcat), assim como a instalação de um serviço final

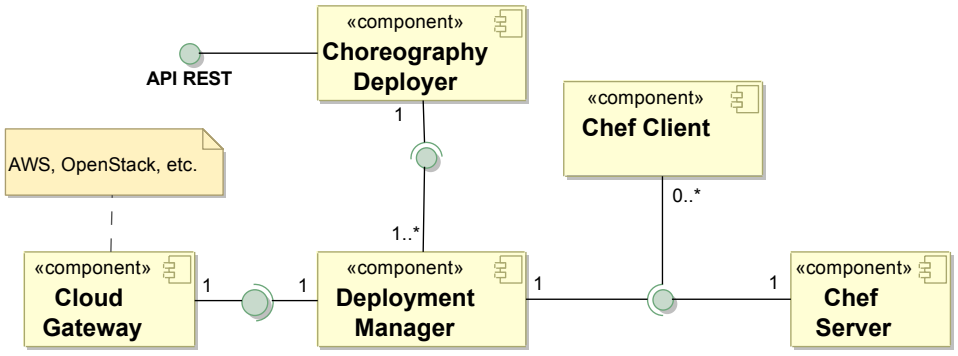

Figura 2. Arquitetura do CHOReOS Enactment Engine.

implantado sobre o middleware já configurado (p.ex., um serviço web). Essas receitas e as associações entre receitas e nós são armazenadas no Chef Server.

Chef Client. Em cada nó gerenciado pelo Chef, é instalado um programa denominado Chef Client, que utiliza a API REST do Chef Server para obter as receitas relacionadas ao nó e assim atualizar suas configurações de software.

Deployment Manager. Implanta os serviços em um ambiente de nuvem associado a uma organização. Através de uma operação REST, o Deployment Manager recebe a especificação de um serviço e seleciona um nó adequado para sua implantação, possivelmente considerando os requisitos não-funcionais do serviço. A especificação do serviço é descrita em formato XML e modelada de acordo com a classe ServiceSpec (Figura 1). O atributo `artifactType` determinará o procedimento para converter a especificação declarativa do serviço em uma receita Chef que implemente seu processo de instalação. Após gerar a receita, o Deployment Manager a armazena no Chef Server e, também utilizando o Chef, cria uma associação entre essa receita e o nó selecionado. Utilizando outra operação REST do Deployment Manager pode-se requisitar que um determinado nó seja atualizado de acordo com as receitas a ele relacionadas. Essa divisão do processo em duas chamadas ao Deployment Manager é adotada para evitar a sobrecarga de execuções desnecessárias do Chef Client.

Choreography Deployer. Expõe uma API REST de operações que automatizam a implantação de coreografias de serviços web. O cliente do Choreography Deployer fornece a especificação declarativa da coreografia (Figura 1). Baseado nessa especificação, o Choreography Deployer coordena invocações aos Deployment Managers implantados nas diferentes organizações que participam da coreografia. Após a implantação dos serviços, para cada par de serviços que se comunicam na coreografia, o Choreography Deployer informa ao serviço consumidor a localização do serviço provedor.

O diagrama de sequência apresentado pela Figura 3 ilustra o processo de encenação de uma pequena coreografia de dois serviços, na qual o serviço A é consumidor do serviço B. O diagrama de sequência ajuda a evidenciar detalhes da interação entre os componentes do sistema.

O processo de implantação se inicia com duas requisições ao Choreography De-

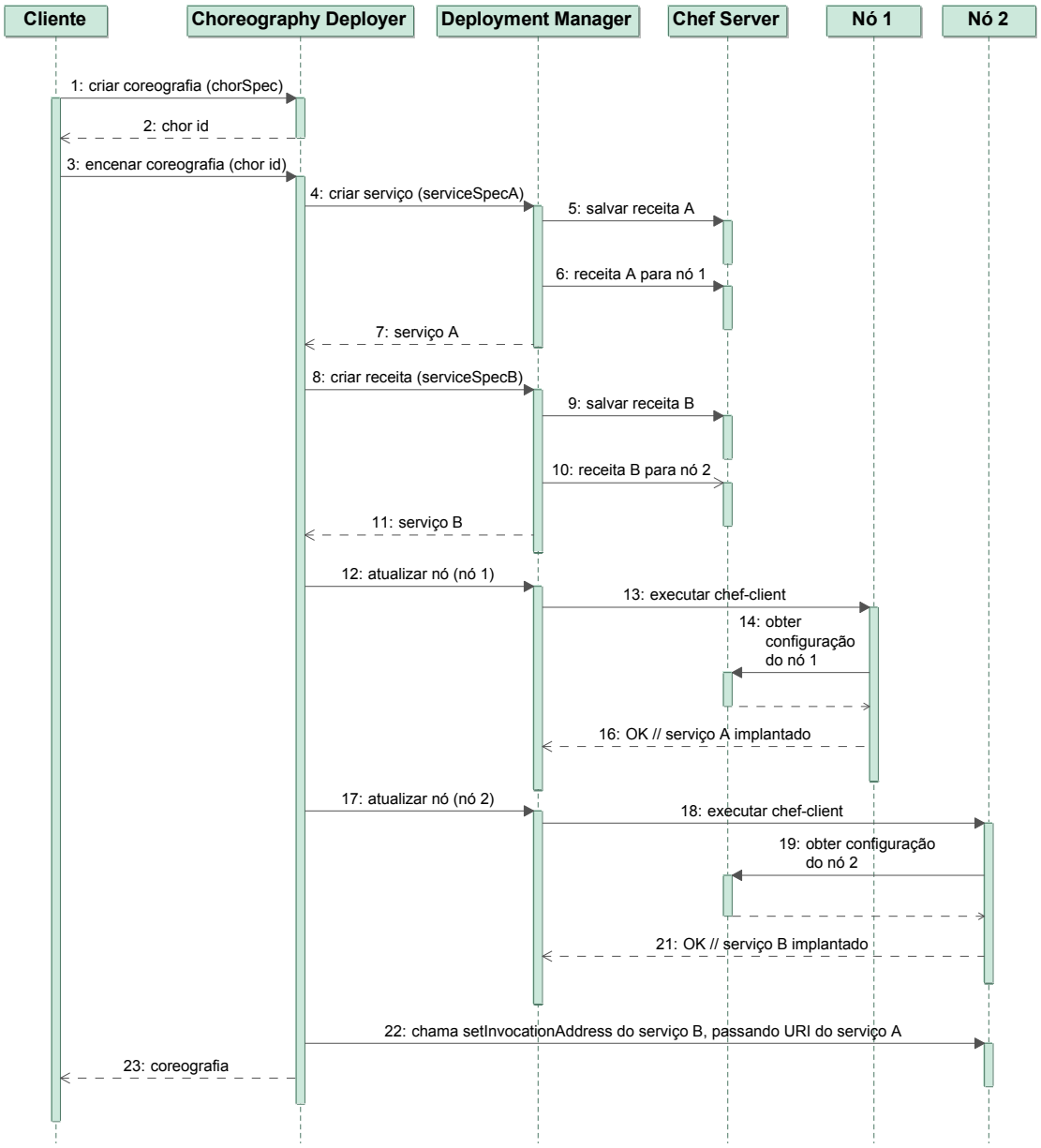

Figura 3. Exemplo de encenação de uma coreografia de dois serviços.

ployer: uma para a criação do recurso[8] *coreografia* associado à especificação fornecida pelo cliente e outra para a encenação da coreografia. Para cada especificação de serviço contido na especificação da coreografia, o Choreography Deployer requisita a criação de um recurso *serviço* ao Deployment Manager. Para isso, o Deployment Manager cria uma receita Chef que automatiza a instalação do serviço, armazena-a no Chef Server e associa a receita a algum nó por ele selecionado. O Deployment Manager retorna ao Choreography Deployer o recurso *serviço*, contendo a URI de acesso a esse serviço e uma referência ao nó selecionado. Em seguida, o Choreography Deployer requisita ao Deployment Manager a atualização de todos os nós selecionados. Quando o Deployment Manager re-

[8]Usamos aqui o termo "recurso" em seu sentido usual em arquiteturas REST.

cebe um pedido de atualização de nó, ele se conecta por SSH a esse nó e executa o Chef Client, que se comunica com o Chef Server para obter as receitas associadas àquele nó e instalar os serviços com base nas receitas obtidas. O Choreography Deployer ainda invoca a operação `setInvocationAddress` do serviço B, informando ao serviço B a URI de acesso do serviço A. Finalmente, o Choreography Deployer devolve ao cliente a representação da coreografia, contendo informações da implantação, especialmente as URIs dos serviços em execução.

Uma observação sobre o diagrama da Figura 3 é que todas as operação foram dispostas de forma sequencial para facilitar a leitura do diagrama, mas o Choreography Deployer realiza as chamadas de criação dos serviços em paralelo, assim como as chamadas de atualização dos nós e as invocações às operações `setInvocationAddress`.

3.3. Pontos de Extensão

Para possibilitar uma maior flexibilidade aos usuários do CHOReOS Enactment Engine em relação às possíveis escolhas de tecnologias adotadas, nosso middleware oferece os pontos de extensão seguintes.

CloudProvider. A implementação dessa interface permite a utilização de outros serviços de infraestrutura para a gerência de máquinas virtuais. Um exemplo seria codificar uma implementação VirtualBoxCloudProvider que utilizasse os recursos do VirtualBox[9] para criar máquinas virtuais no próprio ambiente do desenvolvedor.

NodeSelector. A implementação dessa interface define uma nova política de alocação de serviços em nós da nuvem, que pode levar em conta os requisitos não funcionais do serviço e propriedades dos nós à disposição. Um exemplo simples é a implementação AlwaysCreateNodeSelector, que cria um novo nó para cada requisição realizada.

RecipeBuilder. A implementação dessa interface possibilita o suporte a novos tipos de artefatos implantáveis. Um exemplo seria a codificação de uma implementação PythonRecipeBuilder, que gerasse receitas de serviços web codificados em Python e acessíveis pelo servidor web Apache.

ContextSender. A implementação dessa interface possibilita que o Enactment Engine ofereça suporte a novos tipos de serviços ao se definir um procedimento para a invocação da operação `setInvocationAddress`. Um exemplo de nova implementação seria o JMSContextSender, que definiria uma convenção para a construção da operação `setInvocationAddress` de serviços JMS participantes de uma coreografia.

4. Avaliação Experimental

Quando implantadas continuamente nos ambientes de teste e homologação, aplicações oferecem retorno rápido para os desenvolvedores sobre os efeitos de suas modificações e sobre eventuais defeitos no sistema [Humble and Farley 2011]. Quanto mais demorado for o processo de implantação, mais demorado será o *feedback* aos desenvolvedores e, em alguns casos, a demora nesse processo desmotiva sua execução frequente. Por esses motivos, um dos objetivos do CHOReOS Enactment Engine é oferecer um processo de implantação escalável, no qual o aumento na disponibilidade de recursos oferecidos (tais

[9] www.virtualbox.org

como máquinas virtuais) deve possibilitar que um aumento na carga do processo de implantação (por exemplo, a quantidade de serviços a serem implantados) não provoque um grande aumento no tempo de execução.

Nesta seção, apresentamos a avaliação de escalabilidade do CHOReOS Enactment Engine. A unidade de carga do sistema é a quantidade de coreografias a serem implantadas, enquanto que a unidade de recursos fornecidos ao sistema é a quantidade de máquinas virtuais alocados no ambiente de computação em nuvem. Cada coreografia possui dois serviços: o serviço `TravelAgency`, que é invocado pelo cliente da coreografia, e o serviço `Airline`, que é invocado pelo serviço `TravelAgency`.

Definimos uma *execução* do experimento como o processo de encenações simultâneas de N coreografias, sendo os valores utilizados para N: 1, 5, 10, 15, 20, 25. Cada execução é composta por três *fases*: 1) a criação de $2 * N$ nós no ambiente de computação em nuvem (um nó por serviço), 2) a implantação das N coreografias nos nós criados na fase anterior e 3) a invocação das coreografias para verificação do funcionamento adequado do processo de negócio. Para cada fase em cada execução, registramos três valores: o *tempo total* (t) de duração da fase, o *tempo médio* (m) gasto por cada coreografia naquela fase e o *desvio padrão* (s) dos valores usados no cálculo do tempo médio. Tomando como exemplo a fase de implantação de uma execução com $N = 5$, o tempo médio é a média aritmética dos tempos de implantação das 5 coreografias, o desvio padrão é o desvio padrão dos tempos de implantação e o tempo total é o tempo gasto para que as coreografias sejam implantadas. Para cada valor de N, coletamos uma amostra de 10 execuções sem falhas[10], desconsiderando assim execuções com falhas provenientes do provedor de IaaS.

O Cloud Gateway utilizado foi o serviço EC2 da Amazon e a política de alocação de nós foi a de alocar um nó por serviço. As execuções foram realizadas com os componentes Choreography Deployer e Deployment Manager instanciados em um Mac Book Pro 8 (8GB de RAM, processador Intel Core i7 com 2,7GHz) utilizando o sistema operacional GNU/Linux (kernel 3.6.7) e a JVM OpenJDK 6. O Chef Server utilizado estava instalado em um servidor localizado em nossa universidade, de onde as requisições para as execuções foram originadas. A versão do código-fonte utilizada nas execuções foi marcada com a tag `sbrc2013` em nosso repositório público[11].

Os resultados experimentais da fase de criação de nós são apresentados na Tabela 1, os resultados da fase de implantação na Tabela 2, os resultados da fase de invocações na Tabela 3 e os tempos totais das execuções na Tabela 4. As tabelas com dados de fases (Tabelas 1, 2 e 3) mostram os seguintes valores para cada valor de N: média aritmética dos tempos médios ($\overline{m}$); mediana dos tempos médios ($m_{1/2}$); média aritmética dos desvios padrões ($\overline{s}$); média aritmética dos tempos totais ($\overline{t}$); mediana dos tempos totais ($t_{1/2}$); e desvio padrão dos tempos totais (s_t).

O principal resultado observado é que um aumento de 25 vezes no número de coreografias implantadas apenas dobrou o tempo total das execuções, subindo o tempo

[10] Os dados brutos coletados durante as execuções se encontram em
`http://ccsl.ime.usp.br/baile/files/ee_scalability_sbrc2013.zip`

[11] O código dessa versão pode ser baixado diretamente de
`http://github.com/choreos/choreos_middleware/archive/sbrc2013.zip`

Tabela 1. Tempos da fase de criação dos nós (s)

	m : tempo médio da criação de um nó s : desvio padrão do tempo de criação de nó			t : tempo total da fase de criação de nós		
N	$\overline{m}$	$m_{1/2}$	$\overline{s}$	$\overline{t}$	$t_{1/2}$	s_t
1	103.5	114.0	6.5	120.1	116.5	11.9
5	108.1	107.9	8.0	127.9	125.0	11.4
10	128.0	126.1	18.3	185.3	154.9	95.0
15	141.6	140.5	16.2	180.2	175.7	11.7
20	146.4	142.6	24.3	239.8	208.4	69.7
25	170.0	173.1	26.3	238.0	234.7	16.2

Tabela 2. Tempos da fase de implantação (s)

	m : tempo médio da implantação de uma coreografia s : desvio padrão do tempo de implantação			t : tempo total da fase de implantações		
N	$\overline{m}$	$m_{1/2}$	$\overline{s}$	$\overline{t}$	$t_{1/2}$	s_t
1	155.6	154.2	0.0	155.6	154.3	11.4
5	157.2	159.5	9.2	170.0	169.0	13.7
10	189.1	187.4	12.1	211.1	208.1	10.4
15	217.3	217.8	11.9	242.9	239.3	12.5
20	243.0	243.1	15.6	275.7	274.9	11.7
25	271.3	270.6	18.6	313.7	308.6	21.8

Tabela 3. Tempos da fase de invocação (s)

	m : tempo médio de uma invocação s : desvio padrão do tempo de invocação			t : tempo total da fase de invocações		
N	$\overline{m}$	$m_{1/2}$	$\overline{s}$	$\overline{t}$	$t_{1/2}$	s_t
1	0.80	0.80	0.00	0.80	0.80	0.05
5	0.82	0.81	0.04	0.88	0.88	0.05
10	0.88	0.88	0.06	1.01	0.98	0.06
15	0.87	0.87	0.06	1.02	0.98	0.13
20	0.89	0.88	0.07	1.08	1.01	0.13
25	0.92	0.92	0.14	1.49	1.42	0.48

Tabela 4. Tempos totais das execuções (s)

	t : tempo total de execução		
N	$\overline{t}$	$t_{1/2}$	s_t
1	276.5	281.3	17.2
5	298.7	296.4	20.1
10	397.4	365.9	92.9
15	424.1	419.0	22.5
20	516.6	486.7	75.0
25	553.1	540.1	34.8

de encenação de 4,6 minutos em média para apenas 9,2 minutos em média, o que consideramos um resultado muito satisfatório. Podemos também observar o crescimento dos tempos de execuções e de cada uma de suas fases nas Figuras 4 e 5. É interessante notar que a fase de invocação consome aproximadamente apenas 1 segundo, com uma variância muito pequena. Esse comportamento para a fase de invocação era esperado, considerando que os serviços utilizados são bem leves, que as máquinas virtuais criadas são equivalentes em suas capacidades computacionais e que cada nó recebeu apenas um serviço.

Quando verificamos a evolução da encenação de 1 coreografia até 25 coreografias, notamos que o aumento na média do tempo de criação de nós foi de 64%, enquanto que o aumento na média do tempo total da fase de criação de nós foi de 98%. A média do tempo de implantação das coreografias subiu 75% e o aumento na média do tempo da fase de implantação de coreografias foi de 102%. O aumento maior nos tempos totais das fases se explica pela natureza dos dados coletados: para a fase de criação dos nós, verificamos que a maior parte dos nós criados segue um bom comportamento mas, a cada execução, é comum que pelo menos um dos nós da Amazon EC2 demore bem mais para ficar pronto. Às vezes também ocorre de algum nó não ficar pronto para uso, situação na qual nosso middleware requisita a alocação de um novo nó. Como o tempo total da fase de criação de nós é próximo ao tempo mais demorado de criação de um nó, fica claro que eventos

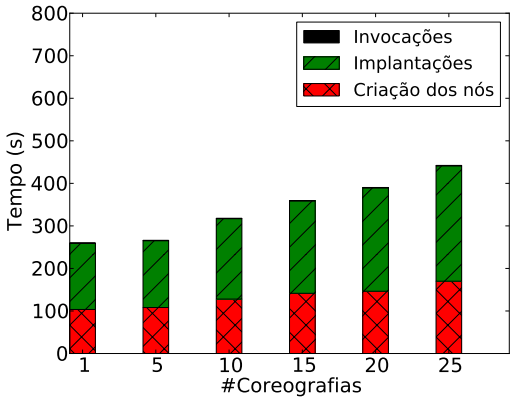

Figura 4. Média dos tempos de encenação das coreografias

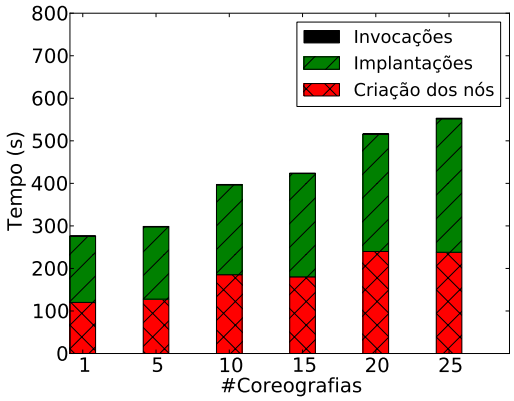

Figura 5. Média dos tempos totais de encenação das coreografias

esporádicos de nós muito demorados afetam mais contundentemente o tempo total da fase do que o tempo médio de criação de nós. Fenômeno similar ocorre, em menor escala, na fase de implantação da coreografia.

A presença de eventos esporádicos que são mais demorados explica também o fato de que as medianas para os tempos totais de fases sejam menores do que as médias correspondentes. Portanto, a mediana representa melhor do que a média o valor esperado de um evento como a criação de um nó. Por outro lado, o fato de a mediana dos tempos médios ser bem próxima à média correspondente sugere que os eventos que "distorcem" a média dos tempos totais são de fato bastante raros. Outra conclusão que podemos tirar sobre a presença desses eventos esporádicos é a de que não podemos assumir uma distribuição normal para eventos aleatórios de criação de nós e de implantações de coreografias. Esses eventos esporádicos também fazem com que os valores representando os desvios dos tempos totais das fases sejam maiores do que os desvios dos tempos médios.

Outra observação relevante é que os desvios padrões da criação de nós são maiores do que os desvios padrões da implantação de coreografias. Pode-se chegar à mesma conclusão com o coeficiente de variação (razão entre desvio padrão e a média) no lugar do desvio padrão, já que os desvios na criação de nós são maiores do que os desvios das implantações, mesmo sendo os tempos de criação de nós menores do que os tempos de implantação. Essa análise evidencia que a variabilidade do tempo de alocação de um nó pela Amazon é o principal fator na variabilidade do tempo de execução. Em especial, observamos na Tabela 1 que os desvios padrões das fases de criação de nós foi muito maior para $N = 10$ e $N = 20$, o que se deve principalmente aos já mencionados eventos esporádicos. Na Figura 5, pode-se notar como a presença mais forte dos eventos esporádicos na criação de nós faz com que não haja um crescimento contínuo no tempo total da fase de criação de nós. Outra evidência de como esse comportamento se deve aos eventos esporádicos é a pouca influência no padrão de crescimento dos tempos médios visto na Figura 4.

Avaliamos o resultado obtido como sendo bastante satisfatório, dado que demonstra que o sistema possui escalabilidade adequada, considerando as restrições impostas por sistemas externos. O crescimento e a variação observados nos tempos referentes à fase

de implantação podem ser explicados pelo crescimento na carga (e, portanto, no tempo de resposta) do Chef Server. O servidor em questão manteve-se com 100% de uso de CPU durante longos períodos, sugerindo que o desempenho pode ser significativamente melhorado através da distribuição e replicação do Chef Server. No entanto, como mencionado, o componente responsável pelo maior impacto negativo na escalabilidade é o serviço da Amazon. É possível que diferentes provedores ofereçam resultados melhores, dependendo de suas políticas de alocação de máquinas virtuais, que também podem levar em conta aspectos como economia de energia, distribuição de carga etc.

5. Conclusão

Neste artigo, apresentamos o CHOReOS Enactment Engine, um sistema de middleware disponibilizado como software livre que oferece serviços de plataforma para a implantação distribuída e automatizada de coreografias de serviços web em ambientes de nuvem. Esse sistema disponibiliza pontos extensíveis que ajudam a amenizar a falta de flexibilidade comum a soluções de Plataforma como Serviço (PaaS). Avaliamos a escalabilidade do tempo de implantação, que se mostrou satisfatória dentro da faixa de operação avaliada, uma vez que o tempo de implantação apenas dobrou para um aumento de 25 vezes no número de serviços implantados ao se utilizar sempre uma máquina virtual por serviço. Esse resultado é particularmente positivo quando comparado aos resultados do único artigo de nosso conhecimento que apresenta avaliações empíricas sobre a escalabilidade do processo de implantação [Quéma et al. 2004], no qual o tempo de implantação é aproximadamente proporcional ao aumento conjunto de carga e recursos.

Em trabalhos futuros utilizaremos diferentes ambientes de computação em nuvem em uma mesma encenação de coreografia. Analisar também o desempenho do middleware variando a quantidade de serviços em uma coreografia e realizando experimentos de maior porte, com centenas ou milhares de serviços. Finalmente, pretendemos adequar ainda mais o CHOReOS Enactment Engine aos requisitos de sistemas de grande escala, considerando especialmente a falha ou mau comportamento de sistemas externos, bastante comuns, como pudemos observar na interação com os serviços da Amazon durante nosso experimento. Um dos planos para isso é o uso de *caches* e *pools* de máquinas virtuais para amenizar o impacto do tempo de criação das máquinas virtuais no processo de encenação.

Agradecimentos

Esta pesquisa foi financiada pela Comissão Europeia (FP7/2007-2013) através do projeto CHOReOS (processo 257178) e pela HP através do projeto Baile. Agradecemos aos colegas envolvidos com a implementação de nosso middleware: Daniel Cukier, Carlos Eduardo do Santos, Felipe Pontes e Alfonso Diaz, e também à Nathália Demétrio pelo suporte ao tratamento estatístico dos dados.

Referências

Balter, R., Bellissard, L., Boyer, F., Riveill, M., and Vion-Dury, J.-Y. (1998). Architecturing and configuring distributed application with Olan. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, Middleware '98, pages 241–256. Springer-Verlag.

Barker, A., Walton, C. D., and Robertson, D. (2009). Choreographing Web Services. *IEEE Transactions on Services Computing*, 2(2):152–166.

Ben Hamida, A., Kon, F., Ansaldi Oliva, G., Santos, C., Lorré, J.-P., Autili, M., Angelis, G., Zarras, A., Georgantas, N., Issarny, V., and Bertolino, A. (2012). An integrated development and runtime environment for the future internet. In *The Future Internet*, volume 7281 of *Lecture Notes in Computer Science*, pages 81–92. Springer Berlin Heidelberg.

Dolstra, E., Bravenboer, M., and Visser, E. (2005). Service configuration management. In *Proceedings of the 12th international workshop on Software configuration management*, SCM '05, pages 83–98. ACM.

Humble, J. and Farley, D. (2011). *Continuous Delivery*. Addison-Wesly.

Law, D. (1998). Scalable means more than more: a unifying definition of simulation scalability. In *Proceedings of the I998 Winter Simulation Conference*, pages 781–788. IEEE.

Magee, J., Dulay, N., and Kramer, J. (1994). A constructive development environment for parallel and distributed programs. In *Proceedings of 2nd International Workshop on Configurable Distributed Systems, 1994*, pages 4–14.

Magee, J. and Kramer, J. (1996). Dynamic structure in software architectures. In *Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering*, SIGSOFT '96, pages 3–14. ACM.

Mell, P. and Grance, T. (2011). The NIST definition of cloud computing (draft). Disponível em http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Acessado em 2 de dezembro de 2012.

OMG (2011). Business process model and notation (BPMN), version 2.0. http://www.omg.org/spec/BPMN/2.0.

Quéma, V., Balter, R., Bellissard, L., Féliot, D., Freyssinet, A., and Lacourte, S. (2004). Asynchronous, hierarchical, and scalable deployment of component-based applications. In *Component Deployment*, volume 3083 of *Lecture Notes in Computer Science*, pages 50–64. Springer Berlin Heidelberg.

Steen, M., Pierre, G., and Voulgaris, S. (2012). Challenges in very large distributed systems. *Journal of Internet Services and Applications*, 3(1):59–66.

W3C (2004). Web services architecture. http://www.w3.org/TR/ws-arch/.

Watson, P., Fowler, C., Kubicek, C., Mukherjee, A., Colquhoun, J., Hewitt, M., and Parastatidis, S. (2006). Dynamically deploying web services on a grid using Dynasoar. In *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC 2006*, page 8.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.