Uma Abordagem Baseada em Eventos para Adaptação Automática de Aplicações para a Nuvem*

Michel Araújo Vasconcelos[1], **Davi Monteiro Barbosa**[2]
Paulo Henrique M. Maia[3], **Nabor C. Mendonça**[1]

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321, Edson Queiroz, CEP 60811-905 Fortaleza – CE

[2]Centro Universitário Estácio do Ceará
Rua Vicente Linhares, 308, CEP 60135-270, Fortaleza – CE

[3]Centro de Ciências e Tecnologia (CCT)
Universidade Estadual do Ceará (UECE)
Campus do Itaperi, Av. Paranjana, 1700, CEP 60740-903 Fortaleza – CE

{michel.vasconcelos, davimonteiro.ce}@gmail.com
pauloh.maia@uece.br, nabor@unifor.br

Abstract. *Cloud Computing is a new computing paradigm whose benefits (pay-per-use business model, low infrastructure overhead, high availability and scalability) are attracting the interest of many organizations. However, those companies can face several obstacles to adopt this new paradigm. One major difficulty concerns the migration of their legacy applications since they may require many code changes in order to be deployed in the cloud. These modifications often demand a great effort from the developers and are error prone. This work proposes a novel event-based approach to the automatic adaptation of legacy applications to the cloud that brings as main benefits a greater reusability level and integration with other existing software adaptation solutions.*

Resumo. *A computação em nuvem é um novo paradigma computacional cujos benefícios (como pagamento sob demanda, baixo investimento em infraestrutura física e alta escalabilidade de recursos) têm cada vez mais atraído o interesse do mundo corporativo. Apesar disso, muitas organizações encontram dificuldades em adotar esse novo paradigma. Uma dessas dificuldades é a necessidade de migrar suas aplicações legadas, que podem exigir muitas mudanças em seu código fonte para que possam ser implantadas no ambiente de nuvem. Tais mudanças geralmente demandam um esforço considerável por parte dos desenvolvedores e podem introduzir novos erros de implementação. Este trabalho propõe uma nova abordagem baseada em eventos para a adaptação automática de aplicações para a nuvem, que traz como benefícios principais um alto nível de reusabilidade e integração com outras tecnologias de adaptação de software existentes.*

Palavras-chave: Computação em nuvem, adaptação automática de aplicações, migração, evolução.

*Este trabalho é parcialmente financiado pela *Microsoft Research*, através de um *SEIF Award 2013*.

1. Introdução

A Computação em Nuvem é um novo paradigma computacional que está se disseminando por toda indústria de Tecnologia da Informação rapidamente [Leavitt 2009]. Apesar de seus benefícios, como pagamento sob demanda, baixo investimento em infraestrutura física e alta escalabilidade de recursos, muitas organizações estão tendo dificuldades para migrar seus sistemas de software existentes para provedores de nuvem, como Windows Azure e Amazon EC2. Tal migração é inerentemente complexa porque as decisões de migração devem levar em conta (e podem ser influenciadas por) vários fatores, muitas vezes conflitantes, tanto técnicos (por exemplo, esforço de adaptação, escalabilidade, segurança) quanto não técnicos (por exemplo, perda da governança sobre dados e aplicações, reputação do provedor, restrições legais) [Beserra et al. 2012, Khajeh-Hosseini et al. 2012, Tran et al. 2011].

Este trabalho foca nos desafios técnicos envolvidos no processo de adaptação de aplicações existentes para que elas possam ser efetivamente migradas para a nuvem. Há dois desafios-chave para realizar essa migração [Andrikopoulos et al. 2012, Frey e Hasselbring 2011b]. O primeiro consiste em garantir que a aplicação a ser migrada não viole nenhuma restrição de ambiente imposta pela plataforma de nuvem de destino [Frey e Hasselbring 2011b]. Por exemplo, alguns provedores de serviços de plataforma de desenvolvimento na nuvem (*Platform-as-a-Service* – PaaS), como Windows Azure e Google App Engine, limitam o comportamento dos serviços neles implantados ao executá-los em um *sandbox* que não permite ou restringe o uso de funções básicas do sistema operacional, como abrir conexões de sockets ou acessar o sistema de arquivos local. Portanto, migrar aplicações que usam esse tipo de função para esses provedores pode não ser possível a menos que a aplicação seja alterada para evitar essas restrições. Analogamente, em algumas nuvens que fornecem infraestrutura como serviço (*Infrastructure-as-a-Service* – IaaS), como o Amazon EC2, discos virtuais montados localmente em uma máquina virtual são transientes, o que significa que as aplicações que utilizam o sistema de arquivos local para armazenamento teriam que ser mudados para usar um outro mecanismo de persistência a fim de manter seu correto comportamento na nuvem.

O segundo desafio envolve adaptar a aplicação de forma que ela tire amplo proveito do ambiente de nuvem [Andrikopoulos et al. 2012]. Por exemplo, para reduzir a sobrecarga de comunicação, o mecanismo de interação original da aplicação poderia ser substituída por um serviço de mensagem mais robusto, fornecido nativamente pela plataforma de nuvem. Em outro exemplo, para alcançar uma maior escalabilidade, o banco de dados relacional da própria aplicação poderia ser substituída por um serviço de armazenamento do tipo NoSQL baseado na nuvem.

Para realizar a migração e abordar os dois aspectos supra mencionados, a aplicação em questão pode necessitar mudanças consideráveis em sua arquitetura e código-fonte. Isso requer a realização extensiva de testes na nuvem, para garantir que o comportamento original do sistema foi preservado e que nenhum erro foi introduzido durante o processo de migração. Embora existam estudos recentes sobre detecção automática de potenciais violações de restrição no código-fonte da aplicação a ser migrada para a nuvem [Frey e Hasselbring 2011b], as transformações de código necessárias para que a aplicação fique de acordo com e se beneficie do ambiente de nuvem de destino

não têm sido abordadas por pesquisas recentes ou apoiadas por ferramentas. Além disso, a falta de tais ferramentas torna a tarefa de migração consideravelmente mais complexa para o desenvolvedor e suscetível a erros, pois, para realizar as alterações necessárias no código fonte manualmente, o desenvolvedor precisaria de um profundo conhecimento tanto sobre a estrutura interna do sistema quanto sobre as APIs e restrições técnicas específicas da plataforma de nuvem de destino.

Este trabalho propõe uma nova abordagem para a adaptação automática de aplicações a fim de reduzir a complexidade de migração para a nuvem. Ela se baseia na identificação e captura de trechos de código da aplicação que devem ser alterados para funcionarem na nuvem (de acordo com os desafios mencionados anteriormente) em forma de eventos. Através de uma arquitetura *publish/subscribe* [Eugster et al. 2003], os eventos são publicados e consumidos por adaptadores desenvolvidos para plataformas de nuvem específicas. Por utilizar uma arquitetura desacoplada e extensível, a abordagem proposta traz como benefícios a independência de técnica de interceptação do código e de plataforma de nuvem de destino, o que implica em um alto nível de reusabilidade e integração com outras tecnologias de adaptação de código existentes.

O restante do artigo está organizado da seguinte forma: a seção 2 discute os principais trabalhos relacionados ao tema em questão, enquanto a seção 3 apresenta o modelo de arquitetura *publish/subscribe*, base da nossa abordagem. A seção 4 descreve um exemplo motivador mostrando problemas para migração de uma aplicação para a nuvem. A seção 5 detalha o funcionamento da abordagem proposta. Por fim, as considerações finais e trabalhos futuros são mostrados na seção 6.

2. Trabalhos Relacionados

A maioria dos estudos feitos na migração de aplicações legadas para nuvem propõe metodologias e processos que auxiliem na decisão de migrar ou não uma dada aplicação para uma dada plataforma de nuvem (e.g. [Beserra et al. 2012, Khajeh-Hosseini et al. 2012]). Este trabalho complementa esses processos já que foca em auxiliar o responsável pela migração a tratar as mudanças necessárias nas aplicações a serem migradas para a nuvem.

Recentemente, houve uma mudança no sentido de prover soluções técnicas para os desafios existentes na implantação de aplicações legadas em uma nuvem. Em [Andrikopoulos et al. 2012], os autores elencam desafios e possíveis soluções para migrar uma solução de software para a nova plataforma. Eles identificam categorias para se distribuir um sistema na nuvem que abrangem desde uma implantação simples, em uma nuvem do tipo IaaS, a uma migração completa fazendo uso de vários recursos da plataforma destino. Os autores descrevem uma aplicação através de camadas e as dispõem segundo os modelos possíveis de distribuição em uma nuvem. Para cada camada, eles definem questões básicas a serem resolvidas antes de realizar a migração. Por fim, eles propõem uma comparação entre as camadas lógicas da aplicação, as categorias possíveis de migração para uma nuvem, seus benefícios e desvantagens. Desse trabalho [Andrikopoulos et al. 2012] usamos os questionamentos e desafios listados como insumo para construir as adaptações adequadas, que podem ser usadas para atender a uma categoria específica de migração.

Em [Frey e Hasselbring 2011a], é proposta uma abordagem de migração para nu-

vem chamada CloudMIG. A abordagem define um modelo conceitual e extensível, no qual é possível especificar um ambiente de nuvem, suas restrições[1] e violações[2]. Esse trabalho também propõe um modo de avaliação do sistema candidato à migração e de sua conformidade em relação a uma nuvem específica. Enquanto o CloudMIG foca no processo para identificar restrições e migrar a aplicação, as modificações no sistema base devem ser feitas manualmente. As violações formam elementos de primeira classe responsáveis por delimitar que pontos devem ser ajustados na aplicação base. Já neste trabalho, as violações são substituídas por interceptadores e eventos. O interceptador atua na captura de eventos pré-definidos e os propaga para tratamento (ver seção 5). Usamos os modelos conceituais definidos em [Frey e Hasselbring 2011a] como guias na construção dos primeiros interceptadores (e.g., interceptar evento de escrita em disco em nuvens do tipo PaaS).

Em [Maia et al. 2007], é apresentado um modo de desenvolvimento que permite adaptar *threads* de uma aplicação através de aspectos e aplicá-la em um ambiente de GRID computacional. Esse modo se assemelha com as adaptações aqui propostas já que ambos usam técnicas de transformação de código não-intrusivas. A nossa proposta é mais abrangente pois concentra-se em outros serviços da nuvem além do paralelismo (e.g., armazenamento em disco, persistência de dados, troca de mensagens, etc.).

3. O Modelo de Arquitetura *Publish/Subscribe*

Publish/subscribe é um modelo de arquitetura utilizado em aplicações distribuídas de larga escala que provê uma comunicação desacoplada e flexível entre seus componentes [Eugster et al. 2003]. Ele se baseia em duas entidades principais, produtores e consumidores, que se comunicam de forma assíncrona através de mensagens que carregam dados, chamadas de eventos.

Os produtores (*publishers*) publicam eventos no sistema, enquanto os consumidores (*subscribers*) expressam seu interesse em receber certos eventos através de um mecanismo de subscrição. No momento em que um produtor publica um evento, ele é recebido por um gerenciador de eventos, que é um *middleware* orientado a mensagens denominado de *broker*, o qual sabe quais consumidores devem ser notificados do evento. Por ser uma comunicação assíncrona, tanto produtor quanto consumidor não precisam estar ativos ao mesmo tempo quando uma mensagem é enviada, pois ela é armazenada pelo *broker* durante o tempo necessário até que o consumidor possa recebê-la.

O modelo *publish/subscribe* pode ser classificado em três categorias, de acordo com a forma como os consumidores registram seu interesse por eventos[Eugster et al. 2003]. São elas:

- *baseada em tópico*: é o esquema mais antigo e estende a noção de canais, utilizados em comunicação em grupo, com métodos para classificar e caracterizar o conteúdo dos eventos. Participantes podem publicar eventos e subscrever para tópicos específicos identificados por palavras-chaves.
- *baseado em conteúdo*: essa categoria oferece maior expressividade à inscrição dos consumidores. Ao invés de se registrar em um tópico, um consumidor especi-

[1]Limitações impostas pelo ambiente de nuvem destino

[2]Possíveis ações do sistema que não podem ser executadas devido a alguma restrição da nuvem

fica filtros, baseados no conteúdo dos eventos gerados, que serão aplicados para identificar se um evento é de seu interesse.
- *baseado em tipo*: é uma extensão do modelo baseado em conteúdo onde o tipo dos eventos desejados pode ser especificado em tempo de compilação por meio da parametrização da interface do evento de interesse do consumidor.

4. Exemplo Motivador

Para exemplificar a problemática abordada neste artigo, considere uma aplicação que faz uso intensivo de operações de leitura e escrita em disco. Deseja-se que essa aplicação passe a salvar os arquivos na nuvem ao invés de no disco local. Uma possível solução seria fazer a migração completa da aplicação para a nuvem de destino. Porém, como discutido na seção 1, isso pode envolver uma série de restrições técnicas relativas ao provedor de nuvem escolhido. Além disso, o esforço para fazer essa migração pode ser muito grande e desnecessário, uma vez que não foi solicitado que a aplicação execute na nuvem, mas apenas que salve os arquivos nela. Dessa forma, uma nova solução seria adaptar trechos da aplicação para que, quando solicitado que um arquivo seja salvo, este seja salvo na nuvem e não localmente.

Apesar de eficiente, essa solução pode trazer alguns problemas: primeiramente, o código da aplicação deverá ser alterado, o que pode introduzir novos erros; segundo, caso existam vários pontos onde esse trecho de código se repita, o esforço para adaptação pode ser grande e, consequentemente, mais propício a erros; por fim, a modularidade da aplicação pode ficar comprometida, uma vez que código que não é de interesse funcional da aplicação (código de acesso à nuvem) está sendo introduzido e espalhado entre seus módulos.

Uma outra alternativa seria usar técnicas de interceptação de código, como aspectos [Kiczales et al. 1997], que permitem alterar o fluxo de execução em pontos específicos do código. Neste caso, o aspecto conteria o código de envio de arquivo para a nuvem de destino. Com isso, a aplicação continua executando normalmente, porém é adaptada dinamicamente ao atingir um dos pontos de interceptação (uma chamada de método, por exemplo). As vantagens dessa abordagem são a possibilidade de reuso do aspecto entre aplicações e a separação de interesses. Apesar de melhorar a modularidade, essa solução ainda obriga que código de migração seja específico para uma nuvem de destino, o que pode ser limitante caso seja necessário trabalhar com outro provedor de nuvem em algum momento.

A seguir, mostramos uma solução que permite um maior descoplamento entre as técnicas de interceptação do código da aplicação e as chamadas aos serviços de nuvem oferecidos pelo provedor.

5. Abordagem Proposta

A abordagem proposta tem como principal objetivo possibilitar que uma aplicação legada possa ser adaptada de forma não-intrusiva, ou seja, sem intervenção direta no seu código-fonte, para que essa possa executar em uma determinada plataforma de nuvem. Para tanto, utilizou-se uma arquitetura *publish/subscribe*, conforme mostrada na Figura 1, cujos principais componentes serão discutidos em mais detalhes a seguir.

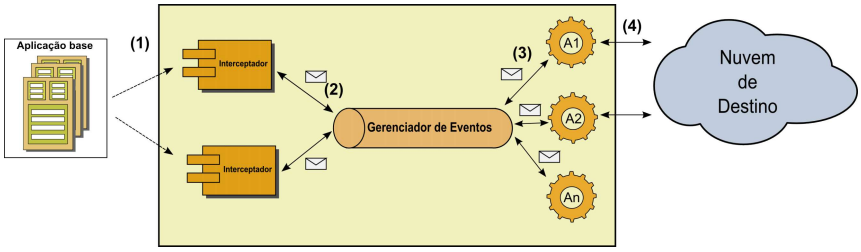

Figura 1. Arquitetura *publish/subscribe* utilizadas na abordagem proposta.

Os interceptadores de código (1) são responsáveis por identificar e capturar, na aplicação legada, os trechos de código que precisam ser adaptados para a plataforma de nuvem. A ideia consiste em interceptar a aplicação em pontos específicos para que seu fluxo de execução seja desviado para a nuvem sem modificar o seu código-fonte. As motivações para realizar uma interceptação podem ser a adição de uma funcionalidade, modificação dos resultados retornados de determinadas funções ou quando o código-fonte da aplicação utiliza uma biblioteca ou arcabouço de terceiros que não pode sofrer modificação.

O interceptador consegue atuar em níveis diferentes, conforme a tecnologia ou necessidade de adaptação do sistema base. Por exemplo, é possível criar um interceptador para atuar em nível de software em uma nuvem PaaS usando aspectos [Kiczales et al. 1997]. Já no caso de uma nuvem IaaS, poder-se-ia construir um interceptador que atue em nível de ambiente (sistema operacional) através de interposição funcional. As informações disponíveis para o interceptador são restritas conforme o seu respectivo nível de atuação. Enquanto no primeiro cenário o interceptador conhece o módulo do sistema que originou a interceptação, o último possui apenas as informações que são passadas para as funções nativas do sistema operacional. Para os nossos testes, usamos o AspectJ[3] para interceptação em nível de software e o EasyHook[4] para interceptação em nível de ambiente.

A criação de um interceptador em nível de software consiste em definir adequadamente quais serão os pontos de interceptação utilizados. Para tratarmos eventos de leitura e escrita em disco, por exemplo, o ponto de interceptação atuaria em classes do tipo *java.io.File, FileInputStream e FileOutputStream*. Já para um interceptador em nível de ambiente no sistema operacional Windows, por exemplo, é necessária a criação de uma DLL (*Dynamic-link library*) que irá conter as funções que serão interceptadas. Essa biblioteca deverá ser registrada, junto com todas suas dependências, dentro do *Global Assembly Cache* (GAC) do Windows, um repositório de código de máquina onde são armazenadas funções que são compartilhadas por vários processos. O registro da biblioteca será feito por um programa executável. Após realizar o registro, a biblioteca será injetada no processo alvo, que é identificado pelo seu PID (*Process ID*). Assim, caso não ocorram erros durante o registro e a injeção da biblioteca, o interceptador funcionará enquanto o processo alvo existir. Quando o processo alvo for encerrado, o interceptador será removido da biblioteca do GAC. A biblioteca que será injetada contém as funções que serão executadas no lugar das funções originais interceptadas. Para tanto, essas funções

[3]http://eclipse.org/aspectj
[4]http://easyhook.codeplex.com

devem ter a mesma assinatura.

Note que mesmo sendo necessário utilizar uma tecnologia de interceptação específica, os códigos de interceptação são independentes de aplicação e podem ser reutilizados. Por exemplo, várias aplicações que fazem uso de escrita em arquivo em disco podem se beneficiar de interceptações desse tipo de função. Os dados da interceptação são encapsulados em forma de evento e publicados no Gerenciador de Eventos (2).

O Gerenciador de Eventos é o elemento principal da abordagem. Ele é um *middleware* responsável por fazer a comunicação entre interceptores de código (produtores de eventos) e adaptadores de nuvem (consumidores), e é implementado utilizando um servidor de mensagens, como o Apache ActiveMQ[5]. O Gerenciador de Eventos permite que tanto os interceptores de código quanto os adaptadores de nuvem sejam implementados em diferentes tecnologias, linguagens e protocolos de comunicação. Com isso, um desenvolvedor pode utilizar as técnicas de interceptação e de adaptação da nuvem que lhes forem mais convenientes para uma aplicação, e outras diferentes para uma outra aplicação. Os adaptadores de nuvem declaram seu interesse em eventos específicos ao Gerenciador de Evventos. O Gerenciador de Eventos mantém uma estrutura do tipo *dicionário*, que permite identificar todos os adaptadores interessados em um evento. Ao receber uma mensagem, o Gerenciador de Eventos verifica a lista de adaptadores interessados e a entrega àqueles adaptadores (3).

Os adaptadores de nuvem processam os eventos gerados pelos interceptadores de código. Eles são específicos para a nuvem de destino e implementam o código que transforma um evento na ação que deve acontecer na nuvem (4). Por exemplo, considerando a situação de interceptação de escrita em arquivo em disco, um adaptador de nuvem que está interessado nesse tipo de evento implementa como o arquivo deve ser gravado em uma nuvem determinada. São exemplos de adaptadores de nuvem o JCloud[6] e LibCloud[7].

Note que a comunicação entre os componentes da arquitetura está ilustrada através de setas de duas pontas. Isso significa que, muitas vezes, o interceptador precisa receber um retorno do adaptador de nuvem sobre o *status* da operação. Isso pode ser feito através de *callback queues*. Por exemplo, a interceptação em nível de ambiente da função do sistema operacional de escrita em arquivo tem como retorno um dado do tipo *boolean*. Portanto, o adaptador de nuvem deve informar ao interceptador se o arquivo foi gravado com sucesso, para que a aplicação legada não tenha seu comportamento modificado.

6. Considerações Finais

Este trabalho apresentou uma abordagem para adaptação automática de aplicações legadas para a nuvem. Ela baseia-se em uma arquitetura *publish/subscribe* que permite que o desenvolvedor responsável pela migração possa escolher uma tecnologia de interceptação de código e uma plataforma de nuvem de destino específicas. Ela traz como benefícios a independência de tecnologia de interceptação e de nuvem e o aumento do grau de reusabilidade e integração com soluções já existentes.

Atualmente, um estudo de caso está em desenvolvimento para avaliar a abordagem

[5]http://activemq.apache.org
[6]http://jclouds.incubator.apache.org
[7]http://libcloud.apache.org

proposta. Nele, uma aplicação que permite o *download* dos arquivos de um site *web* é adaptado para que, ao invés de salvar os arquivos no disco local, envie e salve-os na nuvem. As interceptações estão sendo feitas tanto em nível de software (com AspectJ) quanto de sistema operacional (com EasyHook). Além disso, estão sendo utilizados dois adaptadores para nuvem: JCloud e libCloud.

Como trabalhos futuros, pretendemos avaliar a abordagem proposta através de mais estudos de caso e utilizando métricas que indiquem o custo-benefício da sua utilização, como tempo de resposta e desempenho. Também planejamos evoluir a abordagem na forma de um *framework*, através do qual o desenvolvedor poderá implementar suas próprias interceptações e seus próprios adaptadores de nuvem.

Referências

Andrikopoulos, V., Binz, T., Leymann, F., e Strauch, S. (2012). How to adapt applications for the cloud environment. *Computing*, pages 1–43.

Beserra, P., Camara, A., Ximenes, R., Albuquerque, A., e Mendonca, N. (2012). Cloudstep: A step-by-step decision process to support legacy application migration to the cloud. In *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the*, pages 7–16.

Eugster, P. T., Felber, P. A., Guerraoui, R., e Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131.

Frey, S. e Hasselbring, W. (2011a). The cloudmig approach: Model-based migration of software systems to cloud-optimized applications. *International Journal on Advances in Software*, 4(3 and 4):342–353.

Frey, S. e Hasselbring, W. (2011b). An extensible architecture for detecting violations of a cloud environment's constraints during legacy software system migration. In *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, pages 269–278.

Khajeh-Hosseini, A., Greenwood, D., Smith, J. W., e Sommerville, I. (2012). The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. *Software Practice & Experience*, 42(4):447–465.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J.-M., e Irwin, J. (1997). Aspect-oriented programming. In *Object-Oriented Programming (ECOOP), 1997. 11th European Conference on*, volume 1241, pages 220–242.

Leavitt, N. (2009). Is cloud computing really ready for prime time? *IEEE Computer*, 42:15–20.

Maia, M., Maia, P., Mendonca, N., e Andrade, R. (2007). An aspect-oriented programming model for bag-of-tasks grid applications. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 789–794.

Tran, V., Keung, J., Liu, A., e Fekete, A. (2011). Application migration to cloud: a taxonomy of critical factors. In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, SECLOUD '11, pages 22–28, New York, NY, USA. ACM.