Uma nuvem privada oportunista para execução de aplicações Bag-of-Tasks

Patricia Alanis[1], Abmar Barros[1], Francisco Brasileiro[1], Marcos Nóbrega[1]

[1]Laboratório de Sistemas Distribuídos (LSD)
Universidade Federal de Campina Grande (UFCG)
Caixa Postal 58.429-900 – Campina Grande – PB – Brazil

{patriciaam,abmar,marcosancj}@lsd.ufcg.edu.br, fubica@dsc.ufcg.edu.br

Abstract. *Cloud computing has emerged as a technology capable of providing elastic and flexible access to computational resources, thus meeting the needs of specific applications in a large-scale environment. However, in the typical deployment scenario of a private cloud – unlike the public cloud context, in which there is the Spot instance concept – there is no paradigm matching the needs of BoT applications, which do not require dedicated resources or QoS guarantees. In this paper we propose an opportunistic approach for the private cloud that enables the usage of idle resources in an existing computing infrastructure as a service, allowing the execution of BoT applications in a friendly and elastic fashion.*

Resumo. *A computação em nuvem surgiu como uma tecnologia capaz de prover acesso elástico e flexível a recursos computacionais, assim atendendo às necessidades específicas de aplicações em ambientes de grande escala. No entanto, no cenário típico de implantação de nuvens privadas – diferente do contexto de nuvens públicas, no qual existe o conceito de instâncias Spot – não há um paradigma que atenda à classe de aplicações BoT, que não requerem recursos dedicados ou garantias de QoS. Neste trabalho propomos uma abordagem de nuvem privada oportunista que permite o uso dos recursos ociosos de uma infraestrutura computacional preexistente como um serviço, possibilitando a execução de aplicações BoT de forma amigável e elástica.*

1. Introdução

Atualmente a Computação na Nuvem (Cloud Computing) é um dos principais paradigmas da computação, no qual recursos computacionais são oferecidos como serviços. O termo faz referência ao hardware, software e aplicações entregues como serviços através da Internet [Antonopoulos and Gillam 2010], ou de uma rede local, no caso de nuvens privadas. Neste modelo o principal objetivo é a entrega personalizada e flexível de infraestrutura computacional, software e aplicações como serviços. No modelo de nuvem pública, os usuários pagam apenas pelos recursos consumidos (o uso de espaço em disco, tempo de CPU, a transferência de dados). No modelo privado, busca-se uma melhor utilização dos recursos através da consolidação de servidores.

Os recursos na nuvem são oferecidos através de abstrações de serviços, que atualmente são divididos em três categorias principais: *Software como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infraestrutura como Serviço (IaaS)*. Considerando o modelo de IaaS, motivações distintas levam o usuário a utilizar nuvens públicas ou implantar uma nuvem privada. Na perspectiva de utilização de nuvem pública, o

usuário, ao invés de ter *datacenters* próprios, solicita máquinas, redes e armazenamento sob demanda a um provedor que fornece os recursos via instâncias virtuais, geralmente utilizando protocolos de comunicação REST ou SOAP [Bhardwaj et al. 2010].

Já a implantação de um IaaS privado típico, apesar de requerer a obtenção de *hardware* dedicado, possibilita: a consolidação de servidores, que faz com que diferentes serviços de diferentes demandas de QoS executem sobre uma abstração elástica e flexível de infraestrutura que condiz com suas necessidades, diminuindo desperdício ao aumentar a utilização; e o controle e a segurança providos por uma rede interna [Hudic and Weippl 2012].

O cenário de *deployment* típico de IaaS privado faz sentido para uma série de aplicações corporativas que têm demandas altas de QoS, como serviços que devem manter uma alta taxa de disponibilidade, ou aplicações com prazo de finalização. Porém existem classes de aplicações que ainda se beneficiariam de uma infraestrutura não dedicada de nuvem, que consequentemente entregaria uma QoS mais baixo, que é o caso das aplicações BoT (*Bag-of-Tasks*).

Aplicações BoT são aplicações paralelas cujas tarefas são independentes entre si. Apesar de sua simplicidade, aplicações BoT são usadas em uma variedade de cenários, como mineração de dados, varredura de parâmetros, simulações, processamento de imagens, entre outras [Cirne et al. 2003]. Por serem independentes, a falha de uma das tarefas não causa a falha da execução como um todo, assim essas aplicações são passíveis de serem executadas em uma infraestrutura com baixas garantias de QoS, como por exemplo, nos períodos ociosos dos recursos de uma infraestrutura [Litzkow et al. 1988].

No contexto de nuvens públicas, a Amazon propõe o conceito de instâncias Spot [Amazon 2013], que, entre as soluções de IaaS, é a que provê menores garantias de QoS. Neste paradigma, o usuário dá um lance no preço de instância por hora e utiliza a instância adquirida até que seu lance seja ultrapassado. Este paradigma de baixa QoS não encontra analogia no cenário típico de implantação de nuvem privada.

Além disso, os usuários finais, de uma forma geral, não estão acostumados com a interface da nuvem, que, a priori, se resume a uma coleção de instâncias com acesso SSH. Assim, para ter suas aplicações BoT rodando na nuvem, eles precisam de um *middleware* que distribua suas tarefas na nuvem de forma paralela e elástica.

Assim, este trabalho apresenta uma abordagem de nuvem privada oportunista, que explora os recursos computacionais ociosos disponíveis em uma infraestrutura, com o propósito de construir uma plataforma tecnológica capaz de fornecer suporte a aplicações BoT, que são mais adequadas a ambientes oportunistas e heterogêneos, de uma forma amigável, dinâmica e flexível a um baixo custo.

O restante do documento está organizado como segue. A Seção 2 apresenta uma abordagem de nuvem privada oportunista e um *broker* de nuvem como soluções para os problemas apresentados nesta seção. A Seção 3 descreve os passos necessários para a demonstração desta solução, enquanto a Seção 4 contém as nossas considerações finais.

2. Arquitetura da solução

Em face ao problemas expostos na seção anterior, este trabalho apresenta uma abordagem baseada em uma nuvem privada utilizando o software Eucalyptus, porém de

uma forma oportunista. Assim, é possível descobrir os recursos computacionais ociosos que pertencem a uma infraestrutura física local. Para os laboratórios, isto significa que é possível agregar o poder computacional de *desktops* – normalmente utilizados pelos pesquisadores, mas ociosos fora do horário de trabalho – em uma nuvem privada que independe de recursos dedicados.

Além disso, este trabalho propõe um *broker* de nuvem, que recebe como entrada aplicações BoT (*Bag-of-Tasks*) no formato JSON, e é responsável por iniciar instâncias na nuvem, escalonar as tarefas para as instâncias, transferir arquivos de entrada e saída e executar as tarefas remotamente.

Nesta seção são descritas a arquitetura do Eucalyptus, as mudanças necessárias para torná-lo uma plataforma oportunista, e o projeto de um *broker* de nuvem.

2.1. Arquitetura do Eucalyptus

Eucalyptus é uma plataforma de nuvem baseada em Linux capaz de criar nuvens privadas e híbridas dentro de uma infraestrutura de TI. A arquitetura do Eucalyptus é altamente modular, com componentes internos que consistem em serviços Web tornando-os fáceis de substituir e expandir. A flexibilidade do Eucalyptus lhe permite exportar uma variedade de APIs para usuários através de ferramentas de cliente (Euca2ools). Atualmente, exporta uma interface que é compatível com o EC2 da Amazon e serviços S3, o que permite aos usuários do Eucalyptus agrupar recursos de nuvem privadas e nuvens públicas para formar uma nuvem híbrida.

2.2. Funcionalidades do Eucalyptus

Eucalyptus foi desenvolvido para suportar a computação de alto desempenho (HPC) e pode ser instalado sem modificação em todas as principais distribuições do Linux, incluindo Ubuntu, RHEL, CentOS, openSUSE e Debian. Para a implementação, gestão e manutenção das máquinas virtuais, rede e armazenamento, o Eucalyptus disponibiliza uma variedade de recursos, tais como: administração de chaves SSH, gestão de imagens, gerenciamento baseado em Linux de maquinas virtuais, gerenciamento de endereços IP, gestão de grupos de segurança, de volumes e *snapshots*.

2.3. Componentes do Eucalyptus

A nuvem Eucalyptus contém cinco tipos de componentes. Segue uma descrição detalhada de cada um deles:

- Node Controller (NC): É responsável por executar ações sobre os recursos físicos que hospedam as VMs. Ele roda em cada um dos nós e gerencia o ciclo de vida das instâncias. Interage com o hipervisor e o *cluster controller*. Tem como principais funções controlar atividades da VM, incluindo a execução, monitoração e finalização das instâncias, gerência do ponto da rede virtual, coleta de dados relacionados à disponibilidade dos recursos e entrega de relatórios para o *cluster controller*.
- Cluster Controller (CC): É responsável pela gestão do conjunto de NCs de uma infra estrutura de nuvem. Coordena o fluxo de entrada de pedidos recebidos, monitorando as informações do estado de todas as instâncias dos NCs e decide para qual NC enviar as requisições de instâncias baseado na quantidade de recursos livres disponíveis e na descrição da requisição.

- Cloud Controller (CLC): É responsável pelo processamento de solicitações recebidas de usuários finais ou administradores. Ele pode tomar decisões de programação, processos de autenticação, monitora a disponibilidade de recursos em diversos componentes da infraestrutura, incluindo nós e controladores de cluster, realiza arbitragem de recursos e decide quais *clusters* são usados para fornecer recursos.
- Storage Controller (SC): Fornece um nível de armazenamento em rede que pode ser dinamicamente anexado pelas VMs.
- Walrus fornece um mecanismo para armazenamento persistente e controle de acesso de imagens de máquinas virtuais e dados do usuário.

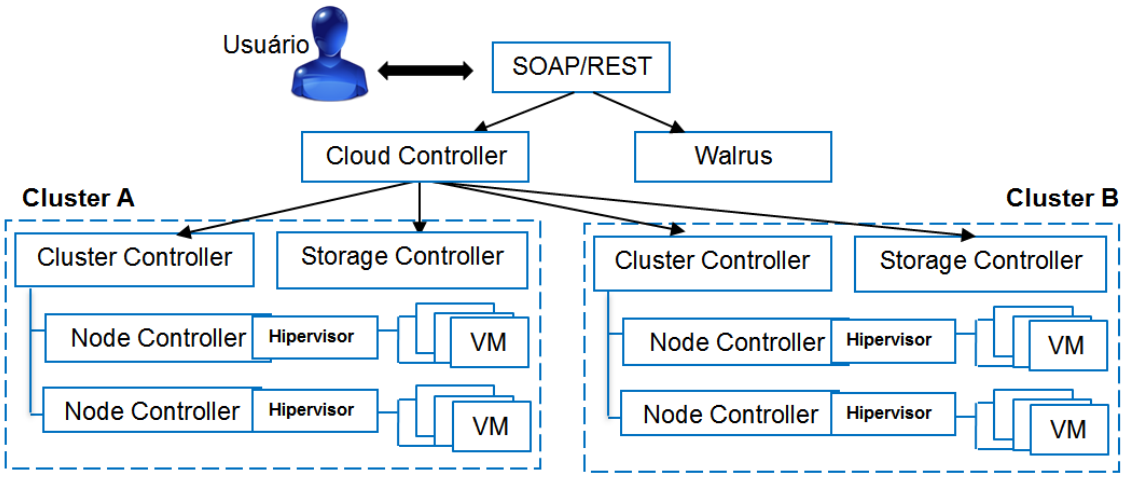

Figura 1. Arquitetura do Eucalyptus

2.4. Um *node controller* oportunista

Para adotar uma abordagem oportunista junto ao Eucalyptus, implementamos um novo *node controller*. A decisão de reimplementar este componente foi motivada pela possível heterogeneidade dos ambientes em que o NC será instalado.

Por utilizar XEN ou KVM como hipervisores, a implementação original do NC impõe uma série de requisitos de sistema para sua implantação. Esses requisitos podem ser de *software*, como versões específicas de kernel de Linux; ou de *hardware*, como extensões de virtualização (Intel VT, AMD-V). Numa infra-estrutura heterogênea, como o conjunto de *desktops* de um laboratório, esses requisitos nem sempre são alcançados., o que inviabiliza a implantação de uma nuvem privada usando a distribuição atual do Eucalyptus.

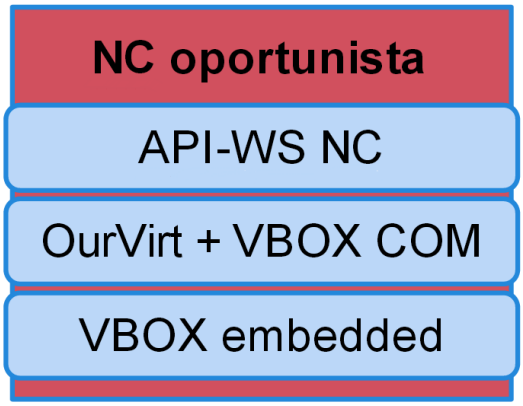

Figura 2. Arquitetura do *node controller* oportunista

Decidiu-se então reimplementar o NC em Java, uma vez que este dependeria somente da existência de uma JVM para ser executado, e utilizar VirtualBox como hipervisor, já que seus executáveis estão disponíveis para uma série de sistemas operacionais – incluindo Linux, Windows e MacOS – e não têm requisitos de *hardware* além da compatibilidade com x86.

A Figura 2 representa a arquitetura do NC oportunista apresentado neste trabalho. A camada de comunicação (API-WS NC), é implementada por meio de um *web-service*, que é forma padrão de comunicação do Eucalyptus. Para se comunicar com o hipervisor, usou-se a API OurVirt (github.com/OurGrid/OurVirt), que por sua vez utiliza a interface XPCOM do VirtualBox para gerenciar as máquinas virtuais.

Como também pode-se notar na figura 2, foi utilizada um versão *embedded* do VirtualBox em detrimento à versão distribuída no site oficial. Isso se deu pelo fato do hipervisor, e principalmente seu SDK, serem extremamentes sensíveis a mudanças inseridas entre versões. Dessa forma, era impraticável manter o NC funcional sem ter a garantia que ele se comunicaria com uma versão específica do VirtualBox.

Para determinar ociosidade, foram utilizadas estratégias diferentes para sistemas operacionais distintos. No Windows e MacOS é possível fazer chamadas de sistema que retornam quando aconteceu a última interação do usuário com o sistema. Para sistemas baseados em Linux, foi criado um script de inicialização do gerente de janelas X, que recupera periodicamente informações de atividade do próprio X e as publicam no espaço de arquivos do node controller.

2.5. Um *broker* de nuvem

A interface de nuvem, em nível de IaaS, não é amigável para o usuário final. Para ter uma aplicação executando na nuvem, o usuário precisa estimar a quantidade necessária de instâncias, criar a requisição de instâncias (que envolve escolher uma imagem, o tipo da instância, configuração de firewall, etc), monitorar os estados das instâncias, transferir arquivos e executar comandos usando conexões SSH, e repetir tarefas no caso de falha de conexão ou execução.

Para viabilizar a utilização da nuvem oportunista por pesquisadores de diferentes áreas, este trabalho apresenta um *broker* de nuvem, que é capaz de executar aplicações BoT na nuvem de forma transparente ao usuário.

A arquitetura do *broker* é constituída por dois componentes: o *broker daemon* e o *broker client*. O *broker daemon* é uma aplicação REST que executa em plano de fundo e é responsável por escalonar tarefas, requisitar instâncias, transferir arquivos e executar comandos. Já o *broker client* é a interface com usuário. Por meio deste o usuário pode submeter tarefas ao *broker* e recuperar a informação de seus estados. A Figura 3 mostra o exemplo de uma aplicação BoT que pode ser submetida ao *broker*.

```
{"name": "primes", "tasks": [
  {
    "init": [{"local": "primes.py", "remote": "primes.py"}],
    "remote": "python primes.py 2 1000 > output",
    "final": [{"local": "output", "remote": "output"}]
  },
  {
    "init": [{"local": "primes.py", "remote": "primes.py"}],
    "remote": "python primes.py 1001 2000 > output",
    "final": [{"local": "output", "remote": "output2"}]
  }
]}
```

Figura 3. Exemplo de aplicação BoT como entrada para o *broker* de nuvem

Ao receber um novo conjunto de tarefas, o *broker daemon* a armazena em memória e dispara uma execução de seu escalonador – um *workqueue* com reexecução no caso de falhas – para determinar quais instâncias alocar e quantas requisições o *broker* vai precisar fazer para executar as tarefas.

Para isso, o *broker* se comunica com o Eucalyptus através de um *webservice* compatível com o padrão Amazon, utilizando as credenciais do usuário, que estão armazenadas no arquivo de configurações do *broker*. Quando as instâncias estão disponíveis, o *broker* verifica se as portas SSH das instâncias estão abertas, transfere os arquivos de entrada via SCP, executa os comandos remotos via SSH e por fim recupera as saídas também via SCP.

Quando todas as saídas de um conjunto de tarefas são recuperadas, o *broker daemon* marca a execução como terminada, estado que pode ser constatado pelo usuário através do *broker client*.

3. Proposta de demonstração

A demonstração proposta assume a instalação e configuração dos componentes e dependências de Eucalyptus, e do *node controller* em máquinas *desktop*, para isso utilizaremos a infraestrutura do LSD (Laboratório de Sistemas Distribuídos) e notebooks *onsite*. Já o *broker* de nuvem estará executando na máquina dedicada à demonstração.

A proposta é de executar, por meio do *broker* de nuvem, uma aplicação BoT que utilize entrada em tempo real – como processamento de fotos capturadas durante o simpósio – usando a infraestrutura de nuvem oportunista instalada no LSD. Os resultados das execuções também serão exibidos em tempo real, por meio de um *script* que monitora os estados das execuções.

O código do *node controller* oportunista está disponível em https://github.com/OurGrid/opportunistic-nc, enquanto sua documentação pode ser encontrada em https://github.com/OurGrid/opportunistic-nc/wiki.

Já o *broker* de nuvem está disponível em https://github.com/OurGrid/cloudbroker, enquanto sua documentação se encontra em https://github.com/OurGrid/cloudbroker/wiki.

4. Conclusão

Empresas, instituições acadêmicas e científicas estão migrando para soluções em nuvem, por motivos que envolvem custo, flexibilidade e elasticidade. Porém nem todas as organizações conseguem custear o uso de nuvens públicas ou privadas. Este trabalho apresenta a abordagem de nuvem privada oportunista, que se apresenta como uma solução factível e útil para a classe de aplicações BoT.

As principais contribuições alcançadas neste trabalho foram o desenvolvimento de uma cloud privada oportunista capaz de monitorar os recursos de um ambiente computacional com o proposito de realocar esses recursos para aplicações Bag-of-Tasks; e a implementação de um *broker* de nuvem, que facilita o uso dessa infraestrutura por usuários de diversas áreas da ciência ou do mercado.

Com a realização deste trabalho, concluiu-se que o monitoramento e gerenciamento oportunista de uma infraestrutrua possibiliat um melhor aproveitamento dos recursos, maximizando e otimizando seu uso e, simultaneamente, amortizando os custos relacionados a sua aquisição. Além disso, contribui para a difusão do uso de ambientes oportunistas na nuvem que funcionem como suporte para o desenvolvimento de projetos corporativos, acadêmicos ou de investigação.

Referências

Antonopoulos, N. and Gillam, L. (2010). *Cloud Computing*. Springer London.

Bhardwaj, S., Jain, L. and Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology*, v. 2, n. 1, p. 60–63.

Hudic A. and Weippl E. (2012). Private Cloud Computing: Consolidation, Virtualization, and Service-Oriented Infrastructure. Computers & Security 31(4).

Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-neto E. and Medeiros R. (2003) Grid computing for bag of tasks applications. *Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and E-Government.*

Amazon Web Services. Amazon EC2 Spot Instances. (2013). Disponível em: <http://aws.amazon.com/pt/ec2/spot-instances/>. Acesso em: 01 mar. 2013.

M. Litzkow, M. Livny, and M. Mutka (1988). Condor: A Hunter of Idle Workstations. *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104-111, June 1988.