Análise de Mecanismos de Autoscaling para Aplicações Baseadas em Containers

Van Eyck Silva, Marcus Carvalho

Universidade Federal da Paraíba
Departamento de Ciências Exatas
Rio Tinto - Paraíba - Brasil

{van.silva, marcuswac}@dce.ufpb.br

Resumo. *A computação em nuvem vem crescendo muito ao longo dos anos e com ela algumas de suas principais características como virtualização e escalabilidade estão se tornando cada vez mais populares. As aplicações estão cada vez mais leves, robustas e fáceis de se adaptar a cargas de trabalho inconstantes, e isso se dá muito pelo o uso de containers e suas ferramentas de autoscaling. Este artigo tem como objetivo realizar de forma teórica uma análise de mecanismos de autoscaling em aplicações baseadas em containers, realizar um levantamento dos diferentes tipos de autoscaling, assim como ferramentas e serviços que fornecem esse tipo de método, além de reunir as principais características das ferramentas/serviços em busca de ajudar a apoiar a tomada de decisão de quais mecanismos um usuário deve escolher em determinada situação.*

Abstract. *Cloud computing has been growing significantly over the years and with it some of its key features like virtualization and scalability are becoming increasingly popular. Applications are becoming lighter, safer and easier to adapt with variable workloads, and this is due to the use of containers and their autoscaling tools. This paper aims to perform a theoretical analysis of autoscaling mechanisms in container-based applications, to perform a survey of the different types of autoscaling, as well as tools and services that provide this type of method, besides gathering the main features of the tools/services in order to help support the decision making of which mechanisms a user should choose in a given situation.*

1. Introdução

A Internet atualmente está mais do que presente em nosso dia a dia. Podemos não perceber, mas a sensação de estarmos conectados com toda uma gama de informações se dá através da *cloud computing* (computação em nuvem). Serviços que estamos habituados a acessar, como *Gmail, Dropbox, Netflix* e *YouTube* estão cada vez mais presentes na nossa vida, e com a rápida evolução da tecnologia ao longo dos anos, a facilidade e a velocidade com que as pessoas têm

---

Trabalho de Conclusão de Curso (TCC) na modalidade Artigo apresentado como parte dos pré-requisitos para a obtenção do título de Bacharel em Sistemas de Informação pelo curso de Bacharelado em Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE), Campus IV da Universidade Federal da Paraíba, sob a orientação do professor Marcus Williams Aquino de Carvalho.

acesso à Internet, acaba se tornando cada vez mais acessível a todos, permitindo uma utilização ainda maior de serviços desse tipo.

Para Vaquero et al. (2009), o conceito de computação em nuvem seria um grande conjunto de recursos virtualizados que podem ser acessados de forma fácil e de qualquer lugar. Esses recursos, que podem ser plataformas de desenvolvimento, hardware ou até mesmo serviços, podem ser reconfigurados de acordo com seu uso, tornando-se uma boa alternativa para otimização de tais recursos.

Uma das definições mais importantes sobre computação em nuvem e de grande relevância no meio acadêmico é da NIST (*National Institute of Standards and Technology*), que a define como uma infraestrutura que tem a capacidade de fornecer recursos computacionais de maneira rápida e ágil, podendo tanto aumentar como diminuir seus recursos à medida que for necessário. O usuário fica com a impressão de que os recursos consumidos por um serviço parecem ser infinitos, podendo ser obtidos a qualquer instante e em qualquer quantidade [Righi 2013].

Com base nesses conceitos, podemos então perceber algumas características fundamentais que a computação na nuvem nos oferece, tais como: pagamento apenas pelos recursos que utilizamos; fácil acesso através da internet; alta disponibilidade; virtualização; não ter que se preocupar com a infraestrutura do serviço; e elasticidade, que é a característica em que vamos nos aprofundar um pouco mais.

Mesmo com todos esses benefícios, ainda há algumas limitações como: segurança, pois não se sabe onde e quais são os dados que estão juntos das informações dos usuários; confiabilidade, que está relacionado à periodicidade em que ocorrem falhas no sistema, assim como o impacto que causaram; e disponibilidade, pois necessita de acesso à Internet sem interrupções para um uso adequado de um ambiente de computação em nuvem [Pedrosa e Nogueira 2011].

Outra tecnologia que vem crescendo no mundo da computação em nuvem são os containers. Um container[1] consiste em um tipo de virtualização em que apenas a aplicação fica empacotada dentro do sistema, com todos os seus atributos necessários para ser executada como bibliotecas e todos os seus binários, ao contrário das máquinas virtuais (VMs) que são mais robustas por empacotar também o sistema operacional. Os containers isolam a aplicação de seu ambiente, proporcionando que a aplicação que está no container sempre será executada da mesma maneira em qualquer ambiente.

Para Kukade e Kale (2015), o uso de containers para virtualizar microsserviços, que é uma tendência de arquitetura distribuída, é muito mais vantajoso do que utilizar VMs, principalmente pelo desempenho superior. Para poder gerenciar melhor os microsserviços em containers, se faz necessário um orquestrador de containers, ou seja, uma ferramenta que auxilia na criação, remoção, manutenção e principalmente na forma de realizar a escalabilidade desse tipo de aplicação [Filho 2016].

Realizar o provisionamento automático (*autoscaling*) de um recurso tem a ver com a técnica de aumentar/diminuir a capacidade deste recurso conforme a demanda de uso do serviço

[1] https://www.docker.com/resources/what-container

aumenta/diminui. Ou seja, quando há um pico na demanda, o serviço deve aumentar sua capacidade de recursos; por outro lado, se a demanda for reduzida, o serviço deve diminuir a quantidade dos mesmos. Esse equilíbrio é importante, pois o cliente só paga pelo que realmente está usando do provedor de nuvem e reduz significativamente o desperdício de recursos e consequentemente os seus custos. Há dois tipos de autoscaling mais comuns: vertical e horizontal. No autoscaling vertical, as VMs ou containers são redimensionados em tempo de execução; já no autoscaling horizontal, a quantidade de VMs ou containers é que é aumentada ou diminuída. Em ambos os casos, esses ajustes são feitos através das ferramentas e/ou serviços de orquestração de forma automática, com base em métricas e metas de desempenho. Há também o autoscaling em um cluster que, enquanto o vertical e horizontal redimensionam a capacidade ou quantidade de containers/VMs, no cluster autoscaling há o reajuste de servidores que vão executar todas as aplicações do cluster. Em geral, esses servidores hospedam containers, que por sua vez, são executados em máquinas virtuais.

Neste contexto de fornecer uma infraestrutura como serviço sob demanda (IaaS), este artigo tem como objetivo apresentar uma análise teórica de mecanismos de autoscaling em ambientes de nuvens computacionais, com foco em ferramentas e serviços de containers, bem como reunir as principais características dos mesmos a fim de auxiliar o usuário na escolha das ferramentas e serviços a serem usados de acordo com os mecanismos de autoscaling suportados.

O artigo está organizado da seguinte maneira. Na seção 2, a fundamentação teórica é apresentada trazendo conceitos básicos importantes. Na seção 3, é descrito como foi realizada a pesquisa. Na seção 4, são abordadas algumas ferramentas e provedores de serviços de orquestração de containers. Na seção 5, apresenta-se uma tabela em que se pode resumir as principais características de cada ferramenta. Na seção 6, é descrito alguns trabalhos que possuem diferenças e semelhanças com o presente artigo. Finalmente na seção 7, é apresentado a conclusão e também possíveis trabalhos futuros.

2. Fundamentação teórica

A elasticidade é uma das principais características da computação na nuvem, o que a leva a ser adotada em grande parte das aplicações atualmente. Na literatura, uma aplicação dita elástica tem a capacidade de reajustar seus recursos, tanto horizontalmente como verticalmente, de acordo com sua carga de trabalho que pode variar ao longo do tempo. Aplicações dessa natureza são geralmente baseadas em balanceadores de carga e em um conjunto de servidores com as mesmas características. Em um ambiente de cluster, os servidores seriam físicos, mas em ambientes de computação na nuvem os servidores podem ser hospedados em máquinas virtuais ou até mesmo em containers. Um sistema de autoscaling é encarregado de realizar as melhores escolhas sobre ações de reajuste de carga de trabalho de forma automática [Lorido-Botran et al. 2014].

Para Sedaghat et al. (2013), sistemas que podem realizar o gerenciamento de forma automática são fundamentais, principalmente quando se fala de infraestruturas na nuvem, muito pelo fato da complexidade e pela proporção da infraestrutura propriamente dita, como também pelo fato de que as ações que são tomadas no gerenciamento devem ser realizadas de maneira

muito rápida. O ponto principal de sistemas de autoscaling é a capacidade de realocar automaticamente a quantidade de recursos de uma determinada aplicação dependendo da sua carga de trabalho, principalmente utilizando seus recursos de maneira eficiente nos picos de carga e quando os recursos não estão sendo tão utilizados.

A provisão dinâmica de aplicações de nuvem de acordo com uma carga de trabalho inconstante pode ser realizada através de dois tipos de tecnologias de virtualização: baseado em VMs ou baseado em containers [Taherizadeh e Stankovski 2017]. Atualmente, há um grande crescimento na utilização da abordagem utilizando containers, tendo em vista seu melhor desempenho e praticidade.

Como podemos ver na Figura 1, containers são uma forma de empacotar apenas a aplicação, sem a necessidade de virtualizar toda uma VM. Utilizando esse princípio, é como se na construção de um container já fossem colocados todos os atributos necessários para a execução de uma dada aplicação, ou seja, todas as dependências como bibliotecas e binários já estariam prontas para serem utilizadas. Esse desacoplamento permite que aplicações baseadas em containers se tornem muito mais portáveis, leves e consistentes. Em uma virtualização com base em VMs, se virtualiza todo o sistema operacional (SO), o que pode causar desperdício de recursos. Já com containers, ao invés de virtualizar todo um SO, apenas é virtualizada a aplicação desejada; todos os outros recursos que o container desejar ele irá utilizar da máquina em que está sendo executado o container (também chamada de *host*) [Filho 2016].

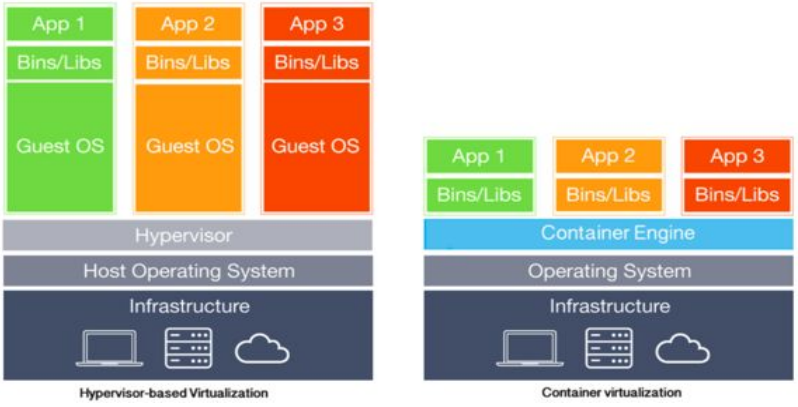

Figura 1 - Diferenças entre virtualização utilizando máquinas virtuais e containers

Fonte: http://brunoizidorio.com.br/blog/docker-o-que-e-docker/

Para auxiliar no gerenciamento de aplicações em containers, se faz necessário ferramentas de orquestração, principalmente quando se fala em grandes serviços que precisam manter o acordo de nível de serviço SLA (*Service Level Agreement*). Orquestradores de containers são utilizados para tornar mais fácil a gerência do ciclo de vida dos containers. Esse modelo de gerenciamento é quase sempre feito de forma automática, mas também pode ser manual. De maneira geral, a orquestração permite que se configure como os containers irão se comportar de acordo com a demanda de serviço. Tarefas como: disponibilidade e redundância de containers; adicionar ou remover containers de acordo com o uso; realizar balanceamento de

carga entre containers; promover ambiente de implantação adequado, são algumas ações que são previamente definidas de acordo com as políticas adotadas [Casalicchio 2016].

Orquestradores também são muito úteis no gerenciamento de clusters de servidores que hospedam containers, chamados de *nodes* (nós). Como esses *nodes* geralmente rodam em máquinas virtuais, podemos então realizar um outro tipo de autoscaling, o cluster autoscaling. Se tratando da ferramenta do Kubernetes[2], por exemplo, essa abordagem consiste no reajuste de forma automática no tamanho do cluster de acordo com o uso, ou seja, cria-se mais nós no cluster quando os pods (conjunto de containers que possuem características similares e devem ser executados juntos) existentes já não são suficientes para executar a aplicação, e os nós são excluídos quando estão ociosos por um grande período de tempo, fazendo com que seus pods, sejam transferidos para outros nós. Toda essa realocação faz com que o cliente pague apenas pelos recursos que utilizou.

Os orquestradores de containers mais usados no mercado atualmente são: Kubernetes, Docker Swarm e Apache Mesos (utilizando o Marathon). Da mesma forma, existem os serviços que também fazem esse tipo de orquestração como: Azure da Microsoft e o AWS da Amazon.

3. Metodologia de pesquisa

A pesquisa realizada neste trabalho foi de natureza exploratória, visando levantar as principais características dos mecanismos de autoscaling de aplicações baseadas em containers, visto que este tema ainda é pouco abordado no meio acadêmico.

Foi feito um levantamento de ferramentas e serviços de autoscaling baseados em containers. Nesse caso, visto que na literatura não há tantas informações detalhadas sobre tais, foi pesquisado em mecanismos de busca como o Google, em blogs, nos sites das ferramentas e nas próprias documentações. As ferramentas e serviços de orquestração de containers mais populares encontrados, que foram considerados nessa pesquisa, foram: Kubernetes, Docker Swarm, Apache Mesos, Microsoft Azure e Amazon AWS.

As principais características que foram descritas para cada uma delas foram: quais tipos de autoscaling suportam; quais recursos podem ser provisionados e quais métricas podem ser utilizadas para tomada de decisões. Essas características são descritas a seguir.

3.1. Tipos de autoscaling

No que se refere a aplicações baseadas em containers, há dois tipos mais comuns de realizar a provisão dinâmica de recursos: autoscaling horizontal e autoscaling vertical de aplicações.

No autoscaling horizontal, a quantidade de recursos pode ser aumentada e diminuída conforme a demanda. Nesse caso, entende-se por recursos um servidor, por exemplo; então, à medida que o número de requisições aumenta, utilizando a abordagem horizontal, mais réplicas dos servidores vão sendo adicionadas ao sistema. Por outro lado, na abordagem vertical, acontece o redimensionamento dos recursos; ou seja, ao invés de adicionar mais servidores, ela

[2] https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-autoscaler?hl=pt-BR

consiste em reajustar a capacidade dos servidores, aumentando ou diminuindo a memória RAM, a capacidade de CPU ou até a largura de banda da rede [Lorido-Botran et al. 2014].

Outro tipo que surgiu recentemente com a introdução dos containers foi o chamado *cluster autoscaling*. Enquanto as abordagens de autoscaling horizontal e vertical servem para aumentar/diminuir a quantidade ou a capacidade de containers ou VMs de uma determinada aplicação, no cluster autoscaling há o reajuste de servidores que vão executar todas as aplicações do cluster. Geralmente, esses servidores que hospedam containers são executados em máquinas virtuais, o que permite a sua criação e remoção de forma mais flexível.

Com relação a quem são os interessados em utilizar cada um desses tipos, observa-se basicamente dois tipos de usuários. Para realizar o autoscaling horizontal e vertical os interessados geralmente são os operadores da aplicação, visto que seu objetivo é escalar uma única aplicação; seu papel é de usuário da nuvem ou administrador da aplicação.

O outro tipo de interessado seria para realizar o cluster autoscaling, pois seu maior interesse seria escalar um cluster de containers com várias aplicações e vários usuários. Seu papel é de administrador do cluster de orquestração de container ou provedor da nuvem.

3.2. Recursos provisionados

A segunda característica relevante na utilização do autoscaling são os recursos que são provisionados. É o reajuste dos recursos que faz com que o cliente pague apenas pela quantidade que for usado [Jiang et al. 2013].

Além disso, permite detectar quais quantidades dos recursos de computação são necessários para executar uma aplicação com a garantia de que possa reagir bem com diferentes cargas de trabalho [Qu et al. 2018]. Os recursos mais comuns atualmente que podem ser monitorados para tomada de decisão de autoscaling ou serem reajustados são CPU e memória, usadas nas próprias instâncias da aplicação. Quando se trata de cluster, podem ser os nós do mesmo, e também pode depender da ferramenta que está sendo usada. No caso do Kubernetes por exemplo, os pods são os recursos que também podem ser redimensionados.

3.3. Métricas utilizadas

Para acontecer o redimensionamento dos recursos, é preciso definir como eles irão se comportar através da nossa terceira característica que são as métricas estabelecidas. Segundo Qu et al. (2018), há alguns níveis de hierarquia no qual podemos extrair as métricas. As métricas de baixo nível são retiradas do servidor ou da camada de virtualização (VMs/containers), que podem ser utilização de CPU, memória e de rede. Já as métricas de alto nível são mais voltadas à questão de desempenho da aplicação, alguns exemplos são: média de tempo de resposta, taxa de solicitação de aplicações, taxa de criação de sessões, taxa de transferência, requisições por segundo dentre outros. E também há as métricas híbridas que combinam os outros dois modelos mencionados anteriormente.

4. Ferramentas e provedores de serviços de orquestração de containers

Nesta seção, abordaremos algumas plataformas que são usadas para realizar a orquestração de containers, bem como alguns serviços que também podem ser utilizados, descrevendo como é feito o autoscaling em cada uma delas. É importante destacar que ambos podem ser utilizados de forma integrada, ou seja, pode-se ter uma plataforma (que seriam as ferramentas) que executam dentro de uma infraestrutura de um provedor de nuvem para gerenciar serviços.

4.1. Autoscaling em ferramentas de orquestração de containers

4.1.1. Kubernetes

O Kubernetes é uma ferramenta que auxilia no gerenciamento de containers, principalmente em clusters. Desenvolvida inicialmente pela Google, é uma ferramenta *open source* (de código aberto) capaz de realizar o redimensionamento do cluster e de aplicações de maneira automática de acordo com a demanda de serviço.

O autoscaling no Kubernetes se dá através de três maneiras:

- Horizontal Pod Autoscaling (HPA);
- Vertical Pod Autoscaling (VPA).
- Cluster Autoscaling;

Antes de explicar como o autoscaling funciona, é interessante apresentar alguns conceitos a respeito do Kubernetes para melhor compreensão. Como podemos ver na Figura 2, um cluster é formado por vários nós, que podem ser máquinas físicas ou virtuais. Um conjunto de nós dentro do cluster é chamado de *pool* de nós. *Pods* são conjuntos de containers que compartilham recursos entre si. Os pods são executados dentro dos nós. *ReplicationController/Deployment* (ou RC) é uma espécie de supervisor de pods que supervisiona vários pods em vários nós.

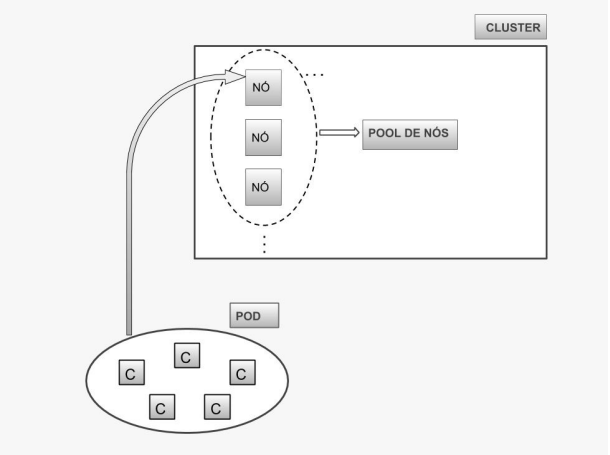

Figura 2 - Alguns termos da topologia do Kubernetes

No HPA[3], o que é reajustado são os pods. Então, se uma aplicação está executando um número determinado de réplicas e, em um dado momento, vem mais requisições do que a pool suporta, adiciona-se mais pods à aplicação. Esse dimensionamento é feito através do RC que ajusta o número de réplicas com base na utilização média da CPU, ou em outras métricas personalizadas.

É importante notar que, para funcionar o HPA de maneira satisfatória, os nós devem ter recursos suficientes para suportar o crescimento da demanda; é aí que entra o Cluster Autoscaling[4]. Os nós do cluster juntos é o que forma a capacidade total do cluster, então se a demanda exceder esse total, se faz necessário adicionar mais nós para que a disponibilidade da aplicação não seja comprometida. De forma sucinta, se os pods continuarem aumentando, é necessário aumentar o número de nós.

Assim como o HPA, o VPA[5] também tem seus pods dimensionados, mas ao invés de adicionar mais pods, o VPA tem a abordagem de alocar mais memória ou CPU aos pods já existentes. Esse ainda é um método novo, portanto ainda há algumas ressalvas a serem feitas como: para que as alterações na capacidade dos recursos sejam realizadas, os pods precisam ser reiniciados, o que pode causar indisponibilidade do serviço e instabilidade no desempenho da aplicação; o HPA e o VPA ainda não podem ser aplicados nos mesmos pods; o VPA não reage a todas as situações de falta de memória e o desempenho ainda não foi testado em um cluster de grande porte.

É interessante explanar que nas versões mais recentes do Kubernetes há uma possibilidade de usar métricas customizadas, principalmente no HPA. Geralmente são usados como métricas uso de CPU e memória para realizar o escalonamento, mas também pode ser feito essa configuração de outras maneiras como número de mensagens entregues, tamanho de fila, número de solicitações recebidas, consultas por segundo, conexões permanentes, dentre outras métricas da aplicação que podem ser definidas pelo usuário.

Para configurar métricas personalizadas no Kubernetes é preciso exportar essas métricas de uma camada que os engenheiros chamam de agregadores. O agregador mais usado é o Stackdriver[6]. Essa ferramenta[7] integra métricas, registros e eventos de infraestrutura a fim de tornar mais fácil o entendimento de como o serviço em questão está se comportando.

4.1.2. Docker Swarm

O Docker é uma plataforma de código aberto projetada para auxiliar na criação e manutenção de ambientes isolados. No mundo Docker existe o Docker Swarm, que é uma ferramenta que permite realizar a orquestração de containers de forma mais fácil e rápida, auxiliando também na criação de clusters. Utilizando o Swarm, pode-se adicionar hosts Docker ao cluster,

[3] https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#
[4] https://kubernetes.io/blog/2016/07/autoscaling-in-kubernetes/
[5] https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler#known-limitations-of-the-alpha-version
[6] https://cloud.google.com/kubernetes-engine/docs/tutorials/custom-metrics-autoscaling
[7] https://cloud.google.com/stackdriver/

simplificando o deploy dos containers e também abstraindo a gerência de quais nós estão disponíveis para executar determinada carga de trabalho [Filho 2016].

Mesmo trazendo vários benefícios para orquestração, o Docker Swarm apenas dá suporte para realizar o scaling de forma manual, ou seja, pode-se ajustar o número de réplicas a qualquer momento, mas não é possível realizar esta tarefa de forma automática de acordo com as variações de demanda. Se faz necessário, então, o uso de alguma ferramenta ou serviço externo que faça o ajuste de maneira automática.

Uma solução possível para realizar o autoscaling no Docker Swarm é através da ferramenta Orbiter[8], que foi desenvolvida para, a partir das métricas coletadas, criar *tasks* automatizadas no Docker Swarm[9]. As métricas têm que ser coletadas por outras ferramentas -- uma muita utilizada é a Telegraf da empresa InfluxData[10]. O Telegraf é um plugin de código aberto que coleta e reporta métricas, sendo as mais usadas o uso de CPU e memória, mas permitindo também usar métricas personalizadas.

Uma outra solução é usando a ferramenta Jelastic, que é mais utilizada para realizar o cluster autoscaling [Markova 2018]. Com o Jelastic, o autoscaling pode ser realizado das três maneiras (vertical, horizontal e cluster).

Por padrão, a política de cluster autoscaling vem da seguinte maneira:

- Adiciona-se um nó se o uso da RAM ou da CPU ultrapassar 70% por pelo menos 5 minutos;
- Remove-se um nó se o uso da RAM ou da CPU for menor que 40% por pelo menos 5 minutos.

Como a política pode ser pré-configurada, pode-se adotar *triggers*[11] em cima das métricas, e alterar as condições para que o autoscaling ocorra. Os *triggers* servem como um tipo de gatilho que monitoram as métricas de acordo com a política estabelecida, fazendo com que as mudanças aconteçam de forma automática. Existem cinco tipos de recursos que podem ser monitorados de acordo com seu uso: CPU, RAM, Rede, Disk I/O e Disk IOPS.

4.1.3. Apache Mesos

O Apache Mesos[12] é uma ferramenta que gerencia todo o ambiente de sistemas distribuídos como sendo um único recurso. Ele omite recursos como memória e CPU que estejam em máquinas físicas ou virtuais, possibilitando que sistemas sejam desenvolvidos com tolerância a falhas e de fácil escalabilidade.

[8] https://gianarb.it/blog/orbiter-docker-swarm-autoscaler
[9] No modo Docker Swarm, tasks são comandos que serão executados dentro dos containers por outros nós.
[10] https://docs.influxdata.com/telegraf/v1.8/introduction/getting-started/
[11] https://docs.jelastic.com/automatic-horizontal-scaling?utm_source=blog-docker-swarm
[12] http://mesos.apache.org/

O Mesos não é especificamente um orquestrador de containers, então, para realizar o autoscaling é necessário utilizá-lo em conjunto com a ferramenta Marathon. Há duas maneiras mais comuns de utilizar o Marathon com o Mesos para proporcionar o autoscaling, que são:

- Autoscaling com base em métricas de CPU e memória;
- Autoscaling com base na taxa de chegada de requisições.

Usando as métricas de CPU e memória, um serviço em Python (*marathon-autoscale.py*[13]) faz o autoscaling com base na utilização de recursos que vem do Mesos. De tempos em tempos, o *marathon-autoscale.py* coleta informações sobre a utilização de recursos para todas as tarefas que fazem parte do serviço estabelecido no Marathon. Quando o limite é atingido, a ferramenta ajusta o número de tarefas do serviço que está executando, o que consequentemente aumenta/diminui o números de instâncias. Quando se inicia a aplicação, uma série de parâmetros devem ser definidos para que se forme uma política adequada de autoscaling.

O segundo modo, que é utilizando requisições por segundo, consiste em usar o *marathon-lb-autoscale*[14], que se comunica com o Marathon através de APIs para escalar a aplicação e tentar manter o número desejado de requisições por segundo por instância de cada serviço. Alguns parâmetros que podem ser definidos são: limite de instâncias, máximo de instâncias, mínimo de instâncias, dentre outros.

4.2 Autoscaling em provedores de serviço de orquestração de containers

4.2.1. Microsoft Azure

O Microsoft Azure é uma plataforma que fornece a execução de serviços e aplicativos por meio da computação em nuvem, proporcionando alta disponibilidade, escalabilidade e permitindo que o usuário pague apenas pelos recursos que são utilizados. Essa alta escalabilidade se dá pelo fato de o Azure tornar mais fácil a realização do autoscaling. O usuário da nuvem não tem que se preocupar com toda uma infraestrutura computacional, basta apenas definir sua política de métricas de uma maneira rápida e intuitiva.

O Azure[15] suporta o autoscaling de aplicações baseadas em VMs de modo horizontal. Além disso, o Azure tem um serviço de orquestração de containers baseado no Kubernetes, chamado AKS[16] (Azure Kubernetes Service), que proporciona o horizontal pod autoscaling e o cluster autoscaling; porém, só é possível realizar o cluster autoscaling de forma manual.

Há vários recursos do Azure[17] que geram métricas que são monitoradas, as mais comuns sendo: uso de CPU, uso de memória, contagem de threads, tamanho de fila e uso de disco. Essas métricas são consumidas pelas regras que são definidas posteriormente.

---

[13] https://docs.mesosphere.com/1.7/usage/tutorials/autoscaling/cpu-memory/
[14] https://docs.mesosphere.com/1.7/usage/tutorials/autoscaling/requests-second/
[15] https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling
[16] https://docs.microsoft.com/pt-br/azure/aks/tutorial-kubernetes-scale
[17] https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-overview-autoscale

4.2.2. AWS

A Amazon Web Service[18] (AWS) é uma plataforma que fornece diversos serviços de computação em nuvem, prometendo alto desempenho, flexibilidade, escalabilidade e com modelo de pagamento baseado no uso. Dentre os vários serviços que a AWS oferece estão os de autoscaling.

4.2.2.1. Amazon EC2 Autoscaling

Utilizando o EC2 Autoscaling[19] o usuário da nuvem pode provisionar dinamicamente instâncias do Amazon EC2 de maneira horizontal de acordo com as condições definidas pelo usuário. O autoscaling é feito através de métricas definidas pelo próprio usuário. As três principais funções que auxiliam no provisionamento são [Barclay 2016]:

- Integridade de instâncias em execução: supervisiona a integridade de todas as instâncias em determinados períodos de tempo, garantindo o correto funcionamento das mesmas;
- Substituição automática de instâncias prejudicadas: caso seja detectado alguma falha, as instâncias podem ser substituídas automaticamente;
- Balanceamento de capacidade em zonas de disponibilidade: equilibra o lançamento de novas instâncias do EC2 entre as zonas de disponibilidade[20].

As métricas são coletadas através do CloudWatch[21] para o usuário ser capaz de estabelecer sua própria política. As métricas[22] podem ser visualizadas por grupo ou por instâncias. Exemplos de métricas por grupo: tamanho mínimo e máximo do grupo; capacidade desejada etc; Exemplos de métricas por instâncias: utilização de CPU, leituras de disco, operações de leitura de disco, entrada de rede etc.

4.2.2.2. Amazon Elastic Container Service (ECS)

O ECS[23] facilita o provisionamento de aplicações em containers do Docker na AWS, permitindo escala-los de forma mais rápida . Usando esse serviço não é necessário instalar nenhum software de orquestração. O autoscaling suportado é tanto o horizontal quanto o do cluster.

As métricas utilizadas pelo ECS[24] são:

---

[18] https://aws.amazon.com/pt/what-is-aws/

[19] https://aws.amazon.com/pt/ec2/autoscaling/?hp=tile&so-exp=below

[20] Os recursos no AWS são hospedados em todo o mundo e são divididos em regiões. Cada região também é dividida em zonas de disponibilidade.

[21] Serviço que monitora e gerencia dados do sistema.

[22] https://docs.aws.amazon.com/pt_br/autoscaling/ec2/userguide/as-instance-monitoring.html#as-view-group-metrics

[23] https://aws.amazon.com/pt/ecs/?hp=tile&so-exp=below

[24] https://docs.aws.amazon.com/pt_br/AmazonCloudWatch/latest/monitoring/ecs-metricscollected.html

- Reserva de CPU: Porcentagem da CPU que é reservada na execução de tarefas no cluster;
- Utilização de CPU: Porcentagem de CPU que é usado no cluster ou serviço;
- Reserva de Memória: Porcentagem de memória que é reservada na execução de tarefas no cluster;
- Utilização de Memória: Porcentagem de memória que é usada no cluster ou serviço;

4.2.2.3. Amazon Elastic Container Service for Kubernetes (EKS)

O EKS[25] facilita o gerenciamento, a implantação e o provisionamento de aplicações usando o Kubernetes no AWS. Como o Kubernetes já se integra ao serviço da Amazon, o modo como se configura o EKS é o mesmo do Kubernetes, então, o EKS dá suporte ao HPA e em cluster, e também utiliza o mesmo esquema de métricas.

5. Comparativo entre soluções de autoscaling

A Tabela 1 apresenta um resumo de ferramentas e provedores de serviços que possuem funcionalidades de autoscaling, distinguindo os tipos de autoscaling oferecidos e as métricas que podem ser usadas para guiar as decisões.

Tabela 1 - Comparação entre as ferramentas e provedores de serviços

Ferramenta/Serviço	Métricas			Tipo de autoscaling		
	CPU	Memória	Outras	Horizontal	Vertical	Cluster
Kubernetes	X	X	- Número de mensagens entregues; - Tamanho de fila; - Número de solicitações recebidas; - Consultas por segundo; - Conexões permanentes.	X	X	X
Docker Swarm/Orbiter	X	X		X		X
Docker Swarm/Jelastic	X	X	- Rede; - Disk I/O; - Disk IOPS.	X	X	X
Apache Mesos/Marathon	X	X	- Requisições por segundo.	X		

25 https://aws.amazon.com/pt/about-aws/whats-new/2018/08/amazon-eks-supports-horizontal-pod-autoscaling-with-custom-metric/

Microsoft Azure	X	X	- Tempo de resposta; - Comprimento de fila; - Contagem de threads.	X		
Microsoft Azure/AKS	X			X		
AWS/EC2 Autoscaling	X		- Leituras de disco; - Operações de leitura de disco; - Gravações de disco; - Operações de gravação de disco; - Entrada de rede; - Saída de rede;	X		
AWS/ECS	X	X	- Porcentagem de unidades de CPU que são reservadas executando tarefas no cluster; - A porcentagem de memória reservada, executando tarefas no cluster;	X		X
AWS/EKS	X	X		X		X

Com base no conceitos apresentados na seção anterior e com o auxílio da Tabela 1, podemos então enriquecer nosso conhecimento sobre cada ferramenta/serviço, tornando mais fácil a escolha de uma delas.

No que se trata das ferramentas, o Kubernetes é bastante utilizado em clusters e pode realizar o autoscaling dos três tipos. Além de suas particularidades de seus pods e sua recente alavancada na abordagem vertical, um ponto interessante são as métricas personalizadas que dão maior liberdade de configuração ao usuário/provedor da nuvem. O Docker Swarm, apesar de ser nativo do Docker, apenas dá suporte ao provisionamento de maneira manual tornando-se necessário outras ferramentas para automatizar esse processo. O Mesos junto com o Marathon, apesar de não ser um orquestrador de containers por natureza é bem conceituado, tanto em instâncias quanto em clusters. Uma métrica interessante é a taxa de chegada de requisições, o qual também o usuário/provedor na nuvem pode configurar seus parâmetros ao seu modo.

Já na parte de provedores de serviços, a Microsoft Azure se destaca pela facilidade de configuração em que o usuário tem, abstraindo detalhes de infraestrutura, e também pode fornecer métricas personalizadas através de uma API. Já os serviços da Amazon, além de fornecer a mesma facilidade de configuração da Azure, serviços como o ECS e EKS já são integrados no próprio serviços, simplificando ainda mais a infraestrutura.

6. Trabalhos relacionados

O artigo de Qu et al. (2018) descreve de forma muito ampla e com detalhes como funciona o autoscaling em aplicações web, principalmente baseado em VMs. O objetivo foi analisar os desafios mais relevantes que tornam o uso do autoscaling ainda um pouco complicado,

principalmente para pesquisadores que estão iniciando nessa área. Eles também analisaram os trabalhos de outros autores, identificando os pontos fortes e fracos nesta área. A diferença entre este artigo e o de Qu é que o dele é mais voltado para aplicações baseadas em VMs, enquanto o presente trabalho é mais voltado para aplicações baseadas em containers e tem o foco maior em destacar as características de cada ferramenta/serviço proposto que realiza o autoscaling.

O artigo de Taherizadeh e Stankovski (2017) sugere os principais desafios em realizar o autoscaling no contexto de *edge computing*, sendo relacionado ao presente artigo pois, utilizando containers seria possível montar ambientes de microsserviços de maneira rápida e eficiente para cenários de *edge computing*, junto com as ferramentas e serviços que os orquestram. Sabendo de suas principais características, é possível apoiar a tomada de decisão na escolha de qual utilizar em determinada situação. Desta forma, o presente artigo pode ser um complemento ao trabalho destes autores, para ajudar na tomada de decisões de ferramentas e serviços de orquestração de container no contexto de edge computing.

Outro trabalho de Taherizadeh e Stankovski (2018) propõe um novo método de autoscaling em que as métricas são definidas em tempo de execução, ao contrário do modelo tradicional em que são estáticas. As métricas são ajustadas conforme a carga de trabalho e em tempo de execução, o que proporciona o uso ainda mais eficiente dos recursos. A diferença em relação ao presente artigo é que se apresenta as características das ferramentas/serviços de uma maneira geral, não fixando-se apenas nas métricas.

Galante e Bona (2012) apresentam um amplo estudo sobre mecanismos que fornecem a elasticidade na computação em nuvem. A semelhança entre o presente artigo é o intuito de possibilitar um melhor entendimento sobre a área em questão. No entanto, a diferença entre o presente artigo é que nós não nos atentamos tanto na área da elasticidade, nos voltamos mais para as características das ferramentas, já o deles, além de também realizar um levantamento de ferramentas também elaboraram uma classificação baseado em algumas características da elasticidade.

7. Conclusão e trabalhos futuros

O autoscaling é uma técnica que visa o reajuste de recursos de maneira automática de acordo com a carga de trabalho da aplicação e com a mínima intervenção humana possível, fazendo com que a aplicação, que pode ser baseada em containers ou VMs, tenha maior tolerância a falhas, maior disponibilidade e melhor gerenciamento de recursos. Há várias ferramentas e serviços que proporcionam o uso dessa técnica. No entanto, reunir informações de como cada uma funciona, assim como quais são suas principais características, ainda era uma questão em aberto.

Neste artigo, reunimos as principais ferramentas/serviços buscando apoiar a tomada de decisão de quais mecanismos um usuário deve escolher para aplicações baseadas em containers, com base em algumas características que identificamos, como: tipo de autoscaling; as métricas usadas nas decisões e quais recursos que são provisionados. Este artigo serve como guia para que alguém que seja novo na área, possa ter uma ideia de como as ferramentas que elencamos realizam o autoscaling.

Um trabalho futuro interessante seria realizar uma avaliação de desempenho desses mecanismos e compará-los destacando seus pontos fortes e fracos.

Referências

Barclay, C. (2016). Fleet Management Made Easy with Auto Scaling. Disponível em <https://aws.amazon.com/pt/blogs/compute/fleet-management-made-easy-with-auto-scaling/>. Acessado em 01/10/2018.

Casalicchio, E. (2016). Autonomic Orchestration of Containers: Problem Definition and Research Challenges.

Filho, A., C., A. (2016). Estudo comparativo entre Docker Swarm e Kubernetes para orquestração de contêineres em arquiteturas de software com microserviços.

Galante, G e Bona, L., C., E. (2012). A survey on Cloud Computing Elasticity. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing* (pp. 263-270). IEEE Computer Society.

Jiang, J., Lu, J., Zhang, G., & Long, G. (2013, May). Optimal cloud resource auto-scaling for web applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on* (pp. 58-65). IEEE.

Kukade, P., P. e Kale, G. (2015). Auto-Scaling of Micro-Services Using Containerization.

Lorido-Botran, T., Miguel-Alonso, J., Lozano, J., A. (2014). A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments.

Markova, T. (2018). Docker Swarm Auto-Clustering and Scaling with PaaS Power Armor. Disponível em: <https://jelastic.com/blog/docker-swarm-auto-clustering-and-scaling-with-paas/>. Acessado em 17/08/2018.

Pedrosa, P. H., e Nogueira, T. (2011). Computação em nuvem. Disponível em <http://www.ic.unicamp.br/~ducatte/mo401/1s2011/T2/Artigos/G04-095352-120531-t2.pdf>. Acessado em 30/08/2018.

Qu, C., Calheiros, R. N., & Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, *51*(4), 73.

Righi, R., R. (2013). Elasticidade em cloud computing: conceito, estado da arte e novos desafios.

Sedaghat, M., Hernandez-Rodriguez, F., Elmroth, E. (2013). A virtual Machine Re-packing Approach to the Horizontal vs. Vertical Elasticity Trade-off for Cloud Autoscaling.

Taherizadeh, S. e Stankovski, V. (2017). Auto-scaling Applications in Edge Computing: Taxonomy and Challenges.

Taherizadeh, S., e Stankovski, V. (2018). Dynamic Multi-level Auto-scaling Rules for Containerized Applications. *The Computer Journal*.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2009). A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev.,39(1):50–55.