Wesley Rodrigues de Oliveira

Aplicações web elásticas em uma nuvem de computadores: um estudo de caso no provedor Amazon Web Services

São Luís - MA

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Oliveira, Wesley Rodrigues de.
Aplicações web elásticas em uma nuvem de computadores : um estudo de caso no provedor Amazon Web Services / Wesley Rodrigues de Oliveira. - 2019.
54 f.

Orientador(a): Mário Antonio Meireles Teixeira.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2019.

1. Aplicação Web. 2. Avaliação de Desempenho. 3. Computação em Nuvem. 4. Simulação. I. Teixeira, Mário Antonio Meireles. II. Título.

Wesley Rodrigues de Oliveira

Aplicações web elásticas em uma nuvem de computadores: um estudo de caso no provedor Amazon Web Services

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão como parte dos requisitos necessários para obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mário Antonio Meireles Teixeira

São Luís - MA

2019

Wesley Rodrigues de Oliveira

Aplicações web elásticas em uma nuvem de computadores: um estudo de caso no provedor Amazon Web Services

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão como parte dos requisitos necessários para obtenção do grau de bacharel em Ciência da Computação.

Trabalho aprovado em 10 de julho, São Luís - MA, 2019:

Prof. Dr. Mário Antonio Meireles Teixeira
Orientador
Universidade Federal do Maranhão

Prof. Dr. Davi Viana dos Santos
Examinador
Universidade Federal do Maranhão

Prof. Dr. Geraldo Braz Junior
Examinador
Universidade Federal do Maranhão

São Luís - MA
2019

Dedico este trabalho aos meus amigos e todos que contribuíram para o final desta jornada.

Agradecimentos

Primeiramente a Deus por ter me concedido sabedoria e força para concluir o curso.

Aos meus pais, tias e irmão, por sempre me incentivarem a estudar e nunca desistir, pelas repreensões e ensinamentos passados que contribuíram para tornar a pessoa que sou hoje.

Ao meu orientador, Mário, por me acompanhar desde o início do curso e dedicar tanto tempo.

Aos meu amigos Erik e PH pelas caronas de todos os dias. Aos meus amigos Eduardo e Anderson por sempre estarem presentes e me auxiliarem durante a graduação. Obrigado pelos roles gourmet após dias estressantes.

Aos meus Amigos Jackteles (Gabriel) e Zaynab por fazerem parte do desenvolvimento da ferramenta utilizada no trabalho e deixarem utilizá-la.

À Atlética Lorde por fazer sair da rotina de estudos, trazer momentos de diversão e entretenimento durante o curso além de fazer novas amizades.

A Carlos por me ajudar nos trabalhos, no estágio e por compartilhar histórias engraçadas.

A Pedro por sempre reiniciar o servidor quando era necessário e ter me ajudado na organização do trabalho.

As pessoas do grupo codebuilders que me acompanharam nesta longa trajetória.

Ao meu parceiro de laboratório Gabriel Rezende por ajudar na reta final do trabalho.

A equipe da Una-Sus por ter me concedido o estágio e pelo conhecimento adquirido.

A você que tirou um pouco do seu tempo para prestigiar este trabalho.

Quando tiver que escolher entre estar certo e ser gentil, escolha ser gentil. (Dr. Wayne W. Dyer; Extraordinário)

Resumo

Durante a rotina de acessar aplicações web acabamos nos deparando com sistemas lentos e que acabam não atendendo a requisição do usuário. De acordo com estudos o tempo de resposta do sistema influencia o usuário a querer continuar acessando o sistema. Uma das possíveis causas para essa lentidão é a sobrecarga ocasionada pelo acesso simultâneo de muitos dispositivos. Para solucionar este problema foi utilizado os recursos de balanceamento de carga e escalonamento automático oferecidos pela nuvem para que assim o sistema possa ter o mesmo desempenho independente do número de acessos simultâneos. Por meio de geração de carga sintética foi possível simular um conjunto de usuários entrando em uma aplicação web e monitorar o comportamento dela através do software JMeter e o próprio ambiente da nuvem. Foram desenvolvidos dois ambientes para teste: um fazendo uso dos recursos de balanceamento de carga e escalonamento automático, e outro não. Por meio das técnicas de coletada de dados e modelagem analítica foi possível analisar ambos os cenários e compará-los entre si de acordo com as métricas escolhidas, tais como, tempo de resposta, vazão, utilização da memória e taxa de erro.

Palavras-chaves: Computação em Nuvem. Avaliação de Desempenho. Simulação. Aplicação Web.

Abstract

During the routine of accessing web applications we are faced with slow systems that end up not meeting the user's request. According to studies the response time of the system influences the user to want to continue accessing the system. One of the possible causes for this slowness is the overload caused by the simultaneous access of many devices. To solve this problem it was used the resources of load balancing and automatic scheduling offered by the cloud so that the system can have the same performance independent of the number of simultaneous accesses. Through the generation of synthetic load it was possible to simulate a set of users entering a web application and monitor its behavior through the JMeter software and the cloud environment itself. Two environments were developed for testing: one making use of load balancing and automatic scaling resources, and the other not. Through data collection techniques and analytical modeling it was possible to analyze both scenarios and compare them with each other according to the metrics chosen, such as response time, flow rate, memory utilization and error rate.

Keywords: Cloud Computing. Performance Evaluation. Simulation. Web App.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

APP	Application
AZs	Availability Zones
AWS	Amazon Web Services
BD	Banco de Dados
CPU	Central Processor Unity
EC2	Elastic Compute Cloud
ELB	Elastic Load Balancer
GB	Gigabytes
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
MBPS	Megabits per second
MYSQL	My Structured Query Language
PaaS	Platform as a Service
PHP	Hypertext Preprocessor
RAM	Random Acess Memory
Saas	Software as a Service
TI	Tecnologia da Informação

Lista de símbolos

λ	Letra grega Lambda
μ	Letra grega Mu
$\prod$	Notação matemática de produtório
$\sum$	Notação matemática de Somatório

Sumário

1 Introdução

O uso de aplicações web e websites é frequente, levando em consideração os diversos meios de acesso à internet que temos. A web pode ser entendida como um conjunto de informações que são interligadas através de hipermídias (ligações através de textos, documentos, músicas e vídeos) espalhadas pela internet. Em seus primórdios, as páginas web possuíam apenas HTML de forma estática, sem interação com o usuário. Com a evolução da tecnologia, novas formas surgiram para representar tais páginas de maneira dinâmica, em tempo de execução, com scripts executando junto aos servidores web, e os dados começaram a ser armazenados em banco de dados (BD) ou outros meios.

Uma Aplicação Web é qualquer programa de computador que executa uma função específica utilizando o navegador como cliente. A aplicação pode ser tão simples quanto um quadro de mensagens ou um formulário de contato em um site, ou tão complexo quanto um processador de texto ou um aplicativo de jogos para vários jogadores que você pode baixar para o seu celular (NATIONS, 2018).

Como uma aplicação é acessada por qualquer navegador, a quantidade de usuários/ dispositivos que podem acessá-la é potencialmente ilimitada. Um dos pilares para um bom desempenho de uma Web App é a realização de uma análise do número e frequência de acesso dos seus usuários, bem como a utilização de recursos, por exemplo, banco de dados, banda larga e outros. No entanto, tal análise torna-se defasada ao longo do tempo, sendo necessário realizar uma nova análise e entrar em contato com o provedor de serviços e pedir um upgrade ou downgrade dos recursos contratados, variando de acordo com o número de acesso dos usuários. Provisionar novos recursos pode levar horas e até mesmo dias, com a possibilidade da aplicação ficar indisponível durante esse tempo.

Segundo Cardellini, Casalicchio e Yu (2002), a velocidade das redes de computadores cresce de maneira mais rápida que a capacidade de processamento dos servidores, tornando o lado do servidor web um possível gargalo para todo o sistema. Desta forma, aplicações que recebem muitas requisições simultaneamente podem torna-se lentas por certo período de tempo. Isto pode ser notado ao acessar site comerciais em data comemorativas ou em período de promoção. Este trabalho trata sobre essa problemática.

De acordo com Menasce e Almeida (2002), o tempo que o usuário leva para acessar o sistema influencia em fazê-lo desistir do acesso e, consequentemente, o proprietário do site acaba não gerando lucro. Um dos fatores que influencia sobre o tempo de resposta do sistema ou ocasiona erros são as configurações inadequadas para receber a quantidade incomum de usuários e tem, por consequência, a sobrecarga do sistema.

O funcionamento de uma aplicação web é do mesmo modo de uma fila e

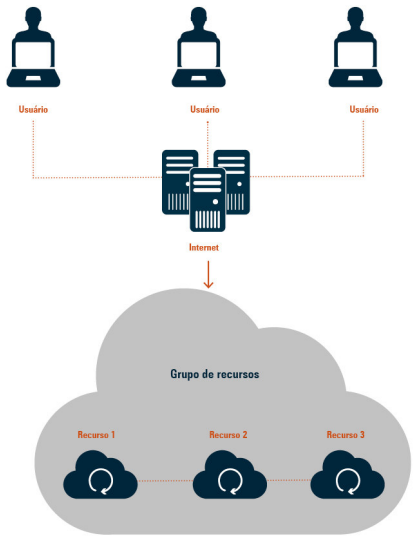
consequentemente uma maior demora no atendimento. Caso, a demanda de clientes seja maior que a capacidade de atendimento a fila existirá. Pelo de motivo de não conhecer a quantidade exata de clientes que o sistema deve atender, então o sistema deve estar preparado para ter o tamanho da fila o menor possível.

Considerando o contexto descrito, a computação em nuvem surgiu como uma solução para a demora no provisionamento, além de diversas outras vantagens, tais como: escalonamento automático, elasticidade, atendimento sob demanda, paga somente pelo que usar (pay per use) entre outras. Assim, é possível atender as requisições do usuário quase em tempo real.

De acordo com Parkhill (1966), a computação em nuvem é um modelo, o qual remete a década de 60, que demorou a ser posto em prática devido a largura de banda oferecida pelos provedores. Atualmente, permite acesso à rede de forma onipresente, conveniente e sob demanda a um conjunto compartilhado de recursos de computação configuráveis que podem ser rapidamente alocados e liberados com o mínimo de esforço de gerenciamento ou interação com o prestador de serviço (NIST, 2016).

Computação em nuvem é uma plataforma que dinamicamente provê, configura, reconfigura e libera servidores de acordo com as necessidades, e emprega grandes datacenters e serviços para serem utilizados via internet (IBM; VERAS, 2014). A partir dela, é possível que pequenas, médias e grandes empresas façam uso da mesma tecnologia com um preço acessível. Na figura 1 é possível representar como é a interação dos usuários com a nuvem de computadores.

Figura 1 – Acesso a plataforma via internet



Fonte: (SANTOS, 2018)

1.1 Objetivos

1.1.1 Objetivo geral

Avaliar/comparar um sistema computacional implantando numa nuvem de computadores e verificar o desempenho do mesmo com a utilização dos recursos de escalonamento automático e balanceamento e sem a utilização destes.

1.1.2 Objetivos específicos

- Utilizar métricas de avaliação de velocidade;
- Determinar os parâmetros que afetam o desempenho do sistema;
- Utilizar técnica de modelagem analítica e coleta de dados;
- Analisar, interpretar e apresentar os resultados.

1.2 Contribuições

Ao final deste trabalho são deixadas as seguintes contribuições:

- Disseminar a utilização da computação em nuvem para hospedar aplicações web;
- Realizar testes de desempenho na aplicação para verificar como se comporta o sistema.

1.3 Organização do trabalho

Além do Capítulo 1, este trabalho está organizado em mais 4 Capítulos, sendo que o Capítulo 2 apresenta os trabalhos relacionados na área de computação em nuvem e avaliação de desempenho e a fundamentação teórica utilizada como base para o desenvolvimento do trabalho proposto. Esse capítulo explica os fundamentos da computação em nuvem, avaliação de desempenho, teoria das filas, ferramentas usadas e a aplicação utilizada.

O Capítulo 3 apresenta os materiais utilizados durante o desenvolvimento deste trabalho, cenários desenvolvidos e a simulação feita. Os resultados colhidos após a aplicação dos testes são apresentados e discutidos no capitulo 4, e por fim, o Capítulo 5 contém as considerações finais.

2 Fundamentação teórica

Neste capítulo são apresentados os trabalhos relacionados, principais fundamentos da computação em nuvem, os motivos e vantagens de utilizá-la, a plataforma escolhida para o trabalho, princípios da avaliação de desempenho aplicadas, teoria das filas, além do meio para a geração de cargas na nuvem e a aplicação web utilizada.

2.1 Trabalho relacionados

Tendo como objetivo aproveitar o máximo de recursos oferecidos por recursos computacionais de maneira inteligente, novas metodologias são desenvolvidas para aproveitá-los. Neste Capítulo serão descritos alguns trabalhos com essas metodologias e apresentados os fatores que diferenciam este trabalho de seus semelhantes.

A tese apresentada por Souza (2004) em que desenvolveu o PSMS (Process Scheduling Monitoring System), tem como intuito oferecer uma infraestrutura de monitoração que pode ser usada por qualquer software de escalonamento, sendo a avaliação baseada em métricas escolhidas de acordo com o objetivo de escalonamento, utilizando de daemons para executar benchmarks.

No trabalho de Padilha e Padoin (2016) foi desenvolvida a plataforma CharmCharm++ para equilibrar a carga computacional entre os processadores com o propósito de reduzir o consumo de energia e diminuir o tempo de execução. Com a possibilidade de escolher qual tipo de balanceador poderia utilizar, através do benchemark lb-test foi possível gerar a carga computacional.

O trabalho desenvolvido por Costa (2009) fez uma análise experimental utilizando um cluster de alta disponibilidade para um software de missão crítica. Sistemas de missão crítica são sistemas que devem ter alto grau de disponibilidade e continuar a responder às requisições, mesmo em presença de falhas. Ele propôs uma solução para melhorar o desempenho e ganhar escalabilidade através do balanceamento de carga e aumentar a confiabilidade utilizando técnicas de alta disponibilidade.

O estudo de caso feito por Farias, Sousa e Nascimento (2016) utilizando o software JMeter para acessar documentos de um site a partir de um certo número de usuários, por um período de tempo, em um ambiente não controlado, verificando o comportamento do site através das métricas de vazão e tempo de resposta.

Este trabalho, diferente dos apresentados não propõe criar um escalonador de processos, nem criar uma plataforma para teste. Visa utilizar um escalonador já existente dentro da plataforma da nuvem AWS. Fazer uso do software JMeter para acessar uma

aplicação web, verificar o comportamento das métricas e tentar melhorar o desempenho a partir dos recursos que a nuvem oferece.

2.2 Computação em nuvem

Em NIST (2011) e Herbst (2013), define-se que computação em nuvem (em inglês, Cloud Computing) como um modelo para permitir acesso a serviços de modo onipresente, conveniente e sob demanda, constituídos em um conjunto compartilhado de recursos de computação configuráveis (p. ex., redes, servidores, armazenamento, aplicativos e serviços), que podem ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação com o prestador de serviços.

Em Brebner (2012) e Lorido-Botran (2014), destaca-se que Computação em Nuvem é um serviço que permite disponibilizar recursos computacionais sob demanda, seja de forma automática, seja de forma manual, independentemente de sua localização física. De fato, Computação em Nuvem permite melhor aproveitar os recursos disponíveis em uma infraestrutura computacional, diminuindo desperdício ou sua não utilização e, consequentemente, melhorando a qualidade do serviço fornecido, além de reduzir os custos (KUPFERMAN; SILVERMAN, 2016).

Vale frisar que a computação em nuvem é o armazenamento de informações, aplicações e demais serviços na área de TI sob uma plataforma online que pode ser acessada de qualquer lugar, a qualquer momento, sem a necessidade de instalar algum programa. O acesso é feito remotamente e o valor de custo é de acordo com o uso dos serviços.

A computação em nuvem revolucionou o mundo dos negócios, potencializou o modo de como as empresas oferecem seus serviços em que antes o ciclo tradicional era produção, venda e transferência de posse. Atualmente, os serviços passam a ser “alugados” enquanto o mesmo ainda é útil. Para exemplificar como os serviços estão indo parar nas “nuvens”, a empresa Microsoft transformou a tradicional plataforma Office em um dos serviços que mais cresce (Office 365) (VERAS, 2015). Um software muito comum em desktops usado de maneira ilegal (software pirata) passou a ser uma plataforma online com custo acessível para as empresas, desta forma, é possível usá-lo de maneira legal .

Apesar de haver uma certa incerteza durante os primórdios da Computação em nuvem por se tratar de algo desconhecido, graças a aproximação com a indústria e pesquisas feitas na área, ela pode evoluir de maneira exponencial desde o projeto da arquitetura de data center até agendamento e gerenciamento de recursos, virtualização de servidores e redes, armazenamento, frameworks de programação, gerenciamento de energia, preços, conectividade de serviços para segurança e privacidade (SOSINSKY, 2015).

NIST (2011) (do inglês, "National Institute of Standards and Technology") definiu cinco características essenciais que uma nuvem deve possuir, são elas:

1. Autoatendimento sob demanda: novos recursos são alocados à máquina do usuário sem a necessidade de interação humana, apenas pelo reconhecimento automático já pré-configurado pelo usuário inicialmente. Desta maneira, todo hardware e software é reconfigurado.

2. Amplo acesso a serviços de rede: os recursos computacionais são acessados através da internet, podendo ser acessado de qualquer lugar, através de um navegador simples. Não há necessidade de escrever códigos para manipular alguma operação, apenas alguns cliques e tudo está pronto.

3. Pool de recursos: os recursos computacionais (físicos ou virtuais) do provedor são divididos em pools para que possam atender a múltiplos usuários simultaneamente.

4. Elasticidade rápida: os recursos podem ser alocados de maneira rápida, precisa e elástica, as vezes até de forma automática e liberados da mesma maneira. Para o usuário, os recursos parecem ser infinitos em qualquer quantidade e em qualquer momento.

5. Serviço medido: esse tipo de sistema tem o controle sobre todos os recursos usados, bem como sua quantidade, por meio de uma funcionalidade de medição. É possível controlar a largura de banda, processamento e a capacidade de armazenamento.

   Vale ressaltar que nem toda nuvem possui todas as características citadas, pois elas se diferem em relação ao tipo. Para exemplificar, uma nuvem privada não possui a característica de amplo acesso, pois não é interessante permitir que pessoas de ambiente externo tenham acesso, é perigoso. A definição de NIST não é mais tão precisa quanto antes, já que as nuvens evoluíram desde a sua definição e englobam muito mais características.

2.2.1 Modelos de nuvem

Ainda segundo NIST, existem quatro modelos de nuvem que são implantados, cada um com um propósito diferente. São estes:

- Nuvem pública: seu uso é disponível para uso público, ou seja, qualquer pessoa pode contratar o serviço. Quem disponibiliza geralmente é proprietário de uma organização que vende serviços em nuvem.

- Nuvem privada: seu uso é específico para uma organização e o acesso é restrito a poucas pessoas, não necessariamente a nuvem deve estar localizada dentro da

organização, ela pode estar externa e o acesso remoto a ela é restrito à pessoas autorizadas.

- Nuvem híbrida: pode-se entender como uma combinação das das nuvens anteriores (pública e privada) em que o hardware é compartilhado mas com acesso restrito a parte privada. Geralmente uma nuvem híbrida é uma nuvem privada que não utiliza todos os recursos e abre espaço para a pública.

- Nuvem comunitária: foi projetada para atender uma função ou interesse em comum. É compartilhada por várias organizações que possuem preocupações comuns, como: missão, valor, segurança, políticas e assim por diante. A nuvem pode ser gerenciada pela(s) organização(ões) participantes ou por terceiros.

2.2.2 Modelos de serviço

Existem diversos provedores de nuvem, mas nem todos possuem os mesmos serviços, esse é um dos fatores na hora de levar em consideração qual provedor utilizar. Existem diversos tipos de serviços encontrados na literatura com a seguinte forma:

XaaS - '<something> as a service' ou, alguma coisa como serviço. No entanto, existem três que são universalmente aceitos, são estes:

- Saas - Software como serviço: são serviços que já estamos acostumados a utilizar, como já citado o office 365, o dropbox e o gmail. São serviços que basta fazer o login para acessar ou pagar pelo aluguel em alguns casos.

- IaaS - Infraestrutura como serviço: são serviços na parte de infraestrutura da rede em relação, como cabo, energia, armazenamento, sistema operacional, servidores virtuais e outros. O provedor trata de todas essas questões e você apenas gerencia.

- PaaS - Plataforma como serviço: oferece uma plataforma como serviço para facilitar o desenvolvimento de aplicações, por exemplo, o Amazon Beanstalk. Pode-se dizer que este é a combinação do Saas e IaaS, pois fornece um ambiente de desenvolvimento específico para desenvolver a aplicação.

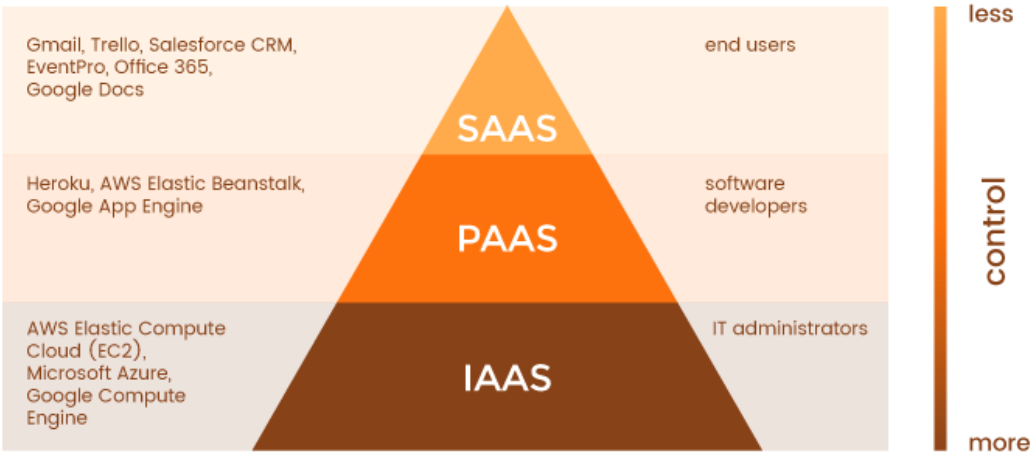
Abaixo na figura 2 é possível ilustrar como está a hierarquia do serviços da nuvem e grau de controle que tem em determinada camada.

2.2.3 Motivos para utilizar

Abaixo são destacadas algumas razões do porquê fazer uso da computação em nuvem segundo (O que..., 2015):

[1] Disponível em: https://rubygarage.org/blog/iaas-vs-paas-vs-saas. Acesso em: 20 jun. 2019

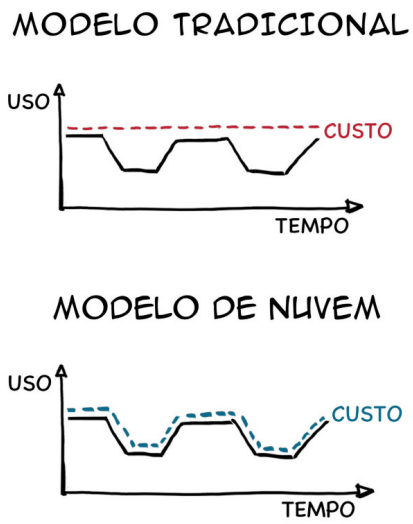
Figura 2 – Serviços oferecidos pela nuvem



Fonte: Página do rubygarage[1]

1. Custo:semelhante ao sistema de energia elétrica e água, o custo é proporcional ao consumo. Caso o sistema precise ficar online apenas durante o horário comercial, basta programá-lo para ligar e desligar em determinado horário. O número de acesso à aplicação é inferior ao esperado? basta reduzir as configurações da máquina. Este tipo de modelo é conhecido no mundo computacional como "pay per use" traduzido como pagar somente pelo o que usar. Na figura 3 é possível analisar como o custo é proporcional ao tempo e uso.

Figura 3 – Comparação do custo da nuvem com servidor tradicional



Fonte: (O que..., 2015)

2. Agilidade: o tempo para alocar ou desalocar recursos na máquina, tais como: memória RAM e disco rígido, são apenas alguns minutos sem a necessidade de interromper o

funcionamento do sistema.

3. Flexibilidade: se houver necessidade de alterar ou interromper algum recurso ou até mesmo desligar a máquina, basta alguns cliques e tudo é alterado.

4. Alta disponibilidade: para manter um sistema em alta disponibilidade é necessário que haja uma cópia do mesmo, caso o principal falhe, o secundário deve assumir imediatamente. No entanto, manter dois sistemas operantes idênticos gera um custo maior, pois apenas um deles estará funcionando de fato. Na nuvem é possível deixar o sistema secundário operante em formato de custo reduzido, como se estivesse em standy by, e alocado o mais rápido possível quando o principal falhe.

5. Inovação: favorece a criação de novos negócios ou a migração de negócios tradicionais, pois elimina os elevados custos iniciais para montar uma infraestrutura computacional de ponta, além de reduzir os riscos, caso a criação do negócio não progrida.

2.2.4 Vantagens e dificuldades em implantar esta tecnologia

Ainda segundo O que... (2015) são apresentados algumas vantagens e dificuldades em utilizar esta tecnologia.

Vantagens:

- Redução de custo para manutenção, eletricidade e segurança;
- Maior rapidez para colocar uma aplicação no ar;
- Possibilidade de realizar testes iniciais e caso haja algo incomum pode ser alterado rapidamente;
- Acesso a ferramenta para rastreamento e auditoria de sistemas.

Por outro lado existem ainda algumas dificuldades dentre elas são:

- Resistência da equipe interna, pois alega que este tipo de serviço aumenta a complexidade do trabalho;
- Resistência dos gestores de TI, que temem ficar obsoletos e perderem suas funções;
- Aspectos legais e de segurança em relação dados que serão salvos;
- Reações negativas e céticas ao termo Computação em Nuvem.

2.2.5 Plataforma Amazon Web Services (AWS)

A AWS é uma nuvem que presta serviços de TI para empresas desde 2006.

Segundo a AWS (2019), sua plataforma de nuvem é a mais utilizada e abrangente no mundo, contando com mais de 165 serviços completos de datacenters em todo o mundo. Desde pequenos clientes e startups até órgãos governamentais utilizam os serviços oferecidos para diminuir os custos e aumentar a agilidade.

Possui diversos data centers espalhados por todo o mundo para que o usuário possa utilizar o mais próximo de sua localidade. Esses são localizados no Brasil, Europa, Estados Unidos, Austrália, Japão e Cingapura. Cada data center é instalado em uma zona de disponibilidade chamada de Availability Zones (AZs), em nosso país, possuímos a zona mais próxima em São Paulo. Ao todo, a AWS opera em 66 zonas de disponibilidade e em 21 regiões demográficas, conforme mostra a figura 4.

Figura 4 – Zonas e regiões da AWS

Fonte: (AWS, 2019)

2.2.6 Amazon Elastic Compute Cloud (EC2)

O EC2 é um dos serviços mais utilizados da AWS, a partir dele é possível criar instâncias de servidores virtuais na nuvem com variadas configurações de sistema operacional, assim como diferentes de processador, memória e armazenamento de disco (LECHETA, 2014).

É basicamente uma hospedagem em que o usuário tem acesso e pode configurar conforme a quantidade e configuração necessários. A partir dela é possível instalar servidor web, banco de dados, servidor de aplicação, entre outros, e colocar uma aplicação online.

2.2.7 CloudWatch

O CloudWatch é um serviço oferecido pela nuvem da AWS que permite monitorar recursos e aplicativos utilizados. Através dele é possível monitorar instâncias EC2, banco de dados e outros recursos.

Utilizando a instância como exemplo, é possível monitorar métricas como utilização da CPU, memória e quantidade de operações de disco durante determinado período de tempo. Além disso, é possível definir alarmes para avisar ao administrador da nuvem caso determinada situação estabelecida aconteça, por exemplo, caso a utilização da CPU ultrapasse 50% o administrador deve receber um e-mail alertando.

2.2.8 Elastic Load Balancer (ELB)

Esta técnica é utilizada para aumentar a disponibilidade de serviços oferecidos. Os balanceadores de cargas são softwares inteligentes que distribuem a carga de requisições entre os vários servidores (EC2).

Significa que a partir desse serviço oferecido, é possível distribuir as cargas entre duas ou mais instâncias e essas podem estar localizadas em zonas distintas. Isso é importante para que aplicações com muitas requisições possam manter o mesmo desempenho independente da quantidade de requisições recebidas.

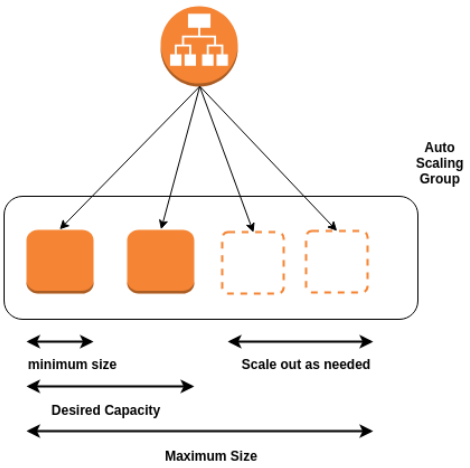
A utilização deste recurso sozinho é favorável se conhecer a quantidade de requisições que irá receber, mas isso dificilmente acontece, por isso ele deve ser utilizado com o recurso apresentado a seguir.

2.2.9 Escalonamento automático do EC2 (Auto Scaling)

Este é o recurso que torna uma aplicação web normal em uma aplicação web elástica. A partir dele é possível realizar o escalonamento horizontal, isto é, adicionar cópias de instâncias (EC2) idênticas a que já existe. Este recurso deve ser utilizado junto com o ELB, pois combinados terão um melhor desempenho.

O auto scaling permite automatizar a tarefa de adicionar e remover instâncias EC2 dentro do balanceador de carga (LECHETA, 2014). Ao configurá-lo junto ao ELB, definem-se as regras utilizando o cloudwatch para tomar decisões, tais como, a quantidade de instâncias mínima e máxima e quando adicionar ou remover instâncias, conforme a política estabelecida. Desta forma, alocando novas instâncias de acordo com a demanda, espera-se que o desempenho possa ser mantido. A figura 5 mostra o funcionamento.

Figura 5 – Funcionamento do autoscaling



Fonte: (AWS, 2019)

2.3 Avaliação de desempenho

Jain (1991) define Avaliação de Desempenho como uma obra de arte bem sucedida por ela não ser produzida mecanicamente ,e sim, de maneira abstrata, pois cada avaliação requer um conhecimento íntimo do sistema para ser modelado e uma seleção cuidadosa da metodologia, carga de trabalho e ferramentas. A maioria dos problemas apresentados pelo usuário final são expressos como um sentimento abstrato de áspero esboço. Definir o problema real e converter em um formulário no qual são estabelecidos as ferramentas e técnicas a serem utilizadas é papel do analista.

Consiste em analisar o sistema de modo que ele funcione da forma mais eficaz possível, obtendo a máxima eficiência dos recursos com o mínimo custo e tempo de resposta. Para tal avaliação, é necessário obter informações do próprio sistema com a utilização de técnicas de aferição. Obtido as informações, é possível abstrair as características mais relevantes e modelar o sistema da maneira mais coerente e assim, tentar resolver o problema de desempenho.

Avaliar um sistema é interessante para qualquer tipo de empreendimento, pois previne futuras imprecisões além de tornar o sistema mais confiável e econômico (MENASCE; ALMEIDA, 2002).

2.3.1 Planejamento do experimento

O ponto chave para uma boa avaliação do sistema é planejá-lo. A partir dele são escolhidos os dados para serem analisados e a quantidade que serão coletados durante o experimento. Tem como objetivo obter o máximo de informações com um número mínimo

de experimentos, analisar os efeitos que cada fator causou e determinar o grau de significado no resultado. Para isso é importante definir alguns conceitos.

- Variável de resposta (métricas): valor de saída do experimento;
- Fatores: variável que afeta diretamente as variáveis de resposta e pode admitir alternativas distintas;
- Níveis: valores que um determinado fator pode assumir;
- Fatores primários: são fatores que estão diretamente ligados a variável de resposta e por esse motivo têm grande influência;
- Fatores secundários: são fatores que não têm grande influência sob a variável de resposta ou são insignificantes e por isso não devem ser considerados;
- Replicação: ato de repetir parte ou por completo o experimento;
- Projeto: determina o número de experimentos que serão considerados, assim como o número de fatores e níveis;
- Carga de trabalho: dados de entradas e/ou requisições que são enviados para que o sistema possa processar.

Manipula-se de forma planejada certas variáveis independentes (fatores), definindo-se os valores mais prováveis que essas variáveis podem assumir (níveis) para verificar o efeito que esta manipulação provoca na variável de resposta (variável dependente) (SANTANA; SANTANA, 2000-2012).

Para compreender o funcionamento do sistema é necessário saber como ele funciona (falando de um sistema real), para isso é necessário considerar as cargas de trabalho que atuam no sistema. A partir delas é possível simular como o sistema se comportará a partir de um certo ponto.

2.3.2 Tipos de planejamentos

Para fazer a avaliação é necessário escolher qual tipo de planejamento será utilizado. Destacam-se três tipos com as vantagens e desvantagens de cada um.

1. Planejamento Simples: O sistema possui uma configuração inicial e são fixados valores fixo para os fatores que são variados apenas um por vez, após isso, verifica

qual fator afeta o desempenho do sistema. Tem a seguinte equação para k fatores onde com n_i índices do fator i.

$$n = 1 + \sum_{i=1}^{k}(n_i - 1) \tag{2.1}$$

Possui a vantagem de ser fácil para executar, mas não permite relacionar dois fatores, ou seja, ela não é estaticamente eficiente.

2. Planejamento fatorial completo: faz-se uso de todos os fatores com todos os níveis. Tem a seguinte equação para k fatores e n_i níveis no fator i.

$$n = \prod_{i=1}^{k} n_i \tag{2.2}$$

Possui a vantagem de avaliar todos os fatores e a relação deles, no entanto, o custo é muito alto devido a quantidade de fatores.

3. Planejamento fatorial parcial: neste planejamento, diferente do planejamento fatorial completo, o número de experimentos é reduzido para 2^k. De acordo com os resultados são selecionados os fatores primários e a partir deles faz uma nova análise variando os níveis dos fatores. É recomendado selecionar poucos fatores e apenas dois níveis por fator. Tem a vantagem de não ser tão difícil de implementar e poder analisar a relação entre os fatores.

2.3.3 Técnicas para avaliação de desempenho

Para cada tipo de sistema existem técnicas apropriadas para avalia-lo. Avaliar o desempenho de um sistema computacional é diferente de avaliar o motor de uma moto. Desta forma, elas são divididas em duas categorias.

1. Técnica de aferição: utilizada em sistemas reais ou similares e são utilizadas ferramentas como: protótipos, benchmarks e coleta de dados para aferir. Possui a vantagem de oferecer resultados precisos mas, tem um custo elevado e sofre influência de outros sistemas.

2. Técnica de modelagem: são criadas abstrações do sistema através do desenvolvimento de um modelo e solução para esse. Utiliza ferramentas como modelagem analítica, que fornece resultados aproximados, e simulação, esta técnica tem um baixo custo.

Além de selecionar escolher a técnica é necessário saber o que irá medir. Podemos medir:

- Velocidade: através do tempo (tempo de resposta), taxa, throught e recurso (utilização).
- Confiabilidade: probabilidade e intervalo entre erros.
- Disponibilidade: duração do evento e intervalo entre eventos.

2.4 Teoria das filas

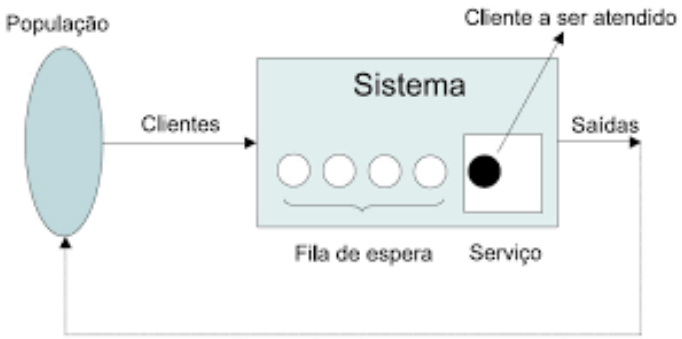
As filas ainda são consideradas um grande empecilho desde décadas passadas, seja ela de forma física ou digital. Estão presentes em todas as partes, seja no banco, banheiros e padarias. Num sistema computacional, ela não deixa de existir na hora de alocar um processo para CPU. As filas existem pelo motivo de a demanda de serviço ser maior que a capacidade de atendimento.

Segundo Sucena (2013), Teoria das Filas é uma técnica de pesquisa operacional que trata de problemas de congestionamento de sistema, onde clientes solicitam serviços, estes, são limitados por restrições intrínsecas do sistema, gerando assim, filas. Na figura 6 é possível mostrar como funciona uma fila.

É um ramo da probabilidade que estuda o fenômeno da formação de filas de solicitantes de serviços, fornecidos por um determinado recurso (TEIXEIRA, 2004).

De acordo com Fogliatti (2007), a teoria das filas consiste na modelagem analítica de processos ou sistemas que resultam em espera e tem como objetivo determinar e avaliar quantidades, denominadas medidas de desempenho, que expressam a produtividade e/ou operacionalidade desses produtos, e de posse destas informações, buscar meios para minimizar os impactos negativos das esperas nos processos.

Figura 6 – Funcionamento da fila



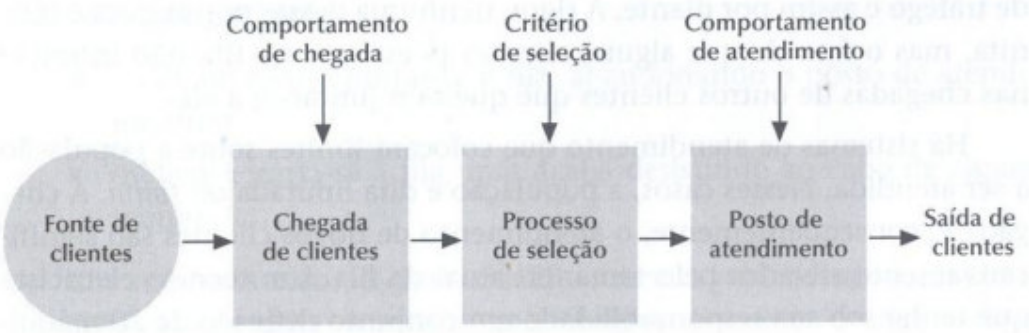
Fonte: IME-Unicamp[2]

[2] Disponível em: https://www.ime.unicamp.br/ hlachos/ME323-Teoria%20Filas.pdf. Acesso em: 20 jun. 2019

2.4.1 Estrutura básica de um sistema de fila

Conforme Moreira (2011), são estruturada em quatro partes principais: a fonte de clientes; a chegada de clientes; o processo de seleção e o posto de atendimento, conforme a figura 7. Os clientes são indivíduos de uma população que chegam ao local da prestação do serviço de acordo com determinado comportamento estatístico, para serem atendidos de acordo com um critério de seleção preestabelecido, e serão atendidos de acordo com características próprias.

Figura 7 – Estrutura da fila



Fonte: (MOREIRA, 2011)

2.4.2 Fonte de clientes

Os clientes são uma pequena parcela de uma população maior. A fonte, por sua vez pode ser finita ou infinita. No primeiro caso, a probabilidade de chegada é afetada pelo fato de alguns clientes já estarem na fila. No último caso, a probabilidade não é afetada de forma significativa pelo fato de alguns clientes já estarem na fila.

2.4.3 Chegada de clientes

O processo de chegada dos clientes dá-se pelo comportamento do fluxo de chegada dos mesmos. Caso o número de chegada seja conhecido e o momento que eles acontecem, então ele é dito determinístico; do contrário, é dito aleatório e possui um comportamento estocástico, que é caracterizado por uma distribuição de probabilidade. Tal distribuição é especificada através de um parâmetro denominado taxa de chegada, que corresponde ao número médio de clientes que chegam ao sistema por unidade de tempo (MOREIRA, 2011).

Geralmente a taxa de chegada é representada por λ e existem duas formas de chegadas do cliente: número de clientes que chegam dado um certo intervalo de tempo e o tempo decorrido entre duas chegadas consecutivas. Em teoria das filas é comum utilizar a distribuição de Poisson para configurar a taxa de chegada.

A seguir, na tabela 1, são apresentadas as distribuições de chegada de clientes mais comuns.

Tabela 1 – Distribuição de probabilidade utilizadas nas taxas de chegadas de clientes

Grandezas	Distribuição de chegada	Médias
Número de chegadas na unidade de tempo (taxa de chegada)	Poisson	λ
Tempo	Exponencial	$\frac{1}{\lambda}$

Fonte: (MOREIRA, 2011)

2.4.4 Processo de seleção

De acordo com Moreira (2011) o processo de seleção é caracterizado pelo comportamento do fluxo de atendimento, sendo análogo ao de chegada de clientes. O atendimento geralmente é representado por μ e do mesmo modo do modelo de chegada, existem duas formas: número de atendimentos na unidade de tempo e tempo decorrido entre dois atendimentos consecutivos.

A seguir, na tabela 2, são apresentadas as distribuições do atendimento de clientes mais comuns.

Tabela 2 – Taxa de atendimento de clientes no sistema de filas

Grandezas	Distribuição de chegada	Médias
Número de atendimentos na unidade de tempo(taxa de atendimento)	Poisson	μ
Tempo decorrido entre dois atendimentos consecutivos	Exponencial	$\frac{1}{\mu}$

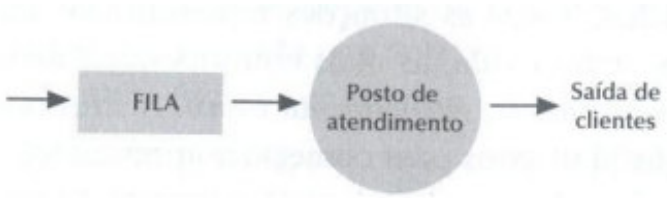
Fonte: (MOREIRA, 2011)

Os modelos de atendimento podem apresentar diversas configurações, conforme a seguir:

- Canal único: caracterizado por possuir apenas um ponto para atendimento, podendo ter um mais posto, desde que sejam em séries;
- Canal múltiplo: apresenta no mínimo dois pontos de atendimento em paralelo;
- Atendimento múltiplo: é realizado por mais de um ponto em série, dependendo uma da outra, como um pipeline;.
- Atendimento único: o atendimento é feito somente por um único ponto;

Nas figuras 8, 9, 10 e 11 é possível ilustrar a combinação das configurações

Figura 8 – Canal único, atendimento único

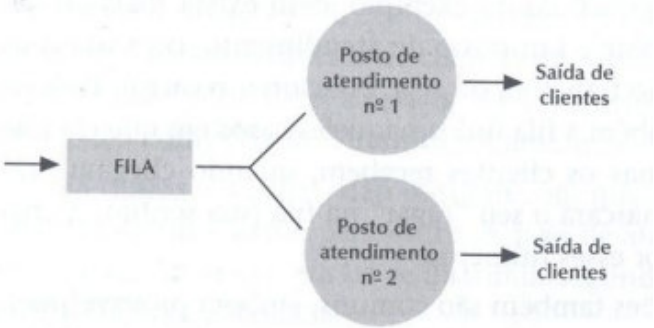


Fonte: (MOREIRA, 2011)

Figura 9 – Canal único, atendimento múltiplo

Fonte: (MOREIRA, 2011)

Figura 10 – Canal múltiplo, atendimento único



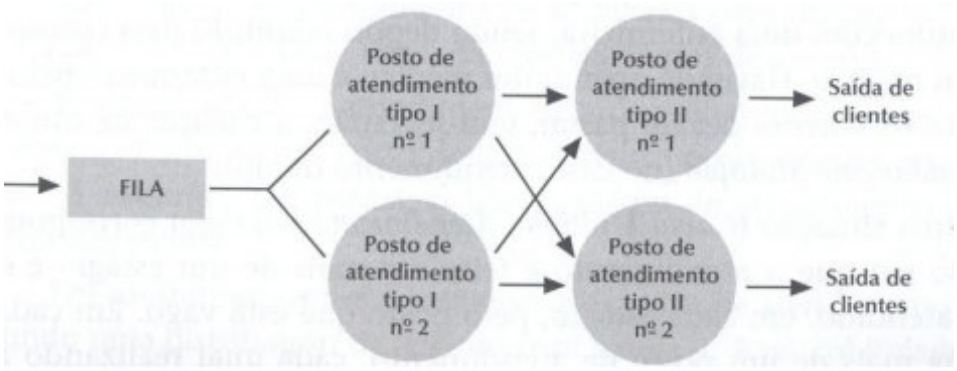
Fonte: (MOREIRA, 2011)

2.4.5 Disciplina de filas

O modo de atendimento de uma fila varia conforme a regra adotada. Os tipo de atendimentos mais comuns são:

- FIFO (*first in - first out*): o primeiro a entrar é o primeiro a sair (ser atendido);
- LIFO (*last in - last out*): o último a chegar é o primeiro a sair (ser atendido);
- PRI (*priority-service*): atendimento por prioridade, ou seja, existem regras que estabeleçam quem deve ter prioridado sobre o atendimento;
- SIRO(*service in random order*): o atendimento é feito de forma aleatória;

Figura 11 – Canal múltiplo, atendimento múltiplo



Fonte: (MOREIRA, 2011)

- SPT (*Shortest Processing Time first*): o cliente a ser atendido primeiro será o que levará menor tempo durante o atendimento.

2.4.6 Medidas de desempenho no sistema de filas

De acordo com o modelo adotado existem as características operacionais ou indicadores de desempenho para cada um (MOREIRA, 2011). Segundo Fogliatti (2007) e Mendonça (2014), a partir delas é possível avaliar a eficiência de um sistema por meio da análise de suas características, utilizando medidas operacionais/desempenho. São consideradas variáveis aleatórias, pois na maioria das vezes, ao longo do tempo, elas mudam, mas os valores podem ser utilizados como medidas de desempenho do sistema no regime estacionário. Dentre elas podemos destacar:

- Número médio de usuários na fila (L_q) e no sistema (L);
- Tempo médio de espera de um usuário qualquer na fila (L_q);
- Tempo médio de permanência de um usuário qualquer no sistema (W);
- Taxa de utilização do servidor; é uma medida de congestionamento do sistema, ρ;
- Probabilidade de se ter no máximo um número n_0 pré-fixado de usuários no sistema, $P(N \leq n_0)$;
- Probabilidade de um usuário qualquer ter que aguardar mais do que um determinado tempo t na fila, $P(T_q > t)$;
- Probabilidade de se ter algum servidor ocioso em um sistema com c posto de atendimento, $P(N < c)$;
- Probabilidade de que haja mais de k clientes na fila, $P(N > k)$.

2.4.7 Modelos básicos de fila

A maioria dos modelos elementares de fila utilizam o processo de nascimento e morte (markoviano). Neste contexto, o nascimento significa a entrada de um novo cliente na fila e a morte corresponde a saída do cliente. A partir do modelos descritos por Fogliatti (2007) são apresentados os seguintes modelos:

- Modelo M/M/1/∞/FIFO: existe apenas um único ponto de atendimento, não existe limite de espaço na fila e atendimento é realizado conforme o sistema FIFO;
- M/M/1/K/FIFO: há apenas um único ponto de atendimento com limite de espaço na fila e o atendimento é realizado conforme o sistema FIFO; A taxa de ingresso no sistema (λ) é diferente de chegada λ para $n \geq K$ pela existência do limite de (K);
- M/M/C/∞/FIFO: há 'C' posto de atendimento sem limite de espaço na fila e o atendimento é realizado utilizando o sistema FIFO;
- M/M/C/K/FIFO: há 'C' postos de atendimento com limite de espaço para fila e é realizado conforme o sistema FIFO; A taxa de ingresso no sistema (λ) é diferente de chegada λ para $n \geq K$ pela existência do limite de (K).

2.5 Apache JMeter

O Apache JMeter é um software de código aberto feito totalmente em Java e foi projetado para carregar o comportamento funcional do teste e medir o desempenho. Ele foi originalmente projetado para testar aplicativos da Web, mas desde então, expandiu para outras funções de teste (JMETER, 2019).

A partir do software é possível gerar cargas de trabalho sintética para testar recursos estáticos e dinâmicos, além de aplicativos web. É possível definir o grupo de usuários para acessar a aplicação, tempo de inicialização entre cada usuário, definir a quantidade de interações, gerar gráficos e outras funcionalidades.

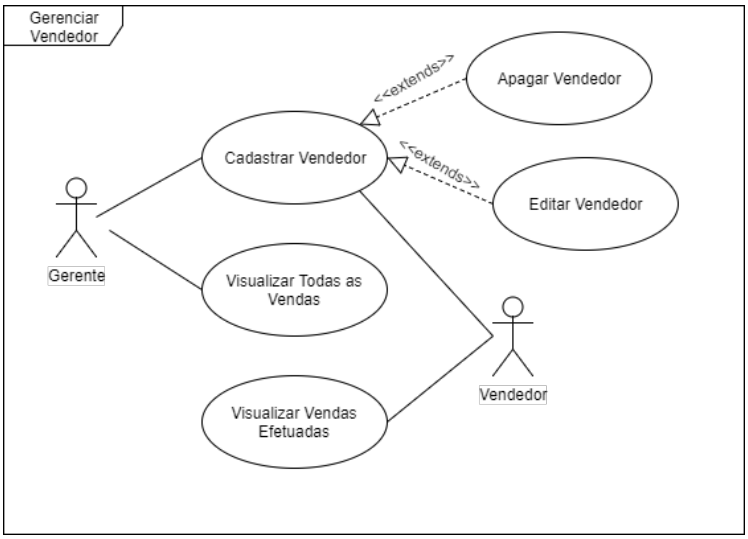
2.5.1 Blazemeter

O Blazemeter é um plugin utilizado juntamente com o JMeter para automatizar o processo de teste de uma aplicação ou website. Ele é capaz de gravar os passos e requisições do teste e salvar num arquivo de formato JMX. A partir deste arquivo, o JMeter lê o teste e é configurado a quantidade de usuários para interagir com o sistema.

2.6 Aplicação Web - CarController

A aplicação Carcontroller foi desenvolvida por um grupo de alunos, incluindo o responsável por este trabalho apresentado, durante a disciplina de Projeto e Desenvolvimento de Software cursada na graduação. Tem como objetivo auxiliar gerente e vendedores na venda de veículos automobilísticos. Através do software, é possível realizar login entre os dois tipos de funcionários existentes. O gerente é responsável por cadastrar vendedores e veículos no sistema, além de acompanhar a venda realizada por seus funcionários. O vendedor é responsável por cadastrar clientes e realizar venda de veículos disponíveis. Na figura 12 é o diagrama de caso de uso das funções que o gerente pode executar.

Figura 12 – Caso de uso do gerente



Fonte: Autor do trabalho

A ferramenta não apresenta irregularidades quanto ao seu funcionamento, independente de onde esteja hospedada. Esta foi utilizada no trabalho pelo motivo de o autor participar do desenvolvimento e conhecer por completo o seu funcionamento. Segundo a definição de (NATIONS, 2018) ela se encaixa no que é uma aplicação web. No entanto, poderia ser utilizada qualquer outra aplicação desde que atenda a definição.

O software foi desenvolvido utilizando a linguagem de programação PHP com auxílio de desenvolvimento web (HTML, CSS e Javascript), para representar e para guardar os dados foi utilizado o MYSQL.

3 Metodologia

Este capítulo descreve a metodologia utilizada. Na seção 3.1 são apresentas as configurações das máquinas utilizadas durante o experimento e o ambiente de simulação. Na seção 3.3 são apresentados os cenários desenvolvidos e os passos para geração dos usuários, utilizando a aplicação em cada cenário.

3.1 Ambiente de testes

3.1.1 Hardwares e softwares da EC2

Para hospedar a aplicação web carcontroller foi preparado o ambiente de hospedagem com as seguinte configurações de hardware: processador Intel Xeon Family de 64 bits com CPU @ 2,5GHz, 1GB de memória RAM, 8GB de armazenamento e velocidade de internet de baixa moderação. Todas essas características se resumem a uma instância do tipo t2.micro oferecido pelo plano gratuito da AWS.

Os softwares foram: sistema operacional Ubuntu Server 18.04 LTS (HVM) - (x86), Apache 2.4.29, PHP 7.1.27, MYSQL 5.5.56

3.1.2 Características do ambiente de simulação de carga

Configurações de hardware: processador Intel®$Core^{TM}$ i5-4210U de 64 bits com CPU @ 1.70GHz, 4GB de memória RAM e velocidade de internet de 30mbps com grande variação.

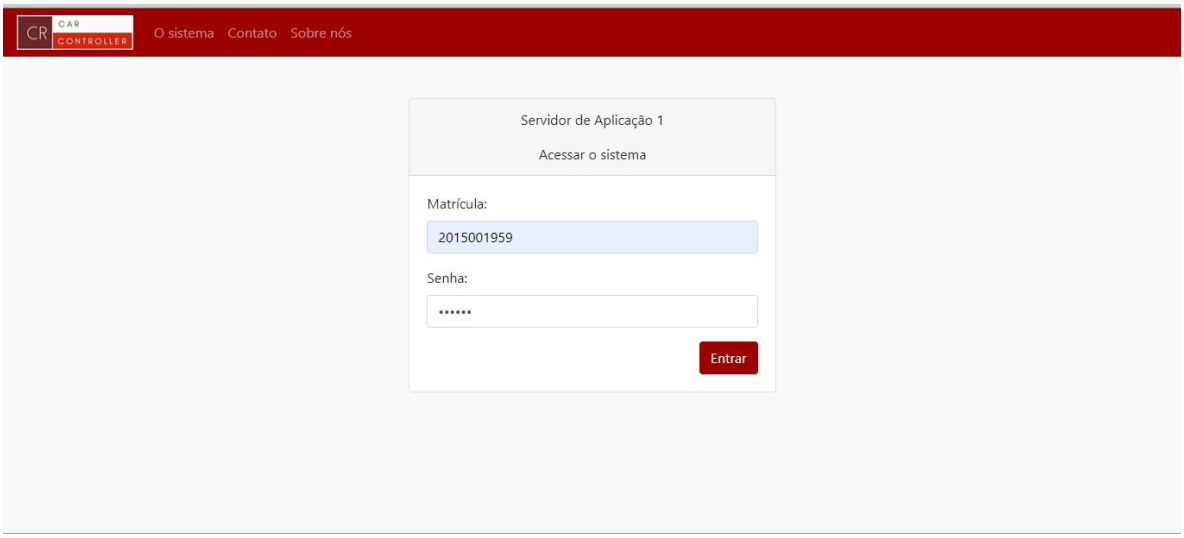
Software utilizados: Windows 10 64-bit Home Single Language e Apache Jmeter versão 4.0.

3.1.3 Experimentos de simulação de carga

Para a geração de carga, foi utilizada a ferramenta Apache JMeter e o plugin Blazemeter, descritos anteriormente. Cada qual utilizados na versão 4.0 e 4.6.0, consecutivamente.

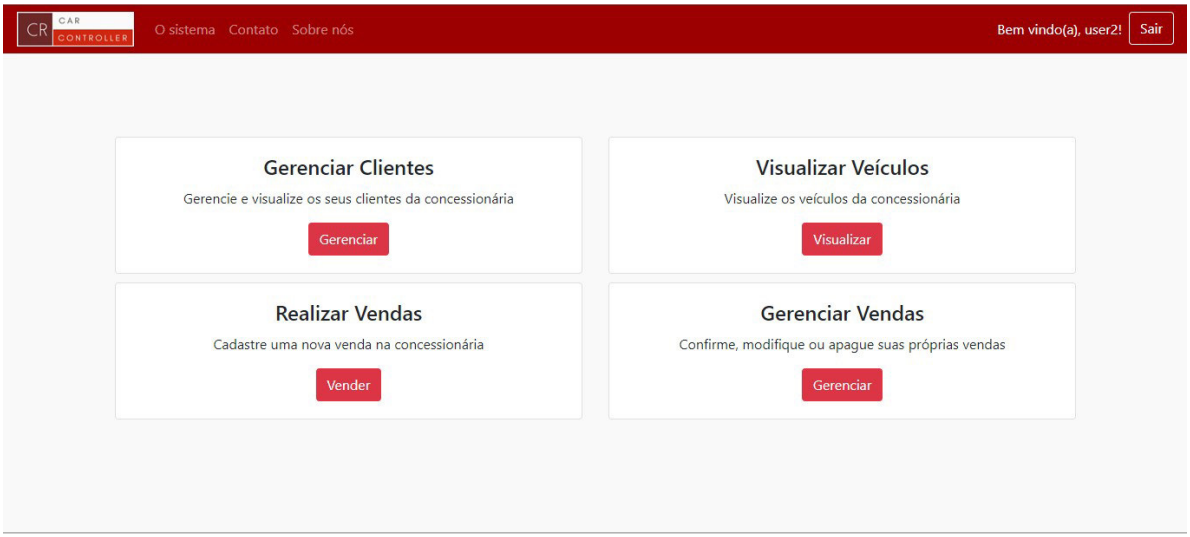
Com o Blazemeter foi gravado a interação de um usuário entrando no sistema CarController a partir de um login (figura 13), a visualização após o login (figura 14), e por fim, o logout voltando para a tela de login.

Figura 13 – Tela de login do usuário



Fonte: Autor do trabalho

Figura 14 – Tela de visualização após o login



Fonte: Autor do trabalho

3.1.4 Parâmetros de entrada

Após a gravação utilizando o Blazemeter, cada simulação de carga recebe os seguintes dados de entrada:

- O arquivo JMX gerado pelo Blazemeter(contendo as requisições HTTP, gerenciador de autorização HTTP, gerenciador de cookie, gerenciador de cache, gerenciador de cabeçalho HTTP e gerenciador de DNS);
- Um arquivo CSV com dados de matrícula e senha;

- Número de usuários virtuais que irão acessar o sistema;
- Tempo de inicialização dos usuários;
- Duração do teste;

3.2 Cenários

Para os testes foram criado dois cenários, cada um com diferentes configurações. Para cada cenário foram realizados 12 experimentos, três para cada nível e fator alterado.

3.2.1 Cenário 1

Este cenário foi montado utilizando o meio mais comum para hospedar uma aplicação num servidor, fazendo uso de apenas uma instância EC2 para hospedar a aplicação web e outra para hospedar o banco de dados de acordo com as configurações descritas no item 3.1.1. Assim, ao fazer realizar o experimento, os recursos utilizados seriam de apenas duas máquinas

3.2.2 Cenário 2

Diferente do cenário 1, a aplicação foi hospedada em mais de uma instância EC2, foram utilizadas no mínimo duas instâncias e no máximo três, para hospedar. Então, de início, já estava sendo feito o balanceamento de carga pelo ELB. O mínimo significa que a web app está ativa em duas instâncias ao mesmo tempo e podendo acessar ambas com o mesmo resultado. Optou-se em iniciar com duas (uma criada de forma manual e outra fazendo uso do autoscaling) pelo motivo de ao criar somente pelo escalonador não ser possível controlar qual instância será desligada após a política do escalonamento ser aplicada, assim seria inviável monitorar o desempenho da aplicação instanciada naquela EC2. Desta forma, foi criada uma de forma manual e ela ficou como "fixa"e possível de monitorá-la. Para utilizar o escalonador automático, é necessário ter no mínimo uma instância criada por ele e somando com a primeira criada de forma manual, fica garantido que sempre haverá duas instâncias atuando.

A política aplicada para o autoscaling foi tomando como base a utilização da CPU, caso ultrapassasse a utilização de 1% pelo período de tempo de 1 minuto seria criada uma nova instância, e caso fosse menos de 1% pelo período de tempo de 5 minutos uma instância seria desligada. No entanto, o ideal seria utilizar a memória RAM como política, mas não foi possível em razão da AWS não permitir, ficando restrito apenas a poucos recursos de hardware para CPU, como: leitura e gravação de dados, pacotes de entrada e saída.

Este cenário então, poderia utilizar os recursos de até quatro instâncias, no total, caso a política fosse acionada e configurada da seguinte forma: até três instâncias para aplicação e uma instância para o banco de dados

3.2.3 Descrição do planejamento adotado

Utilizando o planejamento fatorial parcial, foram escolhidos como fatores primários: o número de usuários que acessa a aplicação e o tempo entre cada requisição. Para o número de usuários, os níveis considerados foram de 10 e 100. Para o tempo entre cada requisição foi considerado de 1s e 10s.

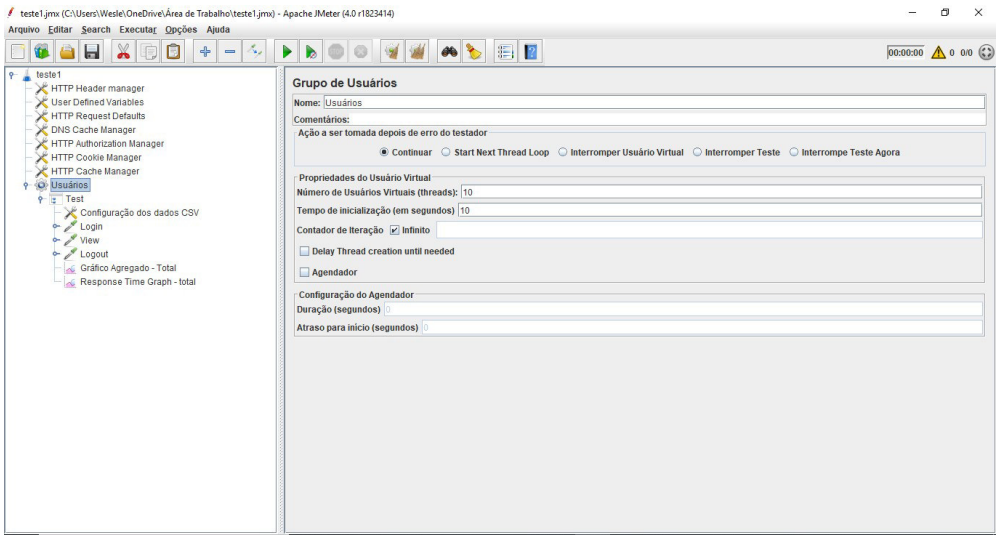
Primeiramente foi testado utilizando o Cenário 1 variando os níveis de cada fator, em seguida foram feitos os testes no cenário 2 da mesma forma.

3.3 Simulação

3.3.1 Geração de carga

A geração de carga se deu pela ferramenta JMeter (figura 15), após o arquivo ser gerado pelo Blazemeter. A partir dela, foi possível exportar gráficos e tabelas referentes ao tempo de resposta da aplicação. Através dela foi possível limpar dados da memória cache e cookies para não interferirem nos testes. Foi possível definir a quantidade do grupo de usuários exata para acessar a aplicação e o tempo de inicialização para cada um deles, além de definir a duração de cada teste.

Figura 15 – Interface do JMeter



Fonte: Autor do trabalho

3.3.2 Geração de demandas

Ao final de cada teste foram colhidos gráficos de tempo médio para atendimento das requisições, throughput (vazão), utilização da memória e percentual de erros. A partir da AWS foi possível acompanhar a utilização da taxa de CPU, utilização da memória RAM e se fez uso do recurso autoscaling.

4 Experimentos e resultados

Neste capítulo são descritos os experimentos realizados para os fatores e níveis propostos, e os resultados de cada um.

4.1 Experimentos realizados

Considernado o tempo de 60 minutos para cada um dos testes, no entanto, o número de amostras varia de acordo com os fatores e níveis alterados, além da variação da velocidade da rede. Mas foram colhidos a média total das amostras de cada experimento conforme as tabelas 3 e 4.

Tabela 3 – Média de requisições no cenário 1

Testes - Cenário 1	Amostras (requisições)
10 req./s	13347,33333
10 req./10s	12856,66667
100 req./s	30036,66667
100 req./10s	29286,66667

Fonte: Autor do trabalho

Tabela 4 – Média de requisições no cenário 2

Testes - Cenário 2	Amostras (requisições)
10 req./s	10569,33333
10 req./10s	11422,66667
100 req./s	22429,6666
100 req./10s	17402,66667

Fonte: Autor do trabalho

A partir dos dois cenários elaborados foi possível avaliar resultados importantes e a combinação entre eles para analisar o desempenho da aplicação, temos os seguintes:

- Tempo médio;
- Throughput (Vazão);
- Memória utilizada da aplicação;
- Percentual de erro;
- Memória utilizada da aplicação x Percentual de erro;

- Memória utilizada do banco de dados x Tempo médio;
- Memória utilizada do banco de dados X Throughput.

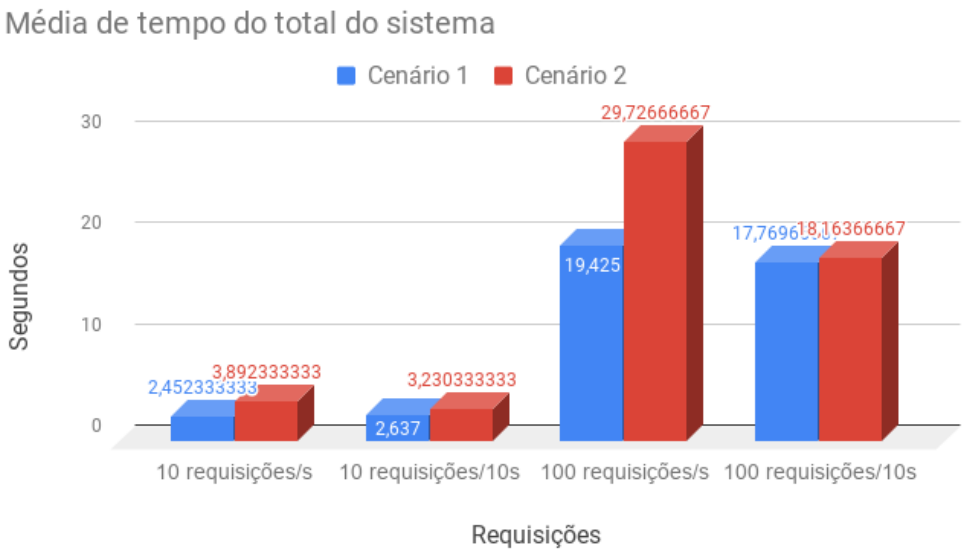
No apêndice A é possível visualizar o resultados detalhado de cada teste efetuado.

4.2 Resultados

4.2.1 Tempo médio

Para este resultado, procuramos comparar o tempo de resposta entre os dois cenários existentes.

Figura 16 – Média de tempo total do sistema



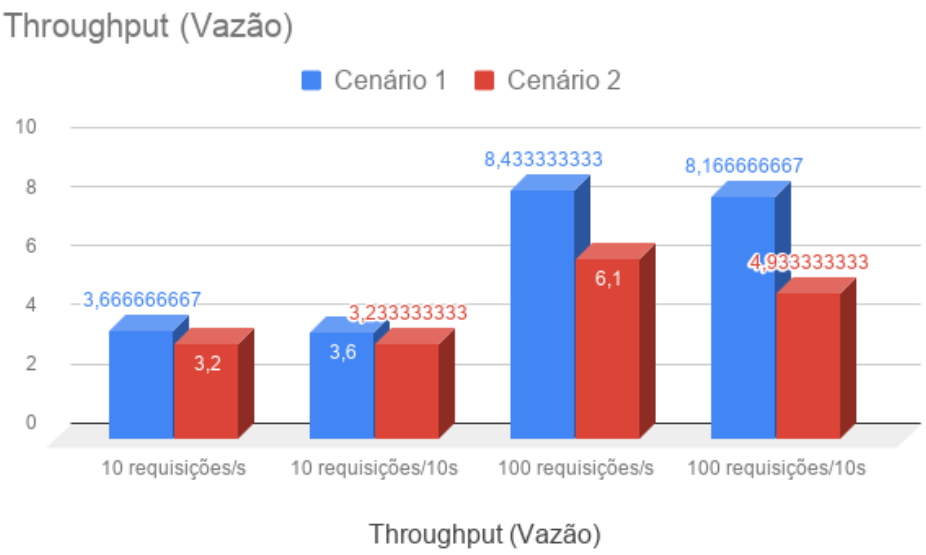
Fonte: Autor do trabalho

Notou-se na figura 16 que ao utilizar a aplicação somente em uma instância, o tempo de resposta é menor comparado ao utilizar mais de uma instância. O fator responsável por esse tempo ser maior é o número de instâncias que interagem com o banco de dados, apesar de haver várias instâncias para a aplicação. Existe somente uma para o banco de dados, isto é, o banco de dados deve responder a todas as consultas feitas por todas as instâncias. Isto será analisado melhor quando comparada a utilização de memória RAM da instância em que se encontra o banco de dados com o tempo de resposta.

4.2.2 Throuput (Vazão)

Considerando a vazão do sistema que é a quantidade de requisições que são atendidas por um período de tempo.

Figura 17 – Média de vazão do sistema



Fonte: Autor do trabalho

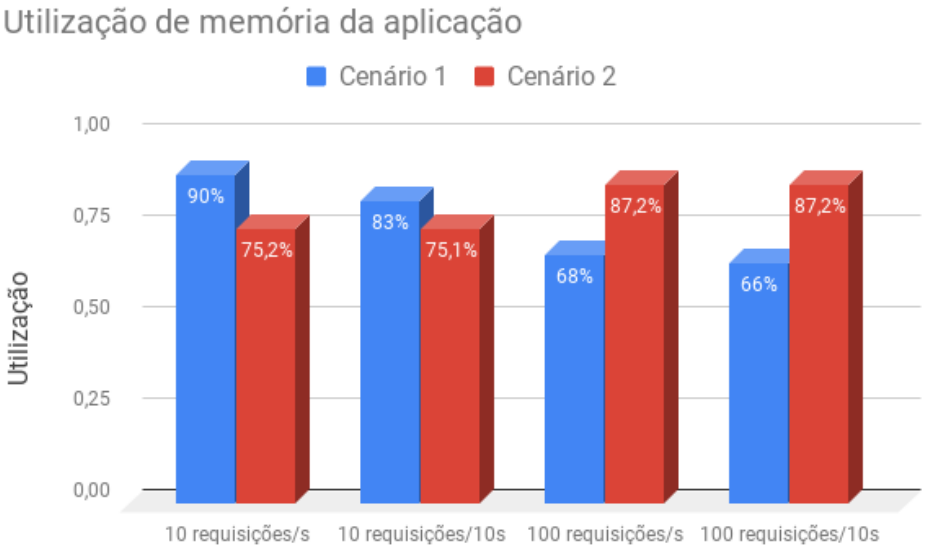
Ao analisar a figura 17, percebe-se mais uma vez que o cenário 1 tem vantagem comparado ao 2, pois assim pode atender um maior número de requisições por segundo, e desta forma, acaba atendendo um maior número de requisições. No entanto, nem sempre quanto maior a vazão melhor é o desempenho da aplicação, é necessário levar em consideração outros fatores como a taxa de erro, pois, se for alta, significa que nem todas as requisições foram concluídas com sucesso, elas simplesmente foram interrompidas por falta de comunicação. Ao comparar o throughput com a utilização da memória da aplicação teremos explicação melhor.

4.2.3 Memória utilizada da aplicação

Considerando a memória utilizada no cenário 1 e a média das memórias do cenário 2 temos o seguinte resultado (figura 18).

É fato que nas dois primeiros experimentos, o cenário 1 utiliza mais memória comparado ao 2, isto deve-se ao fato dele conseguir atender as requisições carregando-as somente na memória RAM, por consequência tem a vazão maior. No entanto, nos dois últimos experimentos a utilização de memória diminui e a do cenário 2 aumenta mesmo com a vazão menor. Isto ocorre devido ao uso de memória virtual que para evitar uma sobrecarga o sistema começa a carregar os processos na memória secundária do computador, comumente conhecido como paginação, e libera a memória principal.

Figura 18 – Média de utilização da memória do sistema

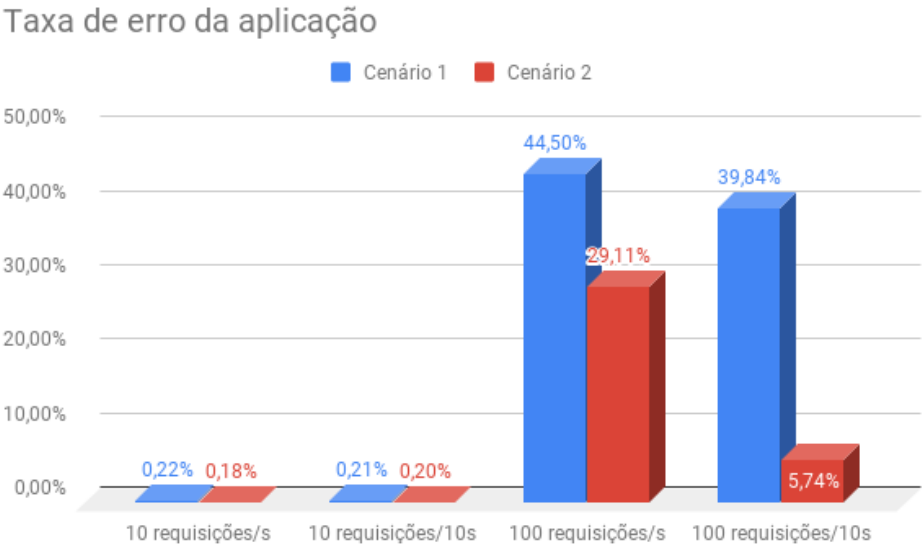


Fonte: Autor do trabalho

4.2.4 Taxa de erro

A taxa de erro consiste na média total de requisições que não foram atendidas por completo durante os testes.

Figura 19 – Taxa de erro do sistema



Fonte: Autor do trabalho

Nesta figura 19, percebe-se que a taxa de erro no cenário 1 é maior em todos os testes comparado ao cenário 2. Para analisar melhor, é necessário compará-la com a

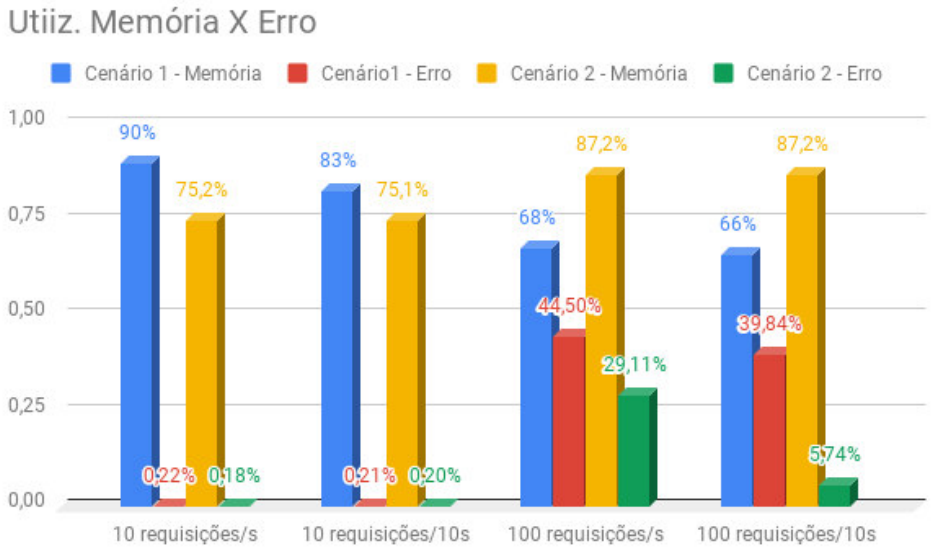
utilização de memória, pois assim será possível entender o motivo de haver tanto erro.

4.2.5 Memória utilizada x Percentual de erro

Combinar duas métricas é o melhor jeito para entender os resultados, assim, na figura 20 é possível notar que nos dois primeiros testes do cenário 1, a utilização da memória foi alta e houve uma mínima taxa de erro, e no cenário 2, a memória utilizada foi razoável com uma taxa de erro ainda menor.

Nos dois últimos testes, percebe-se que no primeiro cenário a taxa de utilização da memória diminui e a de erro aumenta drasticamente, enquanto ocorre o contrário no cenário 2. Isto deve-se ao fato de nos dois últimos, a instância utilizada no cenário 1 ficar sobrecarregada e acabada fazendo uso da memória virtual para tentar diminuir a utilização da memória principal, deixando assim, o sistema lento, gerando muitos erros. Enquanto no cenário 2, ele é capaz de suportar as requisições dividindo as requisições entre todas a instâncias criadas e ocasionando menos erros.

Figura 20 – Comparação entre memória e taxa de erro



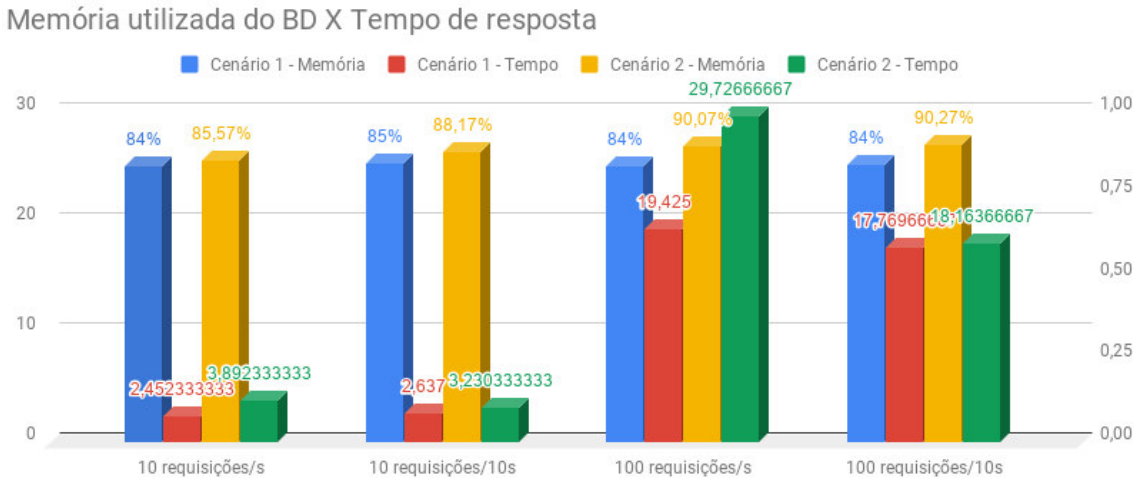
Fonte: Autor do trabalho

4.2.6 Memória utilizada do Banco de dados x Tempo médio

Para entender o porquê do tempo de resposta ser maior para aplicações que fazem uso do autoscaling, é necessário analisar não somente as instâncias que são escalonadas às aplicações, mas também, aquelas que estão conectadas a elas. Por isso, analisando a figura 21, é possível comparar o tempo de resposta de cada teste com a utilização de memória do banco de dados. É possível notar que quanto maior a utilização de memória

do BD, maior é o tempo de resposta. Deve-se ao fato de que neste cenário o BD deve responder as consultas feitas por todas as instâncias e, consequentemente, levando um maior tempo.

Figura 21 – Comparação entre memória utilizada do BD e tempo de resposta



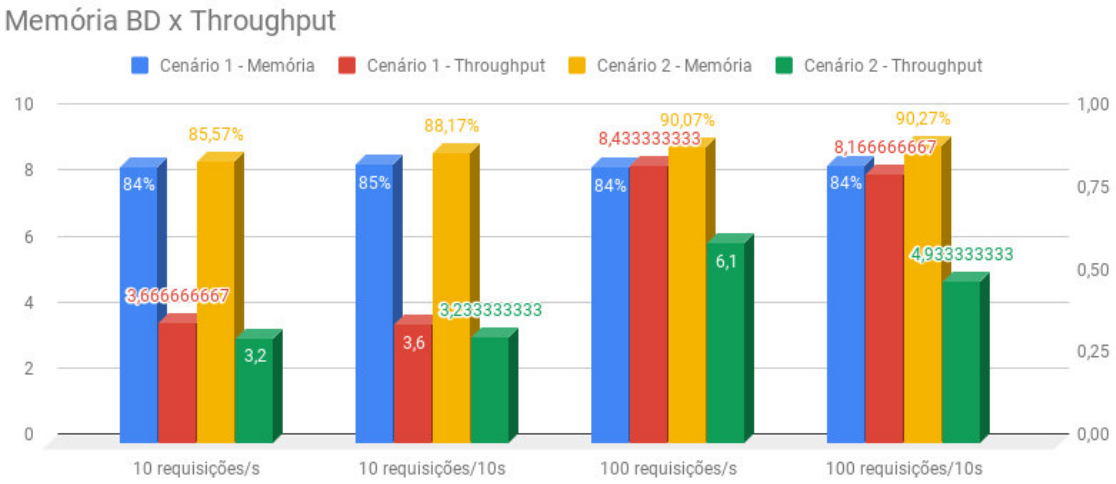
Fonte: Autor do trabalho

4.2.7 Memória utilizada do Banco de dados X Throughput

Esclarecido o motivo de ter um maior tempo de resposta no cenário 2, é possível aplicar a mesma explicação para a vazão. Ao examinar a figura 22, é possível constatar que quanto maior a utilização da memória do BD menor é a vazão do sistema. Devido ao fato de o BD ser consultado por mais instância, acaba levando um tempo maior para responder todas as consultas.

Uma possível solução dentro do ambiente da nuvem para tornar o tempo de resposta e vazão maiores, é utilizar um banco de dados escalável, conhecido como Amazon Relational Database Service (RDS). Parecido com o autoscaling, utilizando o princípio de ser redimensionável. Outra solução seria a utilização de uma EC2 de grande capacidade computacional para atender a todas as requisições, no entanto, esta não estaria fazendo uso da eficiência que a nuvem oferece.

Figura 22 – Comparação entre memória utilizada do BD e Throughout



Fonte: Autor do trabalho

5 Considerações finais

A utilização da computação em nuvem é um importante recurso na hora de desenvolver uma aplicação web por todas as características e vantagens citadas. Através dela é possível monitorar todos os recursos utilizados e descobrir em quais pontos estão falhando para assim ter um melhor desempenho. Além disso, é considerável avaliar como anda o sistema computacional utilizando ferramentas e técnicas adequadas para não cometer erros que interfiram no resultado final da análise.

Ao se deparar com um sistema com um tempo de resposta tão grande e elevada taxa de erro, a probabilidade de um usuário querer continuar nele diminui drasticamente, caso seja um sistema de venda, o dono do sistema provavelmente não terá lucro.

Os resultados satisfatórios desta simulação em uma aplicação web elástica reforçam a aplicabilidade do uso deste recurso para a tomada de decisão na hora de desenvolver/hospedar uma aplicação cujo número de acessos seja desconhecido. Foram feitos testes na aplicação utilizando link de internet dedicados e os resultados tornaram-se melhores (menor tempo de resposta e taxa de erro em ambos os cenários).

A realização do trabalho possibilitou ampliar nosso conhecimento em computação em nuvem desde conceitos básicos até a implementação, amplo conhecimento na parte de avaliação de desempenho, além de conhecer as ferramentas, técnicas e análises para resolver um problema.

5.1 Trabalhos futuros

Para trabalhos futuros pretende-se investigar as demais áreas da computação em nuvem tais como Docker swarm e Kubernet. Aprofundar o conhecimento sobre a AWS e descobrir o tipo de escalonamento utilizado.

5.2 Produção científica

Este trabalhou foi selecionado como um dos 10 melhores para a apresentação oral do SEMIC do ano de 2018 na UFMA, Campus Bacanga.

Referências

AWS. *Computação em nuvem com a AWS*. 2019. Disponível em: <https://aws.amazon.com/pt/what-is-aws/?nc2=h_ql_le>. Acesso em: 28 Maio 2019. Citado 2 vezes nas páginas 24 e 26.

BREBNER, P. C. *Is your cloud elastic enough? Performance modelling the elasticity of infrastructure as a service (iaas) cloud applications*. [S.l.]: In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE '12, 2012. 263 – 266 p. Citado na página 19.

CARDELLINI, V.; CASALICCHIO, M. C. E.; YU, P. S. *The State of the Art in Locally Distributed Web-Server Systems*. [S.l.]: ACM Computing Surveys, 2002. 263-311 p. Citado na página 15.

COSTA, H. L. A. Alta disponibilidade e balanceamento de carga para melhoria de sistemas computacionais crÍticos usando software livre: Um estudo de caso. Programa de Pós-gradução da UFV; Viçosa, 2009. Citado na página 18.

FARIAS, E. M. B.; SOUSA, E. S. de; NASCIMENTO, R. P. do. Avaliação de desempenho de um website utilizando apache jmeter: Um estudo de caso do website institucional da universidade federal do oeste do pará. XXXIV SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 2016. Citado na página 18.

FOGLIATTI, M. C. *Teoria das filas*. 2007. Disponível em: <<https://books-.google.com.br/books?id=l69OSAAACAAJ>. Citado 3 vezes nas páginas 29, 33 e 34.

HERBST, N. R. *Elasticity in cloud computing: What it is, and what it is not*. [S.l.]: In Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), 2013. 23—27, p. Citado na página 19.

IBM; VERAS. [S.l.]: s.d apud DIÓGENES, 2014. 6 p. Citado na página 16.

JAIN, R. *The Art of Computer Systems Performance*. [S.l.]: Wiley, 1991. Citado na página 26.

JMETER, A. *Apache JMeter*. 2019. Disponível em: <https://jmeter.apache.org/>. Acesso em: 28 Maio 2019. Citado na página 34.

KUPFERMAN, J.; SILVERMAN, J. *Scaling into the cloud. ADVANCED OPERATING SYSTEMS*. [S.l.: s.n.], 2016. 1 - 8 p. Citado na página 19.

LECHETA, R. R. *AWS para desenvolvedores*. [S.l.]: novatec, 2014. 86 p. Citado 2 vezes nas páginas 24 e 25.

LORIDO-BOTRAN. *A review of auto-scaling techniques for elastic applications in cloud environments*. [S.l.]: Journal of Grid Computing, 2014. Citado na página 19.

MENASCE, D.; ALMEIDA, V. A. F. *Planejamento de capacidade para serviços na web:métricas, modelos e métodos*. [S.l.]: Campus, 2002. Citado 2 vezes nas páginas 15 e 26.

MENDONçA, E. B. de. *Teoria das Filas Markovianas e Aplicações. Dissertação(Dissertação de Monografia)*. [S.l.]: Universidade Estadual da Paraíba, 2014. Disponível em: <https://goo.gl/i0pRB>. Acesso em: 23 jun. 2019. Citado na página 33.

MOREIRA, D. A. *Pesquisa operacional - curso introdutório*. 2. ed. [S.l.]: Cengage: Brasil, 2011. Citado 4 vezes nas páginas 30, 31, 32 e 33.

NATIONS, D. *What Exactly Is a Web Application?* 2018. Disponível em: https://www.lifewire.com/what-is-a-web-application-3486637. Acesso em: 23 de Fev. 2019. Citado 2 vezes nas páginas 15 e 35.

NIST. *The nist definition of cloud computing*. [S.l.]: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf., 2011. Citado 2 vezes nas páginas 19 e 20.

NIST. [S.l.]: apud BRASIL, 2016. Https://www.nist.gov/. Citado na página 16.

O que... *O que você realmente precisa saber sobre Computação em Nuvem*. 1. ed. [S.l.]: Opus software, 2015. 11 - 17 p. Citado 3 vezes nas páginas 21, 22 e 23.

PADILHA, B. S.; PADOIN, E. L. AnÁlise de desempenho da aplicaÇÃo de balanceamento de carga em benchmark sintÉticos. Salão do conhecimento - Ciência alimentando o Brasil; XXIV Seminário de Iniciação Científica, 2016. Citado na página 18.

PARKHILL, D. *The challenge of the computer utility*. 1966. Citado na página 16.

SANTANA, M. J.; SANTANA, R. H. C. *Avaliação de Desempenho*. [S.l.]: Departamento de Sistemas de Computação, Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo., 2000–2012. Citado na página 27.

SANTOS, T. *Fundamentos da Computação em Nuvem*. [S.l.]: Senac, 2018. 15 p. Citado na página 16.

SOSINSKY, B. *CLOUD SERVICES, NETWORKING, AND MANAGEMENT 1. ed.* [S.l.]: Indianopolis, 2015. 408 p. Citado na página 19.

SOUZA, M. A. de. *Uma abordagem para avaliação do escalonamento de processos em sistemas distribuídos baseada em monitoração*. [S.l.]: Instituto de Ciências Matemáticas e Computação ICMC - USP, 2004. Citado na página 18.

SUCENA, M. *Teoria das filas*. 2013. Engenharia de Produção. Brasil: Faculdade Estácio v.1. Citado na página 29.

TEIXEIRA, M. M. *Teoria das filas*. 2004. Disponível em: <http://www.deinf.ufma.br/~mario/grad/filas/filas.pdf>. Acesso em: 22 de jun. 2019. Citado na página 29.

VERAS, M. *Computação em Nuvem*. [S.l.]: BRASPORT, 2015. Citado na página 19.

Apêndices

APÊNDICE A – Resultado completo dos experimentos realizados

Cenário 1										
	Amostras	**Média(s)**	**Mediana(s)**	**95% line**	**min.(s)**	**max.(s)**	**% erro**	**vazão/s (X)**	**Mémoria Aplicação**	**Memória BD**
Experimento 1 (10us/seg)										
teste1	13272	2530	2151	6067	35	29085	0,15%	3,7	89%	84%
teste2	13660	2371	2063	5350	113	27767	0,29%	3,8	90%	84,50%
teste3	13110	2456	2136	5237	96	26987	0,21%	3,5	91%	84,50%
Média total	13347,33333	2,452333333	2,116666667	5,551333333	0,08133333333	27,94633333	0,22%	3,666666667	90%	84%
Experimento 2 (10 req./10seg)										
teste1	13682	2359	1964	5478	167	25280	0,18%	3,8	84%	84,70%
teste2	12070	2840	2298	7047	169	26444	0,14%	3,4	82%	84,90%
teste3	12818	2712	2163	6065	171	26799	0,30%	3,6	83%	84,70%
Média total	12856,66667	2,637	2,141666667	6,196666667	0,169	26,17433333	0,21%	3,6	83%	84,77%
Experimento 3(100 req./s)										
teste1	30739	20392	21001	63447	166	172643	48,57%	8,8	70%	83,80%
teste2	27861	17988	9847	52902	157	3027085	36,30%	7,7	67%	83,60%
teste3	31510	19895	20999	63039	93	241031	48,63%	8,8	67%	83,70%
Média total	30036,66667	19,425	17,28233333	59,796	0,1386666667	1146,919667	44,50%	8,433333333	68%	83,70%
Experimento 4 (100 req./10s)										
teste1	31161	19800	19373	60429	168	3488399	47,39%	8,7	65%	84,10%
teste2	30844	20358	21000	63031	12	173008	48,79%	8,6	67,5%	84,20%
teste3	25855	13151	4913	47424	181	2694754	23,35%	7,2	64%	85,10%
Média total	29286,66667	17,76966667	15,09533333	56,96133333	0,1203333333	2118,720333	39,84%	8,166666667	66%	84,47%
Cenário 2										
	Amostras	**Média**	**Mediana**	**95% line**	**min**	**max**	**% erro**	**vazão/s**	**Memória Aplicação**	**Memória BD**
Experimento 1 (10 req./seg)										
teste1	9752	4485	3220	13130	185	63390	0,16%	2,4	74,5%	85,40%
teste2	11599	3313	2576	8825	54	42921	0,19%	3,8	76,16%	85,40%
teste3	10357	3879	2774	8885	67	44958	0,18%	3,4	75%	85,90%
Média total	10569,33333	3,892333333	2,856666667	10,28	0,102	50,423	0,18%	3,2	75,2%	85,57%
Experimento 2 (10 req./10seg)										
teste1	12067	3061	2238	8053	73	126174	0,16%	3,4	72,5%	85,40%
teste2	11204	3420	1870	7352	33	222381	0,25%	3,1	75,5%	89,60%
teste3	10997	3210	1786	6889	42	201187	0,19%	3,2	77%	89,50%
Média total	11422,66667	3,230333333	1,964666667	7,431333333	0,04933333333	183,2473333	0,20%	3,233333333	75,1%	88,17%
Experimento 3 (100 req./s)										
teste1	21915	29859	13480	111009	35	358566	28,54%	6	88,50%	89,90%
teste2	22230	29179	17339	101358	186	896280	32,97%	6,2	86%	90%
teste3	23144	30142	15876	103567	76	786755	25,82%	6,1	87%	90,30%
Média total	22429,66667	29,72666667	15,565	105,3113333	0,099	680,5336667	29,11%	6,1	87,17%	90,07%
Experimento 4 (100 req./10s)										
teste1	20181	16857	8987	50963	188	3599614	8,11%	5,6	85,5%	90,30%
teste2	14329	17646	8707	44806	32	3591595	2,74%	4	84,75%	90,50%
teste3	17698	19988	8618	47981	60	3576841	6,37%	5,2	84%	90%
Média total	17402,66667	18,16366667	8,770666667	47,91666667	0,09333333333	3589,35	5,74%	4,933333333	84,8%	90,27%