CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U nº 198, de 14/10/2016
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Talles Alldelamsims da Silva Lopes

BALANCEAMENTO DE REQUISIÇÕES DE SERVIÇOS WEB EM CLUSTER DO PORTAL ACADEMICO CELP/ULBRA

Palmas – TO
2019

Talles Alldelamsims da Silva Lopes

BALANCEAMENTO DE REQUISIÇÕES DE SERVIÇOS WEB EM CLUSTER DO PORTAL ACADEMICO CELP/ULBRA

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC II) do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e. Jackson Gomes de Souza

Palmas – TO

2019

Talles Alldelamsims da Silva Lopes

BALANCEAMENTO DE REQUISIÇÕES DE SERVIÇOS WEB EM CLUSTER DO PORTAL ACADEMICO CELP/ULBRA

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC II) do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e.Jackson Gomes de Souza

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.ScJ, Jackson Gomes de Souza

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof.M.S.c, Madianita Bogo Marioti

Avaliador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Fábio Castro Araújo

Avaliador

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2019

Dedico este trabalho ao meus pais, que sempre me apoiaram e me corrigiram em todas as minhas decisões.

Também é dedicado a empresa LB-LINK que não mediu esforços para me oferecer todo ambiente necessário para o desenvolvimento desde trabalho.

AGRADECIMENTOS

Primeiramente a Deus por ter me dado sabedoria e entendimento para que eu pudesse manter focado e persistente até o fim dessa etapa da minha vida. A minha mãe que sempre me deu amor, força e conselho nas horas certas, ao meu pai que sempre foi espelho de persistência e caráter.

A empresa LBLink que me deu a oportunidade e recursos para aplicar meus conhecimentos e desenvolver essa monografia. Ao meu orientador Jackson Gomes, por sua paciência e conhecimento que fizeram com que eu me empenhasse cada vez mais. A minha namorada, que sempre me motivou, meus amigos que direto e indiretamente me ajudaram meu eterno AGRADECIMENTO.

RESUMO

Este trabalho visa mostrar a importância de balanceamento de servidores web e técnicas de balanceamento de requisições para que possa garantir alta disponibilidade e alta escalabilidade no portal acadêmico CEULP/ULBRA. A disponibilidade de informações diversas na internet, tem aumentado o número de usuários e consequentemente o tráfego de dados. E com isso é necessário estar preparado para atender esta demanda crescente de processamento e comunicação. Neste sentido, o uso de cluster de servidores web em conjunto com o balanceamento de requisições tem sido uma estratégia importante pelo fato de distribuir os serviços entre vários processadores de forma balanceada. O presente trabalho descreve questões de arquitetura e operação sobre o balanceamento de requisições, abortando técnicas e soluções para melhora do portal acadêmico CEULP/ULBRA.

Palavras-chave: Balanceamento de requisições, Cluster, Servidores web.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Process Unit
DMA	DirectMemory Access
EJB’s	Enterprise JavaBeans
JSP’s	Java Server Pages
Link	Referência a outros documentos dentro de um hypertext
MMU	Memory Management Unit
RMI	Remote Method Invocation
HTTP	Protocolo de transferência de hipertexto
HTTPS	Protocolo de transferência de hipertexto seguro

SUMÁRIO

1. INTRODUÇÃO

A utilização da Internet e o consequente aumento de usuários fizeram com que a Web se tornasse o maior meio de comunicação com acesso a aplicações e serviços remotos. Hoje, a disponibilização de aplicações e serviços Web permite que atividades do dia a dia se tornem mais fáceis e ágeis. Isto produz intensamente tráfego de dados na Internet, exigindo que os grandes provedores de conteúdo e serviços implementem seus servidores Web, para que sejam capazes de fornecer os serviços de forma ininterrupta e lidar com a grande demanda de dados que cresce de acordo com a futura necessidade de desempenho.

Segundo Cardellini (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. Um site popular pode receber milhares de requisições de acesso por segundo, exigindo alta disponibilidade e escalabilidade.

Segundo Oggerino (2001), um sistema possui alta disponibilidade é quando seus serviços continuam funcionando ainda que ocorram falhas em componentes isolados, ou seja, necessárias tarefas de manutenção ou ampliação desse sistema. A alta disponibilidade é uma qualidade de um sistema computacional que supre as necessidades de empresas em manter disponíveis seus serviços e aplicações. É essa característica que indica por quanto tempo os sistemas conseguirão ficar acessíveis aos usuários.

Porém, é necessário mais que disponibilidade para que serviços e aplicações funcionem de forma satisfatória. Há cada vez mais usuários utilizando os serviços disponibilizados na web. Esse aumento constante na demanda exige escalabilidade dos ambientes fornecedores desses serviços, que é a condição de um sistema lidar com uma quantidade crescente de trabalho sem prejudicar as características do produto ofertado (FindUP, 2017).

A disponibilidade e a escalabilidade podem ser obtidas com o balanceamento de requisições entre vários servidores de um cluster. Cluster de servidores Web é a solução mais conhecida para a construção de uma plataforma que ofereça

escalabilidade, trazendo economia de *hardware* e *software* em relação aos grandes servidores de última geração.

Nesse contexto, o trabalho tem como objetivo simular o ambiente atual do portal CEULP/ULBRA em um ambiente vital na plataforma Azure, serão apresentados algoritmos de balanceamento de requisições de serviços web em cluster, com alta disponibilidade e escalabilidade, será abordado como configurar um cenário composto por servidores web para garantir maior *desempenho* para conteúdos estáticos.

.

2. REFERENCIAL TEÓRICO

2.1. SISTEMAS DISTRIBUIDOS

“Sistema distribuído é um conjunto de computadores independentes que aparecem para o usuário do sistema como um único computador. Outra definição para sistema distribuído é: “um sistema composto de vários computadores os quais se comunicam através de uma rede, hospedando processos que usam um conjunto comum de protocolos distribuídos para auxiliar à correta execução de atividades” (TANENBAUM, 1995 apud SILVA, 2005, p. 16).

De acordo com Coulorins (2001 apud Goulart 2002), os sistemas distribuídos possuem seis características principais:

- compartilhamento de recursos;
- expansibilidade;
- concorrência;
- escalabilidade;
- tolerância a falhas;
- transparência.

Compartilhamento de recursos de sistemas distribuídos ressalta o compartilhamento de recursos, disponibilizando informações, dispositivos ou processamento.

Em expansibilidade o sistema pode ser expansível em relação a hardware e software, ou seja, novos recursos podem ser adicionados para o aumento da capacidade tanto de processamento, quando de funcionalidade.

Concorrência em um sistema computacional, algumas funções como a existência de vários processos em um único processador caracteriza uma concorrência. Em um sistema distribuído, onde há mais de um processador, esses processos podem executar em paralelo, delegando funções para cada processador de forma que as instruções passadas retornem com seus resultados.

Escalabilidade está relacionado com o aumento de recursos disponíveis para uma aplicação. Um sistema é escalável se ele permanece ativo mesmo quando há um aumento no número destes recursos e elementos. Em um sistema escalável as

aplicações não precisam ser modificadas quando os recursos forem adicionados, apenas a aplicação passa a usá-los.

Tolerância a falhas em sistemas computacionais, oriundas de hardware ou software, podem tornar o sistema indisponível ou produzir resultados incorretos. Um sistema distribuído deve ser capaz de detectar e corrigir estas falhas.

Transparência é a capacidade que o sistema possui de ficar oculto ao usuário e ao programador da aplicação, o sistema não mostrará a separação de componentes, de forma que o sistema seja percebido como um todo, em vez de um conjunto de componentes.

Os sistemas distribuídos podem ser aplicados em vários contextos, e uma das áreas que está em grande demanda é a distribuição em servidores web.

2.2. SERVIDOR WEB

Um servidor web é um servidor preparado para receber requisições de clientes, em sua grande maioria via browsers. Essas requisições são feitas utilizando-se do protocolo HTTP e HTTPS.

De acordo com (Oliveira, 2016, on-line).

> O servidor web é a peça mais importante da infraestrutura de um site na internet. Ele é um programa que usa o HTTP (*Hypertext Transfer Protocol*) para servir os arquivos que formam páginas da web para os usuários, em resposta aos seus pedidos, que são encaminhadas pelos clientes HTTP de seus computadores.

“A principal atribuição de um servidor web é prestar serviço de armazenamento, processamento e entrega de páginas da web aos clientes. Para tanto, é utilizado o protocolo HTTP como forma de comunicação entre o servidor e seus clientes” (Rodrigues, 2017, on-line).

HTTP é um protocolo que os clientes e os servidores usam para se comunicar, essa comunicação é baseada em requisições (*request)* e respostas (responses) (VARALHO, 2012). O protocolo HTTP é baseado em requisições e

respostas entre clientes e servidores (VIEIRA, 2007). Segundo Vieira (2007), o cliente usando de um navegador ou dispositivo solicita um determinado recurso *resource,* enviando um pacote de informações contendo especificamente uma URL. O servidor recebe estas informações e envia uma resposta, que pode ser um recurso ou um simplesmente outro cabeçalho.

Como o HTTP é um protocolo baseado em texto, ou seja, toda a informação transmitida está em texto, os dados do usuário e do servidor podem ser interceptados ou alterados no meio do caminho. Partindo desse contexto surge o HTTPS.

De acordo com Gonçalves (2019), “O HTTPS é uma extensão segura do HTTP. Os sites que configurarem um certificado SSL/TLS podem utilizar o protocolo HTTPS para estabelecer uma comunicação segura com o servidor”.

Para que os servidores web consigam atender às altas demandas de troca de informação é necessário um balanceamento de carga, para que não haja um alto desempenho durante o acesso.

2.3. BALANCEAMENTO DE CARGA

O “Balanceamento de carga é a técnica de distribuir a carga de trabalho de maneira uniforme entre computadores, a fim de aperfeiçoar recursos e garantir maior desempenho, e assim tornando altamente disponíveis e adaptáveis” (TECH NET, 2015). Essa é uma solução para altos tráfegos de acesso, que utiliza algoritmos inteligentes a tecnologia para distribuir requisições ou processos que chegam entre computadores do sistema, para evitar a sobrecarga. A Figura 1 ilustra um exemplo deste balanceamento.

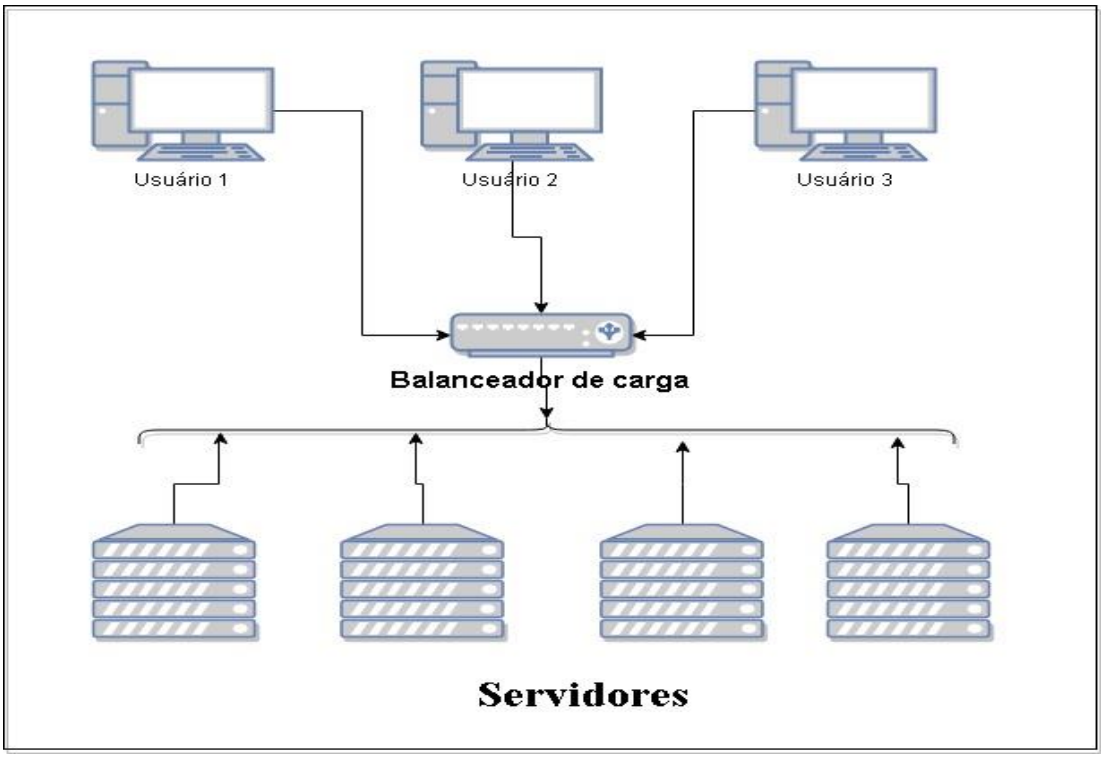
2.3.1. Finalidade dos balanceamentos de carga

Kopparapu (2002) classifica os balanceadores de carga em quatro tipos, de acordo com sua finalidade:

- balanceamento de carga em servidores;
- balanceamento de carga em servidores globais;
- balanceamento de carga em *firewall;*
- *transparente* cache *switching.*

O Balanceamento de carga em servidores, segundo Nobrega (2013), distribui as conexões entre vários servidores de acordo com o trafego de cada um deles e evita direcionarem essas conexões para máquinas com estado de falha, como ilustrado na Figura 1.

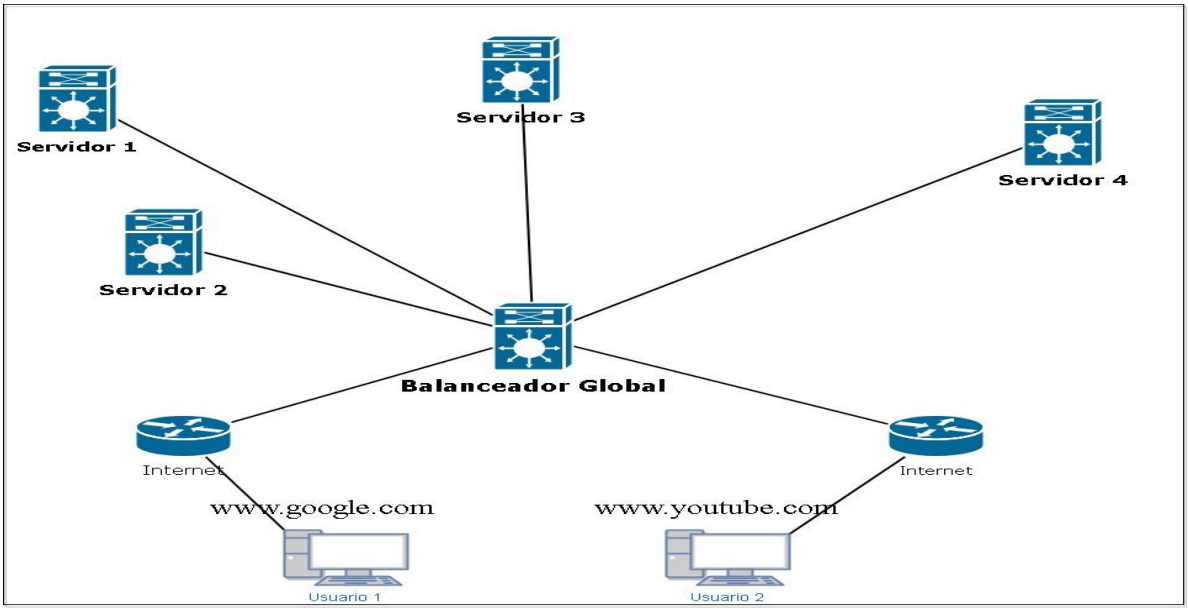
Figura 1 – Balanceamento de carga em servidores.



A Figura 1 ilustra três usuários fazendo requisições a um servidor, chamadas de balanceado em operação, que redireciona as requisições para os demais servidores de acordo com os algoritmos pré-definidos.

Balanceamento de carga em servidores globais (Figura 2), segundo Nobrega (2013), distribui as conexões para servidores espalhados pelo planeta, escolhendo a melhor opção de acesso para o usuário e, teoricamente, apresenta o menor tempo de acesso.

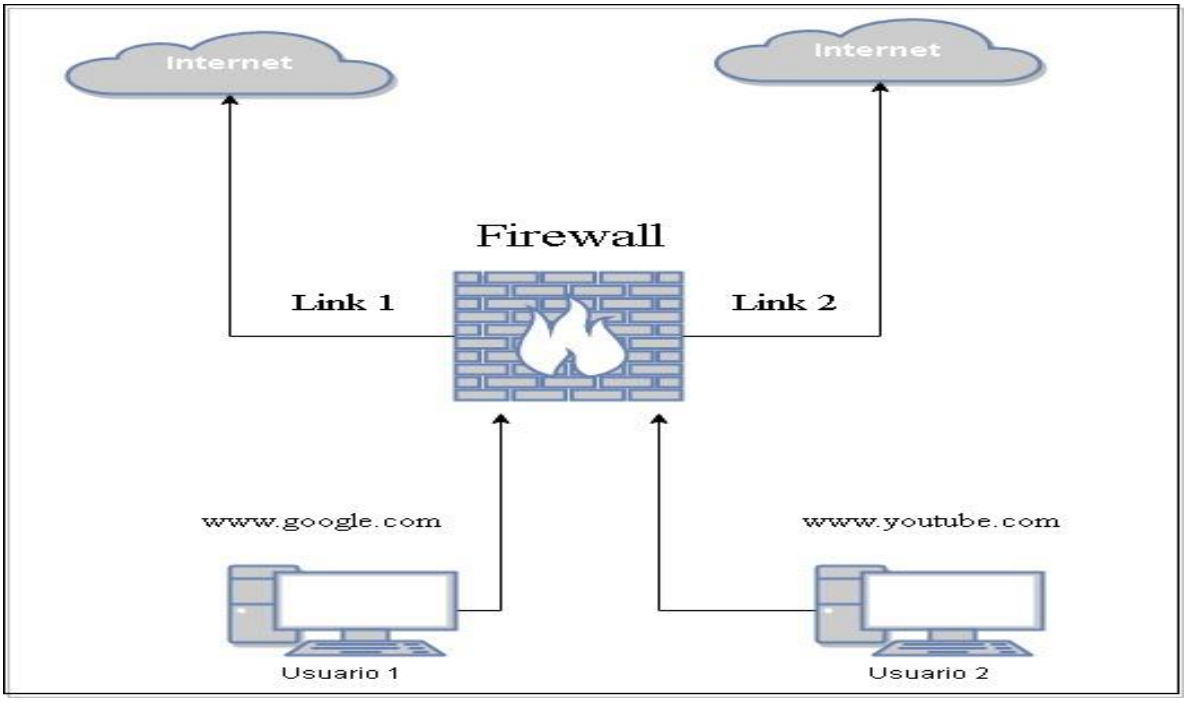
Figura 2 – Balanceamento de carga em servidores globais.



A Figura 2 ilustra dois usuários acessando dois sites distintos, onde a requisição é encaminhada para um balanceador global, que faz o balanceamento de carga para o servidor mais próximo, que no caso mostrado acima é o servidor dois.

Balanceamento de carga em firewalls (Figura 3), segundo Bruno (2013), distribui as requisições entre *firewalls*, evitando que uma única maquina sofra sobrecarga.

Figura 3 - Balanceamento de carga em *firewalls.*



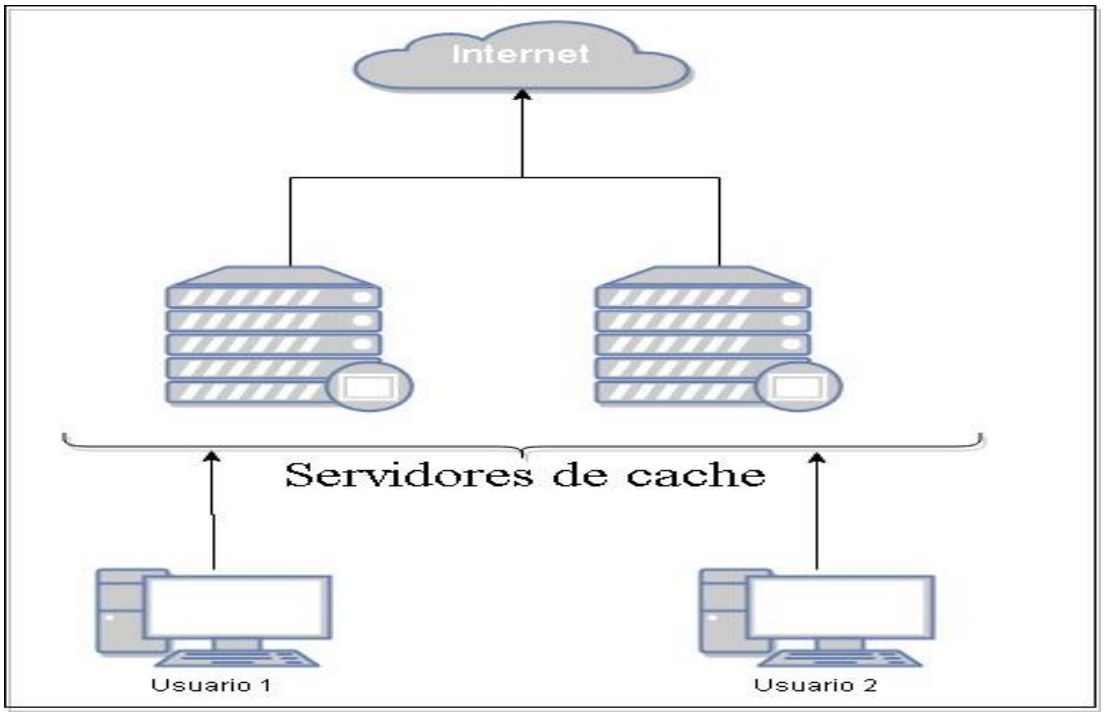
A Figura 3 ilustra dois usuários acessando sites distintos, onde a requisição é encaminha para o *firewall*, que faz o balanceamento de carga gerenciando qual link cada usuário irá acessar.

Transparent cache *switching* (Figura 4), segundo Nobrega (2013), direciona a carga dos computadores de acesso para um dispositivo de acesso rápido, proporcionando uma navegação mais rápida para o usuário.

A Figura 4 ilustra o balanceamento de servidores caches, essa técnica é utilizada para acelerar os pedidos de acesso a um determinado serviço, armazenando as requisições de acesso em cache e disponibilizando-as para próxima vez que o usuário precisar desta mesma requisição.

Além dos quatro tipos de balanceadores de carga apresentados, Leandro (2005) acrescenta “o balanceamento de carga em servidores, que é o uso da distribuição balanceada de requisições em um sistema de servidores Web, usando uma política que distribua a carga recebida de maneira uniforme para todos os servidores, não sobrecarregando um deles enquanto outro está disponível”.

Figura 4 - *Transparent* cache *switching.*

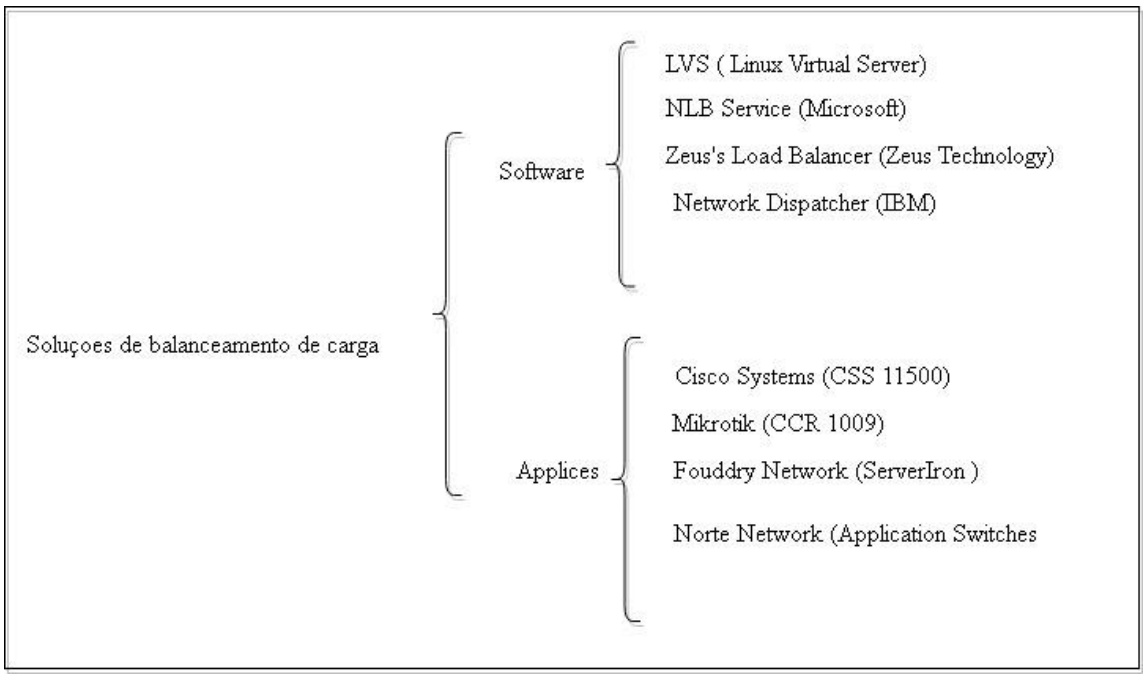


2.3.2. Soluções de balanceamento de carga

O número de aplicações para internet tem aumentado muito atualmente, com esse crescimento instituições educacionais e indústrias tem se preocupado com soluções para balanceamento de carga eficiente.

Segundo Schlossnagle (2007), existem duas soluções de se fazer o balanceamento de carga: via *appliances* (hardwares* que programam funcionalidades específicas na rede) ou via software.

Figura 5 - Soluções de balanceamento de carga.



A Figura 5 demonstra dois tipos de soluções para balanceamento de carga que são:

- ***Softwares****:* algoritmos executados em servidores genéricos de balanceamento de carga coordenam a distribuição dos dados no ambiente;
- ***Appliances****:* são produtos que incluem o software e o hardware necessários para efetuar a distribuição da carga em ambiente web. Em geral são computadores com um hardware e sistema operacional customizado para este fim.

2.3.3. Algoritmos de balanceamento de carga.

A técnica de balanceamento de carga escolhida deve ser ao mesmo tempo simples e eficiente. Para isso, pode-se escolher entre uma variedade de algoritmos que se adequam melhor a diferentes situações. Os algoritmos têm como objetivo distribuir as requisições de forma a maximizar a utilização de todos os servidores disponíveis. Entre os mais usados estão: Round-Robin, Round-Robin ponderado e Menor Carga.

De acordo com Silva:

> Um algoritmo de balanceamento de carga deve ser escolhido de modo que não deteriore ainda mais o desempenho do serviço. O uso de despachante introduz mais uma entidade entre o consumidor e o provedor. Por isso, se ele não for bem planejado e implementado, pode trazer mais problemas do que soluções, inclusive a completa indisponibilidade do serviço (2005, p.66).

De acordo com a (IBM, 2012) “[...] o algoritmo *round robin* mantém uma lista de servidores e encaminha uma nova conexão para o próximo servidor na lista de membros”.

Esse método usa a técnica de sempre direcionar as requisições para o próximo servidor disponível de uma forma circular. Por exemplo, as conexões de entrada são dirigidas para o servidor um depois servidor dois e assim sucessivamente, de acordo com a quantidade de servidores que tiver a estrutura, e depois retorna ao servidor um.

IBM (2012) Define o algoritmo *round robin* ponderado como:

> Uma lista ponderada de servidores que encaminha as novas conexões proporcionalmente ao peso ou à preferência de cada servidor. Este algoritmo usa mais tempos de cálculo do que o algoritmo round Robin básico. No entanto, o cálculo adicional resulta em distribuir o tráfego de modo mais eficiente para o servidor que for mais capaz de manipular a solicitação.

Segundo a IBM (2012), “[...] o algoritmo mínimo de conexões ou menor carga mantém um registro de conexões do servidor ativas e encaminha uma nova conexão para o servidor com o menor número de conexões ativas”. O servidor com a menor carga tem maior possibilidade de ser selecionado para receber as requisições. Por exemplo, se o servidor X está com 20 conexões e o servidor Y está 19, a próxima requisição será direcionada para o servidor Y.

2.4. CLUSTER

Segundo Gropp, Lusk e Sterling (2003) *cluster* é um computador paralelo construído com componentes e que rode software. Um cluster é composto de nós, cada um contendo um ou mais processadores, memória que é compartilhada por todos os processadores do nó e dispositivos periféricos adicionais, conectados a uma rede que permita que os dados transitem entre os nós.

De acordo com Sloan (2004):

> O “clusters” e “computação de alto desempenho” eram sinônimos. Hoje o significado de cluster expandiu, deixando de ser apenas alto desempenho, incluindo clusters de alta disponibilidade (HA) e clusters de balanceamento de carga (LB). Entretanto estes tipos de clusters não são separados entre si. (2004).

A seguir serão apresentados os principais tipos de clusters.

2.4.1. Cluster de Alto Desempenho

Este tipo de cluster também é conhecido como Cluster de Alta *Performance,* pois tem como objetivo utilizar processamento de um determinado trabalho entre diversos processadores paralelamente, de diferentes computadores, fazendo com que a capacidade de processamento seja dividida. Assim, é possível aumentar a capacidade de processamento.

2.4.2. Cluster de Alta Disponibilidade

Nos cluster de disponibilidade o foco é manter os serviços ou servidores sempre ativos, mantendo o sistema sempre operacional.

> Clusters de alta disponibilidade, também conhecidos como *High Avaiability* Clusters e *failover* clusters, são normalmente utilizados em aplicação crítica. Um cluster de alta disponibilidade é formado por múltiplos computadores que podem prover um serviço apropriado. Em caso de falha no computador primário, um computador secundário passará a prover os serviços (Eduardo, 2011, p.5).

O principal benefício de um cluster de alta disponibilidade é que sempre haverá uma redundância para suprir com as possíveis falhas futuras.

2.4.3. Cluster de Aplicação web

Em clusters de aplicações web é comum a divisão dos serviços lógicos em camadas, principalmente em soluções corporativas. Esta separação é útil porque muitas vezes pretende-se criar níveis de clusters de acordo com o conteúdo que cada um deles vai retornar (NOBREGA, 2013, p.31).

Segundo Gorino (apud NOBREGA, 2013, p.32), as camadas mais populares são:

- **camada web**: fornece conteúdo web estático, como páginas HTML, imagens, folhas de estilo, scripts, entre outros. É geralmente o primeiro ponto de contato entre os clientes e a aplicação;
- **camada de apresentação**: retorna conteúdo dinâmico para os clientes. Em aplicações Java web, por exemplo, podem ser servlets e JSP's. Pode englobar a camada web, caso responda também por conteúdo estático;
- **camada de objetos:** fornece objetos Java, como EJB's, classes RMI e webservices e sua lógica de negócio associada. "Ao separar esta camada, muitas vezes se evita a duplicação de código encontrada em aplicações de uma mesma empresa, já que tudo fica centralizado em um único ponto". Esta divisão de responsabilidades da aplicação web permite que tenhamos mais de um cluster envolvendo a mesma aplicação. Desta forma, é possível adicionarmos camadas ao sistema, conforme pode ser visto a seguir.

2.4.4. Cluster de servidores web

O aumento contínuo de serviços e aplicações Web tem causado um crescimento nos problemas de desempenho no acesso ao seu conteúdo, como altos tempos de resposta, congestionamento na rede e negação e indisponibilidade de serviços. Além disso, atualmente, a maioria dos conteúdos acessados na Web é dinâmica. Conteúdo dinâmico é aquele conteúdo que o navegador envia uma solicitação para o servidor web e o servidor responde com uma página solicitada de maneira exclusiva para o usuário (CULTURA MIX, 2015).

Esse tipo de conteúdo aumenta a demanda nos servidores, principalmente para operações de entrada e saída e de CPU. Isso faz com que o servidor, e não

mais a rede e a sua largura de banda, se torne o ponto mais crítico no acesso a sites Web.

Leandro (2005, p.14) afirma que, “para solucionar esse problema, adquirir uma máquina mais poderosa para o servidor Web nem sempre é ideal, tanto pela pouca escalabilidade quanto pelo alto custo”. Uma solução para dar suporte a esse aumento de processo seria utilizar cluster, de processadores que trabalham conjuntamente, servindo como um único servidor Web.

> Cluster de servidores é a junção de duas ou mais máquinas**,** interligadas, que permitem maior disponibilidade e escalabilidade. As máquinas que compõem o cluster são chamadas de nó. Os servidores são interligados através de cabos físicos e de um software de gerenciamento. Caso ocorra falhas em um dos servidores, o serviço é automaticamente assumido por outro servidor (LEANDRO, 2005, p.22).

Essa solução possibilita que múltiplas máquinas trabalhem como um único servidor, tornando o sistema mais poderoso, pois os recursos de todos os computadores que formam o cluster são compartilhados para atender às requisições dos clientes. Entre as vantagens existentes na utilização de um cluster de servidores Web encontram-se:

- **aumento de desempenho**: há aumento na capacidade de receber solicitações;
- **melhor confiabilidade e disponibilidade:** o servidor pode continuar operando mesmo se uma das máquinas no cluster falhar. A disponibilidade é provida através de redundância de servidores. Quando a falha em algum servidor, esta falha não é visível ao usuário, pois as requisições são enviadas a outros servidores ativos, fazendo com que o serviço fique sempre ativo.
- **tolerância a falhas:** Capacidade de fazer modificações nos servidores sem perder a disponibilidade dos serviços em operação;
- **escalabilidade facilitada:** é simples aumentar a capacidade de um cluster pela adição de novos servidores; e
- **interface externa**: é única para todos computadores do cluster; assim, alterações na configuração do cluster não afetam as aplicações clientes.

Esses conjuntos de vantagens no uso de cluster de servidores web permitem aos profissionais da área de TI maior facilidade no gerenciamento dos servidores, manutenção e remoção de um servidor nos momentos de paralização ou sobrecarga.

3. MATERIAIS E MÉTODOS

3.1. MATERIAIS

Para realização do presente trabalho foram usados diversos materiais, técnicas e ferramentas, os quais são apresentados por esta seção que os descreve e informa como foram utilizados.

A plataforma Azure foi utilizada para a criação dos **ambientes virtuais centralizado e distribuído**. O GTmetrix foi utilizado para verificar o desempenho da página. Para o teste de estresse e simulação de acesso foi utilizada a ferramenta JMeter. Além disso, foi utilizado o servidor web Apache[1] e a plataforma de gerenciamento de conteúdo via web Wordpress[2].

3.1.1. GTmetrix

O GTmetrix foi desenvolvido pelo GT.net como uma ferramenta para os clientes de hospedagem gerenciada determinarem facilmente o desempenho de seus sites (GTMETRIX, 2018).

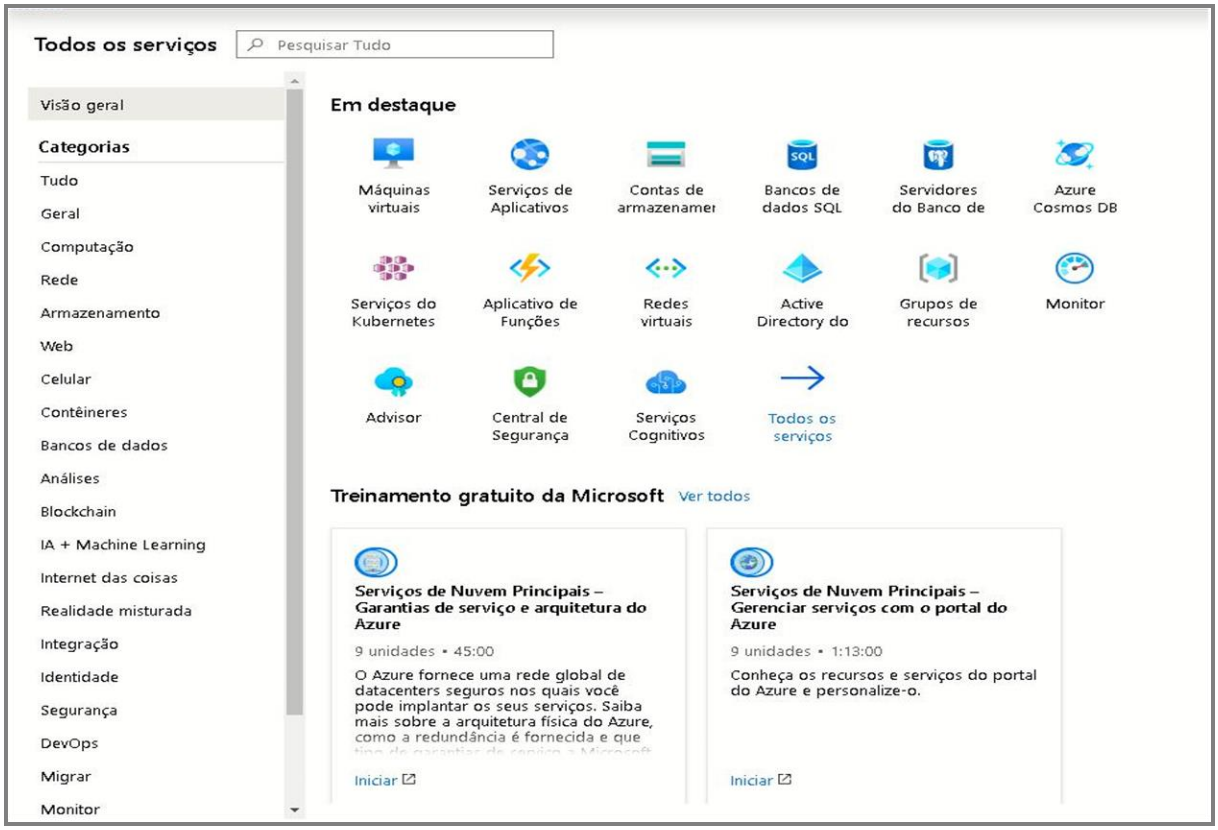
Esta ferramenta foi utilizada para verificar o desempenho do Portal Acadêmico CEULP/ULBRA, mostrar tempo de carregamento da página e identificar possíveis problemas como CSS, JavaScript e imagens.

3.1.2. Microsoft Azure

A Microsoft Azure é uma plataforma provedora de aplicativos e serviços de Infraestrutura de Computação em Nuvem, focada para desenvolvedores de software e demais serviços que buscam agilidade e segurança para seus produtos. Tem como objetivo oferecer armazenamento e serviços escaláveis, com alta disponibilidade e interface de gerenciamento simplificada. A Figura 6 apresenta uma das telas da ferramenta de gerenciamento dos serviços e ilustra alguns dos recursos disponíveis.

[1] Disponível em:< https://www.apache.org/> Acesso em: 16 de novembro de 2019.
[2] Disponível em:< https://br.wordpress.com/> Acesso em: 16 de novembro de 2019.

Figura 6 – Portal Azure.

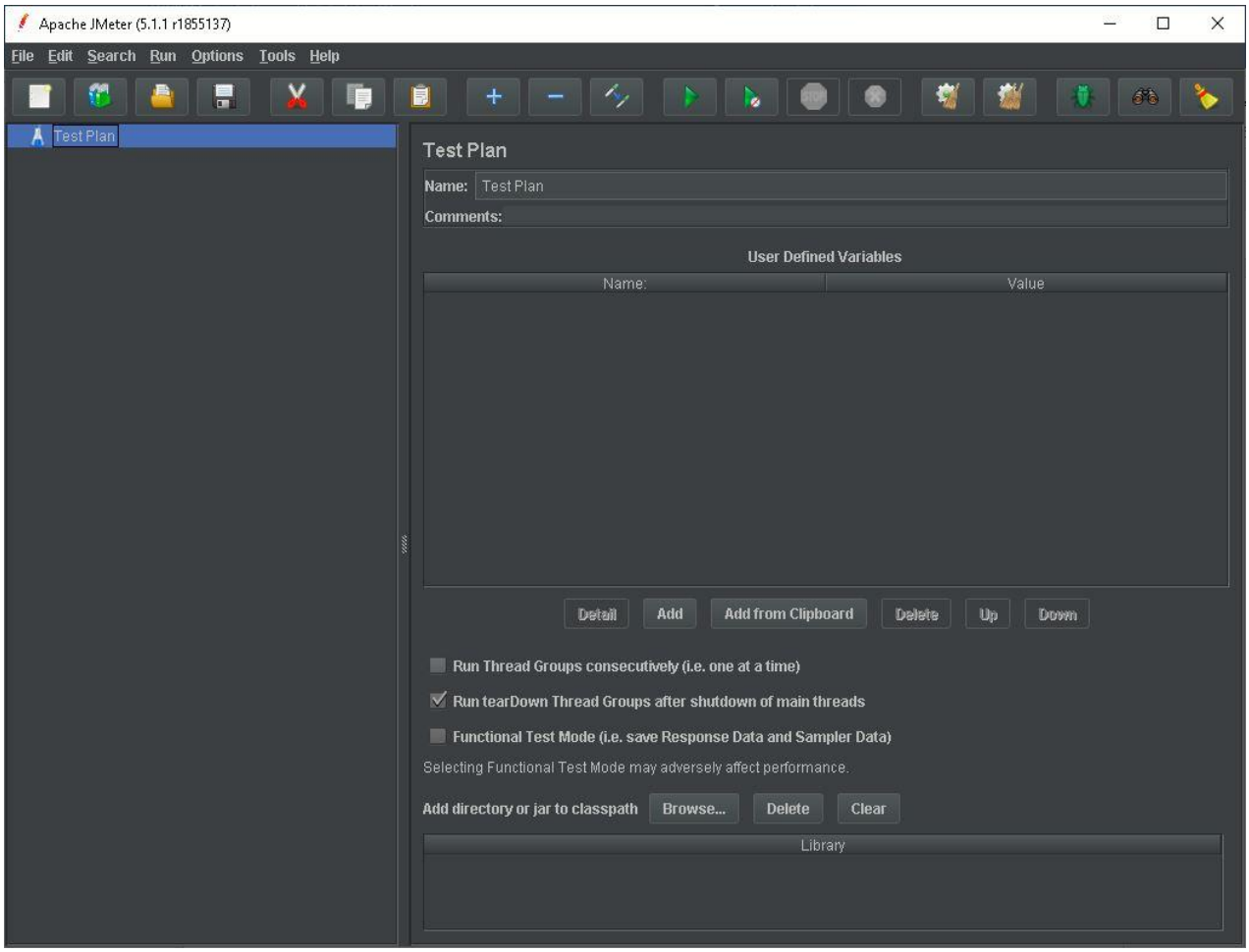


A plataforma Azure foi utilizada para a criação de dois ambientes com implantação dos servidores virtuais, um serviço de banco de dados e para os serviços web do Portal Acadêmico CEULP/ULBRA.

3.1.3. ApacheJmeter

O Apache *JMeter* é um software de código aberto, um aplicativo Java 100% puro projetado para executar o comportamento funcional do teste e medir o desempenho, originalmente projetado para aplicativos da Web (JMETER, 2018). A Figura 7 apresenta a janela do *JMeter* e ilustra alguns dos seus recursos.

Figura 7 – Tela inicial JMeter.



O *JMeter* foi usado para simular requisições nos servidores web do Portal Acadêmico CEULP/ULBRA, com o intuito de submeter os servidores a uma avaliação de stress para avaliar se os resultados estão de acordo com o esperado, garantindo assim a qualidade do portal.

3.2. MÉTODOS

O presente trabalho foi conduzido por meio de execuções dos seguintes processos:

1. **Planejamento**: planejamento para mensurar e estruturar o ambiente a ser utilizado, envolvendo as etapas:
    a. **Estrutura tecnológica**: planejamento e a adequação das tecnologias utilizada durante o projeto, topologia, inserção dos

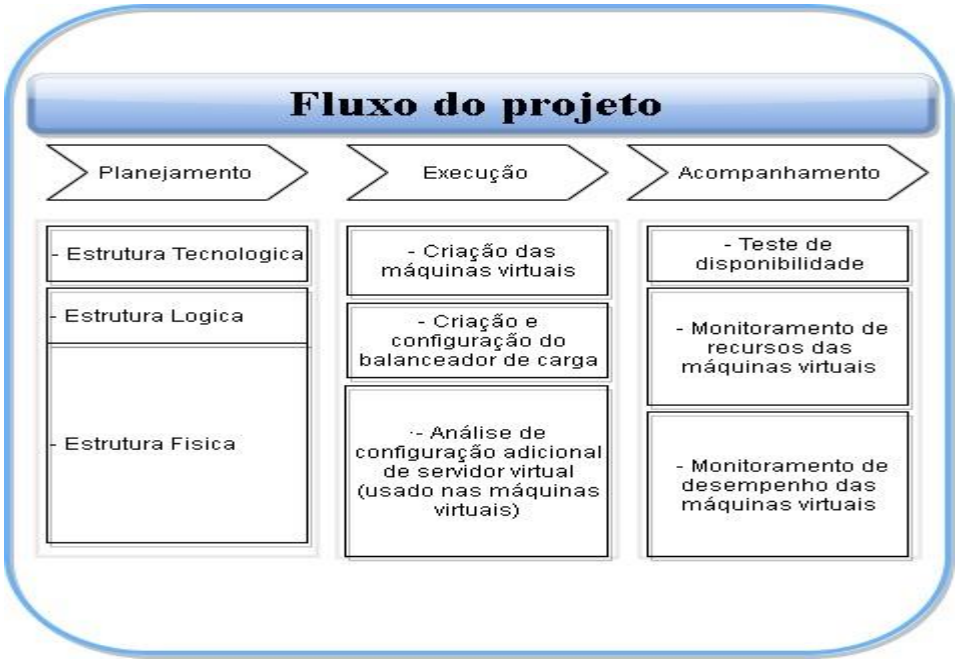
recursos e ferramentas virtuais, sistemas operacionais, aplicações web, e os demais softwares;

b. **Estrutura lógica**: definição de conexões, links de internet;
c. **Estrutura física**: estrutura do ambiente, servidores, roteadores;

2. **Execução do planejamento**: envolve as etapas:
   a. criação das máquinas virtuais;
   b. criação e configuração do balanceador de carga;
   c. configuração adicional de servidor virtual (usado nas maquinas virtuais);
3. **Acompanhamento das funcionalidades envolvidas:** envolve as etapas:
   a. testes de disponibilidade;
   b. monitoramento de recursos das máquinas virtuais; e
   c. monitoramento de desempenho das máquinas virtuais.

Para melhor entendimento os processos serão detalhados a seguir, começando pela Figura 8, que descreve o fluxo de implantação do projeto e suas etapas.

Figura 8 – Fluxo de implantação do projeto.



No **Planejamento** foi feito um estudo da capacidade de requisições simultâneas que os ambientes suportariam. Isso foi analisado por meio da

porcentagem de uso dos recursos computacionais dos ambientes (CPU, RAM e HD), cujos valores foram obtidos a partir de testes por meio do *JMeter*. Nesse sentido, os testes consideraram 1.000 (mil) usuários para metrificar o desempenho de acesso e estresse dos ambientes.

A topologia planejada foi a de dois ambientes:

a) **centralizado**: contendo um servidor virtual ligado a um serviço de banco de dados; e

b) **decentralizado**: com dois servidores virtuais ligados a um serviço de banco de dados.

Para a medição de desempenho dos servidores foi escolhido o próprio painel de monitoramento da plataforma *Azure*, que analisa e disponibiliza dados referentes ao tempo de execução dos processos executados, consumo de memória RAM, média de uso dos discos, média de uso da CPU e tráfego de rede.

O *JMeter* foi escolhido para analisar as requisições entre cliente e servidor(es), usando o gráfico agregado, resultados em tabela e árvore de resultado, o tempo de resposta de cada uma, e a quantidade de rejeições durante a tentativa de acesso.

As aplicações *Apache*, *PHP* e *Wordpress* foram instaladas nos servidores Debian contidos dois ambientes, para fins de simulação do cenário do portal acadêmico CEULP/ULBRA.

Na **Execução** do projeto, que é o processo de realização do trabalho definido no plano de gerenciamento do projeto para atingir os objetivos, foram criados os dois ambientes virtuais, como descrito no Planejamento. As máquinas virtuais foram criadas na plataforma *Azure*, onde foi criado um grupo de recurso para organização do projeto. Os dois ambientes, com as ferramentas e serviços necessários para execução do projeto, foram implantados no grupo de recursos.

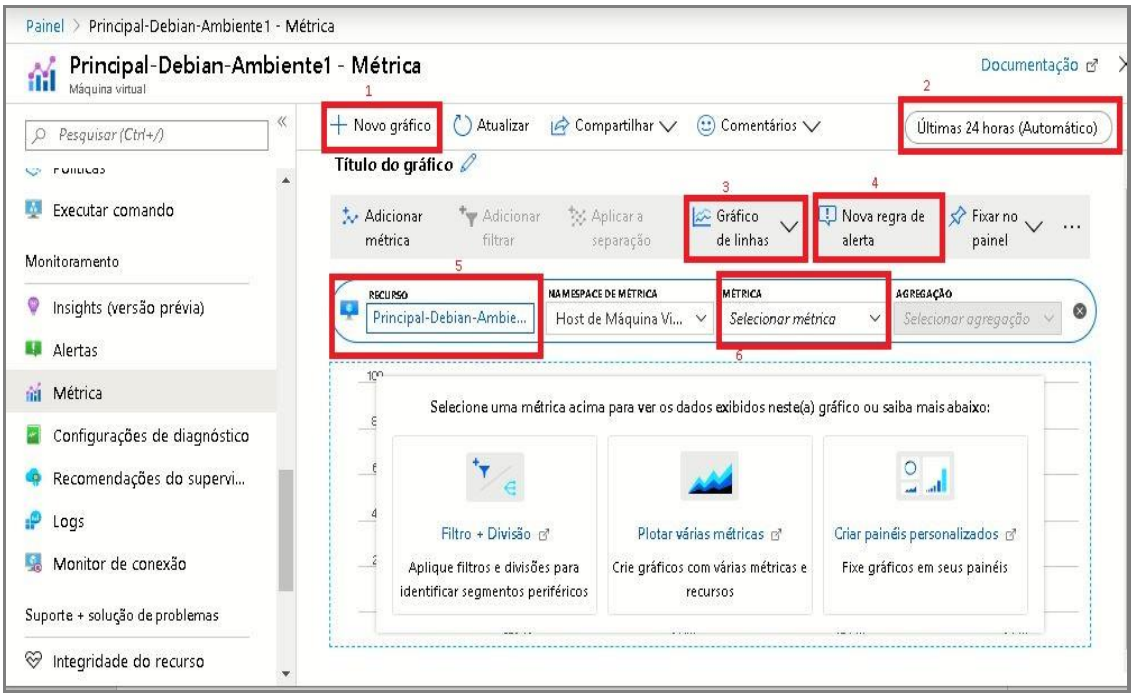
No primeiro ambiente foi criada uma máquina virtual principal alocada na região sul do Brasil, com suas devidas configurações de rede, sistema operacional, endereço de IP privado, público e configurada para acessar um serviço de banco de dados, já disponibilizado pela Azure.

No ambiente descentralizado foram criadas duas máquinas virtuais: uma primária e outra secundaria, alocadas na região Oeste da Europa, com suas devidas configurações de rede, sistema operacional, endereço de IP privado, público e configuradas para acessar um serviço de banco de dados, já disponibilizado pela Azure.

Foi criado um balanceador de carga do tipo *Round-Robin* para fazer a distribuição das requisições entre os servidores e execução de disponibilidade do ambiente descentralizado. Através de um único IP público o balanceamento distribui todas as requisições entre os servidores e, caso ocorram erros ou falhas, o balanceador se encarrega de enviar as requisições para o servidor em funcionamento.

No **Acompanhamento** dos processos envolvidos foram monitorados, coletando informações que dizem respeito à execução de uma dada atividade. Foram realizados testes de disponibilidade no ambiente descentralizado, onde manualmente foi desligada uma das máquinas e verificada a continuidade dos serviços. No monitoramento de recursos e desempenho das máquinas virtuais foi utilizado o próprio painel de métricas da Azure (Figura 9), que disponibiliza dados baseado por período de horas ou dias.

Figura 9 – Painel de métricas Azure.



A figura 9 ilustra 6 principais itens destacados para a criação de um gráfico personalizado para análise de recursos, com as seguintes características:

1. **Novo gráfico:** Permite que você crie vários gráficos na mesma tela.
2. **Período:** Permite definir o período de monitoramento em horas ou dias.
3. **Estilo do gráfico:** Exibe o gráfico em linhas, barra, dispersão ou grade.
4. **Regra de alerta:** Permite criar regras de alertas e soluções de reparo pra um determinado recurso. Ex: (Consumo de CPU em 80, gera um alerta de alto consumo).
5. **Recurso:** Permite selecionar o tipo de recurso que será monitorado. Ex: (Máquina virtual, balanceador, banco de dados).
6. **Métrica:** Permite selecionar o tipo de métrica de um determinado recurso que será monitorado. Ex: (CPU, RAM, HD, REDE).

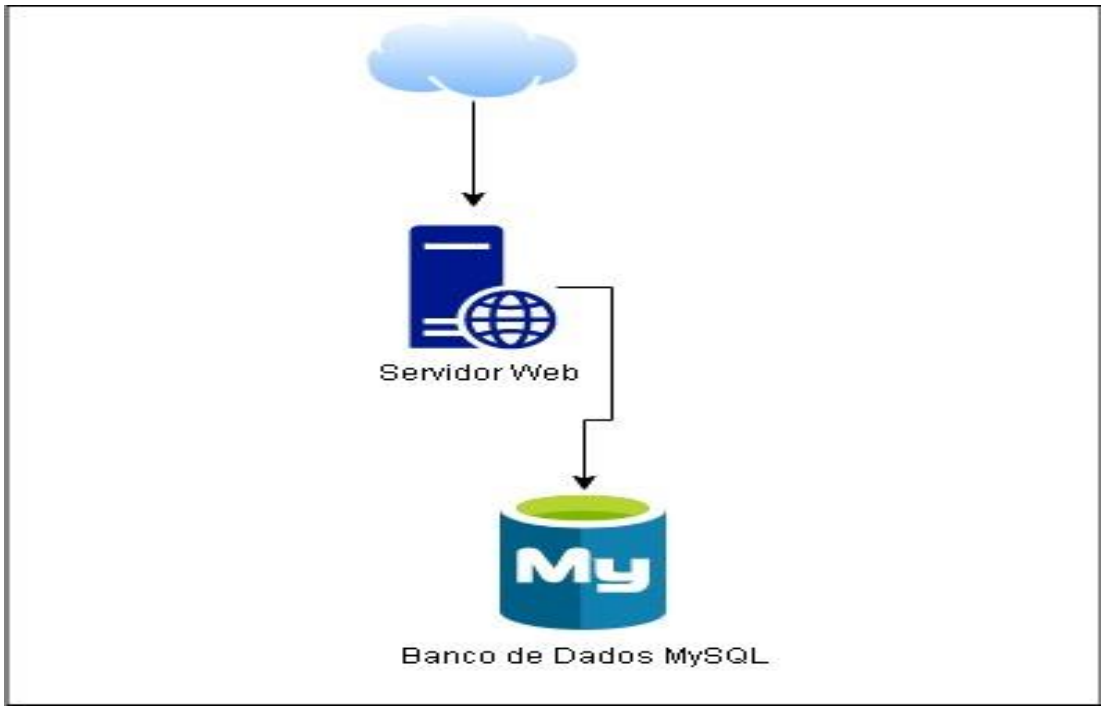
4. RESULTADOS E DISCUSSOES

Este capitulo demostra os resultados obtidos através do balanceamento de requisições, de serviços web em cluster do portal acadêmico CELP/UBRA, onde as informações obtidas foram adquiridas por meio de dois ambientes, um centralizado com uma máquina virtual e um serviço banco dados e outro decentralizado com duas maquinas virtuais e um serviço banco de dados.

4.1. AMBIENTE CENTRALIZADO

O ambiente centralizado foi criado na plataforma virtual Azure para simular a estrutura do ambiente atual do portal acadêmico CELP/UBRA. A figura 10 ilustra o ambiente centralizado.

Figura 10 – Ambiente Centralizado.



A Figura 10 ilustra que ambiente centralizado é composto por: Servidor Web e Banco de dados MySQL. No *Azure*, o Servidor Web é representado por uma máquina virtual cujo hardware tem as características:

1. Processador de 1 núcleo;
2. Memória RAM com capacidade de 2GB;
3. Disco rígido com capacidade de 30GB.

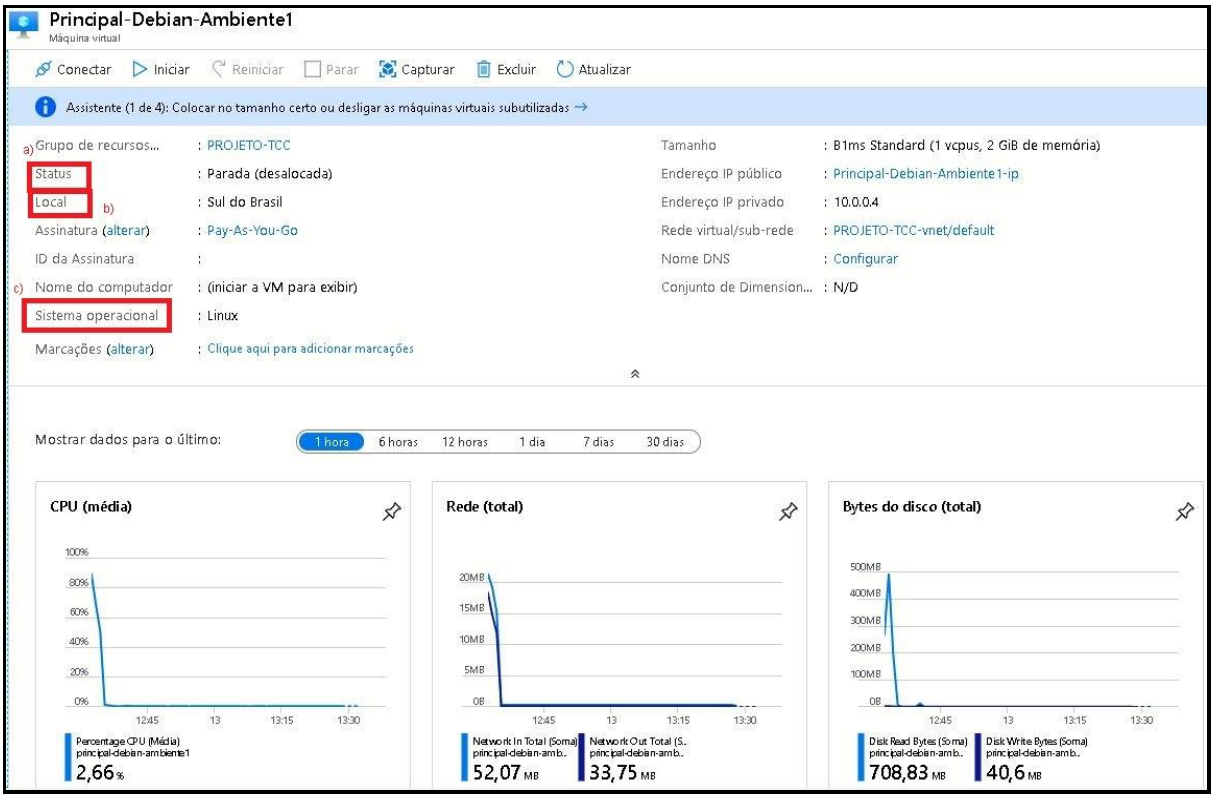
A estrutura tecnológica desta máquina virtual foi caracterizada pelos seguintes itens:

1. Sistema operacional Debian 8;
2. Apache 2.5;
3. PHP 5; e
4. WordPress;

O banco de dados *MySQL* ilustrado é um serviço oferecido pela *Azure,* que já vem as configurações de *hardware* pré-configuradas para o ambiente desejado, e necessitando apenas do preenchimento do nome do banco, usuário e senha.

A Figura 11 ilustra a tela da visão geral da máquina virtual.

Figura 11 – Máquina virtual primaria ambiente Centralizado.



A visão geral da máquina virtual, como ilustrado pela Figura 11, indica características, como:

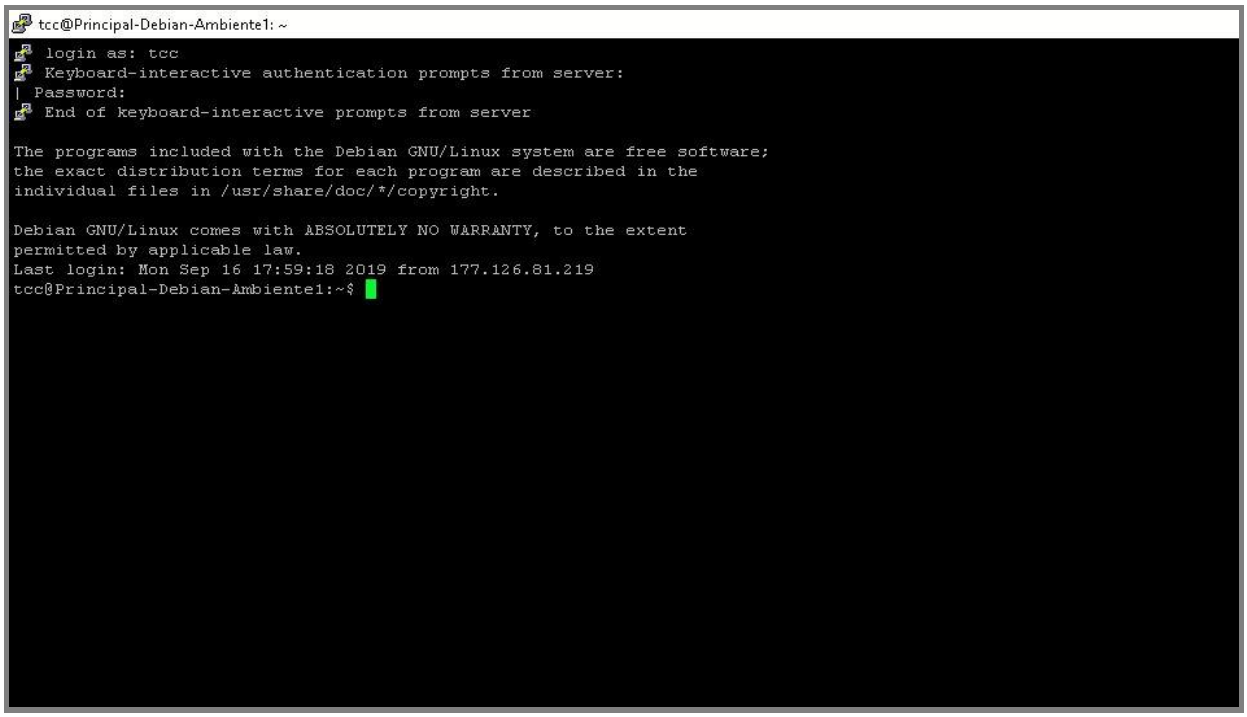
a) Status (Parada)
b) Local (Sul do Brasil)
c) Sistema operacional (Linux Debian 8.11)
d) Endereço IP Público (191.232.164.234)

Além disso, a figura apresenta gráficos de monitoramento dos recursos (ex.: CPU e Rede) que podem ser mostrados em um período a escolha do utilizador (ex.: 1 hora ou 7 dias).

Conforme mostrado na figura 11, o ambiente centralizado contém um servidor web centralizado que tem como funcionalidade simular o recebimento de todas as requisições solicitadas através da aplicação *WordPress.*

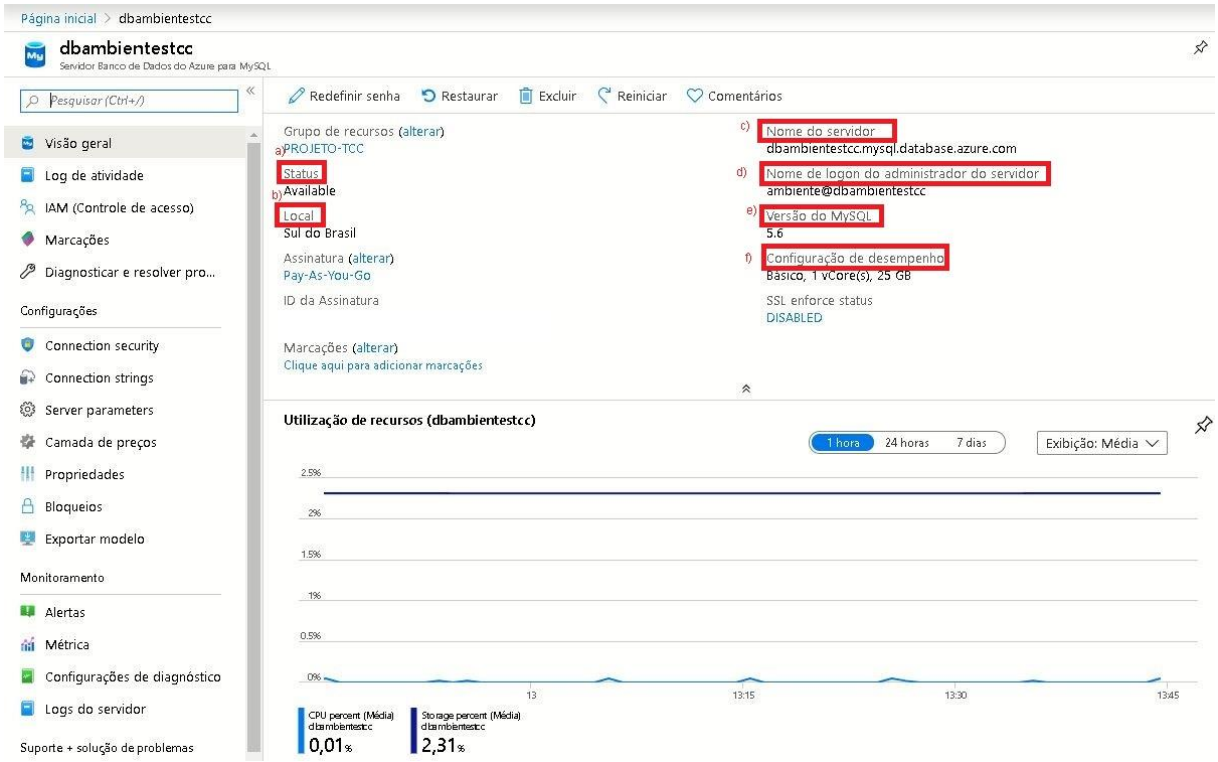
Na imagem 12 é mostrado o acesso externo ao servidor web via aplicação *Putty*.

Figura 12 – Acesso a máquina virtual ambiente centralizado.



Como forma de acesso ao servidor web foi escolhido o software de emulação *Putty, como* mostra a figura 12. Foi instalado o servidor de SSH no servidor web para permitir o acesso a máquina virtual e fazer as devidas alterações, instalações e configurações dos serviços web que foram utilizados neste trabalho. Ainda, no mesmo ambiente foi configurado o serviço de banco de dados disponibilizado pela Azure como demonstra a figura 13.

Figura 13 – Serviço de banco de dados MySql.



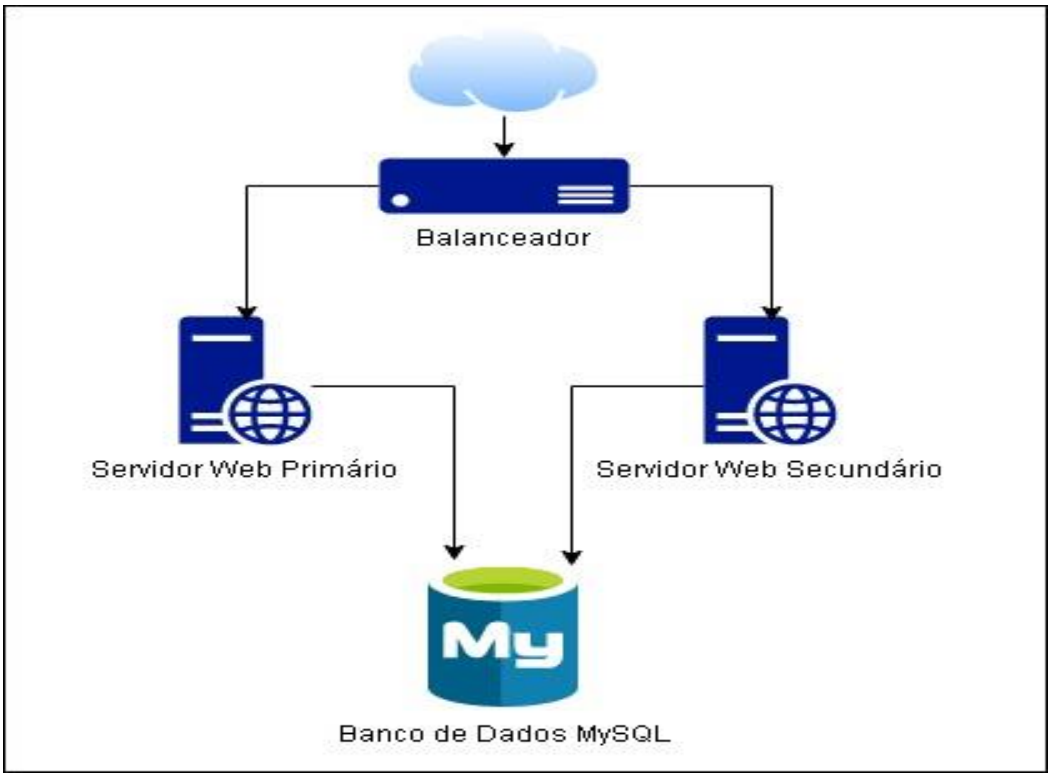
A visão geral do serviço de banco de dados ilustrado na figura 13, apresenta características como:

a) Status (Avaliable)
b) Local (Sul do Brasil)
c) Nome do Servidor (dbambientestcc.mysql.database.azure.com)
d) Nome de logon (ambiente@dbambientestcc)
e) Versão do MySQL (5.6).
f) Configuração de desempenho (Básico, 1 vCore(s), 25 GB)

4.2. AMBIENTE DESCENTRALIZADO

O ambiente descentralizado também foi criado na plataforma virtual Azure para simular o ambiente do portal acadêmico CELP/UBRA. Porém utilizando-se de recurso de balanceamento em cluster. A figura 14 ilustra a topologia deste ambiente.

Figura 14 – Topologia Descentralizada.



O ambiente mostrado na figura 14 foi criado com um balanceador, dois servidores web virtuais e um serviço de banco de dados com as seguintes características:

1. Processador de 1 núcleo;
2. Memória RAM com capacidade de 2GB;
3. Disco rígido com capacidade de 30GB.

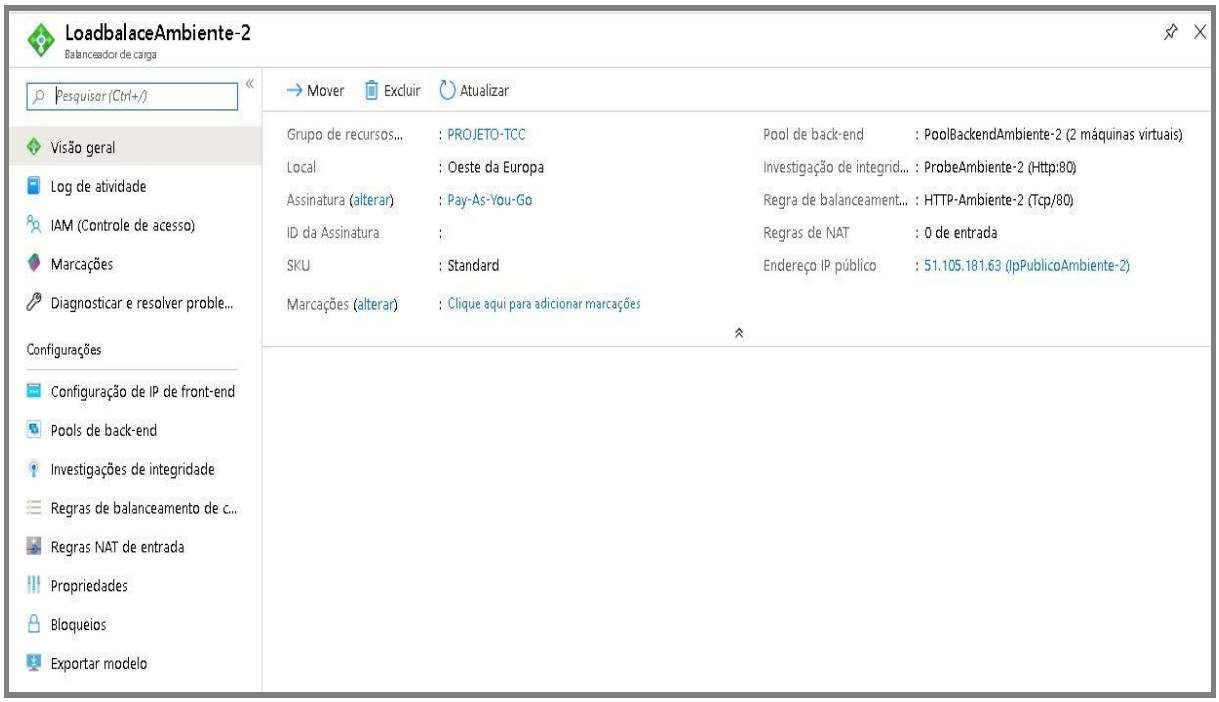
A estrutura tecnológica desta máquina virtual foi caracterizada pelos seguintes itens:

5. Sistema operacional Debian 8;
6. Apache 2.5;
7. PHP 5; e

8. WordPress;

A figura 15 ilustra o balanceador de carga, que é o responsável pelo balanceamento das requisições que são solicitadas ao servidor web. O balanceador recebe por padrão um IP público que receberá as solicitações. Quando o pedido é feito ao IP público, que também poderá ser feita através uma publicação DNS, o pedido é distribuído ao cluster de servidores de forma alternada e sem prioridades.

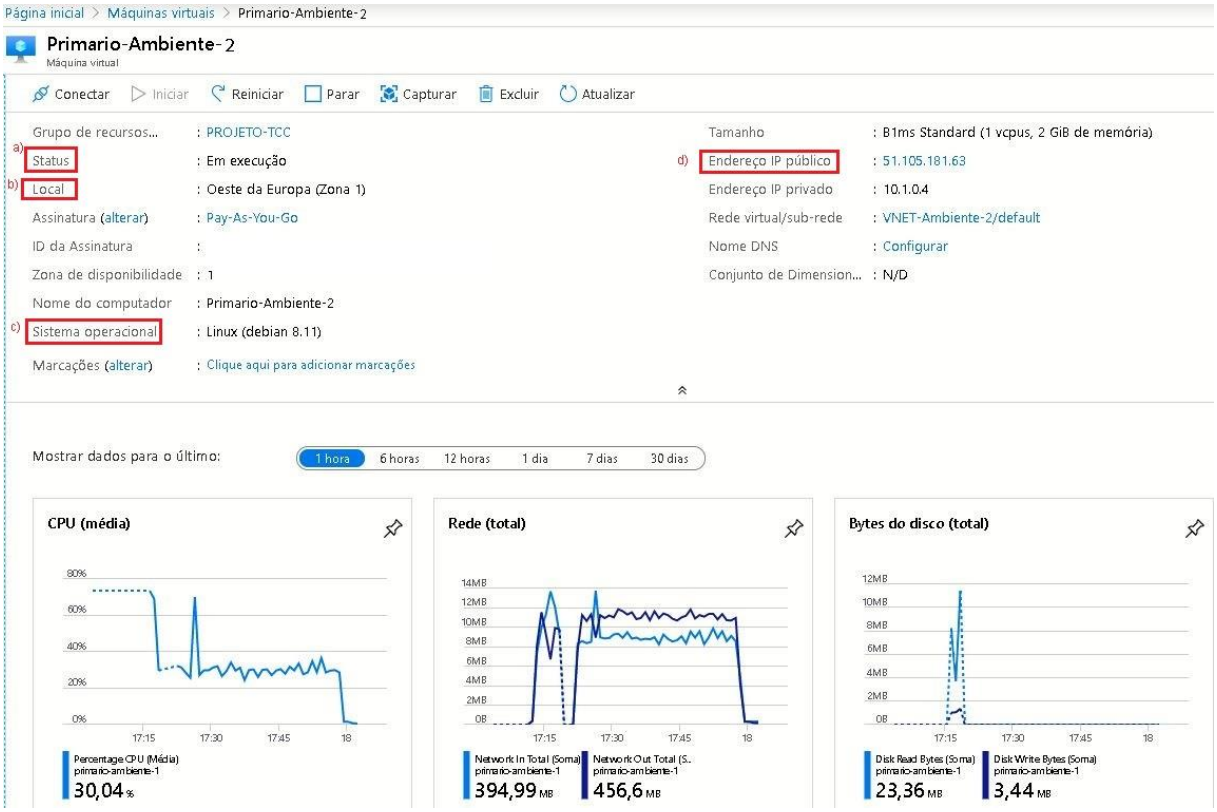
Figura 15 – Balanceador de carga.



O método responsável por alternar as requisições entre os servidores é chamado de Round-Robin, que além de distribuir as requisições, permite a alta disponibilidade, caso ocorra a perda de um dos servidores web, todas as requisições são automaticamente encaminhadas ao servidor ativo. Este método é disponibilizado pela plataforma *Azure* através da configuração de um perfil de roteamento e definindo o método de distribuição (**roteamento ponderado round-robin**).

A Figura 16 ilustra a tela da visão geral da máquina virtual primaria do ambiente descentralizado.

Figura 16 – Máquina virtual primaria ambiente descentralizado.



A visão geral da máquina virtual primária, como ilustrado pela Figura 16, indica características, como:

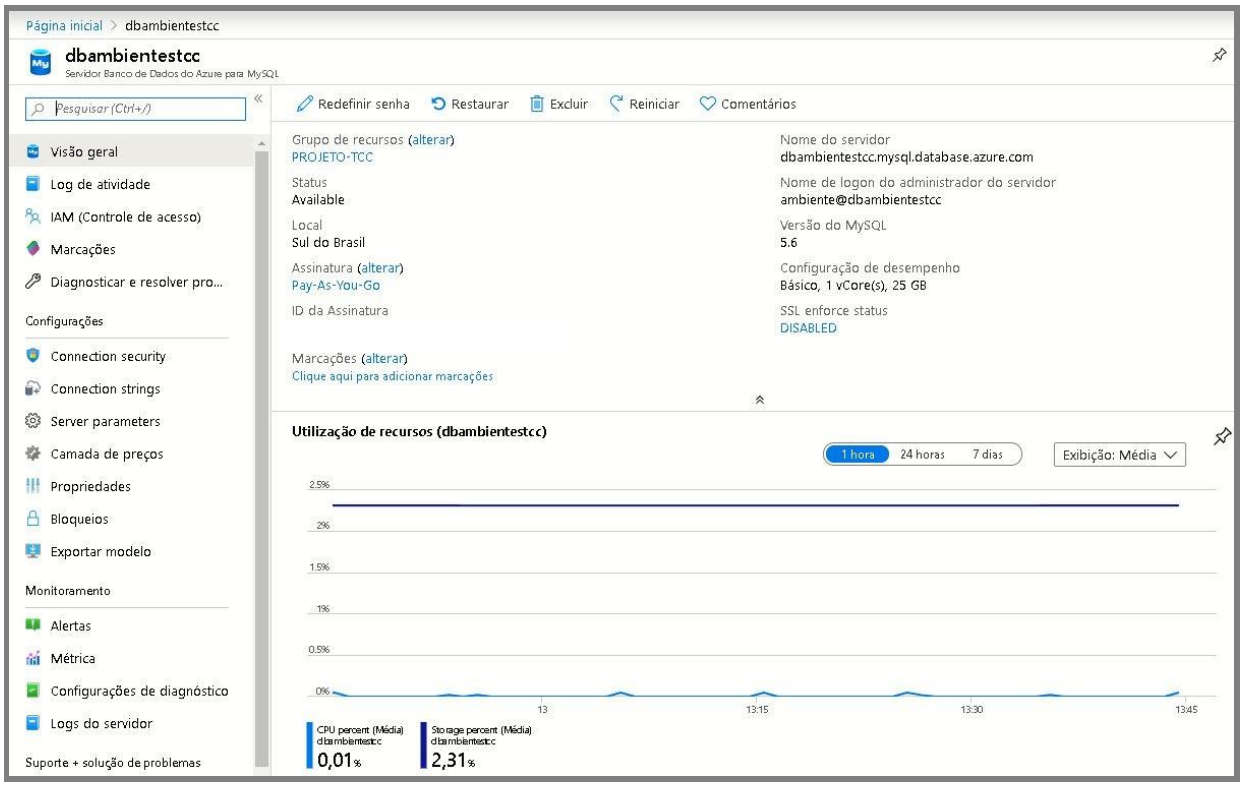
a) Status (Deslocando)
b) Local (Oeste da Europa)
c) Sistema operacional (Linux Debian 8.11)
d) Endereço IP Público (51.105.101.63)

A figura apresenta gráficos de monitoramento dos recursos (ex.: CPU e Rede) que podem ser mostrados em um período a escolha do utilizador (ex.: 1 hora ou 7 dias).

O ambiente decentralizado contém um servidor web primário que faz parte de um cluster e recebe as requisições solicitadas através do balanceador, que tem como funcionalidade simular o recebimento das requisições em um ambiente descentralizado. Além do servidor primário o Cluster contém um servidor secundário que contém as mesmas características de hardware e estrutura tecnológicas do servidor primário.

A figura 17 demostra a visão geral do banco de dados *MySQL* criado no ambiente descentralizado na plataforma *Azure.*

Figura 17 – Serviço virtual de banco de dados MySQL.



O serviço de banco de dados oferecido pela plataforma Azure foi criado com alta disponibilidade e de fácil configuração. O banco de dados neste ambiente descentralizado está funcionando de forma centralizada, servindo como base de acesso para o servidor primário e secundário.

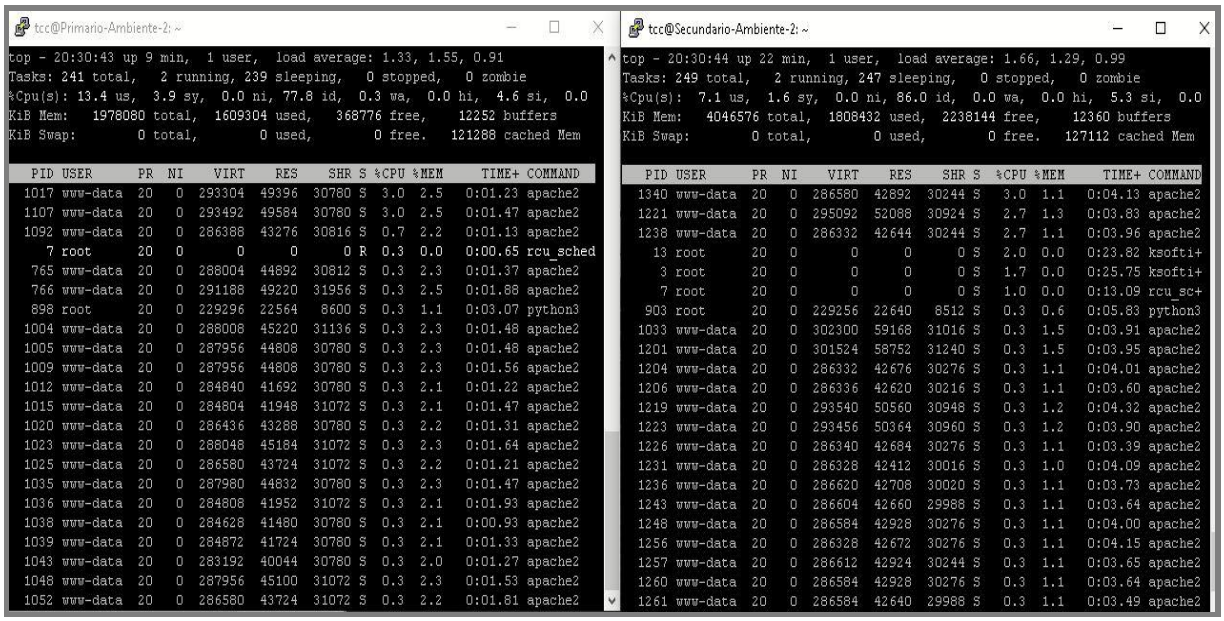
4.3. EXPERIMENTO

Neste trabalho foram realizados dois tipos de experimentos: um que testa a disponibilidade do ambiente descentralizado e outro que testa o desempenho dos dois ambientes. As seções a seguir descrevem como os testes foram feitos e o resultado obtido.

4.3.1. Disponibilidade

Para que o balanceamento seja realmente funcional é preciso que haja disponibilidade no ambiente. As Figuras 18 e 19 demostram os testes feitos no ambiente descentralizado, onde a dois servidores e poderá ser feito a simulação de possíveis falhas em um deles.

Figura 18 - Teste de disponibilidade.

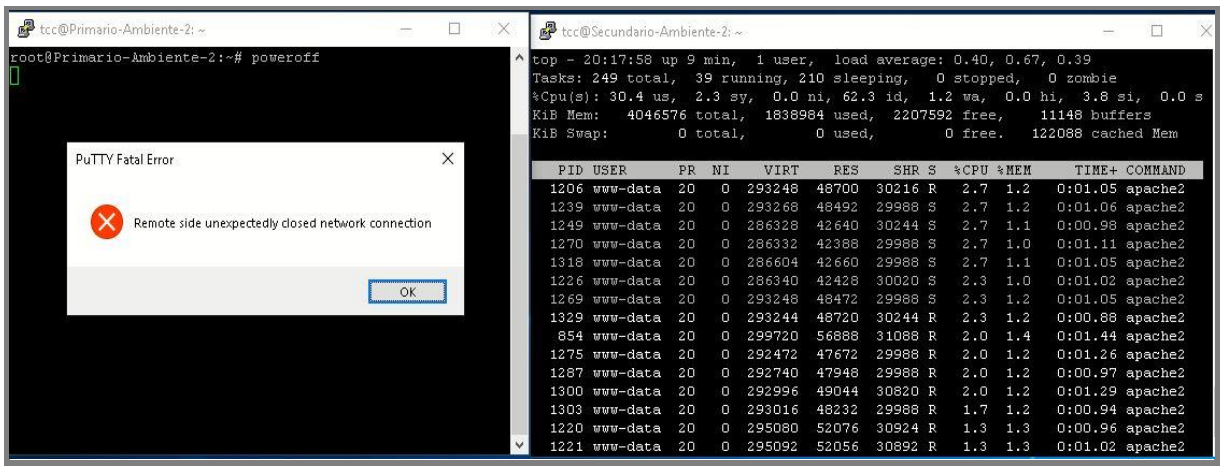


A Figura 18 demonstra requisições sendo solicitadas para os dois servidores web e a distribuição de carga entre eles. Com o comando *TOP* é possível visualizar todos os processos em execução no sistema, através deste comando foi possível verificar as seguintes características:

a) PID (Numero identificador de cada processo)
b) USER (Usuário que está efetuando a conexão)
c) %CPU (Porcentagem de consumo de CPU de cada conexão)
d) %MEM (Porcentagem de consumo de RAM de cada conexão)
e) TIME (Tempo de duração de cada conexão)

Na Figura 19 é possível verificar a indisponibilidade do servidor web primário e toda a carga sendo redirecionada para o servidor web secundário.

Figura 19 – Teste de disponibilidade 2.



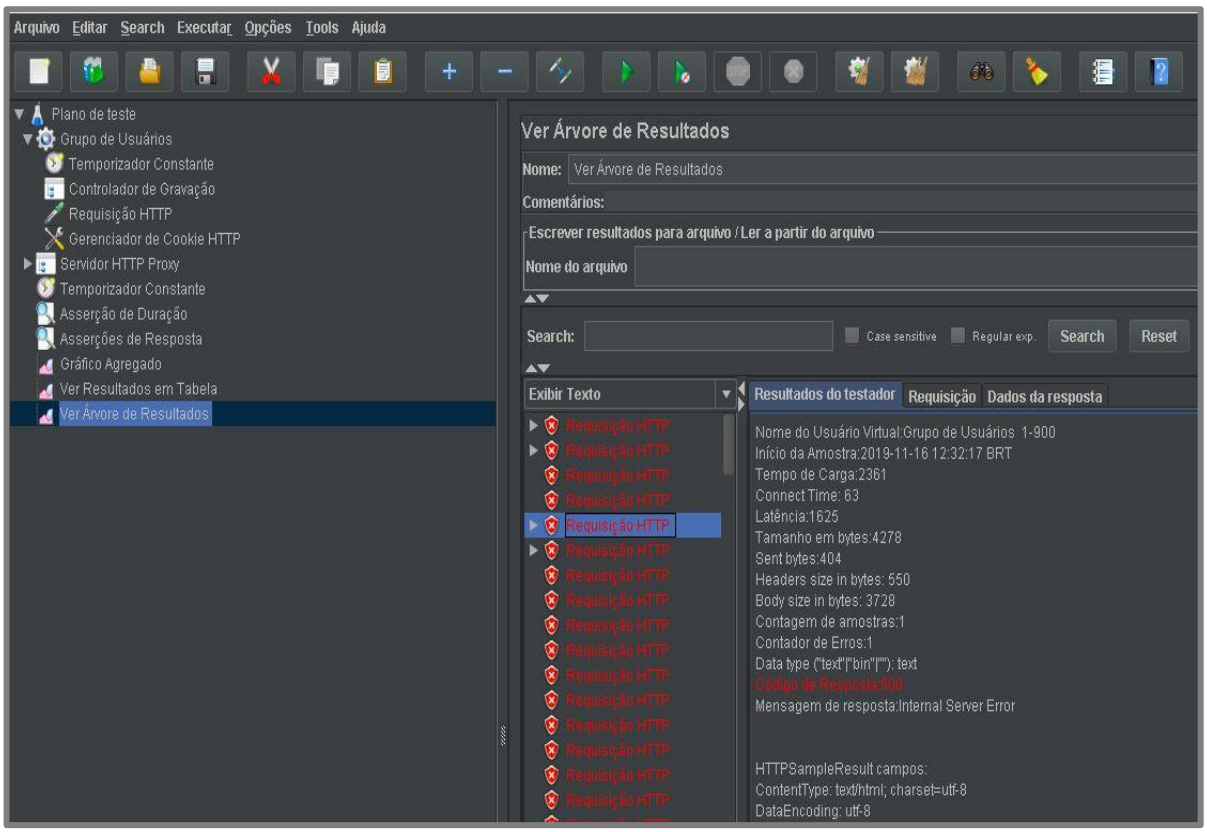
Foi feito o desligamento manual do servidor web primário com o comando poweroff demostrado na Figura. A mensagem de aviso mostrada diz que a conexão foi fechada, isso foi necessário para simulação de inatividade dos serviços web e verificar se as requisições solicitadas seriam redirecionadas para o servidor secundário. O teste demonstrou que em uma possível falha ou erro de um dos servidores o balanceador se encarrega de encaminhar todas as requisições para o servidor ativo.

Após o teste de disponibilidade é preciso identificar a performance dos servidores de acordo com o fluxo de requisições e quando há uma possível falha. Esse experimento serve para identificar possíveis necessidades de ajustes de infraestrutura do ambiente.

4.3.2. Perfomance

Para verificar a performance dos ambientes foram feitos testes de *stress* com a ferramenta *JMeter*, que simula conexões de acesso de acordo com a quantidade definida de usuários. Este teste serviu para identificar o limite de requisições que os ambientes suportam. Na Figura 20 é demostrado o teste feito no ambiente centralizado.

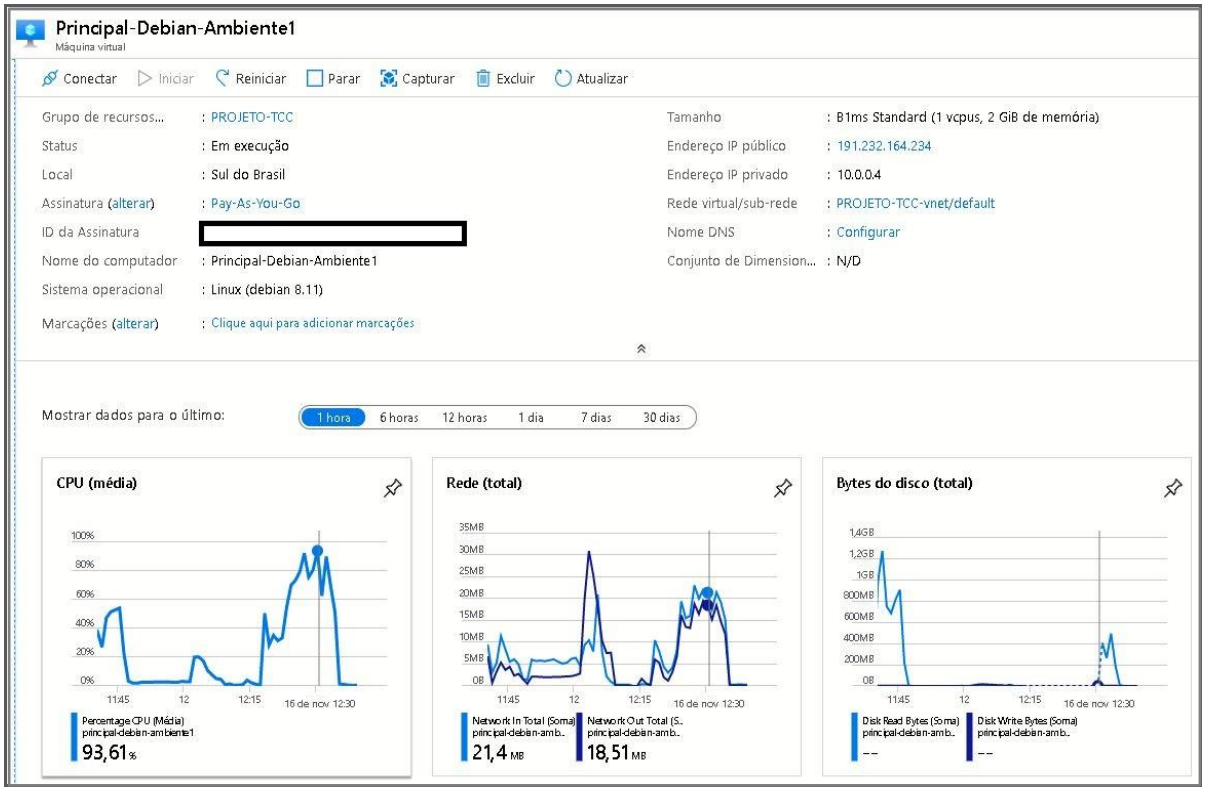
Figura 20 - Teste de Conexões JMeter Ambiente centralizado.



A Figura 20 demonstra um teste de requisição feito ao ambiente centralizado. Foi criado um grupo de teste chamado "ambiente 1", simulando conexões de mil usuários ao servidor web no intervalo de 1 segundo cada um. O resultado é mostrado em *árvore de resultados* (à direita da figura) onde as informações em **vermelho** mostram o momento em que as requisições começaram a ser negadas, com um retorno de código 500 do servidor, (mensagem de resposta *Internal server error*). Além da mensagem de negação do servidor são apresentadas informações como: latência, tempo de carga e tempo de conexão.

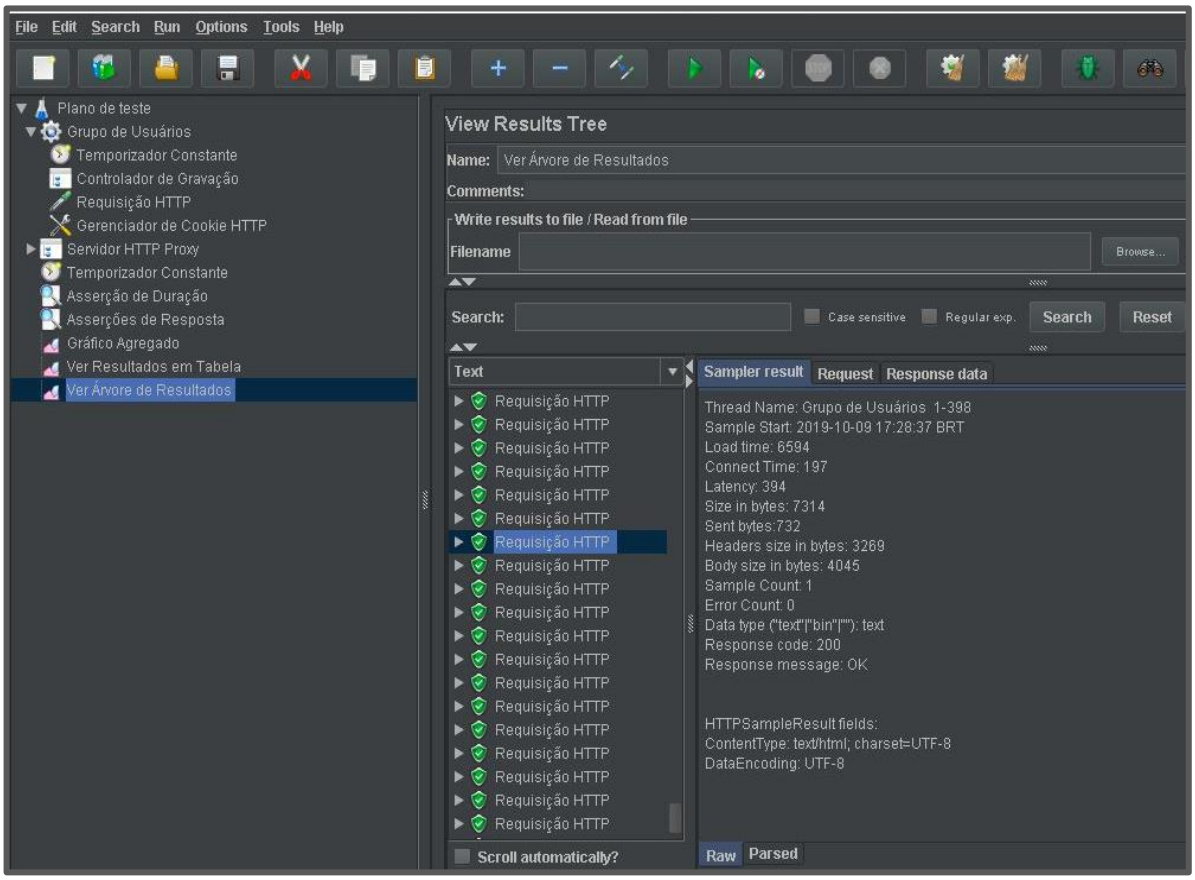
Durante o teste de *stress* foi monitorado o desempenho do hardware do servidor web, o resultado desse monitoramento é ilustrado na Figura 21.

Figura 21 – Monitor de desempenho servidor web Ambiente centralizado.



A Figura 21 mostra um painel de monitoramento de recursos (CPU, REDE e DISCO). A simulação das conexões feitas pelo JMeter ocasionou o consumo de 93,61% de CPU, o que fez com que o servidor web recusasse a maioria das requisições, ocasionando a mensagem de *internal server erro* mostrada na Figura 20. O mesmo teste foi feito para o ambiente descentralizado, como ilustra a Figura 22.

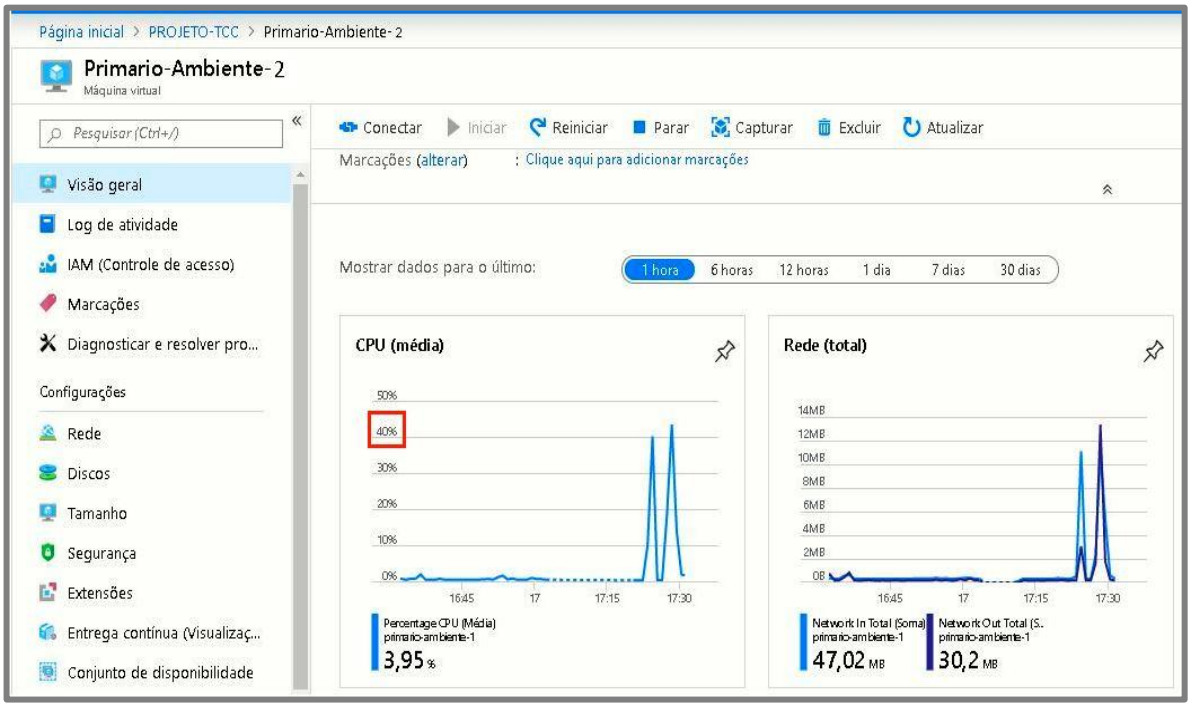
Figura 22 – Teste de conexão JMeter Ambiente descentralizado.



A Figura 22 demonstra um teste de requisição feito ao ambiente descentralizado. Da mesma forma como no teste do ambiente centralizado, o teste atual considera requisições de mil usuários. Na *árvore de resultados* é mostrado o resultado onde as informações em **verde** mostram que as requisições tiveram retorno do servidor com código 200 (mensagem *OK*).

Assim como o ambiente centralizado foi monitorado durante as simulações de requisições pelo JMeter, o ambiente descentralizado passou pela mesma análise como ilustra a Figura 23.

Figura 23 – Monitor de desempenho servidor web primário ambiente descentralizado.



A Figura 24 ilustra um painel de monitoramento de recursos (CPU, REDE e DISCO) e indica que a simulação das conexões feitas pelo JMeter ocasionou o consumo de 40% de CPU com o grupo de 1000 usuários.

4.4. ANÁLISE COMPARATIVA DOS RESULTADOS

Após a realização dos experimentos do ambiente centralizado e descentralizado, foi feito um comparativo das métricas obtidas, buscando mostrar qual o ambiente apresenta melhor desempenho mediante os testes apresentado.

A análise comparativa dos resultados obtidos foi apresentada na Tabela 1.

Tabela 1: Comparativo de consumo de recursos dos Ambientes centralizado e decentralizado.

Ambientes	Usuários	CPU (%)	Memória (MB)	Disco (MB/s)	Rede (KB/s)	Requisições atendidas
Centralizado	1000	93.61	1898	3.5	70	600
Decentralizado	1000	40	1609	2.39	196	1000

Conforme apresentado na Tabela 1, os resultados foram obtidos com a mesma quantidade de usuários e de acordo com o ambiente, com as seguintes características e resultados:

a. **Porcentagem de consumo de CPU:** pode ser observado que o consumo de CPU utilizado no ambiente centralizado é de 57.27 % maior que o ambiente descentralizado na simulação de 1.000 (mil) conexões. Essa diferença pode ser explicada devido a quantidade de servidores disponíveis para resposta das requisições, causando assim uma sobrecarga do ambiente centralizado e a rejeição das requisições.
b. **Consumo de memória RAM:** no consumo de memória RAM o ambiente centralizado mostrou um aumento de 15.23% em relação ao ambiente descentralizado
c. **Consumo de Disco:** é observado que o consumo do ambiente centralizado é 31.71% maior que o ambiente descentralizado no momento dos testes. Esta diferença se deu pela capacidade de recurso de leitura e escrita do ambiente centralizado ser menor que o ambiente descentralizado.
d. **Consumo de rede:** conforme apresentado na tabela 1 o ambiente centralizado mostrou diferença de consumo na rede de 64.29% menor referente ao ambiente decentralizado. Esta diferença se deu pelo fato da distribuição do ambiente descentralizado, permitindo mais (KB/s) durante a simulação e aceitação das requisições.
e. **Requisições atendidas:** o ambiente centralizado mostrou perca de 40% de resposta de requisições referente ao ambiente descentralizado.

5. CONSIDERAÇOES FINAIS

Como forma de resolver o problema de disponibilidade e consequente aumento de usuários que utilizam de serviços e aplicações web, os objetivos propostos pelo trabalho foram cumpridos através da criação de uma solução de balanceamento de requisições de serviços web em cluster.

Foi verificado que a utilização do balanceamento de requisições de serviços web em cluster se mostrou satisfatória por apresentar características como a disponibilidade e alto desempenho. O balanceamento escolhido neste trabalho foi voltado para o melhor uso dos recursos dos servidores web.

No portal acadêmico CEUP/ULBRA foi observado que a utilização do balanceamento de requisições de serviços web em *cluster* pode trazer melhor desempenho durante os acessos, mostrando nos experimentos ganhos do ambiente descentralizado de 57.27% de CPU, 64.29% a mais de tráfego de rede, 15.23% de ganho de memória RAM e 40% a mais de aceitação das requisições.

Foi observado, também, que além do ganho de performance do ambiente descentralizado houve o ganho da característica de disponibilidade, fazendo com que este ambiente se torne mais seguro para uso de aplicações web.

Com a utilização do ambiente descentralizado, foi observado o ganho do tempo de reparo e a continuidade dos serviços web quando há erros ou falhas de configurações ou fisicas no servidor. Esse ganho é ocasionado pelo fato de o ambiente obter mais de um servidor ativo, que em uma eventual necessidade de reparo pode-se manter ativos e funcionando os demais servidores.

Como trabalho futuro pode-se criar o balanceamento de serviços de banco de dados no modo *Master-Slave*, buscando a melhoria e alta disponibilidade de requisições em servidores web.

6. REFERÊNCIAS

ARREGOCES, MAURICIO; PORTOLANI, MAURIZIO: **Data Center Fundamentals**. Indianapolis : Cisco Press. 2004.

ABREU, Thiago W. M. Sba. **Controle & Automação Sociedade Brasileira de Automática** - 15/05/2006. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-17592012000100004&lng=pt&nrm=iso>. Acessado em: 10 de Abril de 2018.

GONÇALVES, Ariane. **O que é SSL/TLS e HTTPS?**. Disponivel em: <https://www.hostinger.com.br/tutoriais/o-que-e-ssl-tls-https/#gref>. Acessado em 23 de janeiro de 2019.

ALMIR, José. **Como funciona um servidor web**. Disponivel em: <https://blog.mettzer.com/referencia-de-sites-e-artigos-online/>. Acessado em 24 de novembro de 2018.

NOBREGA, Pablo**. PROPOSTA DE UM AMBIENTE DE ALTA DISPONIBILIDADE PARA SISTEMAS JAVA WEB USANDO COMPUTAÇÃO EM NUVEM.** Disponivel em: < http://www.uece.br/mpcomp/index.php/arquivos/doc_download/332-dissertacao(107)PROPOSTA DE UM AMBIENTE DE ALTA DISPONIBILIDADE PARA SISTEMAS JAVA WEB USANDO COMPUTACAO EM NUVEM.pdf>. Acessado em 15 de dezembro de 2018.

CADINELLI, Claudia. **CLUSTER DE SERVIDORES, ENTENDA O CONCEITO**. Disponível em: < https://indicca.com.br/cluster-de-servidores-entenda-o-conceito/>. Acessado em 05 de dezembro de 2019.

COULOURIS G, DOLLIMORE J, KINDBERG T. 3 EditionDistributed Systems Conceptsand Design. Addison Wesley, 2001.

EDUARDO, Carlos. **Implantação de um serviço de Balanceamento de Carga utilizando LVS/Piranha**. Disponível em: <http://repositorio.ufla.br/bitstream/1/5357/1/mono-CarlosEduardo_0.pdf>. Acessado em 30 de dezembro 2018.

FINDUP. **Descubra a importância da escalabilidade**. Disponível em: <https://findup.com.br/descubra-a-importancia-da-escalabilidade/>. Acessado em: 1 de janeiro de 2019.

IBM. **Algoritmos para tomar Decisões sobre balanceamento de carga.** Disponível em: <https://www.ibm.com/support/knowledgecenter/pt-br/SS9H2Y_7.6.0/com.ibm.dp.doc/lbg_algorithms.html>. Acessado em: 27 de novembro de 2018.

JACKSON, Bryan.**GTmetrix**, 2018.Disponivel em: <https://kinsta.com/pt/blog/teste-de-velocidade-gtmetrix/>. Acessado em: 24 de novembro de 2018.

LEANDRO, Sabrina da Silva**. Balanceamento de Carga em Web Services**. 2005. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_146/Relat%F3rioFinal_BalanceamentoCargaWS_SabrinaLeandro.pdf>.

OLIVEIRA, Paulo. **Servidor Web - O que é e como escolher um para seu site**. 19 de fevereiro 2016. Disponível em: <https://www.escolalinux.com.br/blog/servidor-web-o-que-e-e-como-escolher-um-para-seu-site>. Acessado em 24 de novembro de 2018.

RODRIGUES, André. **Como funciona um Servidor HTTP/Web**. 2007. Disponivel em: < https://www.portalgsti.com.br/2017/08/como-funciona-um-servidor-http-web.html>. Acessado em 23 de novembro de 2018.

SCHLOSSNAGLE, THEO: **Scalable Internet Architectures. Indianapolis**:Sams. 2007. Disponível em: <http://lethargy.org/~jesus/misc/Scalable%20Ti.pdf>. Acessado em: 10 de junh de 2018.

SMANIOTTO, Carlos Eduardo. Artigo da SQL Magazine 24 - **Replicação e alta disponibilidade no PostegreSQL**. Disponível em: <http://www.devmedia.com.br/artigoda-sql-magazine-24-replicacao-e-alta-disponibilidade-no-postgresql/6140> Acessado em : 15 de Abril de 2018.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **DataBase System Concepts**. 5. ed. 2006.

SILVA LEANDRO, Sabrina. **BALANCEAMENTO DE CARGA EM WEB SERVICES.** Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_146/Relat%F3rioFinal_BalanceamentoCargaWS_SabrinaLeandro.pdf> Acessado em: 25 de dezembro de 2018.

GTmetrix. **Sobre o desenvolvedor**. Disponível em:<https://gtmetrix.com/>. Acessado em: 24 de novembro de 2018.

TECHNET**. Como funciona o Balanceamento de Carga de Rede**. Disponível em: <https://technet.microsoft.com/pt-br/library/cc738894%28v=ws.10%29.aspx>.Acessado em : 14 de Abril de 2018.

TIVIT, One Cloud. **O QUE É DIGITAL OCEAN – SÉRIE PROVEDORES DE CLOUD.** Disponível em: <http://blog.tivit.com/o-que-e-digital-ocean-serie-provedores-de-cloud>. Acessado em: 24 de novembro de 2018.

JMETER, Apache. **ApacheJmeter**. Disponível em: <https://jmeter.apache.org/>. Acessado em: 24 de novembro de 2018.

TANENBAUM, A. S.; STEEN, M. V. Sistemas Distribuídos: **Princípios e Paradigmas**. 2 ed. São Paulo: Prentice-Hall, 2007. 416 p.

VINICIUS, Thiago. **Como funcionam as aplicações web**. 2012. Disponível em: <https://www.devmedia.com.br/como-funcionam-as-aplicacoes-web/25888>. Acessado em: 25 de novembro de 2018.

VIEIRA, Nando. **Entendendo um pouco mais sobre o protocolo HTTP**. 05 de maio 2007. Disponível em: https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>. Acessado em 03 de dezembro de 2018