UNISINOS

Programa de Pós-Graduação em

Computação Aplicada

Mestrado/Doutorado Acadêmico

Thiago Roberto Lima Lopes

DiCloud: Um Modelo Para Compartilhamento Transparente De Imagens Médicas Com Compressão e Elasticidade Proativa Na Nuvem

São Leopoldo, 2020

L864d Lopes, Thiago Roberto Lima.
DiCloud : um modelo para compartilhamento transparente de imagens médicas com compressão e elasticidade proativa na nuvem / Thiago Roberto Lima Lopes – 2020.
98 f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2020.
"Orientador: Prof. Dr. Rodrigo da Rosa Righi"

1. Compartilhamento. 2. Computação de alto desempenho. 3. DiCloud. 4. DICOM. I. Título.

CDU 004.932

Dados Internacionais de Catalogação na Publicação (CIP)
(Silvana Dornelles Studzinski – CRB 10/2524)

(Esta folha serve somente para guardar o lugar da verdadeira folha de aprovação, que é obtida após a defesa do trabalho. Este item é obrigatório, exceto no caso de TCCs.)

Dedico este trabalho aos meus pais Paulo Lima Lopes e Dalva da Silva Villa, que sempre acreditaram no meu potencial e contribuíram com essa conquista.

If I have seen farther than others,
it is because I stood on the shoulders of giants.
— SIR ISAAC NEWTON

AGRADECIMENTOS

Agradeço a todos que estiveram presentes em minha trajetória acadêmica: colegas, amigos, família, os mais chegados e a todos que contribuíram com sua força, conselhos, ajuda e colaborações. Em especial, aos meus pais, Paulo e Dalva que sempre me apoiaram nos estudos e nas horas difíceis. A minha namorada Mariana que sempre és compreensiva, amável e empenhada a me ajudar no que for preciso. Ao meus amigos Lucas Pfeiffer, Helen Appelt e Igor Fontana de Nardin pelos conselhos dados ao longo deste trajeto, pelas discussões e argumentações construtivas sobre este trabalho. Agradeço ao professor e orientador Dr. Rodrigo da Rosa Righi, que me incentivou a escrever artigos acadêmicos com conteúdo de qualidade, por seu profissionalismo, dedicação e paciência ao me auxiliar neste trabalho, tornando-o em realidade.

Agradeço o apoio do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico - http://www.cnpq.br) e CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - http://www.capes.gov.br) pela bolsa concedida.

"*Ninguém abre um livro sem que aprenda alguma coisa*".
(Anônimo)

RESUMO

A partir de 1970, com o surgimento da tomografia computadorizada, as imagens médicas se tornaram parte fundamental do fluxo de diagnóstico. A padronização das imagens através do formato DICOM (Digital Imaging and Communications in Medicine) representou um grande avanço para a interoperabilidade das máquinas de capturas das imagens médicas, bem como para os softwares que realizam a visualização. Todavia, ainda no presente existem desafios que precisam serem perpassados para aperfeiçoar os processos, a fim de agilizar o diagnóstico clínico. O servidor de armazenamento das imagens (PACS), em geral, se localiza internamente nos hospitais, não sendo possível compartilhar essas imagens com outros médicos que estejam fora do domínio do hospital, dificultando por exemplo, a colaboração de um especialista que esteja em um local geograficamente distante. Contudo, essa arquitetura pode ser distribuída usando *cloud computing*, que além de colaborar para o compartilhamento de imagens médicas, colabora com o gerenciamento mantendo a qualidade de serviço mesmo sobre o aumento de carga de trabalho. Utilizando elasticidade é possível alocar e desalocar recursos conforme algumas métricas, como CPU, visando melhorar o desempenho da aplicação. Existem diversos trabalhos que utilizam elasticidade para melhorar o desempenho da aplicação, mas a maioria dos trabalhos baseiam-se na elasticidade reativa, onde as ações são executadas quando alguma métrica atinge um determinado limite. Além disso, outros trabalhos que abordam compressão das imagens a fim de reduzir a demanda por espaço em disco, o fazem apenas no *pixel-data* e não se atém em questões relacionados ao tempo de transmissão das imagens e velocidade da conexão com a internet. O acesso limitado à pesquisadores proposto nos trabalhos relacionados desatende os profissionais da saúde que podem se beneficiar de um acesso remoto e mais rápido às imagens. Neste trabalho, se utiliza uma abordagem diferenciada, com a elasticidade proativa e um algoritmo baseado em séries temporais para prever valores futuros de uma métrica e tomar decisões de antemão. Foram modeladas quatro cargas de trabalho, utilizadas na avaliação do modelo: constante, onda, crescente e decrescente. É demonstrado resultados de predição de valores com a simulação de cargas de trabalhos reais. Para cada carga de trabalho foi avaliado o melhor modelo do ARIMA para predição. O algoritmo mais eficiente foi escolhido e aplicado nos demais testes, que incluem mensurar o nível de compressão alcançado para cada um dos métodos de compressão avaliados. O modelo teve sua contribuição validado junto a um software de servidor PACS real, nos aspectos de desempenho, seu grau de compressão e a efetividade em compartilhar imagens com outro hospital. Este trabalho contribui nos aspectos de tornar possível o compartilhamento de imagens médicas, mesmo em ambientes com internet com baixa largura de banda, características em países em desenvolvimento, através de compressão das imagens para melhor aproveitamento da conexão com a internet. O melhor algoritmo de compressão foi o Zip+PPMd, que obteve uma redução do tamanho do *dataset* de 72.32%. Também contribui provendo elasticidade proativa de aplicações, onde os resultados demonstram que é possível manter a qualidade de serviço (0% de erros) e reduzir os custos da nuvem (menor quantidade de VMs ligadas), através da correta alocação e liberação dos recursos.

Palavras-chave: Elasticidade. Escalabilidade. DICOM. Desempenho. Computação de Alto Desempenho. Compartilhamento. DiCloud.

ABSTRACT

Since 1970, with the advent of computed tomography, medical images have become a fundamental part of the diagnostic flow. The standardization of images through the DICOM format (Digital Imaging and Communications in Medicine) represented a significant advance for the interoperability of the medical image capture machines, as well as for the software that performs the visualization. However, in the present, some challenges need to be overcome to improve the processes in order to speed up the clinical diagnosis. The image storage server (PACS), in general, is located internally in hospitals, and it is not possible to share these images with other doctors outside the hospital's domain, making it difficult, for example, to collaborate with a specialist who is in a location geographically distant. However, we can distribute this architecture using cloud computing, which also collaborates for sharing the medical images and helps the management supporting the quality of service even on the increase in workload. When using elasticity, it is possible to allocate and deallocate resources according to some metrics, such as CPU, to improve the performance of the application. Several works use elasticity to improve the application's performance, but most of the works apply reactive elasticity, where one performs actions when a metric reaches a specific threshold. Also, works that address image compression to reduce the disk space demand do so only on pixel-data and do not focus on issues related to the image transmission time and the speed of the internet connection. The researchers-only access proposed in related works dismisses health professionals who can benefit from remote and faster access to medical images. In this work, a differentiated approach is employed, using a proactive elasticity and an algorithm based on time series to predict future values of a metric and make decisions beforehand. Four workloads were modeled and used to evaluate the model: constant, wave, ascending, and descending. We evaluate different ARIMA parameters for each workload, to choose which parameter set is the best ARIMA to forecasting in our model. Besides using the most efficient ARIMA algorithm, the tests also include measuring the compression level for each of the compression methods available on 7-Zip. We validated this model with real PACS server software, in terms of performance, compression rate, and the effectiveness in sharing images with another hospital. This work contributes to making the sharing of medical images possible even in environments with low internet bandwidth (commonly found in developing countries), through images compression for better use of the internet connection. The best compression algorithm was Zip + PPMd, which obtained a reduction of 72.32% in the dataset size. It also contributes by providing proactive elasticity of applications, where the results demonstrate that it is possible to maintain the quality of service (0% errors) and reduce the costs of the cloud (fewer connected VMs), through the correct allocation and release of resources.

Keywords: Elasticity. Scalability. DICOM. Performance. high-performance computing. Sharing. DiCloud.

LISTA DE FIGURAS

Figura 1 – Incremento anual de volume de armazenamento necessário nos hospitais dos Estados Unidos (EUA), juntamente com a média de espaço requerido a cada novo procedimento de captura de imagens médicas. 26

Figura 2 – Fluxograma das etapas de desenvolvimento da pesquisa. 30

Figura 3 – Estrutura de um arquivo JPEG. 38

Figura 4 – Elasticidade Vertical e Horizontal. 42

Figura 5 – Suavização após recurso disponível. 43

Figura 6 – Processo de seleção de artigos. 48

Figura 7 – Visualização simplificada do modelo DiCloud proposto neste trabalho. Através do DiCloud outros hospitais e médicos podem acessar imagem médicas remotamente. 57

Figura 8 – Modelo DiCloud proposto neste trabalho. Dentro do ambiente hospitalar se tem apenas a adição do componente DiCloud Client. O escopo do Serviço DiCloud é uma composição de diversas tecnologias existentes, modificadas ou autorais que compõe a solução apresentadas neste trabalho. Os **itens assinalados com um circulo vermelho fazem parte da proposta deste trabalho**, e os demais itens ou já existem no ecossistema atual ou foram simplesmente aplicados neste contexto sem modificações. 59

Figura 9 – Representação do cenário atual (A) e o cenário com a implantação do DiCloud (B). 61

Figura 10 – Fluxo de envio de uma imagem para o Serviço DiCloud 61

Figura 11 – Fluxo de decisão de elasticidade de recursos 66

Figura 12 – Cargas de trabalho 72

Figura 13 – Protótipo 74

Figura 14 – Protótipo 76

Figura 15 – Quantidade de conexões ativas em cada um dos serviços executando o componente Stream API com o algoritmo *Round-robin*. As máquinas são iniciadas e paradas pelo componente Elasticity Manager. 80

Figura 16 – Quantidade de conexões ativas em cada um dos serviços executando o componente Stream API com o algoritmo *Leastconn*. As máquinas são iniciadas e paradas pelo componente Elasticity Manager. 80

Figura 17 – Análise comparativa (da linha real, em azul) do tipo de carga em onda, com suas predições (demais series em outras cores). São apresentados apenas os algoritmos ARIMA com melhor desempenho. Quanto mais alinhadas as series ARIMA estiverem com a linha real (azul) melhor. 84

Figura 18 – Taxa de uso de CPU de todas as máquinas virtuais que estão servindo a aplicação Stream API no modo **proativo**. A linha em vermelho (eixo Y esquerdo) representa a carga (% taxa de uso de CPU) que o sistema está demandando. As barras verdes (eixo Y direito) representam a quantidade de núcleos de processamento disponíveis no sistema a cada instante de tempo. A tarja em amarelo representa o *threshold* superior, a partir de onde o sistema deve provisionar mais máquinas para ajudar no processamento. 85

Figura 19 – Taxa de uso de CPU de todas as máquinas virtuais que estão servindo a aplicação Stream API no modo **reativo**. A linha em vermelho (eixo Y esquerdo) representa a carga (% taxa de uso de CPU) que o sistema está demandando. As barras verdes (eixo Y direito) representam a quantidade de núcleos de processamento disponíveis no sistema a cada instante de tempo. A tarja em amarelo representa o *threshold* superior, a partir de onde o sistema deve provisionar mais máquinas para ajudar no processamento.. 85
Figura 20 – Tela de pesquisa e configuração de rede do *software* visualizador de imagens MicroDICOM. 87
Figura 21 – Tela de exibição de exames do *software* MicroDICOM. O Exame exibido foi acessado através do DiCloud Client. 88

LISTA DE TABELAS

Tabela 1 – Tamanho das imagens em megabytes (MB) por estudo. 34
Tabela 2 – Questões de Pesquisa. 47
Tabela 3 – Comparação dos trabalhos relacionados. A sigla NA é usada para indicar que o item não se aplica ao trabalho relacionado, ou literalmente não foi abordado pelo autor do trabalho. 54
Tabela 4 – Valores de erros dos algoritmos. Foram avaliados o erro médio (EM) e desvio médio absoluto (DMA) 83
Tabela 5 – Valores dos resultados dos testes realizados usando a ferramenta JMeter. Os resultados são a média de todas as requisições realizadas durante os 30 minutos de duração do teste programados. 86
Tabela 6 – Média de tempo necessário para submeter uma imagem DICOM da modalidade (simulador) aos serviços de armazenamento. 89
Tabela 7 – Valores dos ganhos (em %) de cada um dos métodos de compressão testados, sobre cada um dos arquivos do *dataset*. O nome dos arquivos tem o prefixo "1.3.6.1.4.1.5962.99.1.2280943358.716200484.1363785608958.", que foi substituído por ** na tabela a fim de simplificação. 90
Tabela 8 – Valores dos ganhos (em %) de cada um dos métodos de compressão testados, sobre sobre a soma de todos os arquivos comprimidos do *dataset*. 90

LISTA DE SIGLAS

API	Application Programming Interface
ACR	American College of Radiology
CR	Computed Radiography
CT	Computed Tomography
DICOM	Digital Imaging and Communications in Medicine
HIS	Hospital Information System
IaaS	Infrastructure as a Service
MRI	Magnetic Resonance Imaging
NEMA	National Electrical Manufactures' Association
PaaS	Platform as a Service
PACS	Picture Archiving and Communication System
RIS	Radiology Information System
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
VM	Virtual Machine

SUMÁRIO

1 INTRODUÇÃO **25**
1.1 Motivação 26
1.2 Questão de Pesquisa 27
1.3 Objetivos 28
1.4 Etapas de Desenvolvimento da Pesquisa 29
1.5 Organização do Texto 30

2 FUNDAMENTAÇÃO TEÓRICA **31**
2.1 DICOM - Digital Imaging and Communications in Medicine 31
2.1.1 Escopo Geral 32
2.1.2 Principais Características 33
2.1.3 Estrutura DICOM 33
2.1.4 Áreas de Aplicação 34
2.2 PACS - Picture Archiving and Communication Systems 35
2.2.1 Desafios e Oportunidades para PACS 35
2.3 Compressão de Imagem 36
2.3.1 JPEG 37
2.3.2 JPEG2000 Sem Perdas 37
2.4 Cloud computing 39
2.5 Elasticidade 40
2.5.1 Elasticidade Vertical 41
2.5.2 Elasticidade Horizontal 41
2.5.3 Alocação de Recursos 41
2.6 Escalabilidade 43
2.7 Balanceamento de Carga 44
2.8 Séries Temporais 45
2.8.1 Estacionaridade 45
2.8.2 ARIMA 45
2.9 Proliot - Proactive Elasticity Model 46

3 TRABALHOS RELACIONADOS **47**
3.1 Metodologia de Pesquisa e Escolha dos Trabalhos Relacionados 47
3.2 Análise e Oportunidades de Pesquisa 48
3.2.1 Performance Analysis of Medical Image Compression 48
3.2.2 Performance Analysis of Medical Image Compression Techniques 49
3.2.3 Towards a Hybrid Row-column Database for a Cloud-based Medical Data Management System 50
3.2.4 An Adaptive Streaming Technique for Interactive Medical Systems in Mobile Environment 51
3.2.5 Clustering of distinct PACS archives using a cooperative peer-to-peer network . . 51
3.2.6 Web-based Picture Archiving and Communication System for Medical Images . . 52
3.2.7 MIAPS: A web-based system for remotely accessing and presenting medical images 52
3.2.8 Integration of a Zero-footprint Cloud-based Picture Archiving and Communication System with Customizable Forms for Radiology Research and Education . . 53
3.2.9 A Community-driven Validation Service for Standard Medical Imaging Objects . 53
3.3 Comparação dos trabalhos relacionados 54

4 MODELO DICLOUD 57
4.1 Decisões de Projeto 57
4.2 Arquitetura 58
4.2.1 MODALITY 59
4.2.2 WORKSTATION 60
4.2.3 PC ou Dispositivo Móvel 60
4.2.4 PACS SERVER 60
4.2.5 DICLOUD CLIENT 61
4.2.6 SERVIÇO DICLOUD 62
4.2.7 STREAM API 62
4.2.8 COMPRESSOR / DECOMPRESSOR 62
4.2.9 TAGS DB 63
4.2.10 FILESYSTEM 64
4.2.11 LOAD BALANCER 64
4.2.12 ELASTICITY MANAGER 64
4.3 Atores e Casos de Uso 67

5 METODOLOGIA DE AVALIAÇÃO 69
5.1 Protótipo Elasticity Manager 69
5.1.1 Infraestrutura de Testes 70
5.1.2 Parâmetros do ARIMA 72
5.2 Protótipo Stream API 73
5.3 Protótipo DiCloud Client 75

6 RESULTADOS 79
6.1 Balanceamento de Carga 79
6.2 Elasticity Manager 80
6.3 Resultados do DiCloud 84
6.4 Operação fim-a-fim 87
6.5 Compressão 89

7 CONCLUSÃO 91
7.1 Contribuições 92
7.2 Limitações 92
7.3 Trabalhos futuros 93

REFERÊNCIAS 95

1 INTRODUÇÃO

Imagiologia médica é a visualização de partes do corpo, tecidos ou órgãos, para uso no diagnóstico clínico. O desenvolvimento tecnológico e a tomografia computadorizada, no início dos anos 1970, trazem imagens digitais para fazer parte do fluxo de diagnóstico (ALHAJERI, 2016). Atualmente, há uma grande demanda de recursos computacionais por hospitais, clínicas e consultórios, e podemos identificar a necessidade de prover escalabilidade de infraestrutura para esses ambientes, para tornar essas instituições mais eficientes e produtivas (JIN; CHEN, 2015; TENG; KONG; WANG, 2019). Um Sistema de Informações Hospitalares (SIH) contém um grande conjunto de informações digitais, que incluem dados financeiros, gerenciais, de pacientes (Registro Eletrônico do Paciente - EPR) e Sistema de Informações de Radiologia (RIS). Por causa do tipo de tecnologia empregada, a imagem médica é considerada como um sistema separado e é organizada em um sistema de transmissão e arquivamento de imagens médicas chamado Picture Archiving and Communication System (PACS) (SEERAM, 2019).

O PACS é um sistema que fornece armazenamento para imagens geradas por equipamentos médicos, de forma padronizada, permitindo que informações do paciente e suas respectivas imagens digitalizadas, armazenadas em mídia eletrônica, sejam compartilhadas e visualizadas em monitores de alta resolução distribuídos em locais fisicamente distintos (LIU; HUANG, 2020). Para a comunicação de dados computacionais entre diferentes sistemas, é necessário padronizar a linguagem utilizada. Inicialmente, o equipamento produzia diferentes formatos de imagem digital (GIF, JPEG, BMP, entre outros) e o uso crescente de computadores em aplicações clínicas por fabricantes de equipamentos levou à necessidade de um método padrão para arquivar e transferir imagens e informações entre dispositivos produzidos por diferentes fabricantes (TABATABAEI; LANGARIZADEH; TAVAKOL, 2017).

Na sequência da tomografia computadorizada, foram introduzidas outras modalidades de diagnóstico digital (RUI; HUANG; CHANG, 1999; ALHAJERI, 2016), e o uso de computadores em aplicações clínicas aumentou, onde os dispositivos produziam uma variedade de formatos de imagem digital. O Colégio Americano de Radiologia (ACR) e a National Electrical Manufacturers Association (NEMA) reconheceram a necessidade emergente de um método padrão para transferir e armazenar imagens e informações associadas entre dispositivos fabricados por diferentes fornecedores, e formaram um comitê, o qual começou a desenvolver um padrão em 1983, e publicado em 1985, chamado Digital Imaging and Communications (DICOM) (ZHANG; GE; WANG, 2008; YAN, 2018; HOLST, 2019).

O Padrão DICOM facilita a interoperabilidade de equipamentos de imagens médicas, especificando (LUBLINER, 2015): 1. Para comunicações de rede, um conjunto de protocolos a serem seguidos por dispositivos que reivindicam conformidade com o Padrão. 2. A sintaxe e semântica de comandos e informações associadas que podem ser trocadas usando esses protocolos. 3. Para comunicação, um conjunto de serviços de armazenamento de mídia a ser seguido por dispositivos que reivindicam conformidade com o padrão, bem como um formato

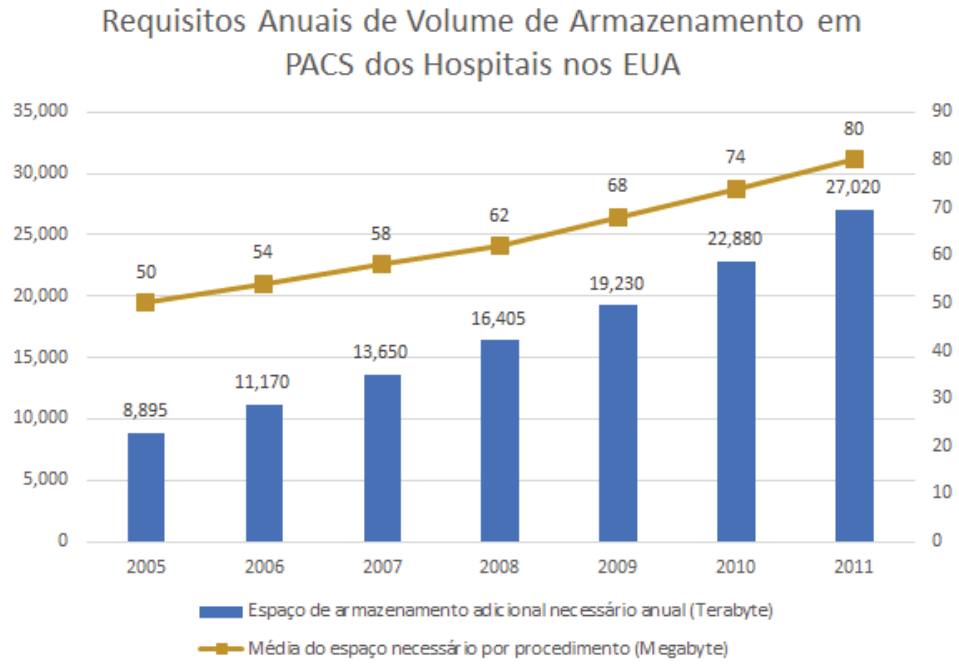
de arquivo e uma estrutura de diretório médica para facilitar o acesso às imagens e informações relacionadas armazenadas na mídia de troca.

1.1 Motivação

Imagens médicas desempenham um papel crucial no diagnóstico mais preciso e com menores taxas de insucesso terapêutico. Assim, o acesso a imagens médicas através de qualquer dispositivo, lugar ou momento se torna uma necessidade corrente dos especialistas (LI et al., 2020). Atualmente, estas imagens, no entanto, consomem uma grande quantidade de recursos para serem armazenadas ou transmitidas. Verifica-se a necessidade de que tais imagens sejam analisadas por diferentes especialistas com o propósito de facilitar o diagnóstico (ANUJA; JEYAMALA, 2015). Porém, há poucas formas de compartilhamento desses arquivos atualmente descritas na norma DICOM ISO 12052 (NEMA, 2014, 2016).

O compartilhamento de imagens faz parte de um quadro maior de agregação de valor e a capacidade de compartilha-lhas em uma organização ou em outras organizações é fundamental para criar assistência médica segura, rápida e econômica. As imagens médicas dos pacientes são uma informação crucial que muitas vezes falta quando os pacientes são transferidos no meio do evento ou tratados em vários locais diferentes. Acordo com Bruthans (BRUTHANS, 2020), além de pesquisas feitas com membros de hospitais e fabricantes de PACS, o compartilhamento das imagens DICOM é o próximo grande desafio a ser explorado.

Figura 1 – Incremento anual de volume de armazenamento necessário nos hospitais dos Estados Unidos (EUA), juntamente com a média de espaço requerido a cada novo procedimento de captura de imagens médicas.



Fonte: Adaptado pelo autor baseado em (SULLIVAN, 2018).

Também que a falta da capacidade adequada de compartilhamento gera problemas em vários aspectos, como o retardo nas análises dos exames, multiplicidade de exames desnecessários por falta de comunicação entre hospitais e clínicas (impactando no funcionamento ágil) e incompatibilidade de mídias como CDs entre máquinas PACS (ERICKSON, 2011). Além disso, também é dito que com o uso CDs, uma mídia frágil e que não fornece mecanismos de autorização para acesso aos dados nele contido, surge exposição de dados sensíveis do paciente (REACTIONDATA, 2015; GUO; ZHUANG, 2009; MCEVOY; SVALASTOGA, 2009).

Outro desafio crítico com o gerenciamento de imagens médicas hoje é o volume dos dados. O crescimento da quantidade de imagens e nos tamanhos dos arquivos colocou uma enorme pressão sobre os requisitos de armazenamento (Figura 1). Embora o crescimento tenha abrandado recentemente nos procedimentos anuais no PACS, a quantidade de armazenamento adicional necessária continua a crescer a uma taxa expressiva (SULLIVAN, 2018).

A necessidade de armazenamento não se dá somente pelo fato da quantidade de exames terem aumentado, mas também pela qualidade das imagens capturadas, que tem se aperfeiçoado, gerando arquivos maiores. O tamanho de um arquivo é um aspecto relevante quando se aborda sua transferência sobre uma rede de internet limitada. Este ponto então, está diretamente ligado ao anseio de prover um acesso rápida e econômico, que tenha o menor intervalo possível entre a captura da imagem e sua disponibilidade ao especialista, esteja ele no local ou geograficamente distante (LI et al., 2020).

1.2 Questão de Pesquisa

A questão de pesquisa que o modelo proposto busca responder é a seguinte:

Como seria um modelo com elasticidade proativa em nuvem para compartilhamento de imagens DICOM entre paciente, hospitais e clínicas com o mínimo de modificações no ambiente hospitalar, com internet limitada?

A ideia dessa proposta é a criação de um sistema PACS em modelo de *software* como serviço (SaaS), onde pode obter-se benefícios da virtualização de nuvem pública, permitindo a aplicação de modelos para gerenciar a elasticidade das aplicações que recebem, enviam e comprimem imagens do tipo DICOM.

Esta proposta também abrange a questão de compartilhamento das imagens entre hospitais, onde o modelo atuará como um *hub* realizando a interligação. Ao não se restringir a um escopo reduzido de um único hospital, o modelo demanda ser escalável para atender a múltiplas requisições de armazenamento e transferência em tempo hábil. Nesta proposta, o usuário não precisa especificar as métricas, nem os limiares que serão usados na tomada de decisão, sendo responsabilidade do modelo definir isso para cada serviço, bem como o algoritmo de compressão utilizado.

1.3 Objetivos

Conhecendo as motivações, o contexto e a questão de pesquisa no âmbito deste trabalho, é possível estabelecer seu objetivo geral:

> *Desenvolver um modelo de **elasticidade proativa** e **transparente** ao usuário para o **gerenciamento** de dados DICOM, utilizando **compressão** antes da transmissão de arquivos e viabilizando o **compartilhamento** deles entre usuários e hospitais.*

Contudo, visando detalhar este objetivo, assim como seus requisitos e resultados, é necessário dividi-lo em termos dos seus componentes, ou seja, os objetivos específicos desta pesquisa, sendo eles:

(i) Propor o DiCloud que contemple elasticidade e boa *performance* para preencher as lacunas encontradas na análise realizada;

(ii) Desenvolver um protótipo que implemente os elementos definidos pelo modelo proposto;

(iii) Avaliar os resultados obtidos a partir dos testes, verificando o desempenho e capacidade de escalabilidade através de elasticidade, bem como a redução no tempo de transferência através da compressão do arquivo DICOM.

Sendo assim, o objetivo deste trabalho é criar o DiCloud, um modelo para compartilhamento de imagens médicas com elasticidade proativa na nuvem, que atinja o objetivo de ser viável em ambientes com uma largura da banda de internet reduzida. Esse modelo engloba alguns conceitos chaves que para elucidar seus entendimentos dentro do escopo deste trabalho, os detalhamos nos seguintes item:

Compartilhamento de imagens médicas e seu gerenciamento - Em âmbito hospitalar, há pelo menos três atores envolvidos, sendo eles, o próprio paciente, o especialista e o hospital. Em geral, o hospital detém os métodos de capturas de imagens médicas, bem como, controla a forma de acesso a estas imagens após o exame. Não há uma forma digital e universal vigente para a disponibilização dessas imagens médicas de fácil acesso tanto para especialistas quanto para os pacientes.

Quando o paciente é transferido de hospital, em muitos casos é necessário refazer os exames radiológicos afim que o outro hospital possa obter as imagens para serem analisadas pelos seus especialistas. No mesmo sentido, a consulta a especialistas que estão em local geograficamente distante faz parte da telemedicina, que é limitada pela falta de um meio homologado para obter as imagens médicas do paciente. Um gerenciamento de imagens médicas, controlando o acesso, fazendo pesquisas, o envio e o recebimento, é algo que agrega valor e colaboraria para um atendimento clínico com maior agilidade.

Elasticidade proativa - O método de elasticidade proativa é de grande valia em ambientes onde a qualidade de serviço é importante, uma vez que através de análises ela antecipa o

conhecimento de que o sistema entrará em sobrecarga, podendo de maneira antecipada iniciar medidas, como aumentar a quantidade de instâncias rodando o sistema. Outra vantagem da abordagem proativa é que ela visa atuar de forma transparente ao administrador do sistema, pois tendo o histórico do funcionamento do sistema é possível atuar de forma autônoma e com maior precisão.

Transparente ao usuário - Quando se desenvolve uma aplicação que visa se integrar em um ambiente já consolidado, é importante minimizar as alterações no ambiente de modo a não gerar instabilidades e nem alterações de configuração, que em ambientes de grande escala ou com políticas corporativas restritivas, podem impedir a implantação do sistema. Com um baixo impacto no ambiente também se evita alteração de configurações que demandariam novos treinamentos aos usuários e administradores.

Compressão - A conexão coma internet é um desafio em muitas partes do mundo, principalmente em países subdesenvolvidos ou em desenvolvimento. A área hospitalar também se enquadra nas limitações impostas por uma internet de baixa qualidade, o que acaba por limitar a execução de procedimentos, diagnósticos ou laudos remotamente. Enquanto a evolução da malha de internet não alcança uma qualidade satisfatória, cabe a área de engenharia de software elaborar métodos e sistemas resilientes e inteligentes, que saibam aproveitar da melhor forma o link disponível.

Várias perspectivas são aplicáveis, como uma priorização do uso do link tal e qual exposto por Oliveira (OLIVEIRA et al., 2018), onde *workstation* DICOM e PACS fazem parte do grupo prioritário, ou como a compressão, que é uma das técnicas que colaboram para que os envios de informações sejam mais enxutos, reduzindo a utilização da banda disponível no link.

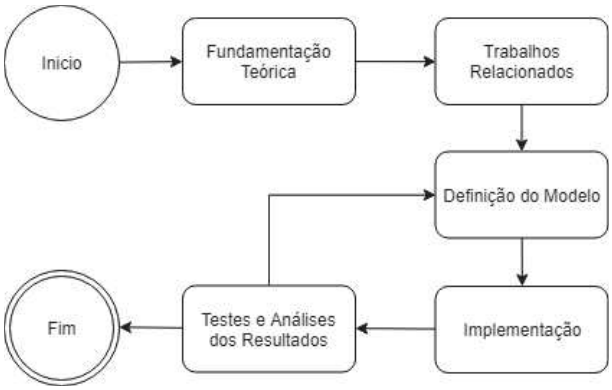
A principal área alvo deste trabalho é a área da computação, ou seja, apesar de serem abordados diversos aspectos da imagiologia, deve-se observar que o foco está na inadequação das ferramentas existentes e não nos aspectos da área de sua de aplicação, tais como cultura ou fatores políticos.

1.4 Etapas de Desenvolvimento da Pesquisa

O desenvolvimento da pesquisa ocorreu de acordo com a ordem apresentada na Figura 2. Foram 5 etapas, sendo elas: (1) Fundamentação teórica; (2) Trabalhos relacionados; (3) Descrição do modelo; (4) Implementação; (5) Testes e análise dos resultados.

Inicialmente, foi realizado um estudo das teorias envolvidas no tema de pesquisa para formar o referencial teórico. Em seguida, iniciou-se a etapa de levantamento dos trabalhos relacionados ao tema de pesquisa. Essa etapa visa buscar trabalhos com objetivos semelhantes a essa pesquisa, identificando assim possíveis lacunas. A terceira etapa consistiu em propor e desenvolver um modelo que preencha as lacunas encontradas nos trabalhos relacionados e que responda a questão de pesquisa, bem como atenda os objetivos do trabalho. Até o presente momento apenas as etapa 1, 2 e 3 foram executadas. As etapas 4 e 5 estão em andamento e

Figura 2 – Fluxograma das etapas de desenvolvimento da pesquisa.



Fonte: Elaborado pelo autor.

consistem em implementar o modelo proposto, testar a implementação e analisar os resultados obtidos. A etapa de teste e análise poderá gerar modificações no modelo proposto de acordo com o necessário.

1.5 Organização do Texto

A proposta está organizada em cinco seções. Inicialmente, o capítulo 2 apresenta conceitos fundamentais para a compreensão do restante do trabalho. Em seguida, o capítulo 3 apresenta os trabalhos relacionados aos temas dessa pesquisa, demonstrando um comparativo com essa proposta . Esse capítulo visa apresentar o que já existe no estado da arte, bem como identificar onde existem lacunas a serem preenchidas. O capítulo 4 apresenta o modelo proposto nesse trabalho que visa preencher as lacunas identificadas no capítulo anterior, bem como atingir os objetivos desse trabalho. O capítulo 5 apresenta a metodologia de avaliação e os cenários de uso. O capítulo 6 apresenta os resultados dos testes descritos no capítulo anterior, com gráficos e tabelas que demonstram os ganhos obtidos com o uso deste modelo. Por fim, o Capítulo 7 apresenta as conclusões que puderam ser obtidas e quais são as conclusões esperadas no fim desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão abordados alguns conceitos básicos para o entendimento da proposta de dissertação. Esses conceitos servirão como base para o restante do trabalho. Inicialmente, apresenta-se conceitos relacionados a captura de imagens médicas e por conseguinte, temas relacionados a computação como compressão de imagens, computação em nuvem e suas categorias, escalabilidade e tipos de elasticidade, bem como modelos de predição de padrões de comportamento no tempo usando series temporais. Com isso espera-se prover o embasamento necessário para o entendimento dos conceitos propostos neste trabalho.

2.1 DICOM - Digital Imaging and Communications in Medicine

Na área pertencente ao campo de tecnologias da informática voltadas para a medicina, há normatizações internacionais para desenvolvimento de soluções e fabricação de aparelhos. Equipamentos para captura de imagens médicas e seus sistemas acessórios são amparados por uma dessas normatizações conhecida como DICOM (NEMA, 2016). Para que dispositivos desenvolvidos para obtenção de imagens médicas possam funcionar com outros dispositivos, de diferentes fabricantes, foi estabelecido esse padrão internacional. Todo fabricante que reivindicar conformidade com a norma deve adotar as recomendações definidas neste compêndio (PIANYKH, 2009; MAANI; CAMORLINGA; ARNASON, 2012)

Esta normatização dá ênfase às imagens médicas de diagnóstico, estabelecendo um arcabouço para disciplinas médicas que atuam com a obtenção de imagens. Esse recurso de apoio ao diagnóstico é utilizado na radiologia, cardiologia, odontologia, oftalmologia e disciplinas afins, bem como nas terapias baseadas em imagem como radiologia intervencionista, radioterapia e cirurgia. O padrão DICOM (NEMA, 2016) é adotado como referência também para a medicina veterinária e seus ambientes médicos.

Este padrão pode ser aplicado a um largo espectro de procedimentos de diagnose, bem como para os dados que acompanham esses estudos e para os exames relacionados. Este padrão tem como principal finalidade, facilitar o relacionamento e a interoperabilidade entre sistemas que reivindicam conformidade à norma em questão. Surgiu da necessidade de estabelecer comunicação entre diferentes fabricantes de dispositivos de captura de imagem atuando no mercado (PIANYKH, 2009).

DICOM não é por si só uma garantia de interoperabilidade, cabendo a cada implantação adotar as medidas necessárias para o seu correto funcionamento. DICOM serve como referência internacional para serviços relacionados a imagens médicas e seus metadados, é parte do esforço conjunto entre a ACR (American College of Radiology) e da NEMA (National Electrical Manufactures Association) o padrão faz parte da ISO 12052:2006 (MILDENBERGER; EICHELBERG; MARTIN, 2002).

2.1.1 Escopo Geral

O desenvolvimento do padrão DICOM ocorreu através da inciativa e cooperação entre a ACR e a NEMA, citadas anteriormente, que fundaram um comitê para estudar formas de comunicar dados entre fabricantes distintos. No ano de 1985 essas duas entidades publicaram o que pode ser chamado hoje de versão 1.0 da norma. Nos anos seguintes foram feitas revisões, e novas versões foram lançadas, até o ano de 1993, onde foi apresentada a versão 3.0 chamada de "Digital Communications in Medicine."

A principal característica dessa versão foi a inclusão de um protocolo de rede baseado no modelo de referência OSI (Open System Interconnection) e o uso de TCP/IP como forma de garantir interoperabilidade entre diferentes desenvolvedores (MILDENBERGER; EICHELBERG; MARTIN, 2002). O padrão DICOM facilita a interoperabilidade dos equipamentos de imagens para apoio do diagnóstico médico, e manutenção dos dados relacionados a cada estudo através de especificações distintas definidas para diferentes áreas da computação (NEMA, 2016):

- Para todas as comunicações de rede norteadas pelo padrão, são definidos um conjunto de protocolos e métodos a serem seguidos por dispositivos para garantir conformidade com a norma;

- A sintaxe e a semântica dos comandos e das informações associadas que podem ser trocadas entre entidades distintas dentro de uma rede DICOM, usando estes protocolos;

- Para os meios de comunicação, a norma define um conjunto de serviços de armazenamento de mídia a ser seguido, bem como um formato de arquivo a ser adotado, e uma estrutura de diretórios para facilitar o acesso a imagens e informações armazenadas em outras mídias;

- São definidas ainda as informações que devem ser fornecidas junto com cada implementação para que a conformidade com a norma seja garantida.

De acordo com alguns autores, o padrão DICOM tem problemas relacionados com a entrada de dados. Uma das desvantagens encontradas na pesquisa diz respeito a possibilidade de entrada de múltiplos campos opcionais (metadados) após a obtenção das imagens. Esta desvantagem é mais evidente em inconsistências como a possibilidade de deixar campos contendo informações importantes em branco em determinados estudos, ou inserir dados em excesso sobre as imagens impossibilitando a interpretação das mesmas. Alguns atributos da imagem podem ainda estar incompletos ou ilegíveis, enquanto outros podem ser preenchidos com dados incorretos (MUSTRA; DELAC; GRGIC, 2008);

2.1.2 Principais Características

De uma forma geral, o padrão em estudo é separado atualmente em 20 partes distintas, as quais incluem desde as características gerais na introdução, até especificações de suporte para troca de mensagens, definições de estruturas de dados, perfis de segurança a serem adotados, etc. Entre outras explicações, a norma define os protocolos a serem utilizados, os serviços Web oferecidos, especifica as classes disponíveis em diferentes linguagens de programação, além de dicionários de dados, mídias para armazenamento e formato de arquivos padrão.

DICOM é fundamentado no modelo de referência OSI, o qual também possui diferentes camadas, cada camada com funções distintas estabelecidas a priori (PIANYKH, 2009). As camadas do modelo OSI são frequentemente separadas em: física, dados, rede, transporte e aplicação. DICOM é um protocolo que atua na camada de aplicação; O padrão garante a comunicação pela Internet, uma vez que é compatível com o conjunto de protocolos TCP/IP.

É importante identificar os termos adotados na norma para referir-se a aplicações utilizando o protocolo DICOM chamadas de Entidades de Aplicação (AE) na sigla original (MAANI; CAMORLINGA; ARNASON, 2012). DICOM possui uma coleção de serviços que consistem em ferramentas voltadas a facilitar a interação com os objetos dessa arquitetura, na maioria das vezes envolvendo transmissão de dados sobre a rede ou armazenamento dos mesmos em servidores específicos para esse fim (PACS, RIS, etc.) descritos na seção 2.4.

O serviço de armazenamento é descrito na parte 10 do padrão ISO 12052 (NEMA, 2016), e é um dos serviços mais importantes na norma, sendo utilizado para enviar imagens e outros objetos persistentes (relatórios e metadados) diretamente a um servidor de armazenamento, ou estação de trabalho para conferência de qualidade e futura armazenagem.

O padrão DICOM (NEMA, 2014, 2016) especifica ainda um modelo geral para o armazenamento de imagens e informações médicas em mídia removível. O objetivo desta parte é de proporcionar um quadro que permita o intercâmbio de vários tipos de imagens médicas e informações relacionadas em uma ampla variedade de mídias de armazenamento físico. Ao ter seus estudos gravados nessas mídias, o próprio paciente pode encaminhar seus exames para outros especialistas, ou mesmo o médico, de posse desses dados, pode valer-se da contribuição de uma equipe multidisciplinar, composta por profissionais alocados em diferentes hospitais da rede.

2.1.3 Estrutura DICOM

A orientação a objetivos proporciona não apenas uma maneira de descrever as informações existentes no modelo, mas o que fazer com estas informações ou como obtê-las sobre a coleção de objetivos existentes. O Padrão DICOM faz uso deste conceito para definir serviços, como "armazenar imagem"ou "obter informações do paciente". Estes serviços são implementados no DICOM usando construções chamadas de operações e notificação. É definido um conjunto de operações e notificações genéricas que são chamadas de elementos de serviço de mensagem no

DICOM (DIMSE - DICOM message service elements). A combinação de um IO e um serviço de DIMSE é chamado de par serviço-objeto (SOP - service object pair). Um IO pode ser usado com um conjunto de serviços e o resultado desta combinação é chamada de classe SOP (SOP class)

Tabela 1 – Tamanho das imagens em megabytes (MB) por estudo.

Modalidade	Tamanho minimo	Tamanho máximo	Tamanho médio	Desvio padrão
CR	7.0	34.0	13.6	8.7
MRI	21.0	242.0	81.9	52.6
CT	16.0	1150.0	294.0	260.0

Fonte: Adaptado de (MWOGI et al., 2018)

Assim como a estrutura dos arquivos DICOM variam, o tamanho do arquivo também, principalmente a parte conhecida como *pixel-data*, que é a *tag* dentro do arquivo onde se localiza a parte binária da imagem. A Tabela 2.1.3 apresenta os resultados de um estudo feito por (MWOGI et al., 2018), que mostra dados sobre os tamanhos médios encontrados em hospitais no Kenya.

2.1.4 Áreas de Aplicação

Diversas áreas da medicina adotam DICOM como referência para o estudo de imagens e este padrão é aplicável a muitos campos da saúde no qual a obtenção de imagens possa ter algum uso direto ou indireto (MAANI; CAMORLINGA; ARNASON, 2012). Isso inclui áreas relacionadas a radiologia, cardiologia, oncologia, radioterapia, neurologia, ortopedia, obstetrícia, oftalmologia, odontologia, dermatologia, sendo utilizado ainda como padrão para ensaios clínicos, alcançando inclusive áreas da medicina Veterinária.

Segundo a Parte 8 do padrão DICOM (NEMA, 2016), sua aplicação principal é capturar, armazenar e distribuir imagens médicas. A norma se preocupa em especificar serviços relacionados à imagem, como impressão em filmes físicos ou mídia digital. Relatar ainda o status de procedimentos e operações relacionadas a aquisição de dados, como a conclusão na obtenção de uma imagem, ou a confirmação de um arquivamento bem sucedido em um servidor de armazenamento. Existem também serviços relacionados ao conjuntos de dados, que removem informações de identificação de paciente dos dados selecionados, a fim de garantir confidencialidade quando necessário. Há ainda serviços de organização de layouts de imagens para avaliação, que economizam manipulações e anotações, calibrando e codificando a imagem que é exibida.

DICOM também é implementado por meio de dispositivos associados com imagens ou fluxo de trabalho de imagem, incluindo servidores de armazenamento e comunicação tipo PACS (Picture Archiving and Communication Systems) que serão melhor abordados na Seção 2.4, visualizadores de imagens e estações de consulta para interpretação por especialistas, CAD (Computer

Aided Detection/Diagnosis Systems), sistemas de visualização 3D, aplicações em análises clínicas, impressoras, scanners, gravadores de media que exportam arquivos para CD, DVD, RIS (Radiology Information Systems), VNA (Vendor Neutral Archives), EMR (Electronic Medical Record Systems), e Radiology Reporting Systems (NEMA, 2016)

2.2 PACS - Picture Archiving and Communication Systems

Um sistema PACS integra-se a norma DICOM agindo como servidor de armazenamento de estudos médicos, facilitando a comunicação e visualização de imagens radiológicas e metadados, dentro de redes de medicina (MOGHADAM et al., 2015). Como forma de simplificar diagnóstico e tratamentos baseados em imagem, foram desenvolvidas ferramentas para visualizar, anexar interpretações e arquivar esses dados utilizando PACS.

Ao passo que, dotado desta riqueza de recursos, a decisão por parte do especialista, sobre o tratamento adequado para diferentes tipos de enfermidades, torna-se mais eficaz e precisa. Percebe-se que decisões quanto ao procedimento médico acertado, principalmente para aquelas doenças que produzem alterações corporais visualizáveis através de imagens, serão mais adequadas do que as conclusões simplesmente amparadas nos sinais e sintomas descritos pelo próprio paciente durante a consulta.

Tendo o foco de suas aplicações sobre a diagnose de doenças dos ossos e tumores diversos, PACS é elemento estruturante de suma importância dentro da medicina. Os benefícios em potencial da utilização destes sistemas em clínicas e hospitais, são descritos em diversas revisões da literatura científica disponível. Do ponto de vista econômico, PACS também funciona como um amortizador de custos no armazenamento e manutenção de imagens.

Apesar de ter seu foco em ampliar a capacidade e a eficiência do procedimento de diagnóstico, PACS pode reduzir gastos ao eliminar a necessidade de manutenção de filmes, tratada como dificuldade real encontrada por mantenedores em hospitais ao redor do mundo (MOGHADAM et al., 2015). Esses sistemas começaram a ser concebidos no início dos anos 80, mas só foram adaptados para uso comercial durante os anos 90. Atualmente, apesar do conceito ser conhecido e relativamente difundido, ele ainda não é amplamente adotado por todos os locais que atuam na aquisição de imagens.

2.2.1 Desafios e Oportunidades para PACS

Conforme aumenta a necessidade por imagens e relatórios médicos para apoio ao diagnóstico, há uma lacuna por sistemas PACS que suportem o compartilhamento dessas imagens. A digitalização de filmes físicos também é apontada como uma oportunidade de desenvolvimento nessa área, eliminar a manipulação de cópias materiais facilitaria o armazenamento e compartilhamento destes dados no futuro. Já existem padrões em desenvolvimento para acesso web a objetos DICOM, porém há uma demanda crescente por soluções que estabeleçam métodos ao

manipularmos imagens e metadados através da rede. PACS tem capacidade de revolucionar a maneira como nos relacionamos com a prática médica, diminuindo a exposição desnecessária a fontes de radiação presentes no momento do exame em muitas modalidades. Ao oferecer a oportunidade de diagnósticos emitidos por especialistas à distância, estes sistemas acrescentam qualidade na hora do tratamento. Sem sair do foco da arquitetura, PACS torna-se uma solução com capacidade de difundir a distribuição de imagens e relatórios entre médicos e pacientes (DREYER et al., 2006)

2.3 Compressão de Imagem

A compactação de imagem é um tipo de compactação de dados aplicado a imagens digitais, para reduzir seu custo de armazenamento ou transmissão. Os algoritmos podem aproveitar a percepção visual e as propriedades estatísticas dos dados de imagem, assim fornecendo resultados superiores em comparação com os métodos genéricos de compactação de dados.

Documentos digitalizados são apresentados como uma sequência de imagens digitais, o que gera um grande volume de informação e exige que os sistemas computacionais apresentem grande capacidade de armazenamento, além das demandas na distribuição e apresentação dos arquivos. Um recurso utilizado para otimizar essas demandas de armazenamento, distribuição e apresentação é a aplicação de técnicas de compressão de dados, mais especificamente, em compressão de imagens.

A compressão de dados é alcançada ao reduzir a repetição de símbolos em um arquivo. Exemplos históricos de codificação de dados que se utilizou da compressão e da redundância dos símbolos para transmissão de informação é o código braile e o código Morse. Existem quatro tipos de redundância encontradas em imagens (CONCI; AZEVEDO; LETA, 2008):

- Redundância de codificação de tons ou cor: quando são utilizados mais símbolos de codificação do que o necessário. Por exemplo, uma imagem em preto e branco que foi codificada usando uma tabela de 256 tons de cinza;

- Redundância espacial: também chamada de redundância inter-pixel, é resultante das relações estruturais ou geométricas dos objetivos da imagem. Por exemplo, quando os objetos presentes em uma imagem podem ser descritos apenas por seu contorno;

- Redundância espectral: ocorre quando valores espectrais na mesma posição em faixas diferentes na matriz de *pixels* são correlacionados. Essa redundância é explorada na compressão com perdas do JPEG;

- Redundância psicovisual: relaciona o fato de o sistema visual humano (HVS) não responder com a mesma sensibilidade a todos os estímulos visuais. Por exemplo, o ser humano não consegue detectar pequenas variações de crominância.

Existem dois tipos de compressão de dados: compressão sem perdas (*lossless*), onde todo a informação é transmitida e pode ser fielmente reproduzida após sua descompressão, e a compressão com perdas (lossy), também chamada de destrutiva ou irreversível, uma vez que parte da informação é descartada para aumentar a efetividade da taxa de compressão.

2.3.1 JPEG

JPEG é um método de compactação com perdas comumente usado em imagens digitais, especialmente para aquelas imagens produzidas pela fotografia digital. O grau de compactação pode ser ajustado, permitindo uma troca selecionável entre o tamanho do armazenamento e a qualidade da imagem. O JPEG normalmente alcança a compactação 10:1 com pouca perda perceptível na qualidade da imagem. A figura 3 apresenta a estrutura de um arquivo do tipo JPEG.

O JPEG é um formato de propriedade da Joint Photographic Experts Group (BROWN; SHEPHERD, 1995), que segue o padrão bitmap de representação e permite o armazenamento de imagens coloridas com até 24 bits, não sendo, no entanto, definido nenhum modelo de cor. O tipo de mídia MIME para JPEG é "image/jpeg", exceto em versões anteriores do Internet Explorer, que fornece um tipo MIME de "image/pjpeg"ao carregar imagens JPEG. Arquivos JPEG geralmente têm uma extensão de nome de arquivo .jpg ou .jpeg.

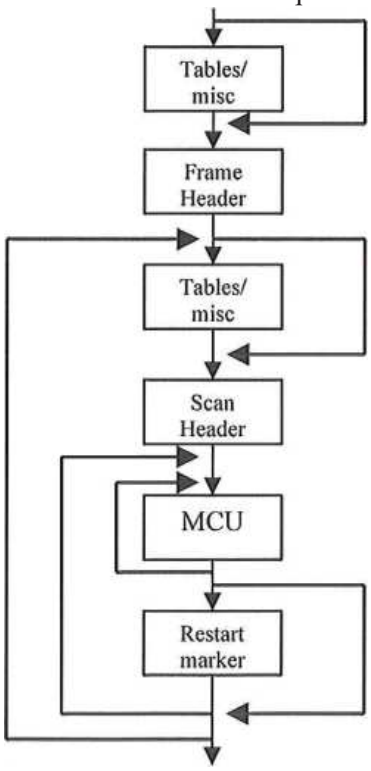
Grande vantagem deste formato é o fato dele permitir o uso de diversas técnicas de compressão, na maioria com perdas: é o formato que oferece a maior taxa de compressão existente para imagens foto gráficas. As técnicas de compressão sem perda aceitas pelo JPEG consistem de uma codificação preditivas sem perdas aliada a alguma outra técnica de compressão, a codificação de Huffman e a codificação aritmética são as mais comuns. Trabalhos realizados por J.F. Neto (NETO; ALCOCER, 1996) obtiveram bons resultados com a técnicas de compressão JPG sem perda utilizando um preditor simples e a codificação de Huffman.

A compactação JPEG é usada em vários formatos de arquivo de imagem. JPEG/Exif é o formato de imagem mais comum usado por câmeras digitais e outros dispositivos de captura de imagens fotográficas, juntamente com JPEG/JFIF, é o formato mais comum para armazenar e transmitir imagens fotográficas na World Wide Web. Essas variações de formato geralmente não são diferenciadas e são chamadas simplesmente de JPEG.

2.3.2 JPEG2000 Sem Perdas

Entre os formatos populares de apresentação de imagens digitais que utilizam técnicas de compressão sem perdas encontra-se o JPEG2000. Apresentado em meados de 2000 como um novo padrão internacional de compressão de imagens, desenvolvido de forma conjunta pela International Organization for Standardization (ISO), International Electrotechnical Commission (IEC) e a JPEG (Joint Photographic Experts Group) (ACHARYA; TSAI, 2005).

Figura 3 – Estrutura de um arquivo JPEG.



Fonte: Elaborado pelo autor.

O Padrão JPEG2000 é baseado na transformação discreta de *wavelets* (DWT), transmissão progressiva de imagem, segundo critérios de qualidade, resolução, componentes ou localização espaciais, e codificação de região de interesse. A compressão JPEG2000 é realizada em uma série de etapas. Inicialmente, é feita o mapeamento da imagem de entrada, seguindo da transformada discreta de *wavelets*, quantização dos coeficientes e pôr fim a codificação de entropia.

Para a descompressão do arquivo, são efetuados os passos inversos, excetuando o passo de quantização, da mesma forma que o padrão JPEG. Uma grande diferença entre esses dois padrões é o fato do JPEG2000 oferecer compressão com perdas e sem perdas, enquanto o JPEG permite apenas uma compressão com perdas. Para cada tipo de compressão são executados algoritmos diferentes no JPEG2000.

Em novembro de 2001, o DICOM Working Group adicionou o suporte para o padrão JPEG 2000, O JPEG 2000 baseado em transformada, e este formato ainda é a técnica de codificação mais adequada até o presente (BRUYLANTS; MUNTEANU; SCHELKENS, 2015). Ele fornece excelente desempenho de distorção de taxa para conjuntos de dados médicos volumétricos (Schelkens et al., 2003), suporta codificação sem perda, escalabilidade de resolução, formatos de arquivo flexíveis e resiliência contra erros de transmissão.

2.4 Cloud computing

Cloud computing (ou computação em nuvem) é um modelo que possui uma rede ubíqua de acesso a recursos de computação configuráveis, que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2011). O avanço das tecnologias de rede proporcionalizou um acesso confiável e de alta velocidade à recursos distantes, através da internet. Esse avanço fez ganhar força a ideia de processamento centralizado em grandes *datacenters* espalhados pelo mundo, ao invés de executar localmente (MARINESCU, 2013).

Computação na nuvem deixou de ser um termo estritamente acadêmico, sendo utilizado nos mais diversos tipos serviços providos através da internet (muitas vezes, sem transparecer ao usuário) (MARINESCU, 2013). Atualmente, foram definidos três modelos de serviço (MELL; GRANCE, 2011):

• *Software as a Service (SaaS)*: Usuário contrata uma aplicação pronta que foi disponibilizada através da nuvem. Toda a infraestrutura fica por conta de quem disponibiliza o serviço, sendo que o usuário não gerencia ou controla os recursos de infraestrutura do ambiente da aplicação (MELL; GRANCE, 2011).

• *Plataform as a Service (PaaS)*: Neste modelo de serviço, o provedor da *cloud* disponibiliza um ambiente configurado para o desenvolvimento de aplicações. Assim como no *Saas*, o usuário não acesso a infraestrutura do ambiente, porém tem acesso as configurações da aplicação a ser disponibilizada através dessa plataforma. Um exemplo de *PaaS*, seria uma *virtual machine* configurada em um servidor, que possua todo o *framework* para desenvolver aplicações em *Python* (MELL; GRANCE, 2011);

• *Infrastructure as a Service (IaaS)*: O provedor de nuvem disponibiliza a infraestrutura para os usuários. Assim, é possível desenvolver ou executar qualquer aplicação, tendo acesso e controle sobre o ambiente, tais como, a rede, o sistema operacional, armazenamento, CPU, etc (MELL; GRANCE, 2011);

Aplicações como servidores de e-mail (como por exemplo, GMail) são um exemplo clássico de um *SaaS*. Os usuários não possuem acesso as configurações da nuvem em que a aplicação está sendo executada, sendo que eles têm apenas acesso a configurações da própria aplicação. Já como *PaaS* temos alguns exemplos como o *Pythonanywhere*, serviço que disponibiliza um ambiente em nuvem pronto para submissão e desenvolvimento de aplicações *Python*. Por fim, *IaaS* disponibiliza um ambiente mais customizável, permitindo configuração de rede, definição do sistema operacional, *storage*, CPU, entre outros.

Além do modelo de serviço, a *cloud computing* possui três modelos de implantação, que dizem respeito a forma de acesso a esses serviços (MELL; GRANCE, 2011). São eles:

• *Private cloud*: São *clouds* providas para um uso específico de uma organização. Pode ser gerenciada pela organização, por terceiros ou por uma mescla entre os dois anteriores. Normalmente, a nuvem está em um ambiente restrito, sem que os dados estejam disponíveis a qualquer

usuário;

• *Public cloud*: São nuvens gerenciadas por uma organização terceirizada, que provê toda a infraestrutura. Uma grande questão nesse modelo de implantação é que a provedora possui acesso a todos os dados da nuvem. Por outro lado, facilita o *deploy* de aplicações sem a necessidade de grandes investimentos em infraestrutura própria;

• *Hybrid cloud*: Combinação entre uma nuvem privada e uma pública. É pertinente em situações em que uma organização possui uma infra de nuvem (*private*), porém pode precisar de mais recursos de uma *public cloud*.

Uma característica importante das *Public clouds*, não característico das *Privates*, é seu conceito de *pay-as-you-go*, que consiste em cobrar o usuário pelo tempo de utilização e quantidade de recursos alocados.

A *cloud computing* só é possível devido a virtualização de recursos através de máquinas virtuais (VM). As VMs podem realizar as mesmas funções computacionais de um computador físico (BIRMAN, 2012). Normalmente, existem grandes *datacenters* que permitem alocar diversas máquinas virtuais, conforme as necessidades de seus usuários (MARINESCU, 2013).

Essa virtualização proporciona alguns benefícios em relação aos multicomputadores antecessores. O primeiro deles é o gerenciamento que é muito mais simplificado, exigindo poucos gerentes para configurar e lançar novos recursos (BIRMAN, 2012). Através de interfaces de acesso (gráfica, linha de comando ou programação), é possível ajustar configurações das máquinas virtuais, tais como, memória, CPU, entre outros. Em comparação a meios físicos, alterar alguma dessas configurações exige a compra de *hardware* e gastar horas de instalação e configuração (BIRMAN, 2012).

Outro ponto forte da computação na nuvem é o baixo gasto energético. Com um bom gerenciamento da *Cloud computing* é possível manter ativas apenas as VMs que são necessárias em determinado momento (RIGHI, 2013). Além disso, os servidores de *cloud* são otimizados para um baixo custo energético, em comparação com a mesma quantidade de processamento de máquinas isoladas que ele substituí.

Dos pontos apresentados até então, nenhum fica muito distante de algo já apresentado em outras formas de sistemas distribuídos existentes antes da *cloud*. O grande diferencial do modelo de computação na nuvem é a elasticidade (RIGHI, 2013).

2.5 Elasticidade

Na definição de computação em nuvem o NIST (National Institute of Standards and Technology) (MELL; GRANCE et al., 2011) aponta que os recursos podem ser provisionados e liberados elasticamente, em alguns casos automaticamente, para escalar de maneira rápida, aumentando os recursos disponíveis e os reduzindo de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento geralmente parecem ser ilimitados e estarem adequados em qualquer quantidade a qualquer momento. Isso faz com que a elasticidade seja

apontada frequentemente como uma das principais características da computação em nuvem (MOLTÓ et al., 2013).

Em termos mais detalhados a elasticidade é a capacidade de adaptação dos recursos *on the fly* de acordo com a necessidade. Existem duas modalidades de tratamento da elasticidade em computação na nuvem, a elasticidade vertical e a elasticidade horizontal (RIGHI, 2013). Independente da forma de elasticidade, ela pode ser um elemento chave na utilização de *cloud* para o tratamento de aplicações que exigem alto desempenho. Por exemplo, podem ser alocadas mais máquinas virtuais para dar conta das tarefas (elasticidade horizontal) ou podemos aumentar algum recurso da máquina virtual (elasticidade vertical) a fim de que ela consiga finalizar mais rapidamente sua tarefa (RIGHI, 2013).

2.5.1 Elasticidade Vertical

A elasticidade é a vertical no contexto de máquinas virtuais é a capacidade de ajustar as configurações de *harware*, aumentando ou diminuindo a sua capacidade (SPINNER et al., 2014). Ou seja, é a possibilidade de aumentar a CPU (quantidade de *cores* e frequência, memória, discos, largura de banda da rede, IOPS, dentre outras possibilidades de configurações, disponibilizando maior capacidade de alocação para a aplicação dentro do mesmo nó computacional.

2.5.2 Elasticidade Horizontal

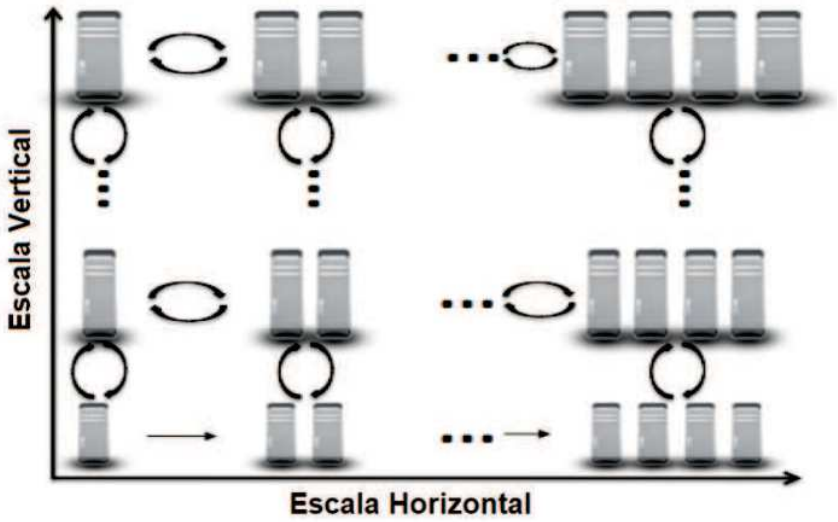
Elasticidade horizontal tem a capacidade de rapidamente provisionar e disponibilizar computacionais (máquinas virtuais, contêineres, discos, dentre outros), a fim de lidar com uma mudança na carga de trabalho (MOLTÓ et al., 2013). Por exemplo, se existe uma máquina virtual encarregada de lidar com as requisições web e em determinado horário de pico o número de requisições é tanto que apenas uma máquina não será suficiente.

Assim, ao identificar este pico o gerente pode (manualmente ou automaticamente) provisionar novas máquinas com um baixo esforço. A figura 4 ilustra tanto o conceito de escalabilidade vertical quanto horizontal, onde no eixo horizontal é possível ver o aumento de quantidade de nós computacionais, através da elasticidade horizontal, e no eixo, vertical é possível ver o aumento de tamanho dos nós, representando o aumento dos recursos dos mesmos, através da elasticidade vertical.

2.5.3 Alocação de Recursos

A alocação de recursos pode seguir duas abordagens (RIGHI, 2013). A primeira é a alocação de recursos manualmente, que exige uma ação do usuário. Os *middlewares* públicos e privados dispõem de ferramentas para efetuarem esses ajustes, normalmente através de uma interface gráfica, linha de comando ou API de programação (RIGHI, 2013). A outra abordagem é de

Figura 4 – Elasticidade Vertical e Horizontal.



Fonte: (AL-DHURAIBI et al., 2017), traduzida pelo autor

forma automática, sendo que essa pode ser dividida em duas formas de tratamento: reativa e proativa (RIGHI, 2013).

Na forma reativa, são tomadas decisões de elasticidade de acordo com regras definidas estaticamente. Grande parte dos sistemas comerciais atuais, como Amazon AWS, Microsoft Azure e Google Cloud Platoform, utilizam esta forma de lidar com a elasticidade. As regras definidas se baseiam em limiares (também chamado de *thresholds*), que quando atingidos, disparam uma ação de elasticidade. Um exemplo de elasticidade reativa é indicar que, se uma máquina virtual atingir 20% de CPU, ela deve ser desligada (elasticidade horizontal) ou deve diminuir sua quantidade de CPU-Cores ou frequência (elasticidade vertical).

A nuvem se adapta de acordo com os limiares estabelecidos, porém, a elasticidade reativa tem dois grandes problemas. O primeiro deles é definir quais são os melhores *thresholds* para uma determinada aplicação. Isso não é algo trivial, requer habilidade do administrador na ferramenta de gerenciamento e uma análise de cada aplicação separadamente (RIGHI, 2013).

O segundo problema é que a elasticidade reativa toma uma decisão quando um recurso atingir um estado não desejável (definido pelo limiar). Na elasticidade horizontal, após disparar a ação de adicionar um recurso novo, existe um período de tempo em que esse recurso leva para ser provisionado e disponivel para uso. Ou seja, após atingir um estado não desejável, ficará nesse estado até que o novo recurso esteja pronto para ser utilizado. Esse tempo varia em cada gerenciador de nuvem, podendo variar de acordo com o tamanho da VM, hardware do *host*, tempo de inicialização dos sistemas e aplicações dentro da VM, etc.

Mas alguns autores indicam que o tempo de instanciação de uma máquina virtual está entre 5 e 15 minutos (BANKOLE; AJILA, 2013; BREBNER, 2012). Visando resolver os problemas da forma reativa (principalmente, o segundo problema), existe a abordagem proativa. Nessa abordagem é utilizado os dados históricos para detectar padrões e prever qual o melhor momento para tomar uma decisão de elasticidade (RIGHI, 2013). Assim, o problema do tempo de

inicialização de um recurso é suavizado, tendo em vista que será levado em conta esse tempo na previsão.

Para essa previsão, normalmente são usados algoritmos de aprendizagem de máquina ou cálculos de séries temporais (ROSA et al., 2014; GONG; GU; WILKES, 2010; LOFF; GARCIA, 2014; ROY; DUBEY; GOKHALE, 2011; MOORE; BEAN; ELLAHI, 2013; BARRETT; HOWLEY; DUGGAN, 2013; NIKRAVESH; AJILA; LUNG, 2017). Na Figura 5 é apresentado um exemplo simplificado de como poderia se comportar a CPU de um sistema e como funciona a predição. Ao prever que será necessário um novo recurso, ele será instanciado. Após isso, a máquina virtual leva alguns ciclos para estar pronta, porém ao estar disponível gera uma suavização da CPU, apresentado na figura 5.

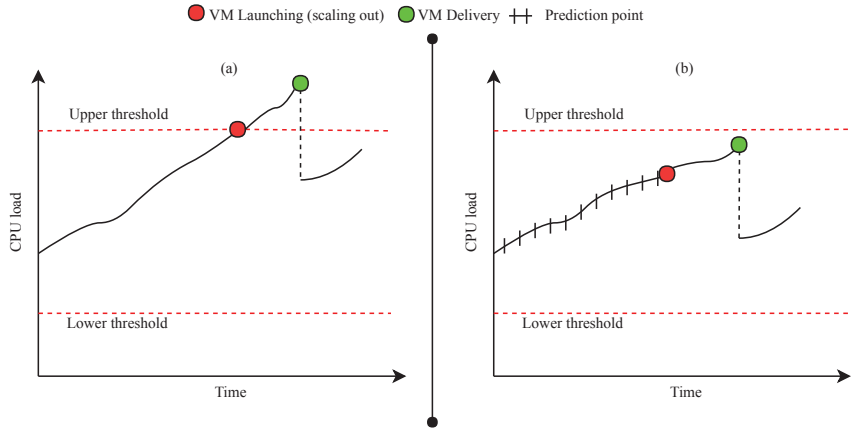


Figura 5 – Suavização após recurso disponível.

Um exemplo da utilização de elasticidade horizontal, pode ser uma *Web Farm*, que de acordo com a carga de trabalho, adicionam-se mais VMs para lidar com o aumento de requisições. Esse é um exemplo clássico de elasticidade que utiliza a estratégia de replicação (RIGHI, 2013). Além da estratégia acima, existem duas outras que podem ser utilizadas para prover elasticidade. Uma delas, é a migração de recursos, que consiste em migrar VMs entre recursos físicos (RIGHI, 2013).

2.6 Escalabilidade

Escalabilidade é a capacidade de um sistema para lidar com o crescimento na quantidade de trabalho a ser feito. Diz-se que um sistema é escalável mesmo se o seu comportamento se degrada definitiva e visivelmente, desde que tal degradação permaneça aceitável para os usuários do sistema. Segundo Buyya (XU; TIAN; BUYYA, 2017), escalabilidade em sistemas computacionais surge quando esses sistemas são capazes de satisfazer maior demanda ou prestar um volume maior de serviços com um aumento adequado de recursos.

Para um sistema distribuído, uma das características mais importantes é ser escalável no número de componentes envolvidos e com isto o sistema deve manter, por exemplo, o mesmo

nível de disponibilidade à medida em que o número de componentes cresce, aumentando o desempenho global do sistema. Em termos mais práticos a escalabilidade é a habilidade que uma aplicação tenha de suportar um crescente número de requisições a serviços e demais recursos do sistema oriundos dos usuários.

Exemplo: se uma aplicação leva 1ms para responder uma requisição, quanto tempo ela levará para responder 1.000 requisições concorrentes? Escalabilidade infinita permitiria que a aplicação respondesse essa carga em 1ms, ou seja, gastaria-se o mesmo tempo para responder 1 requisição ou 1.000 requisições. Escalabilidade, então, pode ser entendida como uma medida que depende de vários fatores, incluindo o número de usuários simultâneos que um sistema computacional configurado como um *cluster* pode suportar e o tempo que se leva para responder uma requisição.

2.7 Balanceamento de Carga

Em nuvens computacionais é comum a utilização de um balanceador de carga (XU; TIAN; BUYYA, 2017; MILANI; NAVIMIPOUR, 2016; SINGH; JUNEJA; MALHOTRA, 2015) como um ponto único de contato com os clientes. Cada balanceador de carga pode receber diferentes requisições de acesso utilizando protocolos e portas distintos, e encaminhar essas solicitações para diferentes recursos de acordo com as condições, alvos e prioridades configuradas de previamente. Estes sistemas distribuem os pedidos provenientes de diferentes fontes com base em métricas e algoritmos definidos, podendo balancear o uso de recursos computacionais como, máquinas virtuais, banco de dados, discos ou mesmo redes.

Problemas de desempenho, incluindo baixos tempos de resposta, congestionamento da rede e interrupções de serviço são consequências de crescimento constante e na maioria das vezes desarticulado dos sistemas computacionais e também de infra-estrutura de redes de comunicação de dados. Pode ser adicionado a este o fato de que todo hardware ou qualquer outro recurso de um sistema computacional possui um limite físico. Uma possível abordagem na tentativa de melhorar o desempenho do sistema é desenvolver o mecanismo de balanceamento de carga para a aplicação que se encontra sobrecarregada.

Balanceamento de Carga (*Load Balancing*) é um mecanismo usado para atingir escalabilidade, dividindo a carga de processamento entre um conjunto de duas ou mais máquinas (TEODORO et al., 2003). O objetivo específico de desenvolver balanceamento de carga para um sistema computacional crítico é promover sua melhoria de desempenho, através da distribuição das tarefas a serem executadas. O funcionamento se resume em distribuir o tráfego das chamadas ao sistema, fazendo com que as diferentes máquinas funcionem como uma única.

Também mantém o tempo de resposta das requisições, de acordo com valores limites, e oferece escalabilidade de serviços e recursos, ou seja, à medida em que houver aumento de demanda (novas aplicações, maior número de usuários conectados, etc), mais máquinas podem ser incorporadas ao conjunto, otimizando assim o poder de resposta. Estas soluções podem ser

especializadas em pequenos grupos sobre os quais se faz um balanceamento de carga, o intuito é otimizar o uso de vários recursos, tais como processador, disco, ou de rede. Qualquer uma delas introduz o conceito de *cluster*, ou *server farm*, já que o balanceamento será, provavelmente, feito para vários servidores. Um *cluster* é formado por um conjunto de computadores, que utiliza-se de um tipo especial de sistema operacional classificado como um sistema distribuído (BUYYA et al., 1999).

2.8 Séries Temporais

A análise de séries temporais tem como objetivo buscar alguma relação de dependência temporal sobre os dados, identificando o mecanismo gerador da série, extraindo-se periodicidades que sejam relevantes nas observações, descrever o seu comportamento e realizar previsões.

2.8.1 Estacionaridade

Uma conjectura adotada com frequência em uma serie temporal, é que a ela seja estacionaria, significando que ela se desenvolve ao longo do tempo aleatoriamente em torno de uma média constante, estando assim, em equilíbrio estável. No entanto, na prática, a maioria das series são não-estacionarias. O modelo ARIMA é capaz de descrever de maneira conveniente a series estacionarias e não-estacionarias (MORETTIN; TOLOI, 2006)

2.8.2 ARIMA

A classe de modelos ARIMA (Autorregressivos Integrados de Médias Móveis), propostos por Box & Jenkins em 1970 (por esse motivo, também são conhecidos na literatura como modelos Box & Jenkins), é a mais utilizada em análise de séries temporais. O modelo ARIMA pode ser identificado como um todo no modelo ou parcialmente, ou seja, pode-se identificar modelos AR(p) (apenas a parte autorregressiva é importante para modelar a série em questão) ou modelos MA(q) (apenas a parte de médias móveis é interessante para o modelar a série).

Neste âmbito, tem-se os modelos mistos compostos pela parte autorregressiva e de médias móveis, com ou sem diferenciação. A notação utilizada é ARIMA(p, d, q), em que assume-se que p, d, q $\in \mathbb{Z}+$ (BAYER et al., 2008), onde (EHLERS, 2007):

- p é o número de termos auto regressivos;
- d é o número de diferenças não sazonais necessárias para a estacionariedade;
- q é o número de erros de previsão atrasados na equação de previsão.

De acordo com essa parametrização, o ARIMA terá diferentes comportamentos. Por exemplo, ARIMA(0,0,0) é um média simples. Se aumentarmos o valor de p, o modelo passa a ser

um modelo auto regressivo (AR). Já ARIMA(0,1,0) é considerado um *random walk*. Por fim, ARIMA(0,1,1) é uma *Exponential smoothing* (EHLERS, 2007). Esses são apenas alguns dos exemplos de parametrização do ARIMA, sendo possível variar estes valores para definir o melhor modelo para um problema.

Tendo em vista a previsão, cada um desses algoritmos apresentará resultados diferentes. Por exemplo, o ARIMA(0,0,0) retornará como um valor previsto a simples média dos valores anteriores. Já o ARIMA(0,1,1) retornará a média ponderada dos valores mais recentes observados (EHLERS, 2007). Um algoritmo interessante do ARIMA é o (0,2,0). Este algoritmo é uma evolução do ARIMA(0,1,0) que é um *random walk*. O algoritmo de *random walk* prevê os valores futuros retornando o valor mais recente apresentado. Então se o último valor lido for 35, o valor futuro será apresentado como 35 também. Já o (0,2,0) utiliza o valor mais recente, porém aplica sobre ele uma tendência, calculada com a taxa de modificação dos últimos valores.

2.9 Proliot - Proactive Elasticity Model

Na Internet das Coisas (IoT) o uso do serviço de elasticidade da computação em nuvem é importante pois os *middlewares* devem ser capazes de tratar um alto volume de informações em tempo real. Esses dados podem chegar no *middleware* em paralelo, tanto se tratando de dados de entrada lido de sensores quanto dados de saída de resposta a requisições feita por usuários.

O modelo Proliot, é um modelo que combina computação de alta performance para tratar problemas de escalabilidade em aplicações que lidam com Internet das Coisas na arquitetura EPCGlobal. Neste modelo é criado um *middleware* que tem os mesmos contratos (API) do EPCIS, mas oferece suporte da elasticidade via computação em nuvem no EPCIS. O Proliot consiste de um formalismo matemático que usa um modelo auto regressivo integrado de médias móveis (*autoregressive integrated moving average* ou ARIMA, na sigla em inglês) e média móvel ponderada para prever comportamento da carga a ser requisitada no sistema, então, podendo antecipar o escalonamento deixando o provisionamento das VMs mais próximas ao momento ideal, evitando que o sistema fique sobrecarregado.

O modelo também usa média móvel exponencial, também conhecida como *Aging*, para prever o melhor momento de desalocar os recursos, ou seja, o momento que eles não são mais necessários. A média móvel exponencial é responsável por suavizar as informações de carga do sistema, evitando assim falsos positivos nas operações de escalabilidade.

3 TRABALHOS RELACIONADOS

Este capítulo tem como objetivo apresentar os trabalhos relacionados ao modelo proposto, considerando pesquisas privadas e acadêmicas. Um dos pontos iniciais deste trabalho foi entender o estado da arte em relação à imagens médicas (DICOM), no aspecto de performance e elasticidade. Para isso foi utilizada a metodologia de mapeamento sistemático, para encontrar os artigos relacionados a este trabalho. Os critérios utilizados na seleção dos trabalhos são especificados na Seção 3.1. Na Seção 3.2, são apresentados os trabalhos relacionados e por fim, na Seção 3.3, pode-se conferir as oportunidades de pesquisa.

3.1 Metodologia de Pesquisa e Escolha dos Trabalhos Relacionados

A primeira etapa foi a elaboração de algumas perguntas pertinentes à temática que deveriam ser respondidas ao longo do mapeamento sistemático, apresentadas na Tabela 2.

Tabela 2 – Questões de Pesquisa.

ID	Questão
1	Quais protocolos descritos no DICOM e usados na prática?
2	Como a tecnologia está sendo aplicada para prover desempenho ao DICOM?
3	Como a tecnologia está sendo aplicada para prover elasticidade e escalabilidade ao DICOM?
4	Quais as métricas e algorítimos são usados para avaliar a performance do DICOM?

Para selecionar os trabalhos relacionados que responderiam as perguntas uma *string* de busca foi criada a partir dos termos “DICOM”, “performance”, “elasticity” e “scalability”:

Listing 3.1 – *String* de busca do mapeamento sistemático.

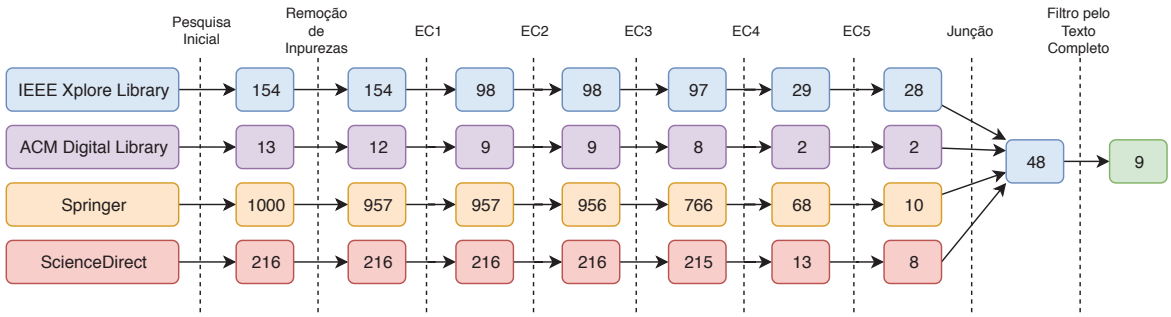
```
"DICOM" AND ("performance" OR "elasticity" OR "scalability")
```

A pesquisa dos trabalhos relacionados foi realizada utilizando-se os seguintes portais: *IEEE Xplore Library*, *ACM Digital Library*, *Science Direct* e *Springer Library*. Cada portal possui sua forma de pesquisa, sendo que a *string* apresentada pode variar de portal para portal. As consultas executados resultaram em um grande volume de artigos, e para que fosse reduzido o ruído e assim filtrar os trabalhos relacionados à temática, foram estabelecidas alguns critérios de exclusão (EC) que ajudaram na classificação e agrupamentos dos artigos, os quais são apresentadas a seguir.

- EC1: Estudos que foram publicados anteriormente à 2009;
- EC2: Estudos que não foram escritos em Inglês;
- EC3: Teses ou estudos publicados em revistas;

- EC4: Estudos que não abordam o desempenho do DICOM, escalabilidade ou compressão de arquivos DICOM;
- EC5: Estudos que não estão relacionados a questão de pesquisa.

Figura 6 – Processo de seleção de artigos.



Fonte: Elaborado pelo autor.

A Figura 6 demonstra o processo de seleção dos artigos. Inicialmente, encontraram-se 1383 artigos e, após refinamento, onde foram removidas impurezas, excluiu-se 1335 (96,52%) artigos. Foram removidos resultados que não eram artigos, tais como, conferências e resumos sem dados ou detalhamentos, que se enquadravam na *string* de pesquisa.

Em seguida, foram mesclados os artigos de cada *digital library*, resultando num total de 48 artigos. Após, aplicou-se um filtro por leitura de texto completa, que consistiu na avaliação de cada um dos trabalhos selecionados, para verificar se estes eram relevantes para o tema da pesquisa. Esta etapa resultou num total de 9 (18,75%) artigos.

Após estes critérios atendidos, 9 trabalhos que concentram-se no uso de compressão em imagens DICOM, transferência de imagens DICOM via Internet e também armazenamento destas imagens foram selecionados como trabalhos relacionados e serão apresentados nas próximas subseções.

3.2 Análise e Oportunidades de Pesquisa

Nesta seção, serão apresentados os detalhes de cada um dos trabalhos selecionados como trabalhos relacionados, visando utilizar os resultados para determinar o modelo a ser criado.

3.2.1 Performance Analysis of Medical Image Compression

No trabalho de (VIDHYA; SHENBAGADEVI, 2009), é apresentado um cenário onde a comunidade médica tem sido muito relutante em adotar algoritmos de compressão com perdas. Em contrapartida as informações de diagnósticos produzidas por hospitais tem aumentado geometricamente, e a técnica de compressão requerida para reduzir a quantidade de espaço de armazenamento e melhor aproveitar a velocidade de transmissão via rede.

Então, neste trabalho é apresentado um algoritmo efetivo para comprimir e reconstruir imagens DICOM, tendo esta nova abordagem melhor performance, quando comprimindo imagens médicas. Atualmente o padrão DICOM é baseado na compressão JPEG, no entanto, neste trabalho, apresenta-se que codificação wavelets é mais eficiente que a compressão do JPEG. A Transformada discreta de wavelets da imagem é calculada com o filtro bi ortogonal CDF. O coeficiente da wavelets são codificados usando SPIHT, dando assim um melhor resultado de compressão que outros métodos tradicionais.

No experimento foram usados um conjunto de imagens DICOM contendo as series completas de imagens transversais MR. As imagens tinham resolução de 256 x 256 pixels e 512 x 512 pixels. O tempo de codificação e decodificação aumenta conforme o bit rate aumenta. Como foi executado o experimento em várias imagens, elas não foram processadas em paralelo, quando o codificador e o decodificador terminam todas as fatias (slices) em uma sequência (uma imagem), ele muda para processar a próxima sequência de fatias (outra imagem).

O um dos objetivos deste trabalho é comparar as medidas de qualidade objetiva do método proposto com compressão padrão JPEG e algoritmos de compressão de imagem Fractal. Então, como resultado temos que o desempenho do método proposto para imagens DICOM é muito melhor do que as técnicas JPEG e Fractal.

3.2.2 Performance Analysis of Medical Image Compression Techniques

A compactação de imagens tornou-se uma das disciplinas mais importantes da eletrônica digital, devido à crescente popularidade e ao uso da Internet e dos sistemas multimídia combinados com os altos requisitos de largura de banda e espaço de armazenamento. O crescente volume de dados gerados por alguns equipamentos de imagens médicas, o que justifica o uso de diferentes técnicas de compressão para diminuir o espaço de armazenamento e a eficiência de transferência as imagens através da rede para acesso a registros eletrônicos de pacientes.

O trabalho de (PINTO; GAWANDE, 2012) aborda a área de compactação de dados, conforme aplicável ao processamento de imagens, apresentando um algoritmo eficaz para comprimir e reconstruir imagens digitais médicas (DICOM). Vários algoritmos de compressão de imagem existem no mercado comercial de hoje. Este artigo descreve a comparação de métodos de compressão como JEPG, JEPG 2000 com codificação SPIHT com base na taxa de compressão e na qualidade de compressão.

A comparação desses métodos de compressão é classificada de acordo com diferentes imagens médicas, como ressonância magnética e tomografia computadorizada. Para compactação de imagem baseada em JPEG, as técnicas de codificação RLE e Huffman são usadas variando os bits por pixel. Para compressão de imagem baseada em JPEG 2000, o método de codificação SPIHT é usado. Os métodos DCT e DWT são comparados variando os bits por pixel e medidos os parâmetros de desempenho do MSE, PSNR e taxa de compressão. No método JPEG 2000, é comparado as diferentes wavelets como Haar, CDF 9/7, CDF 5/3, dentre outras. É avaliado

a taxa de compressão e a qualidade de compressão. São apresentados vários resultados, sendo eles que:

1. Pode-se alcançar maior taxa de compressão para imagens de ressonância magnética do que imagens de tomografia computadorizada;

2. Para o método de compactação JPEG de imagem de tomografia computadorizada, supera o PSNR e o grau de compactação do que o método de compactação wavelets;

3. Embora o RLE seja um método sem perdas e mantenha a qualidade da imagem bem, ela não pode ser aplicada a imagens médicas, pois tem uma taxa de compactação muito baixa;

4. Para uma taxa de compactação menor, o JPEG gerou uma imagem de qualidade superior à wavelets;

5. Embora os codificadores JPEG de imagem baseados em DCT tenham um desempenho muito bom em taxas de bits moderadas, em taxas de compactação mais altas, a qualidade da imagem diminui devido aos artefatos resultantes do esquema DCT baseado em bloco

6. A codificação baseada em wavelets, proporciona melhoria substancial na qualidade da imagem em baixas taxas de bits, devido à sobreposição de funções básicas e melhor propriedade de compactação de energia de transformadas wavelets.

7. O filtro wavelets CDF 9/7 fornece os melhores resultados para PSNR e CR do que o CDF 5/3 e o Haar;

8. SPIHT é o método mais eficiente em relação à taxa de compressão e ao valor PSNR. Com o aumento do grau de compressão, o SPIHT mantém a qualidade da imagem;

9. O tempo de codificação e decodificação aumenta conforme a taxa de bits aumenta;

10. E por fim, concluem que os resultados do seu algoritmo são muito satisfatórios em termos de taxa de compressão e qualidade de imagem comprimida, porém ainda se tem que aplicar essa técnica em diferentes imagens médicas, para mais constatações.

3.2.3 Towards a Hybrid Row-column Database for a Cloud-based Medical Data Management System

No trabalho de Mohamad (2012) (MOHAMAD; D'ORAZIO; GRUENWALD, 2012), faz uso da computação em nuvem com suas características de elasticidade, alta disponibilidade e cobranças de demandas de uso, para gerenciar o armazenamento dos dados de imagens médicas (DICOM). Como esses arquivos contém informações heterogêneas é proposto um novo modelo

hibrido de arquitetura de banco de dados linha-coluna. (Reduzir os custos totais de armazenamentos. Elasticidade garante um bom tempo de resposta, mesmo nos momentos de pico do sistema, porque oferece escalabilidade ao aumentar os recursos computacionais, e deslocarmos quando não mais necessários, isso inclusive aumenta a tolerância a falhas.)

A solução proposta neste trabalho é de fácil uso, extensível, de alta performance e faz uso da elasticidade e modelo de cobrança do ambiente em nuvem. O DICOM foi decomposto em 3 categorias: (1) atributos obrigatórios / frequentemente usado. (2) atributos frequentemente acessados juntos. (3) atributos opcionais / privados. Foi proposto uma camada especifica para armazenar cada categoria e cada uma dessas camadas são vinculadas através de um identificador único “row-id”, que permite a reconstrução dos arquivos DICOM.

3.2.4 An Adaptive Streaming Technique for Interactive Medical Systems in Mobile Environment

Os tópicos relacionados aos serviços médicos móveis são também importantes na hora de projetar e implementar sistemas de informação médica. (PARK et al., 2009) propõem um método para a sincronização de informações entre dispositivos móveis e *datacenters* médicos. Isto ajuda a médicos a obter a informação mais atualizada dos pacientes, quem serão utilizadas em diagnósticos instantâneos. Se propõe que isto seja feito através de ambientes *wireless*, que além de informações dos pacientes também possibilitam a transferência de imagens em *streaming*.

As tecnologias envolvidas na demonstração do conceito são DICOM, JPEG2000, protocolo interativo (JPIP) para *streaming* e WiBro para conexão *wireless* com a internet. Essa abordagem torna as informações médicas de fácil acesso, de muitas diferentes formas e prove uma informação precisa ao integrar vários sistemas médicos, isso apoia a triagem a qualquer momento, o que encurta o tempo da chegada na sala de emergência até o tratamento.

O sistema ajusta a qualidade da imagem de acordo com o dispositivo que fará a exibição. Essa alteração na qualidade da imagem é realizada por um *middleware*, localizado entre o PACS Server (Service Class Provider - SCP) e o cliente (Service Class User - SCU). O uso do *framework* desenvolvido neste trabalho demonstra uma melhoria na qualidade do cuidado e da segurança do paciente aumento a produtividade clínica e reduzindo os riscos de erros médicos.

3.2.5 Clustering of distinct PACS archives using a cooperative peer-to-peer network

Para enfrentar os exigentes requisitos dos ambientes clínicos, os servidores PACS (Picture Archiving and Communication System) precisam ser resilientes e confiáveis, suportando alta disponibilidade e tolerância a falhas. Muitas vezes, para garantir que não haja perda de dados, os arquivos do PACS retêm duas cópias de imagens em máquinas físicas separadas, usando recursos de armazenamento de dados distribuídos. No entanto, o PACS não aproveita as várias réplicas para melhorar as taxas de transferência de imagens médicas.

Isso acontece principalmente porque o padrão DICOM não está de acordo com a distribuição distribuída de fragmentos de imagens durante a execução de uma loja. Inspirados por essa oportunidade inexplorada, (RIBEIRO; COSTA; OLIVEIRA, 2012) projetaram e implementaram uma nova solução que aproveita as réplicas de imagens distribuídas e, ao mesmo tempo, respeita o padrão DICOM. Nossa estratégia trouxe melhorias significativas nas taxas de câmbio, balanceamento de carga e disponibilidade de arquivos PACS instalados. Além disso, a estratégia adotada forma um conjunto de arquivos PACS que permite, de forma transparente, o escalonamento horizontal, facilita a criação de backups e oferece aos profissionais de saúde uma visão unificada dos repositórios distribuídos.

3.2.6 Web-based Picture Archiving and Communication System for Medical Images

O trabalho de (Palma et al., 2010) apresenta um sistema PACS versão web utilizando a arquitetura cliente-servidor, ou seja, um dos lados trata todas as imagens e informações, que permitem o envio e recebimento de imagens. Do outro lado temos a implementação de uma ferramenta para visualização e manipulação das imagens médicas.

Sua implementação tem como requisito ser uma aplicação totalmente baseada na web, ou seja, online. Desta forma a aplicação se torna mais portável e sustentável, com tecnologias que permitem estar disponível em qualquer lugar e a qualquer momento. Na implementação do sistema é utilizado C# com o framework Windows Presentation Foundation (WPF) e a comunicação é utilizado o framework Windows Communication Foundation (WCF).

O modelo deste trabalho contém 3 módulos: visualizador do paciente (com informações sobre o paciente), visualizador de imagem (onde é feito o diagnóstico) e gerador de relatório (onde são inseridos elementos na imagem para identificação de observações). No lado do servidor são implementados três serviços web. São dois serviços de imagem DICOM (um para imagens grandes e outro para imagens pequenas), o terceiro serviço web é dedicado para transmissão de informações do paciente de forma codificada em base64.

3.2.7 MIAPS: A web-based system for remotely accessing and presenting medical images

Segundo (SHEN et al., 2014) acesso remoto à imagens médicas através da web é importante para a radiologia moderna. Segundo os autores do trabalho isso viabiliza que radiologistas possam acessar imagens médicas através de um navegador de qualquer lugar, a qualquer momento, através de diferentes tipos de dispositivos, sejam eles estacionários ou moveis, sem a necessidade de manutenção de um sistema cliente. Isso ajuda a reduzir a carga de trabalho da radiologia e a aumenta a produtividade.

Também apoia em soluções mais eficientes para a tele-radiologia através do compartilhamento de imagens médicas. MIAPS é um sistema para web para acesso remoto a imagens DICOM. Este sistema é acessível de um navegador via internet. MIAPS oferece quatro carac-

terísticas fundamentais: recuperação, manutenção, apresentação e saída das imagens DICOM. Um dos pontos relevantes deste trabalho é o modelo DIIE (DICOM Image Indexing Engine), que realiza a indexação das informações contidas em um arquivo DICOM afim de possibilitar consultar e recuperação de informações.

O servidor que fornece o serviço HTTP responsável por responder as requisições feitas pelo navegador está instalado dentro dos domínios do hospital, requerendo mudanças no ambiente de TI para suportar estes serviços. Este sistema não tem a intenção de substituir softwares comerciais existentes na área de radiologia, mas apenas oferecer uma solução web para tele radiologia. O Sistema foi testado em 39 hospitais por 10 meses na China. Ao final do período foi realizada uma análise qualitativa, que demonstrou que 90.90% dos hospitais acreditam que o sistema tem valor para o diagnóstico clínico.

3.2.8 Integration of a Zero-footprint Cloud-based Picture Archiving and Communication System with Customizable Forms for Radiology Research and Education

O propósito do estudo proposto por (HOSTETTER; KHANNA; MANDELL, 2018), é criar uma ferramenta de armazenamento de imagens e comunicação (PACS) baseado na web, para beneficiar a educação e a pesquisa em radiologia. Para isso ele usou tecnologias de front-end e back-end, que incluem vue.js, node.js e mongodb, e integrou dentro de um ambiente de nuvem. As aplicações baseadas na web podem ser acessadas através de qualquer navegador de internet moderna, no desktop ou em dispositivos moveis, e permitem a criação de páginas personalizadas e formulários de questões de vários tipos.

Cada formulário de questões pode ser vinculado a um arquivo DICOM individual ou uma coleção de exames DICOM. Existem algumas limitações importantes no projeto deste trabalho, por exemplo, os exames DICOM precisam ser exportados manualmente de um PACS e enviados para o sistema web, o que pode demandar tempo. Imagens de ultrassom e imagens pós processados não são suportados, e não é possível envia-las ao sistema web pois a anonimização a nível de pixel não é realizada.

3.2.9 A Community-driven Validation Service for Standard Medical Imaging Objects

Atualmente, nem o DICOM Standards Committee nem o NEMA têm uma ferramenta oficial de valide as inconsistências nas implementações DICOM, então, no trabalho de (SILVA et al., 2019) propõem-se um serviço web para validação dos objetos DICOM, que não requer configuração do usuário, promove a validação dos resultados e a possibilidade de compartilhamento preservando a privacidade do paciente. A arquitetura da plataforma proposta é composta de três camadas: *frontend*, *backend* e um painel colaborativo para edição de arquivos de descrição.

No *frontend* temos um desenvolvimento utilizando Angular e Boostrap, consumindo o *backend* via REST API. Tem-se como vantagem de o serviço ser *web*, a não ser necessidade im-

plantar um sistema local, diferenciando-se também dos seus trabalhos relacionados que tem sua capacidade de estar atualizados em relação aos padrões. Ademais, a plataforma expõe-se como colaborativa, onde programadores e times de pesquisa acadêmicos e da indústria podem desenvolver novos módulos de validação, decidir quais módulos estão incluídos no seu fluxo de validação, bem como compartilhar seu módulo desenvolvido com a comunidade.

Os autores não mencionam questões de performance e escalabilidade da sua aplicação, mas apontam que como a análise e validação não requer informações da imagem capturada, mas sim, das *tags* incluídas no arquivo, remove-se a parte de *pixel-data*, o que colabora ativamente para redução da necessidade do espaço em disco.

3.3 Comparação dos trabalhos relacionados

Na Tabela 3 são apresentados os trabalhos com maior correlação, encontrados nas pesquisas realizadas. Essa tabela demonstra algumas oportunidades de pesquisa, onde é importante analisar qual o tipo de aplicação apresentado no artigo, pois as aplicações possuem comportamentos diferentes entre si. Assim, o sistema proposto pode se beneficiar da soma das características das aplicações encontradas.

Tabela 3 – Comparação dos trabalhos relacionados. A sigla NA é usada para indicar que o item não se aplica ao trabalho relacionado, ou literalmente não foi abordado pelo autor do trabalho.

Artigo	Tipo de trabalho	Algoritmo de Compressão	Arquitetura / Ambiente	Elasticidade	Modelo de decisão	Telemedicina
(VIDHYA; SHENBAGADEVI, 2009)	Algoritmo de Compressão	Proprietário	Local	Sem Elasticidade	NA	NA
(PINTO; GAWANDE, 2012)	Algoritmo de Compressão	Proprietário	Local	sem elasticidade	NA	NA
(MOHAMAD; D'ORAZIO; GRUENWALD, 2012)	Sistema de Armazenamento	Sem Compressão	Nuvem Publica	Não Mencionado. Provavelmente, Vertical e Horizontal Oferecida Pela Nuvem	Não Mencionado. Provavelmente, Reativa Oferecida Pela Nuvem	NA
(PARK et al., 2009)	Sistema Móvel Para Integração De Dados Dos Pacientes Incluindo DICOM	JPEG2000	Local	sem elasticidade	NA	Sem Compartilhamento
(Palma et al., 2010)	Sistema Web Para Acesso A Imagens DICOM	JPEG (com perdas)	Nuvem Privada	Sem elasticidade	NA	Sem Compartilhamento
(SHEN et al., 2014)	Sistema Web Para Acesso DICOM Externo Ao Hospital	JPEG2000	Local	Sem Elasticidade	NA	Sem Compartilhamento
(HOSTETTER; KHANNA; MANDELL, 2018)	Sistema PACS Em Nuvem Para Estudos Científicos	Sem Compressão	Nuvem Publica	Não Mencionado. Provavelmente, Vertical e Horizontal Oferecida Pela Nuvem	Não Mencionado. Provavelmente, Reativa Oferecida Pela Nuvem	Compartilhamento Com Estudantes. Sem telemedicina, pois envio das imagem não é posteriormente)
(SILVA et al., 2019)	Sistema Validador De Formato De Imagens DICOM	Sem Compressão	Local	Sem Elasticidade	NA	NA

Fonte: Elaborado pelo autor.

A primeira oportunidade, mais evidente, é que não foram encontrados artigos que façam o compartilhamento das imagens médicas entre hospitais. Foi identificado um trabalho que aborda o compartilhamento das imagens (HOSTETTER; KHANNA; MANDELL, 2018), porém apenas para fins de estudos, onde as imagens são enviadas em lote para o servidor antes de executar-se um estudo analítico. Com isso, esta abordagem não busca viabilizar o compartilhamento das imagens em âmbito geral, o que agilizaria o tempo de atendimento ao paciente.

É crítico para um sistema médico que seu funcionamento e desempenho não sejam afetados por questões de limitação de hardware, tal qual, é característico em modelos de elasticidade reativa. Nenhum dos trabalhos relacionados apresentou um modelo de elasticidade proativa ou outra abordagem para tratar desse tema, que auxilia a manter o nível de serviço. Um ponto limitante para o compartilhamento de imagens é a largura de banda que conecta os hospitais à internet.

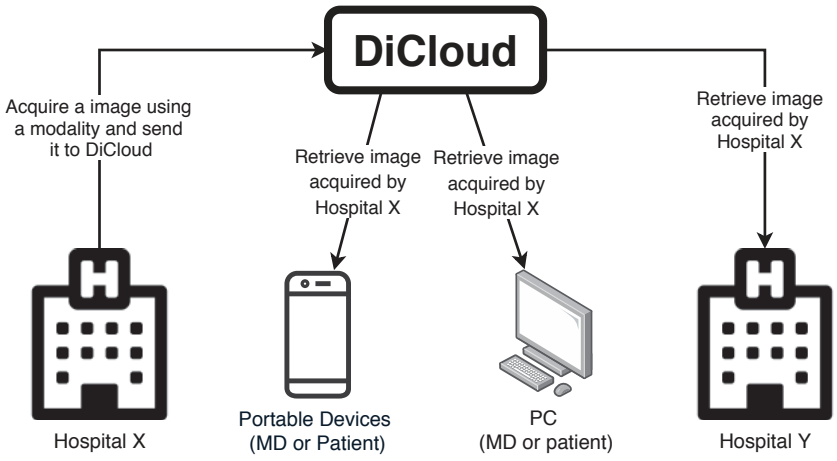
É importante otimizar as informações que são enviadas e recebidas, de modo a minimizar o tráfego intensivo, não congestionando o *link* com a internet. Foram encontrados trabalhos que abordam métodos de compressão de imagens DICOM, porém, os trabalhos que visam disponibilizar as imagens DICOM via web, não aplicam compressão. A aplicação de compressão nestes casos é basilar, uma vez que arquivos DICOM podem chegar a tamanhos relativamente grandes (como apresentado no item 2.1.3), demandando uso sequencial e intensivo do *link*, podendo afetar os demais sistemas hospitalares que também necessitem de conexão com a internet.

4 MODELO DICLOUD

Este capítulo objetiva descrever o modelo DiCloud, um modelo de compartilhamento de imagens médicas que se beneficia de elasticidade proativa em computação em nuvem para reduzir custos e evitar efeitos negativos na performance das aplicações.

O DiCloud aborda a comunicação entre hospitais, pacientes e especialistas, atuando como um *hub* de interconexão entre eles. Cada elemento exerce um papel distinto no modelo, inclusive, com fluxo de dados diferentes entre si. A Figura 7 apresenta uma visualização simplificada da comunicação entre os elementos, cujos quais desmembraremos em detalhes nas subseções.

Figura 7 – Visualização simplificada do modelo DiCloud proposto neste trabalho. Através do DiCloud outros hospitais e médicos podem acessar imagem médicas remotamente.



Para facilitar a apresentação e compreensão do modelo este capítulo está dividido da seguinte forma: A Seção 4.2 apresenta a arquitetura do modelo, em seguida a Seção 4.1 apresenta as decisões de projeto. A Seção 4.3 com os atores e seus casos de uso e por fim metodologia de avaliação na Seção 5.

4.1 Decisões de Projeto

Algumas decisões tomadas são intrínsecas a proposta do modelo, e foram apresentadas juntamente à descrição de cada componente, como por exemplo, o modelo ARIMA, o tipo de elasticidade, a métrica CPU adotada, os *thresholds* mínimos e máximos, dentre outras decisões. Por outro lado, existem decisões que são a nível de implementação, e que podem ser alteradas sem alterar o sentido ou intenção do modelo. Tais decisões são descritas nesta seção, juntamente com suas justificativas.

A proposta deste trabalho demanda o desenvolvimento de alguns componentes, como o DiCloud Client, Elastiticy Manager, Stream API e também os compressores e descompressores. Para o desenvolvimento destes componentes foi escolhida a linguagem de programação C# com framework .NET Core. Esta escolha se dá por serem tecnologias gratuitas, *Cross-platform* e

Open source, bem como ter uma grande comunidade ativa de desenvolvedores e usuário, além de uma extensiva documentação. Esta combinação forma uma plataforma unificada onde se é possível desenvolver aplicação para diversos seguimentos, como: Web, dispositivos móveis, desktop, microserviços, jogos, aplicação com Machine Learning, aplicação que rodam em *cloud computing*, Internet das Coisas, dentre outros.

Seguindo nesta mesma linha, a ferramenta de compressão adotada foi a 7Zip [1], também *Open source*, *Cross-platform* e capaz de fornecer altas taxas de compressão. Este trabalho não aborda a escolha automática ou dinâmica do algoritmo de compressão. A partir dos resultados dos testes, será definido estaticamente o uso do método que oferece melhor taxa de compressão.

Neste trabalho, defini-se que não haverá modificações no protocolo DICOM, nem na estrutura dos arquivos de imagens médicas, visando manter a compatibilidade original, avitando qualquer falha no processamento ou interpretação do especialista. Também não será dado suporte à operação C-GET, em prol do suporte ao C-MOVE que demonstra-se mais eficiente e é a opção padrão em visualizadores de imagens DICOM modernos.

4.2 Arquitetura

O modelo DiCloud concentra-se na integração de dados originados em diversas fontes relacionadas a saúde, operando no nível PaaS. É um sistema capaz de disponibilizar o compartilhamento de arquivos de imagens e metadados (informações do paciente), hospedado em uma arquitetura em nuvem. A Figura 8 apresenta visualmente a composição do modelo, onde os itens destacados com o sinal vermelho fazem parte da proposta do modelo.

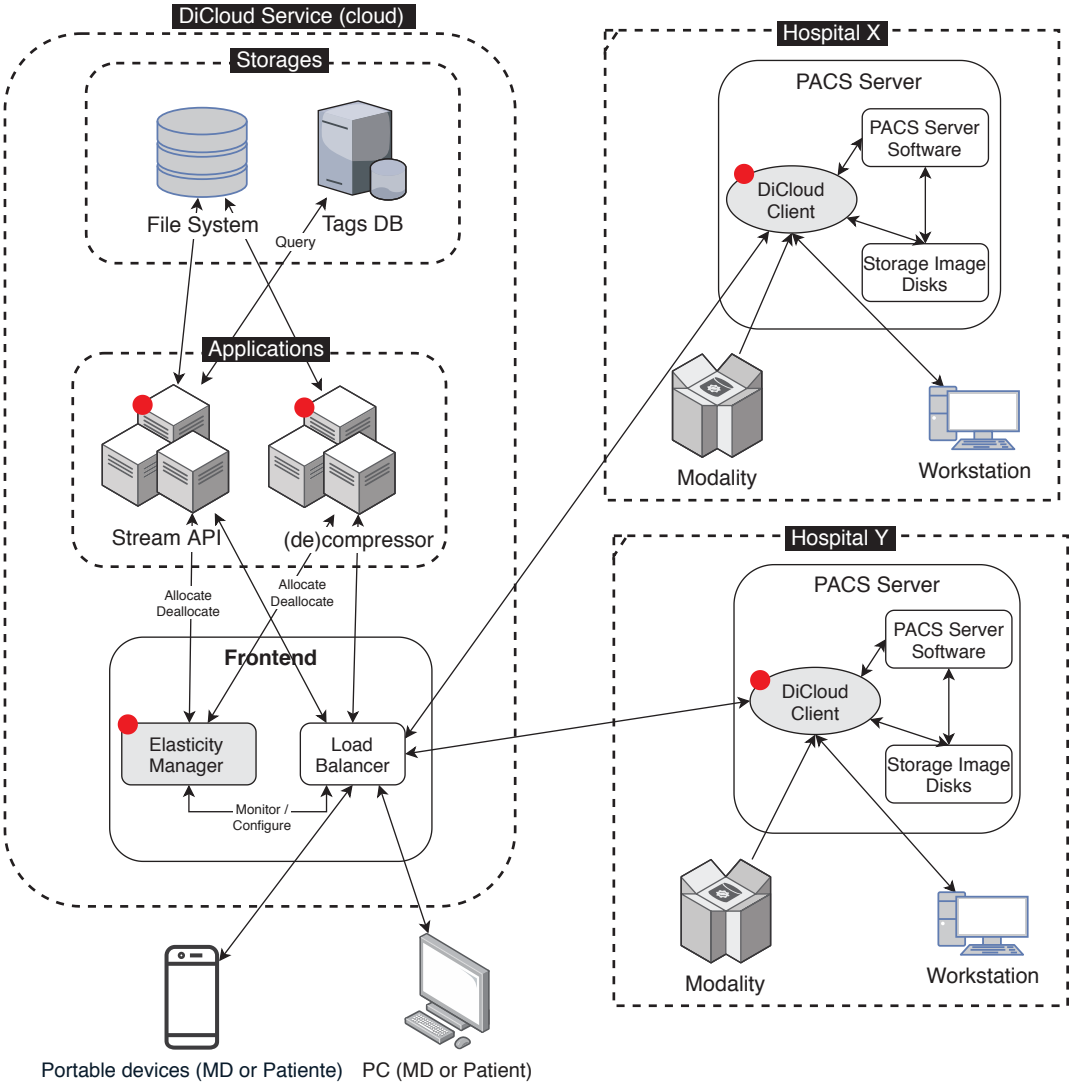
O modelo segue a arquitetura Orientada a Serviços (SOA), onde o DiCloud Client atua como cliente do Serviço DiCloud. O cliente deve ser instalado dentro de cada unidade de cada hospital que deseja obter os benefícios deste modelo. Por sua vez, o Serviço DiCloud se localiza em uma nuvem computacional. É necessário ter dentro do hospital o componente cliente para que seja possível realizar a compactação das imagens antes do seu envio, bem como realizar a descompactação quando receber novos arquivos.

Na outra extremidade, do lado do servidor temos também a presença de um compressor e descompressor para realizar a operação inversa realizada pelo cliente. A comunicação entre os hospitais acontece por meio do Serviço DiCloud e não diretamente entre os hospitais, possibilitando alguns benefícios:

- controle de acesso;
- compartilhamento entre vários hospitais, sem a restrição de pertencerem a mesma organização;
- viabilização do acesso do especialista e paciente fora do ambiente hospitalar;

[1] $https://www.7-zip.org/$

Figura 8 – Modelo DiCloud proposto neste trabalho. Dentro do ambiente hospitalar se tem apenas a adição do componente DiCloud Client. O escopo do Serviço DiCloud é uma composição de diversas tecnologias existentes, modificadas ou autorais que compõe a solução apresentadas neste trabalho. Os **itens assinalados com um circulo vermelho fazem parte da proposta deste trabalho**, e os demais itens ou já existem no ecossistema atual ou foram simplesmente aplicados neste contexto sem modificações.



Fonte: Elaborado pelo autor.

- *caching* da imagem já consultada, evitando novo gasto do link de internet do hospital, pois a imagem já estará no Serviço DiCloud, e pode ser diretamente provida ao solicitante.

4.2.1 MODALITY

Modalidade são os equipamentos que realizam a captura de imagem médicas, que desde a introdução da computação na medicina são armazenadas em formato digital. O especialista/operador realiza a captura das imagens e as envia para PACS, de modo a estarem acessíveis ao especialista através de qualquer estação de trabalho dentro da rede conectado ao PACS.

4.2.2 WORKSTATION

A estação de trabalho interna no hospital é utilizada pelo especialista (um médico por exemplo) para acessar e analisar as imagens médicas, inclusive, valendo-se do ferramental oferecido pelo software de visualização para realizar anotações sobre deu laudo. Em geral, a estação de trabalho conecta-se apenas no PACS que estão na sua rede local, limitando o acesso somente a imagens que foram capturadas dentro do próprio hospital, e em alguns casos dentro do próprio setor específico, apenas. Neste modelo, essa limitação será contornada de forma transparente, pois a estação de trabalho continuará acessando os dados do seu PACS local, no entanto, o PACS conseguirá obter as imagens que estão em outras unidades hospitalares.

4.2.3 PC ou Dispositivo Móvel

O computador pessoal representa um ponto de acesso tanto para o paciente quanto para o médico/especialista em um meio externo ao hospital/clínica. Atualmente, os PACS implantados dentro dos hospitais têm seu acesso viabilizado dentro dos domínios do hospital/clínica. No modelo DiCloud há o armazenamento em nuvem, podendo-se então viabilizar o acesso de qualquer lugar, apenas considerando-se questões de segurança e privacidade, como as abordadas no trabalho (Galletta et al., 2017; ARYANTO; OUDKERK; OOIJEN, 2015; PAN et al., 2012).

Alguns trabalhos recentemente publicados abordam questões de anonimização (desidentificação) dos metadados presentes no cabeçalho dos arquivos DICOM, porém, vemos que essas abordagens visam o compartilhamento dos arquivos de forma geral, objetivando a disponibilização para a comunidade acadêmica e cientifica realizar estudos. Como o foco do DiCloud é o compartilhamento dos arquivos às outras clínicas, hospitais e especialistas sob a autorização do paciente, cujo qual pode revogar o acesso a qualquer momento, entendemos que a anonimização se torna desnecessária.

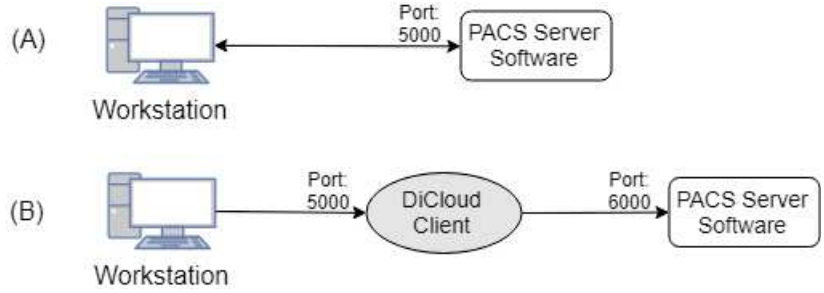
4.2.4 PACS SERVER

Um servidor PACS é um computador para armazenamento, recuperação, gerenciamento, distribuição e apresentação de imagens médicas de curto e longo prazo. Modalidades podem armazenar as imagens médicas no servidor PACS, bem como, outros computadores, equipados com um software cliente, podem armazenar e recuperar imagens diretamente do servidor PACS. Dentre as operações que são servidos pelo PACS estão C-Move, C-Store, C-Find, C-Get, dentro outros.

4.2.5 DICLOUD CLIENT

Este componente é parte fundamental da proposta deste trabalho, cujo qual, tem mais de uma responsabilidade dentro do modelo. Primeiramente, ele deve receber as conexões dos softwares DICOM clientes, como por exemplo, os instalados nos *workstations*, e até mesmo as modalidades. O DiCloud Client deve transparecer ser um PACS, mantendo a compatibilidade com os softwares clientes DICOM. Porém, seu objetivo deve ser o recebimento e envio das imagens, tanto para os clientes quanto para o Serviço DiCloud. As demais operações que lhe forem solicitadas devem ser repassadas diretamente ao o software PACS oficial, atuando então como um *proxy*.

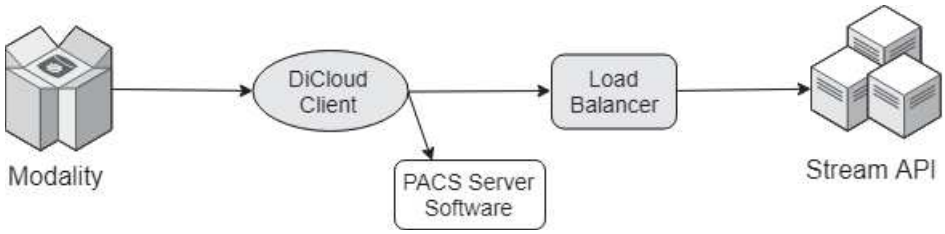
Figura 9 – Representação do cenário atual (A) e o cenário com a implantação do DiCloud (B).



Fonte: Elaborado pelo autor.

Afim de não ser necessário reconfiguração dos softwares clientes instalados nas estações de trabalho (*workstations*) é alterar a porta de comunicação do *software* PACS Server, onde o DiCloud Client começará a utilizar esta porta. A Figura 9 demonstra a troca das portas, onde no (A) temos a configuração aplicada tradicionalmente, e no item (B) temos a nova porta do *software* PACS Server será utilizada somente para comunicação com o DiCloud Client.

Figura 10 – Fluxo de envio de uma imagem para o Serviço DiCloud



Fonte: Elaborado pelo autor.

Por segundo, o DiCloud Client é responsável por fazer *upload* das imagens DICOM para o Serviço DiCloud. Uma etapa que deve ser realizada antes do envio da imagem é a compressão deste arquivo, afim de reduzir seu tamanho, poupando banda e reduzindo o tempo de envio. Este componente pode implementar diferentes algoritmos de compressão e para decidir qual o

que melhor se encaixa no contexto de imagens médicas, iremos realizar testes de compressão e descompressão de grupo de imagens.

4.2.6 SERVIÇO DICLOUD

O Serviço DiCloud é um escopo em nuvem que engloba vários componentes para provedor um ambiente escalável de compartilhamento de imagens médicas.

4.2.7 STREAM API

A Web API de *streaming* é responsável por realizar a serialização o arquivo de imagem DICOM e transmiti-lo, bem como fazer a recepção de uma serialização e persisti-la em disco. Esta Web API é de fato o meio de comunicação com os sistemas externos, cabendo a si então a responsabilidade de responder as requisições feitas pelos sistemas externos, do hospitais, médicos ou pacientes. Esta Web API pode ser replicada e ter sua carga distribuída entre suas instâncias.

Esta API requer que seja informado junto aos dados da requisição que lhe é feita, um *token* de autorização. Este *token* de acesso serve para identificar o requerente, usando este dado para verificar se ele tem permissão de acesso ao dado solicitado. O gerenciamento desses *tokens* será feito através de sistema exposto através desta mesma Web API.

As autorizações de acesso aos arquivos DICOM podem ser delegados tanto pelo paciente quanto pelo hospital realizador do exame, cedendo acesso para outra entidade ou especialista em especifico. Quando uma entidade cede acesso à outra, isso significa que o DiCloud Client instalado em outro hospital ou clínica pode fazer o download dos arquivos para dentro do seu PACS Server local, deixando-o acessível para as estações de trabalho do outro hospital.

A Stream API é responsável por salvar no TAGS DB as informações de *tags* das imagens DICOM, bem como os cabeçalhos HTTP relativos a imagens que está recebendo na hora. Da mesma forma, é responsável por atualizar no registro a data do último acesso em caso de download da imagem, para que mecanismos de limpeza de arquivos antigos possam usar esta data como parâmetro de decisão de quais arquivos podem ser removidos ou movidos para uma mídia de armazenamento de custo inferior.

4.2.8 COMPRESSOR / DECOMPRESSOR

Este componente é crítico a performance do modelo, uma vez que sua atuação gera consumo intensivo de CPU. Uma vez que a escolha do algoritmo de compressão tenha sido feita, a forma de melhorar a performance da operação do sistema excetuando-se a modificação do algoritmo, é a ampliação do recurso de *hardware* disponível para ser consumido. Sendo assim, esta parte do sistema é o foco principal do mecanismo de escalabilidade. O algoritmo adotado neste componente tem que ser obrigatoriamente o mesmo que foi escolhido para se utilizar no DiCloud

Client. Existem pelo menos duas abordagens macro em relação a como aplicar a compressão.

Pode-se aplicar a compressão somente na parte da imagem (também conhecida como *pixel-data*), ou pode-se aplicar a compressão em todo o arquivo, ou seja, incluindo tanto o *pixel-data* quanto os meta dados que são embutidos nos arquivos DICOM. Como requisito temos que a compressão aplicação seja sem perdas, uma vez que a aplicação de compressão com perdas nas imagens ainda é uma discussão em aberto na comunidade médica e de radiologia, e ainda não se tem constatações definitivas sobre seus efeitos no diagnóstico.

Neste trabalho adota-se a compressão do arquivo como um todo (*as-is*), para que não haja modificação no conteúdo das imagens médicas. Inclusive, nos casos em que o *pixel-data* já tenha passado por uma compressão sem perdas, ainda é possível comprimir o *pixel-data*, seu dicionário de compressão do *pixel-data* e os meta dados, obtendo ganhos extras.

4.2.9 TAGS DB

O padrão DICOM contém diversas possibilidades de *tags* que podem estar contidos em um arquivo DICOM. Essas *tags* servem para armazenar informações de diversos tipos, seja sobre o paciente, equipamento ou sobre o local onde está sendo feito a captura da imagem. Este banco de dados armazena estas *tags* afim de indexar as informações principais, agilizando consultas feitas pelos softwares clientes.

Quando o DiCloud Client realiza uma consulta ao Serviço DiCloud, este banco de dados é usado para resolver a *query*. Essa abordagem é mais eficiente do que ler cada *tag* dentro de cada arquivo em disco. Quando um arquivo é enviado, a Stream API coleta as informações das *tags* que são usadas como fonte de dados para pesquisa e salvar no TAGS DB.

Listing 4.1 – Exemplo de documento/Tupla salva no banco de dados Tags DB.

```
{
   "Source" : "hospital␣1",
   "SavedDate" : ISODate(2019-12-10T22:05:19+0000),
   "LastAccessDate" : ISODate(2019-12-10T22:05:19+0000),
   "Path" : "/storage/79fea081-0f27-428d-ad17-703f49beb686",
   "Tags" : {
      "PatientName" : "Anonymous␣Patient␣1",
      "PatientID" : "1",
      "SOPInstanceUID" : "1"
      ...
    }
}
```

A cada acesso (*download*) realizado em determinada imagem o campo "LastAccessDate"é atualizado, de modo que seja possível implementar um sistema onde arquivos mais velhos sejam apagados, liberando espaço na nuvem, caso seja conveniente ao administrador. Uma vez com o arquivo apagado fisicamente do disco, é possível recuperá-lo consultando o DiCloud Client no

hospital referenciado pelo campo "Source".

4.2.10 FILESYSTEM

Representa o sistema de arquivos utilizado nos discos onde são armazenados os arquivos DICOM. No ambiente do PACS, geralmente os sistemas de arquivos são FAT. O sistema de arquivos é compatível com a plataforma PC e, portanto, requer que a mídia seja organizada em setores. O sistema de arquivos do PC deve ser organizado como um sistema de arquivos não particionados, usando a tabela de alocação de arquivos (FAT) de 12 bits ou 16 bits. O FAT e estruturas de arquivo relacionadas são compatíveis com o DOS 4.0 e sistemas de arquivos posteriores.

4.2.11 LOAD BALANCER

Um balanceador de carga é um dispositivo que distribui o tráfego de rede ou carga de trabalho em um *cluster* de servidores e/ou aplicações. O balanceamento de carga melhora a capacidade de resposta e aumenta a disponibilidade de aplicativos. Comumente, o balanceador consiste de um software instalado em uma máquina física ou virtual (mais recentemente, conteinerizado) que recebe requisições de clientes, aplica um algoritmo de balanceamento de carga para decidir para qual nó deve ser repassada a requisição para que seja processada, e após o resultado retornado ao cliente.

4.2.12 ELASTICITY MANAGER

Tradicionalmente, a elasticidade é aplicada de modo reativo, deixando a aplicação entrar em estado de sobrecarga até tomar-se a ação de aumentar a quantidade de instâncias provendo o serviço. Este componente gerenciador de elasticidade tem por objetivo aplicar o modo de elasticidade proativa de forma automática. Por automática, entende-se que não é requerido intervenções manuais reativas e nem fazer modificações na aplicação para implementar elasticidade.

O componente Elasticity Manager deverá gerenciar e tomar as decisões sem intervenção, de forma autônoma e antecipada aos eventos de congestionamento do sistema. Foi escolhida a elasticidade horizontal, pois, com elasticidade vertical, a quantidade de recursos é limitada aos recursos disponíveis em um nó virtualizador. Para alcançar a proatividade, tem-se que prever o comportamento da carga do sistema ao longo do tempo. Este componente implementa um algoritmo desenvolvido utilizando séries temporais, e busca tomar as decisões de elasticidade de recursos de forma automática, visando manter a qualidade de serviço das aplicações que estão rodando no escopo do Serviço DiCloud.

Séries temporais baseiam-se na análise de uma métrica ao longo do tempo. Para isso, é ne-

cessário analisar periodicamente o valor dessa métrica, mantendo todos os valores do histórico para uma análise completa. Tendo um histórico de valores ordenados pelo momento que foram obtidos é possível entender o comportamento de uma métrica. Sabendo seu comportamento, podemos estimar possíveis valores futuros para ela. Então, séries temporais se tornam uma análise poderosa de padrões de comportamentos ao longo do tempo.

Séries temporais não é a única forma de prever valores de uma métrica ao longo do tempo. Outra forma de fazer isso é utilizando algoritmos de aprendizado de máquina. Esses algoritmos utilizam o histórico da métrica para tentar aprender como ela se comporta e com base nisso prever valores futuros. Redes neurais é uma forma de implementar aprendizado de máquina, onde o algoritmo utiliza componentes chamados de neurônios para aprender com a variável observada.

Em comparação a séries temporais, os algoritmos de aprendizado de máquina produzem resultados melhores, porém exigem um período mais longo de aprendizado. Ou seja, enquanto séries temporais exigem poucos valores iniciais para permitir prever novos valores, aprendizado de máquina necessita de mais valores para que consiga prever os valores corretamente. Tendo em vista essa demora em conseguir começar a estimar, aprendizado de máquina não se torna uma boa opção para avaliarmos métricas e tomarmos decisões de elasticidade rapidamente. Por isso, mesmo não prevendo valores ótimos, nossa escolha foi aplicar séries temporais.

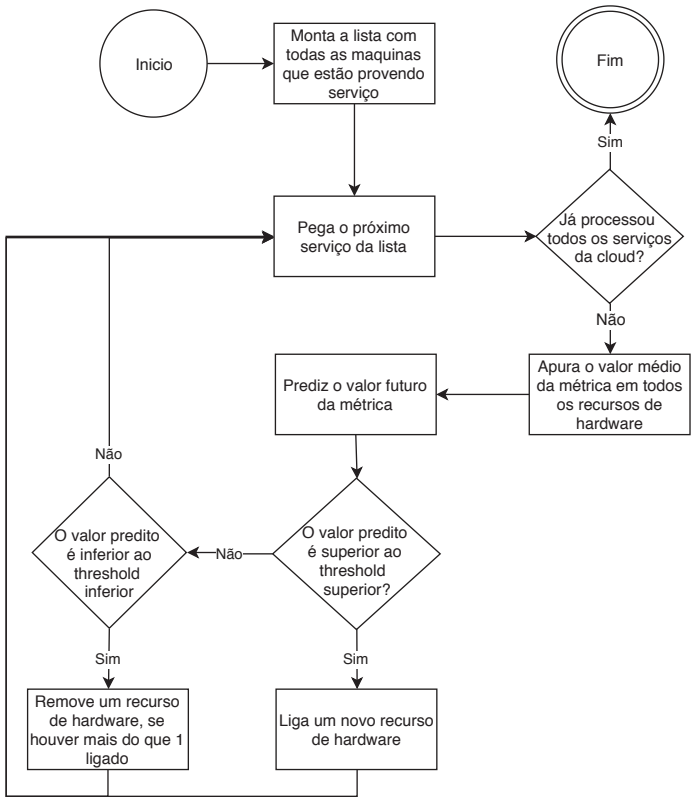
Como explicado acima, as séries temporais precisam analisar uma métrica ao longo do tempo. Então, teremos que escolher uma métrica para avaliar e ela precisará ser monitorada em intervalos de tempos fixos. A métrica precisa ser algo relevante para a operação da aplicação que está sendo executada, e depende do intuito de cada aplicação, onde as mais comuns são memória, CPU, espaço livre em disco, IOPS e trafego de rede. No caso do modelo DiCloud a métrica mais impactante é a CPU, pois no caso especifico deste modelo ela indica o quão ocupado um recurso ou servidor está.

Isso é necessário para avaliar se o serviço está conseguindo processar as requisições satisfatoriamente, mantendo um nível de serviço aceitável. Para analisar a CPU em termos de elasticidade de recurso, iremos coletar o valor atual dessa métrica em intervalos de tempo fixos. Em trabalhos relacionados, utilizando o *middleware OpenNebula*, avaliou-se a CPU em intervalos fixos de 15 segundos entre as medidas para avaliar a carga de processamento em máquinas virtuais (DA ROSA RIGHI et al., 2019). Tomaremos o mesmo intervalo, porém poderemos mudar de acordo com as necessidades encontradas.

A Figura 11 demonstra o fluxo de decisão para a elasticidade de recursos. O Elasticity Manager mantém uma lista com todas as instâncias que estão rodando, podendo assim, avaliar cada um deles, fazendo a média da métrica CPU de cada instância que esteja rodando. Com esse valor médio, irá montar a série temporal, utilizando os valores anteriores, para prever o valor futuro da métrica. Se o valor predito, for superior a um limiar (*threshold*) previamente definido, indicará que em um futuro próximo aquele serviço atingirá um estado não desejável. Assim, ele irá alocar uma nova instância. Se o valor que foi predito for menor que o inferior,

irá remover uma instância. O sistema irá escolher a instância com menos CPU utilizada, sendo que ele se torna mais candidato a ser desligado.

Figura 11 – Fluxo de decisão de elasticidade de recursos. Para cada serviço, irá avaliar a média da métrica utilizando valores medidos em todos os *containers*. Com essa média, monta uma série temporal e avalia o valor futuro dessa métrica. Se ele for superior à um valor pré-definido, tentará adicionar um novo *container*, mesmo que seja necessário ligar uma nova máquina. Se for inferior, irá remover um container.



Fonte: Elaborado pelo autor.

Na Seção 2 apresentamos alguns algoritmos de predição de séries temporais, onde um deles foi o ARIMA, que pode ser utilizado para prever valores de métricas que variam durante o tempo. Neste modelo adotaremos o ARIMA como modelo de séries temporais para avaliar as métricas da aplicação. Esse modelo foi escolhido, pois apresentou resultados satisfatórios quando utilizados por outros autores, tais como (DA ROSA RIGHI et al., 2019; ROSA RIGHI et al., 2016).

O ARIMA possui algumas parametrizações para configurar seu comportamento, e para decidir a melhor configuração de predição a utilizar, realizamos testes simulando cargas de trabalho. Esses testes são apresentados na seção 6.2. Além disso, foi definido um parâmetro de *lookahead* para o sistema, baseando-se na estratégia elaborada em (DA ROSA RIGHI et al., 2018) que ajudará a parametrizar o funcionando do Elasticity Manager.

$$lookahead = \frac{Abs(Max(scaling\_out\_time))}{monitoring\_observation\_period} \tag{4.1}$$

A Equação 4.1 descreve a quantidade ideal de ciclos para que uma decisão seja prevista. *scaling_out_time* descreve os tempos para inicializar servidores já observados até o momento, onde a função Max() pegará o maior valor, *monitoring_observation_period* descreve o ciclo entre observações da métrica, e *lookahead* indica quantas instruções a frente deverá ser observado. Esse parâmetro, definirá o quanto no futuro iremos estimar a variável da série temporal.

Um valor muito alto irá tentar prever um futuro muito distante e tende a ter uma taxa de erro maior. Um valor muito baixo não será efetivo o suficiente para deixar a máquina pronta para uso, quando necessária. Logo, saber o valor correto para esse parâmetro é essencial para o bom funcionamento do sistema. Definiremos o valor desse parâmetro como o tempo em que as máquinas levam em média para a partir de um estado desligadas, estarem disponíveis e prontas para efetuarem processamentos.

A utilização de *thresholds* fixos é a abordagem mais utilizada na elasticidade reativa, onde quando a métrica atinge um valor determinado, o gerenciador executa uma ação. No Elasticity Manager, a decisão será muito semelhante à abordagem reativa, porém se utilizará para a decisão um valor futuro de uma métrica, antevendo a necessidade de recursos. Serão fixados os limiares mínimo e máximo.

Com base nos artigos de (DA ROSA RIGHI et al., 2019; GALANTE et al., 2016; ROSA RIGHI et al., 2016, 2015; NETTO et al., 2014) estipulamos como *theshold* máximo e mínimo os valores de 80% e 20% respectivamente, removendo a necessidade do usuário configurar esses valores. Esses valores são adequados para uma aplicação proativa, tendo em vista que possuem uma boa faixa de valores válidos, sendo que pequenas flutuações nos valores previstos não tomarão decisões equivocadas.

4.3 Atores e Casos de Uso

O modelo opera de forma abrangente onde interage com diferentes tipos de atores, que tem diferentes funcionalidades e objetivos. Temos como atores o **paciente**, o **especialista**, o **hospital** e **clínicas** de radiologia. O especialista é qualquer pessoal que deseja obter acesso a imagem para realizar um diagnóstico clínico, estando este dentro ou fora do hospital. Tanto o hospital quanto as clínicas são domínios de atuação, ou seja, agrupamento de profissionais e detentores de um sistema PACS. No texto deste trabalho tanto uma clínica quanto um hospital têm os mesmos propósitos e funcionalidades, então para simplificação trataremos os dois simplesmente por hospital. Alguns dos cenários de uso mais evidentes são apresentados a seguir.

Especialista usando um workstation - o especialista dentro do hospital não terá seu fluxo de trabalho alterado, sendo transparente o acesso ao sistema. Ou seja, a estação de trabalho estará acessando o DiCloud Client, porém não é necessário que o especialista realize nenhuma configuração. Ao realizar uma consulta de imagem através do seu sistema na estação de trabalho, o DiCloud Client receberá essa consulta, e caso a imagem não esteja presente no PACS local do hospital, fará essa consulta no Serviço DiCloud usando as credenciais do hospital.

Especialista usando seu computador particular ou externo ao ambiente hospitalar - normalmente, os hospitais restringem o acesso ao seu servidor PACS à acessos realizados dentro da rede interna do hospital (intranet), não sendo possível acessá-los estando fora da rede, como no caso de uma rede móvel ou uma rede residencial. O modelo DiCloud apresenta uma viabilização do acesso aos arquivos de imagens médicas, mas não relaciona-se com a sua exibição. Portanto, o especialista deve ter instalado um software visualizador de imagens DICOM, semelhante ou igual ao que existe no ambiente hospitalar. Neste caso, é necessário configurar este software para acessar o Serviço DiCloud. Esta configuração é simples, bastando o especialista entrar na seção de configuração de rede do visualizador de imagens e cadastrar como fonte o IP ou domínio (DNS) do Serviço DiCloud. Além disso, o usuário do especialista deve ter recebido permissão para acessar os arquivos de determinado paciente. Essa permissão é outorgada pelo paciente ou entidade hospitalar que tem posse dos arquivos, através de um sub-sistema gerenciador de autorização presente no modelo.

Especialista usando um dispositivo móvel - igualmente ao caso de acesso do especialista em um computador fora do ambiente particular, deve-se ter instalado no dispositivo um aplicativo capaz de abrir o arquivo DICOM. Este aplicativo deve permitir que seja configurado o servidor de acesso as imagens, bem como o usuário do especialista deve ter recebido permissão de acesso as imagens.

Paciente usando um dispositivo móvel ou um computador particular - o paciente tem acesso ao sistema para três casos: (a) acessar seus arquivos de imagem médicas; (b) para conceder permissão de acesso a um hospital ou um especialista especificamente; (c) para revogar permissão de acesso a um hospital ou um especialista especificamente. O Acesso ao sistema é realizado pelo *website* da DiCloud que é responsivo e ajustável a dispositivos móveis. É importante se observar que quando o acesso é concedido a um hospital, todos os especialistas do hospital que usam uma estação de trabalho, terão acesso a imagem permitida, pois a autorização não acontecerá com credenciais do especialista, mas com a credencial do hospital impersonada diretamente pelo DiCloud Client ao consultar o Serviço DiCloud.

5 METODOLOGIA DE AVALIAÇÃO

Como apresentado nas seções anteriores, o DiCloud é baseado em 2 componentes macro: o DiCloud Client e Serviço DiCloud. Os dois componentes se complementam, de forma que em conjunto podem estabelecer uma comunicação que visa compartilhar imagens médicas, mantendo uma qualidade de serviço por meio da correta alocação de recursos conforme a necessidade.

Por isso, o objetivo da avaliação é verificar o pleno funcionamento do fluxo de comunicação, da modalidade até a estação de trabalho de outro hospital. Também, avaliar se a abordagem proativa terá benefícios sobre a abordagem reativa no que concerne a qualidade do serviço e alocação adequada de recursos (considerando a taxa de erro e tempo de resposta das aplicações). As implementações seguiram os padrões e tecnologias detalhados no item 4.1, e terão seus detalhamentos específicos explanados nos itens seguintes.

5.1 Protótipo Elasticity Manager

O Componente Elasticity Manager foi desenvolvido em C#. Uma de suas sub-rotinas é a predição usando ARIMA. Existem mais duas sub-rotinas que são executadas por este componente, que são dependentes da plataforma de virtualização adotada. A integração com esta plataforma foi desenvolvida também em C# através de uma biblioteca[1] desenvolvida pela própria Microsoft, que abstrai a comunicação com a WebAPI da plataforma através de classes no namespace Microsoft.Azure.Management.Fluent. Esta biblioteca pode ser encontrada e baixada através do gerenciador de pacotes Nuget [2] ou via seu *website* [3].

O mecanismo de balanceador de carga será o HAProxy, sendo que ele implementa Round-Robin para distribuir as requisições. Para essa parte da aplicação, não importa em qual nó será executado. A aplicação que fica do lado do servidor, recebe uma requisição, processa ela e devolve um resultado ao cliente. Para a implementação serão utilizados os seguintes parâmetros:

- *scaling_out_time*: será medido o tempo médio que leva para as máquinas serem instanciadas. De acordo com esse tempo medido, iremos definir esse parâmetro;
- *tempo_observacao*: 15 segundos;
- *threshold do servidor superior*: 80%;
- *threshold do servidor inferior*: 20%.

[1] $https://docs.microsoft.com/en-us/dotnet/azure$
[2] $https://www.nuget.org/downloads$
[3] $https://www.nuget.org/packages/Microsoft.Azure.Management.Compute.Fluent/$

5.1.1 Infraestrutura de Testes

Neste estudo, adotamos a plataforma Microsoft Azure [4] como ambiente de virtualização. Foram provisionadas três máquinas para hospedar a Strem API (dicloud-trl-1, dicloud-trl-2 e dicloud-trl-3), sendo elas configuradas dentro da mesma sub-rede, com o sistema operacional Ubuntu 16.04 e de tamanho "Standard A2 v2", isto é, com as seguintes configurações de hardware:

- 2 vCPU (Intel(R) Xeon(R) Platinum 8171M CPU @ 2.60GHz);
- 4 GB de memória RAM;
- 30GB de espaço em disco HDD;
- Limite de 2000 IOPS;

O arquivo de configuração do HAProxy precisa ser alterado para incluir o endereço de IP, porta dos servidores da aplicação de Stream API e o algorítimo de balanceamento que será utilizado. Geralmente o arquivo de configuração localiza-se em /etc/haproxy/haproxy.conf. No seguinte exemplo, temos três servidores que dividirão a carga entre si, quando ativos.

Listing 5.1 – Bloco de configuração do HAProxy responsável por indicar quais servidores estão oferecendo o serviço Stream API. Para simplificar as maquinas dicloud-trl-1, dicloud-trl-2 e dicloud-trl-3 foram chamadas de web01, web02 e web03, respectivamente.

```
backend nodes
        mode http
        balance roundrobin
        server web01 172.16.2.1:8080 check fall 1 rise 2
        server web02 172.16.2.2:8080 check fall 1 rise 2
        server web03 172.16.2.3:8080 check fall 1 rise 2
```

Para coletar as informações sobre utilização de recursos foram utilizados três ferramentas de terceiros: *Prometheus* [5], *Node Exporter* [6] e *HAProxy Exporter* [7].

O *Prometheus* é uma ferramenta que consolida as informações de diversas fontes de dados em uma API única. Assim, ele foi configurado para coletar as métricas exportadas de todas as máquinas virtuais envolvidas no Serviço DiCloud e armazenar em seu próprio banco de dados, que é do time *Time-Series*. Dessa forma, o Elasticity Manager efetua requisições HTTP apenas ao *Prometheus* para saber o estado da saúde de todas as máquinas virtuais da *cloud*.

[4] $https://azure.microsoft.com/pt-br/$
[5] $https://hub.docker.com/r/prom/prometheus/$
[6] $https://github.com/prometheus/node_exporter$
[7] $https://github.com/prometheus/haproxy_exporter$

O *Node Exporter* foi configurado para executar uma réplica em cada máquina virtual. Ele basicamente coleta estatística de uso do sistema, como CPU e Load Average. O *Node Exporter* é uma ferramenta capaz de coletar uma séria de métricas do sistema, cujas quais não cabem dentro da análise deste presente trabalho, e como cada coleta de métrica demanda um tempo computacional extra, efetuou-se a desabilitação das demais métricas que aqui não são envolvidas. Esta configuração é feita via parâmetro na linha de comando utilizada para dar start na aplicação. É necessário utilizar a sequencia de *flags* –collector.* para habilitar e –no-collector.* para desabilitar o coletor de métrica:

Listing 5.2 – Comando utilizado para iniciar a aplicação Node Exporter coletando somente as métricas necessárias para este trabalho.

```
/usr/local/bin/node_exporter \
            --collector.cpu \
            --collector.loadavg \
            --collector.meminfo \
            --no-collector.netstat \
            --no-collector.stat \
            --no-collector.vmstat \
             .....
```

O *HAProxy Exporter* foi configurado para rodar dentro da própria máquina do Prometheus, pois sua coleta de informações é feita via protocolo HTTP, direcionado à API de estado do software HAProxy, não necessitando que ele seja instalado dentro do mesmo sistema onde está o HAProxy. Para iniciar esta aplicação deve-se informar via parâmetro a URL da API de estado:

Listing 5.3 – Comando utilizado para iniciar a aplicação HAProxy Exporter.

```
docker run -p 9101:9101 prom/haproxy-exporter -haproxy.scrape-
    uri="http://dicloud-frontend.eastus.cloudapp.azure.com:8404/
    monitor?stats;csv"
```

Para avaliar se essas integrações estão funcionando corretamente utilizamos a interface web disponibilizada pelo *Prometheus*, que dispõe de uma seção de consulta à métricas que estão sendo coletadas. A métrica **node_load1**, é responsável por retornar o *Load Average* médio do último minuto, e sua consulta comprova a correta coleta dos dados de todas à máquinas quando comparado ao dado bruto visualizado através da ferramenta **top** [8], presente no próprio sistema operacional Linux, acessada através de uma conexão SSH.

O próximo passo é testar se a integração que cria e remove as máquinas virtuais está funcionando corretamente. Os testes feitos na Seção 6.2 indicam os parâmetros que devem ser

[8] $http://man7.org/linux/man-pages/man1/top.1.html$

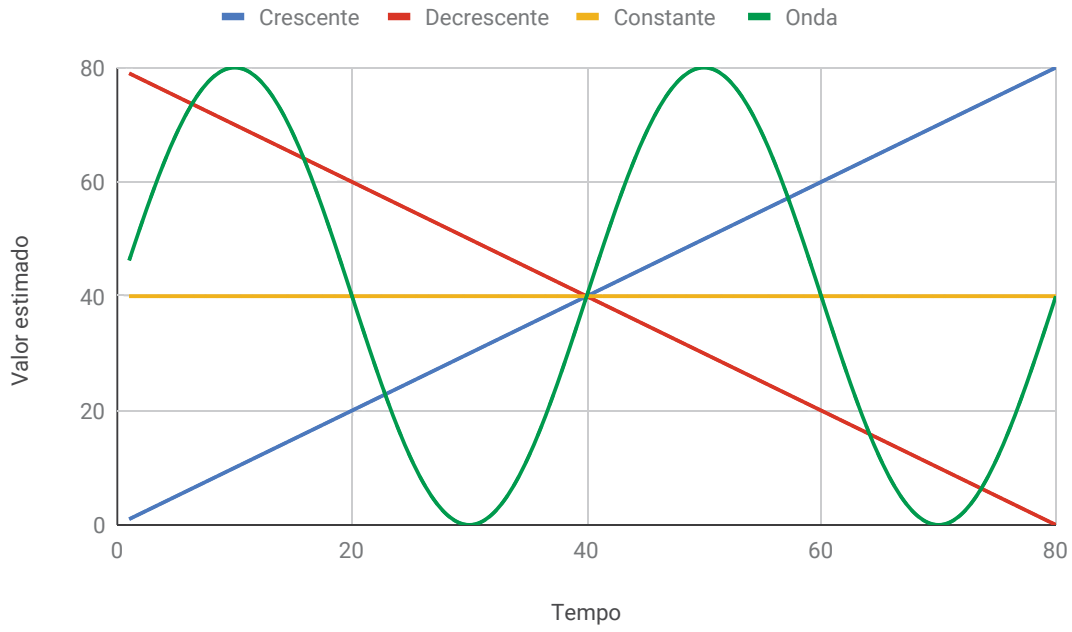
aplicados no sistema, deixando-o calibrado para sua análise. Com base no resultado da predição do motor, o avaliador das métricas utilizará *thresholds* inferior e superior para avaliar se deve adicionar/remover novos recursos computacionais. Por exemplo, se o motor indicar que *threshold* superior será atingido, o avaliador indicará que será necessário adicionar novos recursos. Essa decisão é repassada ao módulo de acesso à nuvem que implementará a API de integração com a nuvem.

5.1.2 Parâmetros do ARIMA

Se este componente não for eficiente, o gerenciamento da elasticidade não apresentará ganhos para a aplicação. Para ser definido a melhor forma de predizer valores futuros, é necessário avaliar quais as cargas de trabalho que o Serviço DiCloud poderá ser submetido. Assim, o primeiro passo foi definir as cargas de trabalhos que serão avaliadas pelo modelo de predição. Adotou-se as quatro categorias de carga elaboradas em (ROSA RIGHI et al., 2016), sendo elas: crescente, decrescente, constante e onda.

A série temporal ARIMA foi apresentada na Seção 2.8. Um dos elementos do DiCloud é seu algoritmo de predição da elasticidade proativa de recursos, sendo que se ele não for eficiente, o gerenciamento da elasticidade não apresentará ganhos para a aplicação. Para avaliar sua eficácia, o primeiro passo foi definir cargas de trabalho para submeter ao ARIMA. Essas cargas de trabalho são as mesmas apresentadas na Figura 12.

Figura 12 – Cargas de trabalho simuladas. São basicamente quatro cargas: crescente, decrescente, constante e onda. Cada carga simula um possível caso real de utilização.



Fonte: Elaborado pelo autor.

Essas cargas de trabalho tentam simular casos reais de uma aplicação comum. Por exemplo, a carga crescente simula um acréscimo nas requisições de um microsserviço, gerando assim um aumento gradativo de CPU. Para a simulação, foi definido 0 como valor mínimo e 600 como valor máximo. Na carga de trabalho crescente, os valores aumentam de 0 até 600 gradativamente. Já na carga de trabalho decrescente, os valores decrescem de 600 até 0 gradativamente. A carga constante, mantém o valor de 300 do início ao fim da simulação. Por fim, a carga de onda, executa a seguinte equação:

$$y = (\sin((2 * \pi)/60 * x) * 300) + 300$$

Essa fórmula determina o valor do eixo y de acordo com o valor da coordenada x no momento do cálculo. A divisão por 60 é feita, pois foi estimado um total de 120 instantes do eixo x. Sendo assim, gera uma onda que se repete uma vez durante um ciclo completo. O valor de 120 foi definido para simular 30 minutos de execução da aplicação, sendo que cada coleta de valor ocorreria em intervalos de 15 segundos. O valor de coleta em intervalos de 15 segundos é uma prática adotada em outros trabalhos pelos autores DA ROSA RIGHI et al. (2019) e ROSA RIGHI et al. (2016), pois o *middleware* de *cloud* atualiza os valores das métricas de tempos em tempos.

Efetuar a requisição em intervalos menores pode ocasionar repetição dos mesmos valores já processados. Na fórmula de onda, foi multiplicado por 300 para que a onda inicie no ponto médio do gráfico. Por fim, adiciona-se 300 para deslocar a onda, de modo a evitar valores negativos. Com o resultado da aplicação dessas cargas de trabalho se encontrará qual é a parametrização do ARIMA que melhor se encaixa no cenário de uso da aplicação Serviço DiCloud.

5.2 Protótipo Stream API

É vital para este componente que sua disponibilidade não seja afetada por questões de desempenho. O Elasticity Manager é o responsável por gerenciar a elasticidade da nuvem contornando este problema. Afim de testar o desempenho deste componente usar-se-á um teste de carga com a ferramenta Apache JMeter [9]. Com esta ferramenta é possível simular acessos em lote neste componente, obtendo relatórios sobre o tempo de resposta e taxa de falha nas requisições. Estipulou-se o periodo de 30 minutos como sendo um intervalo suficiente para verificar o escalonamento das máquinas virtuais, como pode ser visto na projeção da Figura 13.

Portanto, o plano de teste configurado na ferramenta, visou aumentar a carga do sistema, submetendo o sistema à necessidade de instanciar mais máquinas virtuais para apoiar o processamento, em uma escala de onda, baseando-se na fórmula apresentada nos testes do ARIMA na seção 6.2, e realizando um corte do período de 30 minutos. A ferramenta JMeter estabelece automaticamente o fator de incremento das *threads* utilizadas para requisitar o sistema.

[9] https://jmeter.apache.org/

Figura 13 – Projeção automática criada pela ferramenta JMeter onde a quantidade de *threads* varia de 1 à 300 e o tempo demandado para a execução é de 30 minutos.

Fonte: Autogerado pela ferramenta JMeter.

Além da configuração do escalonamento de *threads*, também foi configurado o método utilizado para se comunicar com a Stream API. Isso se dará através do verbo POST do protocolo HTTP. Esta requisição será do tipo multipart/form-data, pois ela submeterá consigo um arquivo de imagem DICOM. Os seguintes campos são informados via form-data: PatientName, PatientID e SOPInstanceUID, além do arquivo de imagem, que tem o tamanho de 6.049.792 bytes, aproximadamente 5.76MB.

Escolheu-se um arquivo de imagem de tamanho abaixo da média para evitar que a largura de banda de internet, interferisse na capacidade máxima de *threads* em execução, bem como, um tempo maior de arquivo deixaria a máquina ociosa por mais tempo à espera de IO de rede, sendo necessário um número maior de *threads* em execução paralela para obter um aumento de carga no sistema destino. O código de retorno 200 foi utilizado como fato de decisão do sucesso da requisição, ou seja, qualquer status diferente de 200 é contato como erro.

A máquina virtual utilizada para executar a ferramente JMeter foi criada na provedora de serviços de Cloud Hosting, Umbler. A máquina tem as seguintes características de *software* e *hardware*, onde essas configurações foram super dimensionadas a fim de evitar gargalos na execução dos testes:

- 12 vCPU (Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz);
- 30 GB de memória RAM;
- 80GB de espaço em disco SSD;
- Limite de 5000 IOPS;
- Microsoft Windows Server 2016 Standard;

Foi alterado a configuração da JVM que executa o JMeter para que fosse possível reconhecer uma quantidade de memória RAM além da padrão.

Listing 5.4 – Configurações que foram aplicadas ao iniciar a JVM, a fim de reconhecer e poder utilizar até 25GB de memória RAM. Essa configuração pode ser realizada via variável de ambiente ou dentro do arquivo jmeter.bat na pasta bin da ferramenta.

```
set HEAP=-Xms1g -Xmx25600m -XX:MaxMetaspaceSize=256m
```

5.3 Protótipo DiCloud Client

Um dos primeiros passos realizados foi a criação do ambiente PACS. Para isso, foi utilizado o Conquest [10], que é uma implementação *Open Source* de um servidor PACS. Este software foi instalado em duas máquinas virtuais diferentes, afim de simular dois hospitais diferentes.

Para nos assegurar as configurações do software foram aplicadas corretamente, usou-se o visualizador de imagens DICOM disponível no próprio Conquest, bem como o visualizar de imagens médicas gratuito MicroDicom [11]. Com este ambiente, foi possível testar as operações que estão envolvidas no desenvolvimento deste componente. Por exemplo, as operações C-Find, C-Store e C-Move que foram tratados pela nossa aplicação, e as demais operações são simplesmente repassados diretamente ao software servidor DICOM (servidor PACS).

No próximo passo foi o desenvolvimento do DiCloud Client, implementado na linguagem C# conforme definido na Seção 4.1. Usamos como base a biblioteca fo-dicom [12], cuja qual tem implementações capazes de ler documentos DICOM e estabelecer comunicação com softwares servidores PACS.

Temos os seguintes pontos que foram testados em relação ao DiCloud Client:

- o correto funcionamento das estações de trabalho (*workstation*) sem a necessidade de mudar configurações. Isso está alinhado com a premissa de ser transparente ao usuário e ao administrador dos sistemas; A porta TCP que era originalmente escutada pelo software PACS (porta padrão do Conquest é 5678), foi configurada no DiCloud Client, e o Conquest configurado para escutar a porta 5679. Com isso não foi necessário alterar nenhuma configuração de nenhuma estação de trabalho (*workstation*);

- se este componente conseguia consultar corretamente o servidor PACS. Para isso cadastrou-se imagens médicas de teste no sistema e consultou-se através da estação de trabalho. A correta exibição das imagens na estação de trabalho indicou que o DiCloud Client está operando corretamente na parte de consulta local;

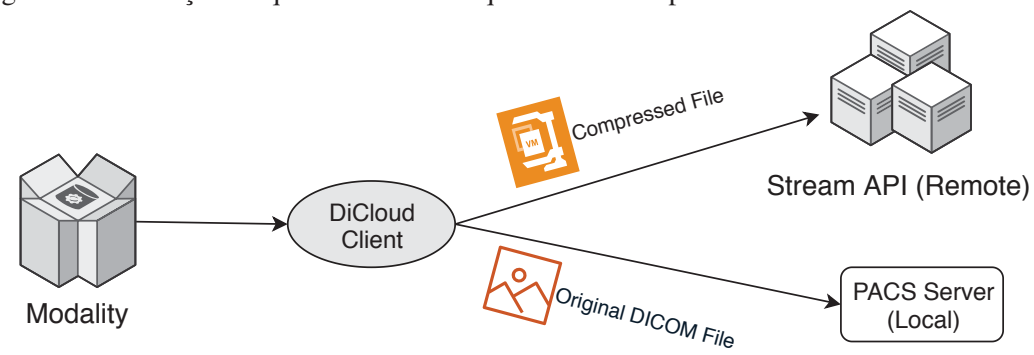
- teste de comunicação com o DiCloud Client. Para isso, cadastrou-se no Serviço DiCloud imagens de teste. Novamente usando a estação de trabalho para realizar a consulta a estas imagens, podendo aferir o funcionamento do DiCloud Client através da correta exibi-

[10] $https://ingenium.home.xs4all.nl/dicom.html$
[11] $http://www.microdicom.com/$
[12] https://github.com/fo-dicom/fo-dicom

ção das imagens. Com isso também, se estará testando o mecanismo de descompressão presente neste componente;

- sobre a compressão durante o estágio de envio das imagens, simulou-se a geração de uma nova imagem, sendo enviada com destino ao servidor PACS. Na rota até o PACS, essa imagem passa pelo DiCloud Client, que faz a compressão da imagem e faz seu envio também para o Serviço DiCloud;

Figura 14 – Ilustação de qual formato de arquivo é enviada para cada destino.



Fonte: Elaborado pelo autor.

O sucesso desse procedimento verificou-se com a constatação do correto armazenamento da imagem no PACS local, e também através de uma consulta realizada no Serviço DiCloud através de uma estação de trabalho (*workstation* com o *software* MicroDicom) do ambiente simulado de um segundo hospital. Pode-se verificar o sucesso no envio e na compressão da imagem inspecionando-se o Serviço DiCloud ou realizando uma consulta através de uma nova estação de trabalho, em um contexto de um novo hospital.

Diferentes tipos de modalidades, geram tamanhos variados de arquivos, portanto, serão utilizados diferentes cargas de trabalho durante a confecção do protótipo e realização dos testes. Temos disponíveis publicamente alguns *datasets* com imagens DICOM na internet. Adotamos neste trabalho o CQ500[13]. Ele contem 491 imagens de tomografia computadorizada da parte da cabeça de pacientes que tiverem seus dados sensíveis anonimizados.

Para complementar os testes, diversificando no tipo de exame e modalidade, a compressão/-descompressão também utilizou-se imagens obtidas de forma avulsa na internet, que variavam entre 5MB até 1GB, como os encontrados no site da fabricante de PACS Server OsiriX [14]. Para a correta avalização dos resultados obtidos em cada cenário foram definidas métricas para medir a eficiência na utilização dos recursos computacionais do modelo, bem como o seu desempenho. Tais métricas são destacadas a seguir:

- Tempo de Execução: Esta métrica corresponde ao tempo total, em segundos, de execução de determinada tarefa em relação a carga de dados aplicada;

[13] $http://headctstudy.qure.ai/dataset$
[14] $http://www.osirix-viewer.com/resources/dicom-image-library/$

- Utilização de recursos no Serviço DiCloud: quantidade média de uso de CPU das máquinas que estão executando o Stream API e Compressor/Descompressor;

- Largura de Banda: Corresponde a capacidade de comunicação de certa quantidade de dados por determinado período de tempo;

$$L(n) = \frac{D(n)}{T(seg)} \tag{5.1}$$

- Envios de Imagens por Minuto: Corresponde ao total de mensagens transmitidas em cada cenário dividido pelo tempo de execução. Equivalente a taxa de transferência (throughput) do modelo;

$$T(n) = \frac{M(n)}{T(min)} \tag{5.2}$$

- Taxa de Falha: quantidade de vezes que a Stream API ou Descompressor retornam respostas diferente de sucesso; Essa métrica indica se o Elasticity Manager está sendo efetivo na sua ação de não deixar o sistema entrar em sobre-carga;

- Nível de Compressão: Observa o nível de compactação relacionado ao tempo e a quantidade de dados compactados; Essa taxa será expressada em percentagem e indicará se está havendo ganho em realizar a compressão, e qual o melhor algoritmo para a maioria dos casos.

6 RESULTADOS

Nesse capítulo serão apresentados os resultados obtidos pelo DiCloud. Os resultados são baseados na metodologia apresentada na Seção 5. Inicialmente, serão apresentados resultados de uma analise realizada sobre os algoritmos aplicados no balanceador de carga da aplicação Stream API. Após, é apresentado os resultados do Elasticity Manager, e sua sub-rotina de predição usando ARIMA, incluindo os resultados dos testes realizados para definir quais parâmetros de configuração do ARIMA foram utilizados no DiCloud na Seção 6.2. Em seguida, são apresentados resultados dos testes de carga, demonstrando a alocação e desalocação de recursos de hardware, bem como as taxas de erro em ambos os cenários de elasticidade reativa e proativa. Após isso, serão apresentados os resultados da junção da aplicação DiCloud Client, com o Serviço DiCloud e do gerenciador de elasticidade na Seção 6.3. Essa Seção demonstra a comparação entre a elasticidade e o número fixo de recursos, baseando-se nas métricas apresentadas na Seção anterior. Por fim, são apresentados os resultados dos testes com diferentes algoritmos de compressão, demonstrando qual oferece a melhor taxa de compressão para o *dataset* escolhido.

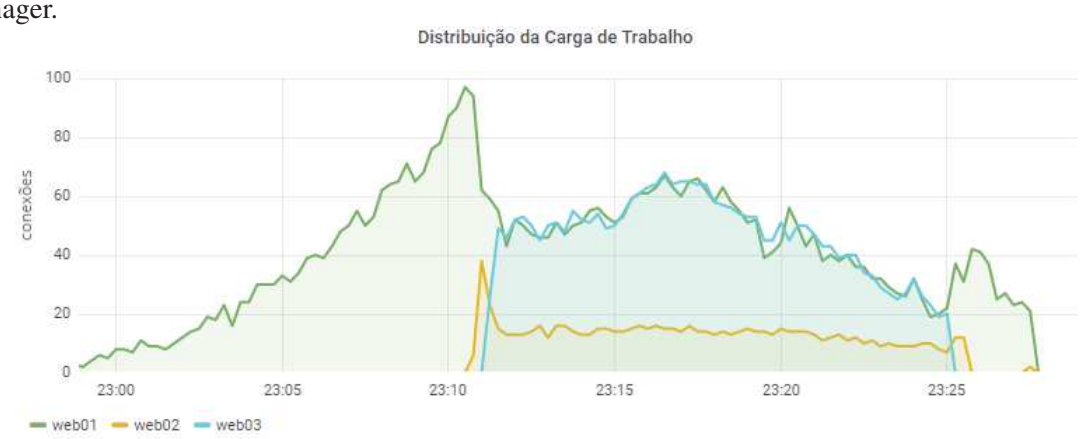
6.1 Balanceamento de Carga

Inicialmente, acreditava-se que o algoritmo de balanceamento *Round-Robin* seria ideal para distribuir a carga de trabalho entre os nós que hospedam a Stream API. No entanto, o comportamento observado durante a execução dos testes foi diferente, onde a distribuição realizada deixava máquinas com mais carga do que outras, mais precisamente, as últimas máquinas a serem ligadas ficavam com menos carga e as primeiras com mais carga. Isso se dá pelo fato de o algoritmo *Round-Robin* distribuir igualmente as novas conexões em formato de lista circular, desconsiderando o fato de conexões já existentes, antes das novas máquinas serem ligadas. Na Figura 15, que exemplifica esse comportamento, as máquinas dicloud-trl-1, dicloud-trl2 e dicloud-trl3, foram chamadas de web01, web02 e web03, respectivamente, a fim de simplificação.

De fato, o algoritmo *Round-Robin* é muito conhecido e difundido, sendo em muitas vezes o algoritmo padrão dos balanceadores de carga no ambiente web. Todavia, essa sua característica de distribuição lhe faz mais adequado para ambientes onde as conexões sejam de curta duração, onde o ciclo de novas conexões seja recriado em um curto período de tempo.

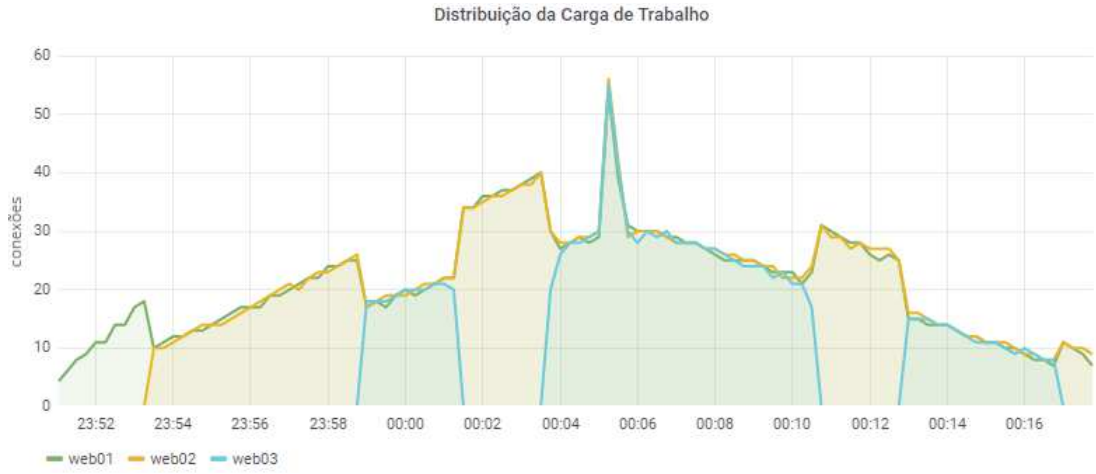
No cenário deste trabalho, o envio de arquivos (de imagens DICOM ou arquivo comprimido), aumenta o tempo que a conexão fica aberta com o servidor. Além disso, há o tempo para comprimir ou descomprimir o arquivo de imagem que está sendo recebido ou provido. Neste contexto, foi necessário a alteração do algoritmo de balanceamento para outro que tomasse como base a quantidade de conexões abertas em cada um dos servidores de Stream API. O *software* balanceador HAProxy que usamos possui como opção um algoritmo denominado

Figura 15 – Quantidade de conexões ativas em cada um dos serviços executando o componente Stream API com o algoritmo *Round-robin*. As máquinas são iniciadas e paradas pelo componente Elasticity Manager.



leastconn que fornecesse exatamente o comportamento que desejamos.

Figura 16 – Quantidade de conexões ativas em cada um dos serviços executando o componente Stream API com o algoritmo *Leastconn*. As máquinas são iniciadas e paradas pelo componente Elasticity Manager.



Quando alterado para este novo algoritmo temos o comportamento expresso na Figura 16, onde podemos ver que sempre que uma nova máquina virtual é ligada e começa a fornecer o serviço do Stream API, o balanceador busca rapidamente deixar todas as máquinas com a mesma quantidade de conexões. Isso implica em que as máquinas mais antigas parem de receber novas conexões, sendo o tráfego direcionado 100% para as novas máquinas até que a carga atual seja processada ou ao menos igualada antes que volte a receber novas conexões.

6.2 Elasticity Manager

Como foi apresentado na Seção 3, o ARIMA possui três parâmetros básicos de configuração: p, d e q. Então, para utilizarmos o ARIMA adequadamente, foi necessário definir os parâmetros de utilização. Escolher a melhor parametrização não é uma tarefa fácil e exige um

conhecimento de todas as possibilidades, bem como da aplicabilidade de cada modelo em cada cenário de utilização. Um modelo pode ser ótimo para uma carga de trabalho e péssimo para outra.

Por isso, foi criado um programa C# para simular as quatro cargas de trabalho definidas na Subseção 5.1.2, referente a criação da parametrização do ARIMA na metodologia de avaliação. Além disso, esse programa deve aplicar cada possibilidade do ARIMA em cada uma das cargas de trabalho, estimando valores futuros. Para isso foi utilizada a biblioteca R.Net [1] para integrar o C# com a linguagem R. Essa linguagem é própria para esses cálculos de previsão (*forecast*), pois possui diversos algoritmos prontos, entre eles o ARIMA. Na Listagem 6.1 é apresentado um pseudo-código de como o algoritmo se comporta e a implementação real está disponível no *github* [2].

Listing 6.1 – Pseudo-código para a avaliação do algoritmo ARIMA. A entrada deste algoritimo é uma lista com os valores medidos de CPU. A saída deste algoritmo é o valor projetado de futuro uso de CPU.

```
1. workload = workload_initialize();
2. for (p=0; p < 3; p++){
3.     for (d=0; d < 3; d++){
4.         for (q=0; q < 3; q++){
5.             workload_arima = {};
6.             workload_arima_previsto = {};
7.             arima_setup(p,d,q);
8.             for (i=0; i < workload.length; i++){
9.                 workload_arima = workload.next_value();
10.                arima_assign(workload_arima);
11.                workload_arima_previsto = arima_forecast(lookahead);
12.            }
13.            print(workload_arima_previsto);
14.        }
15.    }
16. }
```

O pseudo-código da Listagem 6.1 inicializa a variável workload na linha 1. Essa variável possui o *array* completo dos valores da simulação da carga de trabalho. Nas linhas 2, 3 e 4, são variados os parâmetros do ARIMA. Para cada nova execução do ARIMA, são inicializados variáveis de workload_arima e workload_arima_previsto, nas linhas 5 e 6.

O ARIMA é inicializado, na linha 7. Na linha 9 workload_arima irá receber cada um dos valores do *array* de workload sequencialmente, sendo que a linha 10 atribui os valores já carregados nessa variável para o ARIMA. Após isso, é chamada a função de predição (arima_forecast) na linha 11, sendo que esta retorna o próximo valor a ser adicionado no *array*. No final, na linha 13, são impressos os resultados.

[1] https://github.com/rdotnet/rdotnet
[2] https://github.com/trlthiago/Evaluator.git

Com base nos resultados dessa implementação foram comparados os valores retornados pela predição, com os valores reais da simulação. A Equação 6.1 descreve a apuração do erro, onde e_t é o erro do período t, A_t é o valor real no período t e P_t é a previsão para o período t.

$$e_t = A_t - P_t \tag{6.1}$$

Para encontrar o valor do erro da execução, foram utilizados dois cálculos de erro para a predição: Erro médio (EM) e Desvio médio absoluto (DMA). Em ambas estimativas de erros, valores mais próximos de zero indicam um melhor resultado. O erro médio, nada mais é do que uma média da diferença entre o valor esperado e o valor previsto de todas as observações. A Equação 6.2 apresenta como esse valor é apurado, onde e_t é o erro apurado anteriormente e n o número de períodos utilizados.

$$EM = \frac{\sum_{t=1}^{n} e_t}{n} \tag{6.2}$$

O problema dessa forma de avaliação é que se o algoritmo de predição errar a mesma diferença para mais e para menos, seu valor será zero, gerando um falso positivo de que o algoritmo é adequado. Assim, em uma nova etapa foi utilizado o Desvio médio absoluto. Este calcula o valor absoluto de cada uma das diferenças para fazer a média do valor absoluto. Assim, se o algoritmo errar a mesma diferença para mais e para menos, o valor estimado irá indicar isso mais adequadamente. A Equação 6.3 descreve a forma como o desvio médio absoluto é apurado.

$$DMA = \frac{\sum_{t=1}^{n} |e_t|}{n} \tag{6.3}$$

A Tabela 4 demonstra os resultados das estimativas de erro de cada algoritmo do ARIMA. Em amarelo, foram destacados os algoritmos que, em nossa análise, tiveram uma melhor performance. O algoritmo que apresentou valores mais aceitáveis foram os ARIMA(2,0,1) e ARIMA(0,2,0). Como já foi explicado na Seção 2, a predição com esse modelo utiliza o valor mais recente como um provável próximo valor, porém aplica sobre ele uma tendência, calculada com uma taxa de modificação dos últimos valores.

Dessa forma, o algoritmo ARIMA(0,2,0) foi perfeito nas cargas crescentes, decrescente e constante, bem como apresentou um valor razoável para a carga em onda. Porém, normalmente as aplicações reais não possuem uma carga muito bem definida como as cargas crescente, decrescente e constante. Aplicações reais variam bastante seu comportamento, de forma que o mais próximo da realidade seria a carga em onda. Sendo assim, o algoritmo ARIMA(2,0,1) apresentou bons resultados na carga em onda, sem ter problemas nas cargas crescente, decrescente e constante.

Para compará-lo aos demais modelos, é apresentado o gráfico da Figura 17. Nesse gráfico, comparamos os modelos que foram destacados na Tabela 4, visando comparar apenas os algoritmos que tiveram um bom desempenho. Além disso, é importante perceber que as séries temporais necessitam de alguns ciclos iniciais para começarem a prever, de forma que seus

Tabela 4 – Valores de erros dos algoritmos. Foram avaliados o erro médio (EM) e desvio médio absoluto (DMA)

	Crescente		Decrescente		Constante		Onda	
	EM	DMA	EM	DMA	EM	DMA	EM	DMA
Arima(0,0,0)	-186.25	186.25	186.25	186.25	-330.00	330.00	102.42	223.04
Arima(0,0,1)	-186.25	186.25	186.25	186.25	0.00	0.00	100.39	221.02
Arima(0,0,2)	-186.25	186.25	186.25	186.25	0.00	0.00	98.29	219.00
Arima(0,1,0)	-40.00	40.00	40.00	40.00	0.00	0.00	28.43	147.29
Arima(0,1,1)	-37.53	37.53	37.53	37.53	0.00	0.00	26.56	139.18
Arima(0,1,2)	-35.08	35.08	35.08	35.08	0.00	0.00	24.71	130.89
Arima(0,2,0)	0.00	0.00	0.00	0.00	0.00	0.00	-1.21	78.08
Arima(0,2,1)	302.50	302.50	-237.50	238.92	0.00	0.00	-1.48	69.72
Arima(0,2,2)	302.50	302.50	-297.50	297.50	0.00	0.00	-1.65	61.77
Arima(1,0,0)	-41.13	41.13	41.13	41.13	-330.00	330.00	70.01	157.79
Arima(1,0,1)	-39.13	39.13	39.54	39.54	0.00	0.00	60.90	140.34
Arima(1,0,2)	-33.15	33.15	33.54	33.54	0.00	0.00	33.22	128.60
Arima(1,1,0)	262.50	262.50	-262.50	262.50	0.00	0.00	-0.64	77.66
Arima(1,1,1)	92.55	135.57	-90.09	91.04	0.00	0.00	-1.04	73.22
Arima(1,1,2)	117.22	141.74	-39.81	55.94	0.00	0.00	-1.27	61.58
Arima(1,2,0)	302.50	302.50	-302.50	302.50	0.00	0.00	-5.63	116.02
Arima(1,2,1)	302.50	302.50	-237.50	238.92	0.00	0.00	-3.21	111.49
Arima(1,2,2)	302.50	302.50	-297.50	297.50	0.00	0.00	-6.11	108.10
Arima(2,0,0)	214.07	244.17	-213.72	244.53	-330.00	330.00	0.98	9.59
Arima(2,0,1)	-48.35	87.45	20.72	106.58	0.00	0.00	-2.35	9.32
Arima(2,0,2)	79.91	164.42	11.73	70.32	0.00	0.00	3.62	20.86
Arima(2,1,0)	262.50	262.50	-262.50	262.50	0.00	0.00	-1.37	22.03
Arima(2,1,1)	302.50	302.50	64.72	186.42	0.00	0.00	-4.77	9.36
Arima(2,1,2)	302.50	302.50	-278.16	285.61	0.00	0.00	1.75	5.24
Arima(2,2,0)	302.50	302.50	-302.50	302.50	0.00	0.00	19.27	124.69
Arima(2,2,1)	302.50	302.50	-237.50	238.92	0.00	0.00	-0.01	17.83
Arima(2,2,2)	302.50	302.50	-297.50	297.50	0.00	0.00	7.12	15.89

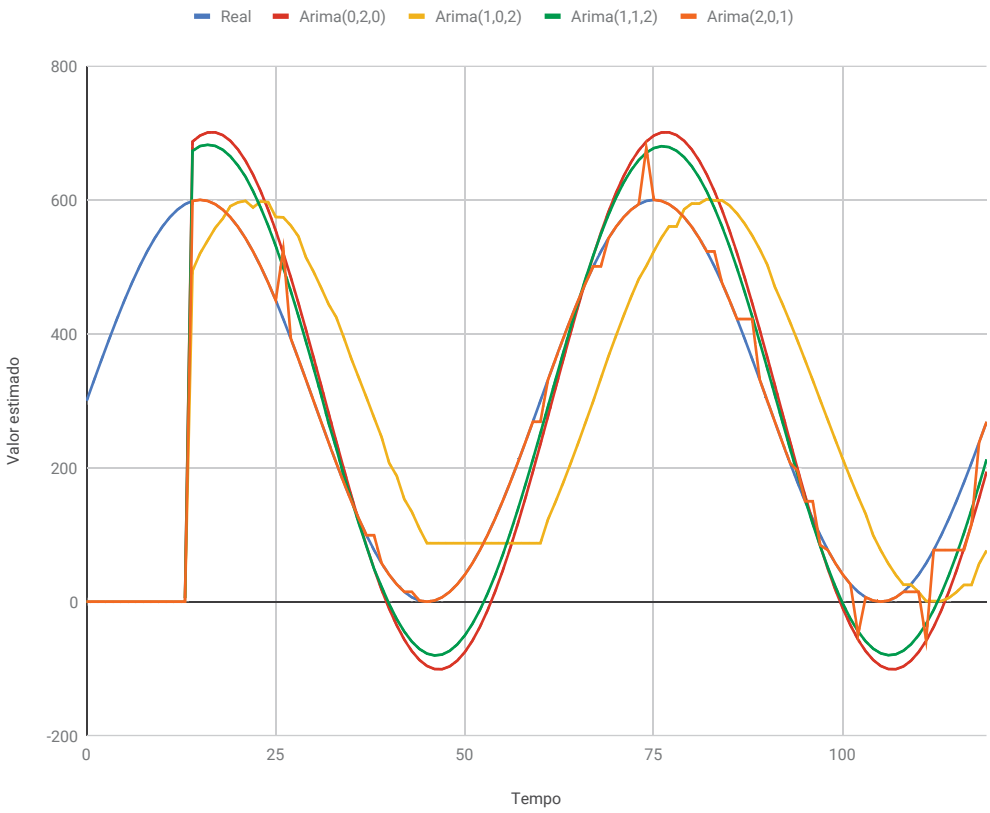
Fonte: Elaborado pelo autor.

valores iniciais são retornados como zero. Isso deverá ser considerado ao implementarmos o protótipo de predição.

O algoritmo ARIMA(0,2,0) apresentou resultados perfeitos para as cargas de trabalho constante, crescente e decrescente. Para a carga de trabalho em onda, pode-se ver na tabela que o resultado destes parâmetros não é o melhor. O melhor algoritmo, analisando o Desvio Médio Absoluto, seria o modelo ARIMA(2,1,2). Não avaliamos esse modelo, pois seus resultados foram bastante insatisfatórios para as cargas de trabalho crescente e decrescente.

Então, o segundo melhor algoritmo para a carga de trabalho em onda foi o ARIMA(2,0,1), que está demonstrado no gráfico. Ele possui um desempenho melhor para a carga em onda e razoável para os algoritmos crescente e decrescente. Analisando seu gráfico de onda pode-se perceber um efeito indesejado na forma da onda, que apresenta alguns pontos de pico nos valores previstos. Porém, na maior parte do tempo ele acompanha a carga de trabalho adequadamente. Já o modelo ARIMA(0,2,0) apresenta uma onda deslocada da onda original, prevendo valores abaixo ou acima do real. Sendo assim, fica claro que para esse tipo de carga de trabalho, o algoritmo ARIMA(2,0,1) é o mais adequado.

Figura 17 – Análise comparativa (da linha real, em azul) do tipo de carga em onda, com suas predições (demais series em outras cores). São apresentados apenas os algoritmos ARIMA com melhor desempenho. Quanto mais alinhadas as series ARIMA estiverem com a linha real (azul) melhor.



Fonte: Elaborado pelo autor.

6.3 Resultados do DiCloud

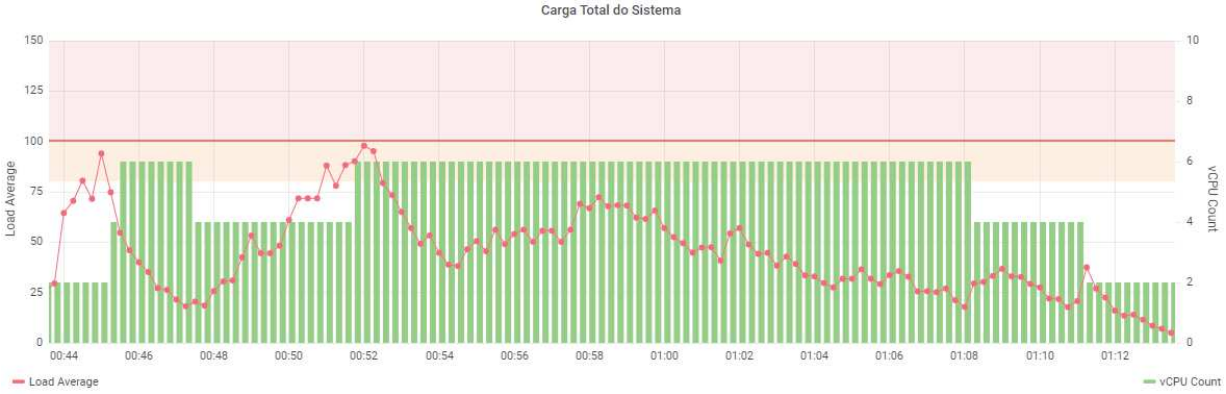
Nessa Seção, serão apresentados os resultados obtidos da execução dos testes detalhados na Seção 5. Como apresentado na Seção anterior, foram modeladas quatro cargas de trabalho: crescente, decrescente, constante e em onda. Essas cargas de onda simulam aplicações reais. Tomando a carga de onda como representante do tipo de carga de trabalho aplicada na Stream API, temos as seguintes representações de alocação de recursos: a Figuras 18 apresenta o escalonamento usando o ARIMA, acrescentando proatividade na alocação de desalocação de recursos e a Figura 19 o modelo sem uso do ARIMA.

Nestes gráficos temos duas informações que se complementam. A linha em vermelho representa a carga de CPU que o sistema está demandando. Por sistema entende-se a média de todos os núcleos de processamento que estão respondendo requisições da Stream API, e nas barras em verde, com valores no eixo Y secundário temos a quantidade de núcleos ativos em cada instante de tempo. Conforme especificado na Seção de metodologia o tamanho de instâncias de máquinas provisionados incrementam a quantidade de núcleos de dois em dois.

Figura 18 – Taxa de uso de CPU de todas as máquinas virtuais que estão servindo a aplicação Stream API no modo **proativo**. A linha em vermelho (eixo Y esquerdo) representa a carga (% taxa de uso de CPU) que o sistema está demandando. As barras verdes (eixo Y direito) representam a quantidade de núcleos de processamento disponíveis no sistema a cada instante de tempo. A tarja em amarelo representa o *threshold* superior, a partir de onde o sistema deve provisionar mais máquinas para ajudar no processamento.

Um algoritmo de gestão de carga proativo visa manter a melhor combinação de alocação de recursos sem comprometer a qualidade do sistema. Sem um sistema de gestão proativo os recursos podem ser alocados tardiamente ou ficarem rodando por mais tempo que o necessário. Tal cenário reativo não se demonstra otimizado para ambientes onde custo financeiro é um parâmetro de decisão no negócio.

Figura 19 – Taxa de uso de CPU de todas as máquinas virtuais que estão servindo a aplicação Stream API no modo **reativo**. A linha em vermelho (eixo Y esquerdo) representa a carga (% taxa de uso de CPU) que o sistema está demandando. As barras verdes (eixo Y direito) representam a quantidade de núcleos de processamento disponíveis no sistema a cada instante de tempo. A tarja em amarelo representa o *threshold* superior, a partir de onde o sistema deve provisionar mais máquinas para ajudar no processamento..



Na Figura 18 vemos que máquinas são instanciadas e desalocadas mais vezes em comparação a Figura 19. Essas decisões foram tomadas baseando-se no resultado do algoritmo de predição do ARIMA, e buscam deixar a alocação de máquinas o mais próxima possível da

demanda que realmente tem-se presente no sistema. Apesar da quantidade de *threads* programadas no JMeter respeitarem um padrão visível de onda como apresentado na Figura 13, o comportamento do uso de CPU do sistema não segue este mesmo formato, por alguns motivos que envolvem justamente a alocação e desalocação de máquinas, e o tempo de *upload* das imagens que é variável, uma vez que passam pela internet.

Na Figura 19 pode-se observar que as três máquinas permanecem ativas por mais tempo. Tomando que cada barra verde do gráfico é um instante de tempo, temos que as três máquinas permanecem ligadas em paralelo durante 74 instantes de tempo, no modo reativo. Enquanto que no modo proativo, essa quantidade reduz para 54. Dessa forma, se comprova que quando existe uma variação na carga de trabalho é menos custoso utilizar o sistema do que deixar 3 máquinas ligadas o tempo todo ou ainda mesmo algum método reativo.

A Tabela 5 mostra os dados sumarizados dos testes realizados, apresentados nas Figuras 19 e 18. Observa-se que mais máquinas ligadas conseguem dar uma vazão maior, pois foram possíveis atender 5148 requisições ao total, contra 5032 requisições quando usando o modo proativo, que reduz a quantidade de máquinas a fim de otimizar o uso dos recursos. Essa quantidade menor de requisições não aconteceu por causa de negação de serviço ou alta taxa de erros, que se pode verificar que em ambos os casos a taxa foi de 0% de erros. Essa diferença se justifica pela configuração do JMeter que mantém um determinado número de *threads* ativas conforme configuração da quantidade de carga que se deve exercer sobre o sistema.

Então, o parâmetro a se considerar pelo ponto de visto do servidor que hospeda a Stream API é o tempo médio de resposta, pois este impacta na taxa de envios. O cenário sem ARIMA conseguiu atender na média 442 ms mais rápidos as requisições, o que elevou a taxa de 2.8 requisições por segundo para 2.9 requisições por segundo. Todavia, há de se ponderar se 0.1 a mais justifica o custo computacional, financeiro e energético de deixar uma máquina extra ligada por mais tempo.

Tabela 5 – Valores dos resultados dos testes realizados usando a ferramenta JMeter. Os resultados são a média de todas as requisições realizadas durante os 30 minutos de duração do teste programados.

Método	**Requisições**	**Tempo médio**	**Std. dev. do tempo médio**	**Taxa de envio**	**% erro**
Sem Arima	5148	16614 ms	5908.36	2.9/sec	0.00%
Com Arima	5032	17056 ms	6639.20	2.8/sec	0.00%

A Tabela 5 também apresenta o desvio padrão em relação ao tempo médio de resposta das requisições. Esse valor é relevante para entendermos o quanto de dispersão de tempo se tem em relação ao tempo médio. O quanto mais baixo é o valor do desvio padrão, mais consistente é o conjunto das informações. Neste caso, temos que o desvio padrão para simulação sem ARIMA é de 5908.36 ms, e 6639.20 ms para simulação com ARIMA. Ambos os valores são satisfatórios quando os analisamos sobre a ótica que valores razoáveis são aqueles que ficam menores ou igual à metade da média. Ou seja, no caso da simulação com ARIMA por exemplo, temos que a metade da média é 8528 ms, estando portanto, 6639.20 ms dentro desse valor.

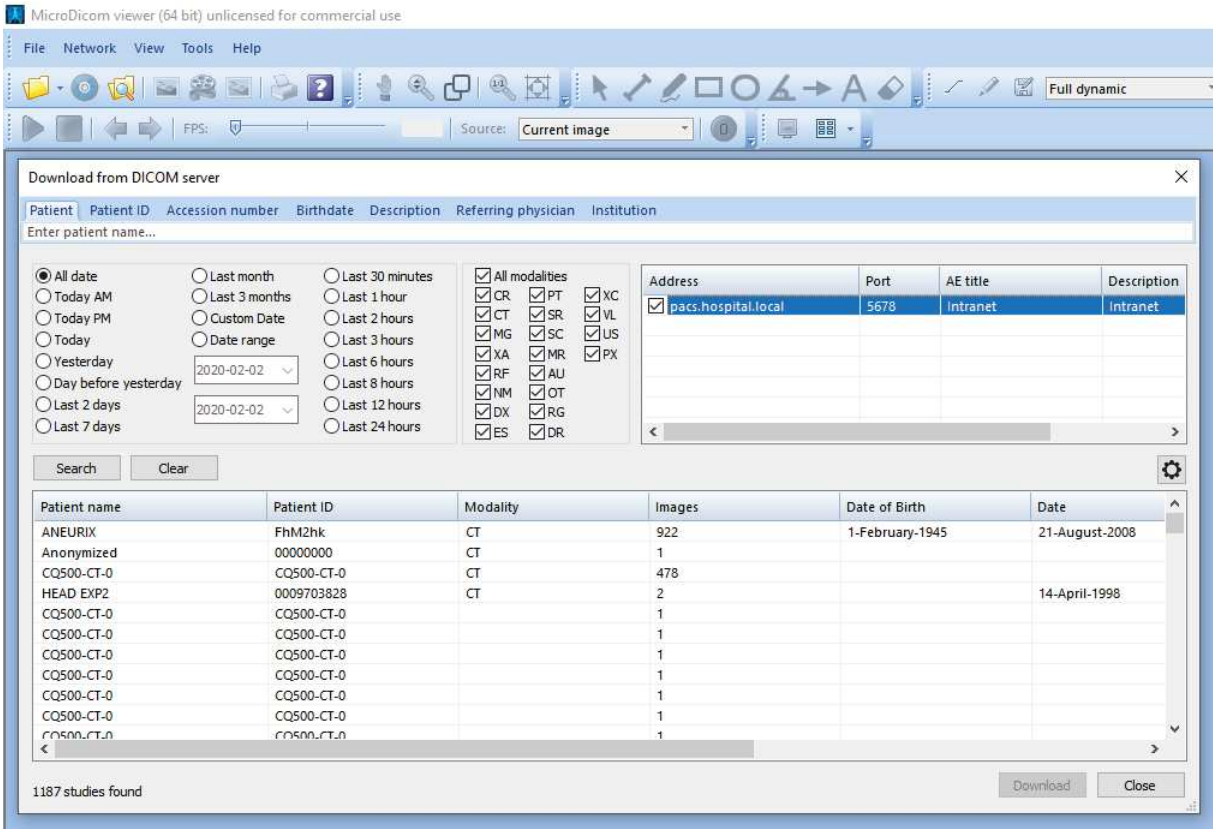
6.4 Operação fim-a-fim

Um dos itens a serem testados neste trabalho é a operação do sistema dentro do ambiente hospitalar, sem gerar alterações as configurações do ambiente de trabalhos dos médicos e especialistas. Como descrito na Seção de apresentação do modelo proposto por este trabalho, o DiCloud Client substitui a porta de acesso atual do sistema PACS do hospital, colocando-se como intermediário das comunicações entre os softwares visualizadores de imagens DICOM presente nas estações de trabalhos médicas e o PACS.

As operações que precisam ser manipuladas para realizarem consultas externas, são tratados pelo DiCloud Client e os demais são repassados diretamente ao software PACS (*bypass*). A Figura 20 mostra a tela de configuração de rede do software MicroDICOM utilizado nos testes. Como esperado, não foi necessário fazer nenhuma alteração, mantendo-se o endereço e porta de conexão. Nesta tela também vemos as opções de pesquisa que são disponibilizadas e na parte inferior a listagem dos exames encontrados ao se clicar no botão "*Search*". A pesquisa é realizada através da operação C-FIND, cujo qual, foi tradado pela aplicação DiCloud Client.

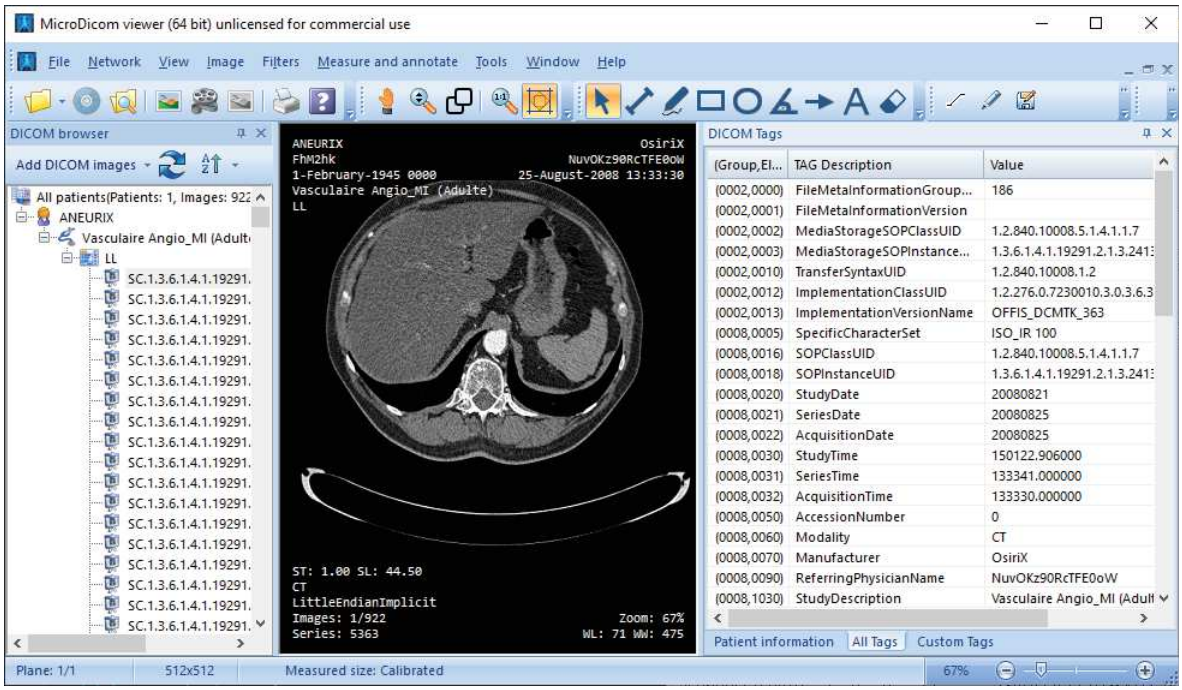
Ao receber uma requisição C-FIND, o DiCloud Client verifica a existência de imagens no PACS local que atendam os critérios de pesquisa especificados, e por conseguinte, consulta o Serviço DiCloud, unindo os resultados e retornando ao visualizador.

Figura 20 – Tela de pesquisa e configuração de rede do *software* visualizador de imagens MicroDICOM.



Ao se efetuar duplo-click em alguma das imagens exibidas na listagem de resultados, o software executa o procolo de C-MOVE para fazer o download da imagem. Esta operação também foi tratado pela aplicação DiCloud Client, que executa o download da origem, seja o PACS local ou do Serviço DiCloud, e neste caso, efetua a descompressão do arquivo, e por conseguinte responde ao MicroDICOM com o arquivo solicitado. A Figura 21 mostra o MicroDICOM exibindo corretamente uma imagem baixada através do DiCloud Client. Ao lado esquerdo temos todos os *frames* que o exame possui e ao lado direito temos todas as *tags* (metadados) que estão contidas no arquivo DICOM baixado.

Figura 21 – Tela de exibição de exames do *software* MicroDICOM. O Exame exibido foi acessado através do DiCloud Client.



Um item a se considerar neste trabalho é a adição do DiCloud Client, sendo este um item a mais no processo de submissão de imagem da modalidade ao PACS. Neste caso, podemos estimar qual é o *overhead* que esta abordagem trás consigo. Usando a mesma imagem utilizada nos testes da Stream API, foi desenvolvido um programa que submete este mesmo arquivo 100 vezes sequencialmente para o DiCloud Client. Em cada iteração do código foi anotado o tempo gasto em milissegundos. O mesmo teste foi executado também enviando a imagem diretamente ao PACS.

A Tabela 6 apresenta o resultado dos testes realizados, onde os 3 sistemas (código simulador de modalidade, PACS e DiCloud Client) estavam rodando na mesma máquina, removendo a variável rede/internet, buscando observar somente o *overhead* adicionado pelo processamento das aplicações. O envio pela internet certamente introduziria um tempo extra de acordo com o tamanho do arquivo e velocidade da conexão, porém, este tempo não bloqueia o uso da modalidade ou demais sistemas, uma vez que ocorre de forma assíncrona, como uma tarefa de segundo

Tabela 6 – Média de tempo necessário para submeter uma imagem DICOM da modalidade (simulador) aos serviços de armazenamento.

Caso	Tempo em milissegundos
Do simulador de modalidade diretamente ao PACS	113
Do simulador de modalidade ao DiCloud Client, enviando somente ao PACS	134
Do simulador de modalidade ao Dicloud Client, enviando ao PACS e a Stream API	679

plano. Com estas informações, pode-se responder diretamente ao item de avaliação da Subseção 5.3, ao expor que é possível enviar 447 imagens por minuto quando enviando do DiCloud Client ao PACS somente, e 88 imagens por minuto enviando a imagem ao PACS e também ao Stream API.

6.5 Compressão

A ferramenta definida neste trabalho para realizar a compressão e descompressão dos arquivos DICOM foi a 7Zip conforme expresso na Subseção 4.1, mas o tipo de arquivo a ser gerado, o método e o nível de compressão a serem usados, precisavam de avaliação para se definir qual tem a maior taxa de compressão. O Centro Médico da Universidade de Pittsburgh (UPMC [3]) disponibiliza um *dataset* [4] de tomografia de peito usando as técnicas FFDM (*Full-Field Digital Mammography*) e Ductogram, com 25 exames anonimizados.

O tamanho total desdes 25 exames somam 4.97GB (5.338.031.302 bytes). Adotou-se este *dataset* para avaliar os algoritmos de compactação oferecidos pelo 7Zip, pois contém arquivos gerados por diferentes modalidades, contendo diferentes tamanhos, com o menor arquivo sendo de 9338756 bytes e o maior de 941643712 bytes. A Tabela 7 apresenta os valores das taxas de ganho de cada um dos métodos de compressão testados.

A Tabela 7 demonstra que há ganhos expressivos em vários casos, que colaboram diretamente na redução de uso de link de internet e tempo para realizar seu envio. A linha de Taxa Média % (destacado em amarelo) mostra a média das 25 taxas de ganho obtidas no teste. As taxas médias obtidas indicam que há vantagem em compactar arquivos DICOM antes de realizar seu envio.

Não obstante, os valores podem ser vistos pela ótica de um período de tempo, onde analisase o tamanho total do *dataset* sem compressão *versus* o tamanho total de todos os arquivos comprimidos, que é de fato o que será transmitido pela internet. Neste caso, a Tabela 8 demonstra que os ganhos são ainda melhores por esta nova perspectiva. Também é possível estabelecer que o melhor método de compressão que a ferramenta 7Zip pode oferecer para imagens DICOM

[3] $https://www.upmc.com/$

[4] $https://dl.dropbox.com/s/iluqskc0ybo4zkl/MammoTomoUPMC_Case14.tar.bz2?dl=1$

Tabela 7 – Valores dos ganhos (em %) de cada um dos métodos de compressão testados, sobre cada um dos arquivos do *dataset*. O nome dos arquivos tem o prefixo "1.3.6.1.4.1.5962.99.1.2280943358.716200484.1363785608958.", que foi substituído por ** na tabela a fim de simplificação.

		Taxa de ganho na compressão em cada método							
	File Bytes	**7z + LZMA**	**7z+LZMA2**	**7z+BZip2**	**7z+Deflate**	**gzip**	**BZip2**	**zip+deflate**	**zip+PPMd**
**.33.0.dcm	9338756	2.55%	2.53%	2.18%	2.77%	2.77%	2.19%	2.77%	2.19%
**.30.0.dcm	9588172	2.43%	2.41%	2.12%	2.76%	2.76%	2.13%	2.43%	2.29%
**.28.0.dcm	11271500	2.52%	2.50%	2.04%	2.80%	2.80%	2.04%	2.80%	2.55%
**.22.0.dcm	13537426	1.92%	1.90%	1.56%	2.22%	2.22%	1.56%	2.21%	1.99%
**.24.0.dcm	14190858	1.87%	1.85%	1.50%	2.19%	2.19%	1.50%	2.19%	2.06%
**.26.0.dcm	14423482	2.05%	2.04%	1.55%	2.35%	2.35%	1.56%	2.35%	2.19%
**.2.0.dcm	17043738	47.26%	47.17%	52.30%	36.46%	36.46%	52.30%	36.46%	54.98%
**.8.0.dcm	17043740	48.24%	48.20%	53.09%	37.63%	37.63%	53.09%	37.63%	55.87%
**.6.0.dcm	27267290	81.92%	81.91%	84.09%	77.45%	77.45%	84.09%	77.45%	84.95%
**.16.0.dcm	27267402	66.37%	66.36%	69.42%	59.12%	59.12%	69.42%	59.12%	71.13%
**.10.0.dcm	27267404	68.99%	68.99%	72.33%	61.68%	61.68%	72.33%	61.68%	74.10%
**.13.0.dcm	27267420	65.50%	65.49%	69.08%	57.99%	57.99%	69.08%	57.99%	70.91%
**.19.0.dcm	27267420	73.25%	73.25%	76.08%	67.40%	67.40%	76.08%	67.40%	77.41%
**.46.0.dcm	59147056	2.35%	2.33%	1.34%	2.79%	2.79%	1.34%	2.79%	2.65%
**.44.0.dcm	64526444	1.27%	1.25%	0.65%	1.91%	1.91%	0.65%	1.91%	1.44%
**.42.0.dcm	74570536	1.70%	1.68%	0.67%	2.45%	2.45%	0.67%	2.45%	1.53%
**.36.0.dcm	84948550	1.43%	1.41%	0.52%	2.18%	2.18%	0.52%	2.18%	1.25%
**.40.0.dcm	92664552	1.28%	1.26%	0.41%	2.11%	2.11%	0.41%	2.11%	0.96%
**.38.0.dcm	105655550	1.47%	1.46%	0.51%	2.22%	2.22%	0.51%	2.22%	1.36%
**.48.0.dcm	525219532	71.20%	71.16%	73.86%	63.55%	63.55%	73.86%	63.55%	75.22%
**.52.0.dcm	743720262	80.74%	80.72%	82.44%	75.78%	75.78%	82.44%	75.78%	83.66%
**.56.0.dcm	774897182	76.53%	76.50%	78.59%	70.75%	70.75%	78.59%	70.75%	79.32%
**.50.0.dcm	804323032	78.22%	78.19%	80.50%	72.33%	72.33%	80.50%	72.33%	81.25%
**.58.0.dcm	823940286	81.62%	81.59%	83.43%	77.02%	77.02%	83.43%	77.02%	84.02%
**.54.0.dcm	941643712	76.81%	76.77%	79.08%	70.85%	70.85%	79.08%	70.85%	79.88%
Total	5338031302								
% Média		37.58%	37.56%	38.77%	34.27%	34.27%	38.77%	34.26%	39.81%

de tomografia de peito é Zip + PPMd, destacado em amarelo na tabela.

Tabela 8 – Valores dos ganhos (em %) de cada um dos métodos de compressão testados, sobre sobre a soma de todos os arquivos comprimidos do *dataset*.

Método	**Soma de todos os arquivos comprimidos**	**Diferença (em MB)**	**Ganho total**
zip+Deflate	1895357864 bytes	3283.189238	64.49%
7z+Deflate	1895325642 bytes	3283.219967	64.49%
GZip	1895321889 bytes	3283.223546	64.49%
7z+LZMA2	1624803645 bytes	3541.209847	69.56%
7z+LZMA	1623289212 bytes	3.459623167	69.59%
7z+LZip2	1524956815 bytes	3636.431205	71.43%
BZip2	1524950965 bytes	3636.436784	71.43%
Zip+PPMd	1477407270 bytes	3681.777985	72.32%

É importante ressaltar que diferentes conjuntos de imagens podem produzir diferentes resultados de taxas de compressão. Isso pode acontecer pelo fato da quantidade de *tags* que são aplicadas em um arquivo DICOM ser variável, tanto pelo tipo de exame, fabricante da modalidade quanto pelas edições médicas que podem ser feitas nos arquivos, adicionando informações, anotações e observações.

7 CONCLUSÃO

A partir de 1970, com o surgimento da tomografia computadorizada, as imagens médicas se tornaram parte fundamental do fluxo de diagnóstico. A padronização das imagens através do formato DICOM representou um grande avanço para a interoperabilidade das máquinas de capturas das imagens médicas, bem como para os softwares que realizam a visualização. Todavia, atualmente, ainda existem desafios que precisam ser perpassados para aperfeiçoar os processos, visando agilizar o diagnóstico clínico.

A Seção 1.2 apresentou a seguinte questão de pesquisa: ***Como seria um modelo com elasticidade proativa em nuvem para compartilhamento de imagens DICOM entre paciente, hospitais e clínicas com o mínimo de modificações no ambiente hospitalar, com internet limitada?*** Para respondê-la, foi criado o modelo para compartilhamento de imagens médicas DiCloud.

O DiCloud foi desenvolvido baseado nas características comuns e nas lacunas identificadas nos trabalhos relacionados, com o intuito de sanar os problemas levantados na questão de pesquisa. Como demonstrado na Seção 3, existem oportunidades de melhorar a acessibilidade de imagens médicas, provendo o compartilhamento e melhorando os tempos de envio e recebimento das imagens DICOM, além de poupar largura de banda de tráfego total de dados.

Por isso, o modelo DiCloud, aborda essas lacunas existentes, interligando hospitais (incluindo clínicas), médicos e pacientes, bem como provê elasticidade proativa no Serviço DiCloud, dada como crítica, almejando um desempenho igual ou superior a uma abordagem reativa, reduzindo custos operacionais e não requerendo modificações no fluxo de trabalho médico.

O Gerenciador de elasticidade criado avalia a carga do sistema através da métrica CPU, aplicando-a em um algoritmo de séries temporal chamado ARIMA (modelo auto-regressivo integrado de médias móveis), capaz de prever a carga de trabalho futura do sistema. Com uma previsão antecipada da carga do sistema, melhores decisões de escalonamento e elasticidade podem ser adotadas. Os resultados demonstram que a aplicação do ARIMA(2,0,1) reduziu em 27% o tempo de alocação de instâncias, mantendo a taxa de 0% de falhas nas requisições.

Conclui-se que o DiCloud opera de forma efetiva no compartilhamento de imagens médicas, tornando possível que especialistas acessem imagens remotamente, sem a necessidade de mudar o ambiente hospitalar. Essa comunicação foi viabilizada no ambiente em que a velocidade de conexão com a internet é baixa devido a aplicação de compressão nos arquivos transmitidos. Os testes revelaram quais são os algoritmos de compressão que tem as melhores taxas de ganho para o *dataset* de imagens utilizado, sendo o de maior ganho o método Zip+PPMd, com 72.32% de redução no tamanho total do *dataset*.

Com arquivos menores sendo enviados temos um aproveitamento melhor da conexão com a internet, além de um tempo inferior de envio e recebimento das imagens. A transmissão de imagens entre hospitais, médicos e pacientes traz ganhos para o fluxo de trabalho e atendimento agilizado ao paciente, como também trás benefícios administrativos como a redução de duplicação de exames, no que concerne espaço de armazenamento e tempo de alocação da máquina

modalidade que realiza o exame.

7.1 Contribuições

As contribuições do modelo DiCloud focam no compartilhamento das imagens médicas e na tomada de decisão de elasticidade proativa para manter o bom desempenho do Serviço DiCloud, reduzindo os custos operacionais. Definitivamente, uma das principais vantagens do uso deste modelo é oferecer aos profissionais em todo o mundo uma plataforma para acessar imagens médicas que não estão em seu local e poder diagnosticar doenças importantes, onde um diagnóstico de um especialista é essencial em tempo hábil, diferindo-se de um diagnóstico tardio, podendo salvar uma vida (GORDO, 2000).

Essa característica do sistema estar online, sendo acessível de qualquer lugar e a qualquer momento, viabiliza que as informações sejam disponibilizadas através de uma nova gama de dispositivos, que anteriormente não podiam obter acesso devido a necessidade de mudanças na estrutura de TI dos hospitais e clínicas para prover o acesso. Os aspectos deste modelo apontam para a redução de exames duplicados que são realizados quando não se tem meio de acesso à exames realizados por clínicas e hospitais diferentes.

De forma sumária, contribui-se de forma acadêmica-computacional nos seguintes aspectos:

- Modelo transparente ao usuário e com desempenho para compartilhamento imagens médicas com especialistas geograficamente distantes;

- Heurística para decisões de elasticidade proativa de aplicações de compressão, envio e recebimento de imagens médicas, visando manter e/ou melhorar a qualidade do sistema e reduzir os custos operacionais;

7.2 Limitações

Esta proposta está ciente de questões de privacidade e segurança que envolvem a transmissão e acesso dos arquivos de imagens médicas. No modelo prevemos alguns mecanismos conhecidamente seguros para a transmissão dos dados, como TLS e mecanismo de autorização, no entanto, deixou-se para um trabalho futuro propostas que abordem melhorias de segurança diretamente nos protocolos DICOM.

Ademais, as configurações de recursos das máquinas virtuais utilizadas nos testes para servir a aplicação Stream API respeitam algumas limitações impostas pelo tipo de assinatura de serviço. Em um cenário onde a quantidade de máquinas ou recursos não fosse um fator restritivo, se poderia executar cenários de testes simulando cargas de trabalho ainda maiores, através da fórmula utilizada para gerar a onda, na qual os testes com o JMeter se baseiam.

Outra limitação que encontramos é relacionado aos acessos as imagens originais de exames, que não tenham passado por processo de anonimização. Para preservar os dados dos pacientes,

as entidades que detém as imagens antes de liberar o acesso para terceiros, ou até mesmo liberar publicamente, submetem as imagens a um processo que remove dados sensíveis do paciente. Neste processo, as *tags* que contém as informações são removidas, ou algumas vezes o texto que é armazenado na *tag* é substituído por uma *string* vazia ou palavra que denote o significado de "omitido". Existe a hipótese de a taxa de compressão ser maior do que a apresentadas nos testes, dado que os textos originais tendem a ser maiores que os aninomizados.

Por fim, esta proposta se limita aos aspectos computacionais envolvidos no fluxo de recebimento e transmissão de imagens médicas. O modelo se propõe a não gerar mudanças no ambiente de trabalhos dos especialistas da área da saúde, todavia, demais aspectos culturais ou políticos que podem surgir no que se refere a adoção de um sistema que implemente o modelo não são abordados nesta proposta e podem ser objeto de pesquisas futuras, inclusive em conjunto com outras áreas que melhor se relacionam com este tema.

7.3 Trabalhos futuros

Aqui, destacam-se algumas limitações do modelo DiCloud que podem ser vistas como oportunidades de otimizações para trabalhos futuros: Comparar com algoritmos reativos e *machine learning*. Por exemplo, outra possibilidade de algoritmo proativo que poderia ser utilizada na predição do DiCloud é no aprendizado de máquina. Essa abordagem requer um tempo maior de aprendizado, porém gera resultados com uma acurácia maior. Poderia ser implementado algum algoritmo de redes neurais, por exemplo, comparando-o com o ARIMA. Além disso, poderiam ser testados outros algoritmos do ARIMA.

Conforme demonstrado nos testes, existem outros algoritmos que poderiam ser testados, tais como, ARIMA(0,2,0) que apresentou melhores resultados para carga crescente e decrescente. O algoritmo selecionado possui uma tendência conservadora, onde flutuações nas cargas de trabalho não causam um impacto imediato. Dessa forma, seria interessante comparar uma abordagem menos conservadora e ver seus impactos na alocação e desalocação das máquinas virtuais.

O desenvolvimento de um sistema de escolha automática do método de compressão também é um ponto a ser melhor elaborado em futuros trabalhos, que podem ampliar os testes com outros *datasets*, a fim de concluir-se o melhor algoritmo de compressão para cada tipo de arquivo DICOM, visando obter a melhor taxa dentro de um tempo de compressão aceitável.

REFERÊNCIAS

ACHARYA, T.; TSAI, P.-S. **Jpeg2000 standard for image compression**: concepts, algorithms and vlsi architectures. John Wiley & Sons, 2005.

AL-DHURAIBI, Y. et al. Elasticity in cloud computing: state of the art and research challenges. **IEEE Transactions on Services Computing**, v. 11, n. 2, p. 430–447, 2017.

ALHAJERI, M. **Future developments and trends in use of picture archiving and communication systems**. 2016. Tese (Doutorado em Ciência da Computação) — Brunel University London, 2016.

ANUJA, M.; JEYAMALA, C. A survey on security issues and solutions for storage and exchange of medical images in cloud. **International Journal of Emerging Trends in Electrical and Electronics**, v. 11, n. 6, p. 27–32, 2015.

ARYANTO, K.; OUDKERK, M.; OOIJEN, P. van. Free dicom de-identification tools in clinical research: functioning and safety of patient privacy. **European radiology**, v. 25, n. 12, p. 3685–3695, 2015.

BANKOLE, A. A.; AJILA, S. A. Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment. In: IEEE SEVENTH INTERNATIONAL SYMPOSIUM ON SERVICE-ORIENTED SYSTEM ENGINEERING, 2013., 2013, Ottawa, Canada. **Anais...** IEEE, 2013. p. 156–161.

BARRETT, E.; HOWLEY, E.; DUGGAN, J. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. **Concurrency Computation Practice and Experience**, v. 25, n. 12, p. 1656–1674, 8 2013.

BAYER, F. M. et al. Previsão do preço e da volatilidade de commodities agrícolas, por meio de modelos arfima-garch. **Universidade Federal de Santa Maria**, 2008.

BIRMAN, K. P. **Guide to reliable distributed systems**: building high-assurance applications and cloud-hosted services. Springer Science & Business Media, 2012.

BREBNER, P. C. Is your cloud elastic enough? In: WOSP/SIPEW INTERNATIONAL CONFERENCE ON PERFORMANCE ENGINEERING - ICPE '12, 2012, Boston, Massachusetts, USA. **Proceedings...** ACM Press, 2012. n. 1, p. 263.

BROWN, W.; SHEPHERD, B. J. **Graphics file formats; 2nd ed.** Greenwich, CT, USA: Manning Publications Co., 1995.

BRUTHANS, J. The successful usage of the dicom images exchange system (epacs) in the czech republic. **Applied Clinical Informatics**, v. 11, n. 01, p. 104–111, 2020.

BRUYLANTS, T.; MUNTEANU, A.; SCHELKENS, P. Wavelet based volumetric medical image compression. **Signal processing: Image communication**, v. 31, p. 112–133, 2015.

BUYYA, R. et al. High performance cluster computing: architectures and systems (volume 1). **Prentice Hall, Upper SaddleRiver, NJ, USA**, v. 1, p. 999, 1999.

CONCI, A.; AZEVEDO, E.; LETA, F. R. Computação gráfica: teoria e prática. **Rio de Janeiro: Campus**, v. 2, 2008.

DA ROSA RIGHI, R. et al. Enhancing performance of iot applications with load prediction and cloud elasticity. **Future Generation Computer Systems**, 2018.

DA ROSA RIGHI, R. et al. Towards providing middleware-level proactive resource reorganisation for elastic hpc applications in the cloud. **International Journal of Grid and Utility Computing**, v. 10, n. 1, p. 76–92, 2019.

DREYER, K. J. et al. **A guide to the digital revolution**. Springer, 2006.

EHLERS, R. S. Análise de séries temporais. **Laboratório de Estatística e Geoinformação. Universidade Federal do Paraná**, 2007.

ERICKSON, B. J. Experience with importation of electronic images into the medical record from physical media. **Journal of digital imaging**, v. 24, n. 4, p. 694–699, 2011.

GALANTE, G. et al. An analysis of public clouds elasticity in the execution of scientific applications: a survey. **Journal of Grid Computing**, v. 14, n. 2, p. 193–216, 2016.

Galletta, A. et al. An approach to share mri data over the cloud preserving patients' privacy. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS (ISCC), 2017., 2017. **Anais...** IEEE, 2017. p. 94–99.

GONG, Z.; GU, X.; WILKES, J. Press: predictive elastic resource scaling for cloud systems. In: INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT, CNSM 2010, 2010., 2010, Niagara Falls, ON, Canada, Canada. **Proceedings...** IEEE, 2010. p. 9–16.

GORDO, J. A. Cancer de mama. manejo desde atención primaria. **SEMERGEN-Medicina de Familia**, v. 26, n. 10, p. 491–501, 2000.

GUO, X.; ZHUANG, T.-g. Lossless watermarking for verifying the integrity of medical images with tamper localization. **Journal of digital imaging**, v. 22, n. 6, p. 620, 2009.

HOLST, D. **Dicom second generation rt**: an analysis of new radiation concepts by way of first-second generation conversion. 2019.

HOSTETTER, J.; KHANNA, N.; MANDELL, J. C. Integration of a zero-footprint cloud-based picture archiving and communication system with customizable forms for radiology research and education. **Academic radiology**, v. 25, n. 6, p. 811–818, 2018.

JIN, Z.; CHEN, Y. Telemedicine in the cloud era: prospects and challenges. **IEEE Pervasive Computing**, v. 14, n. 1, p. 54–61, Jan 2015.

LI, W. et al. Sp-miov: a novel framework of shadow proxy based medical image online visualization in computing and storage resource restrained environments. **Future Generation Computer Systems**, v. 105, p. 318–330, 2020.

LIU, B. J.; HUANG, H. Picture archiving and communication systems and electronic medical records for the healthcare enterprise. In: **Biomedical information technology**. Elsevier, 2020. p. 105–164.

LOFF, J.; GARCIA, J. Vadara: predictive elasticity for cloud applications. In: IEEE 6TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE, 2014., 2014, Singapore, Singapore. **Anais...** IEEE, 2014. n. February, p. 541–546.

LUBLINER, D. J. **Biomedical informatics**: an introduction to information systems and software in medicine and health. CRC Press, 2015. 227 p.

MAANI, R.; CAMORLINGA, S.; ARNASON, N. A parallel method to improve medical image transmission. **Journal of digital imaging**, v. 25, n. 1, p. 101–109, 2012.

MARINESCU, D. C. **Cloud Computing Theory and Practice**. Morgan Kaufmann, 2013. 21–65 p.

MCEVOY, F. J.; SVALASTOGA, E. Security of patient and study data associated with dicom images when transferred using compact disc media. **Journal of digital imaging**, v. 22, n. 1, p. 65–70, 2009.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. **NIST**, 2011.

MELL, P. M.; GRANCE, T. **The nist definition of cloud computing**. Gaithersburg, MD: National Institute of Standards and Technology, 2011.

MILANI, A. S.; NAVIMIPOUR, N. J. Load balancing mechanisms and techniques in the cloud environments: systematic literature review and future trends. **Journal of Network and Computer Applications**, v. 71, p. 86–98, 2016.

MILDENBERGER, P.; EICHELBERG, M.; MARTIN, E. Introduction to the dicom standard. **European radiology**, v. 12, n. 4, p. 920–927, 2002.

MOGHADAM, A. et al. Evaluation of pacs system with economic interests approach in 5th azar educational hospital in gorgan. **International Research Journal of Applied and Basic Sciences (IRJABS)**, 2015.

MOHAMAD, B.; D'ORAZIO, L.; GRUENWALD, L. Towards a hybrid row-column database for a cloud-based medical data management system. In: INTERNATIONAL WORKSHOP ON CLOUD INTELLIGENCE, 1., 2012, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2012. p. 4.

MOLTÓ, G. et al. Elastic memory management of virtualized infrastructures for applications with dynamic memory requirements. In: PROCEDIA COMPUTER SCIENCE, 2013, Valencia, Spain. **Anais...** Elsevier, 2013. v. 18, p. 159–168.

MOORE, L. R.; BEAN, K.; ELLAHI, T. Transforming reactive auto-scaling into proactive auto-scaling. In: INTERNATIONAL WORKSHOP ON CLOUD DATA AND PLATFORMS - CLOUDDP '13, 3., 2013, New York, New York, USA. **Proceedings...** ACM Press, 2013. p. 7–12.

MORETTIN, P. A.; TOLOI, C. Análise de séries temporais. In: **Análise de séries temporais**. 2. ed. Edgard Blucher; Associação Brasileira de Estatística, 2006.

MUSTRA, M.; DELAC, K.; GRGIC, M. Overview of the dicom standard. In: INTERNATIONAL SYMPOSIUM ELMAR, 2008., 2008. **Anais...** IEEE, 2008. v. 1, p. 39–44.

MWOGI, T. et al. A scalable low-cost multi-hospital tele-radiology architecture in kenya. **Journal of Health Informatics in Africa**, v. 5, n. 2, 2018.

NEMA, P. Iso 12052. **Digital imaging and communications in medicine (DICOM) standard**, 2014.

NEMA, P. Iso 12052, digital imaging and communications in medicine (dicom) standard. **National Electrical Manufacturers Association**, 2016.

NETO, J. F.; ALCOCER, P. R. Compressão de imagens médicas utilizando a técnica jpeg-dpcm. **IV Fórum Nacional de Ciência e Tecnologia em Saúde**, p. 411–412, 1996.

NETTO, M. A. et al. Evaluating auto-scaling strategies for cloud computing environments. In: IEEE 22ND INTERNATIONAL SYMPOSIUM ON MODELLING, ANALYSIS & SIMULATION OF COMPUTER AND TELECOMMUNICATION SYSTEMS, 2014., 2014. **Anais...** [S.l.: s.n.], 2014. p. 187–196.

NIKRAVESH, A. Y.; AJILA, S. A.; LUNG, C.-H. An autonomic prediction suite for cloud resource provisioning. **Journal of Cloud Computing**, v. 6, n. 1, p. 3, 12 2017.

OLIVEIRA, L. B. et al. Uma abordagem sdn para priorização de tráfego em ambientes hospitalares inteligentes. In: XVIII SIMPóSIO BRASILEIRO DE COMPUTAçãO APLICADA à SAúDE, 2018, Porto Alegre, RS, Brasil. **Anais...** SBC, 2018.

Palma, A. P. et al. Web based picture archiving and communication system for medical images. In: NINTH INTERNATIONAL SYMPOSIUM ON DISTRIBUTED COMPUTING AND APPLICATIONS TO BUSINESS, ENGINEERING AND SCIENCE, 2010., 2010. **Anais...** IEEE, 2010. v. 1, n. 1, p. 141–144.

PAN, T. et al. Whitepapers on imaging infrastructure for research part three: security and privacy. **Journal of digital imaging**, v. 25, n. 6, p. 692–702, 2012.

PARK, E. et al. An adaptive streaming technique for interactive medical systems in mobile environment. In: INTERNATIONAL CONFERENCE ON MOBILE TECHNOLOGY, APPLICATION & SYSTEMS, 6., 2009. **Proceedings...** [S.l.: s.n.], 2009. p. 31.

PIANYKH, O. S. **Digital imaging and communications in medicine (dicom)**: a practical introduction and survival guide. Springer Science & Business Media, 2009.

PINTO, S. J.; GAWANDE, J. P. Performance analysis of medical image compression techniques. In: THIRD ASIAN HIMALAYAS INTERNATIONAL CONFERENCE ON INTERNET, 2012., 2012. **Anais...** IEEE, 2012. p. 1–4.

REACTIONDATA, P. **Medial image-sharing quick report**. White Paper - https://www.reactiondata.com/ - https://reactiondata.com/wp-content/uploads/2015/01/Medical-Image-Sharing-Quick-Report-2015.pdf, Peer60.

RIBEIRO, L. S.; COSTA, C.; OLIVEIRA, J. L. Clustering of distinct pacs archives using a cooperative peer-to-peer network. **Computer methods and programs in biomedicine**, v. 108, n. 3, p. 1002–1011, 2012.

RIGHI, R. D. R. Elasticidade em cloud computing: conceito, estado da arte e novos desafios. **Revista Brasileira de Computação Aplicada**, v. 5, n. 2, p. 2–17, 11 2013.

ROSA, B. A. et al. An Experimental Tool for Elasticity Management through Prediction Mechanisms. In: IEEE/ACM 7TH INTERNATIONAL CONFERENCE ON UTILITY AND CLOUD COMPUTING, 2014., 2014. **Anais...** IEEE, 2014. p. 511–516.

ROSA RIGHI, R. da et al. Towards cloud-based asynchronous elasticity for iterative hpc applications. In: JOURNAL OF PHYSICS: CONFERENCE SERIES, 2015. **Anais...** [S.l.: s.n.], 2015. v. 649, n. 1, p. 012006.

ROSA RIGHI, R. da et al. Autoelastic: automatic resource elasticity for high performance applications in the cloud. **IEEE Transactions on Cloud Computing**, v. 4, n. 1, p. 6–19, Jan 2016.

ROY, N.; DUBEY, A.; GOKHALE, A. Efficient autoscaling in the cloud using predictive models for workload forecasting. In: IEEE 4TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING, CLOUD 2011, 2011., 2011, Washington, DC, USA. **Proceedings...** IEEE, 2011. p. 500–507.

RUI, Y.; HUANG, T. S.; CHANG, S.-F. Image retrieval: current techniques, promising directions, and open issues. **Journal of visual communication and image representation**, v. 10, n. 1, p. 39–62, 1999.

Schelkens, P. et al. Wavelet coding of volumetric medical datasets. **IEEE Transactions on Medical Imaging**, v. 22, n. 3, p. 441–458, March 2003.

SEERAM, E. Picture archiving and communication systems. In: SPRINGER (Ed.). **Digital radiography**. Springer, 2019. p. 139–164.

SHEN, H. et al. Miaps: a web-based system for remotely accessing and presenting medical images. **Computer methods and programs in biomedicine**, v. 113, n. 1, p. 266–283, 2014.

SILVA, J. M. et al. A community-driven validation service for standard medical imaging objects. **Computer Standards & Interfaces**, v. 61, p. 121–128, 2019.

SINGH, A.; JUNEJA, D.; MALHOTRA, M. Autonomous agent based load balancing algorithm in cloud computing. **Procedia Computer Science**, v. 45, p. 832–841, 2015.

SPINNER, S. et al. Runtime Vertical Scaling of Virtualized Applications via Online Model Estimation. In: IEEE EIGHTH INTERNATIONAL CONFERENCE ON SELF-ADAPTIVE AND SELF-ORGANIZING SYSTEMS, 2014., 2014, London, UK. **Anais...** IEEE, 2014. p. 157–166.

SULLIVAN, A. F. . **Medical image sharing and management drives collaborative care**: overcoming fragmentation to create unity. White Paper - http://www.frost.com/ - $https://uk.emc.com/collateral/analyst-reports/4_fs_wp_medical_image_sharing_021012_mc_print.pdf$.

TABATABAEI, M. S.; LANGARIZADEH, M.; TAVAKOL, K. An evaluation protocol for picture archiving and communication system: a systematic review. **Acta Informatica Medica**, v. 25, n. 4, p. 250, 2017.

TENG, D.; KONG, J.; WANG, F. Scalable and flexible management of medical image big data. **Distributed and Parallel Databases**, v. 37, n. 2, p. 235–250, 2019.

TEODORO, G. et al. Load balancing on stateful clustered web servers. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, 15., 2003. **Proceedings…** IEEE, 2003. p. 207–215.

VIDHYA, K.; SHENBAGADEVI, S. Performance analysis of medical image compression. In: INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING SYSTEMS, 2009., 2009. **Anais…** IEEE, 2009. p. 979–983.

XU, M.; TIAN, W.; BUYYA, R. A survey on load balancing algorithms for virtual machines placement in cloud computing. **Concurrency and Computation: Practice and Experience**, v. 29, n. 12, p. e4123, 2017. e4123 cpe.4123.

YAN, L. Dicom standard and its application in pacs system. **Medical Imaging Process & Technology**, v. 1, n. 1, 2018.

ZHANG, X.-Y.; GE, L.; WANG, T.-F. Entropy-based local histogram equalization for medical ultrasound image enhancement. In: INTERNATIONAL CONFERENCE ON BIOINFORMATICS AND BIOMEDICAL ENGINEERING, 2008., 2008. **Anais…** IEEE, 2008. p. 2427–2429.