UMA SOLUÇÃO PARA RENDERIZAÇÃO REMOTA DE IMAGENS MÉDICAS TRIDIMENSIONAIS: ANÁLISE DE DESEMPENHO

Fabiano Papaiz
Professor de Sistemas de Informação do Instituto Federal do Rio Grande do Norte, Campus Natal/Central, Mestre em Sistemas e Computação pela Universidade Federal do Rio Grande do Norte. E-mail: fabiano.papaiz@ifrn.edu.br

Bruno Motta de Carvalho
Professor do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte, Doutor em *Computer and Information Science* pela *University of Pennsylvania*. E-mail: motta@dimap.ufrn.br.

Ricardo Alexandro de Medeiros Valentim
Professor do Departamento de Engenharia Biomédica da Universidade Federal do Rio Grande do Norte, Doutor em Engenharia Elétrica e de Computação, Coordenador do Laboratório de Inovação Tecnológica em Saúde do Hospital Universitário Onofre Lopes. E-mail: ricardo.valentim@ufrnet.br.

Antônio Higor Freire de Morais
Professor de Sistemas de Informação do Instituto Federal do Rio Grande do Norte, Campus Mossoró, Doutorando e Mestre em Engenharia Elétrica e de Computação pela Universidade Federal do Rio Grande do Norte. E-mail: higor.morais@ifrn.edu.br.

Robinson Luis de Souza Alves
Professor de Sistemas de Informação do Instituto Federal do Rio Grande do Norte, Campus Natal/Central, Doutor em Engenharia Elétrica e de Computação pela Universidade Federal do Rio Grande do Norte. E-mail: robinson.alves@ig.com.br.

RESUMO
Este trabalho propõe e analisa uma solução para renderização remota de imagens médicas tridimensionais (3D). Nesta solução, todo o processo de renderização volumétrica é realizado por um servidor (ou um *cluster* de servidores) e somente a imagem resultante é enviada para o cliente, permitindo ainda que este realize operações como rotação, deslocamento e zoom. Para a realização da análise de desempenho, foram definidos três diferentes cenários de execução. No primeiro cenário, o processo foi realizado por um servidor com *hardware* gráfico (GPU). No segundo, foi utilizado um servidor sem GPU, mas com alto poder de processamento em paralelo, possuindo 48 *cores* (unidades de processamento). O terceiro cenário foi utilizado para simular o funcionamento padrão da maioria das aplicações médicas de visualização, onde todo o processo de renderização foi realizado localmente em um computador. Com os resultados obtidos, ficou demonstrado que a solução proposta resolveu satisfatoriamente dois dos principais problemas da visualização tridimensional dos exames médicos, sendo eles: o tempo gasto com a transferência dos arquivos DICOM e o poder computacional necessário para realizar o processo de renderização. A arquitetura da solução permitiu que computadores menos potentes e dispositivos móveis, como *tablets* e celulares, pudessem ser utilizados para visualização das imagens médicas 3D.

PALAVRAS-CHAVE: Visualização volumétrica, renderização remota, serviços web, imagens médicas, telemedicina.

A SOLUTION FOR REMOTE RENDERING OF THREE-DIMENSIONAL MEDICAL IMAGES: PERFORMANCE ANALYSIS

ABSTRACT

This paper proposes and analyzes a solution for remote rendering of three-dimensional medical images (3D). In this solution, the entire process of volume rendering is performed by a server (or server cluster) and only the resulting picture is sent to the customer, yet allowing it to perform operations such as rotation, pan and zoom. To perform the analysis of performance, three different execution scenarios were defined. In the first scenario, the process was performed by a server with graphics hardware (GPU). In the second, we used a server without GPU, but with high processing power in parallel, having 48 cores (processing units). The third scenario was used to simulate the default behavior of most medical visualization applications, where the entire rendering process was performed locally on a computer. With the results obtained, it has been shown that the proposed solution solved satisfactorily two of the main problems of three-dimensional visualization of medical examinations, being they: the time spent for the transfer of DICOM files and the computational power required to perform the rendering process. The architecture of solution allows less powerful computers and mobile devices (tablets and mobile phones) being used for visualization of tomography and magnetic resonance images.

KEYWORDS: volumetric visualization, remote rendering, web services, medical imaging, telemedicine.

UMA SOLUÇÃO PARA RENDERIZAÇÃO REMOTA DE IMAGENS MÉDICAS TRIDIMENSIONAIS: ANÁLISE DE DESEMPENHO

INTRODUÇÃO

Na área da medicina, é clara a importância das tecnologias que geram imagens digitais dos pacientes. Exames como Ultrassom (US), Tomografia Computadorizada (CT) e Ressonância Magnética (MR) são comumente realizados no dia-a-dia dos hospitais e suas imagens auxiliam os médicos no diagnóstico de uma doença. Nos hospitais, estas imagens são geralmente guardadas e acessadas através de um Sistema de Comunicação e Arquivamento de Imagens (PACS - *Picture Archiving and Communication System*). Os médicos utilizam estações de trabalho ligadas à rede do hospital para acessar e visualizar tais imagens através de aplicações médicas. Hoje em dia, a maioria dos PACS seguem um padrão mundial denominado DICOM (*Digital Imaging and Communications in Medicine*). O objetivo principal do DICOM é possibilitar e facilitar a interoperabilidade entre computadores, equipamentos e aplicações da área médica. Para isso, o padrão especifica protocolos de redes, estruturas de diretórios, formatos de arquivos e comandos para troca de dados, entre outros itens. Diversas API's (*Application Programming Interface*) estão disponíveis em várias linguagens e podem ser utilizadas pelos desenvolvedores para a criação de aplicações médicas em conformidade com o padrão DICOM [39] [34].

Alguns exames, como CT e MR, geram uma sequência ordenada de imagens bidimensionais (2D) do corpo do paciente, também conhecidas como cortes, fatias ou

slices. Através de algoritmos de renderização de dados volumétricos, tais fatias podem ser agrupadas e reconstruídas de forma a gerar uma projeção 2D de uma imagem tridimensional (3D) do paciente, facilitando assim o trabalho de interpretação por parte do médico. Um exemplo pode ser visto na Figura 1.

Com o avanço tecnológico dos equipamentos de saúde, estão sendo geradas imagens cada vez mais nítidas, com maiores resoluções e, com isso, arquivos com tamanhos cada vez maiores. Isso faz com que o processo de visualização das imagens 3D se torne pesado e necessite de um poder computacional maior para ser realizado. Assim a realização dessa tarefa por dispositivos móveis disponíveis no mercado, como os *tablets* e os celulares, requer destes uma maior capacidade de processamento. Além disso, para que a aplicação possa gerar a imagem 3D é necessário que todas as fatias sejam enviadas para o dispositivo onde ela está sendo executada, o que pode tornar essa tarefa complicada, caso esteja sendo utilizada uma conexão via *internet*, pois uma grande quantidade de dados precisa ser transferida antes que o processamento possa ser iniciado. Atualmente, o conjunto de fatias de um exame tomográfico chega ao tamanho de dezenas ou centenas de megabytes. Por exemplo, um exame tomográfico que gere 200 fatias com resolução de 512x512 *pixels* por imagem, sem compressão, com profundidade de cor de 16 bits (2 *bytes* para cada *pixel*), irá ocupar um tamanho aproximado de 100 *megabytes*.

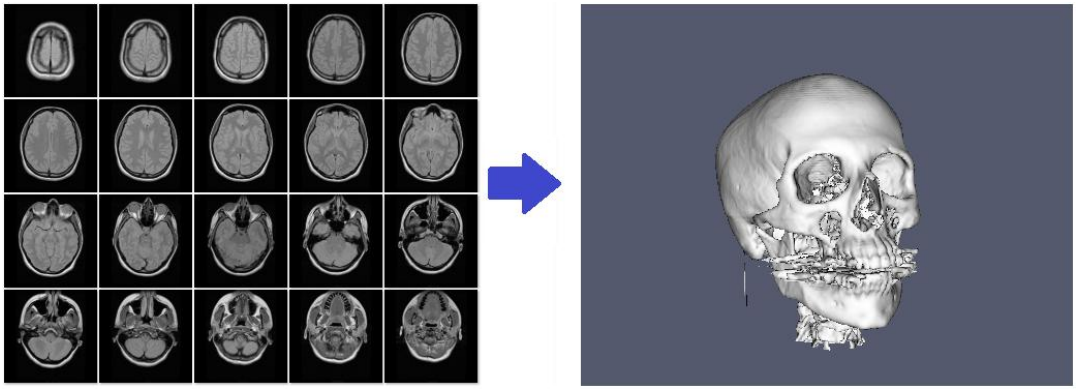

Figura 1 – Exemplo de reconstrução 3D a partir de imagens 2D. No lado esquerdo podemos observar as imagens 2D, a partir das quais foi gerada a renderização da projeção 2D da imagem 3D (imagem do lado direito).

Podemos dizer que o grande tamanho do conjunto de dados é um dos principais problemas enfrentados pela área da Visualização Volumétrica na Medicina. Uma solução para este problema está na renderização remota dos dados volumétricos, onde o processo de renderização é realizado por um servidor (ou *cluster* de servidores) com grande poder computacional, podendo inclusive possuir *hardware* gráfico (GPU), e somente a imagem resultante é enviada para o cliente. Isto faz com que computadores ou dispositivos móveis que possuam baixo poder computacional possam ser utilizados para visualizar dados volumétricos de forma interativa e satisfatória para o usuário final.

Este trabalho teve como objetivo geral o desenvolvimento e, principalmente, a avaliação de desempenho de uma solução que possibilitasse a renderização remota de imagens médicas 3D, sendo possível a utilização de computadores menos potentes e até dispositivos móveis (*tablets* e celulares) para acessar, visualizar e interagir com as imagens. A solução proposta ainda possui a característica de tirar proveito do processamento em paralelo, podendo ser implantada em ambientes de computação distribuída, como um *cluster* de servidores. O

problema com o tamanho do conjunto de dados foi solucionado através da definição de uma arquitetura em camadas. Outra característica importante é que esta solução pode ser utilizada a partir de qualquer aplicação médica em conformidade com o padrão DICOM, desde que esta consiga trocar informações com serviços *web*.

Alguns trabalhos relacionados, como em [45], [25] e [23], diferem da solução proposta principalmente pelo fato de que estes necessitam instalar ou baixar arquivos específicos no dispositivo do usuário para poderem funcionar, enquanto que na solução apresentada o usuário irá necessitar apenas de um navegador *web*. Em outros trabalhos, como em [1] e [3], são apresentadas soluções na área da telemedicina que estão voltadas para auxiliar os médicos durante suas visitas aos leitos dos pacientes, onde estes têm a possibilidade de consultar os dados dos pacientes, inclusive imagens médicas, mas não tratam especificamente da visualização remota de imagens médicas tridimensionais.

MATERIAIS E MÉTODOS

A arquitetura de referência da solução proposta neste trabalho foi dividida em três camadas: Cliente, Renderização e PACS. Uma visão geral da arquitetura pode ser vista na Figura 2. Na Camada Cliente o usuário precisa possuir apenas um navegador *web* para acessar a aplicação de testes, selecionar os exames médicos desejados e visualizar as imagens em 3D. Na Camada PACS são armazenados os arquivos DICOM contendo as imagens, ficando disponíveis para serem acessados pela Camada de Renderização. Na Camada de Renderização, são hospedados os serviços *web* de Aquisição e de Renderização, sendo esta camada a responsável por realizar o processo de aquisição dos arquivos DICOM e de renderização remota das imagens 3D, disponibilizando-as para o usuário final.

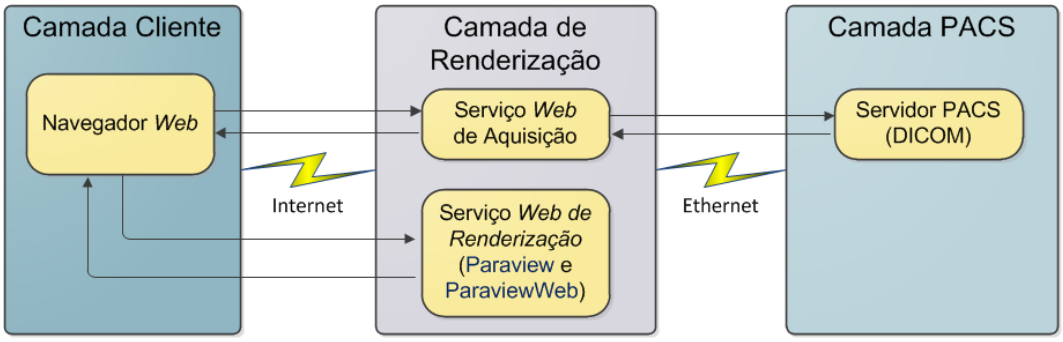

Figura 2 – Arquitetura de referência da solução proposta.

Componentes da Camada PACS

Nesta camada, foi utilizado um servidor com processador Intel Xeon X5650 (2,67 GHz), com 16 GB de memória RAM e sistema operacional Ubuntu 12.04. Neste servidor foi instalado o programa dcm4chee [7], versão 1.4, o qual é um servidor PACS gratuito, em conformidade com o padrão DICOM e *open source*. No arquivo de distribuição do dcm4chee já está incluído o servidor de aplicações JBoss [15], necessário para a sua execução. O dcm4chee pode trabalhar em conjunto com os mais populares Sistemas Gerenciadores de Bancos de Dados, sendo que neste trabalho foi utilizado o MySQL 5.6 [12] pelo fato de ser gratuito e por já termos experiência com a utilização deste banco de dados em trabalhos anteriores.

Para compor a base de dados de testes, foram selecionados oito exames de tomografia computadorizada (CT) de fraturas ósseas, os quais estão detalhados na Tabela 1. Tais exames possuem diferentes quantidades de fatias axiais e são de regiões variadas do corpo humano, sendo a resolução de cada fatia de 512x512 *pixels*.

Tabela 1: Detalhes dos exames (arquivos DICOM) selecionados para os testes.

Exame	Região do Corpo	Fatias	Tamanho (MB)
1	Pé	259	138,8
2	Perna	540	289,1
3	Face	427	228,6
4	Cotovelo	394	211,0
5	Coluna	505	270,4
6	Crânio	170	91,0
7	Punho	474	253,8
8	Crânio	167	89,4

Componentes da Camada de Renderização

A solução proposta foi testada em duas configurações de servidores, sendo um servidor com *hardware* gráfico (GPU) e outro sem GPU, mas com vários processadores. Também foi utilizado nos testes um computador do tipo *notebook*. Os detalhes destes três computadores estão descritos na Tabela 2. No servidor com GPU, o processo de renderização foi realizado pelo *hardware* gráfico. No servidor sem GPU, foi utilizada computação paralela para a renderização das imagens. No *notebook*, as Camadas Cliente e de Renderização foram unificadas para serem executadas localmente, sendo que a renderização foi realizada sem utilização de GPU ou de computação paralela. Nestes três computadores foram utilizados o sistema operacional Ubuntu versão 12.04 e os pacotes de *softwares* utilizados, com suas respectivas versões, estão descritos na Tabela 3.

No servidor sem GPU e no *notebook*, o *software* Paraview [31], responsável pela realização da renderização remota, precisou ser compilado com a opção de suporte à biblioteca MESA-3D [29], a qual é uma implementação *open source* da especificação OpenGL [14]. A biblioteca MESA-3D permite que o processo de renderização seja feito apenas pela CPU, fazendo uma emulação do *hardware* gráfico. No servidor sem GPU também foi necessário instalar o *software* MPICH [33], versão 1.2.7, o qual é uma implementação *open source* do padrão MPI (*Message Passing Interface*) [32] para computação paralela. O *software* ParaviewWeb [18], precisou ser compilado nos 3 computadores antes de poder ser configurado e executado.

Inserção e Disponibilização dos Exames Médicos

Os exames de CT utilizados nos experimentos foram cedidos pelo Serviço de Apoio ao Diagnóstico e Tratamento (SADT) do Hospital Monsenhor Walfredo Gurgel, localizado no município de Natal/RN. A fim de preservar a identidade dos pacientes, algumas das informações originais dos exames foram substituídas por valores aleatórios, como, por exemplo, o nome do paciente. Estes exames foram inseridos na base de dados do servidor

Tabela 2: Configuração dos computadores utilizados nos testes.

Item de Configuração	Servidor com GPU	Servidor sem GPU	Notebook
Memória RAM	12 GB	128 GB	8GB
Processador	Intel Core i7 950	AMD Opteron 6176	Intel Core i5 2410
Clock	3,06 GHz	2,3 GHz	2,3 GHz
Qtd. Processadores	2	4	1
Total de *Cores*	8	48	4
Cores Utilizados	1	8, 16 e 24	1
GPU	GeForce GTX 580	-	-
Qtd. GPU's	2	-	-
GPU's Utilizadas	1	-	-

Tabela 3: Pacotes de *softwares* utilizados nos computadores utilizados nos testes.

Pacote de Software	Versão	Descrição
Apache Tomcat	6.0	Servidor *web* para aplicações Java
Apache AXIS	2.0	*Software* para hospedar serviços *web* que seguem a especificação SOAP/WSDL
Paraview	3.14	*Software* para visualização de dados científicos 2D e 3D. Permite a renderização remota e pode ser configurado para utilizar computação paralela
ParaviewWeb	3.14	*Framework* utilizado em conjunto com o Paraview para possibilitar a renderização remota através de aplicações *web*
Apache ActiveMQ	5.7	*Software* para troca assíncrona de mensagens requerido pelo ParaviewWeb
DCM4CHE	2.0.26	API Java, que implementa o padrão DICOM, utilizada para o desenvolvimento de aplicações médicas
VTK	5.8	API's para processamento, classificação e visualização de imagens digitais
ITK	3.20	

PACS através de um programa utilitário chamado dcmsnd, fornecido pelo dcm4chee. Para disponibilizar o acesso às imagens para a Camada de Renderização, foi configurado no dcm4chee um *Application Entity Title* (AET) para o serviço *web* de aquisição, sendo este um passo requerido pelo padrão DICOM para permitir que as aplicações possam trocar informações com um servidor PACS.

Aplicação *Web* Cliente Para Teste da Solução

Para a realização dos testes foi desenvolvida uma aplicação *web* em Java, onde são listados os exames de CT disponíveis, permitindo ao usuário selecioná-los para realizar a visualização 3D. Na Figura 3 é exibida a tela principal desta aplicação. Além dos exames, o usuário também pode selecionar em qual cenário (GPU, Blade ou Cliente) deseja executar a renderização. Esta aplicação ficou hospedada no servidor *web* Apache Tomcat da Camada de Renderização. A Figura 4 mostra a imagem 3D sendo exibida na Página de Visualização, onde o usuário pode realizar operações na imagem, como rotação, deslocamento ou zoom. O usuário pode ainda alterar o *threshold* de visualização, sendo possível visualizar objetos diferentes de um mesmo exame, como tecidos, órgãos ou ossos.

Serviços *Web* da Camada de Renderização

Na Camada de Renderização, foram desenvolvidos dois serviços *web*: Aquisição e Renderização. Eles foram desenvolvidos utilizando-se tecnologia Java e ficaram disponíveis para serem consumidos pela aplicação *web* de testes através do *software* Apache AXIS2. A seguir estão descritas as funcionalidades principais de cada serviço dentro da solução proposta.

Figura 3 – Página inicial da aplicação *web* utilizada nos testes.

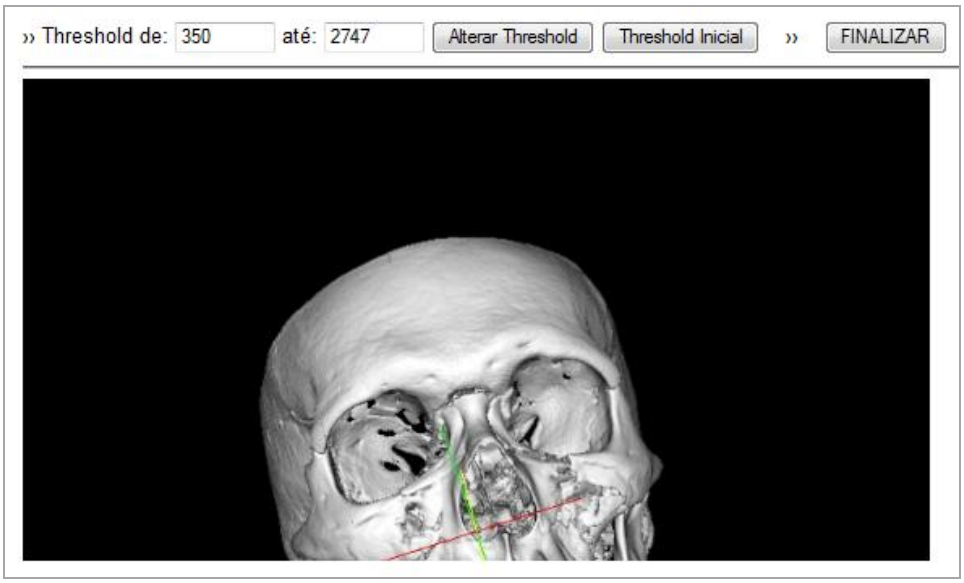

Figura 4 – Página de visualização da imagem 3D.

O serviço *web* de Aquisição é o responsável por:

- Receber da aplicação cliente o identificador do exame (UID) que o usuário deseja visualizar;
- Solicitar ao servidor PACS os arquivos DICOM contendo as imagens do exame selecionado;

- Processar as imagens recebidas para gerar um arquivo onde são armazenadas as informações sobre os dados volumétricos (topologia, geometria, valores dos *voxels* etc). Este arquivo é gerado no formato nativo do Paraview, com extensão .vtk;
- Enviar como resposta para a aplicação cliente o nome do arquivo gerado e a URL da página que deverá ser acessada para a visualização 3D.

O serviço *web* de Renderização, através da utilização do programa Paraview e do ParaviewWeb, é o responsável por:

- Acessar o arquivo que contem os dados volumétricos (gerado pelo serviço de Aquisição);
- Realizar o processamento necessário para a visualização volumétrica, aplicando as funções de transferência de cores e de opacidade;
- Retornar a imagem 3D resultante para o cliente;
- Processar remotamente as operações feitas pelo usuário na imagem 3D, como rotação e zoom, enviando novamente a imagem 3D resultante.

Cenários Para Teste e Avaliação da Arquitetura

Três cenários distintos (GPU, Blade e Cliente) foram definidos para testar e avaliar o desempenho da solução proposta, conforme pode ser visto na Figura 5. Tais cenários foram criados para representarem o processo de renderização de imagens 3D de diferentes formas.

No Cenário GPU, o processo de renderização foi realizado pelo *hardware* gráfico do servidor. Para obtenção das imagens médicas, a comunicação entre o serviço *web* de Aquisição e o servidor PACS foi realizada via protocolo DICOM. A conexão de rede utilizada foi do tipo *Gigabit Ethernet*, diminuindo assim, o tempo necessário para a transferência dos arquivos DICOM contendo as imagens. Para a visualização das imagens médicas 3D, o usuário acessou a aplicação *web* cliente através de uma conexão via *internet*, usando protocolo TCP/IP.

No Cenário Blade, o processo de renderização foi realizado através de computação paralela, onde vários *cores* dos processadores foram utilizados simultaneamente para realizar o processamento. Durante os testes, o programa Paraview foi executado utilizando-se 8, 16 e 24 *cores* em paralelo, fazendo com que este cenário se dividisse em 3 subcenários, denominados *Blade(8)*, *Blade(16)* e *Blade(24)*. A obtenção das imagens e a utilização pelo usuário foram realizadas da mesma forma como foi descrito anteriormente para o Cenário GPU.

No Cenário Cliente, todo o processamento foi realizado localmente no computador do usuário. A conexão com o servidor PACS foi feita via conexão *internet* (protocolo TCP/IP). Este cenário não realizou a renderização de forma remota e foi criado com fins a avaliar o funcionamento padrão das aplicações na área da saúde de visualização de imagens. Nestas aplicações, todos os arquivos DICOM precisam ser transferidos para o computador do usuário antes de realizar (localmente) o processo de renderização da imagem 3D. Neste cenário, a Camada de Renderização foi mesclada com a Camada Cliente, ficando com uma arquitetura de apenas duas camadas. Os *softwares* da Camada de Renderização foram todos instalados e configurados no computador do usuário.

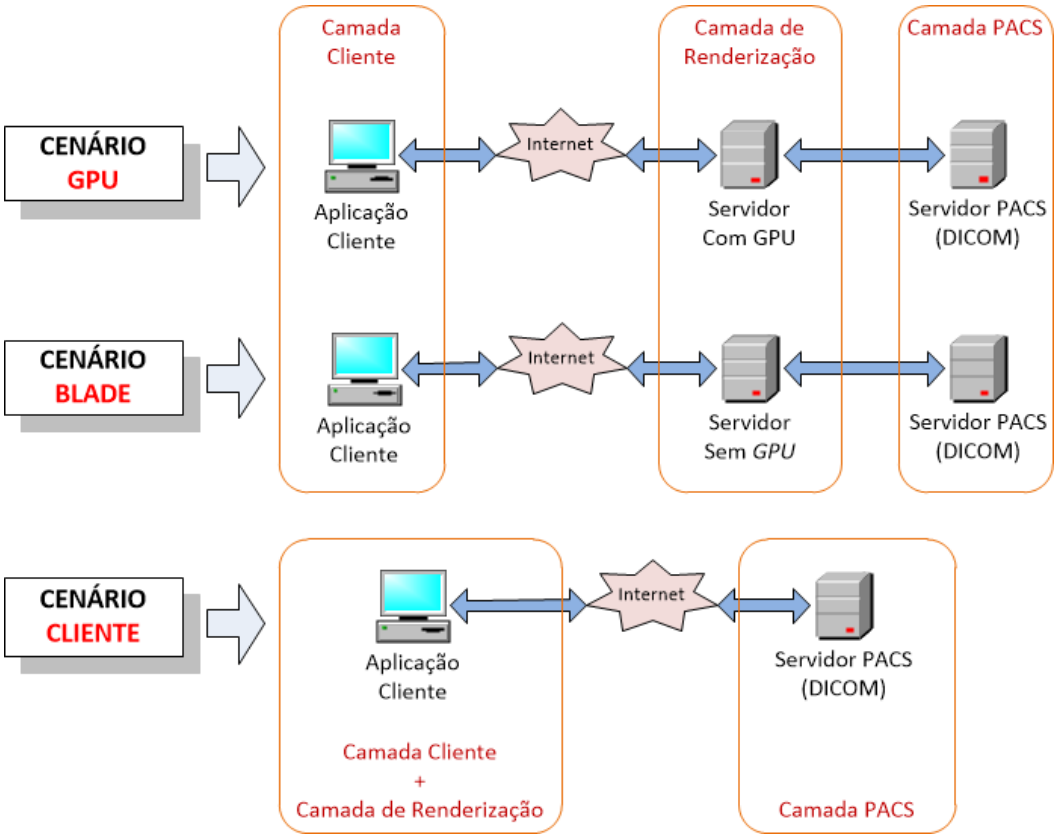

Figura 5 – Cenários para a execução e avaliação da solução proposta.

Velocidades de Conexão Utilizadas nos Cenários

Para obter uma análise de desempenho mais detalhada, foram realizados testes com diferentes velocidades de conexão de rede. As velocidades foram controladas através do pacote wondershaper [46]. Este pacote instala um *script shell* que utiliza o programa utilitário do Linux chamado *Traffic Control*, também conhecido como tc. Com este *script* é possível definir a velocidade máxima em *kilobits* que se deseja atribuir para a uma interface de rede de um computador. Por exemplo, o comando *wondershaper eth0 512 512* define uma velocidade máxima de download e upload de 512kb/s para a interface eth0. O controle da velocidade foi realizado executando o wondershaper nos computadores da Camada Cliente utilizados nos experimentos.

Nos Cenários Blade e GPU, foram utilizadas as velocidades de 256kb/s, 512kb/s, 1Mb/s e 5Mb/s para a conexão entre a Camada Cliente e a Camada de Renderização. Tais velocidades foram escolhidas com base nas velocidades comumente oferecidas aos clientes pelas empresas operadoras de internet à época deste trabalho. No Cenário Cliente, estas duas camadas situam-se no mesmo local e, portanto, o controle de velocidade não foi necessário.

Para a conexão com a Camada PACS, no Cenário Cliente foram realizados testes com as velocidades de 1, 3, 5, 7, 10 e 100Mb/s. As velocidades de 1 a 10Mb/s foram inseridas para determinar a partir de que ponto a velocidade de conexão deixa de afetar consideravelmente o tempo de transferência dos arquivos DICOM. A velocidade de 100Mb/s foi inserida porque era a velocidade de conexão mais comumente encontrada nas redes locais de hospitais e instituições de ensino do RN. Nos Cenários Blade e GPU, a

comunicação entre a Camada de Renderização e a Camada PACS foi através de uma conexão do tipo *Gigabit Ethernet*, com velocidade de 1 *Gigabit* por segundo.

Utilização de Dispositivos Móveis para a Visualização

A solução foi criada visando possibilitar que dispositivos com menor poder computacional pudessem ser utilizados para visualizar as imagens 3D, pois todo o processamento fica a cargo do servidor da Camada de Renderização. Para comprovar essa característica da solução, dois testes adicionais foram realizados. A solução foi acessada a partir de um dispositivo *tablet*, modelo IPad da Apple, e um celular, modelo WT19a da Sony Ericson.

Definição e Registro dos Indicadores de Desempenho

Para a análise de desempenho da solução nos três cenários, foram definidos os seguintes indicadores:

- Tempo de Transferência (TT): representa o tempo para transferir os arquivos DICOM das imagens da Camada PACS para a Camada de Renderização;
- Tempo de Geração (TG): representa o tempo para processar as imagens médicas recebidas e gerar um arquivo onde serão armazenadas as informações sobre os dados volumétricos (arquivo com extensão vtk, no formato nativo do Paraview);
- Tempo de Visualização (TV): representa o tempo decorrido desde a seleção do exame na aplicação *web* cliente até que a primeira imagem 3D fosse visualizada pelo usuário;
- Tempo de Interação (TI): representa o tempo que o usuário precisou para posicionar a imagem 3D através de operações de rotação, deslocamento e zoom, simulando a situação onde a imagem precisasse ser impressa e inserida em um laudo.

A Figura 6 demonstra onde se inicia e finaliza a medição cada indicador dentro da arquitetura da solução. Para medir os tempos TT e TG foram inseridos marcadores no código fonte do serviço *web* de Aquisição. O tempo TV foi medido com marcadores inseridos na Página de Visualização da aplicação *web* cliente. O tempo TI também foi medido pela aplicação *web* cliente, tendo início logo após a medição de TV e finalizado no momento em que o usuário, após interagir com a imagem, clicou no botão "Finalizar" da Página de Visualização. Todos os tempos registrados foram armazenados em uma tabela de um banco de dados criado especialmente para este fim, a qual contém os seguintes campos: Exame, Cenário, TT, TG, TV, TI e Velocidade da Conexão.

Para a automatização dos testes foi utilizada a ferramenta Selenium IDE [42], a qual permite gravar todas as ações realizadas por um usuário em uma aplicação *web* (cliques em *links*, botões etc), para então repetí-las novamente de forma automática e por quantas vezes for desejado. A utilização da aplicação *web* cliente foi gravada de forma que todos os 8 exames foram visualizados. Em seguida, a ferramenta Selenium IDE foi configurada para repetir essa utilização por 20 vezes, ou seja, ao final da execução cada exame foi visualizado 20 vezes, gerando um total de 160 amostras. Essa quantidade de amostras (20 por exame) foi considerada por nós como sendo suficiente para a analisar o desempenho da solução. A execução foi realizada para todas as combinações de cenários e velocidades de conexão definidos. Os testes nos Cenários Blade e GPU foram realizados utilizando-se 3 computadores que, simultaneamente, executaram a aplicação *web* cliente.

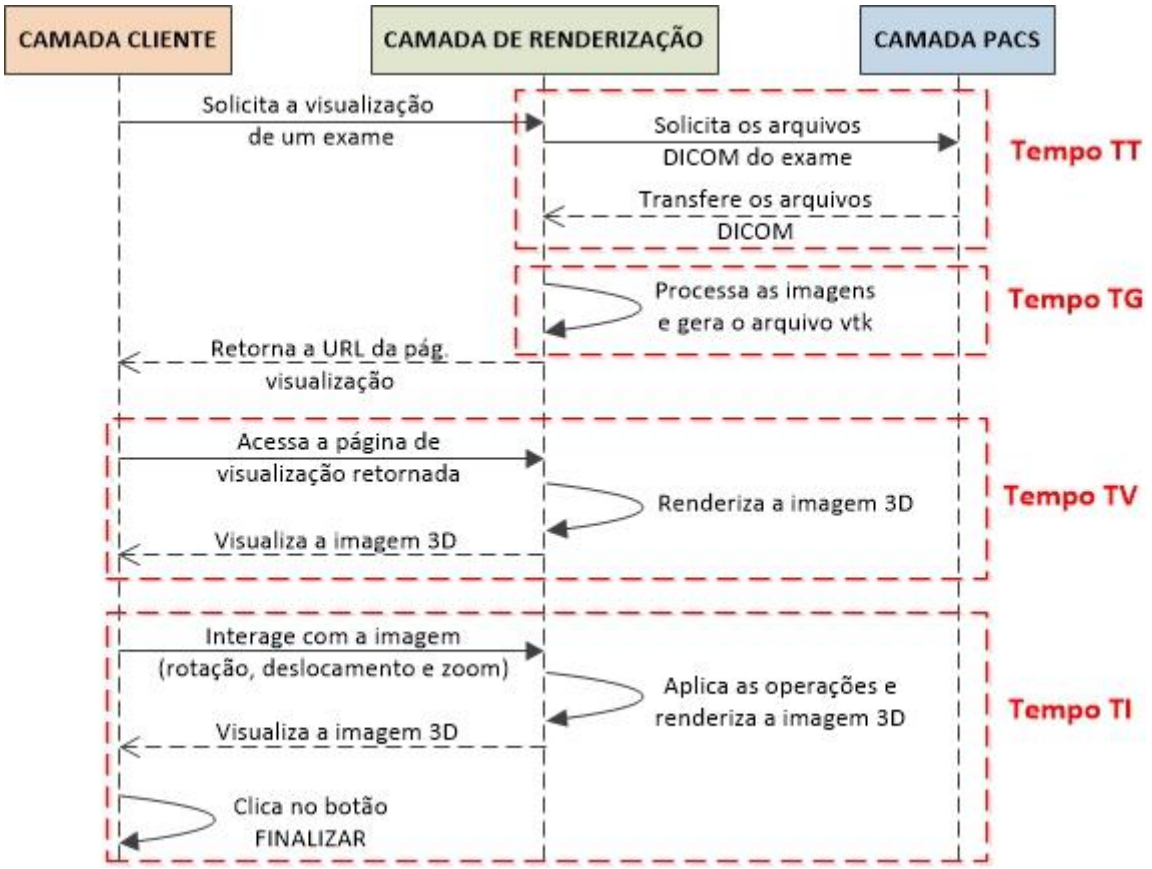

Figura 6 – Locais onde se iniciam e finalizam as medições dos indicadores de tempo dentro da arquitetura da solução.

Para para registrar o tempo de TI, foram selecionados os 2 exames com maior quantidade de fatias, sendo eles o Exame 2 (540 fatias) e o Exame 5 (505 fatias). Estes exames foram escolhidos por serem os maiores em tamanho do arquivo DICOM e, portanto, precisarem de mais recursos computacionais para serem renderizados. Os experimentos de interação foram realizados por 5 usuários (alunos dos cursos de graduação e pós-graduação do DIMAp da UFRN) com familiaridade na utilização de computadores, mas sem familiaridade com aplicações médicas de visualização tridimensional. Cada usuário visualizou e interagiu com os 2 exames em todos os cenários e em todas as velocidades de conexão. Estes usuários foram instruídos a realizar as seguintes atividades durante cada visualização:

1. Selecionar o exame a ser visualizado;
2. Alterar o valor do *threshold* para a faixa de 250 a 2000 para que fossem visualizadas as partes ósseas do paciente. O valor padrão definido foi de -200 a 0, onde o usuário visualizava inicialmente somente os tecidos e os órgãos;
3. Rotacionar, deslocar e fazer ampliações/reduções (zoom) na imagem para posicioná-la de acordo com o guia de imagens que foi entregue (ver Figura 7);
4. Concluir a interação através da opção “Finalizar” da Página de Visualização.

Após a conclusão dos testes, cada usuário foi entrevistado sobre qual cenário ele teve a melhor e a pior experiência de utilização, com base nos tempos de resposta da aplicação e na facilidade de interação com a imagem 3D. E ainda, qual havia sido a operação mais difícil de realizar com as imagens em todos os cenários (rotação, deslocamento ou zoom).
Durante os testes, foram realizadas 4.480 visualizações de exames na aplicação *web* cliente para registrar os tempos TT, TG e TV, sendo todas estas automatizadas através da

ferramenta Selenium IDE. Já para registrar o tempo TI, os usuários realizaram um total de 130 visualizações e posicionamentos das imagens 3D.

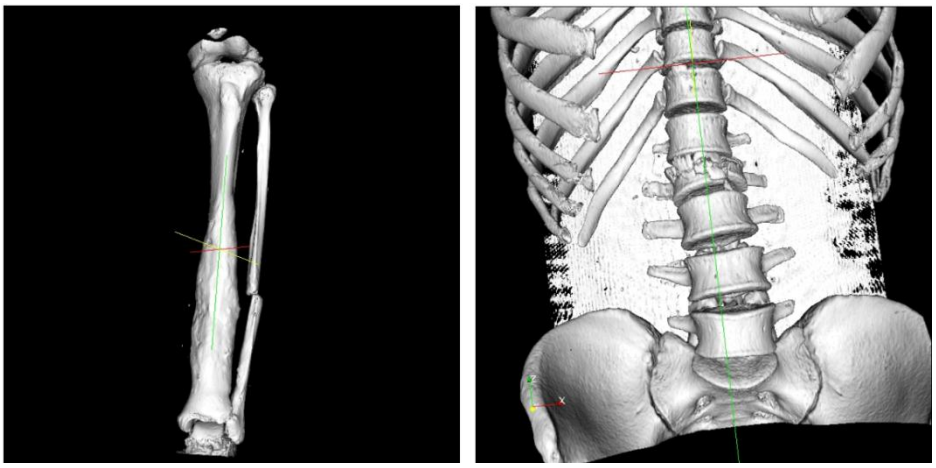

Figura 7 – Guia das imagens entregue aos usuários durante os experimentos para medição do Tempo de Interação (TI). As imagens são referentes aos Exames 2 e 5.

RESULTADOS

A seguir são listados todos os resultados obtidos nos testes para cada combinação de cenário e velocidade de conexão. Todos os indicadores (tempos TT, TG, TV e TI) são exibidos individualmente e, ao final, de forma conjunta para ser possível observar a fração que cada um ocupou no tempo total de processamento.

Tempo de Transferência

A média dos resultados obtidos para o tempo de transferência (TT) dos arquivos DICOM estão exibidos, na forma de gráfico, na Figura 9. A Tabela 4 lista, além da média, o desvio padrão e os valores máximo e mínimo obtidos. Os valores estão expressos em segundos e, para algumas velocidades do Cenário Cliente, os valores também foram expressos em minutos.

Tempo de Geração

O processo de geração do arquivo vtk foi executado utilizando-se apenas 1 *core* em todos os cenários, portanto, não foi executado em paralelo no Cenário Blade e nem pelo *hardware* gráfico no Cenário GPU. As médias dos tempos obtidos estão exibidas na Figura 8 na forma de gráfico. Na Tabela 5, além da média, estão listados o desvio padrão e os valores máximo e mínimo obtidos (valores expressos em segundos).

Tempo de Visualização

As médias dos resultados obtidos estão exibidas na Figura 10 na forma de gráfico e com informações adicionais na Tabela 6 (valores estão expressos em segundos). Para o Cenário Blade, os números que aparecem entres parêntesis indicam a quantidade de *cores* utilizados pelo programa Paraview para realizar a renderização, sendo que nos testes foram utilizados 8, 16 e 24 *cores*.

Tempo de Interação

Os experimentos para medição do tempo TI nos Cenários Blade (8, 16 e 24 *cores*) e GPU foram realizados com as velocidades de 512kb/s, 1Mb/s e 5Mb/s. No Cenário Cliente, a interação foi realizada localmente. As médias dos resultados obtidos estão exibidas no gráfico da Figura 11 e com informações adicionais na Tabela 7, sendo que os valores estão expressos em segundos.

Tabela 4: Valores da média, desvio padrão, máximo e mínimo obtidos para o Tempo de Transferência dos arquivos DICOM.

Cenário	Velocidade	Tempo de Transferência dos Arquivos DICOM			
		Média	Desvio Padrão	Máximo	Mínimo
Blade	*Gigabit*	4,638	1,907	11,893	1,449
GPU	*Gigabit*	4,594	1,904	12,890	1,571
Cliente	1Mb/s	1498,294 (24,971 min.)	791,050	2289,344	707,244
	3Mb/s	401,763 (6,696 min.)	211,968	614,012	189,665
	5Mb/s	181,837 (3,030 min.)	95,888	277,791	85,842
	7Mb/s	90,991 (1,52 min.)	38,048	134,177	41,372
	10Mb/s	16,882	7,004	24,801	7,713
	100Mb/s	16,389	6,090	24,351	6,994

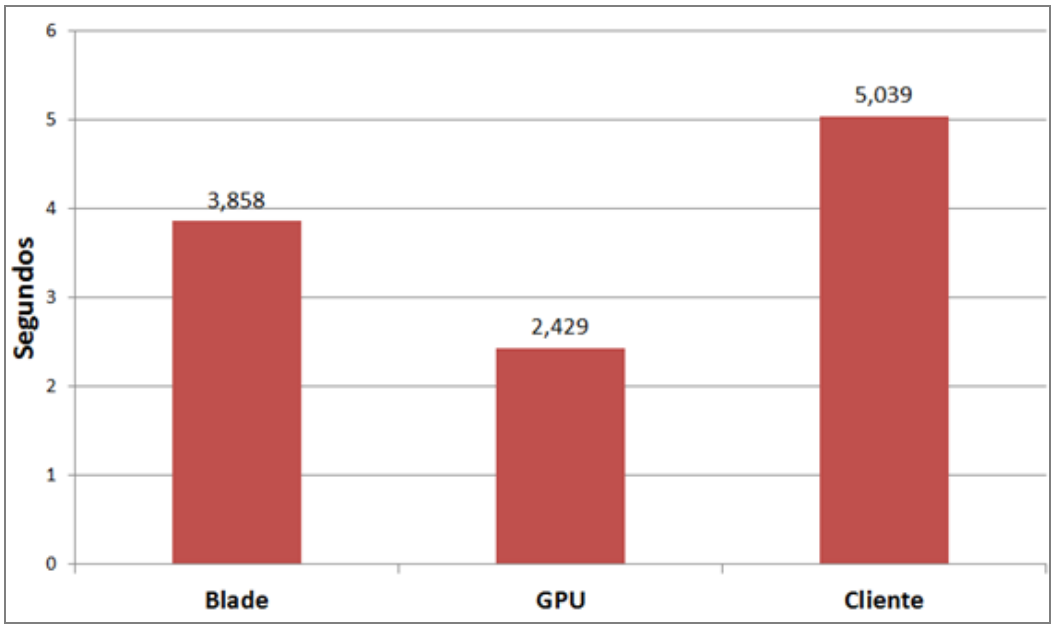

Figura 8 – Média do Tempo de Geração do arquivo vtk.

Tabela 5: Valores da média, desvio padrão, máximo e mínimo obtidos para o Tempo de Geração do arquivo vtk.

Cenário	*Cores*	Velocidade	Tempo de Geração do Arquivo vtk			
			Média	Desvio Padrão	Máximo	Mínimo
Blade	1	Não se aplica	3,858	1,241	6,058	1,597
GPU	1	Não se aplica	2,429	1,092	9,893	0,798
Cliente	1	Não se aplica	5,039	4,686	19,418	0,668

Junção dos Resultados Obtidos

Aqui foi realizada a junção dos tempos obtidos nos testes para analisar a fração que cada um ocupou no processo como um todo. A Figura 12 contém as informações dos Cenários Blade (8, 16 e 24 *cores*) e GPU para as velocidades de 512kb/s, 1Mb/s e 5Mb/s, onde os valores estão expressos em segundos. Na Figura 13 estão as informações do Cenário Cliente para as velocidades de 3, 5, 7 e 10Mb/s, sendo também inseridos os tempos do Cenário GPU com velocidade de 5Mb/s, que foram os melhores obtidos nos testes. Estes valores estão expressos em minutos.

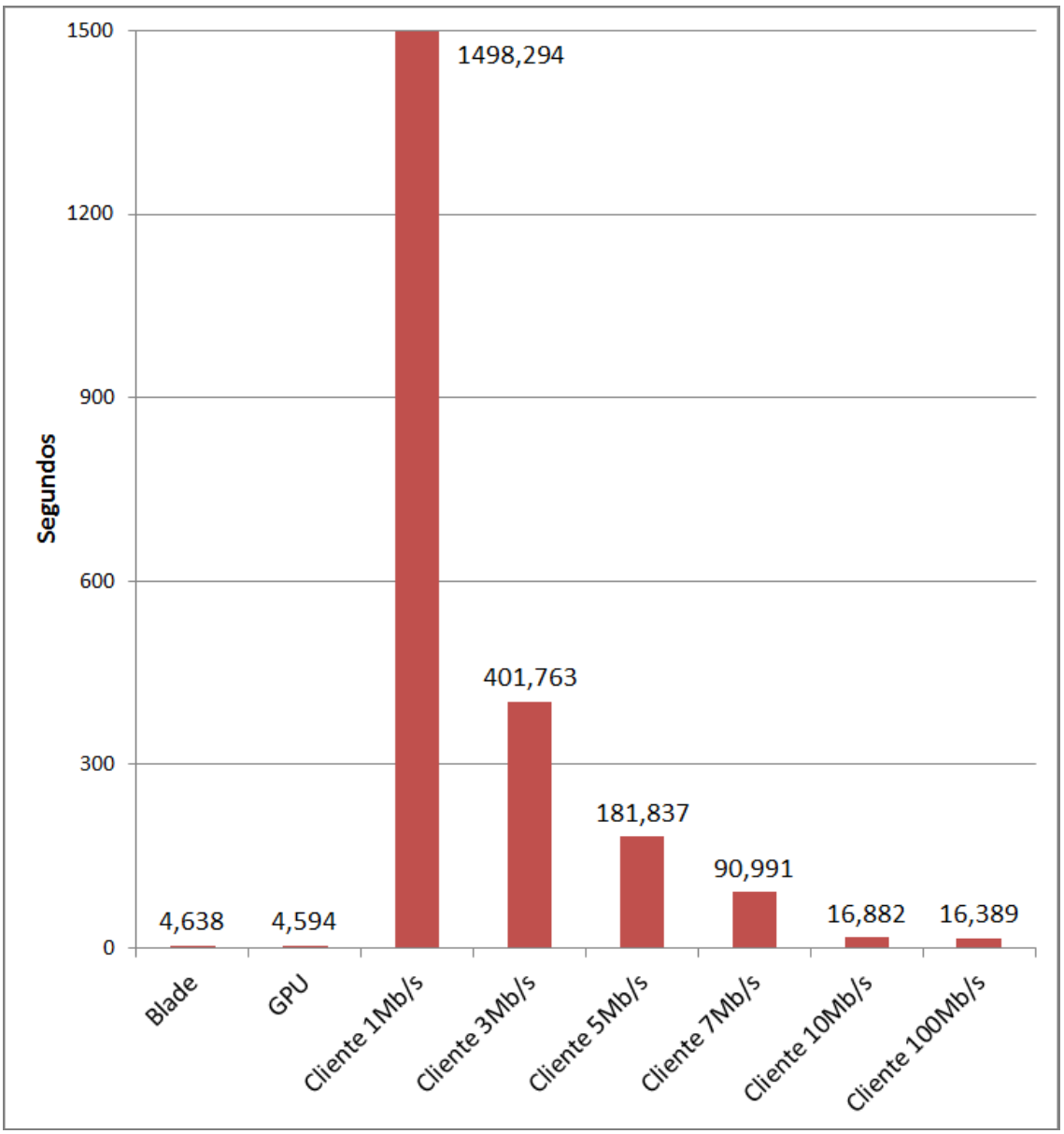

Figura 9 – Média do tempo de transferência dos arquivos DICOM.

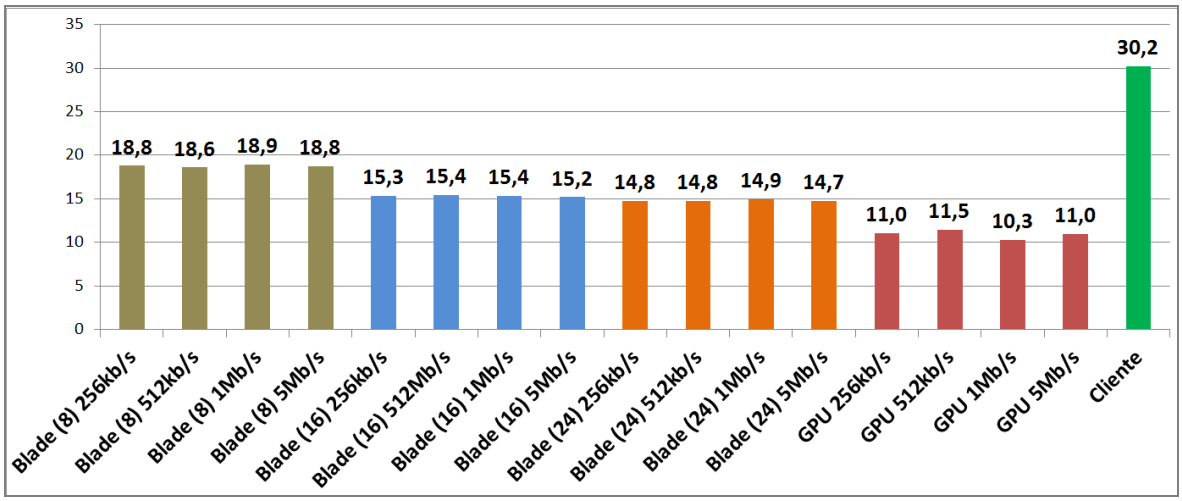

Figura 10 – Média do Tempo de Visualização (em segundos).

Tabela 6: Valores da média, desvio padrão, máximo e mínimo obtidos para o Tempo de Visualização.

Cenário	*Cores*	Velocidade	Tempo de Visualização			
			Média	Desvio Padrão	Máximo	Mínimo
Blade	8	256kb/s	18,778	3,948	24,294	10,511
		512kb/s	18,591	3,883	24,702	10,583
		1Mb/s	18,887	3,831	24,598	10,712
		5Mb/s	18,754	4,061	24,597	10,458
	16	256kb/s	15,340	2,824	24,119	9,655
		512Mb/s	15,387	2,838	20,785	9,136
		1Mb/s	15,362	2,822	19,647	9,451
		5Mb/s	15,184	2,811	19,366	9,203
	24	256kb/s	14,788	2,796	20,649	8,681
		512kb/s	14,773	2,809	21,898	9,036
		1Mb/s	14,907	2,833	21,228	9,095
		5Mb/s	14,731	2,866	21,467	8,687
GPU	1	256kb/s	11,010	3,070	18,187	5,400
		512kb/s	11,481	3,466	19,375	5,434
		1Mb/s	10,283	3,349	18,718	5,429
		5Mb/s	10,986	3,795	20,830	5,401
Cliente	1	(Local)	30,229	9,590	52,376	15,874

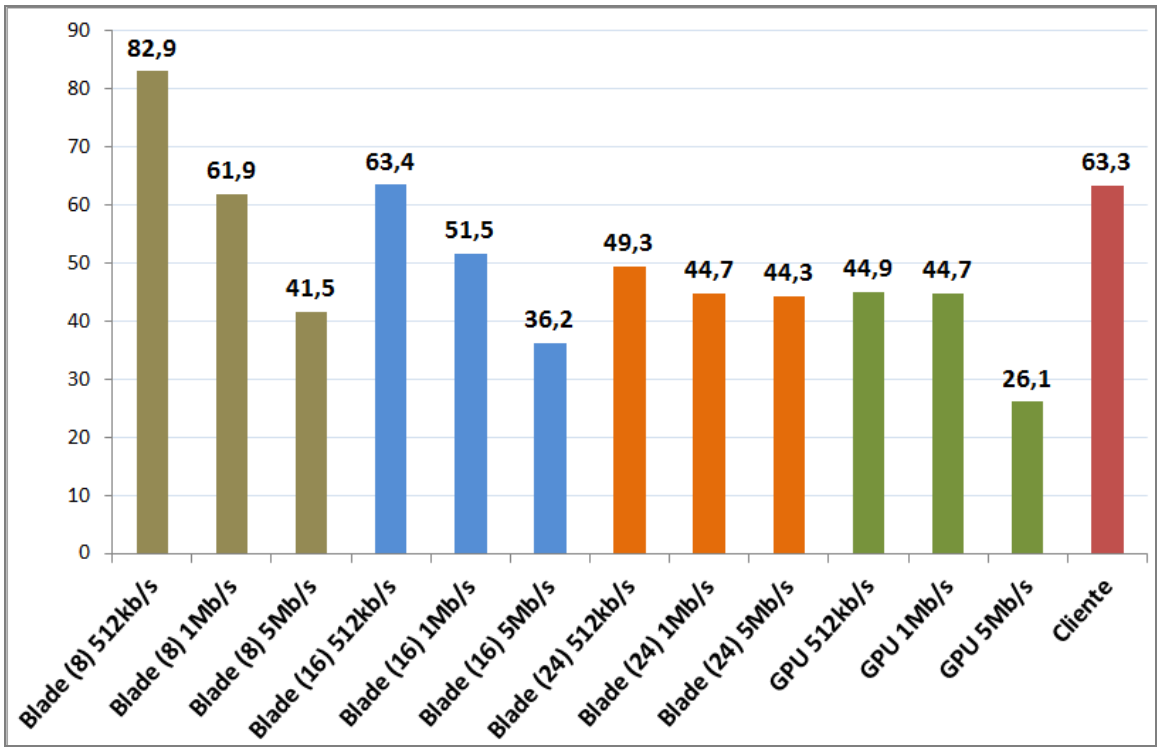

Figura 11 – Média do Tempo de Interação (em segundos).

Tabela 7: Valores da média, desvio padrão, máximo e mínimo obtidos para o Tempo de Interação.

Cenário	*Cores*	Velocidade	Tempo de Interação do Usuário			
			Média	Desvio Padrão	Máximo	Mínimo
Blade	8	512kb/s	82,907	37,172	131,176	26,480
		1Mb/s	61,878	43,761	177,799	32,848
		5Mb/s	41,500	18,062	83,010	21,470
	16	512kb/s	63,398	24,540	113,060	26,425
		1Mb/s	51,478	15,958	79,261	31,862
		5Mb/s	36,205	9,322	47,869	21,169
	24	512kb/s	49,321	15,533	74,977	25,956
		1Mb/s	44,696	7,274	53,561	33,742
		5Mb/s	44,324	12,743	68,124	28,329
GPU	1	512kb/s	44,903	17,302	79,122	22,434
		1Mb/s	44,678	16,111	63,789	14,834
		5Mb/s	26,072	11,066	46,527	11,585
Cliente	1	(Local)	63,319	29,879	114,150	31,361

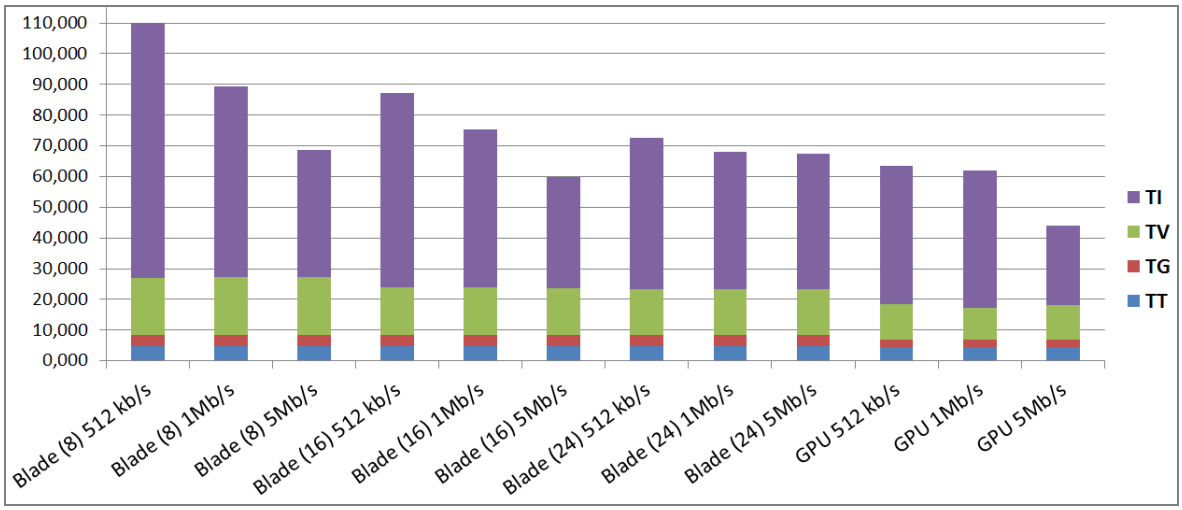

Figura 12 – Junção dos tempos TT, TG, TV e TI obtidos nos testes para os Cenários Blade e GPU (valores em segundos).

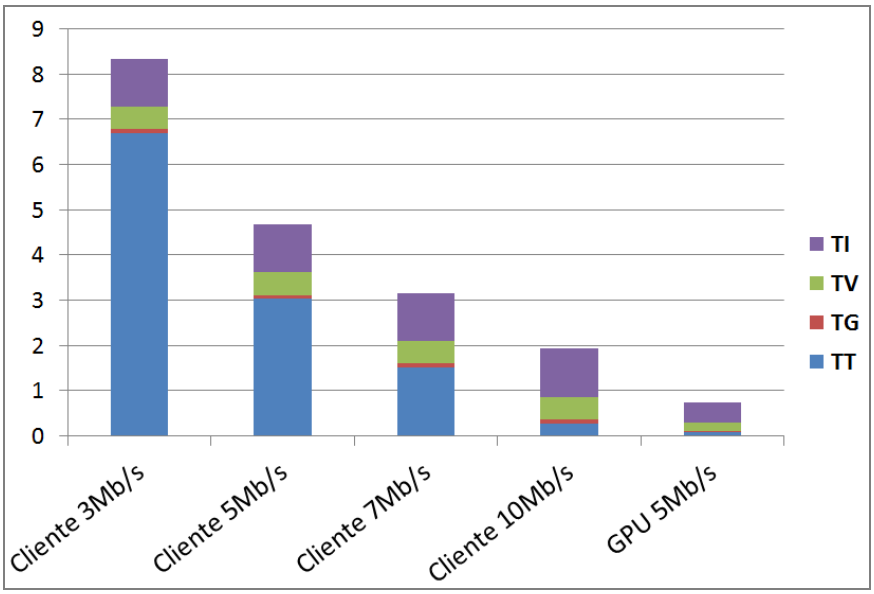

Figura 13 – Junção dos tempos TT, TG, TV e TI obtidos nos testes para o Cenário Cliente (valores em minutos).

Visualização nos Dispositivos Móveis

O cenário utilizado para os testes com o *tablet* e o celular foi o GPU e a conexão foi feita via rede sem fio, com velocidade de 54Mb/s. Para ambos os dispositivos, o tempo médio de visualização da imagem 3D (tempo TV) foi de aproximadamente 11 segundos. Nas Figura 14 e 15 estão ilustradas, respectivamente, as visualizações realizadas nos dois dispositivos móveis.

Figura 14 – Utilização da solução em um dispositivo *tablet*, modelo IPad da Apple.

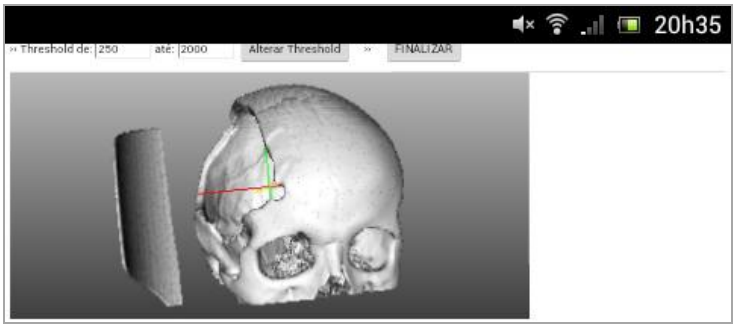

Figura 15 – Utilização da solução em um celular, modelo WT19a da Sony Ericson.

DISCUSSÃO

Nesta seção é realizada uma análise sobre os resultados obtidos para os indicadores de desempenho durante os testes da solução. Foi possível automatizar quase que por completo a realização dos testes, pois excetuando-se o tempo TI (que depende exclusivamente da interação do usuário), os demais tempos puderam ser medidos através da simples repetição da utilização da aplicação *web* cliente através da ferramenta Selenium IDE [42].

Tempo de Transferência

No Cenário Cliente, com uma conexão de 10Mb/s, o tempo médio obtido foi de 16,882 segundos, sendo o máximo 24,801 segundos (Tabela 4). Estes valores foram praticamente iguais aos obtidos para a velocidade de 100Mb/s, demonstrando que não há muita diferença no tempo TT para velocidades entre 10 e 100Mb/s. Em velocidades abaixo de 10Mb/s, os tempos médios obtidos ficaram na casa dos minutos. Isso indica que, para um usuário visualizar estes tipos de exames através da *internet*, tendo que transferir todos os arquivos DICOM para seu computador, ele terá que aguardar em média alguns segundos (≈16) caso sua velocidade seja igual ou superior a 10Mb/s. Como pode ser observado na Tabela 4, quando a velocidade de conexão esteve entre 5 e 7Mb/s o tempo de espera foi de 1,5 a 3 minutos. Para os casos onde a velocidade de conexão esteve entre 1 e 3Mb/s, o usuário teve que aguardar pelo menos 6 minutos para que os arquivos DICOM fossem transferidos para o seu computador.

Nos Cenários Blade e GPU, o tempo médio de transferência foi de aproximadamente 4,6 segundos, sendo que o maior tempo registrado foi de 12,890 segundos. Isso demonstra que, a arquitetura utilizada conseguiu minimizar bastante o problema do tempo necessário para a transferência dos arquivos DICOM.

Tempo de Geração

O tempo TG foi o único que não sofreu influência da velocidade da conexão de rede, pois o processo foi sempre executado localmente no computador onde foi realizada a renderização. Pelos resultados listados na Tabela 5, nota-se que o processador e a memória do computador onde foi realizado o processamento influenciaram diretamente no tempo TG. O tempo no Cenário GPU foi o melhor obtido, sendo em média 1,4 segundos inferior ao tempo do Cenário Blade e praticamente a metade do tempo obtido no Cenário Cliente. Pelo fato do computador utilizado no Cenário Cliente possuir menor capacidade de processamento e de memória RAM, foi registrado um maior valor para o desvio padrão, de 4,686 segundos, demonstrando que houve uma maior dispersão nos tempos obtidos em relação à média.

Tempo de Visualização

Pelo gráfico da Figura 10 pode-se observar que as velocidades de conexão utilizadas nos experimentos não tiveram influência sobre o tempo de visualização da primeira imagem 3D no computador cliente. Os resultados obtidos para diferentes velocidades dentro de um mesmo grupo de experimentos foram muito semelhantes. Isto se deve ao fato de que as imagens 3D resultantes da renderização possuíam um tamanho de no máximo 75kB, não causando um impacto relevante mesmo para conexões com velocidade de 256kb/s. Dessa forma, conclui-se que os tempos registrados representam quase que exclusivamente o tempo necessário para o serviço *web* de Renderização gerar a imagem 3D, já que o tempo necessário para a requisição do serviço *web* pela aplicação cliente e para o envio da imagem como resposta foi muito pequeno.

Por possuir um computador com menor capacidade de processamento e memória, no Cenário Cliente foi obtido o maior valor para o tempo de visualização, demorando cerca de 30 segundos em média (Tabela 6). Nele foi registrado também o maior desvio padrão (9,59) entre os cenários de teste.

No Cenário Blade, os resultados da Tabela 6 mostram que a execução do Paraview com diferentes números de *cores* (8, 16 e 24) não teve um impacto significativo no tempo médio de visualização da primeira imagem 3D. A diferença entre as médias dos experimentos Blade(8) e Blade(24) foi aproximadamente de 4 segundos apenas e, apesar da última configuração alocar 3 vezes mais recursos de processamento (24 *cores*) do que a primeira (8 *cores*), o ganho no tempo de processamento foi de aproximadamente 28%. Com isso, concluímos que para um ambiente similar ao do Cenário Blade não é necessário alocar uma porcentagem elevada dos recursos do servidor para realizar o processo de renderização, liberando-o para poder atender a mais requisições simultaneamente.

No Cenário GPU pôde-se observar o ganho de processamento que o *hardware* gráfico proporcionou na renderização do volume (Tabela 6). Os tempos obtidos foram em média 3,8 segundos abaixo do tempo aferido na configuração com maior poder computacional do Cenário Blade (com 24 *cores*).

Tempo de Interação

Para o tempo de TI, era esperado que os dados obtidos tivessem uma maior dispersão, podendo ser observada na diferença entre os valores máximo e mínimo e pelo desvio padrão obtidos e apresentados na Tabela 7. Isso porque, cada usuário teve pouco tempo para se acostumar com a aplicação e uns tiveram mais dificuldades do que outros na utilização. Mas o objetivo principal da medição do tempo TI era avaliar se a solução proporcionaria ao usuário final uma experiência de utilização que fosse melhor ao da utilização da aplicação no Cenário Cliente, com base nos tempos de resposta para a visualização e interação com a imagem 3D.

Após a entrevista com os usuários, todos informaram que a pior experiência de utilização foi durante o Cenário Cliente, devido ao maior tempo necessário para aguardar o carregamento inicial da imagem 3D na tela. Com base nas respostas dos usuários, percebeu-se que não foi a dificuldade de interação com a imagem o principal fator da escolha deste cenário como o pior, mas sim o tempo necessário até que a primeira imagem 3D fosse visualizada pelo usuário, sendo este tempo a somatória dos tempos TT, TG e TV. Isto é um dos problemas que a solução visa solucionar, pois a renderização é diretamente influenciada pelo *hardware* (processador, memória e GPU) e pela velocidade de conexão de rede que o usuário está utilizando. Em contrapartida, todos os usuários responderam que no Cenário GPU foi onde tiveram a melhor experiência de utilização, sendo esta escolha justificada pelo seu menor tempo de espera para visualizar a primeira imagem 3D e pela maior facilidade de realização das operações de rotação, deslocamento e zoom. Quanto ao questionamento de qual teria sido a operação mais difícil de ser realizada, independentemente do cenário, todos responderam que foi a rotação da imagem, a qual demonstrou ser menos precisa do que as outras (deslocamento e zoom), dificultando o posicionamento correto da imagem na forma que foram instruídos a fazer.

Ao final, pôde-se concluir que o Cenário GPU proporcionou a melhor experiência de utilização aos usuários, mesmo na menor velocidade testada (512kb/s), sendo o tempo médio de interação inferior a 45 segundos. Neste Cenário, as operações com a imagem fluíram de forma mais suave, sem interrupções bruscas na sua visualização, dando a impressão de estarem sendo realizadas localmente. O Cenário Blade proporcionou uma boa experiência principalmente nas velocidades de 1 e 5Mb/s, sendo que a maior dificuldade encontrada pelos usuários foi na configuração com 8 *cores* e velocidade de 512kb/s, onde foi registrada a maior média entre todos os experimentos, de aproximadamente 83 segundos (Tabela 7).

Análise Conjunta dos Tempos Registrados

O problema com o tempo necessário para a transferência dos arquivos DICOM foi bastante minimizado com a solução proposta, onde foram necessários em média apenas 4,5 segundos para transferir todas as fatias de um exame para o servidor da Camada de Renderização, tanto no Cenário Blade como no GPU (Figura 12). No Cenário Cliente, o melhor tempo foi registrado para as velocidades entre 10 e 100Mb/s, sendo de aproximadamente 16,5 segundos (Figura 13). Os demais tempos registrados no Cenário Cliente realçam este problema para o caso onde o acesso aos arquivos DICOM foi feito via *internet*, chegando a demorar mais de 3 minutos com velocidade de até 5Mb/s e superior a 20 minutos para velocidades de até 1Mb/s. Os valores para a velocidade de 1Mb/s foram omitidos do gráfico da Figura 13 para proporcionar uma maior clareza na comparação dos resultados.

O processo de geração do arquivo vtk causou pouco impacto no tempo total da renderização, tendo média de 5 segundos no Cenário Cliente, que foi a maior registrada (Tabela 5).

Outro problema encontrado na área da visualização volumétrica na medicina, está no fato de que nem todos os computadores conseguem renderizar e proporcionar uma interação com a imagem 3D de forma satisfatória para o usuário. É necessária uma boa combinação de *hardware* (processador, memória e GPU) para que a renderização seja realizada mais rapidamente e a interação seja feita de forma mais suave, sem interrupções bruscas na visualização da imagem. A solução também foi testada a partir de dispositivos móveis com menor poder computacional, sendo um *tablet* e um celular. Em ambos os dispositivos, a aplicação conseguiu ser executada e a imagem 3D inicial foi visualizada com um tempo médio de 11 segundos. A alteração do *threshold* funcionou perfeitamente nos dois dispositivos. No *tablet*, as operações de rotação e zoom puderam ser aplicadas, mas não de uma forma tão simples, sendo um pouco mais complicado de realizá-las com os dedos ao invés de um mouse. Já no celular, a interação com os dedos foi muito difícil e geralmente não resultava nas operações esperadas. Estes dois testes confirmaram que dispositivos móveis com limitações de processamento, memória e espaço em disco, podem ser utilizados para a visualização 3D através da solução proposta. Para isso, devem ser desenvolvidas páginas de visualização voltadas especificamente para o dispositivo a ser utilizado, possuindo botões e comandos auxiliares para a realização das operações de rotação, deslocamento e zoom.

CONCLUSÃO

Através do desenvolvimento deste trabalho, concluímos que é possível a utilização de uma solução de renderização remota de imagens 3D para visualização de exames médicos tomográficos. Foi demonstrado que a arquitetura utilizada conseguiu resolver satisfatoriamente os dois principais problemas da visualização volumétrica desses exames: a transferência dos arquivos DICOM e o poder computacional necessário para realizar o processo de renderização da imagem 3D.

Na arquitetura da solução proposta, através da utilização de uma conexão do tipo *Gigabit Ethernet* entre as Camadas PACS e de Renderização (Cenários GPU e Blade), a transferência dos arquivos DICOM conseguiu ser realizada com um tempo médio de 4,6 segundos, sendo bastante inferior quando comparado ao Cenário Cliente, principalmente com velocidades de conexão abaixo de 3Mb/s, onde o menor tempo registrado ficou acima dos 6 minutos.

Quanto ao poder computacional requerido no processo de renderização, a solução apresentada demonstrou que computadores menos potentes, e até mesmo dispositivos móveis, podem ser utilizados para visualizar as imagens 3D, pois todo o processamento é realizado pelo servidor remoto. As operações com a imagem (rotação, deslocamento e zoom) puderam ser realizadas de forma satisfatória em computadores e *notebooks*, conforme pôde ser observado no Cenário GPU, o qual proporcionou a melhor experiência de utilização para os usuários, mesmo a uma velocidade de apenas 512kb/s.

Os resultados dos experimentos realizados servem como base para orientar a implantação de uma solução deste tipo, tanto em ambientes com *hardware* gráfico como naqueles que não possuírem GPU. Este estudo demonstra que seria mais indicado utilizar a solução num

ambiente similar ao do Cenário GPU, pois neste cenário foram obtidos os melhores resultados, sendo uma opção mais barata em relação aos servidores do tipo *blade* (com vários processadores), além da melhor experiência proporcionada aos usuários durante a interação com a imagem 3D.

Soluções como esta podem ser utilizadas no ambiente do Hospital Universitário Onofre Lopes (HUOL) da UFRN, permitindo que sejam realizados diagnósticos e laudos a distância. Esta solução também pode ser aplicada na área acadêmica ou em treinamentos de profissionais de saúde, auxiliando os professores e instrutores durante as aulas que necessitarem da visualização 3D dos exames médicos.

REFERÊNCIAS BIBLIOGRÁFICAS

1. Andrade, Rafael; von Wangenheim, Aldo; Bortoluzzi, Mariana Kessler; Comunello, Eros. Using Mobile Wireless Devices for Interactive Visualization and Analysis of DICOM Data. *CBMS*, pages 97-101, 2003. IEEE Computer Society.
2. Carvalho, Cesar Augusto. Re-Slicing Tomographic Volumes with Shell Rendering. *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing* in SIBGRAPI ’02, pages 240–243, Washington, DC, USA, 2002. IEEE Computer Society.
3. Chittaro, Luca. Visualization of patient data at different temporal granularities on mobile devices.In Celentano, Augusto, editors, *AVI*, pages 484-487, 2006. ACM Press.
4. Chromium. Chromium Homepage. 2012.
5. Cline, H.; Lorensen, W.; Ludke, S.; Crawford, C.; Teeter, B. Two Algorithms for Three-Dimensional Reconstruction of Tomograms. *Medical Physics*, 15(3):320–327, 1988.
6. Cyclops, Group. The Cyclops Group. 2012.
7. DCM4CHE. dcm4che: Open Source Clinical Image and Object Management. 2012.
8. Deitel, Paul J.; Deitel, Harvey M. *Java for Programmers*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2nd edition, 2011.
9. Eclipse. Eclipse. 2012.
10. Erl, Thomas. *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.
11. Foley, James D. ; van Dam, Andries; Feiner, Steven K.; Hughes, John F. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
12. Gehani, Narain. *The Database Book: Principles and Practice Using MySQL*. Silicon Press, Summit, NJ, USA, 2nd edition, 2011.
13. Hearn, Donald; Baker, M. Pauline. *Computer graphics (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
14. Hearn, Donald; Baker, M. Pauline. *Computer graphics with OpenGL (3. ed.)*. Pearson Education, 2004.

15. Jamae, Javid; Johnson, Peter. *JBoss in Action: Configuring the JBoss Application Server*. Manning Publications Co., Greenwich, CT, USA, 2009.

16. Jayasinghe, Deepal; Azeez, Afkham. *Apache Axis2 Web Services, 2nd Edition*. Packt Publishing, 2011.

17. Jomier, Julien; Jourdain, Sebastien; Ayachit, Utkarsh; Marion, Charles. Remote visualization of large datasets with MIDAS and ParaViewWeb. *Proceedings of the 16th International Conference on 3D Web Technology* in Web3D '11, pages 147–150, New York, NY, USA, 2011. ACM.

18. Jourdain, S.; Ayachit, U.; Geveci, B. ParaViewWeb, A web framework for 3D Visualization and Data Processing. *IADIS Intern. Conference on Web Virtual Reality and Three-Dimensional Worlds*, 2010.

19. Kaufman, Arie E. Volume Visualization: Principles and Advances. *Annual Conference on Computer Graphics*, 1997.

20. Kitware. ITK - Insight Segmentation and Registration Toolkit. 2012.

21. Kitware. VTK - Visualization Toolkit. 2012.

22. Kitware, Inc. ParaView Users Guide. 2012.

23. Kroll, Michael; Melzer, Kay; Lipinski, Hans-Gerd. Accessing DICOM 2D/3D-Image and Waveform Data on Mobile Devices.In Bludau, Hans-Bernd and Koop, Andreas, editors, *Mobile Computing in Medicine* in LNI, pages 81-86, 2002. GI.

24. Lacroute, Philippe; Levoy, Marc. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* in SIGGRAPH '94, pages 451–458, New York, NY, USA, 1994. ACM.

25. Lamberti, Fabrizio; Sanna, Andrea. A Streaming-Based Solution for Remote Visualization of 3D Graphics on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):247–260, 2007.

26. Levoy, Marc. Display of Surfaces from Volume Data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.

27. Lorensen, William E.; Cline, Harvey E. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987.

28. Luke, Eric J.; Hansen, Charles D. Semotus Visum: a flexible remote visualization framework. *Proceedings of the conference on Visualization '02* in VIS '02, 2002. IEEE Computer Society.

29. Mesa3D. The Mesa 3D Graphics Library. 2012.

30. Moreland, Kenneth. The ParaView Tutorial. 2012.

31. Moreland, Kenneth; Rogers, David; Greenfield, John; Geveci, Berk; Marion, Patrick; Neundorf, Alexander; Eschenberg, Kent. Large Scale Visualization on the Cray XT3 Using ParaView. *CUG 2008, Helsinki, Finland*, 2008.

32. MPI. The Message Passing Interface (MPI) standard. 2012.

33. MPICH. MPICH: High-performance and Portable MPI. 2012.

34. NEMA. *The DICOM Standard*. 2012.

35. Oakman, Amere. Volume Graphics:The road to interactive medical imaging?. *SURPRISE 96*, 1996.

36. American Paper Optics. Chromatek. 2012.

37. Paiva, A. C.; Seixas, R. B.; Gattass, M. Introdução à Visualização Volumétrica. 2012.

38. Phong, Bui Tuong. Seminal graphics. chapter Illumination for computer generated pictures, pages 95–101. ACM, 1998.

39. Pianykh, O.S. *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide*. Springer, 2008.

40. Potts, Stephen; Kopack, Mike. *Sams Teach Yourself: Web Services in 24 Hours*. SAMS, 2003.

41. Sampaio, Cleuton. *SOA e Web Services em Java*. Brasport, 2006.

42. Selenium. Selenium IDE. 2012.

43. Stripes. Stripes Framework. 2012.

44. W3Schools. WSDL Tutorial. 2012.

45. Wei, Hui; Liu, Enjie; Clapworthy, Gordon. Interactive 3D rendering to assist the processing of distributed medical data. *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia* in IITM '10, pages 119–126, 2010. ACM.

Wondershaper. The Wonder Shaper. 2012.