ANÁLISE DE DESEMPENHO DE RENDER FARM BASEADA EM LOKI RENDER

https://doi.org/10.5281/zenodo.15571849

SALGADO, Rodrigo Lopes, Especialista*

* Faculdade de Tecnologia de Praia Grande
CEETEPS - Centro Estadual de Educação Tecnológica Paula Souza
Praça 19 de Janeiro, 144, Boqueirão, Praia Grande / SP, CEP: 11700-100
Telefone (13) 3591-1303
rodrigo@fatecpg.com.br

RESUMO

O presente trabalho apresenta um estudo de desempenho de uma *render farm* baseada na ferramenta de distribuição de processamento gráfico *Loki Render*. A princípio é apresentada a conceitualização teórica de computação distribuída, visto que a ferramenta estudada se apoia nesta arquitetura. Em um segundo momento o trabalho lista as ferramentas envolvidas neste processo, tanto a ferramenta principal quanto as pré-requeridas por ela e demonstra os dados dos diversos testes de desempenho realizados na *render farm* já implementada. Por fim há uma ponderação dos dados colhidos demonstrando pontos fortes e fracos da arquitetura e da ferramenta, de forma que possa contribuir para futuros estudos sobre a viabilidade de implantação de um *cluster de renderização*.

PALAVRAS-CHAVE: *render farm, renderização, cluster, Loki Render,* Blender 3D, computação distribuída.

ABSTRACT

This paper presents a study about performance of a render farm based on a distributed network rendering Loki Render. At first is presented the theoretical conceptualization of distributed computing, since the main application studied is based on this architecture. Following the paper lists the tools involved in this case both the main tool as prerequired and shows the data of many performance tests conducted in render

farm already in place. Finally there is a weighting of the data collected showing strengths and weaknesses of the architecture and the tool so that may contribute to future studies on the feasibility of implementing a cluster rendering.

KEY-WORDS: *render farm, rendering, cluster, Loki Render, Blender, distributed computing.*

INTRODUÇÃO

A computação distribuída é uma arquitetura de software em que uma coleção de computadores independentes se apresenta ao usuário como um sistema único e consistente, de acordo com Tanembaum (2007). Coulouris (2005), ainda complementa esta definição afirmando que estes computadores independentes interligados são dotados de software que permita o compartilhamento dos recursos do sistema. Portanto, a computação distribuída pode ser entendida como uma arquitetura em que vários computadores atuam como se fossem apenas um, através de um software que, de um lado, gerencia os recursos compartilhados e, de outro, oferece-os ao usuário de forma transparente, sem que este visualize o agrupamento de computadores.

É através da computação distribuída que diversas aplicações, comuns aos usuários atualmente, são viáveis, mesmo que estes usuários não notem a demanda de processamento ou a arquitetura utilizada. Pode-se citar algumas destas aplicações que exigem alta capacidade computacional como exemplo: aplicações de processamentos geoespaciais, simulações topológicas diversas (inclusive na busca por poços de petróleo), sistemas simuladores de previsão climática (previsão de temporais, ciclones, maremotos, etc.), computação gráfica com simulações complexas de materiais, iluminação e física, análise de sinais, dentre tantas outras.

O cenário de estudo deste trabalho ilustra as altas demandas de processamento geradas pelas aplicações de computação gráfica. O processo da computação gráfica responsável por esta elevada necessidade de processamento é conhecido como renderização, e é justamente este processo que será distribuído. No ambiente da computação gráfica,

a utilização de computação distribuída para realizar processos de renderização é peculiarmente chamado de *render farm*. Existem diversas técnicas e ferramentas para se construir uma *render farm*. Podemos criá-la no âmbito do sistema operacional ou na camada de aplicativo. O trabalho apresentará a implementação de uma *render farm* através de um aplicativo cliente, que será instalado nos computadores e responderão ao aplicativo servidor, que será executado do computador de interface com o usuário. A ferramenta escolhida foi o aplicativo *Loki Render*, que é uma aplicação gratuita e de código livre. O *Loki Render* trabalha em conjunto com a aplicação de computação gráfica *Blender 3D*, que também é gratuita e de código livre.

O trabalho consiste na renderização de diferentes projetos, que evoluirão em termos de complexidade, para monitorar se os ganhos de desempenho prometidos teoricamente pelos estudos de computação distribuída se aplicam na prática. Todos os testes serão realizados diversas vezes, incrementando-se a quantidade de computadores na citada arquitetura, para verificar se a administração dos clientes pelo servidor se torna custosa, e até onde a curva de benefício é atrativa.

Por fim, a análise dos dados coletados é apresentada para que futuras pesquisas possam se orientar com dados efetivamente medidos em um ambiente real de produção, sem que seja necessária a construção de todo o cenário de testes que certamente consume muito tempo.

1 COMPUTAÇÃO DISTRIBUÍDA

Andrew Tanembaum (2007) define computação distribuída como uma arquitetura de *software* formada por vários computadores independentes que se apresentam ao usuário como um sistema único e consistente. A partir desta definição, diversos outros autores complementam o conceito de computação distribuída com algumas peculiaridades. Coulouris (2005) incrementa afirmando que estes computadores precisam de um software para orquestrar esta arquitetura.

Atualmente, com a necessidade de consumir recursos de forma sustentável em evidência, aproveitar a ociosidade (mesmo que temporária) de equipamentos com capacidade de processamento torna-se um processo nobre em TI. Além disto, a evolução dos computadores

em termos de potencial de processamento não acompanhou a demanda exigida pela maioria das aplicações. Percebe-se isto através dos computadores pessoais que ganham mais processadores (ou núcleos de processamento), visando incrementar o desempenho. Ainda assim, a limitação na capacidade de processamento impõe sérias restrições a alguns tipos de *softwares*, tais como manipuladores de imagens 3D, jogos, geoprocessamento, simulações climáticas, dentre outras. Uma forma de contornar esta limitação local de processamento é utilizar técnicas que possibilitem o processamento distribuído (DANTAS, 2005).

Tanembaum (2007) sustenta que o sucesso de um projeto de computação distribuída deve-se a elaboração e a adoção de metas que devem ser rigorosamente cumpridas.

1.1 METAS DE SISTEMAS DISTRIBUÍDOS

Segundo Tanembaum e Steem (2007), todos os sistemas distribuídos devem observar algumas metas que fazem com que sistemas computacionais possam ser classificados como distribuídos. São elas:

a) compartilhamento de recursos;
b) transparência;
c) escalabilidade;
d) confiabilidade;
e) flexibilidade;
f) desempenho;
g) heterogeneidade.

A principal meta a ser cumprida para qualquer sistema distribuído é o compartilhamento de recursos, sejam eles computadores, impressoras, dados, processadores, discos ou quaisquer outros elementos que possam ser compartilhados e eficientemente controlados, tornando fácil o acesso destes aos usuários.

A transparência é um termo largamente utilizando em Tecnologia da Informação (TI) em diversos contextos, e sempre denota que algum elemento ou característica desempenha sua função sem que o usuário (ou

desenvolvedor) perceba este elemento. No contexto de sistemas distribuídos, a transparência garante que o usuário não visualize a arquitetura de vários computadores conectados se apresentando como apenas um grande e potente computador. Outras características também são transparentes ao usuário de sistemas distribuídos, como por exemplo, a localização destes computadores, as funcionalidades de disponibilidade, correção de erros, replicação e concorrência.

Outra importante meta é a escalabilidade, que permite que o incremento de novos computadores ao sistema distribuído ocorra da maneira menos impactante possível, em termos de tempo e custo. Coulouris (2005) reforça ainda que a escalabilidade pode se basear em outros aspectos além do tamanho (quantidade de usuários ou de computadores do sistema distribuído). A escalabilidade também pode ser abordada tanto na esfera geográfica quanto na administrativa. Escalabilidade geográfica define a distância máxima entre os nós do sistema distribuído, permitindo (e restringindo) a escala de tamanho. A escalabilidade administrativa permite que centros de domínios administrativos possam ser acrescidos ao sistema da mesma forma que nós, ou seja, de maneira viável, rápida e eficiente.

A confiabilidade de um sistema distribuído é garantida através da implementação de mecanismos de tolerância a falhas que detectam e permitem a recuperação caso algum comportamento imprevisto do sistema possa vir a ocorrer. Outra meta é a flexibilidade em que há a capacidade de manutenção, tanto preventiva quanto corretiva e perfectiva, sem que seja necessária a interrupção do funcionamento do sistema. O desempenho também é uma das razões para se distribuir um sistema, visto que deve ser sempre igual ou superior ao desempenho de um sistema centralizado, característica esta que é obtida pelo fato de ocorrer uma menor necessidade de transferência de dados e sincronização já que o paralelismo computacional é utilizado fazendo com que muitas tarefas ocorram nos nós do sistema distribuído, otimizando uso de *cache*, *threads* e concorrência do sistema como um todo. Além disto, o sistema distribuído deve ser tão heterogêneo quanto o possível, permitindo um elevado grau de portabilidade que garanta que diferentes plataformas possam utilizar os recursos compartilhados por este sistema. Este heterogeneidade é possível utilizando-se protocolos comuns para transferência de dados, por exemplo, o padrão XML[1](eXtensible Markup Language) (DANTAS, 2005).

[1] XML é uma linguagem de marcação recomendada pela W3C que descreve tipos de dados de modo a garantir o compartilhamento de informação através da internet e diferentes sistemas.

1.2 TIPOS DE SISTEMAS DISTRIBUÍDOS

Os sistemas distribuídos podem ser classificados de três formas: sistemas de computação, sistemas de informação e sistemas pervasivos. Cada um deles explorando uma característica acerca da TI e fazendo uso do conceito de sistema distribuído para atingir seu objetivo.

1.2.1 Sistemas de Computação Distribuída

Os sistemas de computação distribuída podem ser de dois tipos: sistemas de computação de grade e sistemas de computação de cluster.

Os sistemas de computação de grade se baseiam na grande diversidade de recursos de computação distribuídos em escala global, tornando-se independente da posição geográfica. É o conceito de computação distribuída mais amplo, tanto no sentido de abrangência territorial quanto na capacidade de acoplar recursos heterogêneos. Os sistemas de computação de *cluster* possuem uma abrangência menor que a computação de grade e se baseiam fortemente na arquitetura cliente-servidor, onde um ponto computacional da rede assume o papel de servidor, tanto no âmbito de sistema operacional quanto de aplicação, gerenciando os demais pontos computacionais desta rede - os clientes - quanto a entrega de tarefas a serem processadas quanto a coleta e montagem dos dados já processados pelos clientes (PEDROSO, 2006).

1.2.2 Sistemas de Informação Distribuída

Segundo Saha (2003), os sistemas de informação distribuída assumem dois tipos: sistema de processamento de transações e sistemas de integração de aplicações empresariais.

Os sistemas de processamento de transações envolvem mais de uma transação que devem ser executadas todas com sucesso ou, caso haja alguma falha, devem ser canceladas na sua completude. Em ciência da computação esta forma de transação é denominada "transação atômica". O exemplo clássico que ilustra tal transação é o procedimento para uma transferência bancária. A transferência é uma transação atômica, que é composta por um débito em uma conta (transação 1) e o crédito em outra (transação 2). Se o débito ou o crédito falhar, toda a transação é desfeita e a transferência não ocorre, pois em hipótese alguma um débito poderá ser feito e o respectivo crédito não.

Sistemas de integração de aplicações empresariais são baseados

em aplicativos que atuam com *middlewares*, integrando aplicações clientes a diversos servidores, de maneira que diferentes tecnologias e plataformas interajam entre si através de requisições e chamadas de procedimentos e métodos remotos.

1.2.3 Sistemas Distribuídos Pervasivos

Saha (2003) cita os sistemas pervasivos como um paradigma em que os ambientes das máquinas e dos humanos se fundem. Um ambiente pervasivo consiste um ambiente em que a tecnologia se torna transparente, desaparecendo do ambiente humano. O papel do humano de entrar no ambiente tecnológico, realizar alguma tarefa e sair deste ambiente é descartado, estando o humano inserido no ambiente tecnológico o tempo todo, de forma que ele não perceba esta imersão.

Os sistemas pervasivos necessitam da infraestrutura da computação distribuída pois diferentes equipamentos de distintas plataformas interagem entre si. A domótica[2] é um exemplo de sistema distribuído pervasivo. Através deste sistema é possível integrar celulares, televisores, geladeiras e qualquer outro eletrodoméstico com capacidade computacional e infraestrutura de rede de forma a compor um único sistema distribuído (OGATA, 2003). Para Satyanarayanan (2001) a computação pervasiva oferece acesso imediato e uniforme às informações de maneira transparente, com dispositivos estáticos e móveis interagindo com o ambiente auxiliando o usuário na execução das suas tarefas.

Outros exemplos de aplicações de sistemas distribuídos pervasivos são os sistema de acompanhamento e monitoramento de saúde, sistemas de automação baseado em sensores e demais sistemas com dispositivos que se interconectam com pouca (ou nenhuma) interação humana auxiliando o homem em suas tarefas diárias.

2 *RENDER FARM*

De acordo com Brito (2010), para definir *render farm* é necessário o entendimento do termo renderização, e para tal é necessário

[2] Domótica é a tecnologia de gestão de recursos computacionais presentes em casas. O termo tem origem na junção das palavras *domus* (do latim, casa) e robótica.

definir *render*. O termo *render* é utilizado em muitas outras áreas além da computação gráfica. Uma de suas traduções é "acabamento", e é a que mais se aproxima do real significado deste termo em computação gráfica. O ato de renderizar compreende as tarefas de transformar um modelo gráfico vetorial em um modelo rasterizado através do cálculo de diversas características deste modelo e dos demais elementos que o cercam. Por exemplo, um modelo de um edifício em tempo de modelagem possui apenas forma e dimensão. Características como material, textura, iluminação, reflexos, transparências e tantas outras são apenas valores configurados que não estão de fato aplicados ao objeto modelado (neste caso o edifício). Para visualizar a imagem do edifício com todas as características que foram configuradas é necessária a renderização deste modelo. É neste momento que dezenas de milhares de cálculos são executados a fim de definir que cor será atribuída a cada *pixel* que compõe a imagem gerada considerando-se tudo o que pode interferir nesta cor (FOLEY, 1995). Esta tarefa consome muito tempo, exigindo assim um enorme potencial de processamento para que estes cálculos de renderização sejam realizados da forma mais rápida possível.

O termo *farm*, que traduzido para o português significa "fazenda", agregado a palavra *render* denota uma "fazenda de renderização", que na área de computação gráfica é um *cluster* de computadores construído com a finalidade de potencializar capacidade de processamento para a realização das tarefas de renderização.

Patoli (2008) define *render farm* como um *cluster* de computadores utilizado por profissionais da computação gráfica com o objetivo de potencializar a capacidade de processamento para a renderização de imagens. O mercado de produção de animações de computação gráfica exige um esforço muito grande em termos de processamento para realizar a renderização das imagens, tornando esta tarefa extremamente lenta caso apenas um processador a desempenhe. Desta forma um *cluster* agruparia diversos processadores e delegaria a eles a tarefa desta renderização.

Esta denominação é mundialmente conhecida e foi criada para fazer analogia a uma fazenda leiteira, onde uma vaca apenas não produz leite suficiente para a demanda. Porém a escalabilidade da fazenda leiteira é diretamente proporcional à quantidade de vacas, portanto, quanto mais vacas, mais leite. A ideia da *render farm* é a mesma.

A escalabilidade originada pela computação distribuída permite inserir computadores com um custo de administração perto do desprezível, aumentando a capacidade de processamento em um mesmo período de tempo.

Existem diversos tipos de *render farms*, variando apenas quanto ao profissionalismo que se emprega em implementá-los. É possível criar uma *render farm* caseira, utilizando-se apenas dois computadores convencionais. Se hipoteticamente estes dois computadores possuírem as mesmas características em termos de processador, o ganho de tempo com a renderização estará na ordem de duas vezes. Não há nenhuma limitação quanto ao equipamento que irá compor o *cluster* de computadores, porém quanto maior o poder de processamento destes equipamentos, maior será o poder de processamento da *render farm*. O mesmo raciocínio é válido para a quantidade de núcleos que os processadores possuírem (BRITO, 2010).

3 *BLENDER FOUNDATION* E *BLENDER 3D*

Blender 3D é um software mantido pela *Blender Foundation*, uma empresa sem fins lucrativos com sede na capital da Holanda, Amsterdã. O criador do aplicativo e presidente da fundação é Ton Rosendaal. O *Blender 3D* começou a ser desenvolvido em 1995 (sendo concluído em 1998) pela empresa recém fundada por Ton Rosendaal chamada *Not a Number* (NaN). A princípio este aplicativo foi criado com o intuito de ser um produto comercial, porém em 2002 a NaN encerrou suas atividades devido a fortes dificuldades financeiras, descontinuando assim o desenvolvimento e manutenção do *Blender 3D*. Neste mesmo ano a *Blender Foundation* foi criada e seus desenvolvedores passaram a desenvolver a aplicação no formato *OpenSource* (BLENDER, 2011).

O aplicativo possui diversas ferramentas para o trabalho do profissional da computação gráfica, entre eles destacam-se modelagem, materiais, motor para desenvolvimento de jogos, simulação física de fluidos, tecidos, balística e corpos moles, animação, editores de áudio e vídeo.

O motor de renderização interno do *Blender 3D* conhecido como *internal render* atende satisfatoriamente diversas funcionalidades

e requisitos para a geração de imagens de alta qualidade. Ainda assim é possível a integração de renderizadores externos aos projetos feitos em *Blender 3D*, ampliando assim a diversidade de recursos de renderização da ferramenta. Por exemplo, existem renderizadores especializados no tratamento de luz, outros que possuem algoritmo de renderização otimizado para animações, alguns que renderizam o trabalho de forma infinita, deixando por conta do usuário o encerramento dos cálculos de aperfeiçoamento da imagem, e tantas outras especializações.

4 *LOKI RENDER*

Loki Render é uma aplicação *OpenSource* que distribui a renderização de imagens do *Blender 3D* para diversos computadores realizarem o processamento, possibilitando desta forma a redução do tempo de espera para que o trabalho seja realizado. É uma ferramenta que permite que diversos computadores (*slaves*) respondam e sirvam a um computador principal (*master*). Pode ser baixada gratuitamente através do endereço http://loki-render.berlios.de.

Dentre suas funcionalidades há a possibilidade de distribuição de uma imagem (*frame*) para cada nó, tarefa comum em animações onde diversos quadros precisam ser renderizados. Outra característica do *Loki Render* é que é possível distribuir uma única imagem para ser renderizada pelos nós do sistema distribuído, por suportar um tipo de renderização chamado de *tile render*, que consiste em fracionar uma imagem a ser renderizada em diversas pequenas partes e enviar cada uma destas partes para os nós, que os renderizam e reenviam de volta para o nó principal juntar, formando a imagem final, já renderizada.

Outra característica importante é o suporte ao renderizador *Yafray* além do renderizador interno do *Blender 3D*. De acordo com o desenvolvedor há a previsão da inclusão de mais motores de renderização no futuro.

Com todos estes aspectos, ainda existe uma característica muito mais significativa em termos de importância, que é sua capacidade de portabilidade. Como foi desenvolvido em sua totalidade em Java, permite que sua versão *client* (que é instalada nos nós escravos - *slaves*) possa ser executada de qualquer plataforma, seja ela Linux, Mac ou

Windows, ampliando o fator de escalabilidade da *render farm*.

Sua última versão, a 0.6.2, lançada em 18 de novembro de 2009 contou com a correção de algumas falhas e o incremento de novas funcionalidades, como o suporte ao sistema físico que simula partículas, roupas e corpos moles. Outra funcionalidade bastante aguardada pela comunidade de desenvolvedores e colaboradores foi a capacidade da versão cliente poder ser executada através de linha de comando, facilitando muito a administração de grandes *render farms*, já que pode-se automatizar o início da aplicação com a chamada específica de alguns parâmetros de ambiente.

O código-fonte da aplicação é mantido em um sistema de controle de versão aberto. A ferramenta utilizada é o *Subversion* (SVN) e pode ser acessado através de svn://svn.berlios.de/loki-render/trunk.

Uma característica dos aplicativos de renderização distribuída é que eles não trabalham com a geração de vídeos e sim com a geração de imagens que representam os quadros da animação em questão. É necessário uma ferramenta de pós produção para a geração do arquivo de vídeo em formato específico.

5 ANÁLISE DE DESEMPENHO

5.1 AMBIENTE DE TESTES

O ambiente de testes é composto por dezesseis computadores com processador Intel® Core™2 Duo CPU E7500 2,93GHz com 3 Gb de memória RAM localizados no laboratório de computação gráfica da FATEC Praia Grande (FATEC-PG). Estes computadores estão interligados em rede entre si e fazem parte de toda a rede da FATEC-PG, portanto não se trata de uma rede isolada para o propósito dos testes.

A topologia de rede utilizada é a estrela, utilizando um concentrador como centro da rede que é interligada por cabeamento de par trançado. Um *switch* interconecta todos os equipamentos do ambiente de testes, portanto há apenas um "salto" de nó na rede entre a comunicação do servidor com os clientes (TANENBAUM, 1999).

5.2 PROJETOS DE TESTES

Foram analisados três projetos, que serão doravante chamados de PROJ I, PROJ II e PROJ III, respectivamente.

O PROJ I consiste em uma cena composta apenas de primitivas gráficas com apenas um quadro, utilizada para testes de *tile rendering*. Foram criados nove objetos: três planos, dois cubos, um cilindro, um cone seccionado e duas esferas. Estes objetos totalizam 2148 vértices e 2255 faces. O formato de saída da imagem renderizada foi configurado como JPEG (imagem rasterizada[3] comprimida), com dimensões de 800 *pixels* de largura por 600 *pixels* de altura. Quanto ao padrão de cores, foi definido o padrão RGB (*red, green and blue*) com 256 tonalidades de cor para cada uma das três cores básicas deste padrão, gerando assim mais de 16 milhões de cores a partir desta combinação (256 cores vermelha x 256 cores verde x 256 cores azul). Utilizou-se recursos de iluminação de ambientes (*Ambient Occlusion*) em conjunto com duas lâmpadas, uma omni direcional produzindo sombras e outra hemi direcional sem produção de sombra.

O PROJ II consiste em uma cena composta por objetos complexos modelados (copos e taças) utilizando alguns recursos de materiais (como reflexo e transparência) e outros de textura (como madeira e ladrilhos). Este projeto contém apenas um quadro, sendo assim, sua renderização gera apenas uma imagem. Devido à complexidade desta cena esta renderização consome mais recursos de processamento pois os cálculos para definição da cor de cada pixel visualizado pela câmera são mais sofisticados, caso se confronte com os cálculos do PROJ I.

Já o PROJ III é composto por uma cena de primitivas gráficas com materiais básicos animada em 128 quadros a uma taxa de 25 quadros por segundo (ou *frames per second - fps*). Este projeto distribui as tarefas de renderização para os aplicativos clientes de forma que cada tarefa seja composta pelo trabalho de execução de um quadro inteiro. Sendo assim, este projeto se difere dos demais por não se tratar de um *tile rendering* e sim *frame rendering*, onde cada nó da rede distribuída renderiza um quadro da animação por inteiro. Por questões de metodologia de análise estatística, a quantidade de quadros foi definida propositadamente em 128 para que a quantidade de quadros fosse múltipla da quantidade

[3] Imagem rasterizada é uma imagem digital baseada em mapa de pixel, ou mapa de bits, também conhecida como formato *bitmap*.

de nós utilizados nos testes, evitando assim que na última rodada de renderização alguns nós ficassem ociosos.

5.3 METODOLOGIA DE TESTES

A metodologia de testes consiste em renderizar os projetos utilizando-se uma quantidade crescente de *nós* na *render farm*. Como todos os equipamentos tem a mesma especificação tratando-se de *hardware*, é praticamente desprezível a diferença de poder de processamento real de cada um destes *nós*. Cada um dos três projetos é submetido a cinco rodadas de renderização. A quantidade de computadores da *render farm* é dobrada a cada rodada, iniciando-se com apenas um equipamento. Assim, na rodada 1 é utilizado apenas um computador, na rodada 2 são utilizados dois computadores, na rodada 3, quatro computadores, na rodada 4, oito computadores e, finalmente, na rodada 5 temos a rede completa, com dezesseis computadores. Efetivamente, como cada computador possui um processador com dois núcleos, temos sempre o dobro de processadores realizando o trabalho, pois em cada computador são considerados, pelo *Loki Render*, dois processadores.

Estas rodadas foram executadas três vezes, em dias diferentes, para que não fosse desprezada a possibilidade de variação de desempenho que pudesse ser causada por qualquer fator que diminuísse a *performance* da rede, já que a rede de testes não estava isolada da rede da faculdade.

5.4 REALIZAÇÃO E RESULTADO DOS TESTES

Todos os testes foram executados durante o mês de dezembro do ano de 2010, no período em que as atividades de rede são significativamente menores pois coincide com o período de provas e apresentação de trabalhos de conclusão de curso. Sendo assim, não há mais aulas em laboratórios e as atividades semestrais de manutenção do departamento de tecnologia da informação e comunicação da FATEC-PG ainda não se iniciaram.

Os resultados dos testes são apresentados nos quadros abaixo, onde constam os projetos e suas respectivas rodadas de testes. Os valores apontados representam a quantidade de segundos necessários para a renderização de cada projeto, seguidos pela média aritmética simples dos três valores apontados (tabelas 1, 2, 3, 4 e 5).

Tabela 1 - Resultados dos testes com 1 computador (em segundos)

		Rodada 1			
	Qtde de Computadores	1			
	Qtde de Núcleos	2			
		Teste 1	Teste 2	Teste 3	Média
Projeto	PROJ I	60,16	60,13	59,84	60,04
	PROJ II	754,53	758,19	755,70	756,14
	PROJ III	8263,49	8290,65	8264,38	8272,84

Tabelas 2 - Resultados dos testes com 2 computadores (em segundos)

		Rodada 2			
	Qtde de Computadores	2			
	Qtde de Núcleos	4			
		Teste 1	Teste 2	Teste 3	Média
Projeto	PROJ I	30,66	30,43	30,61	30,57
	PROJ II	391,04	393,45	391,51	392,00
	PROJ III	4243,70	4247,23	4273,67	4254,87

Tabela 3 - Resultados dos testes com 4 computadores (em segundos)

		Rodada 3			
	Qtde de Computadores	4			
	Qtde de Núcleos	8			
		Teste 1	Teste 2	Teste 3	Média
Projeto	PROJ I	15,82	15,78	15,59	15,73
	PROJ II	202,43	201,15	207,43	203,67
	PROJ III	2225,61	2235,71	2210,47	2223,93

Tabela 4 - Resultados dos testes com 8 computadores (em segundos)

		Rodada 4			
	Qtde de Computadores	8			
	Qtde de Núcleos	16			
		Teste 1	Teste 2	Teste 3	Média
Projeto	PROJ I	8,27	8,12	8,19	8,19
	PROJ II	103,71	105,85	108,86	106,14
	PROJ III	1138,77	1125,33	1176,73	1146,94

Tabela 5 - Resultados dos testes com 16 computadores (em segundos)

		Rodada 5			
	Qtde de Computadores	16			
	Qtde de Núcleos	32			
		Teste 1	Teste 2	Teste 3	Média
Projeto	PROJ I	4,28	4,23	4,17	4,23
	PROJ II	54,54	55,62	54,25	54,80
	PROJ III	598,56	597,55	588,76	594,96

6 ANÁLISE DOS RESULTADOS

Após a realização dos testes pode-se analisar o comportamento da aplicação *Loki Render* ao renderizar diferentes projetos (cada um com uma especificidade de recurso de renderização) com diferentes quantidades de computadores compondo a *render farm*.

A tabela 6 aponta o tempo médio de renderização de cada projeto de acordo com a quantidade de computadores utilizados. E a tabela 7 exibe a taxa de ganho de desempenho de cada rodada em comparação com a rodada anterior.

Tabela 6 - Tempo médio de renderização

Tempo Médio de Renderização (em segundos)					
Rodada	**1**	**2**	**3**	**4**	**5**
Computadores	**1**	**2**	**4**	**8**	**16**
PROJ I	60,04	30,57	15,73	8,19	4,23
PROJ II	756,14	392,00	203,67	106,14	54,80
PROJ III	8272,84	4254,87	2223,93	1146,94	594,96

Tabela 7 - Taxa de Ganho de Desempenho

Taxa de Ganho de Desempenho em Relação a Rodada Anterior					
Rodada	**1**	**2**	**3**	**4**	**5**
Computadores	**1**	**2**	**4**	**8**	**16**
PROJ I	-	1,96	1,94	1,92	1,94
PROJ II	-	1,93	1,92	1,92	1,94
PROJ III	-	1,94	1,91	1,94	1,93

Evidencia-se assim o ganho de desempenho teorizado por muitos autores a respeito da utilização de uma arquitetura de computação distribuída, neste caso um *cluster* de computadores utilizado com o propósito de renderizar imagens para computação gráfica – uma *render farm*. A taxa de ganho de desempenho para todos os três projetos variou dentro da mesma faixa valor, situando-se próximo de 1,93. Sendo assim, a cada vez que se dobra a quantidade de equipamentos (ou processadores) em uma *render farm* obtém-se um ganho médio que se aproxima muito do dobro de desempenho, como ilustrado na figura 1, através da curva de desempenho (tempo de renderização x quantidade de computadores).

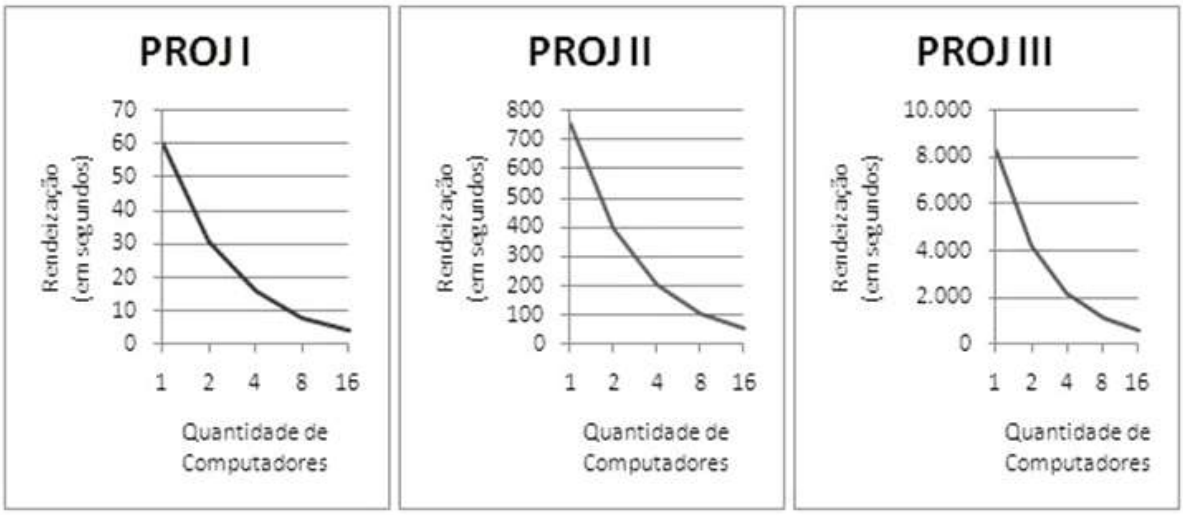

Figura 1 - Gráficos de Desempenho

7 CONCLUSÕES

A arquitetura distribuída é de fato uma solução que potencializa capacidade de processamento. Este estudo analisou o desempenho efetivo proporcionado pelo modelo de computação distribuída aplicado através de um *cluster* com a proposta de renderizar imagens de computação gráfica. A *render farm* que foi estabelecida para esta análise considerou a variação de quantidade de computadores (processadores) que compunham a arquitetura distribuída e a complexidade do projeto a ser renderizado. O controle de distribuição das tarefas foi realizado pelo aplicativo de código aberto *Loki Render*, tendo uma versão cliente em cada computador da arquitetura e uma versão servidora em outro computador que não desempenhou papel de renderizador.

Cinco rodadas de testes foram realizadas com um, dois, quatro, oito e dezesseis computadores sendo utilizados a cada rodada, respectivamente. A cada rodada também foram feitas três medições em diferentes dias para se obter um tempo médio de renderização. Após a coleta dos dados demonstrou-se que o ganho de desempenho proporcionado pela *render farm* atingiu uma taxa média de 1,93 e que esta taxa não variou significativamente com o incremento de computadores nem com a complexidade do projeto renderizado.

Por isso, conclui-se que a computação distribuída, de fato, proporciona ganhos proporcionais a quantidade de computadores que compõem esta arquitetura. Outro ponto a ser observado é a continuidade do estudo com um cenário mais agressivo em termos de quantidade de computadores, a fim de averiguar o ponto em que a gestão por parte da aplicação que distribui as tarefas se torna mais lenta que a resposta dos clientes, verificando se a curva de desempenho se inverte, demonstrando o limite prático de computadores para a *render farm*.

REFERÊNCIAS BIBLIOGRÁFICAS

BLENDER FOUNDATION. **Página oficial da *Blender Foundation***. *Disponível em: http://www.blender.org. Acesso em 12/12/2010.*

BRITO, Allan. **Blender 3D: Guia do Usuário.** 4 ed. Novatec, 2010.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. ***Distributed Systems - Concepts and Design***, 4 ed. Prentice Hall, 2005.

DANTAS, Mário. **Computação Distribuída de Alto Desempenho: redes, *clusters* e *grids* computacionais**. Rio de Janeiro: Axcel Books do Brasil, 2005.

FOLEY, James D.; DAM, Andries van; HUGHES, John. ***Computer Graphics: Principles and Practices***. Addison-Wesley. 1995.

LOKI RENDER. **Página oficial do desenvolvedor do aplicativo Loki Render**. *Disponível em: http://loki-render.berlios.de. Acesso em: 16/11/2010.*

OGATA, Hiroaki; YANO, Yoneo. ***How Ubiquitous Computing can support language learning***. In: KEST, p.1-6, 2003.

PATOLI, Zeeshan. ***How to Build an Open Source Render Farm Based on Desktop Grid Computing*.** Springer eBook, 2008.

PEDROSO, Edson Tessarini. **Segurança em grades computacionais**. Campinas: UNICAMP. Dissertação (mestrado profissional), defesa em julho, 2006. *Disponível em: http://libdigi.unicamp.br/document/?code=vtls000414121. Acesso em 12/12/2010.*

SAHA, D; MUKHERJEE, A. ***Pervasive Computing: A Paradigm for the 21st Century***. IEEE Computer. India: Indian Institute of Management Calcutta – IIM-C, março, 2003.

SATYANARAYANAN, M. ***Pervasive Computing: Vision and Challenges***. IEEE Personal Comunications. New York, v.4, n.8, agosto, 2001.

TANENBAUM, Andrew S.; STEEN, Maarten van. ***Distributed Systems: Principles and Paradigms.*** 2 ed. Prentice Hall, 2007.

TANENBAUM, Andrew S. **Redes de Computadores.** 3 ed. Rio de Janeiro: Campus, 1999.