6.
Resultados

Todas as técnicas apresentadas podem ser vistas como melhorias nos processos de produção de imagens sintéticas. No entanto, faz-se necessário um conjunto de testes para avaliar as efetivas contribuições de desempenho. Dessa forma, este capítulo aborda a metodologia, os testes e os resultados.

6.1.1. Metodologia de Teste de Desempenho

Para a realização dos testes foi desenvolvido um software auxiliar que analisa os dados produzidos pelo gerenciador. Os dados utilizados para as análises são: Uso de *CPU*; Tempo de Renderização de um *Frame*; e o tempo de renderização de uma tarefa (conjunto de *frames*).

Além disso, os testes foram divididos em categorias, de forma a avaliar o desempenho das técnicas do capítulo 3, demonstrando o ganho obtido pelo renderizador. Utilizou-se, para fins de comparação, o renderizador comercial Chaos Group VRay, utilizando como sistema de renderização o *Ray tracing* com o uso de *Irradiance Map*, tomados de 15 em 15 frames, além do renderizador utilizado no trabalho de Barringer R. et al (2014), no qual é utilizado Intel Embree como uma alteração na geração das *BVHs* de forma a melhorar a performance no traçado de raios (essa modificação foi feita dentro do código da própria biblioteca). O renderizador utilizado foi disponibilizado pelos mantenedores do mesmo, visto que o compilador utilizado pelo trabalho é o Intel C++ Compiler XE 14, que difere do utilizado pela tese, o Visual C++ 2013. É sabido que o primeiro é mais eficiente em arquiteturas Intel.

Os testes de desempenho do gerenciador foram divididos em dois grupos. O primeiro não utiliza a otimização integrada, já o segundo a utiliza. Para esses testes foram avaliados os gerenciadores comerciais Autodesk Backburner e Thinkbox Deadline (Thinkbox, 2008) (que apesar do nome não utiliza *deadlines* para a renderização). Ambos são essencialmente similares, no entanto, para o segundo foi criado um pequeno script Python (utilizando a API *Event Plugins*, *Jobs* e *Monitor*) para que, através do sistema de gerenciamento de *cluster* HPCPack da Microsoft, fosse possível adicionar nós no cluster, apesar desse modelo necessitar de uma infraestrutura de comunicação através de VPN, conforme a figura 51. Assim, pode-se avaliar uma forma de transbordo de processamento mais automatizada. Outro teste realizado em conjunto foi o uso do serviço de MPI do HPCPack para disparo de renderização, funcionalidade já presente no sistema da Microsoft.

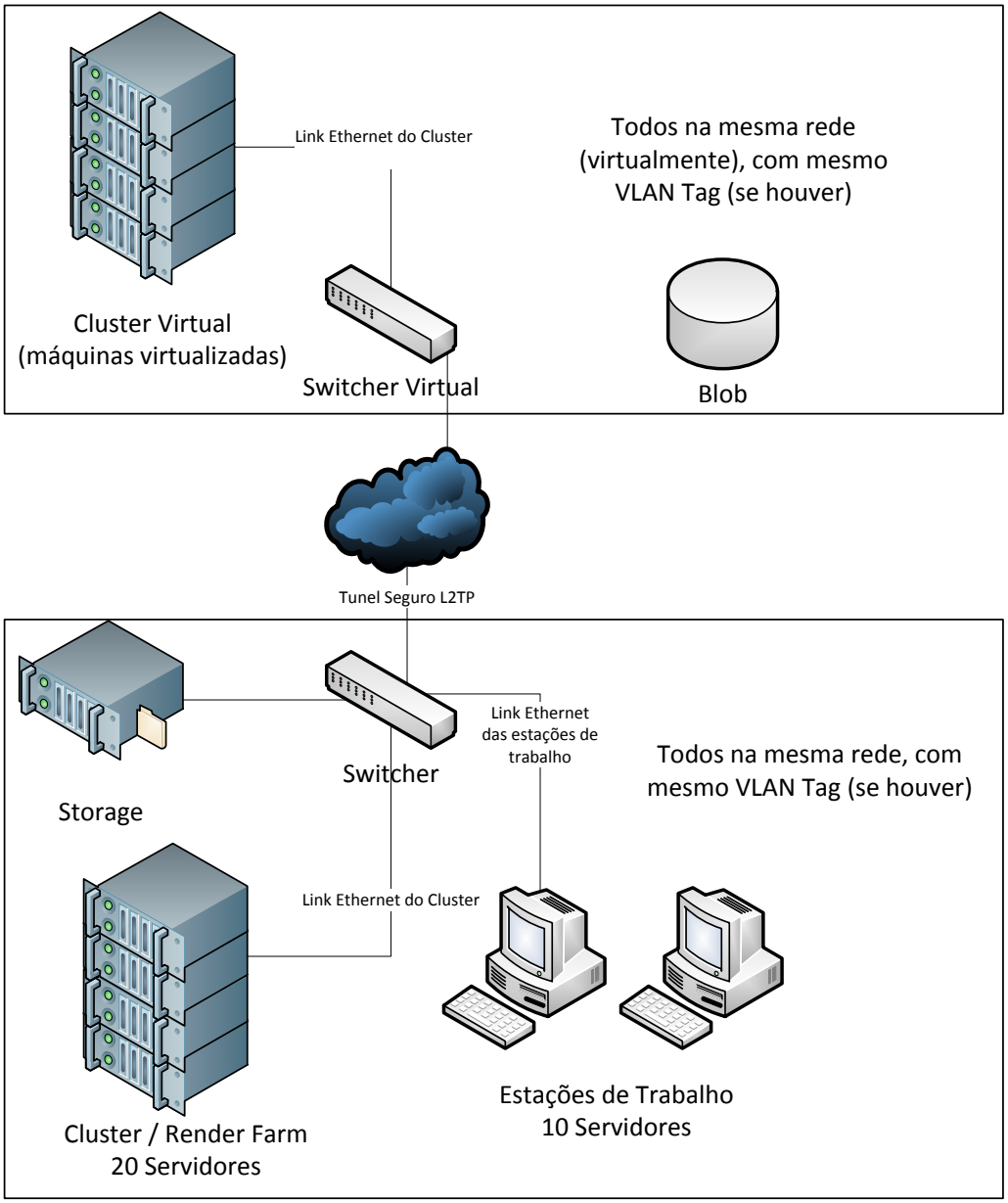


Figura 51: Arquitetura de testes usando o Deadline e o MPI do HPCPack (imagem produzida)

Outra modificação relevante é a adaptação do Chaos Group VRay para a geração de métricas de render, ou seja, uma "*Performance Image*" simplificada, pois utilizando o SDK do VRay não é possível computar muitas informações. No caso, a PI gerada, ao invés de gerar dados como consumo de memória e tempo de processamento de um *pixel*, gera o número de raios gerados na produção do *pixel*. Não é possível no VRay

modificar a forma de subdivisão dos blocos de renderização; no entanto, o dado gerado pode ser usado para uma simples estimativa do custo do *frame*. Essa informação combinada com o tempo gasto para gerar um determinado *frame* permite adaptar o sistema de predição do gerenciador para que ele solicite a alocação de mais servidores na nuvem, porém, o número de servidores é calculado usando um fator aplicado sobre o número de raios de cada *pixel*, gerando o tempo por *pixel*. Esse fato é dado pela média de tempo por raio (equação 15). Observa-se que assim é extremamente complicado estimar o custo computacional dos materiais.

$$t(pixel) = CountRays(pixel) \cdot \frac{TotalTime(frame)}{TotalRays(frame)}$$

Equação 15: Estimador de tempo por pixel no VRay

Na equação 15, $CountRays$ é o número de raios emitidos no *pixel*, $TotalTime$ é o tempo total do frame e $TotalRays$ é o número total de raios emitidos no *frame*. Essa medida não considera as questões de tempo de carga de dados, consultas a licenças, inicializações e outros. Porém, com esses dados, já pode-se estimar o tempo necessário para o processamento de um *frame*.

Todos os testes realizam a renderização no tamanho HDTV. Além disso, foram escolhidas duas cenas para análise, “San Miguel” (Barringer R. et al, 2014) e uma cena de produção da novela Avenida Brasil, por ser mais próxima do uso real. Em ambas as cenas foram utilizadas câmeras *pinhole* com efeito de desfoque usando o canal de profundidade. Essa necessidade é para evitar artefatos específicos de implementação quando comparado com os sistemas mais complexos.

A cena San Miguel (figura 52a) é clássica nos trabalhos de computação gráfica, ela possui 7,8 milhões de triângulos e 6,7 milhões de vértices, além de 256 mapas de texturas, totalizando 80 Megabytes, que se tornarão cerca de 400 Megabytes de texturas e apenas a luz do sol. Já

a cena de Avenida Brasil (figura 52b), que apresenta apenas o Maracanã como elemento a ser renderizado, possui cerca de 25 milhões de triângulos, 3 Gigabytes de texturas, que produzem cerca de 14 GBytes de texturas 600 luzes de área e 1600 luzes pontuais.

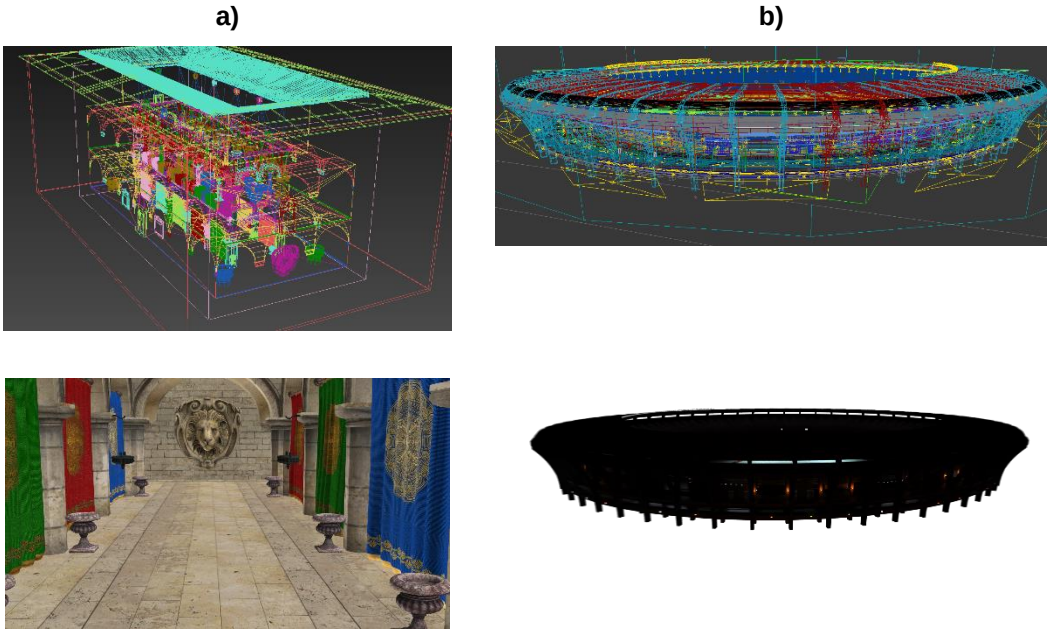


Figura 52: Cenas do teste **a)** cena “San Miguel” (Barringer R. et al, 2014) **b)** Cena do Maracanã, gerado pelo autor para a novela Avenida Brasil (TV Globo, 2012). Imagem reproduzida sob a política de “fair use”.

Para automatizar a análise dos testes, foi desenvolvido um sistema de avaliação de teste. Esse sistema possui uma interface de entrada de linha de comando, a qual permite a criação de arquivos de planilhas eletrônicas (.csv) para a geração de relatórios. A Figura 53 mostra a interface de entrada desse sistema, disponível pelo comando `bullfor-analysis -h`.

```
Bullfor Performance Analyser

This software is part of Rodrigo Marques Thesis

You must type one of the folowing options:

        -s : Compute Statistics (Mean, StdDev, CI)
        -i <id> : Set The Scene ID
        -r <id> : Run Test on Farm, using the Scene ID
        -opts "<options>" : Set the Render options tag (same as Web UI)
        -h : Show Options
```

Figura 53: Opções de linha de comando do sistema de avaliação

Todas as cenas seguem rigorosamente os mesmos parâmetros de navegação e limites de parâmetros de renderização, como profundidade de subraios e amostras de Monte Carlo[9]. Além disso, efeitos complexos e dependentes de customização não estão presentes nos testes (fumaça, fogo, nuvens e *subsurface scaterring*).

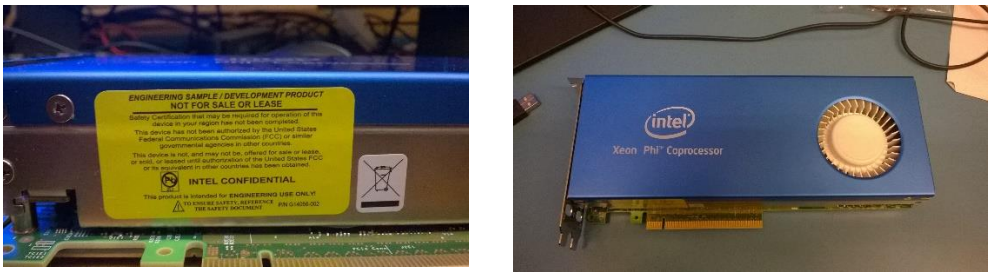
Cada rodada de simulação é realizada a análise estatística, essa produz os dados de tempo máximo, médio, mínimo, desvio padrão e intervalo de confiança.

O hardware utilizado para os testes em nível local foi um computador **(A)** com 64 Gbytes de memória RAM, CPU Dual Intel Xeon 16/32 Cores @ 3.2 GHz, placa de vídeo de NVidia Quadro K6000 com 6 Gbytes dedicados e placa de rede 10 Gbps. Os nós de computação CPU possuem a configuração **(B)** de 64 Gbytes de memória RAM, CPU Dual Intel Xeon 16/32 Cores @ 3.2 GHz e placa de rede 10 Gbps (figura 54b). Ainda foram utilizados em testes específicos um servidor com a configuração (A) e uma placa Xeon Phi 5100 (figura 54a), disponibilizada pela Intel. Outro hardware específico utilizado foi o Nvidia VCA, que possui 256 Gbytes de memória RAM, CPU Dual Intel Xeon 16/32 Cores @ 3.2 GHz, 8 placas de vídeo de NVidia Kepler K20 com 12 Gbytes dedicados e placa de rede 10 Gbps. Não foi possível testar dois servidores Nvidia VCA (figura 54c), no entanto, o sistema suporta intercomunicação através de uma conexão Infiniband 56 Gbps (SDR via

[9] Profundidade Máxima = 8, Número de Amostra de MC = 256

QSFP+). A Figura 54 mostra o sistema do VCA, os Servidores e o Storage.

a)



b)

c)

Figura 54: Hardware empregado nos testes, cedidos pela TV Globo para testes. a) Intel Xeon Phi b) Nós de renderização c) Storage Equalogic e NVidia VCA

Os hardwares dos servidores empregados na nuvem possuem o hardware com 16 Gbytes de memória RAM, CPU Intel Xeon 8/8 Cores @ 2.1 GHz, sem placa de vídeo e placa de rede 10 Gbps. A nuvem utilizada para os testes foi o Azure.

Os testes foram realizados com o sistema operacional[10] Microsoft Windows 7 recém instalado, em servidores locais e Windows Server 2008 nos nós e máquinas nas nuvens, a fim de reduzir ao máximo as trocas de contextos e intromissões de outros aplicativos. Além disso, o sistema de teste foi inicializado com prioridade de tempo real (a mais alta disponível aos usuários de sistema Microsoft Windows). O *cluster* local possui 20 máquinas com a configuração (B) em rede 10 Gbps Ethernet, tendo acesso a um *storage* Dell Equalogic de 180 TBytes de capacidade e 40 Gbps Ethernet de banda através um *trunk* de 4 portas 10 Gbps. O protocolo de cópia utilizado no *storage* é CIFS 2.1 que suporta até 250 MBps nos sistemas operacionais Windows 7 e Linux.

Por fim, cada teste foi repetido 3 vezes para avaliar a variação dos parâmetros. O último teste realizado foi o considerado na análise deste capítulo.

6.1.2. Metodologia de Análise Estatística

A análise estatística realizada utiliza à média aritmética (Equação 16) como estimador de média, pois essa é não viciada (i.e. no sentido de não ser enviesada) e consistente. O estimador de variância (desvio-padrão) escolhido foi a variância amostral (Equação 17), por ser não viciada e consistente.

$$\mu = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Equação 16: Estimador de media

[10] O Nvidia VCA utiliza uma versão modificada do CentOS e a placa Xeon Phi utiliza uma versão embarcada do sistema operacional linux. Em ambos os casos foi utilizada uma adaptação do sistema com Mono e foram recompiladas as partes que utilizam código CUDA ou C.

$$\sigma^2 = S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

Equação 17: Estimador de variância

Para a comprovação de validade de dados, foi utilizado o nível de significância de 5%, o qual é padrão utilizado pela indústria médica (considerada rigorosa) e, portanto, de alta confiabilidade. Para a composição do intervalo de confiança foi utilizada a curva *t-student* (Equação 18), devido à forma de estimação dos parâmetros μ e σ.

$$IC(\mu,(1-\alpha)\%) = \bar{X} \pm Z_{n,\frac{\alpha}{2}} \frac{S}{\sqrt{n}}$$

Equação 18: Cálculo do intervalo de confiança

Para evitar uma poluição visual nas tabelas apresentadas neste capítulo, optou-se por colocar as tabelas completas (com média, desvio-padrão e intervalo de confiança) no site (http://bullfor.azurewebsites.net/tese/). Dessa maneira, as tabelas deste capítulo possuem apenas a média da informação observada.

6.1.3. Testes

As subseções posteriores descrevem e apresentam os resultados dos testes realizados, bem como uma conclusão sobre o mesmo.

Apenas o teste 6.3.6 compara o desempenho entre a *CPU* e *GPU*, ou seja, retirando testes com rasterização, os demais testes não utilizaram a *GPU* com acelerador de renderização. O motivo dessa decisão reside em realizar os testes de forma mais justa e mais próxima da realidade possível. Além disso, no equipamento disponível no momento desse trabalho, 6 GB de memória foram insuficientes para renderizar a cena de Avenida Brasil na *GPU* sem a redução de tamanho de texturas. Além disso, nenhum dos sistemas testados utiliza *GPU* para renderização final. Ainda, os clusters e nuvens utilizados não utilizam *GPUs* para aceleração.

6.1.4. Desempenho do Renderizador

Para a avaliação do desempenho do renderizador foram realizados três modos de teste:

1. Renderização de Conjunto de *Frames* apenas com estimativas.

2. Renderização de Conjunto de *Frames* com PI já computadas a 1 e 2 *frames* de distância (anterior e posterior).
3. Renderização de Conjunto de *Frames* com PI do próprio *frame*

Usando a metodologia apresentada, os seguintes resultados foram obtidos (Eixo Vertical o tempo de renderização (s) e horizontal o *frame*):

Modo 1:

Teste 1.1: Cena San Miguel

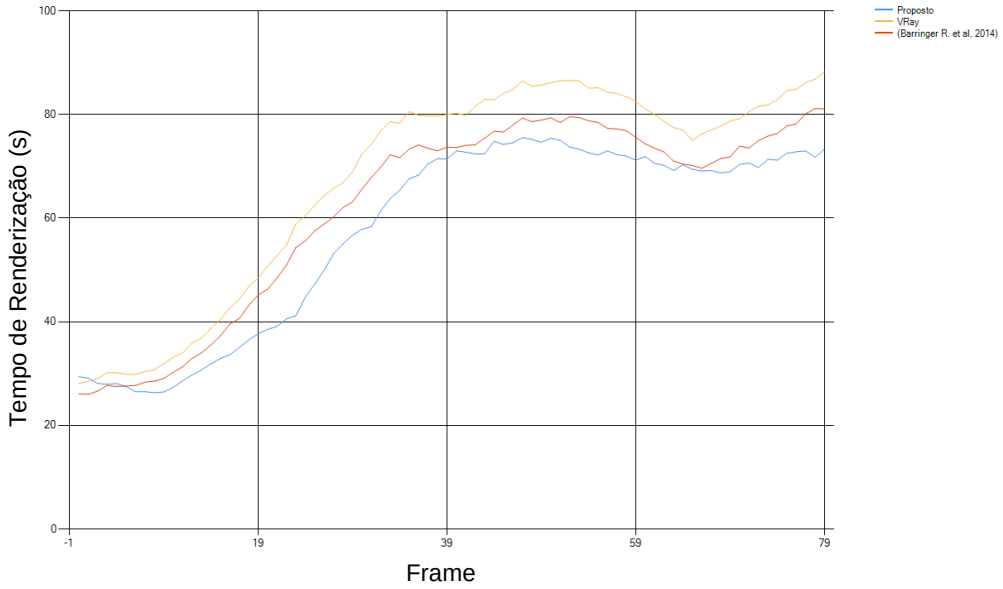


Figura 55: Resultado Gráfico de Frames – Teste 1.1

Observando os dados obtidos com as médias temos a tabela 1:

Tabela 1: Resultados do tempo de renderização do Teste 1.1

	Proposto	**VRay**	**(Barringer R. et al, 2014)**
Média	57.57	67.28	61.87
Máximo	75.42	88.12	81.22
Mínimo	26.45	28.17	25.92
Desvio	18.32	20.52	18.86
Ganho Médio		17.31%	8.09%

Utilizando apenas as estimativas o sistema já produziu um ganho alto quando comparado com o VRay. Nesse caso, o fato da cena possuir apenas a luz do sol melhora as estimativas de custo dos *shaders*.

A comparação com os demais sistemas é adequada, pois, esse tipo de iluminação é muito recorrente em cenas virtuais e de simples implementação (quando não considera efeitos atmosféricos).

Teste 1.2: Cena Avenida Brasil

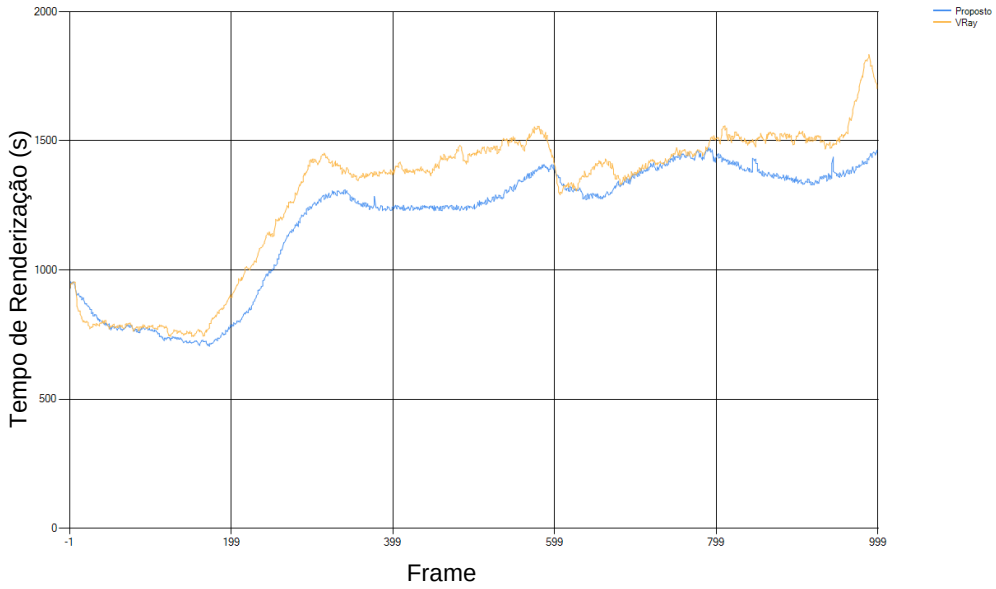


Figura 56: Resultado Gráfico de Frames – Teste 1.2

Observando os dados obtidos com as médias temos a tabela 2:

Tabela 2: Resultados do tempo de renderização do Teste 1.2

	Proposto	**VRay**
Média	1189.14	1292.47
Máximo	1464.83	1828.72
Mínimo	707.19	739.42
Desvio	241.97	279.84
Ganho Médio		8.47%

Observa-se que o ganho médio foi de 8.47%, considerando uma cena de alta complexidade. Deve-se observar que nesse caso, ambos os

renderizadores são mantidos abertos para os frames, no entanto, o VRay não otimiza dados sequenciais.

Modo 2:

Teste 2.1: Cena San Miguel – Um Frame de distância

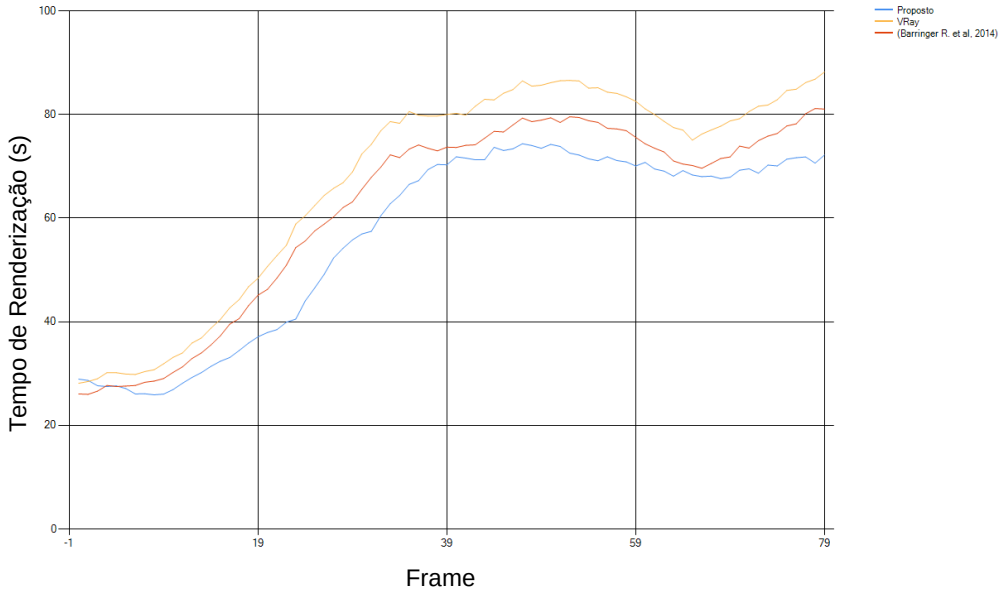


Figura 57: Resultado Gráfico de Frames – Teste 2.1

Observando os dados obtidos com as médias temos a tabela 3:

Tabela 3: Resultados do tempo de renderização do Teste 2.1

	Proposto	**VRay**	**(Barringer R. et al, 2014)**
Média	56.67	67.28	61.87
Máximo	74.23	88.12	81.22
Mínimo	26.03	28.17	25.92
Desvio	18.04	20.52	18.86
Ganho Médio		19.18%	9.82%

Com apenas um frame de distância, a estimativa fica mais apurada, acelerando o processo de renderização.

Teste 2.2: Cena Avenida Brasil – Um Frame de distância

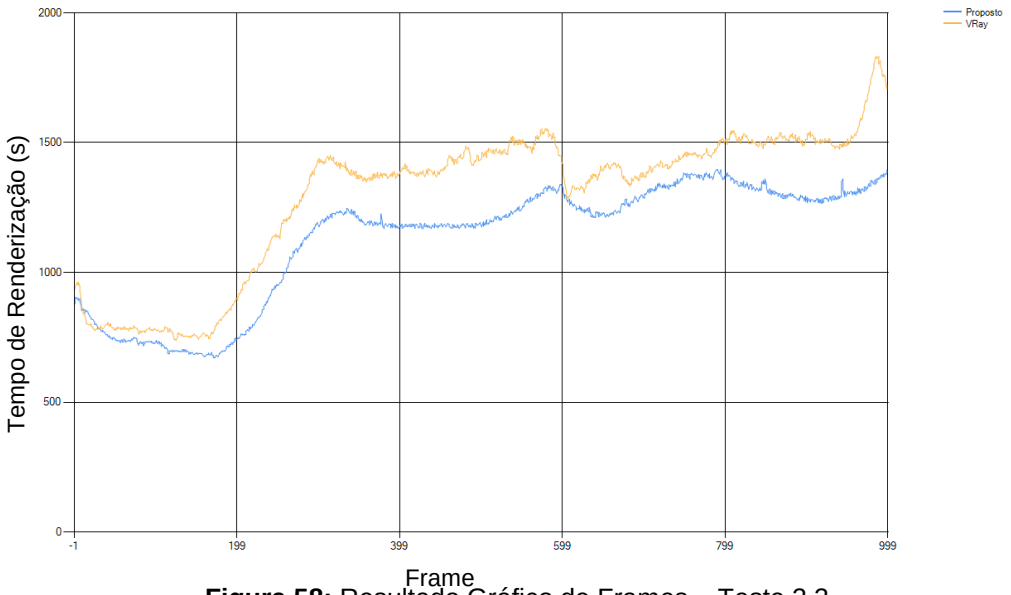


Figura 58: Resultado Gráfico de Frames – Teste 2.2

Observando os dados obtidos com as médias temos a tabela 4:

Tabela 4: Resultados do tempo de renderização do Teste 2.2

	Proposto	**VRay**
Média	1130.33	1292.48
Máximo	1394.65	1309.84
Mínimo	672.93	740.30
Desvio	229.97	280.09
Ganho Médio		14.17%

Observa-se que, com dados mais corretos, o ganho sobe para 14%, mantendo as características da cena. Ainda, há alguns momentos em que o VRay é superior, provavelmente devido à acomodação do processo de subdivisão.

Teste 2.3: Cena San Miguel – Dois Frame de distância

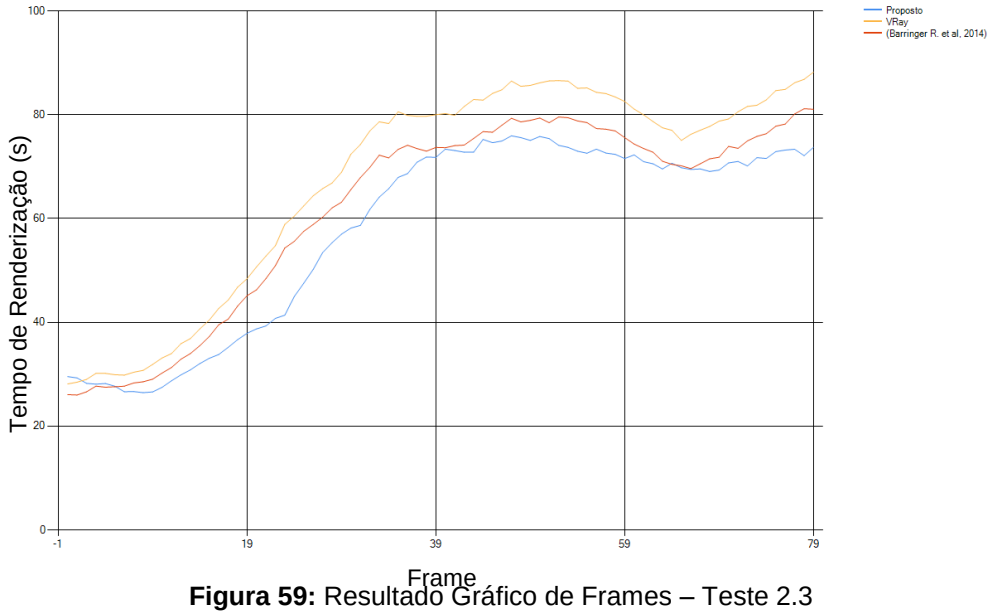


Figura 59: Resultado Gráfico de Frames – Teste 2.3

Observando os dados obtidos com as médias temos a tabela 5:

Tabela 5: Resultados do tempo de renderização do Teste 2.3

	Proposto	**VRay**	**(Barringer R. et al, 2014)**
Média	57.88	67.28	61.87
Máximo	75.81	88.12	81.22
Mínimo	26.59	28.17	25.92
Desvio	18.42	20.52	18.86
Ganho Médio		16.70%	7.53%

Como esperado, o ganho caiu, inclusive abaixo do desempenho utilizando apenas a estimativa. Avaliando a cena, o motivo pode ser o erro de compensação da cena, que possui poucos frames e a câmera se move rapidamente em um movimento em Z, que impede uma melhor compensação usando o canal de velocidade. Esse comportamento pode ocorrer sempre que a animação possuir esse tipo de movimento, visto que a compensação é no plano 2D da imagem.

Teste 2.4: Cena Avenida Brasil – Dois Frames de distância

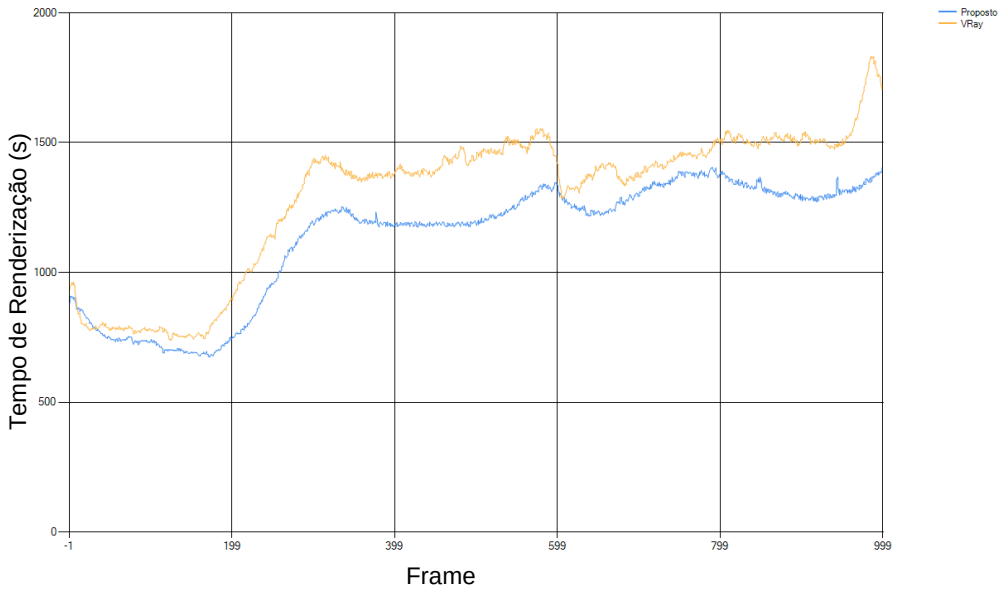


Figura 60: Resultado Gráfico de Frames – Teste 2.4

Observando os dados obtidos com as médias temos a tabela 6:

Tabela 6: Resultados do tempo de renderização do Teste 2.4

	Proposto	**VRay**
Média	1136.15	1292.48
Máximo	1401.84	1830.64
Mínimo	676.40	740.30
Desvio	231.16	280.09
Ganho Médio		13.58%

Nesse teste houve uma pequena queda, o que, provavelmente, pode ser caracterizada pelo erro de estimativa gerado pela compensação usando o canal de velocidade, porém, por ser uma cena muito mais complexa e o movimento muito mais suave, mesmo com a queda, o resultado supera o conseguido apenas com as estimativas.

Modo 3:

Teste 3.1: Cena San Miguel

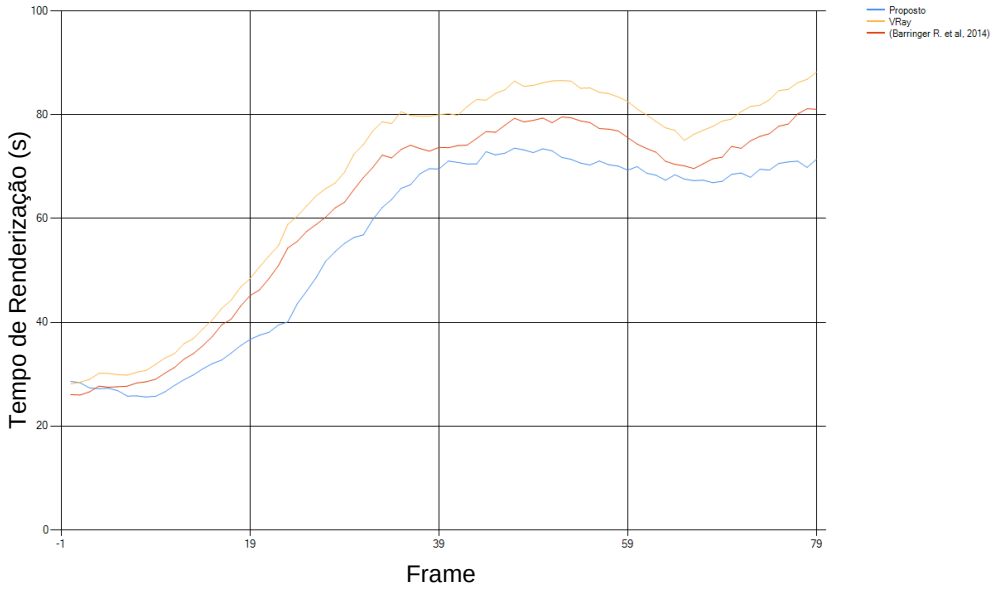


Figura 61: Resultado Gráfico de Frames – Teste 3.1

Observando os dados obtidos com as médias temos a tabela 7:

Tabela 7: Resultados do tempo de renderização do Teste 3.1

	Proposto	**VRay**	**(Barringer R. et al, 2014)**
Média	56.07	67.28	61.87
Máximo	73.44	88.12	81.22
Mínimo	25.75	28.17	25.92
Desvio	17.84	20.52	18.86
Ganho Médio		19.46%	11.00%

Utilizando a análise exata do frame em questão o ganho é amplificado, esse é o ganho máximo que o sistema pode proporcionar para essa cena. Lembrando que esse valor é um ganho médio.

Teste 3.2: Cena Avenida Brasil

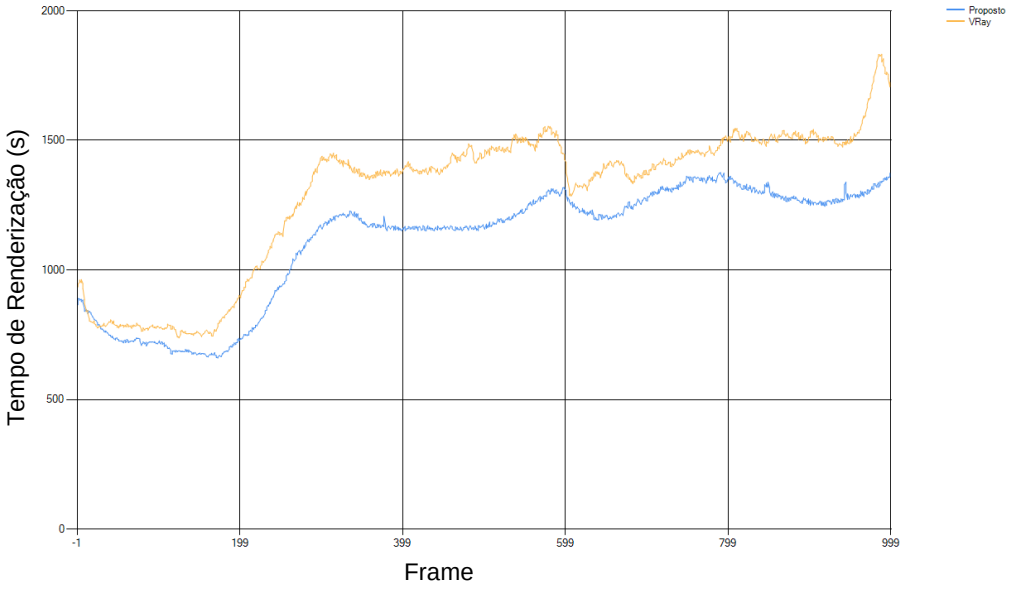


Figura 62: Resultado Gráfico de Frames – Teste 3.2

Observando os dados obtidos com as médias temos a tabela 2:

Tabela 8: Resultados do tempo de renderização do Teste 1.2

	Proposto	**VRay**
Média	1112.85	1292.48
Máximo	1373.09	1830.64
Mínimo	662.52	740.30
Desvio	226.42	280.09
Ganho Médio		15.97%

De forma similar, o teste mostra um ganho com o uso de um dado de *Performance Image* mais preciso.

O resultado gráfico obtido na cena San Miguel pode ser observado na tabela 9:

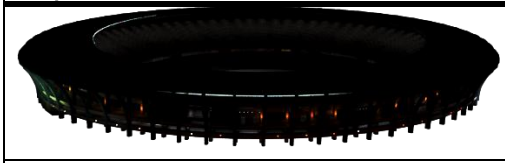
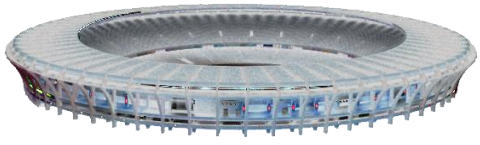
Tabela 9: Resultados Gráficos Teste 1, 2 e 3 – Cena San Miguel

Proposto	**VRay**	**(Barringer R. et al, 2014)**
		
	Diferença VRay	**Diferença (Barringer R. et al, 2014)**
		
	Diferença do SNR	**Diferença do SNR**
	+0.75 db	+ 4 db

Nas imagens da tabela 9 fica claro que a diferença de sinal/ruído do renderizador de Barringer et al (2014) é muito maior do que o VRay e o Proposto. Isso se deve à implementação muito simples da iluminação do sol, o que impacta diretamente na performance, deixando uma imagem mais uniforme. No entanto, o trabalho de Barringer et al (2014) não foca nesse tipo de algoritmo. Porém, é importante a comparação de desempenho com o mesmo para evidenciar os ganhos. É importante notar que a comparação de desempenho entre renderizadores é muito complexa, pois, basta a mudança da forma de se calcular qualquer processo interno, como luz ou Monte Carlo, que os ganhos são completamente alterados.

O resultado gráfico obtido na cena de Avenida Brasil pode ser observado na tabela 10:

Tabela 10: Resultados Gráficos Teste 1, 2 e 3 – Cena Avenida Brasil

Proposto	**VRay**
	
	Diferença VRay
	

	Diferença do SNR
	+0.7992 db

No caso da cena de Avenida Brasil, há uma diferença entre as imagens, mesmo com o mesmo *gamma*. A imagem produzida pelo sistema proposto é mais escura e possui menor nível de espalhamento de cor. Isso se deve pela diferença de algoritmo e principalmente pela forma de perda da energia da luz, visto que o VRay é não conservativo. No entanto, o sistema respeitou a quantidade de raios e a forma de distribuição de Monte Carlo foi realizada de forma similar.

6.1.5. Qualidade de Redução de Técnica

O teste definido para a avaliação da redução de técnica utilizou a cena San Miguel com a redução de um e dois níveis de qualidade, ou seja, realizou a redução de amostras[11] e técnicas (*Path Tracing* para *Ray Tracing*, *Ray Tracing* para *VRL* e *Ray Tracing* para Rasterização[12]), simultaneamente. Além disso, o renderizador não possuía a PI, visto que na redução de técnica, apenas as partes mantidas não sofrem alteração. Nesses testes não foi utilizado nenhum tipo de cache, para avaliar o ganho direto da técnica.

Teste 4.1: Redução

[11] MC = 128

[12] Somente elementos que são compatíveis

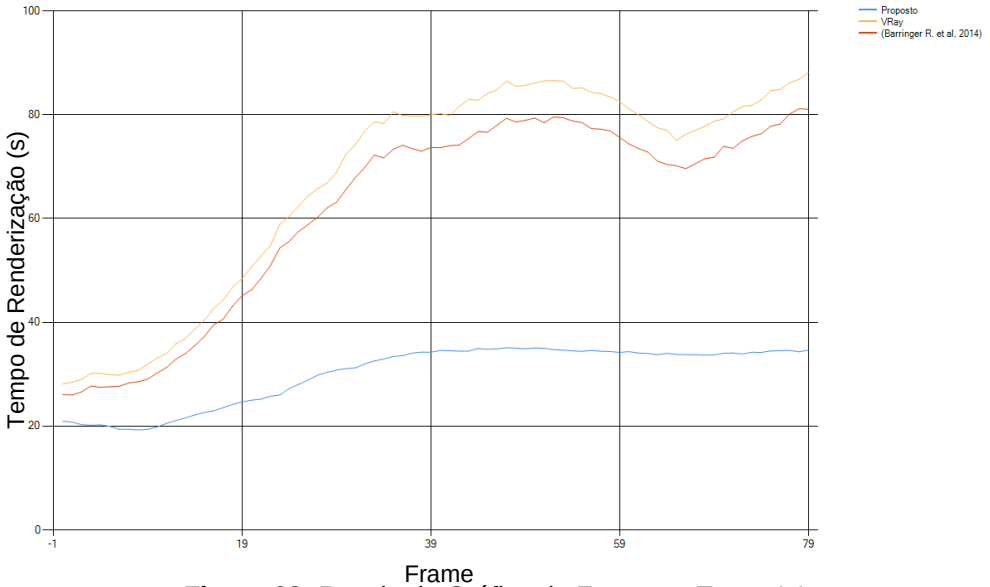


Figura 63: Resultado Gráfico de Frames – Teste 4.1

Observando os dados obtidos com as médias temos a tabela 11:

Tabela 11: Resultados do tempo de renderização do Teste 4.1

	Proposto	**VRay**	**(Barringer R. et al, 2014)**
Média	30.07	67.28	61.87
Máximo	35.08	88.12	81.22
Mínimo	19.36	28.17	25.92
Desvio	5.72	20.52	18.86
Ganho Médio		120.10%	102.73%

O resultado gráfico obtido pode ser observado na tabela 12:

Tabela 12: Resultados Gráficos Teste 4.1

Proposto	**VRay**	**(Barringer R. et al, 2014)**
		
	Diferença VRay	**Diferença (Barringer R. et al, 2014)**
		
	Diferença do SNR	**Diferença do SNR**
	+1.2 db	+4 db

O sistema reduziu para a técnica de rasterização da cena, assim, foi utilizado *Shadow Maps* ao invés de traçado de raio para a geração das

sombras. Isso cria uma suavização não real na borda da sombra, contudo, pode ser utilizada como alternativa. Nas diferenças das imagens fica bem nítido que há um erro na zona de alta iluminação. Observa-se que o ganho é significativamente maior, isso devido a usar uma técnica muito mais eficiente. Ainda, é percebido que o tempo fica em torno de 25 segundos, isso por que é adicionado ao tempo de carga dos elementos, do contrário o tempo fica ainda menor. Dessa forma, a variação do tempo de renderização reduz também.

Na comparação com o VRay, os efeitos da amostragem ficam muito evidentes, pois na rasterização isso não é empregado. Porém, a diferença visual é pequena, bastando um ajuste de cor.

6.1.6. Desempenho do Gerenciador

Os testes realizados com o Gerenciador buscaram avaliar o ganho do mesmo em três situações, sem o uso da colaboração entre nós:

1. **Sem Restrição**: Nesse caso o gerenciador procura otimizar o envio e o compartilhamento de dados, favorecendo a coerência temporal.
2. **Restrição Temporal**: Foi estipulado o prazo de vinte horas para a finalização. Foram disponibilizadas dez máquinas de trabalho (*workstation*) adicionas, ou seja, usadas somente nesses casos. Dessa forma, o prazo é alcançável sem redução de técnicas.
3. **Restrição de Recursos**: Foi estipulado o prazo de onze horas (normalmente o período de interjornadas de trabalho) para a finalização. Foram disponibilizadas dez máquinas de trabalho (*workstation*) adicionas, ou seja, usadas somente nesses casos. Dessa forma, o prazo não é alcançável sem redução de técnicas.

A cena escolhida foi a de Avenida Brasil, por possuir um número muito maior de frames e ser muito mais complexa. Além disso, foi utilizado o VRay e o renderizador deste trabalho para fins de comparação do escalonamento com o uso de PI. No caso 2 e 3 o VRay não se aplica para a análise, pois não é possível controlar sua execução, mesmo utilizando dados da execução do *plugin* criado, o gerenciador apenas antecipa a alocação de recursos. O *plugin* consegue prever o tempo de computação de forma bastante razoável, porém, o uso do mesmo não se mostrou muito útil, devido a total falta de opções para controlar o VRay.

Modo 1:

Teste 5.1: Renderizador Próprio

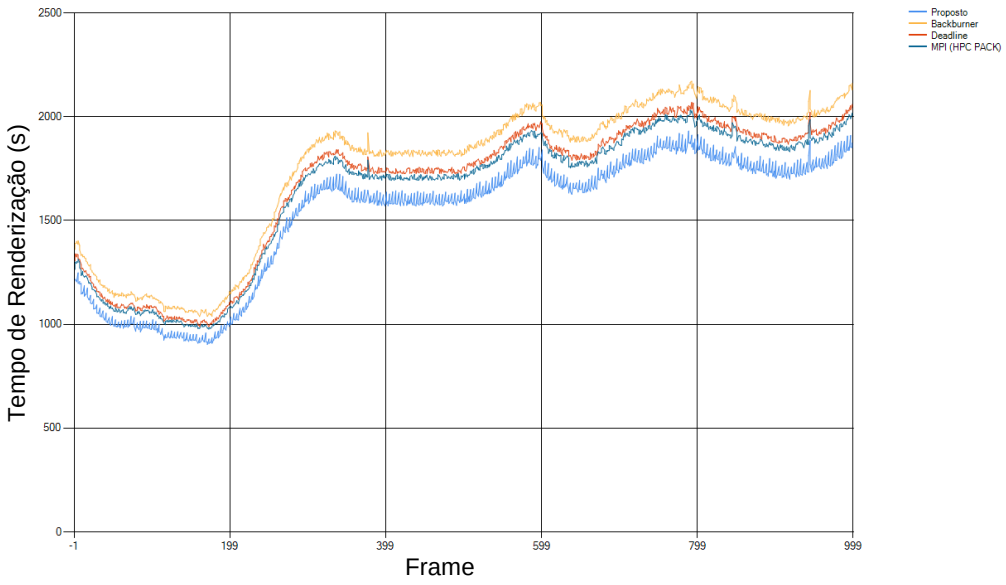


Figura 64: Resultado Gráfico de Frames – Teste 5.1

Observando os dados obtidos com as médias temos a tabela 13:

Tabela 13: Resultados do tempo de renderização do Teste 5.1

	Proposto	Backburner	Deadline	MPI (HPC PACK)
Média	1529.97	1748.89	1669.06	1637.59
Máximo	1924.27	2163.12	2055.59	2026.00
Mínimo	903.65	1041.91	993.80	976.96
Desvio	311.99	355.98	339.57	333.37
Ganho Médio		14.38%	9.21%	7.04%

O comportamento com o gerenciador é bastante diferente da renderização sequencial, isso se deve ao fato do espalhamento dos frames não ser uniforme e nem sequencial. Outro ponto é que os gerenciadores mantiveram o desempenho padrão do renderizador. Houve um aumento significativo no tempo devido principalmente ao acesso de dados remoto e tempo de gerência. O ganho em relação ao Backburner é alto, isto porque a cada renderização ele reenvia todo o arquivo base para o servidor, gerando um grande atraso.

O MPI (HPC PACK) que executa paralelamente a chamada do renderizador nos nós teve bom desempenho, porém boa parte se deve à forma que o renderizador reaproveita o dado (usando a PI do frame anterior). O mesmo comportamento foi observado no Deadline (Thinkbox, 2007).

Teste 5.2: VRay

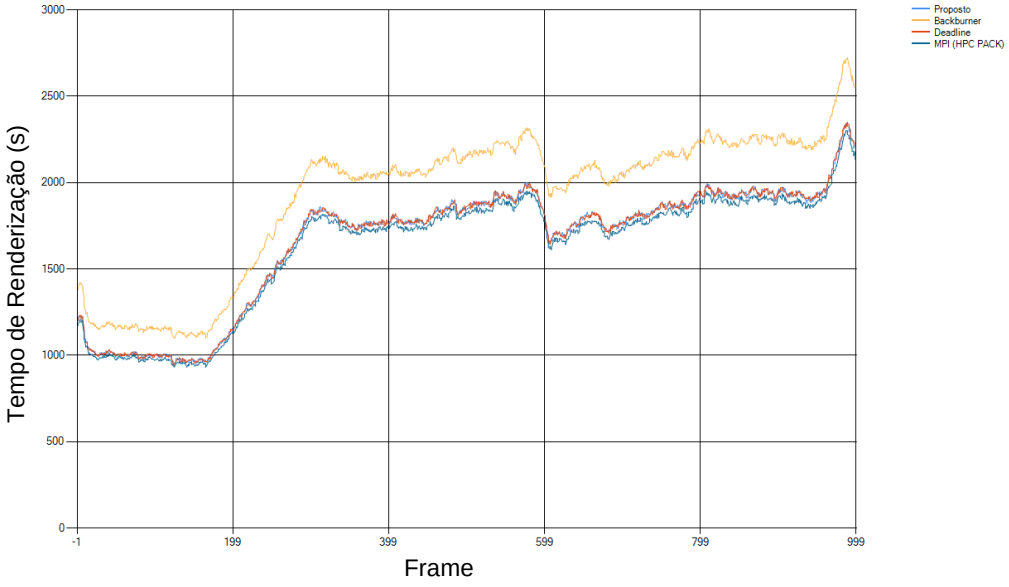


Figura 65: Resultado Gráfico de Frames – Teste 5.2

Observando os dados obtidos com as médias temos a tabela14:

Tabela 14: Resultados do tempo de renderização do Teste 5.2

	Proposto	Backburner	Deadline	MPI (HPC PACK)
Média	1657.82	1919.52	1657.80	1622.98
Máximo	2336.18	2707.72	2338.94	2301.82
Mínimo	948.61	1099.14	942.56	924.19
Desvio	359.38	416.00	359.20	351.46
Ganho Médio		15.87%	0.02%	-3.12%

No caso do VRay, houve ganho em relação ao backburner, porém, apesar de conseguir prever os gargalos, devido às restrições do VRay, o ganho não conseguiu superar o MPI (que praticamente não realiza a gerencia da renderização, apenas o disparo) e o Deadline (Thinkbox, 2007). Outro ponto é que usando o *Plugin*, o dado só fica disponível para o gerenciador depois da conclusão da cena e salvamento de todos os dados (devido a forma que os *plug-ins* operam no 3D Studio Max). Isso causou uma perda significativa de tempo para a análise do gerenciador.

Isso também indica que o gerenciador deve ser capaz de atuar nos renderizadores, do contrário, pouco se pode conseguir, visto que o tempo real de processamento fica a encargo do renderizador.

Modo 2:

Teste 6.1: Renderizador Próprio

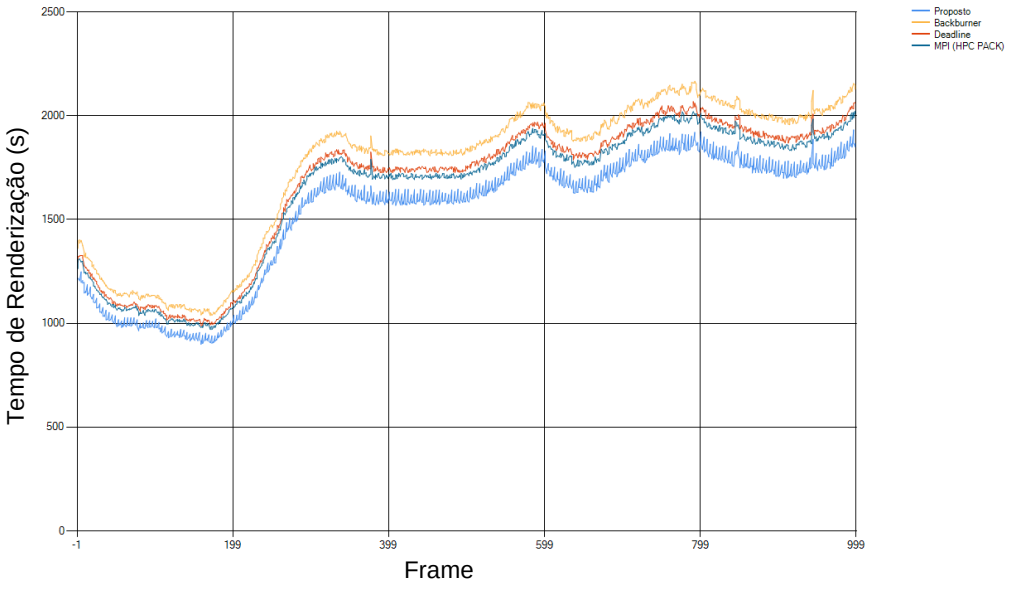


Figura 66: Resultado Gráfico de Frames – Teste 6.1

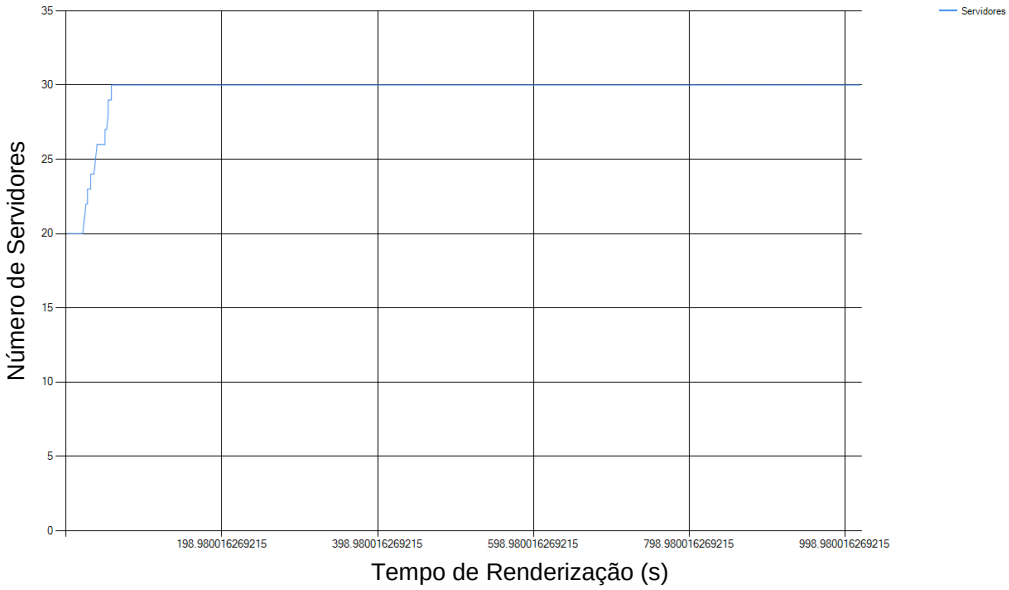


Figura 67: Resultado Gráfico do Número de Servidores Alocados (Servidor x Tempo) – Teste 6.1

Observando os dados obtidos com as médias temos a tabela 15:

Tabela 15: Resultados do tempo de renderização do Teste 6.1

	Proposto	Backburner	Deadline	MPI (HPC PACK)
Média	1529.50	1748.47	1669.13	1637.26
Máximo	1934.20	2156.35	2056.80	2016.86
Mínimo	902.76	1043.68	992.07	974.14
Desvio	312.07	355.60	339.70	333.17
Ganho Médio		14.0%	9.1%	7.0%

Foi observado o comportamento similar ao teste 5.1, sendo que houve pouca variação no tempo e nos ganhos, isso devido à similaridade das máquinas envolvidas. Ainda é percebido o comportamento de alocação de nós conforme as estatísticas ficam prontas.

Modo 3:

Teste 7.1: Renderizador Próprio

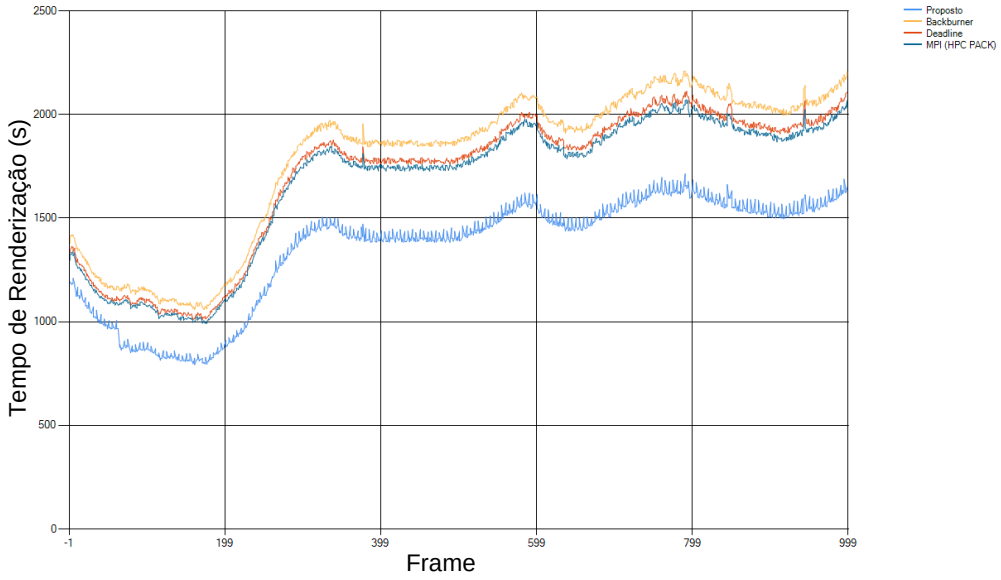


Figura 68: Resultado Gráfico de Frames – Teste 7.1

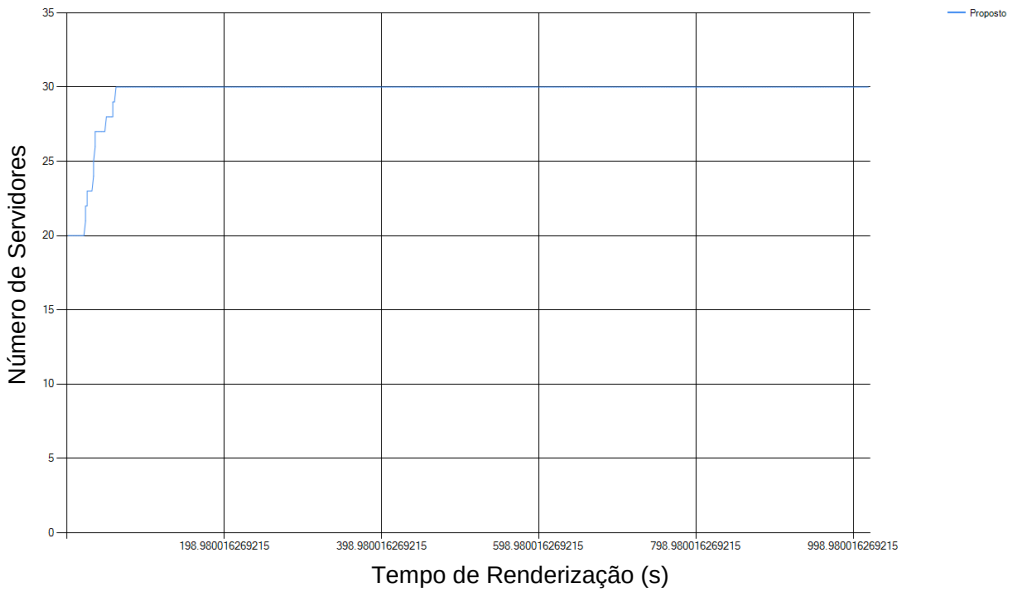


Figura 69: Resultado Gráfico do Número de Servidores Alocados (Servidor x Tempo) – Teste 7.1

Observando os dados obtidos com as médias temos a tabela 16:

Tabela 16: Resultados do tempo de renderização do Teste 7.1

	Proposto	**Backburner**	**Deadline**	**MPI (HPC PACK)**
Média	1353.32	1783.72	1702.70	1670.34
Máximo	1709.87	2202.13	2109.75	2064.38
Mínimo	795.67	1060.11	1011.33	994.48
Desvio	265.93	363.08	346.58	339.71
Tempo Total	12.53 Horas	16.51 Horas	15.76 Horas	15.46 Horas

Observa-se que o sistema logo identifica o problema para finalização da sequência, adotando uma redução de amostragem e a mantendo até o fim da renderização. Esse comportamento demonstra que são necessários um conjunto razoável de frames para se estimar a condição de não conformidade com as metas.

6.1.7. Desempenho do Gerenciador com Colaboração

Os testes com colaboração foram os mesmos realizados em 6.3.3, no entanto, não foi utilizado o renderizador VRay para esses testes pelo fato do mesmo não usufruir desse recurso.

O Modo 3 também não foi realizado por alterar muito a característica da renderização, gerando muito ruído.

Modo 1:

Teste 8.1: Renderizador Próprio

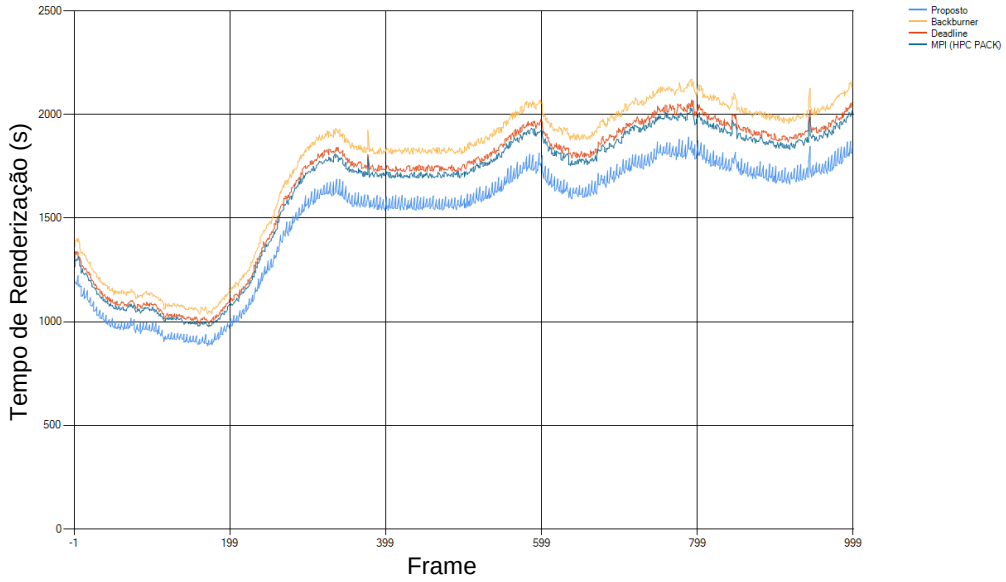


Figura 70: Resultado Gráfico de Frames – Teste 8.1

Observando os dados obtidos com as médias temos a tabela 17:

Tabela 17: Resultados do tempo de renderização do Teste 8.1

	Proposto	Backburner	Deadline	MPI (HPC PACK)
Média	1498.42	1748.89	1669.06	1637.59
Máximo	1884.59	2163.12	2055.59	2026.00
Mínimo	885.02	1041.91	993.80	976.96
Desvio	305.55	355.98	339.57	333.37
Ganho Médio		16.79%	11.41%	9.30%

Observando os resultados, a colaboração teve um ganho de 2% em média na renderização global, dessa forma, pode ser utilizado para o melhor funcionamento do gerenciador. Além disso, ela pode evitar perda de dados.

Modo 2:

Teste 9.1: Renderizador Próprio

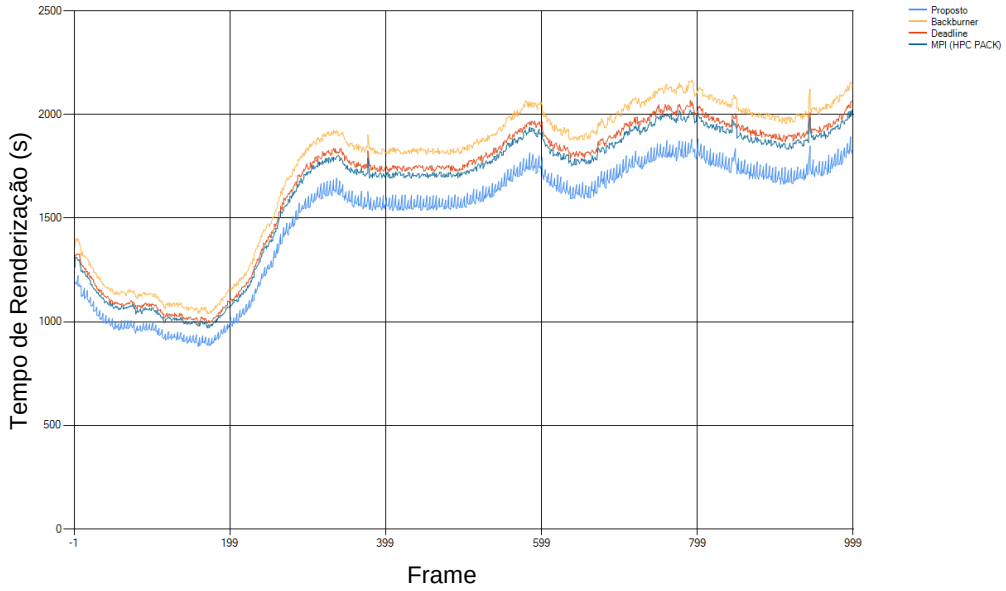


Figura 71: Resultado Gráfico de Frames – Teste 9.1

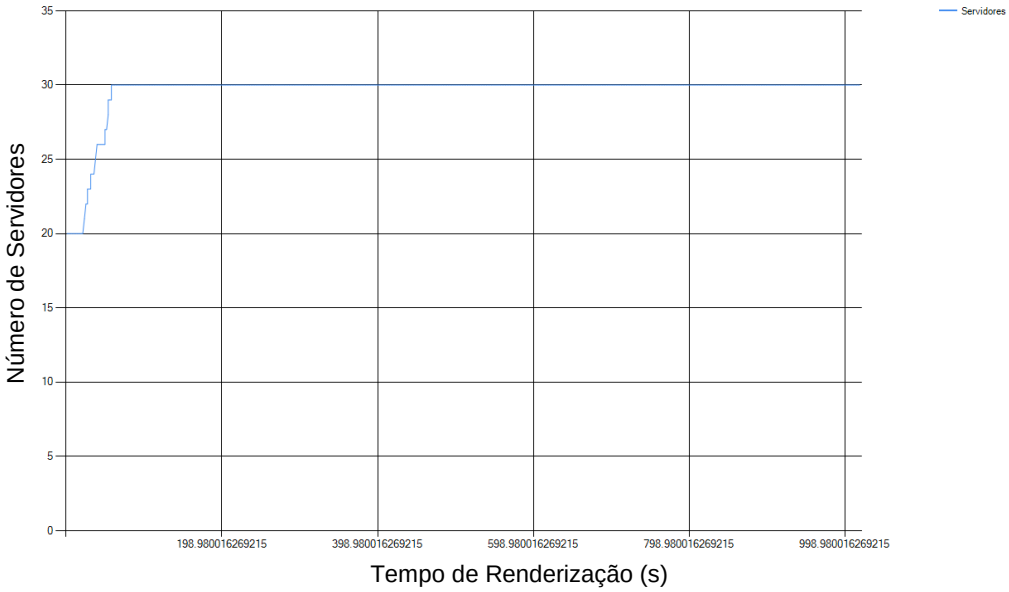


Figura 72: Resultado Gráfico do Número de Servidores Alocados (Servidor x Tempo) – Teste 9.1

Observando os dados obtidos com as médias temos a tabela 18:

Tabela 18: Resultados do tempo de renderização do Teste 9.1

	Proposto	**Backburner**	**Deadline**	**MPI (HPC PACK)**
Média	1497.96	1748.47	1669.13	1637.26
Máximo	1894.32	2156.35	2056.80	2016.86
Mínimo	884.14	1043.68	992.07	974.14

Desvio	305.63	355.60	339.70	333.17
Ganho Médio		16.73%	11.44%	9.30%

De forma similar, o recurso proporcionou um ganho de 2% em média. O gerenciador não utiliza esses dados para validar aumento de servidores, assim, não influencia na alocação.

6.1.8. Desempenho de Transbordo para Nuvem

Os testes realizados com transbordo para a nuvem buscaram avaliar o ganho do mesmo em duas situações, com o uso da colaboração entre nós, sendo:

1. **Restrição Temporal**: Foi estipulado o prazo de cinco horas para a finalização, assim é necessário a alocação de mais servidores para atender a demanda. O gerenciador Deadline e HPC PACK (utilizando o script criado por este trabalho) poderia utilizar até 50 máquinas, as quais já estavam em modo de hibernação. A alocação foi feita de forma sequencial, por não possuir uma métrica de implementação viável para o mesmo. Não foi possível utilizar o Backburner.
2. **Restrição de Recursos**: Foi estipulado o prazo de sete horas e monetariamente o equivalente a 240 horas por núcleo para o gerenciador deste trabalho. O Deadline e o HPC PACK receberam 30 servidores adicionais (o que é mais do que 240 horas podem proporcionar).

De forma similar aos testes anteriores, a cena escolhida foi a de Avenida Brasil. O renderizador VRay não foi utilizado por restrição no número de licenças disponíveis.

Modo 1:

Teste 10.1: Renderizador Próprio

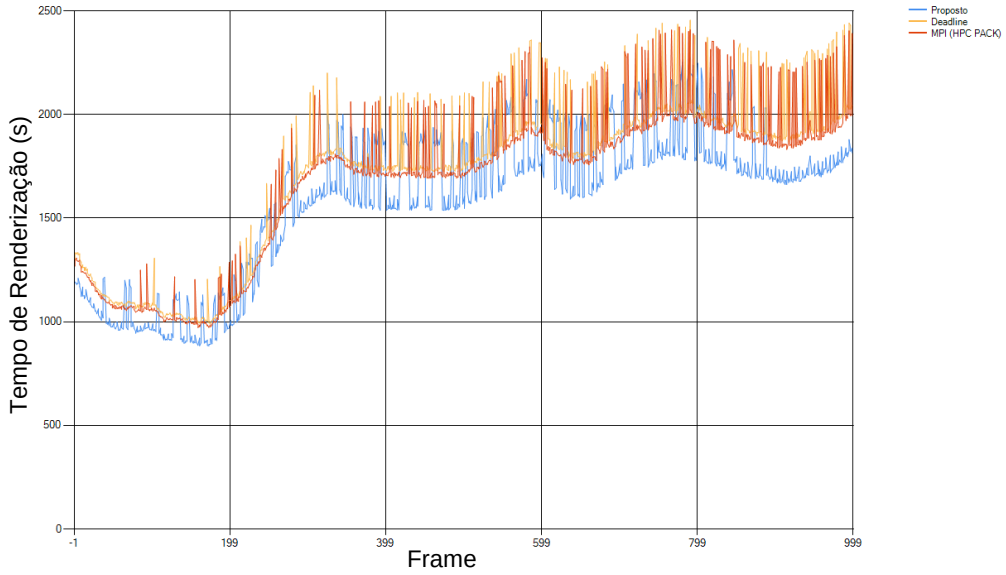


Figura 73: Resultado Gráfico de Frames – Teste 10.1

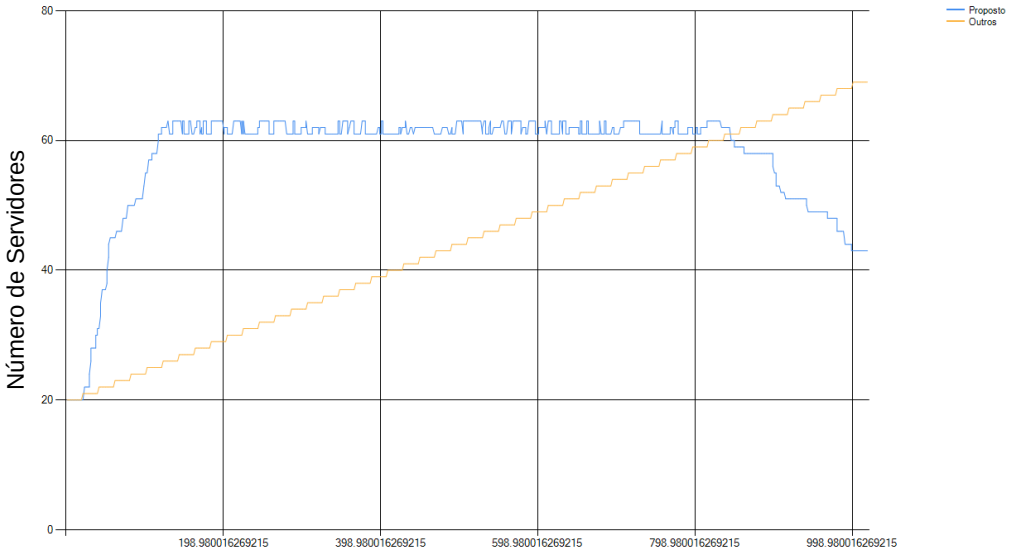


Figura 74: Alocação de Servidores – Teste 10.1

Observando os dados obtidos com as médias temos a tabela 19:

Tabela 19: Resultados do tempo de renderização do Teste 10.1

	Proposto	**Deadline**	**MPI (HPC PACK)**
Média	1595.91	1709.45	1677.10
Máximo	2227.95	2327.92	2411.63

Mínimo	887.86	993.48	974.18
Desvio	345.40	368.65	361.15
Tempo Total	5.3 Horas	8.9 Horas	8.4 Horas

O resultado gráfico fica confuso devido à diferença entre os hardwares locais e remotos. Em termos de ganho por *frame*, não é possível avaliar corretamente entre os sistemas, visto que a forma de alocar as máquinas não é a mesma. Porém, a estratégia adotada proporcionou um tempo muito próximo do estipulado para o sistema atender. É possível que aumentando mais os parâmetros de erro o resultado seja atendido, visto que não foram utilizadas todas as máquinas, representando uma economia.

Modo 2:

Teste 11.1: Renderizador Próprio

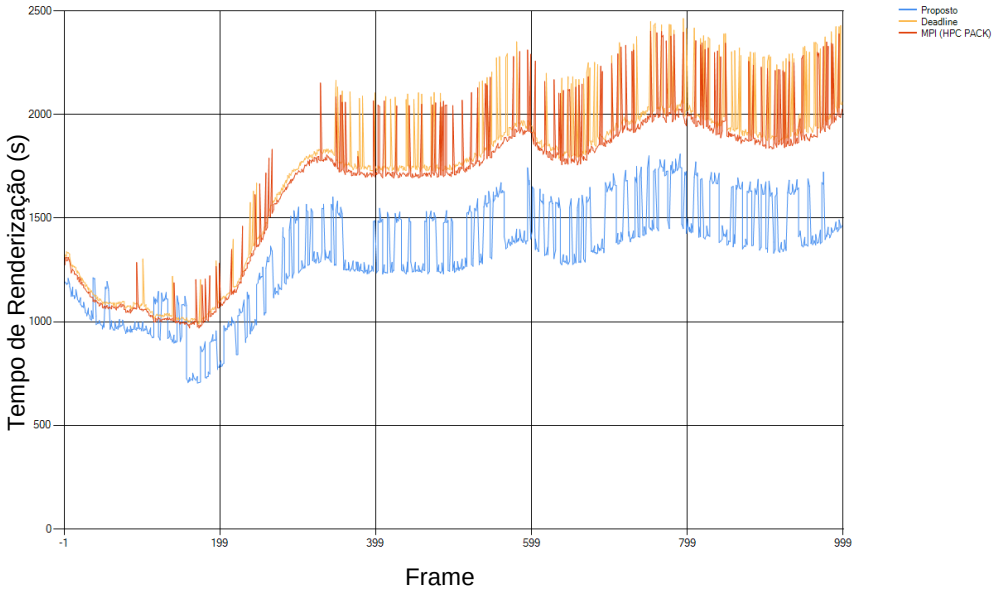


Figura 75: Resultado Gráfico de Frames – Teste 11.1

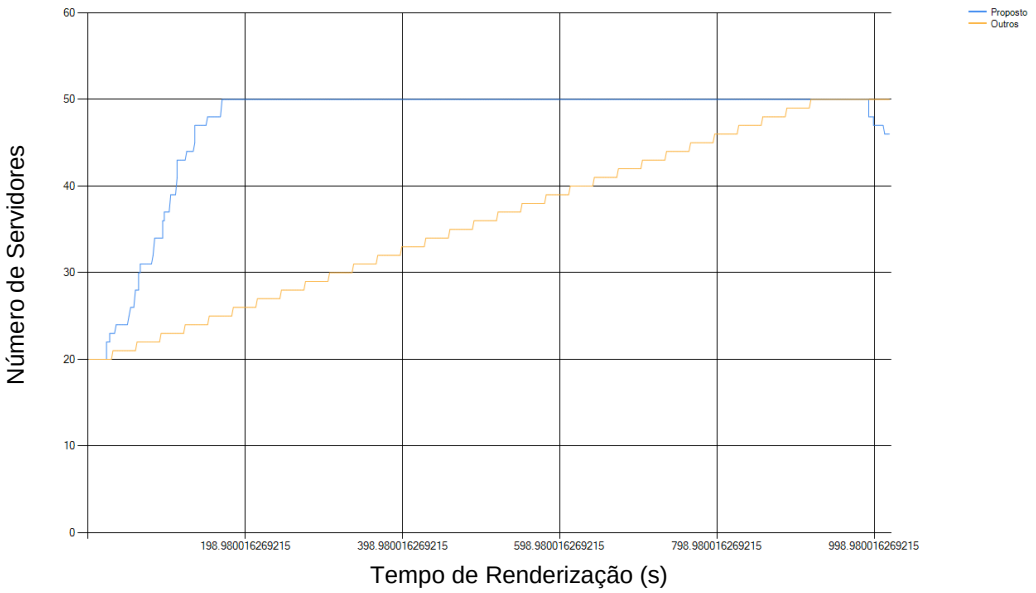


Figura 76: Alocação de Servidores – Teste 11.1

Observando os dados obtidos com as médias temos a tabela 20:

Tabela 20: Resultados do tempo de renderização do Teste 11.1

	Proposto	**Deadline**	**MPI (HPC PACK)**
Média	1323.26	1706.46	1706.46
Máximo	1815.94	2317.27	2267.61
Mínimo	735.02	990.02	974.31
Desvio	330.85	366.04	359.19
Tempo Total	7.35 Horas	13.5 Horas	13.3 Horas

Nesse caso o gerenciador avalia a necessidade de alocar nós e logo identifica que não será possível atender as demandas de processamento. Nesse ponto o sistema inicia uma redução de qualidade (amostras) para atender os requisitos temporais, atendendo razoavelmente o prazo estipulado.

6.1.9. Desempenho GPU x CPU

Os testes realizados comparando o renderizador do trabalho utilizando CPU (Embree) x GPU (OptiX) para o algoritmo de Path Tracing. Nesse caso, foram realizados 2 tipos de testes usando a cena San Miguel.

1. **Servidores Convencionais**: A cena foi renderizada utilizando os servidores convencionais com configuração **(A)**.

2. **Restrição de Recursos**: A cena foi renderizada utilizando os acelerados Nvidia GRID (VCA) e Xeon Phi (Configuração **(A)**).

Modo 1:

Teste 12.1: Renderizador Próprio

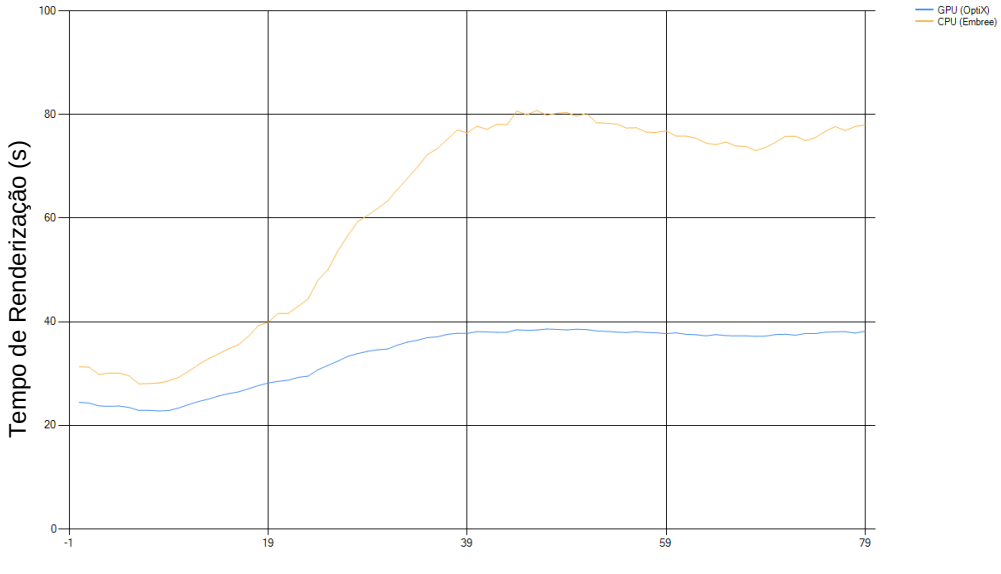


Figura 77: Resultado Gráfico de Frames – Teste 12.1

Observando os dados obtidos com as médias temos a tabela 21:

Tabela 21: Resultados do tempo de renderização do Teste 12.1

	GPU (Optix)	CPU (Embree)
Média	33.57	61.52
Máximo	38.58	80.69
Mínimo	22.86	28.06
Desvio	5.72	19.59
Ganho Médio		80.56%

Nos testes com os servidores convencionais observa-se que a *GPU* aumenta o ganho em 80% em relação a *CPU*, pois a mesma possui

muito mais núcleos que a *CPU*, além da cena não apresentar grande complexidade de rebatimentos. Nota-se que a comparação é justa, pois ambos os frameworks, Optix e Embree, são otimizados para os respectivos hardwares.

Modo 2:

Teste 13.1: Renderizador Próprio

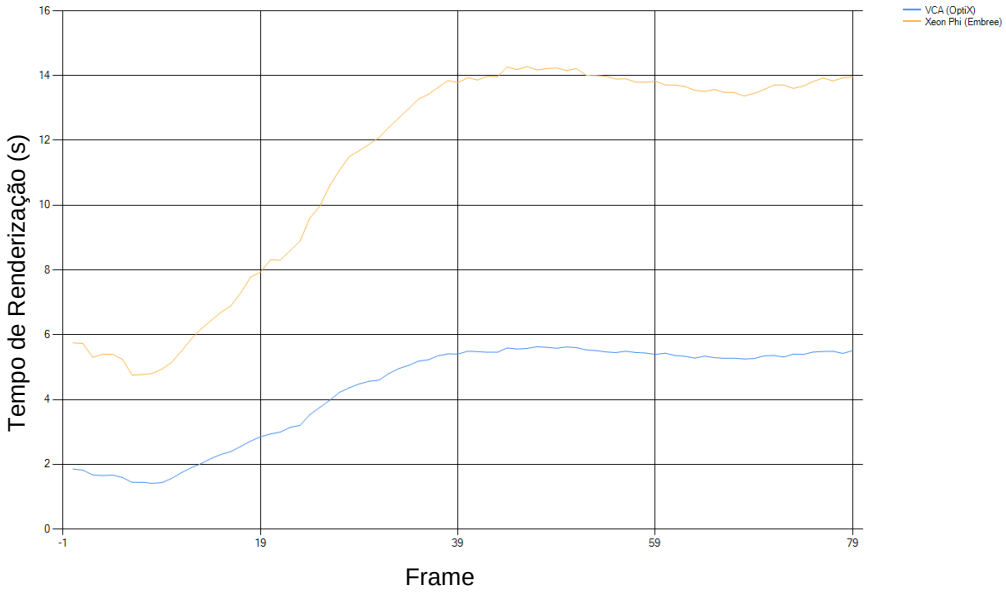


Figura 78: Resultado Gráfico de Frames – Teste 13.1

Observando os dados obtidos com as médias temos a tabela 22:

Tabela 22: Resultados do tempo de renderização do Teste 13.1

	GPU (Optix)	CPU (Embree)
Média	4.28	11.25
Máximo	5.62	14.26
Mínimo	1.432	4.76
Desvio	1.52	3.43
Ganho Médio		170.20%

Claramente o VCA apresenta um desempenho extremamente superior ao Xeon Phi, isso devido ao grande número de cores no sistema e o excelente sistema de barramento do equipamento. Porém, apesar de não atingir o nível do VCA, o Xeon Phi mostrou que tem um excelente ganho em relação aos processadores convencionais. No entanto, não foi

possível observar o ganho em escala, ou seja, adicionando mais placas Xeon Phi, o que com o VCA ocorre naturalmente.

6.1.10. Finalização

Observando os resultados, podemos avaliar que as heurísticas de estimativas são razoáveis, pois permitiram melhorar o desempenho do renderizador. Outro ponto avaliado é que a compensação das imagens de performance podem incorrer em erros quando há deslocamento da câmera no eixo Z. Isso pode ser melhor resolvido quando o movimento é mais lento. Ainda, a alocação dinâmica de servidores permitiu um bom resultado no atendimento da restrição temporal. A redução de técnica traz uma possibilidade interessante de atender os gargalos emergenciais de renderização, mesmo sendo necessário uma intervenção artística para suavizar os ruídos.

Todos os testes utilizaram os mapas de velocidade como forma de realizar a compensação das imagens de performance. Porém, foi realizado um teste com a cena 1 em que se utiliza o algoritmo de fluxo ótico para gerar os mapas de compensação, sendo assim, é adicionado um erro na geração da informação de velocidade da cena. Isso, em média, reduziu a performance em 2.3% e aumentou o nível de processamento do processo de gerência em 14.7%. Neste teste os vetores de velocidade associados a cada pixel obtidos do mapa de velocidade são muito mais precisos do que os obtidos pelo fluxo ótico. Isto é devido ao fato dos vetores de velocidade serem dados próprios do modelo 3D renderizado e não “estimativas” da imagem renderizada. Ademias cenas que contém frequências muito altas o fluxo ótico tem dificuldades de estimar estes vetores com precisão. Um outro ponto é que o objetivo é, por projeto de arquitetura, reduzir a quantidade de frames adjacentes para melhorar explorar a coerência temporal, principalmente em cenas muito dinâmicas. Por isso, nos testes de validação, essa metodologia não foi usada. No entanto, trata-se de um recurso poderoso e que tem validade em processamentos de outro tipo de trabalho, como

em *encoders*. Assim, mesmo não apresentando um resultado melhor do que os mapas de velocidade, ele foi mantido e também enfatizado como importante tópico para futuras investigações

Os testes com o NVidia VCA mostraram que, em média, o tempo de renderização de um frame (passível de renderização usando *GPU*) tem uma redução de 15 vezes no tempo de renderização. Assim, do ponto de vista do gerenciador, o ganho serve para escalar o sistema com o servidor de mais potência e o alocando para o frame de maior complexidade. Os testes do Xeon Phi mostraram um ganho de 5 vezes na renderização dos frames, mesmo sendo inferior ao VCA (170%). Assim, é importante notar que tanto o Optix quanto o Embree utilizam da melhor forma possível o hardware. Por isso, as diferenças de desempenhos não são tão exageradas, pois são comparados algoritmos paralelos e otimizados em ambas plataformas.