Uma ferramenta para auxiliar na análise de desempenho de *cluster* de computadores com memória distribuída

Felippe Vieira Zacarias[1], **Josemar Rodrigues de Souza**[1]

[1] Universidade do Estado da Bahia (UNEB)
Núcleo de Arquitetura de Computadores e Sistemas Operacionais (ACSO)
Rua Silveira Martins, 2555, Cabula.
Salvador-BA, 41195-001
Salvador, Bahia, Brasil

{fvzacarias,josemarsbr}@gmail.com

Resumo. Benchmarks *tem sido desenvolvidos para permitir uma avaliação de desempenho de* cluster *de computadores de maneira padronizada. Contudo, o estabelecimento de modelos ou mecanismos que permitam a análise dos dados obtidos a partir das execuções desses* benchmarks *tornam-se uma tarefa não trivial e muitas vezes esforços na otimização das ferramentas disponíveis atenuam os resultados e análises obtidas. Nesse trabalho é apresentado uma ferramenta que realiza um* benchmark *em* clusters *de memória distribuída, com o qual foi possível perceber no* cluster *utilizado para o teste, que utilizando uma abordagem multi*thread *do* benchmark *HPL obtem-se um desempenho 1,6x melhor que sua versão multi processos e que para a aplicação de processamento de imagem, a versão híbrida que utilize o menor número de processos obtêm os melhores resultados.*

1. Introdução

Uma das principais justificativas para a utilização de computadores paralelos é que o paralelismo é uma forma de evitar o gargalo atingido com os computadores *single core*. A computação de alto desempenho ou *High Performance Computing* (HPC) utiliza super computadores e *clusters* de computadores para resolver problemas que demandam alta capacidade de rescursos computacionais. Neste contexto os *clusters* de computadores tornaram-se a solução de HPC mais usada devido à sua escalabilidade e flexibilidade.

Um *cluster* se caracteriza por ser um conjunto de computadores interconectados através de uma rede para a troca de informação e de recursos, operando sobre o controle de *softwares* específicos. Esse *software* faz com que o conjunto opere como se fosse apenas um único equipamento com capacidade e escalabilidade mais elevada e custo proporcional a capacidade computacional [Cuenca et al. 2005].

Para o usuário é importante selecionar uma configuração de acordo com o as características do conjunto de aplicações que serão executados por ele. Porém é difícil saber isto a *priori*, já que podem ser diferentes aplicações por usuário além de frequentemente terem tempos e frequência de execução desconhecidas. Por isso os *benchmarks*, programas usados para efetuar testes no sistema computacional objetivando avaliar a influência da arquitetura no desempenho, têm sido desenvolvidos para permitir uma avaliação do desempenho do sistema de maneira padronizada baseando-se em características específicas que podem ser mensuradas [Rauber and Rünger 2010].

Por isso uma das razões mais importantes para se medir e avaliar o desempenho dos computadores paralelos é saber se o desempenho atual pode ser melhorado [Zacarias et al. 2013], contudo o estabelecimento de modelos ou mecanismos que permitam a análise dos dados obtidos a partir das execuções desses *benchmarks* tornam-se uma tarefa não trivial.

Neste sentido, este artigo tem como objetivo apresentar uma ferramenta que executa um *benchmark* em *clusters* de memória distribuída de maneira automatizada e transparente ao usuário, fornecendo indicadores de desempenho na arquitetura para uma análise de desempenho. Os indicadores fornecidos pela ferramenta compreendem fatores relacionados a largura de banda, taxa de transferência da interconexão de rede, operações em ponto flutuante e tempo de processamento. O trabalho está estruturado da seguinte forma: na seção 2 serão abordados alguns tipos de *benchmarks* usados em *clusters* de memória distribuída, na seção 3 será abordada a ferramenta proposta e seu funcionamento, na seção 4 e 5 serão abordados respectivamente os resultados experimentas e as conclusões.

2. Benchmarks para Análise de Desempenho

Um *benchmark* é um programa que é usado para efetuar testes de desempenho do sistema computacional, e tem como objetivo avaliar a influência da arquitetura no desempenho. A performance do sistema computacional pode variar significativamente dependendo do programa utilizado. Portanto, o *benchmark* deve ser projetado para fornecer comparações justas e eficazes entre sistemas de computação de alto desempenho [El-Rewini and Abd-El-Barr 2005]. Para o *benchmark* ser significativo, ele deve avaliar o desempenho com fidelidade para o uso pretendido do sistema, para tanto, diferentes *benchmarks* têm sido propostos e usados. Por isso, adequar esta miscelânea de ferramentas acaba muitas vezes desestimulando principalmente os usuários inicantes em seus estudos.

2.1. *Benchmarks* Paralelos

Visando a análise de desempenho de computadores paralelos, esse tipo de *benchmark* é aplicado em máquinas com múltiplos núcleos, processadores ou sistemas constituídos de várias máquinas. Geralmente são associados com as características de desempenho do *hardware* que avaliam, como por exemplo a capacidade de operações de ponto flutuante da CPU.

Um dos mais popular conjuntos de *benchmarks* é o *System Performance Evaluation Cooperation* (SPEC) *benchmark suite* [Rauber and Rünger 2010]. O SPEC é uma corporação formada para estabelecer, manter e apoiar um conjunto padronizado de *benchmarks* relevantes que podem ser aplicados as novas gerações de computadores de alto desempenho [El-Rewini and Abd-El-Barr 2005]. O SPEC MPI2007 é o *benchmark* do SPEC para calcular o desempenho de uma gama de *clusters* em ponto flutuante utilizando MPI.

Outro *benchmark* popular é o *NAS Parallel Benchmarks* (NPB), que é um conjunto de *benchmarks* para avaliação de desempenho de computadores paralelos. Seus problemas são baseados em aplicações e núcleos de aplicações paralelas de dinâmica de fluidos [Pilla 2009]. Seu principal foco era na computação aerospacial, contudo muitos desses *benchmarks* tem uma relevância em várias áreas, sendo que muitas aparecem em várias aplicações do mundo real. O *High Performance Linpack* (HPL) é a principal ferramenta

usada para ranquear os computadores na lista do Top500 [Meuer 2013]. Ele resolve um sistema de equações lineares com precisão dupla em computadores de memória distribuída. Além de fornecer o desempenho da máquina em Flops, sua saída serve tanto para quantificar a precisão da solução quanto o tempo gasto na computação.

2.2. *Benchmarks* para Análise de Rede

De acordo com [Botta et al. 2005] pode-se dividir as ferramentas de análise de redes em: (1) ferramentas de estimativa da capacidade ponto-a-ponto, (2) ferramentas de estimativa da largura de banda e (3) ferramentas de medição da capacidade da largura de banda e transferência de dados sobre *Transmission Control Protocol* (TCP). Além de oferecerem diferentes características para uma análise da rede, essas ferramentas são *benchmarks* que usam grandes tranferências de dados sobre TCP e/ou *User Datagram Protocol* (UDP) para avaliar a taxa de transferência em um caminho ponto-a-ponto [Velásquez and Gamess 2009].

O Netperf é um *software* baseado no modelo cliente-servidor, seu foco é na transferência de uma massa de dados e na performance de solicitação/resposta utilizando os protocolos TCP ou UDP [Velásquez and Gamess 2009]. A saída de sua execução informa o tamanho do *socket* do *sender* e do *receiver* em bytes, o tamanho da mensagem enviada em bytes, a duração do teste e a largura de banda da rede [Velásquez and Gamess 2009].

O *Distributed Internet Traffic Generator* (D-ITG) é uma plataforma de código aberto que segue o modelo cliente-servidor capaz de gerar tráfego a nível de pacote [Velásquez and Gamess 2009]. O D-ITG pode executar a medição da latência de ida e volta de um pacote, a latência de ida de um pacote, avaliação da taxa de perda de pacotes, medição da taxa de transferência e *jitter* da rede.

3. A Ferramenta de *Benchmark*

Apesar de ser uma ferramenta que utiliza uma interface web, devido a necessidade de dados sobre o *hardware* das máquinas do *cluster*, esta ferramenta tem como servidor uma máquina constituinte do *cluster* no qual se deseja realizar o *benchmark*. Isso permitirá que a ferramenta possa configurar e distribuir os arquivos necessários entre as máquinas do *cluster* para sua execução, permitindo ao usuário acessar e executar a aplicação de qualquer lugar com acesso a rede do *cluster*.

3.1. Fluxo de execução da ferramenta

Além de diminuir o esforço do usuário na otimização das diversas aplicações disponíveis, a ferramenta proporciona ao usuário a visualização dos resultados dos indicadores de desempenho do *benchmark* de forma mais comunicativa e detalhada. Através da interface web da ferramenta, o usuário requisita a execução do *benchmark*, que após ser realizado apresenta os resultados em forma de tabelas e cujos gráficos podem ser manipulados de forma interativa.

No processo de execução do *benchmark*, a ferramenta realiza alguns passos claramente definidos para a correta execução da mesma. A figura 1 apresenta o fluxo das macro etapas realizadas pela ferramenta onde uma etapa é realizada logo após a outra. Na sequência cada etapa abordada na figura 1 será detalha.

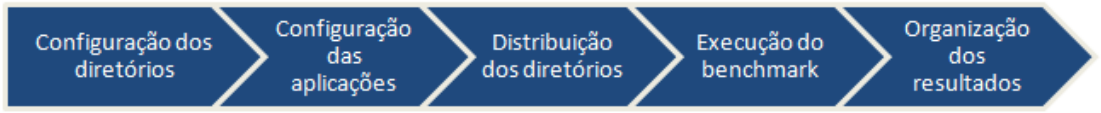

Figura 1. Passos realizados na execução do *benchmark*.

1. **Configuração dos diretórios:** Essa é a primeira etapa, onde a ferramenta cria no nó *master* os diretórios padrões que irão receber os arquivos necessários para a execução do *benchmark*. Este passo é importante pois além de configurar os caminhos que serão usados nos passos seguintes, cria a raíz de trabalho para o *benchmark* na pasta do usuário usado na comunicação via *Secure Shell (SSH)* do *cluster* para que os arquivos de saída não se espalhem no sistema conflitando com outros arquivos.
2. **Configuração das aplicações:** Após configurado os diretórios que serão a raíz de trabalho da ferramenta, é feito o *download* de arquivos essenciais utilizados. Como a ferramenta utiliza-se internamente de outras aplicações, detalhadas na seção 3.2, para a realização de seu *benchmark*, nesta etapa questões como configurações das versões a serem executadas, quantidade de execuções, cargas de trabalho, destino das saídas e número de máquinas participantes são definidas.
3. **Distribuição dos diretórios:** Com os diretórios completos e as aplicações prontas para serem executadas, os diretórios então são distribuídos para as máquinas constituíntes do *cluster* a partir dos arquivos de configurações gerados na etapa anterior.
4. **Execução do *benchmark*:** A ferramenta executa cada versão das aplicações configuradas nas etapas anteriores de acordo com a quantidade de amostras estabelecidas também nas etapas anteriores. Cada versão é executada uma após a outra e ao término de cada execução, os resultados são dispostos em arquivos para que posteriormente sejam formatados.
5. **Organização dos resultados:** Com os resultados em arquivos é possível formatá-los usando comandos de manipulação de arquivos nativos do sistema operacional linux, para que os dados possam ser usados na construção dos gráficos e tabelas visualizados pelo usuário. Ao final da execução também é criado um histórico para visualização de resultados prévios.

Por utilizar bastante de comandos e programas nativos do sistema operacional linux, para a correta execução e obtenção de dados coerentes e consistentes, é necessário que alguns cuidados sejam tomados.

- O usuário do cluster deve estar devidamente configurado com acesso remoto sem senha (SSH) para as outras máquinas.
- O arquivo /etc/hosts deve estar mapeado com os endereços de IP e nomes de todas as máquinas do *cluster* e que irão participar do *benchmark*.
- O OpenMPI deve estar instalado em todas as máquinas do *cluster*.
- Deve-se desabilitar a funcionalidade do processador conhecida como *CPU throttling*. Essa funcionalidade é uma técnica que automaticamente ajusta a frequência

de operação do processador para conservar energia ou reduzir o nível de calor gerado pelo chip. Isso porque a frequência máxima do processador é usada para os cálculos de desempenho teórico, ao usar os dados de frequência em escala reduzida, tanto o valor do desempenho teórico quanto os de eficiência não serão corretos.

3.2. Arquitetura da ferramenta

A ferramenta contempla internamente quatro aplicações que são essenciais para alcançar os resultados desejados. Essas aplicações têm suas saídas exploradas e relacionadas para prover indicadores de desempenho que permitam um melhor entendimento do comportamento da arquitetura analisada.

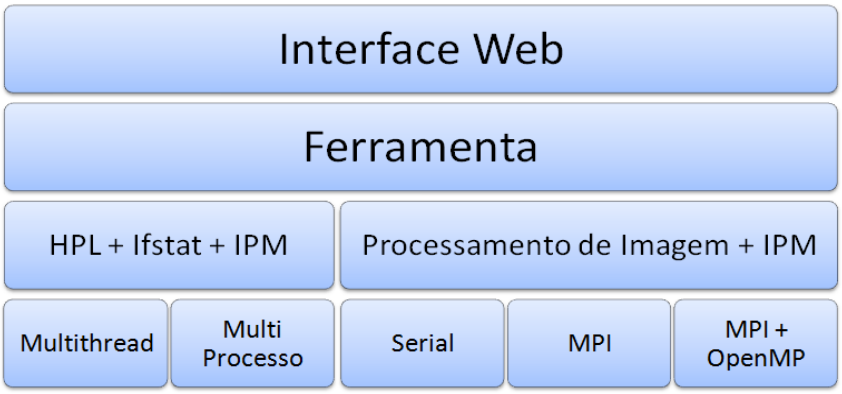

Figura 2. Aplicações constituintes do *benchmark*.

Observando a figura 2, percebe-se que uma das aplicações utilizadas é o *benchmark* HPL versão 2.0. A opção por utilizar o HPL deve-se ao fato de que ele é um *benchmark* consolidado e mundialmente usado para ranquear os supercomputadores mais potentes do mundo. Apesar de apresentar o desempenho em ponto flutuante, a precisão da solução e o tempo gasto no processamento, relacionar sua execução com o tráfego de rede no nó *master* e com as taxas de transferência, permite evidenciar fontes de ineficiência na arquitetura do *cluster*.

Por isso que durante a execução do *benchmark* HPL faz-se necessário a utilização da ferramenta Ifstat versão 1.1, que coleta informações do tráfego da rede para ajudar a compreender o comportamento de comunicação do algoritmo na arquitetura. Além disso, é necessário também a utilização conjunta do HPL com a ferramenta *Integrated Performance Monitoring* (IPM) versão 2.0, que é uma ferramenta de *proffiling* para aplicações paralelas que provê pouca sobrecarga sobre a execução da aplicação enquanto coleta dados.

Desta forma, para permitir uma melhor análise de desempenho com os indicadores apresentados pela aplicação, o *benchmark* HPL é compilado em duas versões (*Multithread* e Multi processo) conforme visto na figura 2. Isso permite ao usuário comparar e perceber a vantagem em utilizar a abordagem com *threads* ou processos em algoritmos que executem na sua arquitetura.

Já a aplicação de processamento de imagem, por ser utilizada em situações reais, necessitar de um grau de processamento considerável e possuir uma problemática

diferente da resolvida pelo *benchmark* HPL, proporciona saber o comportamento da arquitetura para problemas desta classe. O algoritmo em questão é o filtro de mediana, esse filtro possibilita a redução de ruídos na imagem substituindo o valor de cada *pixel* da imagem pelo valor da mediana da vizinhança ao redor do *pixel*. Ambas as versões executadas (MPI e híbrida) estão implementadas sobre o modelo *Master/Slave*. Na execução, apenas é utilizada a ferramenta IPM para ajudar a prover resultados mais detalhados.

Os indicadores apresentados pelo *benchmark* são relevantes para usuários que executem aplicações no modelo de programação *Master/Slave* ou utilizem problemas com estruturas bidimensionais em suas arquiteturas, pois as aplicações utilizadas neste *benchmark* utilizam destes modelos em seus algoritmos. Por isso, questões como comunicação no nó *master* e tempo de comunicação geralmente evidenciam problemas relacionados com gargalos entre o *master* e os outros trabalhadores.

4. Resultados Experimentais

A aplicação foi testada em um *cluster* composto de oito máquinas Hewlett-Packard (HP) ProLiant DL120 G6, com processador *Intel*®*Xeon*®*QuadCore* X3440 2.53 GHz com *HyperThreading*, 2 TB de armazenamento, placa de rede NetXtreme®BCM5723 *Gigabit Ethernet*, 8GB de memória RAM, sistema operacional Linux 3.2.0-32-generic X86-64 GNU/Linux Ubuntu 12.04LTS, conectadas através de uma rede *switched ethernet* com um *switch* GTS®modelo 73.1724S 10/100 Mbit/s utilizando cabos par trançado UTP categoria 5e. Na compilação foi utilizado o compilador mpicc do OpenMPI versão 1.7.2, além do OpenMP versão 3.1 contido no compilador GCC 4.8.1. O *cluster* encontra-se no Laboratório de Modelagem Computacional da Faculdade Senai Cimatec.

4.1. Análise dos resultados obtidos

HPL	Multi Processo	Multithread
Número de Testes	4	4
Desempenho Teórico	323,456 Gigaflops	323,456 Gigaflops
Desempenho Médio	39,3600 Gigaflops	63,7775 Gigaflops
Eficiência	12,00%	19,00%
Tempo Médio de Execução	9447,85 Segundos	5830,85 Segundos
Tempo Total de Execução	37791,4 Segundos	23323,4 Segundos
Tempo Total de Comunicação	30233,12 Segundos	16326,38 Segundos
Percentual de Comunicação	80,72%	70,95%
Max Taxa de Transferência	94,8073 Mbits	94,3891 Mbits
Latência Média da Rede	0,271 milisegundos	0,271 milisegundos

Tabela 1. Resultado médio da execução do *benchmark* HPL.

A tabela 1 apresenta a média dos resultados obtidos a partir dos testes com a execução do *benchmark* HPL pela ferramenta proposta neste trabalho. Diferentemente da saída padrão proporcionada pelo *benchmark* HPL, a ferramenta apresenta ao usuário não só o desempenho obtido no teste, mas relaciona o desempenho obtido com o desempenho teórico calculado para o cenário. Desta relação se obtém a eficiência, que representa o quanto a arquitetura consegue atingir daquele limite máximo estabelecido pelo desempenho teórico, quanto maior a eficiência melhor é o aproveitamento dos recursos.

Além do tempo de resolução do problema é apresentado o tempo total de execução, o tempo médio de execução e a partir do percentual de comunicação pode ser obtido o tempo de comunicação na execução do problema. A máxima taxa de transferência, a latência média da rede no momento da execução e o percentual da comunicação ajudam a identificar possíveis gargalos proveniente da rede de interconexão na resolução do problema.

A partir dos resultados obtidos com a ferramenta, percebe-se que tanto o desempenho obtido via multi processo quanto o obtido via multi*thread* não alcançaram nem 50% do desempenho teórico, porém o desempenho da versão multi*thread* com média de 63 gigaflops teve uma performance 1,6 vezes melhor que desempenho obtido com a versão multi processos que alcançou um desempenho médio de 39 gigaflops. Essa disparidade é refletida na eficiência que apresenta média de 12% para os testes com multi processos e uma média de 19% para os testes com multi*thread*.

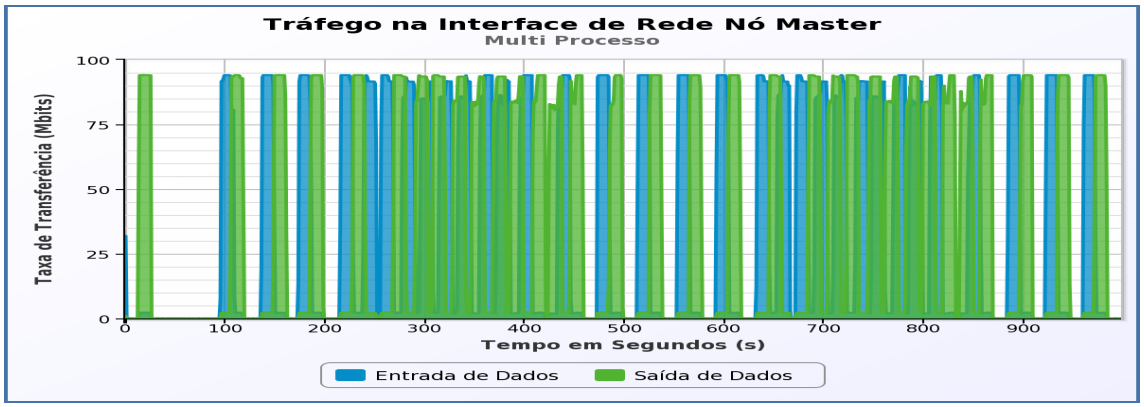

Figura 3. Tráfego na interface de rede do nó maste com o *benchmark* HPL Multi processos.

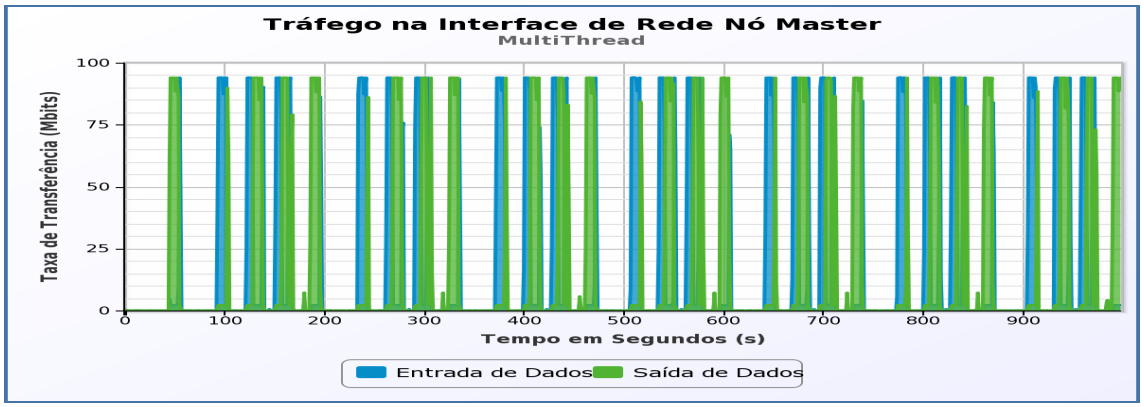

Figura 4. Tráfego na interface de rede do nó maste com o *benchmark* HPL *MultiThread*.

Ao observar os gráficos 3 e 4 produzidos pela ferramenta a partir da ação conjunta do HPL com o Ifstat, o *benchmark* encontra um limite no enlace de rede pertencente ao nó *master* saturando o mesmo em ambas abordagens. Ainda de acordo com a tabela 1, a taxa de transferência máxima obtida pelas abordagens multi processo e multi*thread*

no enlace são respectivamente 94.8073 Mbits e 94.3891 Mbits, por isso observando a figura 3 pode-se perceber que existe um fluxo constante de entrada e saída na interface de rede do nó *master* evidenciando que soluções que tenham muita comunicação são prejudicadas na presente arquitetura. Já na abordagem multi*thread*, representado na figura 4, a comunicação se concentra em picos de envio e recebimento com pouca sobreposição de entrada e saída de dados no enlace do nó *master*, em alguns momentos é possível observar até um ligeiro envio ou recebimento de dados com curta duração.

Na aplicação de processamento de imagem, a imagem de entrada possui tamanho de 30000x30000 *pixels* e para a versão híbrida do algorítmo, a ferramenta explora o nível máximo de *threads* que podem ser obtidos no sistema. Como no *cluster* utilizado cada máquina possui 1 *socket* e 1 processador *quadcore*, fica visível ao sistema a existência de 4 núcleos físicos no processador, então a aplicação configura o número de processos e *threads* até obter o número total de "processadores" visíveis ao sistema. Essa característica permite ao usuário visualizar algum tipo de vantagem em determinado número de processos/*threads* para a solução do problema na arquitetura.

Processos	MPI	Híbrido	
		Híbrido - 1 processo	Híbrido - 2 processos
4	5364,73 seg	5375,40 seg	5330,85 seg
8	2793,66 seg	2761,89 seg	2774,98 seg
12	1921,02 seg	1918,22 seg	1924,33 seg
16	1482,79 seg	1468,65 seg	1484,60 seg
20	1219,01 seg	1209,87 seg	1214,98 seg
24	1050,65 seg	1038,94 seg	1036,70 seg
28	929,05 seg	908,90 seg	911,79 seg
32	990,86 seg	804,20 seg	860,99 seg

Tabela 2. Tempo médio de processamento da aplicação de processamento de imagem.

De acordo a tabela 2 pode-se observar que o tempo de processamento da aplicação diminuiu a medida que processos vão sendo adicionados, além disso o que acontece é que a diminuição no tempo ocorre de forma similar para cada abordagem utilizada. Das três abordagens executadas pela ferramenta, a abordagem híbrida com 1 processo executando 4 *threads* por máquina é a que acaba apresentando no geral os menores tempos de execução com uma leve vantagem em relação a abordagem que utiliza apenas processos MPI por máquina e em relação a abordagem híbrida com 2 processos executando 2 *threads* por máquina.

Essa diminuição no tempo reflete no *speedup* obtido com a ferramenta utilizando o *testbed* que pode ser visto na figura 5. Utilizando até 7 máquinas trabalhadoras (28 processos/*threads*) o *speedup* alcançado pelas abordagens são bastante semelhantes, crescendo de uma maneira linear em relação a adição de máquinas no processo. Contudo, ao adicionar processos trabalhadores na máquina mestre (32 processos/*threads*) o ganho de desempenho obtido com a abordagem que utiliza apenas processos MPI sofre uma queda. Já a abordagem híbrida com 1 processo alcança o melhor ganho de desempenho ao executar o algoritmo, chegando a ser 25,685 vezes mais rápido que a abordagem serial.

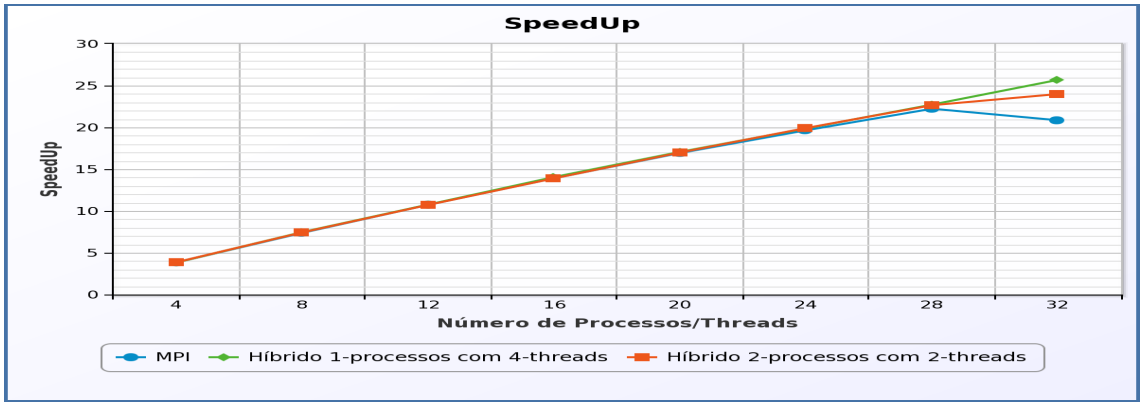

Figura 5. *Speedup* do *cluster* com a aplicação de processamento de imagem.

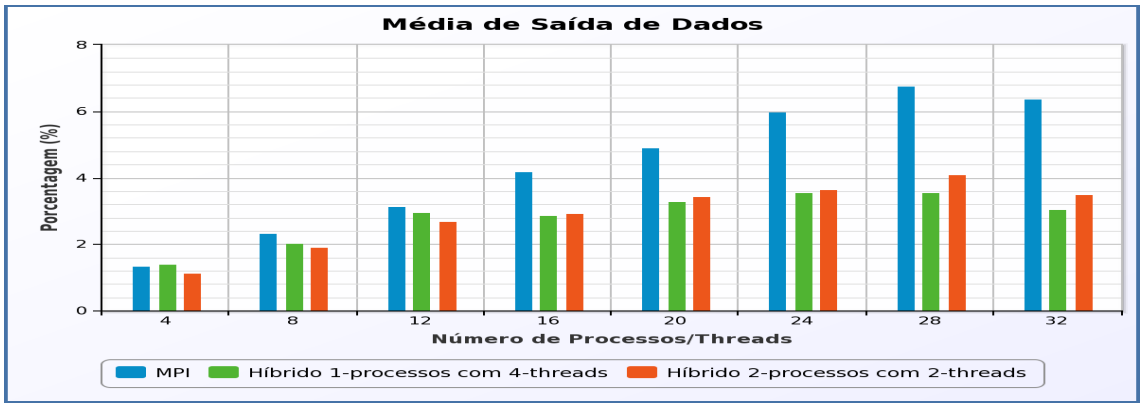

Figura 6. Porcentagem do tempo de envio de dados no *cluster* com a aplicação de processamento de imagem.

Observando a figura 6, apesar do algoritmo não possuir muita comunicação, pelo fato do problema ser totalmente dividido e enviado no início da computação, as abordagens utilizando mais de um processo MPI por máquina acabam tendo um tempo de envio de dados maior. Isso porque com a adição de mais processos na computação, o processo *master* passa mais tempo enviando dados pelo enlace de rede do que a abordagem híbrida que utiliza apenas um processo por máquina.

5. Conclusão

A ferramenta proposta neste artigo, através dos experimentos realizados na seção 4, mostrou ser possível a apresentação de indicadores de desempenho consistentes para uma análise e avaliação do desempenho de um *cluster* de computadores de memória distribuída. A partir da utilização da ferramenta, por meio da interface web, é possível disponibilizar aos usuários um *benchmark* automático, sem a necessidade de intervenções que requeram bastante conhecimento da arquitetura ou permissões avançadas, e um resultado bastante detalhado das aplicações utilizadas como carga no sistema.

No *cluster* utilizado pela aplicação, foi posível perceber que com a versão multi*thread* do *benchmark* HPL foi possível obter 63.77 gigaflops equivalentes a 19% do desempenho teórico que poderia ser atingido com a arquitetura, que se apresenta como sendo 1,6 vezes o desempenho e eficiência obtido com a versão utilizando multi proces-

sos. Esta que por utilizar muitos processos, acabou sofrendo com a limitação da largura de banda do enlace de rede do nó *master* da computação, onde a máxima taxa de transferência na versão foi de 94.8073 Mbits.

Executando a aplicação de processamento de imagem foi possível perceber que utilizando uma versão híbrida com 1 processo trabalhador que utilize todos os *cores* possíveis, obtêm-se desempenhos médios um pouco melhor que outras abordagens híbridas que façam um balanceamento entre processo e *thread* ou até mesmo que a abordagem utilizando apenas processos MPI. Isso porque além de evitar a utilização da rede para o envio de dados para muitos processos, a utilização do nó *master* na computação gera pouco impacto no ganho de desempenho.

Como trabalhos futuros desta pesquisa, uma questão importante a ser tratada é o fato de expandir o número de fatores que afetam o desempenho de sistemas paralelos de memória distribuída agregando indicadores que compreendam dispositivos como a memória. Além disso, pretende-se agregar algoritmos que implementem outros modelos de programação paralela como o divide e consquista e algoritmos que trabalhem com outros tipos de dados significativos para uma vasta gama usuários. Também pretende-se implementar um complemento na ferramenta para facilitar a associação de um outro código paralelo que o usuário deseje executar.

Referências

Botta, A., Pescape, A., and Ventre, G. (2005). On the performance of bandwidth estimation tools. In *Systems Communications, 2005 (ICW '05)*.

Cuenca, J., Garcia, L., Gimenez, D., and Dongarra, J. (2005). Processes distribution of homogeneous parallel linear algebra routines on heterogeneous clusters. *IEEE Computer Society*.

El-Rewini, H. and Abd-El-Barr, M. (2005). *Advanced Computer Architecture and Parallel Processing*. Wiley Series on Parallel and Distributed Computing. Wiley.

Meuer, H. (2013). Top500 supercomputers. Disponível em: <http://www.top500.org/project/>. Acesso em: 10 Set 2013.

Pilla, L. L. (2009). Análise de desempenho da arquitetura cuda utilizando os nas parallel benchmarks. Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, Porto Alegre, 2009. [Orientador: Prof Dr. Philippe Olivier Alexandre Navaux].

Rauber, T. and Rünger, G. (2010). *Parallel Programming: For Multicore and Cluster Systems*. Springer.

Velásquez, K. and Gamess, E. (2009). A comparative analysis of network benchmarking tools. *Proceedings of the World Congress on Engineering and Computer Science*, Vol I:299–305.

Zacarias, F. V., de Souza, J. R., Júnior, A. L. M., and de Oliveira, P. M. (2013). Avaliação de desempenho do cluster gabi. *ERBASE 2013, Itabaiana, Sergipe*, 1.