UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Implantação e Análise de Desempenho de um *Cluster* com Processadores ARM e Plataforma Raspberry Pi

Felipe dos Anjos Lima

SÃO CRISTÓVÃO/ SE
2016

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Felipe dos Anjos Lima

Implantação e Análise de Desempenho de um *Cluster* com Processadores ARM e Plataforma Raspberry Pi

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Edward David Moreno Ordonez

SÃO CRISTÓVÃO/ SE
2016

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

L732i

Lima, Felipe dos Anjos

Implantação e análise de desempenho de um *cluster* com processadores ARM e plataforma raspberry Pi / Felipe dos Anjos Lima ; orientador Edward David Moreno Ordoñez. - São Cristóvão, 2016.

66 f. : il.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Sergipe, 2016.

1. Computação de alto desempenho - Avaliação. 2. Sistemas embarcados (Computadores). 3. Raspberry Pi (Computador). 4. Processamento paralelo (Computadores). 5. Cluster (Sistema de computador). I. Ordoñez, Edward David Moreno, orient. II. Título.

CDU 004.051

Felipe dos Anjos Lima

Implantação e Análise de Desempenho de um *Cluster* com Processadores ARM e Plataforma Raspberry Pi

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Edward David Moreno Ordonez, Presidente
Universidade Federal de Sergipe (UFS)

Prof. Dr. Manoel Eusébio de Lima
Universidade Federal de Pernambuco (UFPE)

Prof. Dr. Ricardo José Paiva de Britto Salgueiro
Universidade Federal de Sergipe (UFS)

Prof. Dr. Wanderson Roger Azevedo Dias
Instituto Federal de Sergipe (IFS)

SÃO CRISTÓVÃO/ SE

2016

Implantação e Análise de Desempenho de um *Cluster* com Processadores ARM e Plataforma Raspberry Pi

Este exemplar corresponde à Dissertação de Mestrado do aluno Felipe dos Anjos Lima para ser avaliada pela Banca Examinadora.

São Cristóvão – SE, 26 de Agosto de 2016

_____

Prof. Edward David Moreno, Doutor
Orientador

_____

Prof. Manoel Eusébio de Lima, Doutor
Membro

_____

Prof. Ricardo José Paiva de Britto Salgueiro, Doutor
Membro

_____

Prof. Dr. Wanderson Roger Azevedo Dias, Doutor
Coorientador

Dedicatória

Dedico à minha família.

Agradecimentos

Aos meus pais, Edilson e Sandra pelo carinho, atenção, preocupação, esforço e dedicação para comigo, além dos conselhos e incentivos na tomada de minhas decisões, os quais sempre estiveram presentes o acompanhamento dos meus estudos.

À minha irmã, Marinalva e aos familiares pelo apoio e companhia em momentos difíceis, os quais contribuíram direta ou indiretamente na minha formação.

Ao Prof. Dr. Edward David pela excelente orientação, ele mesmo com muitos afazeres nunca deixou de auxiliar nas pesquisas e formação acadêmica, tal ajuda de grande importância no desenvolvimento deste trabalho e trabalhos anteriores.

A todos os professores do Departamento de Ciência da Computação não somente pela formação, mas também pela preocupação com meu desenvolvimento acadêmico e humano.

Resumo

Com o desenvolvimento da computação de alto desempenho (HPC), grandes volumes de dados passaram a ser processados de forma rápida, permitindo assim, que avanços significativos fossem alcançados em varias áreas do conhecimento. Para isso, sempre se observou a área de HPC tendo uma infraestrutura complexa. Por outro lado, nos últimos anos, se observa que a capacidade de processamento dos processadores usados em sistemas embarcados, seguindo arquitetura ARM, vem aumentando de forma significativa. Além disso, os custos de aquisição e o consumo de energia dos processadores ARM são menores, quando comparados a processadores de outras plataformas. Neste âmbito, cria-se a possibilidade de ter HPC usando plataformas menores e mais econômicas e com um custo de manutenção mais acessível. Nesse intuito, esta dissertação de mestrado, propõe a análise de desempenho de um *cluster* embarcado de baixo custo composto por processadores da arquitetura ARM e plataforma Raspberry Pi. O trabalho analisa o impacto de usar as bibliotecas MPICH-2 e OpenMPI, executando os programas dos *benchmarks* HPCC e HPL. O trabalho apresenta resultados de desempenho e consumo de energia do *cluster* com esses programas, mostrando que é possível usar *clusters* de plataformas embarcadas de baixo custo e tendo *speedups* e consumo de energia satisfatórios.

Abstract

With the recent advancements in High Performance Computing (HPC), it is possible to rapidly process high volumes of data, allowing accomplishments in several areas of knowledge. Although the HPC area has been observed as an area of complex infrastructure, in the last years, its been observed that the processing power of processors used in embedded systems, using the ARM architecture, has been increasing significantly. Furthermore, the acquisition costs and energy consumption are lower, when compared to processors of other platforms, thus allowing for the possibility of having HPC with smaller and more economical platforms, with lower maintenance cost and more accessible. In this merit, this masters thesis proposes the performance analysis of a low cost embedded cluster composed of processors using ARM architecture and the Raspberry Pi platform. This work analysis the impact of using MPICH-2 and OpenMPI libraries, running benchmark programs HPCC and HPL. The present work show results of performance and energy consumption of this cluster with these programs, proving that it is possible to use clusters of low cost embedded platforms with satisfactory speedups and energy consumption.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API – *Application Programming Interface*

ARM – *Advanced RISC Machine*

CISC – *Complex Instruction Set Computing*

GCC – *Gnu Compiler Collection*

GPU – *Graphics Processing Unit*

HDMI – *High-Definition Multimidia Interface*

HPCC - *High-Performance Computing Challenge*

HPC – *High-Performance Computing*

HPL – *High-Performance Linpack*

LAN – *Local Area Network*

MPI – *Message Passing Interface*

NUMA – *Nouniform Memory Access*

RISC – *Reduced Instruction Set Computing*

SDRAM – *Synchronous Dynamic Random Access Memory*

UMA – *Uniform Memory Access*

USB – *Universal Serial Bus*

Sumário

Capítulo 1

Introdução

Com o desenvolvimento tecnológico, o poder de processamento dos computadores aumentou muito nos últimos anos. No entanto, a quantidade de informações processadas também aumentou. Por isso, a busca por processadores e sistemas cada vez mais velozes e que atendam as novas demandas continua sendo prioridade. Segundo Rauber e Rünger (2013), várias áreas das ciências naturais e da engenharia estão cada vez mais necessitando de poder computacional, pois são áreas que necessitam realizar simulações de problemas científicos que manipulam grandes quantidades de dados, demandando assim grande esforço computacional, visto que um baixo desempenho dos sistemas pode levar a uma restrição das simulações e uma imprecisão nos resultados obtidos.

Atualmente, a tarefa de manter o poder de computação dos processadores aumentando continuamente, seguindo a Lei de Moore (MOORE, 1987), tem sido um grande desafio para os engenheiros e cientistas da computação. Os principais motivos são as dificuldades presentes no desenvolvimento de tecnologias que permitam reduzir o tamanho dos transistores, bem como a quantidade de calor dissipada por esses componentes (BLUME, 2013).

Para resolver esses problemas, os *designers* de processadores passaram a projetar circuitos que pudessem processar dados em paralelo, ou seja, ao invés de tentarem aumentar o poder de processamento de um único núcleo do processador, incluindo mais transistores, os *designers* passaram a considerar a criação de processadores com muitos núcleos em um único circuito integrado. Tais circuitos são conhecidos como processadores *multicore*. Ao tirar proveito do paralelismo com processadores compostos por vários núcleos, foi possível elevar o poder de processamento a taxas mais altas do que as que vinham sendo alcançadas quando apenas um núcleo era utilizado.

No entanto, essa não é a única forma possível de explorar paralelismo, pois é possível construir sistemas que realizam processamento paralelo por meio da associação

de dois ou mais computadores, conhecidos como nós, conectados a uma rede local LAN (*Local Area Network*), tais sistemas são conhecidos como *clusters*. Todos os computadores em um *cluster* são vistos como um sistema único e coerente que realiza o gerenciamento de memória necessário para manter a consistência da computação realizada.

Para tirar proveito dos recursos disponibilizados pelos sistemas que suportam o processamento paralelo, existem atualmente várias APIs (*Application Programming Interfaces*) e bibliotecas que auxiliam o desenvolvimento de aplicações paralelas. Como exemplos, podemos citar as bibliotecas MPI (Message Passing Interface) e OpenMP (Open Multi-Processing). A biblioteca MPI possui um conjunto de funções que permitem que sistemas paralelos com memória distribuída troquem mensagens entre si e assim possam compartilhar dados e instruções de controle. Já a biblioteca OpenMP é utilizada para prover o processamento paralelo entre os núcleos de um processador com memória compartilhada, ou seja, várias *threads* são alocadas e executadas simultaneamente pelos núcleos do processador. Nesta dissertação, utilizamos diferentes implementações da biblioteca MPI, assim foi possível medir e analisar o desempenho do *cluster* composto por processadores *quadcore* da plataforma ARM (*Advanced RISC Machine*).

Nos últimos anos, os processadores da plataforma ARM começaram a dominar o mercado dos sistemas embarcados. Apesar de termos a arquitetura x86 presente em boa parte dos computadores pessoais e dos *notebooks*, por serem mais complexos e possuírem estágios de processamento mais longos, os processadores x86 não são uma solução viável para sistemas embarcados. Já os processadores ARM, por serem baseados na arquitetura RISC (*Reduced Instruction Set Computing*), possuem um conjunto de instruções mais simplificado, o que torna o processador menos complexo.

O desenvolvimento das micro-arquiteturas de hardware também permitiu que sistemas computacionais embarcados fossem criados. No entanto, nem todos os tipos de processadores possuem um bom desempenho em tais sistemas. E isso se deve entre outros fatores ao alto consumo de energia, impedindo que o sistema embarcado possa operar de forma independente por mais tempo. Por isso, criar sistemas que operam com pouca quantidade de energia é fundamental para a viabilização de soluções computacionais embarcadas. Além disso, sistemas que consomem muita energia podem não ser implantados, pois os altos custos envolvidos na manutenção do sistema podem inviabilizar a sua utilização (ZOMAYA, 2012).

1.1 Problemática e Hipótese

Durante o desenvolvimento de sistemas paralelos, várias ferramentas, linguagens e ambientes de programação podem ser utilizados. Por isso, é importante que as vantagens e desvantagens de tais recursos sejam conhecidas. Assim, os projetistas poderão escolher os recursos que melhor se adaptem ao sistema que está sendo construído.

Segundo Bez et al. (2015), soluções baseadas na arquitetura de processadores ARM estão cada vez mais sendo utilizadas, isso se deve ao baixo custo e ao baixo consumo de energia dos processadores da plataforma ARM.

Reed e Dongarra (2015) chamam a atenção para o fato de que sistemas de alto desempenho podem não ser implantados devido a restrições impostas pelo alto consumo de energia, logo, além de medir e analisar o poder computacional dos sistemas paralelos é importante que o consumo energético de tais sistemas também seja medido.

Assim, faz-se necessário analisar o desempenho de *clusters* de baixo custo com processadores da plataforma ARM, levando-se em consideração diferentes implementações da biblioteca MPI e diferentes *benchmarks* paralelos. Dessa forma, será possível verificar qual implementação obteve um melhor desempenho e um baixo consumo de energia.

1.2 Justificativa

Atualmente, o desenvolvimento de sistemas computacionais de alto desempenho tem permitido o processamento de grandes quantidades de dados. No entanto, o poder computacional disponível ainda não é suficiente, pois boa parte dos dados gerados em diversas áreas do conhecimento continua sendo desprezada, devido à indisponibilidade de recursos computacionais capazes de atender a essa demanda.

Diante do grande potencial da computação paralela e do crescimento da capacidade de processamento dos sistemas embarcados, a realização de testes e análises que possam medir o desempenho de *clusters* compostos por processadores embarcados e de baixo custo permite que o desempenho das aplicações executadas por tais sistemas seja otimizado, pois os resultados dos testes serão levados em consideração durante a escolha das aplicações, do ambiente de programação e da configuração do *cluster*.

1.3 Objetivos

O objetivo do presente trabalho é implantar e realizar a análise de desempenho de um *cluster* embarcado com processadores ARM e plataforma Raspberry Pi, a fim de verificar o comportamento do mesmo durante a execução de *benchmarks* paralelos em diferentes ambientes de programação.

1.3.1 Objetivos Específicos

Visando atingir o objetivo geral descrito anteriormente, este trabalho tem os seguintes objetivos específicos:

1. Montar e configurar o *cluster* embarcado de baixo custo;
2. Verificar o impacto que a variação do número de processadores tem sobre o desempenho do *cluster*;
3. Avaliar o desempenho do *cluster* de baixo custo durante a execução dos benchmarks HPL e HPCC;
4. Avaliar o desempenho do *cluster* durante a execução de diferentes implementações da biblioteca MPI.
5. Medir o consumo de energia do *cluster* utilizando um circuito real;

1.4 Organização

Além desta introdução, o trabalho está dividido em cinco capítulos: O Capítulo 2 apresenta a metodologia de desenvolvimento do trabalho, além dos materiais utilizados durante os testes que avaliaram o desempenho do *cluster* embarcado em diferentes cenários; O Capítulo 3 aborda os principais conceitos envolvidos no desenvolvimento de sistemas paralelos e de alto desempenho; O Capítulo 4 apresenta o *cluster* proposto e utilizado durante os testes de desempenho; O Capítulo 5 apresenta os resultados da análise de desempenho do *cluster* com processadores *ARM*, levando em consideração o *benchmark* HPL e as bibliotecas OpenMPI e MPICH-2; O Capítulo 6 apresenta os resultados da análise de desempenho do *cluster* com o *benchmark* HPCC e bibliotecas OpenMPI e MPICH-2; O Capítulo 7 apresenta as conclusões e os trabalhos futuros.

Capítulo 2

Materiais e Metodologia

Este capítulo apresenta uma descrição dos materiais utilizados durante a dissertação, bem como os métodos que foram necessários para a extração e análise estatística dos dados. Os experimentos levaram em consideração o desempenho do *cluster* embarcado em diferentes ambientes de programação. Para isso, foram utilizadas as implementações OpenMPI e MPICH-2 da biblioteca MPI. Com o auxilio dos *benchmarks* HPCC e HPL, foi possível analisar a capacidade de processamento do *cluster*, bem como aspectos referentes à comunicação entre os processadores pela rede. Além disso, também foram realizados testes que mediram o consumo de energia do *cluster* durante a execução dos *benchmarks* em diferentes ambientes de programação.

2.1 Método de Pesquisa

Segundo Wazlawick (2009), uma pesquisa científica tem caráter experimental quando o pesquisador sistematicamente provoca alterações no ambiente pesquisado com o objetivo de verificar se cada intervenção produz o resultado esperado.

Assim, podemos classificar a natureza da pesquisa desenvolvida nesta dissertação como experimental, pois, para que fosse possível verificar a validade ou não das hipóteses de pesquisa, foram realizados testes que levaram em consideração o comportamento do *cluster* embarcado em diferentes cenários, e em seguida, foi realizada a análise estatística dos dados.

2.2 Trabalhos Relacionados

Esta seção apresenta os trabalhos relacionados à dissertação.

2.2.1 Cluster com Processadores ARM e Plataforma Raspberry Pi

Lima et al. (2016) apresenta a análise de desempenho de um *cluster* com processadores ARM e plataforma Raspberry Pi, durante os experimentos, foram realizados testes mediram o desempenho do *cluster* durante a execução do *benchmark* HPL e dos algoritmos de multiplicação de matrizes e cálculo do produto escalar entre dois vetores. Os autores concluíram que o aumento do número de processadores aumentou o desempenho do *cluster* em todos os cenários de testes.

2.2.2 Computação Exascale e Big Data

Reed e Dongarra (2015) mostram que os avanços científicos estão permitindo a criação de sisteams computacionais cada vez mais poderosos. No entanto, existem alguns problemas que impedem o pleno desenvolvimento da computação em *exascale*. Um deles é o alto consumo de energia dos sistemas de alto desempenho que acaba limitando a capacidade de operação dos mesmos.

2.2.3 Consumo de Energia de Aplicações HPC com ARM

Padoin et al. (2014) realizou análises de desempenho e consumo de energia de sistemas embarcados com processadores ARM, a fim de verificar a aplicabilidade de tais sistemas com aplicações que demandam grande poder de processamento, tendo em vista o baixo custo de aquisição e o baixo consumo de energia.

2.2.4 Aplicações HPC com *Cluster* de Baixo Custo

Bez et al. (2015) realizou testes que mediram a eficiência energética de aplicações de geofísica. Foram realizados testes com a aplicação Ondes3D e diferentes *flags* de compilação. Os autores perceberam que o uso apropriado de flags reduziu o

consumo de energia do cluster com processadores. Assim, é possível otimizar sistemas com processadores ARM e obter um desempenho satisfatório.

2.2.5 Desempenho de Aplicações Paralelas

H. Blume *et al*. (2008) verificaram o impacto que a implementação de programas paralelos com o uso da biblioteca OpenMP provocou no desempenho de aplicações. Para a realização dos testes foram utilizadas as aplicações: JPEG, Huffman Codec, Radixsort, AES Encryption, traveling Salesman, Block Matching, Discrete Fourier Transform. As simulações foram realizadas na plataforma ARM11 MPCore. Os resultados das simulações mostraram que um aumento no número de threads implicou em um aumento considerável no poder de processamento e na eficiência. Além disso, o aumento do número de threads provocou um aumento no consumo de energia.

2.2.6 Consumo de Energia de Dispositivos Eletrônicos

Furlinger *et al*. (2012) apresenta uma análise do consumo de energia de um *cluster* montado a partir de cinco dispositivos Apple TV. Cada dispositivo possui um processador ARM Cortex-A8. Para medir o desempenho dos nós individuais do cluster, os autores utilizaram o *benchmark* Coremark. Com o Coremark foi possível realizar testes de desempenho de operações que manipulam números inteiros. Para medir o desempenho de operações que manipulam números de ponto flutuante em cada nó do cluster, foi utilizado o benchmark LINPACK. O melhor desempenho alcançado foi de 160,4 MFlops em operações aritméticas.

2.2.7 Desempenho de Bibliotecas que Manipulam Threads

Silva e Yokoyama (2011) tem o objetivo de comparar o desempenho de bibliotecas que manipulam threads. Para a realização dos testes foram utilizadas as bibliotecas Pth (GNU Portable Threads), Protothreads e PM2 Marcel. Durante os testes, os autores verificaram o tempo gasto para inicialização, criação e operação de join das threads. A análise dos resultados permite concluir que a biblioteca Protothreads obteve o melhor desempenho, isso se deve a forma como a biblioteca é implementada.

2.3 Benchmarks Paralelos

Para medir o desempenho do *cluster*, foram utilizados *benchmarks* que permitiram analisar vários aspectos do *cluster* proposto. Em Ciência da Computação, um *benchmark* é um conjunto de programas que realizam grande quantidade de operações matemáticas, a fim de extrair métricas de desempenho do sistema. Todos os testes podem ser customizados, por meio da passagem de parâmetros, no momento da execução do *benchmark*.

2.3.1 *Benchmark* HPL

O *benchmark* HPL (*High-Performance Linpack*) é uma versão aprimorada do *benchmark* Linpack, que permite medir o desempenho de sistemas paralelos e *clusters*. Para isso, o *benchmark* realiza operações que demandam grande esforço computacional. Durante sua execução, o HPL resolve um sistema de equações lineares denso do tipo A.x = b, onde A é uma matriz densa gerada aleatoriamente de dimensão N x N, e x e b são vetores de tamanho N. O valor da variável N pode ser alterado quando necessário, a fim de permitir uma maior precisão dos resultados.

O HPL também permite a definição de vários cenários diferentes. Por meio de um arquivo de configuração, é possível modificar os parâmetros que serão levados em consideração pelo *benchmark* durante a sua execução. A Tabela x apresenta os principais parâmetros do arquivo de configuração do HPL.

Tabela 2.1: Parâmetros do *benchmark* HPL

Parâmetros	**Significado**
N	Tamanho do problema
NB	Tamanho do bloco
P	Número de linhas de processos
Q	Número de colunas de processos

2.3.2 Benchmark HPCC

O *benchmark* HPCC é um conjunto formado por outros sete *benchmarks*: HPL, STREAM, RandomAccess, PTRANS, FFTE, DGEMM and b_eff Latency/Bandwidth. A descrição de cada *benchmark* é apresentada a seguir:

- **HPL**: realiza o cálculo de um sistema linear denso e como saída apresenta a capacidade de processamento em Gflops/s e o tempo gasto para resolver o problema. A seção anterior apresentou os detalhes do HPL.
- **STREAM**: mede a largura de banda da memória disponível.
- **RandomAccess**: mede a taxa de atualizações aleatórias da memória.
- **PTRANS**: mede a taxa de transferência de grandes *arrays* de dados em sistemas de memória multiprocessados.
- **FFTE**: mede a taxa de execução de pontos flutuantes durante a realização da transformada de Fourier.
- **DGEMM**: mede a taxa de execução de pontos flutuantes durante a multiplicação de matrizes de números reais.
- **b_eff Bandwidth**: mede a largura de banda envolvida na comunicação entre os nós do sistema paralelo.

2.4 Análise Estatística dos Dados

Após a realização dos testes com os *benckmarks* propostos, foi feita a análise estatística dos dados obtidos. Para isso, utilizamos a distribuição *t* de *Student*, que pode ser aplicada quando o número de elementos em uma amostra é menor ou igual a 30 (FREUND, 2206). Assim, com a distribuição *t* de *Student* foi possível realizar a análise probabilística que permitiu calcular o intervalo de confiança para a média dos valores presentes nas amostras coletadas durante os testes. Devido ao tempo gasto durante a realização dos testes, optamos por realizar trinta execuções por teste, alterando os *benchmarks* e as implementações da biblioteca MPI.

O cálculo do intervalo de confiança para a média de amostras com N elementos, onde N é menor ou igual a trinta, compreende os seguintes passos (LARSON, 2010):

1. Cálculo da média da amostra.
2. Cálculo do desvio padrão da amostra.
3. Definição do intervalo de confiança desejado.
4. Definição do grau de liberdade. Geralmente, é igual a N-1.
5. Cálculo da margem de erro no intervalo de confiança.
6. Cálculo dos extremos esquerdo e direito do intervalo de confiança.

2.5 Implementações da Biblioteca MPI

Com o objetivo de testar as hipóteses levantadas no capítulo 1 deste trabalho, utilizamos diferentes implementações da biblioteca MPI, a fim de verificar qual seria o comportamento do *cluster* em diferentes ambientes de programação.

MPI (*Message Passing Interface*) é uma biblioteca de "*Message-Passing*", desenvolvida para ser padrão, inicialmente, em ambientes de memória distribuída, em "*Message-Passing*" e em computação paralela (PACHECO, 2012). Todo paralelismo é explícito, ou seja, o programador é responsável por identificar o paralelismo e implementar um algoritmo utilizando construções com o MPI (ALEXANDER, 2010).

Neste modelo de programação paralela, um mesmo código fonte é executado em cada processador. Assim, as variáveis declaradas são relativas a cada processador. O MPI é o responsável por realizar o gerenciamento de memória entre os núcleos do sistema. A Figura 6 apresenta o modelo de memória usado em *clusters*.

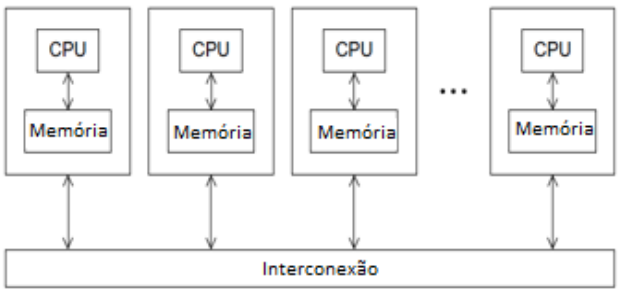

Figura 2.1: Sistema de Memória Distribuída

Os processos que estão sendo executados pelos nós do *cluster* se comunicam através do envio de mensagens. Para isso, o MPI disponibiliza um conjunto de funções que estão descritas a seguir:

- **MPI_Init**: responsável pelas configurações iniciais do sistema. É a primeira função a ser executada pelo sistema.
- **MPI_Finalize**: libera todos os recursos utilizados pelo MPI. É a última função a ser executada pelo sistema.
- **MPI_Send**: carrega as informações que serão trocadas entre os processos.
- **MPI_Recv**: é usado para receber as mensagens que foram enviadas por outros processos.

Todas as funções apresentadas acima podem ser chamadas a partir de programas escritos nas linguagens C, C++, Python e Fortran.

2.6 Circuito Medidor de Energia

Para que fosse possível medir o consumo de energia do *cluster* embarcado, foram utilizados dois circuitos responsáveis pela medição da corrente e da tensão durante a execução dos *benchmarks* HPL e HPCC. A Figura 2.2 apresenta os circuitos usados nas medições.

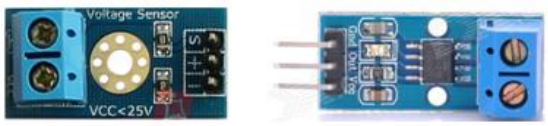

Figura 2.2: Circuitos medidores de tensão e corrente

2.7 Métricas de Desempenho

Durante a análise de desempenho do *cluster* com processadores ARM, foram propostos vários cenários, com o objetivo de verificar o comportamento do *cluster* ao realizar a execução dos *benchmarks* HPL e HPCC. Após os testes, foram realizados cálculos estatísticos que mediram a capacidade de processamento do *cluster* em Gflops/s, o tempo gasto para executar as aplicações e o consumo de energia do sistema.

Capítulo 3

Fundamentação Teórica

Atualmente, grande parte dos sistemas computacionais modernos já se utiliza de recursos que tornam esses sistemas mais poderosos e capazes de atender grandes demandas de processamento de dados. A capacidade de processar dados em paralelo é um deles, permitindo assim, que um sistema computacional execute várias tarefas ao mesmo tempo. No entanto, é importante lembrar que para executar várias tarefas ao mesmo tempo, o sistema precisa ter a sua disposição vários processadores ou mais de um núcleo de processamento em um único chip. Este capítulo apresenta os principais conceitos envolvidos no projeto de sistemas computacionais paralelos.

3.1 Computação de Alto Desempenho

Com o desenvolvimento científico e tecnológico, foi possível aumentar a capacidade de processamento dos computadores. No entanto, a utilização de sistemas computacionais com apenas um processador passou a não suprir a demanda por processamento necessária em diversas áreas do conhecimento. É nesse contexto que surge a Computação de Alto Desempenho ou HPC (*High-performance computing*). Assim, foi possível criar supercomputadores e *clusters* capazes de processar grandes volumes de dados em um curto período de tempo. Como exemplos de aplicações que demandam grande poder de processamento, podemos citar sistemas que realizam análises meteorológicas, sistemas que realizam análises de grandes cadeias de nucleotídeos presentes no DNA humano, e sistemas que buscam dados na internet (PACHECO, 2012). Atualmente, a lista TOP500 apresenta os 500 supercomputadores mais poderosos do mundo. O critério de avaliação utilizado para a inclusão de um supercomputador na lista é o desempenho obtido durante a execução do *benchmark* HPL.

3.2 Tipos de Sistemas Paralelos

Sistemas computacionais podem ser caracterizados de acordo com vários critérios. Uma das classificações mais conhecidas é a proposta por Flynn em 1972 Segundo Flynn, levando-se em consideração o número de fluxos de dados e de fluxos de controle que um sistema pode gerenciar simultaneamente, um sistema computacional pode ser classificado segundo uma das categorias a seguir:

- **Sistemas SISD (Single Instruction, Single Data)**: são sistemas que possuem apenas um processador, e por isso, possuem apenas um fluxo de dados e um fluxo de execução de instruções.
- **Sistemas SIMD (Single Instruction, Multiple Data)**: são sistemas que possuem um conjunto de unidades funcionais que operam simultaneamente e são gerenciadas por um único controlador. Assim, uma mesma instrução pode ser executada sobre dados diferentes.
- **Sistemas MIMD (Multiple Instruction, Multiple Data)**: são sistemas que possuem múltiplos fluxos de instruções e múltiplos fluxos de controle. Fazem parte dessa categoria os processadores *multicore*, os computadores com múltiplos processadores e os *clusters*.

3.3 Sistemas Concorrentes

Segundo Silberschatz (2013), todo programa concorrente possui múltiplas threads de controle (ver Figura 3.1). No entanto, isso não garante que elas sejam executadas em paralelo. Caso o sistema possua apenas um núcleo de processamento disponível, e várias tarefas para executar, as múltiplas threads de controle alternam a utilização do processador durante um intervalo de tempo específico.

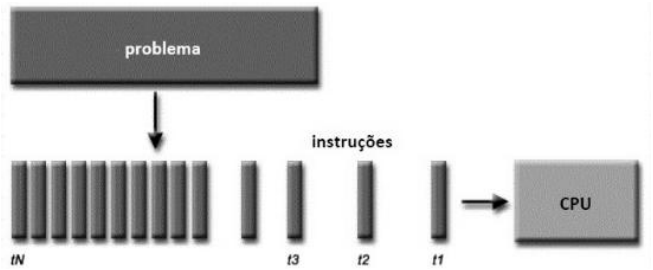

Figura 3.1: Sistema com múltiplos threads

Apesar de não haver de fato o processamento de dados em paralelo, sistemas com um único núcleo de processamento podem aumentar o seu desempenho com o uso de *threads* (GEBALI, 2011).

Como exemplo, podemos citar os sistemas operacionais multiprogramados, onde o objetivo é maximizar o uso do processador. Assim, sempre que um *thread* entra em modo de espera, o sistema operacional passa a executar outro *thread*, impedindo que o processador fique ocioso (HAOQIANG, 2011). Esse mecanismo é conhecido como escalonamento de threads.

3.4 Sistemas Multicore

Segundo Pacheco (2012), à medida que o tamanho dos transistores diminui, o poder de processamento do circuito integrado aumenta. No entanto, o calor dissipado por esses circuitos também aumenta, sendo um problema para o desempenho do sistema. A solução encontrada para contornar esse problema foi colocar em um único circuito integrado várias unidades de processamento. Tais circuitos ficaram conhecidos como circuitos *multicore* (GEBALI, 2011).

Apesar de o termo *multicore* ser comumente usado para designar sistemas que possuem vários núcleos de processamento em um único chip, ele também é usado para designar sistemas onde os processadores estão em chips diferentes (HAOQIANG, 2011). A Figura 3.2 apresenta o esquema de um processador com duas unidades de processamento.

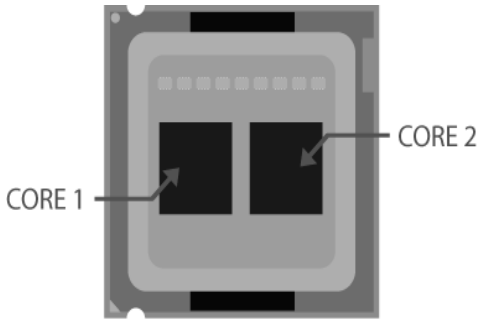

Figura 3.2: Processador Multicore

Para que possam operar corretamente, sistemas *multicore* necessitam de um gerenciamento de memória eficiente, para que seja mantida a consistência dos dados processados pelos vários núcleos.

Em sistemas de memória compartilhada, com múltiplos processadores, a conexão dos processadores com a memória pode ser feita de duas formas (PACHECO, 2012): UMA (*Uniform Memory Access*) e NUMA (*Nouniform Memory Access*).

No modelo de conexão UMA, os processadores são ligados diretamente à memória principal. Dessa forma, o tempo de acesso a todos os endereços da memória é o mesmo para todos os processadores (PACHECO, 2012). A Figura 3.3 apresenta o esquema de funcionamento do modelo UMA.

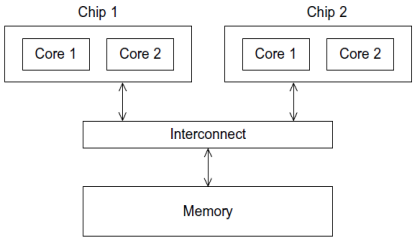

Figura 3.3: Modelo de conexão UMA

Já no modelo NUMA, cada processador tem acesso a um bloco de memória principal própria (GEBALI, 2011). Caso um processador deseje acessar um bloco de memória de outro processador, será necessário utilizar um hardware adicional, pois o acesso não poderá ser feito diretamente. A Figura 3.4 apresenta o esquema de funcionamento do modelo NUMA.

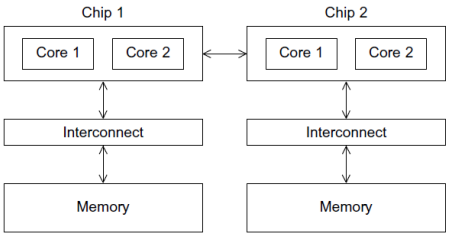

Figura 3.4: Modelo de conexão NUMA

3.5 *Clusters*

Um *cluster* é uma coleção de dois ou mais computadores usados para executar uma dada tarefa (HAOQIANG, 2011). Normalmente, os nós do cluster são conectados por meio de uma rede local que deve permitir a comunicação eficiente entre os mesmos. A Figura 3.5 apresenta o esquema de funcionamento de um *cluster*.

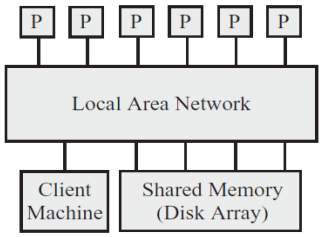

Figura 3.5. *Cluster* com múltiplos nós

Para manter a consistência durante a execução das tarefas, o cluster utiliza o modelo de memória compartilhada NUMA, pois cada processador possui acesso a uma memória própria (PACHECO, 2012). Alguns autores chamam de sistemas de memória distribuída, os sistemas que usam o modelo de compartilhamento de memória NUMA.

3.6 Algoritmos Paralelos

Para que seja possível tirar proveito dos recursos disponibilizados pelas arquiteturas paralelas, é necessário que os algoritmos estejam preparados para operar neste tipo de arquitetura, a fim de melhorar a sua velocidade de processamento.

Atualmente, existem muitos algoritmos que não reconhecem os recursos disponibilizados por arquiteturas de hardware paralelas. Sistemas que realizam processamento de imagens, análises de dados sobre o clima, computação científica, pesquisas energéticas e que manipulam grandes quantidades de dados, podem ter um aumento considerável no desempenho caso tirem proveito dos recursos providos por arquiteturas de hardware paralelas.

Para que um algoritmo paralelo possa ser executado corretamente pelo sistema paralelo, ele deve realizar três tarefas básicas:

- **Mapeamento**: é a distribuição das tarefas para os diferentes processadores.
- **Compartilhamento**: compartilha a execução das tarefas conforme a dependência de dados.
- **Identificação**: identifica os dados que trafegam entre os processadores.

3.7 Níveis de Paralelismo de um Sistema

Existem vários tipos de paralelismo envolvido em um sistema computacional. Para que o algoritmo possa executar de forma paralela, é necessário que o hardware dê o suporte necessário. A seguir apresentamos os principais tipos de paralelismo (PACHECO, 2012).

- **Paralelismo em nível de bit**: os bits de uma instrução são processados em paralelo.
- **Paralelismo em nível de instrução**: através de um recurso conhecido como *pipeline*, os processadores podem processar instruções em paralelo, pois possuem múltiplas unidades funcionais.
- **Paralelismo em nível de dados**: os dados armazenados podem ser acessados em paralelo e em seguida terem seus resultados combinados.
- **Paralelismo em nível de tarefas**: usa um fluxo separado de controle para cada tarefa que será executada de forma independente.

Capítulo 4

Cluster Embarcado com Processadores ARM

Neste capítulo, apresentamos o *cluster* embarcado proposto neste trabalho, enfatizando as características da placa Raspberry Pi, do sistema operacional Raspbian e do processador ARM.

4.1 Plataforma Rasberry Pi

O *cluster* proposto utiliza placas do tipo Raspberry Pi 2 (ver Figura 4.1) com as seguintes configurações:

- Memória SDRAM com 1 GB.
- Processador ARM com 900 MHz.
- 4 entradas USB 2.0.
- GPU Dual Core Videocore IV.
- Interface HDMI para saída de vídeo.
- Interface de rede 10/100 Ethernet RJ45.

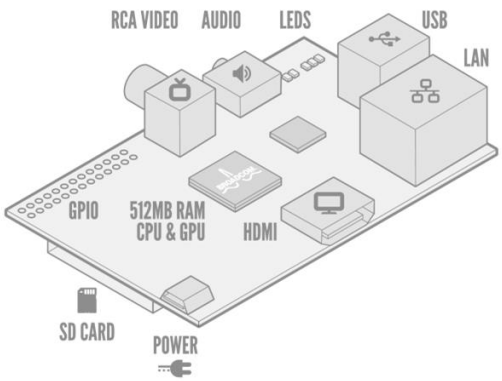

Figura 4.1: Componentes da placa Raspberry Pi

4.2 Sistema Operacional Raspbian

Para controlar cada placa do cluster, será utilizado o sistema operacional Raspbian que é otimizado para executar em placas do tipo Raspbery Pi e foi derivado a partir do sistema operacional Debian.

4.3 Arquitetura ARM

Em outubro de 1983, a empresa Acorn iniciou o projeto dos processadores ARM. O desenvolvimento do conjunto de instruções ficou sob a responsabilidade de Sophie Wilson, já Steve Furber ficou responsável pela modelagem do *chip*. Após o desenvolvimento de algumas versões do processador, a empresa Apple demonstrou interesse nos processadores ARM, no entanto havia a necessidade de se realizarem algumas alterações no projeto. Assim, em novembro de 1990 surgiu a empresa ARM, fundada com recursos da Acorn, Apple e VLSI.

O principal objetivo da ARM não era criar seus próprios chips, mas uma arquitetura que pudesse ser adquirida por empresas que constroem processadores. Por isso, ao invés de vender chips, a ARM licencia sua propriedade intelectual para outras empresas.

Figura 4.2: Processador ARM

4.4 Características do *Cluster* Embarcado

Para a montagem do cluster, será utilizado quatro placas Raspberry Pi do tipo B. As placas se comunicam através dos módulos de rede que cada uma possui. O

processador das placas pertence à plataforma ARM. A Figura 4.3 apresenta uma placa Raspberry Pi do modelo B+.

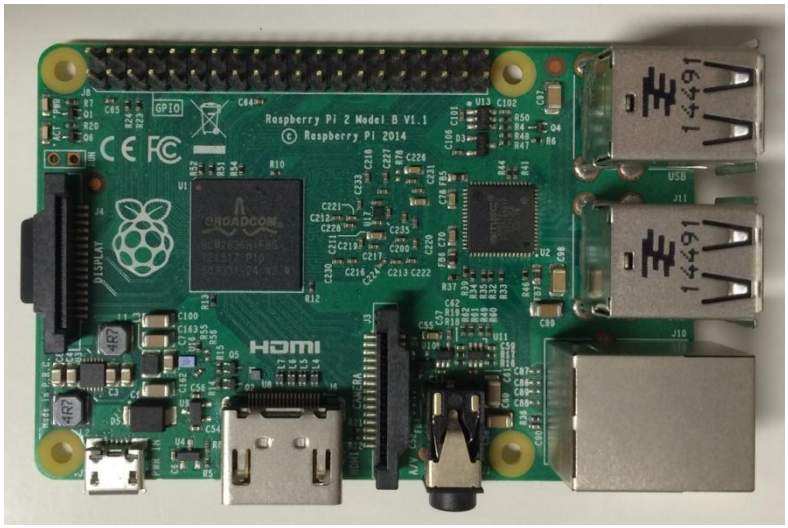

Figura 4.3. Placa Raspberry Pi

Uma das vantagens da utilização de placas Raspberry Pi é o baixo custo, pois ela possui todos os componentes que um computador de custo mais elevado possui. Dessa forma, é possível instalar um sistema operacional que irá gerenciar os componentes da placa.

Após a montagem do *cluster*, as tarefas enviadas pela rede são paralelizadas para que sejam executadas por todas as placas do cluster. A Figura 4.4 apresenta o *cluster* após a montagem.

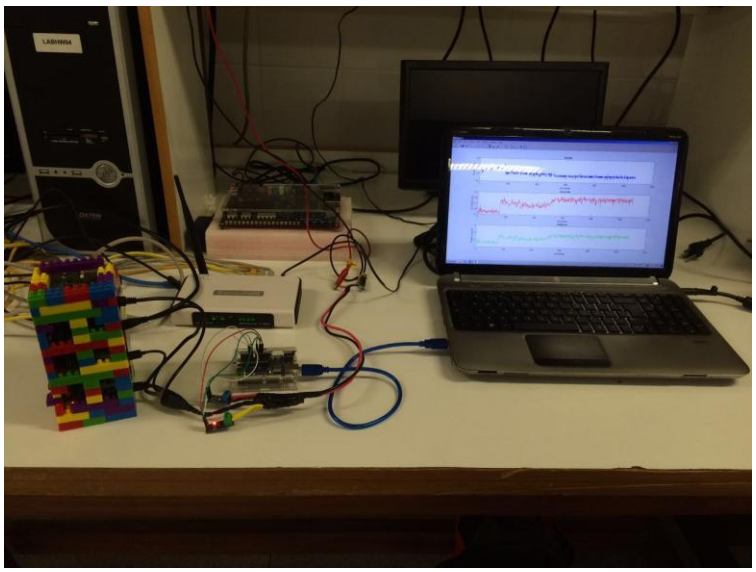

Figura 4.4. Cluster com placas Raspberry Pi

Capítulo 5

Desempenho do *Cluster* com HPL

Neste capítulo, apresentamos a análise de desempenho do *cluster* embarcado com processadores ARM e plataforma *Raspberry Pi*. Para isso, foram realizados testes que mediram o desempenho do *cluster* durante a execução de *benchmarks* que realizaram grandes quantidades de cálculos matemáticos. Além disso, também foram utilizadas diferentes implementações da biblioteca MPI, para que fosse possível medir o desempenho do *cluster* em diferentes ambientes de programação. Com o *benchmark* HPL, foi possível verificar o poder de processamento do *cluster*, já com o *benchmark* HPCC, foi possível extrair várias métricas de desempenho do *cluster* referentes ao tráfego de dados entre os processadores.

5.1 Cenários de Testes

Com o objetivo de testar as hipóteses desta dissertação, foram realizados testes que levaram em consideração diferentes cenários. Assim, foi possível analisar o desempenho e o consumo de energia do *cluster* embarcado com os *benchmarks* HPCC e HPL. Os cenários de testes utilizados são apresentados na tabela 5.1.

Tabela 5.1: Cenários de testes do *cluster* embarcado

Cenário 1	*Cluster* com o Benchmark HPL e OpenMPI
Cenário 2	*Cluster* com o Benchmark HPL e MPICH-2
Cenário 3	*Cluster* com o Benchmark HPCC e OpenMPI
Cenário 4	*Cluster* com o Benchmark HPCC e MPICH-2

5.2 Desempenho do *Cluster* com o Benchmark HPL e OpenMPI

O *benchmark* HPL resolve um sistema linear com N variáveis e ao final exibe a capacidade de processamento em Gflops/s e o tempo gasto para resolver o sistema linear. Para uma melhor análise do desempenho do *cluster* embarcado, os experimentos

levaram em consideração a resolução de sistemas lineares com diferentes valores de N, onde N é o número de variáveis do sistema linear.

Além disso, para verificar a eficiência energética do *cluster* embarcado em diferentes ambientes de programação, neste cenário, também foram realizados testes que mediram o consumo de energia durante a execução do *benchmark* HPL.

Inicialmente, foram realizados testes que mediram a capacidade de processamento do *cluster* à medida que os valores de N e o número de nós aumentavam. Assim, os testes levaram em consideração a resolução de sistemas lineares com 5000 e 10000 variáveis.

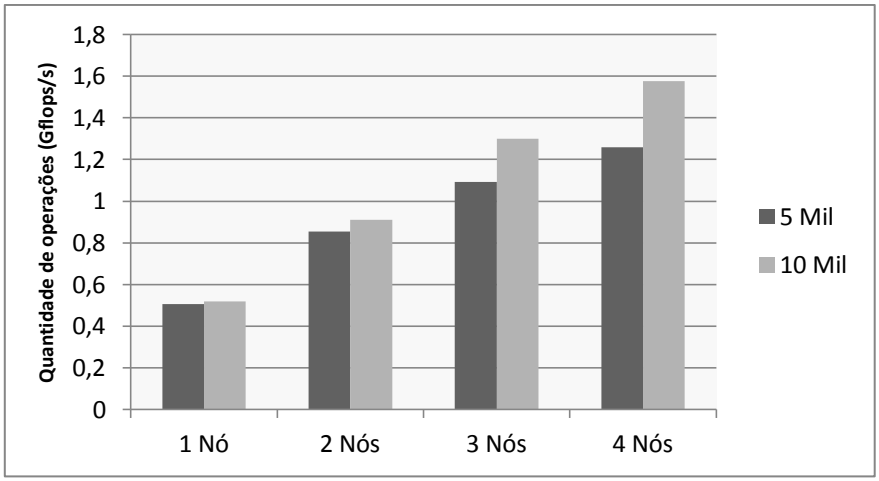

Figura 5.1: Desempenho do *cluster* com HPL e biblioteca OpenMPI

Tabela 5.1: Intervalos de confiança de 95% para N = 5000

Número de Nós	Média	Intervalo de Confiança
1	0,5062	(0,5060 – 0,5064)
2	0,8541	(0,8528 – 0,8554)
3	1,0916	(1,0895 – 1,0937)
4	1,2583	(1,2572 – 1,2594)

Tabela 5.2: Intervalos de confiança de 95% para N = 10000

Número de Nós	Média	Intervalo de Confiança
1	0,5185	(0,5183 – 0,5187)
2	0,9093	(0,9072 – 0,9114)
3	1,2991	(1,2976 – 1,3006)
4	1,5767	(1,5750 – 1,5784)

A análise do gráfico apresentado na Figura 5.1 mostra que um aumento no número de nós do *cluster* embarcado provocou um aumento no desempenho do mesmo. Para N = 5000, podemos perceber que o *cluster* composto por 4 processadores obteve um desempenho em Gflops/s 59,78% melhor, quando comparado ao *cluster* com apenas

um processador. Já para N = 10000, o *cluster* composto por 4 processadores obteve um desempenho em Gflops/s 67% melhor, quando comparado ao *cluster* com apenas 1 nó. As Tabelas 5.1 e 5.2 apresentam os valores médios sob um intervalo de confiança de 95%.

5.2.1 Tempo de Execução do *Cluster* com HPL e OpenMPI

Além de medir a capacidade de processamento do *cluster* em Gflops/s, o *benchmark* HPL também mediu o tempo gasto para realizar o cálculo do sistema linear. A Figura 5.2 apresenta os tempos de execução do HPL para a solução de um sistema linear com N = 5000 variáveis.

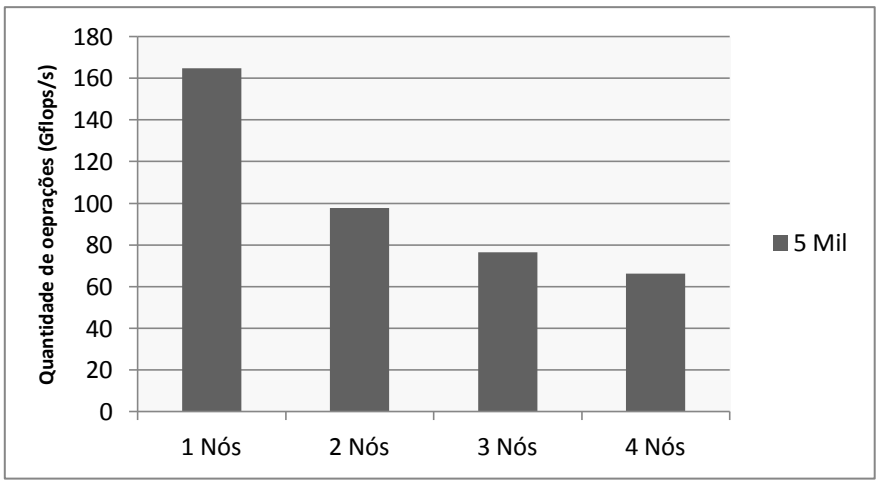

Figura 5.2. Tempos de Execução do HPL para N = 5000

Tabela 5.3: Intervalos de confiança de 95% para N = 5000

Número de Nós	Média	Intervalo de Confiança
1	164,66	(164,64 – 164,68)
2	97,61	(97,58 – 97,64)
3	76,37	(76,36 – 76,38)
4	66,25	(66,23 – 66,27)

A análise dos dados apresentados na Figura 5.2 mostra que a adição de novos processadores ao *cluster* provocou uma redução no tempo de execução do *benchmark* HPL. Além disso, podemos perceber que o *cluster* composto por quatro processadores obteve um desempenho 60% melhor, em relação ao *cluster* composto por um único processador. Ao analisar os *speedups* apresentados no gráfico da Figura 5.3, podemos perceber que o *cluster* com quatro processadores obteve um desempenho 2.5 vezes melhor quando comparado ao *cluster* com apenas um processador. Apesar de a adição

de novos nós ter provocado um aumento no desempenho do *cluster*, pode-se perceber que para a resolução de um sistema linear com 5000 variáveis não houve um aumento considerável no desempenho do *cluster*, pois em um sistema paralelo, além do tempo gasto para executar instruções, existe também o tempo gasto para realizar a comunicação entre os nós.

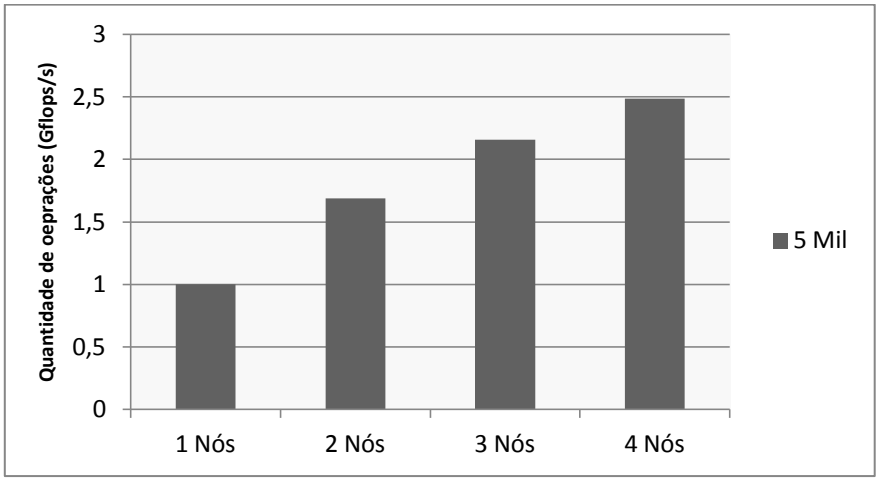

Figura 5.3: *Speedup* do HPL para N = 5000

Para verificar o comportamento do *cluster* após o aumento da carga de trabalho, também foram realizados testes que levaram em consideração a resolução de sistemas lineares com dez mil variáveis. O gráfico da Figura 5.4 apresenta os tempos de execução do *cluster* embarcado.

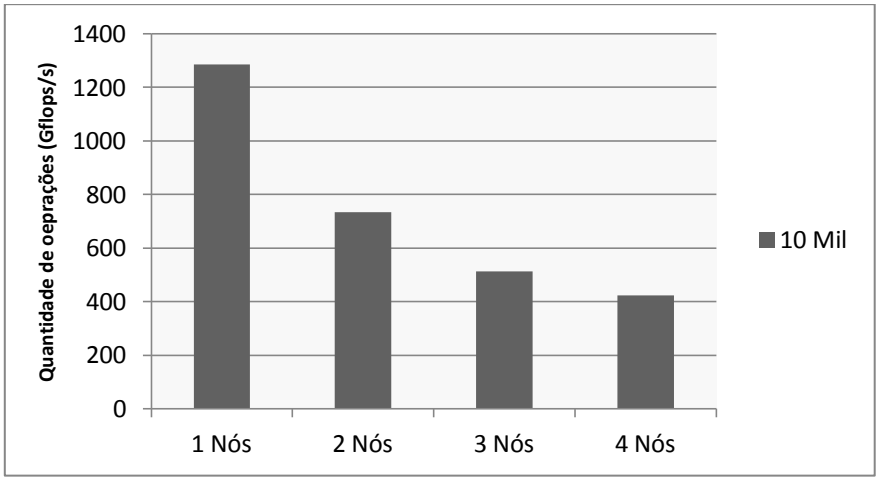

Figura 5.4: Tempos de Execução do HPL para N = 10000

Tabela 5.4: Intervalos de confiança de 95% para N = 10000

Número de Nós	**Média**	**Intervalo de Confiança**
1	1285,92	(1285,90 – 1285,94)
2	733,24	(733,20 – 733,28)
3	513,27	(513,25 – 513,29)
4	422,91	(422,89 – 422,93)

A análise do gráfico apresentado na Figura 5.4 mostra que o *cluster* composto por quatro processadores obteve um desempenho 67% melhor, quando comparado à solução sequencial com apenas um processador. Em relação ao *speedup* alcançado, pode-se perceber que o aumento do tamanho do problema, ou seja, o aumento do número de variáveis do sistema linear provocou um aumento no desempenho do *cluster*. Isso se deve ao fato de os processadores passarem a realizar mais operações sobre os dados, fazendo com que o tempo gasto durante o processamento superasse o tempo gasto durante a comunicação entre os nós. O gráfico da Figura 5.5 mostra que o *cluster* composto por quatro processadores obteve um desempenho três vezes melhor, em relação à solução sequencial com apenas um processador.

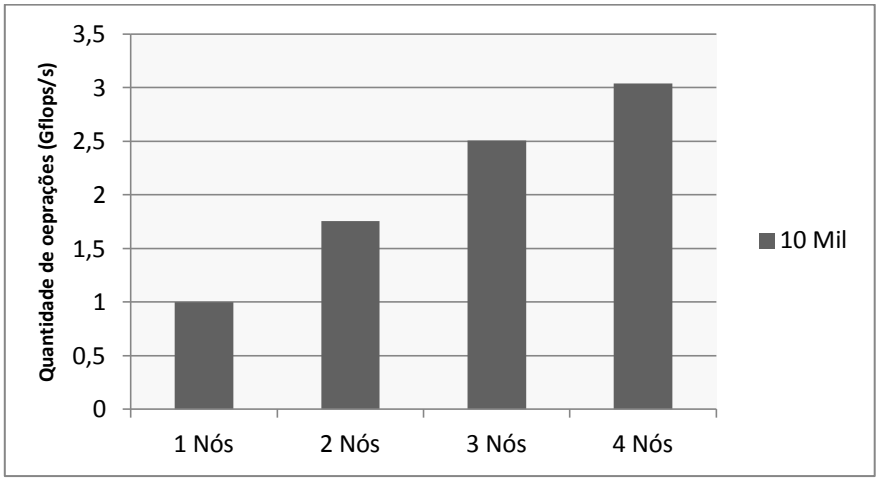

Figura 5.5: Speedup do HPL para N = 10000

5.2.2 Consumo de Energia do *Cluster* com HPL e OpenMPI

Com o objetivo de verificar a eficiência energética do *cluster* embarcado, foi utilizado um circuito real que mediu a corrente, a potência e a tensão durante a execução do *benchmark* HPL. As Figuras 5.6, 5.7, 5.8 e 5.9 apresentam o consumo energético do *cluster* embarcado.

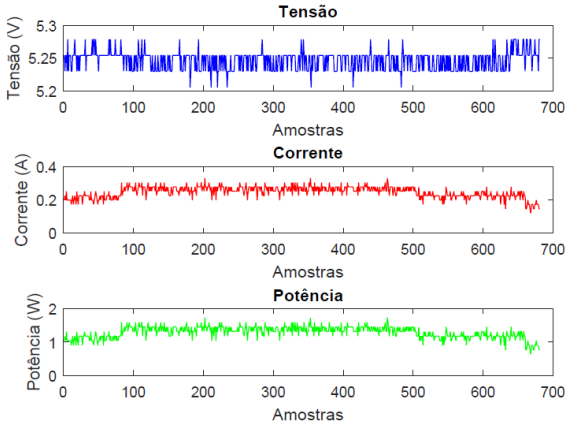

Figura 5.6: Consumo de energia do cluster com 1 processador

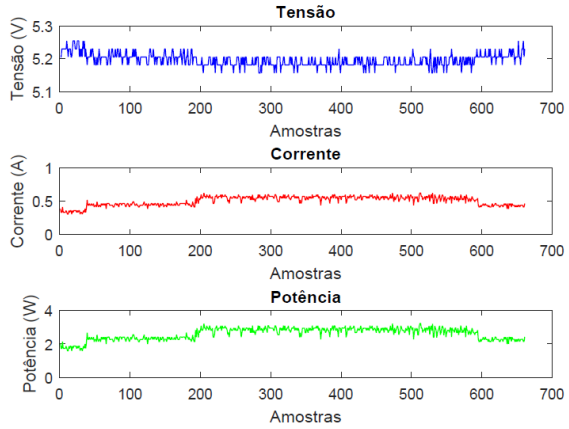

Figura 5.7: Consumo de energia do cluster com 2 processadores

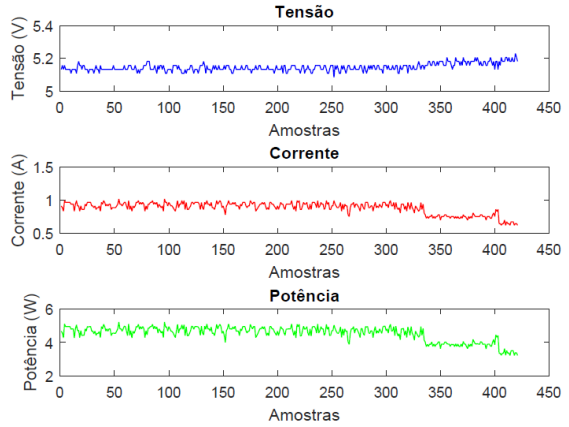

Figura 5.8: Consumo de energia do cluster com 3 processadores

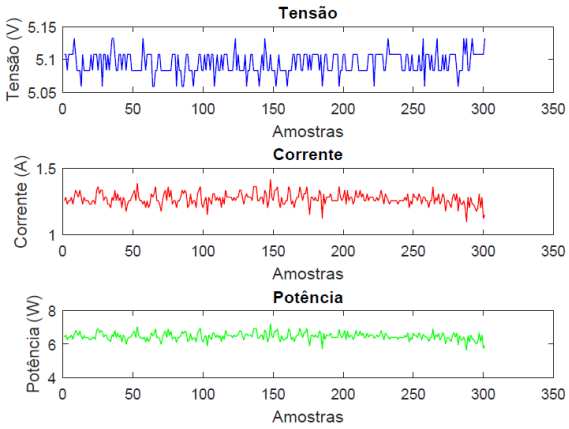

Figura 5.9: Consumo de energia do cluster com 4 processadores

A análise dos dados coletados durante a execução do HPL mostra que o aumento do número de nós provocou um aumento na potência e na corrente consumidas pelo *cluster*. Os gráficos das Figuras 5.10 e 5.11 apresentam uma comparação dos dados obtidos para o consumo de energia do *cluster*.

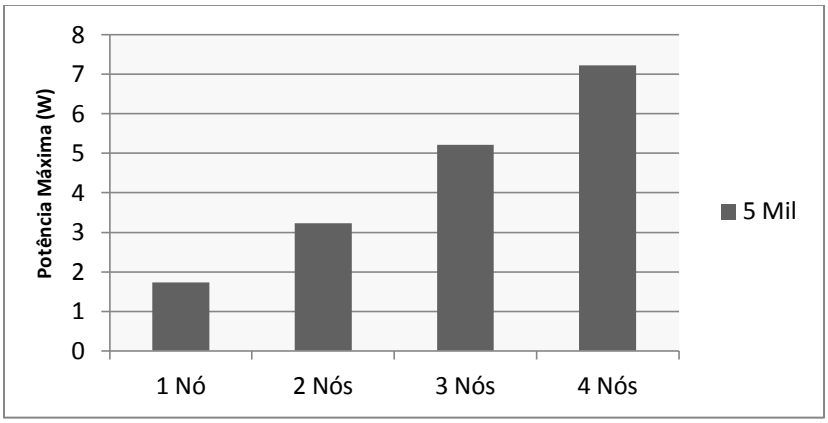

Figura 5.10: Potência Máxima do *Cluster* com Benchmark HPL

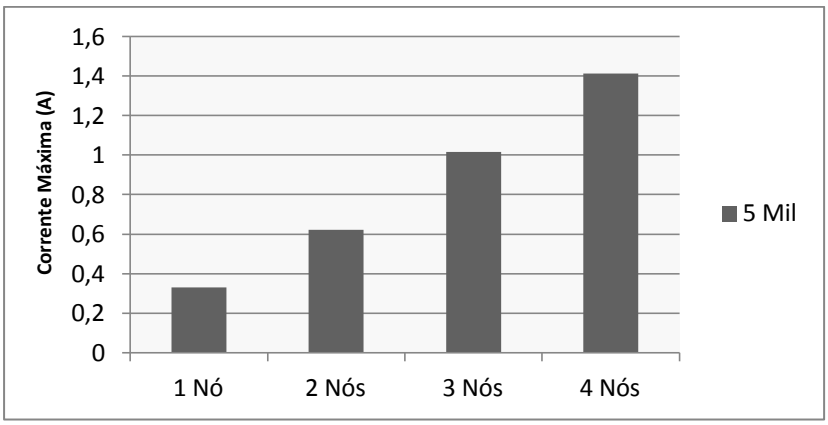

Figura 5.11: Corrente Máxima do *Cluster* com Benchmark HPL

Analisando os gráficos das Figuras 5.10 e 5.11, pode-se perceber que o aumento do número de processadores provocou um aumento no consumo de energia do *cluster*. Com relação à potência máxima obtida, tem-se que o *cluster* com quatro processadores obteve um consumo 76% maior em relação ao *cluster* com apenas um processador. Já em relação à corrente máxima, o *cluster* com quatro processadores, obteve um consumo 76,63% maior, em relação ao cluster com apenas um processador. As Tabelas 5.5 e 5.6 apresentam os valores para a potência máxima, potência média, corrente máxima e corrente média.

Tabela 5.5: Potência Máxima e Média

Número de Nós	Potência Máxima (W)	Potência Média (W)
1	1.7351	1.3210
2	3.2317	2.8395
3	5.2200	4.5292
4	7.2192	6.4615

Tabela 5.6: Potência Máxima e Média

Número de Nós	Corrente Máxima (W)	Corrente Média (A)
1	0.3302	0.2505
2	0.6209	0.5474
3	1.0171	0.8805
4	1.4134	1.2683

5.3 Desempenho do *Cluster* com o *Benchmark* HPL e Implementação MPICH-2

Esta seção apresenta o resultado e a análise dos testes realizados para medir o desempenho do *cluster* durante a execução do *benchmark* HPL com a implementação MPICH-2 da biblioteca MPI. O gráfico da Figura 5.12 apresenta o resultado dos testes realizados para N = 5000 e N = 10000. As tabelas 5.7 e 5.8 apresentam os intervalos de confiança.

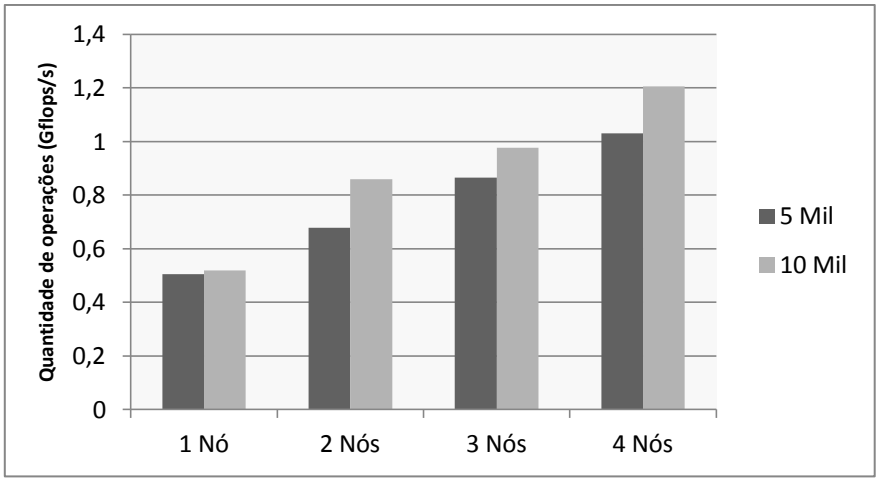

Figura 5.12: Desempenho do *cluster* com HPL e biblioteca MPICH-2

Tabela 5.7: Intervalos de confiança de 95% para N = 5000

Número de Nós	Média	Intervalo de Confiança
1	0,5059	(0,5060 – 0,5064)
2	0,6790	(0,6790 – 0,6820)
3	0,8649	(0,8608 – 0,8690)
4	1,0310	(1,0300 – 1,0310)

Tabela 5.8: Intervalos de confiança de 95% para N = 10000

Número de Nós	Média	Intervalo de Confiança
1	0,5180	(0,5170 – 0,5190)
2	0,8604	(0,8567 – 0,8641)
3	0,9772	(0,9762 – 0,9782)
4	1,2050	(1,2045 – 1,2055)

Após a realização dos testes com o *benchmark* HPL e a implementação MPICH-2 da biblioteca MPI, a análise dos dados apresentados no gráfico da Figura 5.12, permite concluir que o aumento do número de processadores provocou um aumento no desempenho do *cluster*. Esse resultado já era esperado, pois para uma quantidade pequena de dados, o desempenho de um sistema paralelo pode não ser satisfatório, devido ao *overhead* proveniente da comunicação realizada entre os diferentes processadores durante a execução do programa paralelizado. Para N = 5000, a solução paralela com quatro processadores obteve um desempenho 51% melhor, quando comparada a solução sequencial com apenas um processador. Já para N = 10000, a solução com quatro processadores obteve um aumento de 57% no desempenho.

5.3.1 Tempo de Execução do *Cluster* com HPL e MPICH-2

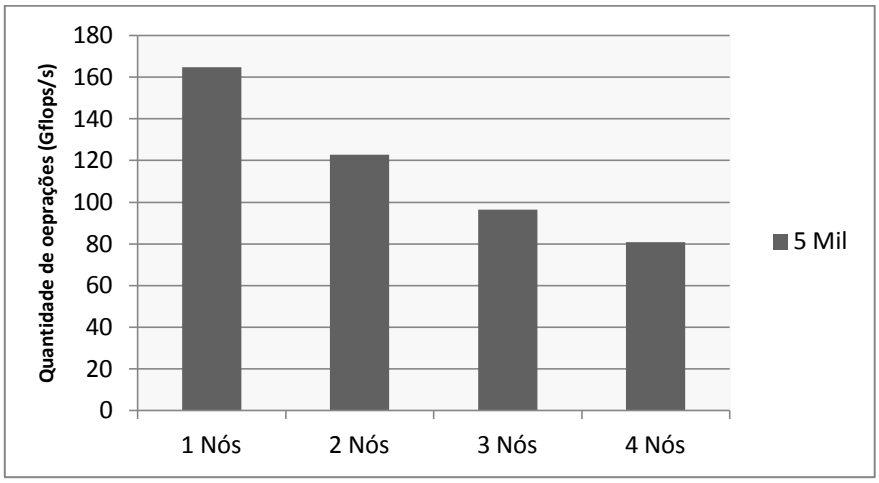

Figura 5.13. Tempos de Execução do HPL para N = 5000

Tabela 5.9: Intervalos de confiança de 95% para N = 5000

Número de Nós	Média	Intervalo de Confiança
1	164,76	(164,75 – 164,77)
2	122,78	(122,77 – 122,79)
3	96,4	(96,2 – 96,6)
4	80,86	(80,84 – 80,88)

Ao analisar os tempos de execução do HPL para N = 5000, pode-se perceber que o aumento do número de processadores do *cluster* provocou uma redução do tempo de execução do *benchmark*. Os dados apresentados no gráfico da Figura 5.13 mostram que o *cluster* com quatro processadores obteve um desempenho 51% melhor, em relação ao *cluster* com apenas um processador. Em relação ao *speedup*, tem-se que a solução com quatro processadores obteve um desempenho duas vezes melhor, em relação à solução com apenas um processador. O gráfico da Figura 5.14 apresenta os *speedups* do *cluster* embarcado.

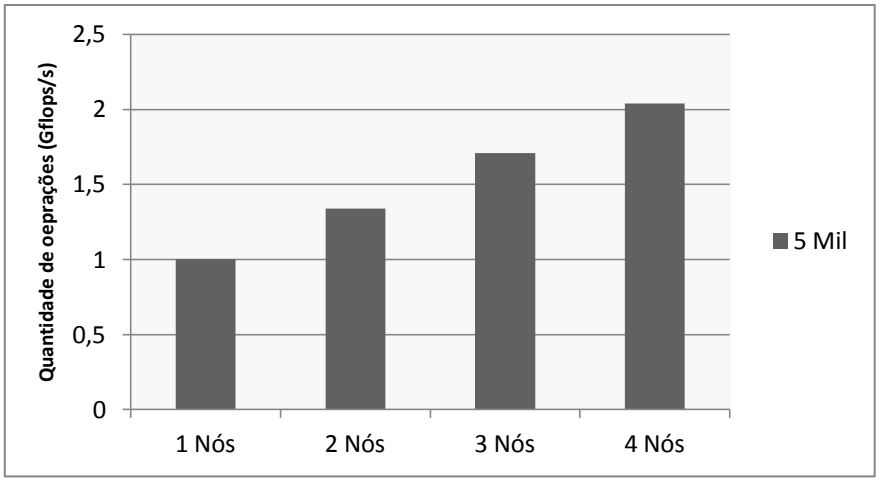

Figura 5.14: *Speedup* do HPL para N = 5000

De forma similar ao que foi feito no cenário apresentado na seção anterior, também foram realizados testes que mediram o desempenho do *cluster* com MPICH-2 e HPL com N = 10000. A análise dos dados apresentados no gráfico da Figura 5.15 mostra que o aumento do número de processadores do cluster provocou uma redução no tempo de execução do benchmark HPL. O *cluster* com quatro processadores obteve um desempenho 57% maior em relação ao *cluster* com apenas um processador. Já em relação aos *speedups* apresentados no gráfico da Figura 5.16, pode-se perceber que o *cluster* com quatro processadores obteve um desempenho 2,3 vezes maior em relação ao *cluster* com apenas um processador.

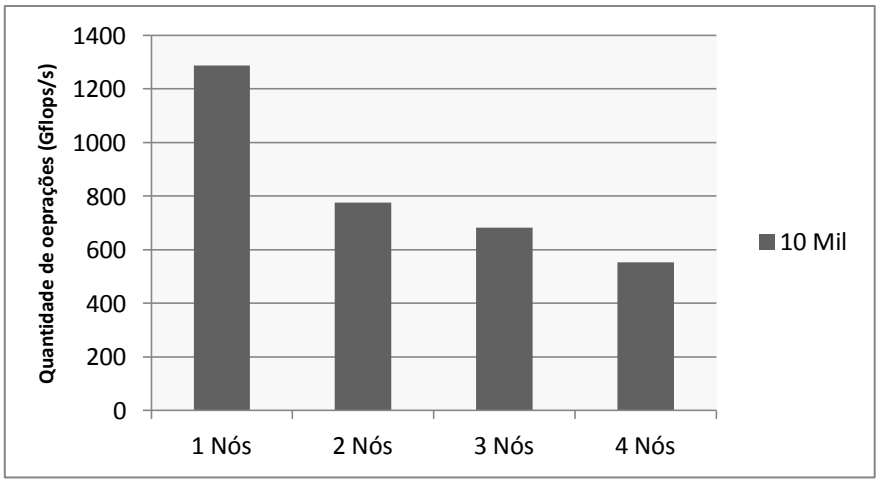

Figura 5.15. Tempos de Execução do HPL para N = 10000

Tabela 5.10: Intervalos de confiança de 95% para N = 10000

Número de Nós	Média	Intervalo de Confiança
1	1287,3	(1287,1 – 1287,5)
2	775,92	(775,91 – 775,93)
3	682,32	(682,31 – 682,32)
4	553,3	(553,29 – 553,3)

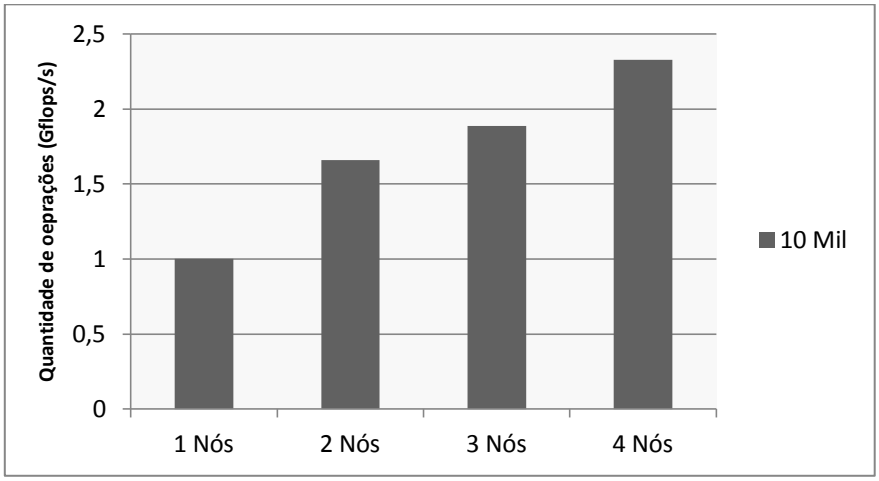

Figura 5.16: *Speedup* do HPL para N = 10000

5.3.2 Consumo de Energia do *Cluster* com HPL e MPICH-2

Após os testes que mediram o desempenho do *cluster*, também foram realizados testes que mediram o consumo de energia durante a execução do *benchmark* HPL e implementação MPICH-2 da biblioteca MPI. Os gráficos das Figuras 5.17, 5.18. 5.19 e 5.20 apresentam os valores para a tensão, corrente e potência.

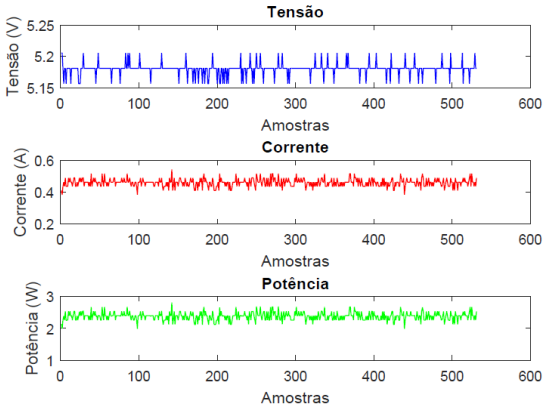

Figura 5.17: Consumo de energia do *cluster* com 1 processador

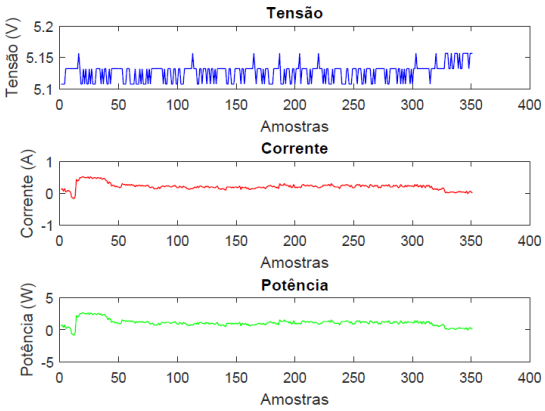

Figura 5.18: Consumo de energia do *cluster* com 2 processadores

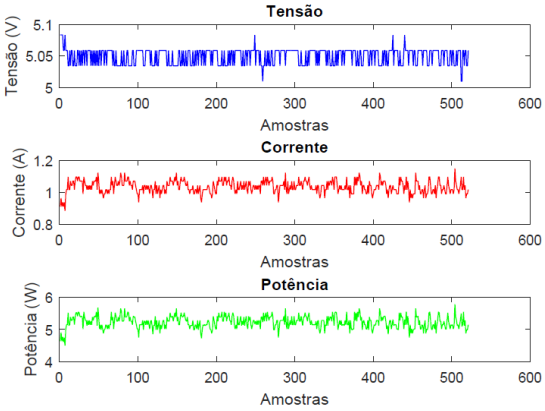

Figura 5.19: Consumo de energia do *cluster* com 3 processadores

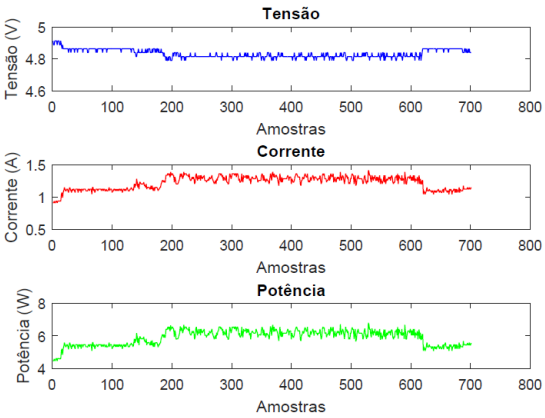

Figura 5.20: Consumo de energia do *cluster* com 4 processadores

Em relação ao consumo de energia, pode-se verificar que o aumento do número de processadores do cluster provocou um aumento no consumo de energia. Os gráficos das Figuras 5.21 e 5.22 apresentam uma comparação dos dados obtidos para o consumo de energia do *cluster*.

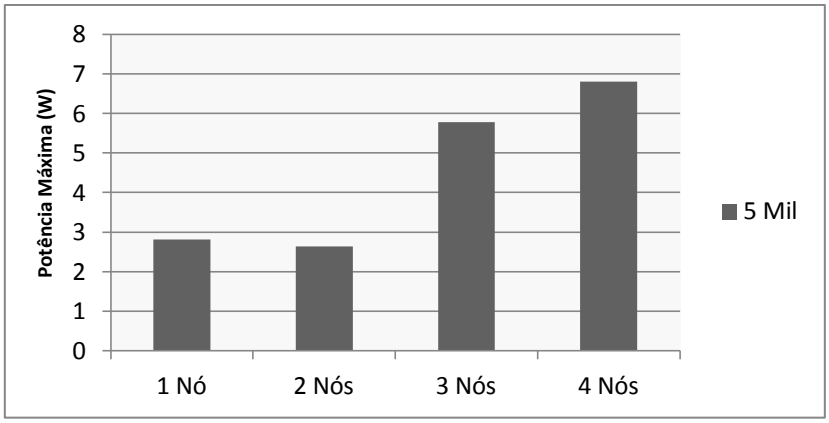

Figura 5.21: Potência Máxima do *Cluster* com Benchmark HPL

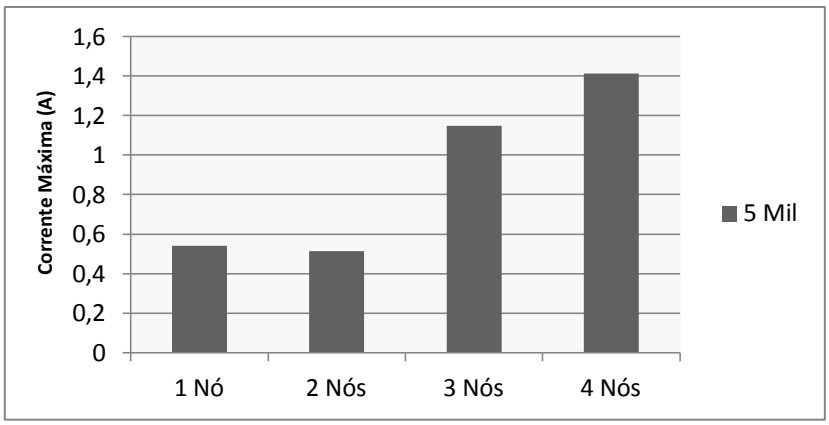

Figura 5.22: Corrente Máxima do *Cluster* com Benchmark HPL

A análise dos dados comparados apresentados nos gráficos das Figuras 5.21 e 5.22 mostram que o aumento do número de processadores do *cluster* provocou um aumento no consumo de energia. Levando em consideração a potência máxima alcançada, pode-se perceber que o *cluster* com quatro processadores obteve um consumo 58% maior, quando comparado ao *cluster* com apenas um processador. Em relação à corrente máxima, o *cluster* com quatro processadores obteve um consumo 62% maior.

Tabela 5.11: Potência Máxima e Média

Número de Nós	**Potência Máxima (W)**	**Potência Média (W)**
1	2.8059	2.3782
2	2.6439	1.0774
3	5.7855	5.1932
4	6.8047	5.8923

Tabela 5.12: Corrente Máxima e Média

Número de Nós	Corrente Máxima (W)	Corrente Média (A)
1	0.5416	0.4591
2	0.5152	0.2103
3	1.1492	1.0349
4	1.4134	1.2197

5.4 Análise Comparada do Desempenho do *Cluster* com HPL e bibliotecas OpenMPI e MPICH-2

As seções anteriores apresentaram os resultados das análises de desempenho do *cluster* embarcado com processadores ARM e *benchmark* HPL. Assim, foi possível verificar o comportamento do *cluster* durante a resolução de problemas de diferentes tamanhos e em diferentes ambientes de programação. Esta seção faz uma análise comparativa do desempenho do *cluster*, a fim de verificar qual das implementações da biblioteca MPI obteve o melhor desempenho.

Levando em consideração a resolução de sistemas lineares com cinco mil e dez mil variáveis, pode-se observar que o *cluster* implementado com a biblioteca OpenMPI obteve um melhor desempenho, quando comparado ao *cluster* implementado com a biblioteca MPICH-2. Os gráficos das Figuras 5.23 e 5.24 apresentam os dados comparados.

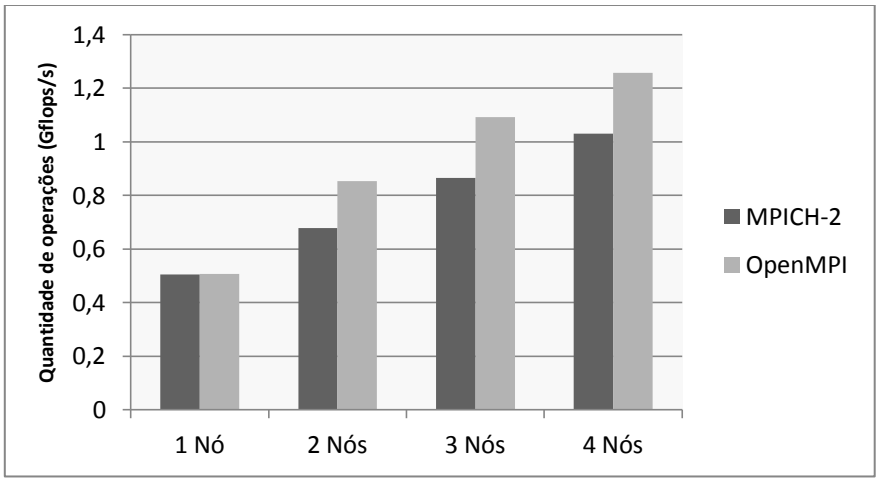

Figura 5.23: Comparação das bibliotecas OpenMPI e MPICH-2 com N = 5000

A análise dos dados apresentados no gráfico da Figura 5.23 mostra que para a solução de sistemas lineares com cinco mil variáveis, o desempenho do *cluster*

composto por quatro processadores e com a biblioteca OpenMPI foi 18% maior, quando comparado ao *cluster* com a biblioteca MPICH-2.

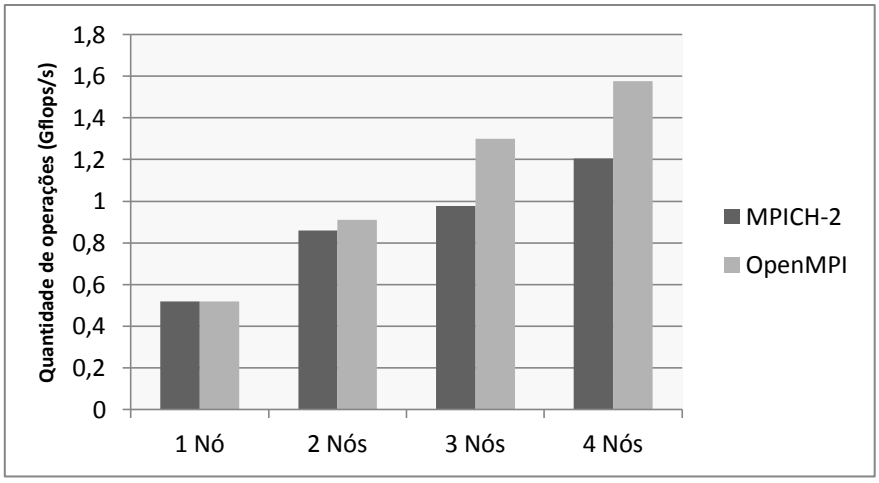

Figura 5.24: Comparação das bibliotecas OpenMPI e MPICH-2 com N = 10000

Para a resolução de sistemas lineares com dez mil variáveis, a análise dos dados apresentados no gráfico da Figura 5.24 mostra que o *cluster* com a biblioteca OpenMPI também obteve um melhor desempenho, quando comparado ao *cluster* com a biblioteca MPICH-2. Para quatro processadores, o aumento no desempenho foi de 24%.

Em relação ao tempo gasto pelo *cluster* para resolver um sistema linear com cinco mil variáveis, tem-se que o *cluster* implementado com a biblioteca OpenMPI obteve um desempenho superior, quando comparado ao *cluster* implementado com a biblioteca MPICH-2. A análise dos dados apresentados nos gráficos das Figuras 5.25 e 5.26 mostram que o *cluster* com quatro processadores e biblioteca OpenMPI obteve um tempo de execução 18% menor, em relação ao *cluster* com MPICH-2. Além disso, quando comparado ao *cluster* com apenas um processador, o *cluster* com OpenMPI alcançou um *speedup* de 2,5, enquanto o *cluster* com MPICH-2 alcançou um *speedup* de 2.

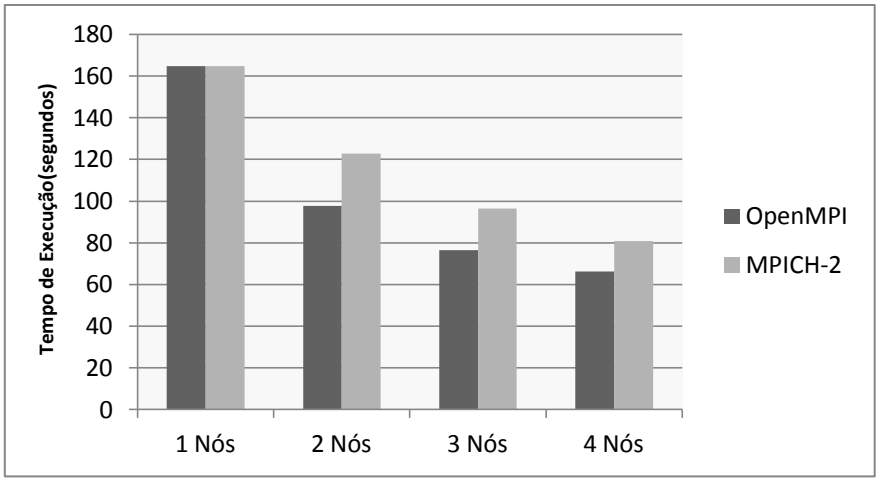

Figura 5.25: Tempos das bibliotecas OpenMPI e MPICH-2 com N = 5000

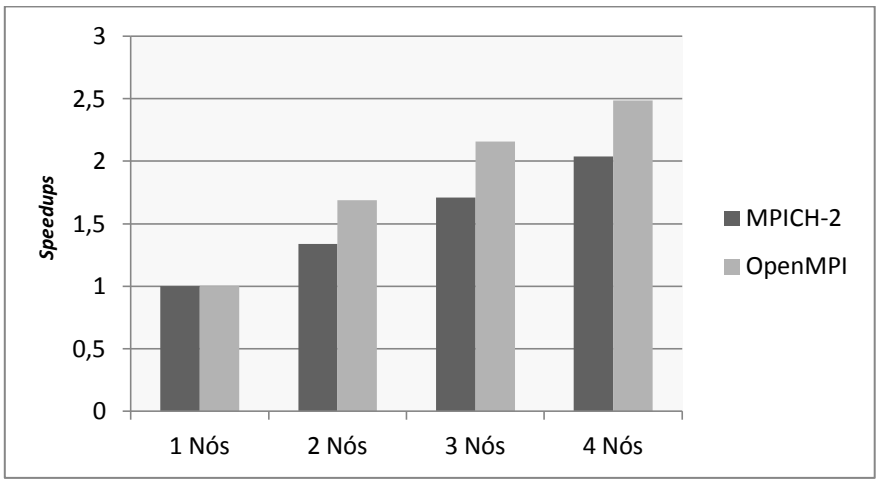

Figura 5.26: *Speedups* das bibliotecas OpenMPI e MPICH-2 com N = 5000

Em relação ao comportamento do *cluster* durante a resolução de um sistema linear com dez mil variáveis, tem-se que o *cluster* implementado com a biblioteca OpenMPI também obteve um melhor desempenho. A análise dos dados apresentados nos gráficos das Figuras 5.27 e 5.28 mostram que para o *cluster* com quatro processadores, o tempo de execução com a biblioteca OpenMPI foi 24% menor, em relação ao *cluster* com a biblioteca MPICH-2.

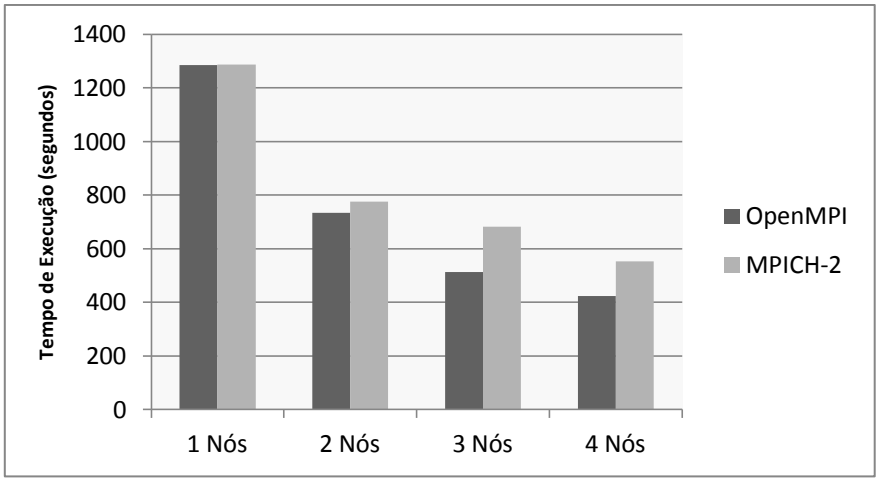

Figura 5.27: Tempos das bibliotecas OpenMPI e MPICH-2 com N = 10000

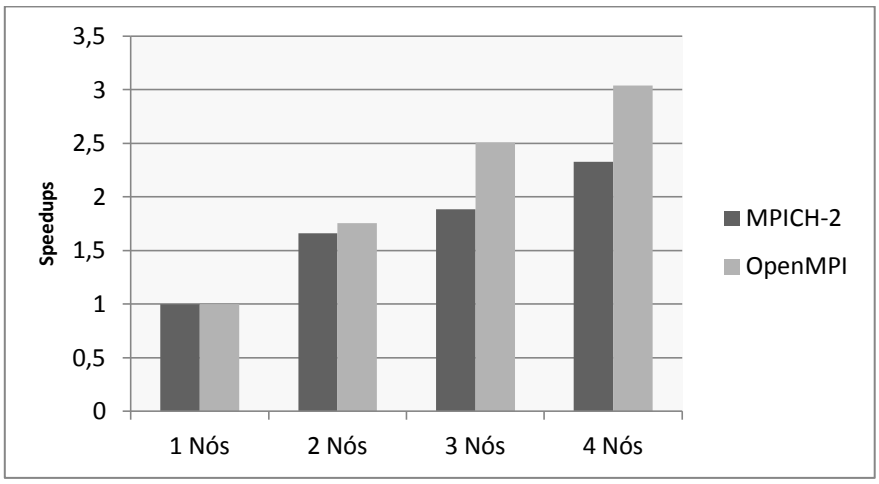

Figura 5.28: *Speedups* das bibliotecas OpenMPI e MPICH-2 com N = 10000

Além da capacidade de processamento e do tempo de execução, também foi medido o consumo de energia do *cluster* com as bibliotecas OpenMPI e MPICH-2. A analise dos dados apresentados nos gráficos 5.29 e 5.30 mostram que a potência máxima alcançada pelo *cluster* com a biblioteca OpenMPI foi 5,7% maior, quando comparada ao *cluster* com a biblioteca MPICH-2. Assim, pode-se concluir que o aumento no poder de processamento do *cluster* com a biblioteca OpenMPI fez com que o *cluster* consumisse mais energia durante o processamento.

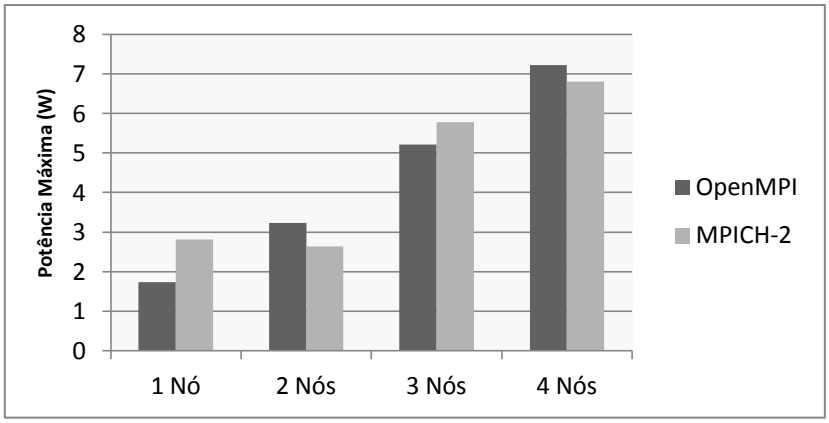

Figura 5.29: Potência Máxima com N = 5000

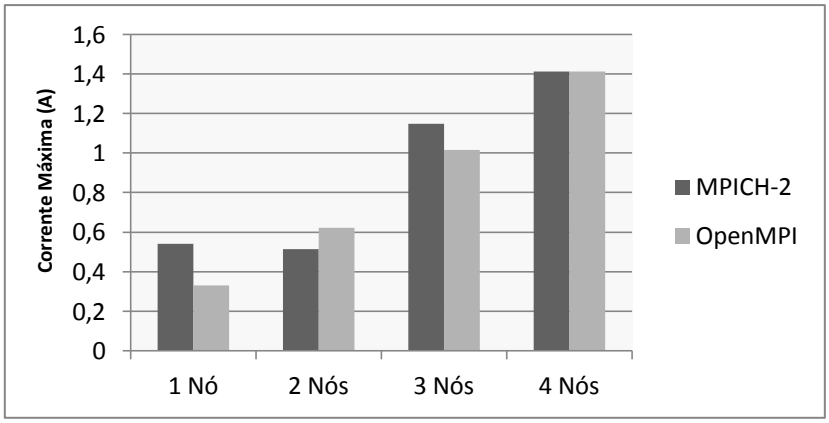

Figura 5.30: Corrente Máxima com N = 5000

Capítulo 6

Desempenho do *Cluster* com HPCC

Este capítulo apresenta o resultado dos testes realizados e que tiveram como objetivo medir e analisar o desempenho do *cluster* embarcado durante a execução do *benchmark* HPCC. Além disso, também foram utilizadas diferentes implementações da biblioteca MPI com objetivo de verificar o impacto provocado por diferentes ambientes de programação.

6.1 Desempenho do *cluster* com HPCC e bibliotecas OpenMPI e MPICH-2

Com o *benchmark* FFT, foi possível verificar a capacidade de processamento do *cluster* após a realização do cálculo da transformada rápida de Fourier. Os dados coletados durante os testes são apresentados no gráfico da Figura 6.1.

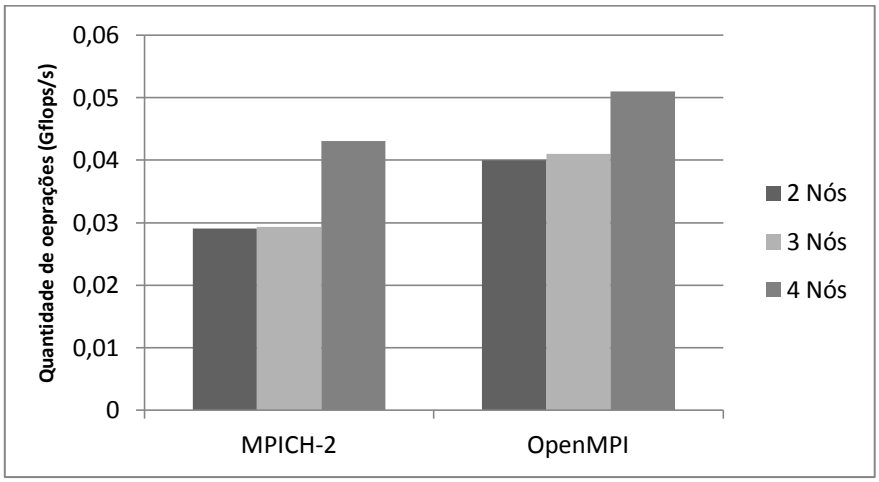

Figura 6.1: *Benchmark* FFT

A análise dos dados apresentados na Figura 6.1 mostram que o *cluster* com o *benchmark* FFT e implementação OpenMPI da biblioteca MPI alcançou um melhor desempenho. Para o *cluster* composto por quatro processadores, pode-se perceber que a implementação OpenMPI obteve um desempenho 15% maior, em relação a implementação MPICH-2.

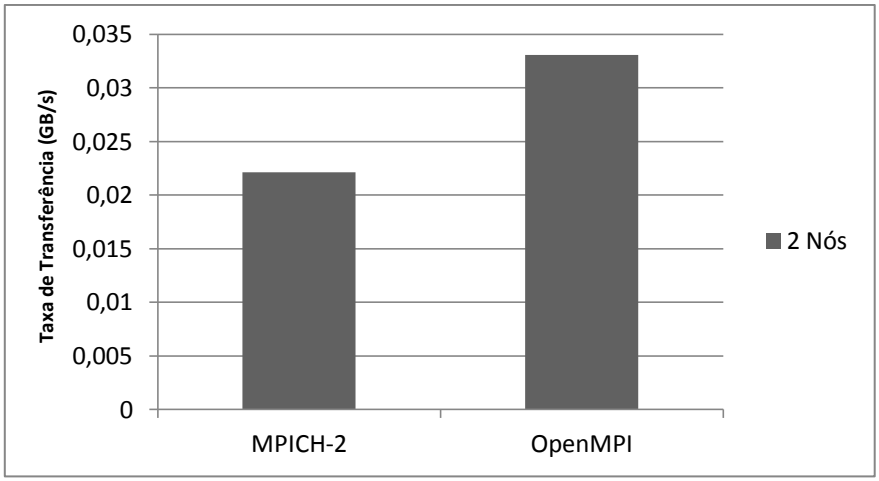

Figura 6.2: *Benchmark* PTRANS

Em relação ao *benchmark* PTRANS, que realiza a medição da taxa de transferência de dados da memória, a análise dos dados apresentados na Figura 6.2 mostra que a taxa de transferência foi 33% maior quando a biblioteca OpenMPI foi utilizada.

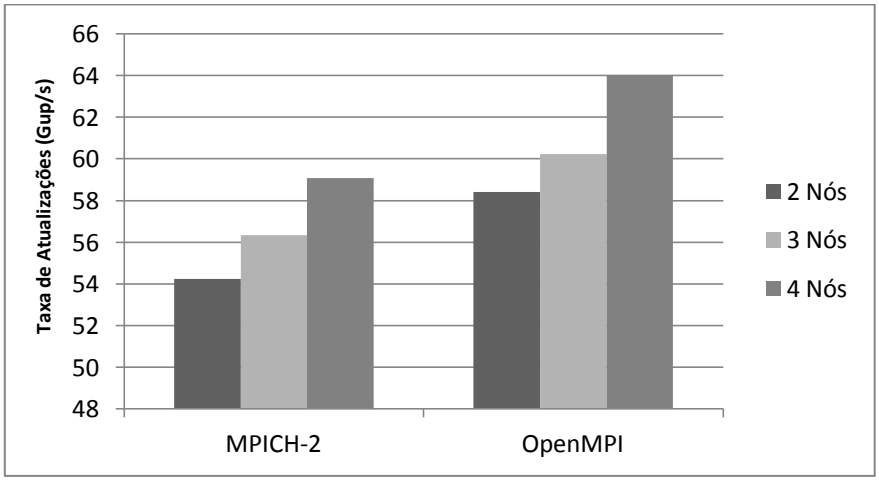

Figura 6.3: *Benchmark* RandomAccess

O *benchmark* RandomAccess mede a taxa de atualização de dados na memória. Para isso, um inteiro é lido da memória, atualizado no processador e em seguida é escrito na mesma posição de memória. Assim, a análise dos dados apresentados no gráfico da Figura 6.3 mostra que para um *cluster* com quatro processadores, a taxa de atualizações da memória foi 8% maior quando a biblioteca OpenMPI foi utilizada.

Com o *benchmark* DGEMM, foi possível medir a taxa de execução de operações com números de ponto flutuante com precisão dupla durante a multiplicação de matrizes formadas por números reais. A análise dos dados apresentados no gráfico da Figura 6.4 mostra que o *cluster* com a implementação OpenMPI obteve um desempenho 7,7% maior, em relação ao *cluster* com a implementação MPICH-2.

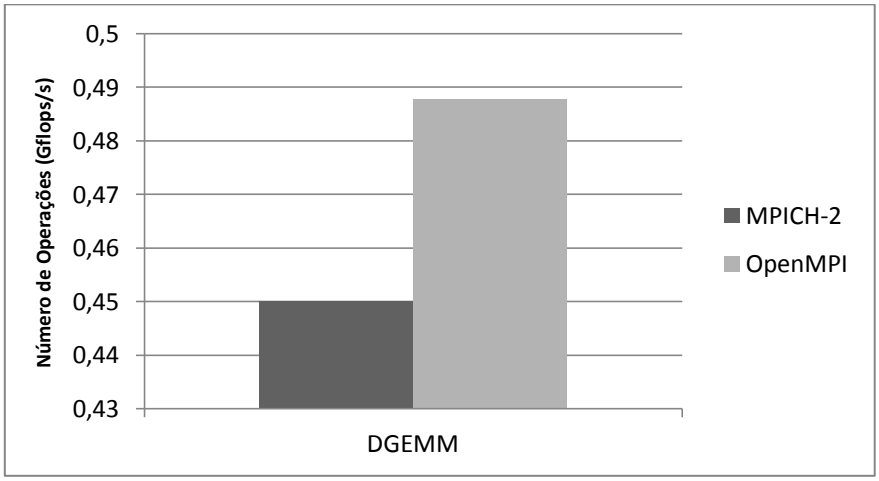

Figura 6.4: Benchmark DGEMM

Com o *benchmark* b_eff foi possível medir a largura de banda utilizada durante a comunicação entre as placas. A análise do gráfico apresentado na Figura 6.5 mostra que a utilização da biblioteca OpenMPI obteve uma largura de banda 8% maior, em relação a biblioteca MPICH-2.

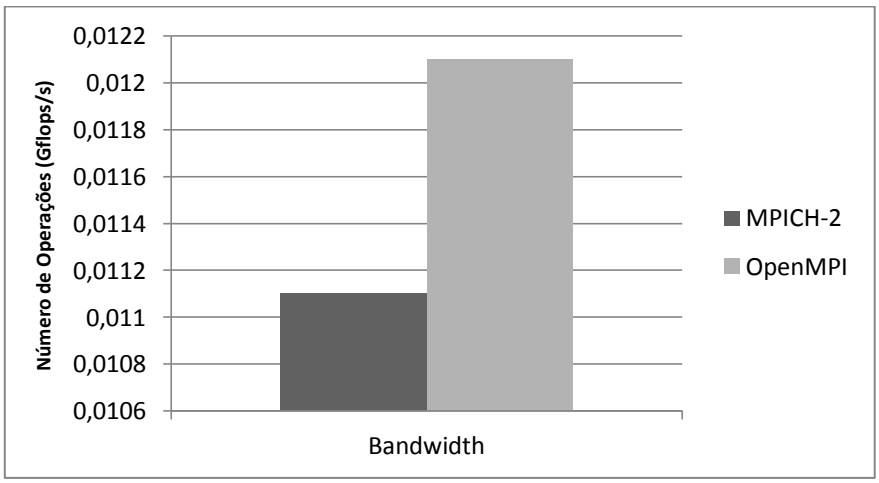

Figura 6.5: *Benchmark* Bandwidth

6.2 Consumo de Energia do *Cluster* com HPCC e MPICH-2

Além de medir e analisar a capacidade de processamento do *cluster* com o *benchmark* HPCC, também foram realizados testes que mediram o consumo de energia do cluster durante a execução do benchmark HPCC. Os gráficos das figuras 6.6, 6.7 e 6.8 apresentam os valores obtidos durante os testes para a tensão, corrente e potência.

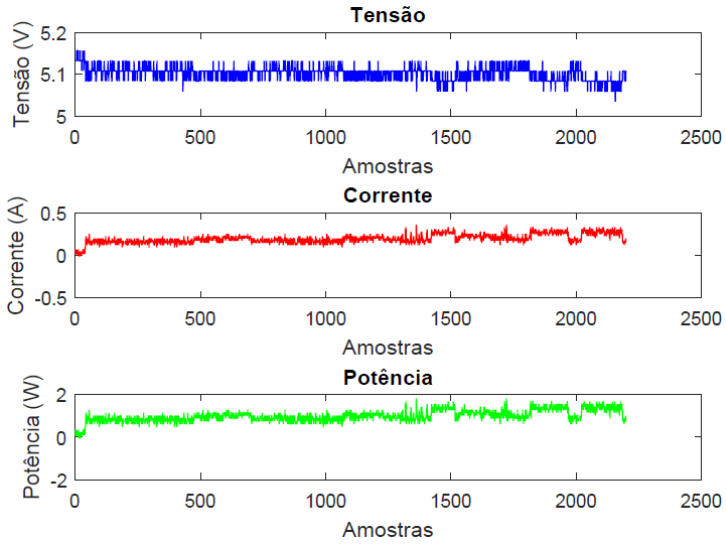

Figura 6.6: Consumo de energia HPCC e MPICH-2 com 2 processadores

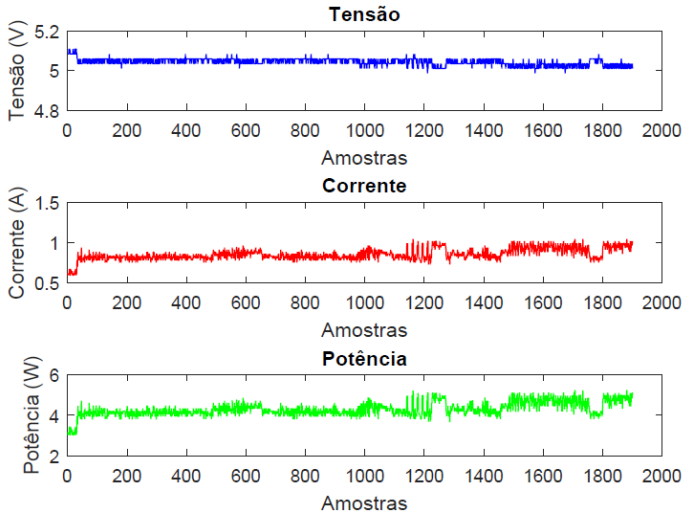

Figura 6.7: Consumo de energia HPCC e MPICH-2 com 3 processadores

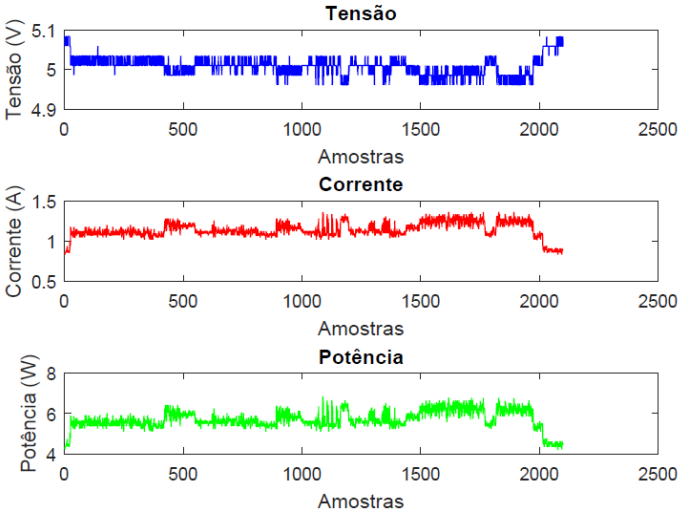

Figura 6.8: Consumo de energia HPCC e MPICH-2 com 4 processadores

A análise dos dados apresentados nos gráficos das Figuras 6.6, 6.7 e 6.8 mostra que a potência máxima obtida pelo *cluster* com quatro processadores foi 73% maior, em relação ao *cluster* com apenas dois processadores.

Tabela 6.1: Potência Máxima e Média MPICH-2

Número de Nós	Potência Máxima (W)	Potência Média (W)
2	1,8304	0,9919
3	5,2535	4,3007
4	6,8495	5,6991

Tabela 6.2: Potência Máxima e Média MPICH-2

Número de Nós	Corrente Máxima (A)	Corrente Média (A)
2	0,3567	0,1945
3	1,0436	0,8530
4	1,3606	1,1383

6.3 Consumo de Energia do *Cluster* com HPCC e OpenMPI

Neste cenário de testes, foi analisado o consumo de energia do cluster durante a execução do benchmark HPCC e biblioteca OpenMPI. Os gráficos das Figuras 6.9, 6.10 e 6.11 apresentam os resultados dos testes para a tensão, corrente e potência.

Levando em consideração a potência máxima obtida durante os testes, pode-se perceber que o cluster com quatro processadores obteve um consumo 13% maior, em relação ao *cluster* com apenas dois processadores.

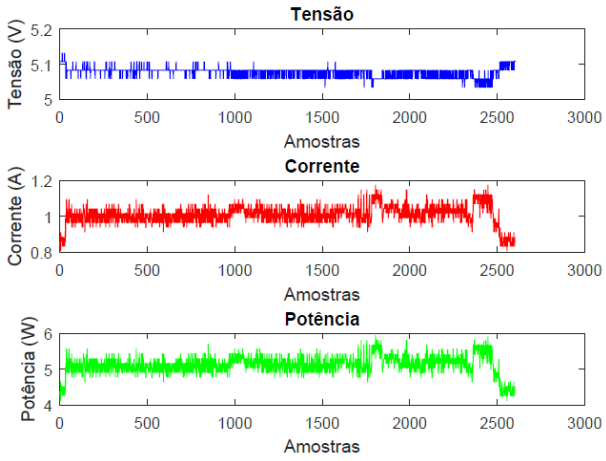

Figura 6.9: Consumo de energia HPCC e OpenMPI com 2 processadores

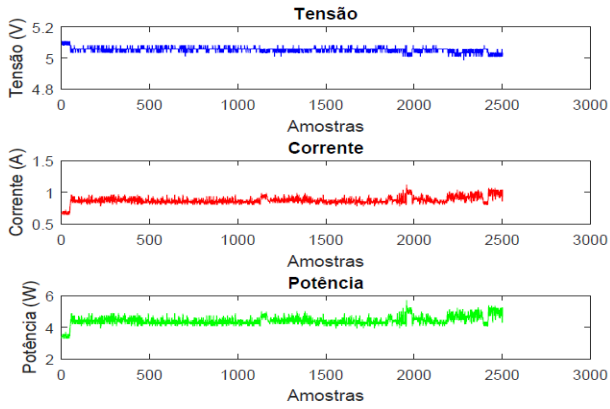

Figura 6.10: Consumo de energia HPCC e OpenMPI com 3 processadores

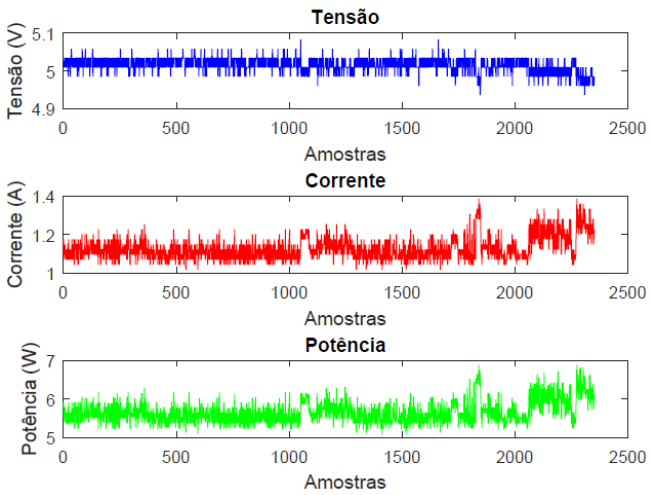

Figura 6.11: Consumo de energia HPCC e OpenMPI com 4 processadores

Tabela 6.3: Potência Máxima e Média OpenMPI

Número de Nós	Potência Máxima (W)	Potência Média (W)
2	5,9473	5,1149
3	5,7074	4,4146
4	6,8809	5,6557

Tabela 6.4: Corrente Máxima e Média OpenMPI

Número de Nós	Corrente Máxima (A)	Corrente Média (A)
2	1,1759	1,0075
3	1,1228	0,8736
4	1,3870	1,1276

6.4 Análise Comparada do Consumo de Energia

Após a realização dos testes com o benchmark HPCC e as implementações MPICH-2 e OpenMPI das bibliotecas MPI, pode-se perceber que a biblioteca MPICH-2 obteve um menor consumo de energia. Os gráficos das Figuras 6.9 e 6.10 apresentam os resultados comparados.

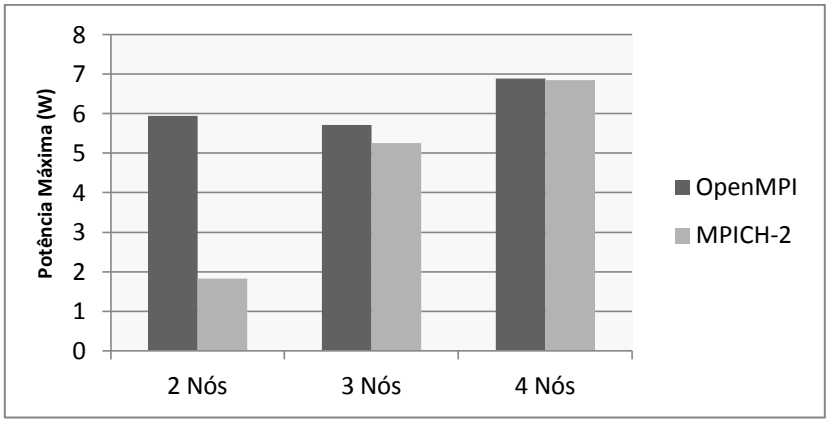

Figura 6.12: Potência Máxima com N = 5000

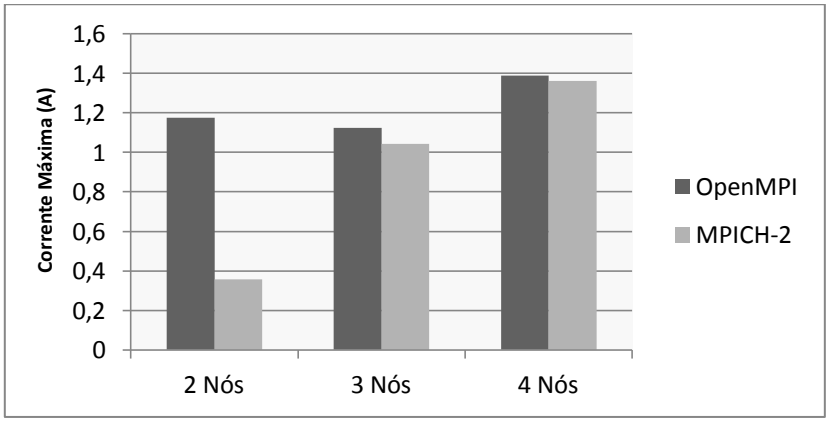

Figura 6.13: Corrente Máxima com N = 5000

Capítulo 7

Conclusões

Esta dissertação teve como objetivo medir e analisar o desempenho de um *cluster* embarcado com processadores ARM e plataforma Raspberry Pi. Para isso, foram criados quatro cenários de testes que levaram em consideração os *benchmarks* HPL e HPCC, além das bibliotecas de processamento paralelo MPICH-2 e OpenMPI.

Após a realização dos testes, foi possível verificar que em relação à capacidade de processamento do *cluster* embarcado com a biblioteca OpenMPI e *benchmark* HPL, o aumento do número de variáveis do sistema linear, de cinco mil para dez mil, fez com que o *cluster* alcançasse um melhor desempenho. Para cinco mil variáveis, o *cluster* com quatro processadores obteve um desempenho 59,78% melhor em relação ao *cluster* com apenas um processador. Já para o *cluster* com quatro processadores e um sistema linear com dez mil variáveis, o aumento no desempenho foi de 67%. O *speedup* do *cluster* com quatro processadores e um sistema linear com dez mil variáveis foi maior, quando comparado ao *cluster* com quatro processadores e um sistema linear com cinco mil variáveis. Quando a biblioteca MPICH-2 foi utilizada, observou-se um comportamento semelhante, pois, para um sistema linear com cinco mil variáveis e um *cluster* com quatro processadores, o aumento no desempenho foi de 51%, já para um sistema linear com dez mil variáveis, o aumento foi de 57%. Assim, pode-se perceber que o *cluster* não terá um bom desempenho ao resolver problemas computacionais simples, pois o *overhead* existente na comunicação entre os processadores pode prejudicar o desempenho total.

Após a realização dos testes com as bibliotecas OpenMPI e MPICH-2, foi possível verificar que a biblioteca OpenMPI permitiu que o *cluster* alcançasse um melhor desempenho. O *cluster* com quatro processadores e biblioteca OpenMPI obteve um desempenho 18% maior durante a resolução de um sistema linear com cinco mil variáveis, e 24% maior durante a resolução de um sistema linear com dez mil variáveis.

Com o *benchmark* HPCC também foram realizados testes que mediram o desempenho do *cluster* durante a sua execução. A análise comparada mostra que o cluster com a implementação OpenMPI da biblioteca MPI obteve um melhor desempenho, assim como ocorreu com o *benchmark* HPL.

Nesta dissertação, também foram realizados testes que mediram o consumo de energia real do *cluster* embarcado durante a execução dos *benchmarks* e com diferentes implementações da biblioteca MPI. A análise comparada dos resultados obtidos, permite concluir que apesar da biblioteca OpenMPI melhorar significativamente o desempenho do *cluster*, ela fez com que o consumo de energia aumentasse, pois o aumento da capacidade de processamento do cluster veio acompanhado do aumento da potência gasta durante a execução dos *benchmarks*.

Como trabalhos futuros, sugerimos a realização de testes que possam medir o desempenho do *cluster* durante a execução de algoritmos de segurança em diferentes plataformas embarcadas e implementações da biblioteca MPI.

REFERÊNCIAS

Rauber, T., Rünger, G. Parallel Programming: for Multicore and Cluster Systems 2nd ed. 2013 Edition

Moore, G. E. "Cramming more components onto integrated circuits". Electronics,Vol. 38, nº 8, Abril 1965.

Blume, H. *et al*. "OpenMP-based parallelization on an MPCore multiprocessor platform – A performance and power analysis". Journal of Systems Architecture 54.

MPI. Disponível em: <http://www.open-mpi.org/>. Acessado em: 20 de junho de 2016.

OpenMP. Disponível em: <http://openmp.org/wp/>. Acessado em: 17 de Junho de 2016.

Bez et al. Análise da Eficiência Energética de uma Aplicação HPC de Geofísica em um Cluster de Baixo Consumo. WSCAD 2015.

Reed, D. A. and Dongarra, J. (2015). Exascale Computing and Big Data. Communication of the ACM, 58(7):56–68.

Wazlawick, R. Metodologia de Pesquisa para Ciência da Computação. 6º Edição.

Lima, F. et al. Design and Performance of a Low Cost *Cluster* using ARM-based Platform. PDPTA 2016

Padoin, E. L., Velho, P., de Oliveira, D. A. G., Navaux, P. O. A., and Mehaut, J.-F. (2014). Tuning Performance and Energy Consumption of HPC Applications on ARM MPSoCs.

HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. Disponível em: <http://www.netlib.org/benchmark/hpl/> Acessado em: 20 de julho de 2016.

Freund, E. Estatística aplicada: economia, administração e contabilidade – 11º Edição. Bookman, 2006

Larson, R. Estatística aplicada – 4º Edição. São Paulo: Pearson Prentice Hall, 2010.

Pacheco, Peter S. "An introduction to parallel programming".

Silberschatz, Abraham, *et al*. "Sistemas Operacionais – Conceitos e Aplicações". 2013.

Gebali, Fayez. "Algorithms and Parallel Computing". 2011.

Jin, Haoqiang. "High performance computing using MPI and OpenMP on multi-core parallel systems". 2011.

Raspberry Pi. Disponível em: <http://www.raspberrypi.org/>. Acessado em: 15 de Março de 2015.

Raspbian. Disponível em: <http://www.raspbian.org/>. Acessado em: 15 de Março de 2015.

ARM Company Profile. Disponível em: <http://www.arm.com/about/company-profile/index.php>. Acessado em 01 de Junho de 2015.

Neill, Alexander Shabarshin Richard, and Carloni, Luca P. "A heterogeneous parallel system running OpenMPI on a broadband network of embedded set-top devices". In Proceedings of the 7th ACM International Conference on Computing Frontiers (CF '10), New York, NY, USA, pages 187-196, 2010.

McCool, Michael *et al*. "Structured Parallel Programming – Patterns for Efficient Computation".

Zomaya, Albert Z. *et al.* "Energy Efficient Distributed Computing Systems". 2012.

Cox, Simon J. *et al*. "Iridis-pi: a low-cost, compact demonstration cluster". Cluster Computing, June 2013.

Silva, Renato *et al*. “avaliação do desempenho de Threads em user level utilizando sistema operacional Linux”. Revista de Informática Teórica e Aplicada, 2011.

Furlinger, Karl et al. The AppleTV-Cluster: Towards Energy Efficient Parallel Computing on Consumer electronic Devices.

H. Blume et al., “OpenMP-based parallelization on an MPCore multiprocessor platform – A performance and power analysis”, Journal of Systems Architecture 54, 2008.