CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

ANÁLISE DE DESEMPENHO E CONSUMO ENERGÉTICO DE UM CLUSTER BASEADO EM COMPUTADORES DE PLACA ÚNICA

Gustavo Rostirolla

Lajeado, junho de 2014

Gustavo Rostirolla

ANÁLISE DE DESEMPENHO E CONSUMO ENERGÉTICO DE UM CLUSTER BASEADO EM COMPUTADORES DE PLACA ÚNICA

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. Ph.D. Pedro Antônio Madeira de Campos Velho

Lajeado, junho de 2014

Gustavo Rostirolla

ANÁLISE DE DESEMPENHO E CONSUMO ENERGÉTICO DE UM CLUSTER BASEADO EM COMPUTADORES DE PLACA ÚNICA

Este trabalho foi julgado adequado para obtenção do título de bacharel em Engenharia da Computação e aprovado em sua forma final pelo Orientado e pela Banca Examinadora.

Orientador: _____

Prof. Pedro Antônio Madeira de Campos Velho, UNIVATES

Pós-Doutor pela UFRGS – Porto Alegre, Brasil

Banca Examinadora:

Prof. Pedro Antônio Madeira de Campos Velho, UNIVATES

Pós-Doutor pela UFRGS – Porto Alegre, Brasil

Prof. Maglan Cristiano Diemer, UNIVATES

Mestre pela UNISINOS – São Leopoldo, Brasil

Prof. Ronaldo Husemann, UNIVATES

Doutor pela UFRGS – Porto Alegre, Brasil

Coordenador do Curso de Engenharia da Computação: _____

Prof. Marcelo de Gomensoro Malheiros

Lajeado, junho de 2014

AGRADECIMENTOS

Agradeço à minha família, por acreditar e investir em mim. Mãe, seu cuidado e dedicação foi que deram, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

A minha irmã pela amizade, companheirismo e pelas palavras de força no final desta etapa.

Ao meu orientador pela paciência, amizade, e inspiração não apenas no desenvolvimento deste trabalho, mas na busca pelo aprofundamento da minha formação científica e profissional.

Aos professores pela competência no compartilhamento de seus conhecimentos.

Aos colegas dos Laboratórios de Informática e de Automação pelos ensinamentos que contribuíram para a realização deste trabalho.

RESUMO

A corrida dos supercomputadores para obtenção de maior desempenho tem desconsiderado o alto consumo energético destas máquinas. O consumo dos computadores de alto desempenho (HPC) em funcionamento atualmente ultrapassa a marca de 200 bilhões de quilowatts hora (kWh), tornando-se o recurso mais caro em um HPC. Para atingir o Exascale é necessário o uso de tecnologias diferentes, focadas na economia de energia, uma vez que o relatório do DARPA recomenda que o consumo destes novos sistemas não ultrapasse 20 MW. Este trabalho tem como objetivo analisar o desempenho e consumo energético das placas de desenvolvimento BeagleBone Black, que utilizam processadores ARM, focados no baixo consumo energético, uma das apostas da computação de alto desempenho do futuro. Para realização dos testes de desempenho e medição do consumo, foi desenvolvido um cluster homogêneo utilizando 10 placas BeagleBone Black, com sistema operacional Linux, comunicando-se através de troca de mensagens com MPI e uma placa de medição de consumo. Os resultados obtidos apontam um desempenho de 27,33 Mflops/Watt, e uma economia de energia até 85,91% quando os nós ociosos são desligados.

Palavras-chave: ARM, BeagleBone Black, Cluster, Consumo de Energia.

ABSTRACT

The race of the supercomputers to achieve higher performance has disregarded the high power consumption. The consumption of high-performance computers (HPC) currently in operation exceeds 200 billion kilowatts hour (kWh), making it the most expensive resource in HPC. To achieve Exascale it's necessary to use different technologies focused in low power consumption as an alert the DARPA report recommends that the consumption of Exascale systems does not exceed 20 MW. This job aims at analyzing the performance and the power consumption of ARM based single board computers. Precisely we use BeagleBone Black that features A8 ARM processors. Focused in low power consumption, this processor architecture can be one of the bets of the high performance computing of the future. To perform power consumption and performance analysis we built a homogeneous clusters featuring 10 BeagleBone Black boards and a custom made power consumption measurement board. The results indicate a performance of 27,33 Mflops/Watt and a power saving of up to 85.91% when the idle nodes are turned off.

Keywods: ARM, BeagleBone Black, Cluster, Power Consumption.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS

ACM	-	Association for Computing Machinery
ARM	-	Advanced RISC Machine
BLAS	-	Basic Linear Algebra Subprograms
BSC	-	Barcelona Supercomputing Center
CI	-	Circuito Integrado
CISC	-	Complex Instruction Set
COTS	-	Commercial Off-The-Shelf
CSMA/CD	-	Carrier Sense Multiple Access/Colision Detect
CUDA	-	Compute Unified Device Architecture
DC	-	Direct Current
DNA	-	Deoxyribonucleic Acid
DSP	-	Digital Signal Processing
DVI	-	Digital Video Interface
eMMC	-	Embedded MultiMediaCard
ESS	-	Earth and Space Sciences
GPU	-	Graphics Processing Unit
HA	-	High Availability
HDMI	-	High-Definition Multimedia Interface
HPC	-	High Performance Computing
HPL	-	High Performance LINPACK
HTTP	-	Hypertext Transfer Protocol
IEEE	-	Institute of Electrical and Electronics Engineers
IP	-	Internet Protocol
LAN	-	Local Area Network
LED	-	Light-Emitting Diode

LHC	-	Large Hadron Collider
MIMD	-	Multiple Instruction Multiple Data
MISD	-	Multiple Instruction Single Data
MMC	-	MultiMediaCard
MPI	-	Massage-Passing Interface
MPP	-	Massive Parallel Processing
NAS	-	NASA Advanced Supercomputing
NASA	-	National Security Agency
NPB	-	NAS Parallel Benchmarks
OV	-	Organizações Virtuais
PUCRS	-	Pontifica Universidade Católica do Rio Grande do Sul
PVM	-	Parallel Virtual Machine
QoS	-	Quality of Services
RAM	-	Random Access Memory
RISC	-	Reduced Instruction Set
SCSI	-	Small Computer System Interface
SD	-	Secure Digital
SDRAM	-	Synchronous Dynamic Random Access Memory
SIMD	-	Single Instruction Multiple Data
SISD	-	Single Instruction Single Data
SO	-	Sistema Operacional
SoC	-	System on Chip
UFPel	-	Universidade Federal de Pelotas
UFRGS	-	Universidade Federal do Rio Grande do Sul
UFSM	-	Universidade Federal de Santa Maria
USB	-	Universal Serial Board
VGA	-	Virtual Graphic Adapter
VPN	-	Virtual Private Network
VSIPL	-	Vector Signal and Image Processing Library
WAN	-	Wide Area Network
WLCG	-	Worldwide LHC Computing Grid
WWW	-	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos surgiu uma forte tendência computacional que possibilitou a realização de sofisticadas simulações de fenômenos complexos, e a exploração de grandes quantidades de dados. O Grande Colisor de Hádrons (LHC, do inglês *Large Hadron Collider*), por exemplo, captura 25 Petabytes de dados todos os anos, tornando impossível o processamento e análise manual destes dados (CORDEIRO et. al., 2013).

Simulações computacionais proveem um método flexível, de baixo custo, e alto retorno em diversas áreas. Como exemplo, pode ser citada a utilização deste tipo de simulação na agricultura, um ramo que abrange desde a análise de sensores, e processamento de imagens de satélite, até análises microscópicas do solo e clima, sendo estes fatores decisivos para maximizar a colheita (MISRA et. al., 2011).

A análise desta enorme quantidade de dados e simulações utilizando modelos complexos demanda cada vez mais processamento, fazendo com que a busca de alto desempenho computacional seja uma das peças chaves para o avanço deste tipo de aplicação. Durante a busca por alto desempenho computacional, destacam-se alguns fatos: a obtenção do Gigascale (10^9) em 1985 com o Cray 2, o Terascale (10^{12}) em 1997 com o Intel ASCI Red, o Petascale (10^{15}) em 2008 com o IBM Roadrunner. O que leva a busca do Exascale, 10^{18} operações (multiplicação ou soma) de dupla precisão (64-bit) por segundo (KOGGE et. al., 2008).

A corrida dos supercomputadores até a obtenção do Petascale focou apenas em uma métrica, o desempenho máximo (SCOGLAND; SUBRAMANIAM; FENG, 2012). Esta busca desenfreada por desempenho, sem levar em consideração o consumo energético fez com que algumas das máquinas do TOP500 consumissem

Megawatts (MW) de energia para atingir Petaflops de desempenho. Se esta máquina fosse ampliada até o Exascale, seriam necessários Gigawatts de energia para mantê-la funcionando, algo equivalente ao produzido por uma usina nuclear de médio porte (PADOIN et. al., 2012).

Os computadores de alto desempenho (HPC do inglês *High Performance Computing*) em funcionamento em todo o mundo apresentam um consumo altíssimo, entre 200 e 300 bilhões de quilowatts hora (kWh) (GÖDDEKE et. al., 2013). O custo da energia está se tornando cada vez mais alto, chegando a ser o recurso mais caro em um HPC ou Datacenter (KOGGE et. al., 2008).

Estes fatores tornaram a eficiência energética uma das principais preocupações no desenvolvimento de qualquer sistema computacional de alto desempenho, e unanimemente reconhecida como a principal limitadora do alcance ao Exascale, conforme relatório do DARPA (*Defense Advanced Research Projects Agency*), o qual estabelece 20 MW como sendo um consumo de energia razoável para obtenção deste nível de desempenho (KOGGE et. al., 2008).

Análises de computadores de alto desempenho mostram que 40% a 60% do consumo energético é atribuído ao processador e a memória, 10% a interfaces de conexão e sistema de armazenamento, e o restante é consumido pela infraestrutura – fontes de alimentação, iluminação, e principalmente refrigeração. Tendo em vista esta distribuição, o maior retorno pode ser esperado melhorando a eficiência dos nós computacionais (processador e memória), o que irá também atenuar o gasto com refrigeração (GÖDDEKE et. al., 2013).

Começam a surgir então projetos como o Mont-Blanc Zero, desenvolvido no Barcelona *Supercomputing Center* (BSC) que utiliza processadores de baixo consumo ARM Cortex-A9, utilizados normalmente em dispositivos móveis. O projeto visa a construção de um HPC capaz de atingir um desempenho de 200 Petaflops consumindo apenas 10 MW de energia (RAJOVICY; PUZOVIC; RAMIREZY, 2012).

O objetivo deste trabalho é a construção de um cluster com *Single-board Computers* (computadores de placa única) de baixo consumo energético que utilizam processadores ARM. As placas utilizadas são do modelo BeagleBone Black, com processadores Cortex-A8. O foco principal foi a análise de desempenho e o consumo

energético sob diversas cargas de trabalho, através do uso do benchmark HPL (*High Performance* LINPACK), utilizando por grandes listas que avaliam o desempenho de clusters, e do Ondes3D, uma aplicação de simulação de predição de terremotos. Além da avaliação da forma como o consumo é medido nos clusters de grande porte atualmente.

Inicialmente, está apresentado um referencial teórico, contendo os conceitos fundamentais de sistemas distribuídos, clusters e grids, além das aplicações e hardwares a utilizados nos testes, o estado da arte da computação de alto desempenho (TOP500), e da computação de alto desempenho com baixo consumo (Green500), e as formas de medição de consumo energético de clusters utilizadas atualmente.

O restante do trabalho está organizado da seguinte forma. O capítulo 3 apresenta trabalhos relacionados, como o projeto de computadores de alto desempenho com a utilização de processadores ARM (Projeto Mont-Blanc), e análise do consumo energético em clusters com Single-board Computer. O capítulo 4 apresenta o cluster desenvolvido, bem com os materiais utilizados para sua alimentação e comunicação, além da forma de medição de consumo e os experimentos realizados. O capítulo 5 apresenta a análise e discussão dos resultados obtidos. E por fim, no capítulo 6, as conclusões que puderam ser obtidas com a realização deste trabalho.

2 REFERENCIAL TEÓRICO

2.1 Sistemas Distribuídos

Entre 1945 e 1985 os computadores eram grandes, ocupando cerca de 64 m^2, e caros, custando cerca de 500 mil dólares. Até mesmo computadores de uso pessoal custavam dezenas de milhares de dólares. Esta situação começou a mudar em meados de 1980 com o início de dois avanços tecnológicos (TANENBAUM; STEEN, 2002).

O primeiro avanço foi o desenvolvimento de microprocessadores mais poderosos – inicialmente de 8 bit, mas logo evoluindo para 16, 32 e 64 bit – com desempenho semelhante ao de um mainframe na época, mas custando uma fração do valor. Segundo Tanenbaum e Steen (2002), o salto foi de máquinas que custavam 100 milhões de dólares e executavam uma instrução por segundo, para máquinas que custavam 1000 dólares e eram capazes de executar 10 milhões de instruções por segundo. Caso tamanha evolução atingisse o setor automobilístico, o preço de um Rolls Royce seria de um dólar, e o mesmo teria autonomia de um bilhão de milhas por galão.

O segundo avanço elencado pelo autor é a invenção de redes de computadores de alta velocidade. Redes de área local (LAN, do inglês *Local Area Network*) permitindo centenas de máquinas trocarem pequenas informações em alguns microssegundos, e as redes de longa distância atualmente trafegando até taxas de Gigabits por segundo tornaram possível a computação em um grande número de

computadores, interconectados em curta ou longa distância. Este tipo de computadores interconectados são chamados de sistemas distribuídos.

2.1.1 Caracterização dos Sistemas Distribuídos

Segundo Coulouris (2001), um sistema distribuído é aquele cujos componentes fazem parte de uma rede de computadores e estão comunicando e coordenando suas ações apenas através de troca de mensagens. Ou ainda citando Leslie Lamport (1978):

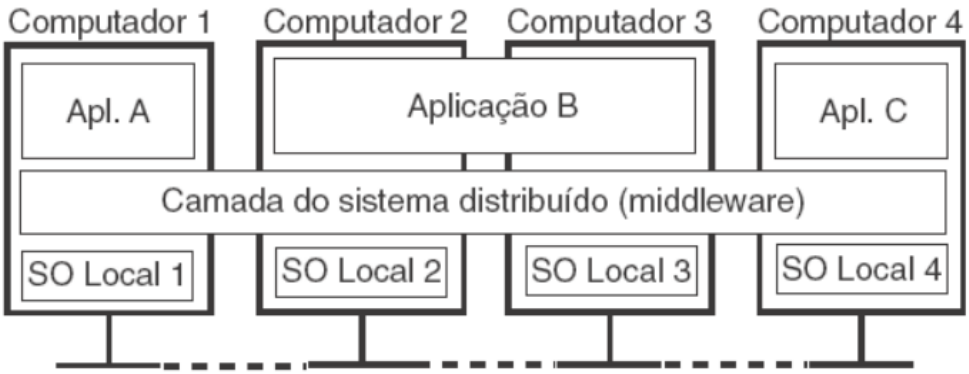
> "Você sabe que existe um sistema distribuído quando a falha de um computador que você nunca ouviu falar impede que você faça qualquer trabalho."

Uma característica importante de sistemas distribuídos é a diferença entre as diversas máquinas que o compõe e a maneira como que se comunicam. Outra característica é a interação entre usuário e a aplicação com o sistema distribuído, a qual deve ser consistente e uniforme, não dependendo de onde ela for executada (TANENBAUM; STEEN, 2002).

Ainda segundo Tanenbaum e Steen (2002), este tipo de sistema também deve ser de fácil expansão ou escalabilidade. Um sistema distribuído deve estar constantemente disponível, apesar de algumas partes ficarem temporariamente fora de funcionamento. Usuários e aplicações não devem perceber que partes do sistema estão sendo substituídas, ou que novas partes foram adicionadas.

Para suportar computadores heterogêneos, e oferecer uma visão única do sistema, sistemas distribuídos são organizados em camadas, conforme a Figura 1, onde a camada de middleware se estende por todas as máquinas.

Figura 1 - Sistema distribuído organizado como middleware.



Fonte: TANENBAUM e STEEN (2002).

Como exemplo de sistema distribuído, é possível citar a internet. Ela é composta por um vasto número de computadores dos mais variados tipos, interconectados. Estas máquinas interagem através de troca de mensagens, permitindo que usuários acessem serviços como a *World Wide Web* (WWW), e-mail e servidores de arquivos localizados em qualquer lugar da rede (TANENBAUM; STEEN, 2002).

2.1.2 Rede e Interconexões

Sistemas distribuídos utilizam tanto redes de área local como redes de longa distância (WAN, do inglês *Wide Area Network*), para a comunicação entre os diversos sistemas. Desempenho, mobilidade, escalabilidade e qualidade dos serviços (QoS, do inglês *Quality of Services*) são características importantes e que definem o comportamento de um sistema distribuído (COULOURIS; DOLLIMORE; KINDBERG, 2001).

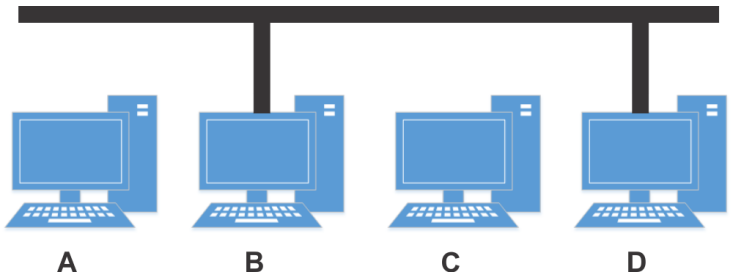
As redes de computadores têm evoluído a cada dia, a fim de prover suporte para este tipo de aplicação, no entanto, a disponibilidade de mais recursos na rede de comunicação não tem evoluído na mesma proporção do desempenho das aplicações (DANTAS, 2005). Nesta seção serão apresentados dispositivos de interconexão convencionais, e de alto desempenho, além dos principais dispositivos utilizados na interconexão utilizados em sistemas distribuídos.

2.1.2.1 Dispositivos de interconexão convencionais

No início dos anos 90, com a evolução da microeletrônica as redes Ethernet baseadas em cabos coaxiais foram substituídas por hubs e switches na interligação de computadores.

Hubs são dispositivos concentradores, quando um dispositivo inicia a sua comunicação, os demais devem aguardar o término dessa conexão, conforme Figura 2. Estes dispositivos operam em faixas entre 10 e 100 Mbps, utilizando cabos de par trançado (DANTAS, 2005).

Figura 2 – Forma como a comunicação é estabelecida utilizando HUB

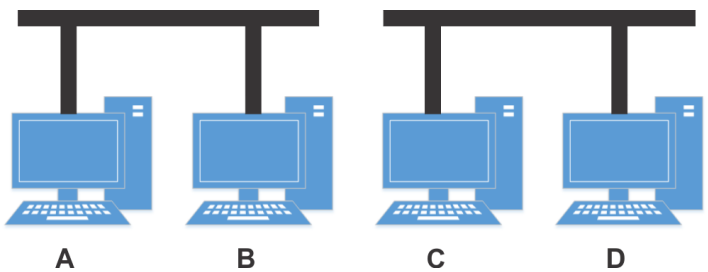


Fonte: Do autor, com base em Dantas (2005).

Este tipo de abordagem faz com que todos os nós consigam se enxergar, sendo necessária a utilização de um protocolo de detecção de colisão, o CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*), que arbitra um tempo de espera aleatório caso dois nós tentem enviar uma mensagem ao mesmo tempo (COULOURIS; DOLLIMORE; KINDBERG, 2001).

Os switches por sua vez são comutadores, que permitem conexões paralelas, bastando que os computadores solicitem a interligação em pares, conforme Figura 3 (DANTAS, 2005).

Figura 3 - Forma como a comunicação é estabelecida utilizando Switch



Fonte: Do autor, com base em Dantas (2005).

Na figura é possível observar as máquinas A e B estabelecendo comunicação ao mesmo tempo que as máquinas C e D. A largura de banda para um switch atualmente está na faixa dos Gbps. Segundo Sterling (2002), com o uso de switches devidamente dimensionados, e um core switch é possível escalar facilmente um cluster acima de 1000 nós.

2.1.2.2 Dispositivos de interconexão de alto desempenho

Em projetos de clusters e grids, a interconexão dos diversos dispositivos é um dos aspectos críticos para obtenção de bom desempenho. Switches com tecnologias Fast Ethernet ou Gigabit Ethernet não representam mais uma solução apropriada para conexões em larga escala.

Para suprir esta necessidade surgem duas grandes classes de dispositivos de interconexão de alto desempenho. O primeiro grupo propõe uma solução baseada na programação de troca de mensagens entre processadores no nível da interface de rede, como a Myrinet, Gigabyte System Network, GigaNet e InfiniBand. O outro grupo propõe a abstração de uma memória virtual única entre todos os dispositivos interligados, como o Quadrics Network (QSNET) e o Dolphin SCI (Dantas, 2005).

A interface da conexão de alto desempenho mais utilizada entre os 30 primeiros computadores do Green500 (apresentado na seção 2.6) até 2011 foi a InfiniBand, como pode ser observado na Tabela 1.

Tabela 1 – Evolução das interconexões das 30 primeiras máquinas do Green500 por ano.

	2007	**2008**	**2009**	**2010**	**2011**
Proprietária/ Customizada	26	12	18	16	5
InfiniBand	4	18	12	13	25
Gig-E	0	0	0	1	0

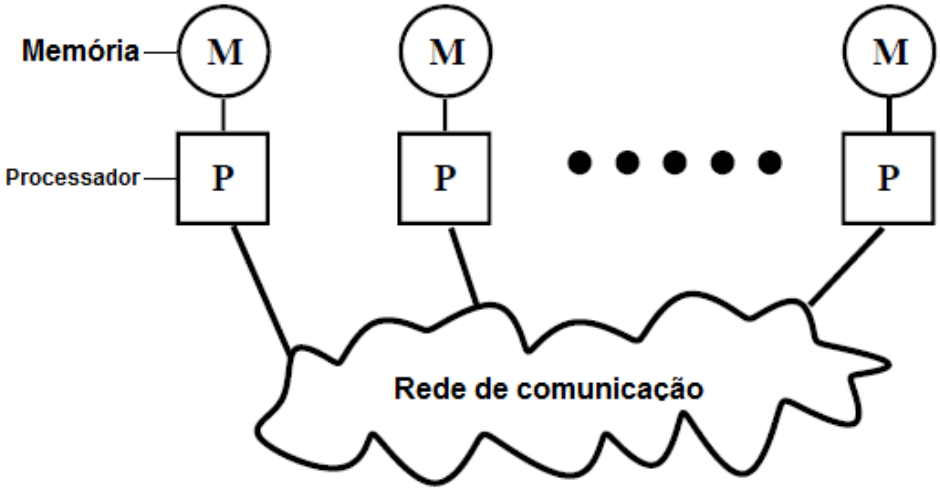
Fonte: SCOGLAND; SUBRAMANIAM; FENG. (2012).

2.1.3 Comunicação entre processos

O paradigma normalmente utilizado por programadores é o sequencial. O programador tem uma visão da máquina, como um processador único, com acesso a certa quantidade de memória, e escrevem programas para serem executados neste ambiente. Este tipo de implementação pode a princípio ser portada para qualquer arquitetura sequencial, no entanto, em ambientes paralelos onde um programa pode ser executado em diversos nós localizados em locais distintos, muitas vezes não compartilhando o mesmo espaço de memória, este paradigma não é adequado (MACDONALD et. al., 1987).

O paradigma de troca de mensagens foi desenvolvido a partir do paradigma sequencial, considerando diversas instancias deste unidas. O programador deve desenvolver um programa imaginando diversos processadores, cada um com seu próprio espaço de memória, trocando mensagens entre si utilizando uma rede de comunicação, chamando sub-rotinas, conforme ilustrado na Figura 4.

Figura 4 - Paradigma de troca de mensagens.



Fonte: Baseado em Macdonald et. al. (1987).

Este paradigma vem crescendo, principalmente por ser suportado por um grande número de plataformas, além de possibilitar a execução destes programas em sistemas multiprocessados com memória distribuída ou compartilhada, e em todos os

tipos de clusters e sistemas monoprocessados (MACDONALD et. al., 1987). Para realizar esta troca de mensagens existem bibliotecas especializadas no tratamento da comunicação, e sincronização dos processos concorrentes, sendo MPI (*Massage-Passing Interface*) e PVM (*Parallel Virtual Machine*) os mais utilizados (GNACIO; FERREIRA FILHO, 2002).

O PVM é um sistema de mensagem de domínio público, desenvolvido inicialmente para rodar em clusters, no entanto possui diversas implementações para rodar em máquinas de processamento paralelo massivo (MPP, do inglês *Massive Parallel Processing*). Neste trabalho optou-se pela utilização e apresentação apenas do MPI, por oferecer mais recurso, opções e parâmetros por chamada que o PVM, além de estar se tornando padrão nas implementações paralelas (GNACIO; FERREIRA FILHO, 2002).

2.1.3.1 Massage Passing Interface (MPI)

O MPI é um padrão de interface para troca de mensagens em máquinas paralelas com memória distribuída, resultado do esforço de aproximadamente 60 pessoas provenientes de 40 organizações dos Estados Unido e da Europa.

O projeto do MPI teve início em abril de 1992 em um seminário realizado pelo Center for Research on Parallel Computing, formalizado como projeto em novembro de 1993 originando a versão oficial do MPI, e tendo sua primeira versão publicada em 1994 o MPI-1. Em 1997 foi lançado o MPI-2, e em setembro de 2012 a versão utilizada atualmente, o MPI-3.0 que contou com a remoção de bindings obsoletos para o C++ e suporte a FORTRAN 2008. Todas as alterações no MPI são escritas e ratificadas pelo MPI Forum (MACDONALD et. al., 1987).

No padrão MPI a aplicação é constituída por um ou mais processos que se comunicam, através do paradigma de troca de mensagens, apresentado anteriormente. A principal finalidade do MPI é prover rotinas para sincronização e controle de processos além da distribuição e consolidação dos dados entre computadores interconectados. Por ser uma especificação o MPI possui diversas implementações no mercado, algumas pagas como o Intel MPI, HP MPI e o MATLab

MPI, e outras *open source* (código aberto) onde se destacam o Open MPI e o MPICH (MACDONALD et. al., 1987).

Apesar de ser um padrão complexo, boa parte dos problemas que utilizam MPI podem ser resolvidos utilizando apenas as seis funções que serão descritas na Tabela 2. Maiores informações sobre as funções disponíveis e exemplos de aplicação podem ser encontrados na documentação oficial fornecida pelo MPI Forum.

Tabela 2 - Funções básicas do MPI.

Nome	**Descrição**
MPI_INIT	Esta rotina deve ser chamada no início de qualquer aplicação que utilize MPI para inicializa-lo.
MPI_FINALIZE	Esta rotina limpa todos os estados do MPI. Uma vez que ela for chamada nenhuma outra rotina MPI deve ser chamada.
MPI_COMM_SIZE	Esta rotina retorna o número de processos associados a uma comunicação.
MPI_COMM_RANK	Esta rotina determina o rank do processo que está sendo executado.
MPI_SEND	Esta rotina envia uma mensagem de modo bloqueante.
MPI_RECV	Esta rotina recebe uma mensagem de modo bloqueante.

Fonte: Elaborado pelo autor, com base em Coelho (2012).

A Listagem 1 apresenta um código em C utilizando a biblioteca MPI. No programa apresentado, cada nó irá informar o seu número, e o total de nós na comunicação. Após isto, o nó mestre (rank 0) irá sortear um número aleatório (linha 19) e enviará a todos os nós escravos (linha 21), que estarão esperando o recebimento (linha 27). Após receber a mensagem, os nós escravos informarão que número receberam e o programa será finalizado.

Listagem 1 - Exemplo de código em C utilizando MPI.

```
1       #include <stdio.h>
2       #include <stdlib.h>
3       #include <mpi.h>
4
5       void main(int argc, char** argv){
6         int rank, size, node, num, master;
7         master = 0;
8         /* Inicia o MPI */
9         MPI_Init (&argc, &argv);
10        /* Obtem o número de nós alocados */
11        MPI_Comm_size (MPI_COMM_WORLD, &size);
12        /* Obtem o rank do nó atual */
13        MPI_Comm_rank (MPI_COMM_WORLD, &rank);
14
15        printf( "Eu sou o nó %d de um total de %d nós\n", rank, size );
16
17        if(rank == master){
18          for( node = 0; node < size; node++){
19            num = (rand( ) % 100)+1;
20          /* Envia uma mensagem com um número aleatório para cada nó */
21            MPI_Send(&num, 1, MPI_INT, node, 0, MPI_COMM_WORLD);
22            printf("O mestre enviou o número %d para o nó %d\n", num, node);
23          }
24        }
25        else{
26          /* Recebe uma mensagem do nó mestre*/
27          MPI_Recv(&num,   1,   MPI_INT,   master,   0,   MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
28          printf("O nó %d recebeu %d do nó %d\n", rank, num, master);
29        }
30        /* Finaliza o MPI */
31        MPI_Finalize();
32      }
```

Fonte: Elaborada pelo autor.

Segundo Coelho (2012) para compilar e executar um programa escrito com o uso de MPI basta seguir os passos descritos abaixo:

a) Incluir as bibliotecas MPI no momento da compilação, ou utilizar um compilador que faça este reconhecimento de maneira automática;
b) Verificar os nós disponíveis para a execução;
c) Escolher os nós que irão executar o programa;
d) Executar o programa passando como parâmetro os nós escolhidos.

2.2 Clusters

Clusters estão se tornando cada vez mais populares em centros de pesquisa e indústrias. Esta tecnologia está sendo utilizada nas mais diversas áreas: mapeamento do DNA humano (CATALYUREK, 2002), predição de terremotos (MICHÉA, 2010), concepção de novos medicamentos (DORN, 2012) são alguns exemplos. Basicamente qualquer aplicação que não pode parar de funcionar, perder dados, ou que necessita de alto poder de processamento pode se beneficiar da utilização de clusters (DANTAS, 2005).

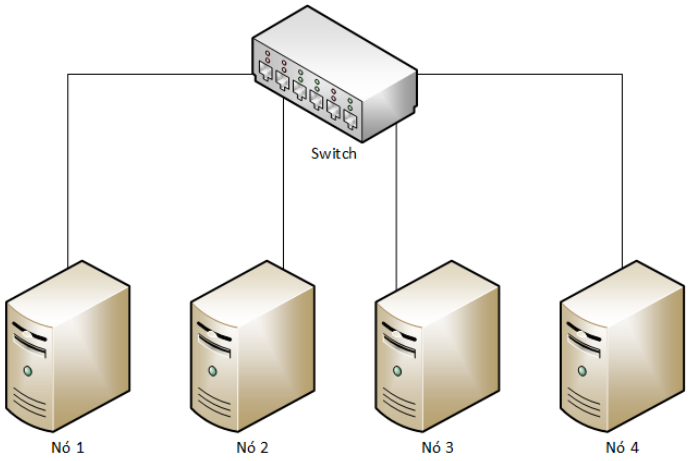
Esta popularização dos clusters deve-se a dois fatores, a grande variedade de soluções presentes no mercado, desde soluções baseadas em software livre e configurações de hardware mais simples de baixo custo, até configurações de alto desempenho com hardwares sofisticados e custo elevado, e a escalabilidade obtida com este tipo de abordagem (PEREIRA FILHO, 2004; KOPP, 2012).

Cluster é apresentado como um aglomerado de computadores que trabalham juntos para criar um sistema muito mais poderoso (VOGEL apud PEREIRA FILHO, 2004). Cada computador que compõe um cluster é chamado de nó (também conhecido como *node* ou nodo). Este agregado de diversos nós deve ser visto pelo usuário como uma máquina única (KOPP, 2012).

Para que os diversos nós sejam percebidos pelo usuário como uma única máquina, é necessário que haja comunicação entre eles. Por serem considerados fracamente agrupados – não compartilharem o mesmo barramento para comunicação

– é necessário estabelecer a comunicação de alguma outra forma, a fim de ordenar as instruções a serem executadas (PEREIRA FILHO, 2004). Na Figura 5 é apresentado um cluster composto por quatro nós interconectados através de um switch.

Figura 5 - Cluster formado por quatro computadores (nós).



Fonte: Do autor com base em Pereira Filho (2004).

2.2.1 Princípios

O conjunto de princípios básicos apresentado abaixo é definido por Pereira Filho (2004) como requisitos para que um Cluster seja considerado útil:

a) Escalabilidade: adicionar ou remover componentes do cluster (nós) deve ser possível sem haver interrupção dos serviços;
b) Comodidade: a estrutura deve ser composta por máquinas normais, conectadas por uma rede genérica, utilizando o mesmo Sistema Operacional (SO) com um software de gerenciamento em comum em todos os nós;
c) Transparência: apesar de ser constituído por um grupo de nós fracamente agrupados, o cluster deve ser apresentado ao usuário como um sistema único. Possibilitando assim a interação do usuário como se fosse um sistema comum de alta performance ou alta disponibilidade;

d) Confiabilidade: deve ser capaz de detectar possíveis falhas, e tomar atitudes para que estas falhas não comprometam o funcionamento dos serviços oferecidos;
e) Gerenciamento e Manutenção: deve dispor de um mecanismo que permita gerenciar de forma simples a complexidade de configuração e manutenção do cluster, facilitando assim o trabalho de administração.

2.2.2 Abstrações

Para que haja a compreensão do comportamento esperado e da função de cada elemento do cluster Pereira Filho (2004) apresenta algumas abstrações dos principais elementos de um cluster.

2.2.2.1 Nó

É a unidade básica do cluster, responsável pelo processamento dos dados. Um nó deve se comunicar com os demais com troca de mensagens sobre uma conexão de rede. Cada nó deve possuir quatro componentes tidos como principais, CPU, memória, repositório de armazenamento e interconexão.

2.2.2.2 Recurso

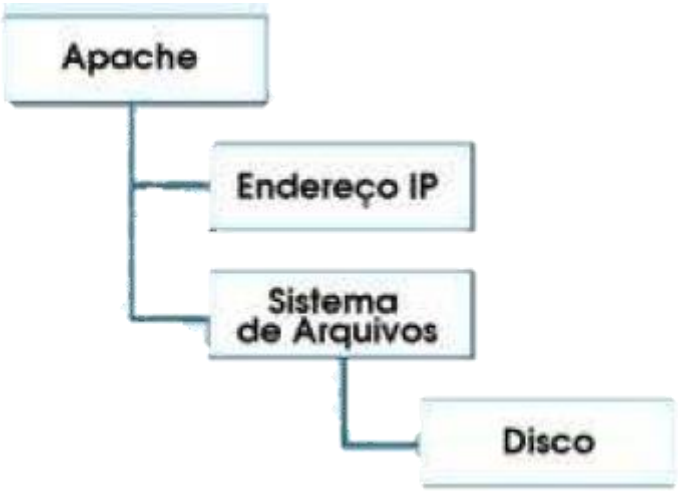
Recurso é uma funcionalidade oferecida por um nó. Pode ser físico, como impressoras ou scanners, ou lógico, como um endereço de IP. Recursos podem migrar de um nó para o outro a fim de manter a disponibilidade dos serviços oferecidos.

O gerenciador do cluster deve ser capaz de iniciar, parar, e monitorar o estado de um recurso, a fim de tomar atitudes em caso de falhas.

2.2.2.3 Dependência de Recursos

Recursos dependentes são aqueles que necessitam de outros recursos para funcionar. Em uma rede, por exemplo, um nome de rede depende de um endereço de IP. Um recurso pode especificar quais são os recursos dos quais dependerá, e ao ser migrado, um grafo de dependências pode ser utilizado para especificar qual a ordem em que os serviços dependentes devem ser inicializados. O grafo de dependências de um servidor HTTP (*Hypertext Transfer Protocol*) é demonstrado na Figura 6.

Figura 6 - Grafo de dependências de um servidor HTTP.



Fonte: Pereira Filho (2004).

2.2.2.4 Grupo de Recursos

Grupo de recursos nada mais é do que o agrupamento de diversos recursos. O objetivo deste grupo é garantir que todos os serviços necessários sejam migrados juntamente, ou seja, a política de migração deve migrar todos os recursos de um mesmo grupo.

2.2.3 Tipos de Cluster

Segundo Alves 2002 os clusters podem ser divididos em duas categorias básicas, Alta Disponibilidade (HA – High Availability) e Alto Desempenho (HPC).

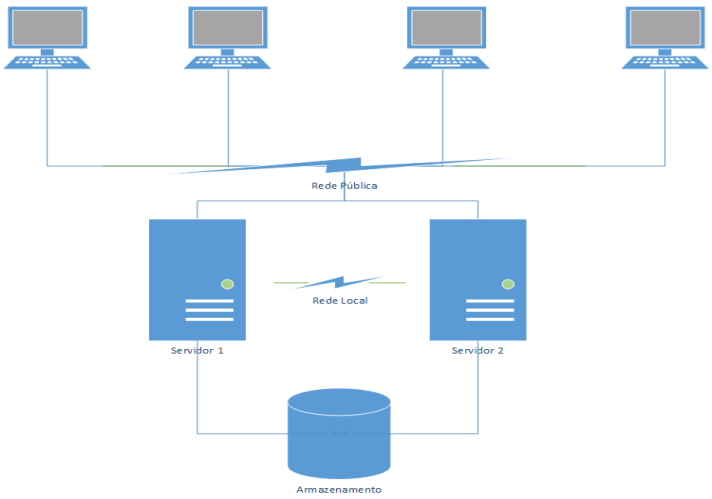
2.2.3.1 Cluster de Alta Disponibilidade

Seja em ambientes comerciais, empresariais ou domésticos, nenhum usuário quer que seu equipamento pare de funcionar. A alta disponibilidade veio para resolver esta situação, garantido o funcionamento dos serviços de maneira ininterrupta (PEREIRA FILHO, 2004).

Neste tipo de cluster, são utilizadas máquinas redundantes, com replicação de serviços, a fim de evitar a paralização total destes serviços. Mesmo que haja alguma perda de desempenho, o objetivo principal será atingido.

Alves aponta que de maneira geral, servidores de boa qualidade apresentam disponibilidade de 99,5%, já os clusters computacionais apresentam 99,99%. Esse aumento percentual é vital em ambientes empresariais, já que em muitos casos, se o sistema para, a empresa também para. A Figura 7 apresenta um modelo simples de cluster de alta disponibilidade.

Figura 7 - Cluster Simples de Alta Disponibilidade.



Fonte: Do autor, com base em Alves (2002).

Este tipo de cluster, e o que será apresentado a seguir (HPC) possuem diversas características em comum, como paradas programadas e serviços contínuos.

2.2.3.2 Cluster de Alto Desempenho

Alto desempenho atualmente faz uso de clusters de computadores, considerados soluções alternativas, sendo utilizados em empresas de pequeno e médio porte, e universidades para fins de estudo, a um custo baixo se comparado ao de supercomputadores (ALVES, 2002).

Clusters deste tipo têm como objetivo fornecer resultados satisfatórios, em tempo hábil, mesmo que para isso seja necessário processar milhares de Gigaflops. Sua arquitetura consiste em computadores interconectados trabalhando juntos para a resolução de um problema computacional.

2.2.4 Homogeneidade X Heterogeneidade

Clusters podem também ser classificados como homogêneos ou heterogêneos. Esta classificação diz respeito aos nós que o compõe, e a interface de comunicação entre os grupos de máquinas (SCHEPKE et. al., 2005).

O principal impacto causado por clusters heterogêneos, ou seja, que possuem nós com configurações diferentes, é com relação ao balanceamento de carga (ALVES, 2002). Este balanceamento é a distribuição imparcial de carga entre os nós, se não for executado de maneira correta neste tipo de cluster, um sistema pode ficar sobrecarregado, enquanto outro está ocioso ou com uma pequena carga de trabalho, tornando necessária a redistribuição das tarefas.

Clusters homogêneos são considerados ideais, no entanto, mais cedo ou mais tarde a tendência, e um dos grandes atrativos de um cluster, é a adição de novas máquinas. Devido ao grande avanço tecnológico sofrido pela informática, dificilmente a máquina nova que será adicionada ao cluster possuirá a mesma configuração. Logo, este tipo de estrutura é a mais comumente encontrada em departamentos de universidades e companhias (ALVES, 2002).

2.2.5 Categorias dos Clusters

Segundo Dantas (2005) a maioria dos fabricantes adotam arquiteturas baseadas em clusters de componentes “comerciais de prateleira” (COTS do inglês Commercial Off-The-Shelf), o que pode ser verificado ao analisar a lista TOP500, onde a maioria dos listados utilizam clusters, tendo como diferencial suas interconexões especiais, com taxas de transferência na ordem de Gbytes/segundo.

A classificação deste tipo de ambiente (COTS) ocorre da seguinte forma:

a) Categoria I: clusters nesta categoria utilizam componentes especializados e pacotes de softwares customizados para configuração, contudo, ainda empregam alguns componentes COTS, no entanto esta utilização é feita de maneira industrial. Como exemplo pode ser citado o multicomputador XT3 da empresa Cray que utiliza processador AMD Opteron de 2.4 GHz (componente COTS), canais de entrada e saída que chegam atingem até 100 Gbytes/segundo, e capacidade de memória entre 4.3 Terabytes com seis gabinetes, ou 239 Terabytes com trezentos e vinte gabinetes.

b) Categoria II: clusters de segunda categoria são formados totalmente por hardware COTS, e pacotes de softwares abertos. São construídos de maneira artesanal (self-made) e utilizam interfaces padrão tanto para comunicação quando para entrada e saída, como Ethernet e SCSI (*Small Computer System Interface*) respectivamente.

2.2.6 Gerenciamento de Clusters

O gerenciamento de um cluster é uma tarefa complexa, que envolve desde a instalação do sistema operacional, até a definição de ferramentas para configuração, manutenção e monitoramento dos nós.

Em geral, as máquinas presentes em um cluster utilizam um mesmo sistema operacional, e devido ao seu grande número, são utilizadas ferramentas para realizar a uma instalação padronizada em todos os nós. Ferramentas como o Alice (Ames Lab ISU *Cluster Environmnet*) e o OSCAR (*Open Source Cluster Application Resources*) fornecem um ambiente de instalação automática do sistema operacional, com o mínimo de interação humana, além da configuração e ativação de diversos serviços (SCHEPKE et. al., 2005).

Além das configurações iniciais, outra tarefa essencial é a monitoração dos recursos utilizados. Desta forma, é possível determinar se há ou não necessidade de aumentar o poder computacional do cluster, agregando novos nós, além da possiblidade de detectar congestionamentos na rede, ou falta de memória. Estas informações são essenciais na tomada de decisões. Existem atualmente diversas ferramentas específicas para monitorar clusters, como o Ganglia e o Parmon, que oferecem interfaces amigáveis, apresentando os dados de forma gráfica através de interfaces *Web* além de serem escaláveis, possibilitando a adição e monitoramento de novos nodos (SCHEPKE et. al., 2005).

2.2.7 Cluster Beowulf

Em janeiro de 1992 iniciou o programa NASA HPCC com o objetivo de alcançar e avançar o estado do processamento massivamente paralelo (MPP), além de aplicá-lo em problemas computacionais. Cientistas do projeto ESS (*Earth and Space Sciences*) precisavam manipular, visualizar e transformar um grande número de dados obtidos através de simulações de fenômenos físicos, evolução das galáxias, entre outros. Surge então, o projeto Beowulf *Parallel Workstation* (ALVES, 2002).

A NASA necessitava de um equipamento que processasse na ordem de Gigaflops, no entanto, computadores que tinham esse desempenho custavam em torno de um milhão de dólares na época, um investimento muito alto para ser utilizado somente por um grupo de pesquisadores. Tomas Sterling e Donald J. Becker decidiram então interligar 16 computadores, cada um com um microprocessador Intel 486 de 100Mhz, utilizando o sistema operacional Linux, e rede Ethernet (10 Mbps),

atingindo cerca de 70 Megaflops a um custo de 40 mil dólares, 10% do valor de uma máquina de desempenho equivalente na época (ALVES, 2002).

Atualmente diversos setores demandam elevado poder de processamento, e armazenamento de dados, no entanto, supercomputadores convencionais ainda possuem um custo muito elevado, e são pouco escaláveis. Como alternativa, surgem os clusters Beowulf, provendo computação de alto desempenho (HPC) a um baixo custo (STERLING, 2002). Um clusters Beowulf é um agregado de nós computacionais de categoria II, conforme apresentado anteriormente, homogêneos ou heterogêneos interligados por uma rede convencional, possuindo um nó encarregado da gerência deste ambiente (*frontend*) e os demais nós sendo escravos, executando tarefas delegadas por este *frontend* (BESERRA et. al., 2011).

2.3 Grid

Até meados da década de 1990, a computação de alto desempenho era realizada em supercomputadores com arquitetura especial, posteriormente, na segunda metade desta década surgiram os Clusters, que nada mais são que um agrupamento de dezenas ou centenas de computadores convencionais, ambos apresentados anteriormente. Na virada do século surgiu a ideia de interconectar diversos clusters através da internet a fim de formar uma grade computacional (ou *grid*). O objetivo era agregar milhares de computadores localizados em diversos locais e disponibiliza-los a cientistas de instituições distintas a fim de permitir o processamento de grande quantidade de dados (CORDEIRO et. al., 2013). Nesta seção será apresentada a definição, a arquitetura, a topologia e casos de uso de uma grade computacional, e a GridRS.

Segundo Dantas (2005), a grid pode ser entendida como uma plataforma heterogênea de computadores geograficamente dispersos, acessados por usuários através de uma interface única. Grids diferem de outros ambientes distribuídos convencionais principalmente pela grande quantidade de serviços e recursos compartilhados. Analogamente a uma rede elétrica, onde o usuário utiliza a energia sem saber de onde vem, em uma grid computacional o uso dos equipamentos deve

ocorrer de forma transparente, sem quem o usuário precise saber onde os recursos estão alocados.

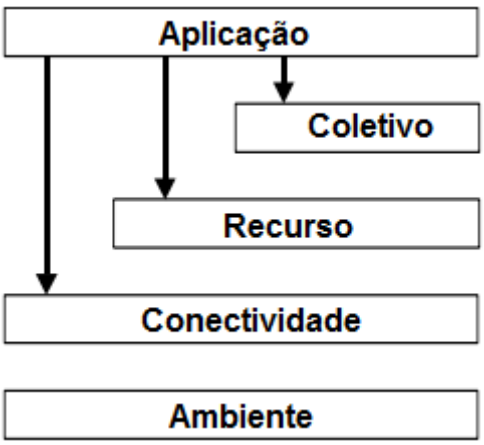
Projetos como o SETI@home, onde usuários comuns doam seu poder de processamento ocioso para dados de radiotelescópios, a fim de buscar vida inteligente fora da Terra podem ser confundidos com uma grid, no entanto este tipo de projeto é classificado como “Oportunista” (DANTAS, 2005).

2.3.1 Arquiteturas e Topologias

Comunidades de pesquisadores como a *Open Grid Forum* vem propondo modelos de arquitetura com o objetivo de padronizar e permitir a interoperabilidade entre as diferentes organizações participantes de uma grid, também conhecidas como organizações virtuais (OV).

Dantas (2005) apresenta dois modelos de arquitetura das grids, que vem ganhando aceitação na comunidade internacional, baseados em Foster (1999) e Tony (2003). Estes modelos serão apresentados na Figura 8 e na Figura 9.

Figura 8 - Modelo de Foster.



Fonte: Dantas (2005).

Este modelo apresenta cinco níveis, e será descrito da base para o topo nos itens a seguir.

a) **Ambiente:** neste nível são implementadas mecanismos como o de negociação de solicitações para obtenção de informações sobre a infraestrutura, o estado e as possibilidades dos recursos, e mecanismos de gerência de recursos;
b) **Conectividade:** neste nível são definidos protocolos como o de transporte, roteamento, serviço de nomes e autenticação, ou seja, os protocolos básicos para autenticação e comunicação na grid;
c) **Recursos:** este nível apresenta individualmente protocolos de autenticação e comunicação do nível inferior. Também é responsável pela chamada das funções do níveis de Ambiente;
d) **Coletivo:** o nível anterior trata os recursos individualmente, já este nível atua nas interações entre coleções de recursos. Baseia-se nos níveis de aplicação e implementa serviços como de diretório e autorização comunitária.
e) **Aplicação:** nível que compreende as aplicações do usuário, que invocam serviços disponíveis nos níveis inferiores (DANTAS, 2005).

Figura 9 - Modelo de Tony.

Fonte: Dantas (2005).

Uma outra proposta de arquitetura de grid é apresentada em quatro camadas, descritas a seguir do topo para a base.

a) **Aplicações e Serviço**: milhares de aplicações diferentes compõe este nível, desde ferramentas de desenvolvimento a aplicações

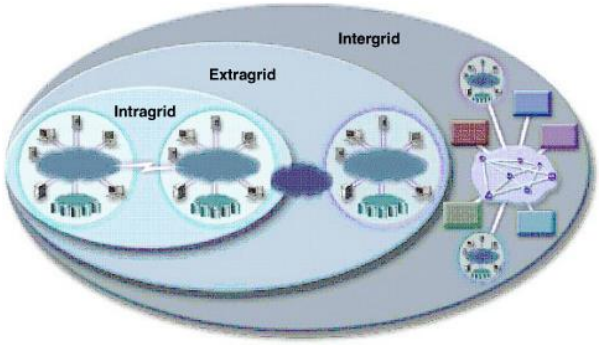
desenvolvidas nas mais diversas áreas. Além destas aplicações, também fazem parte deste nível serviços que proveem funções de gerenciamento, e compartilhamento de recursos entre usuários;

b) **Middleware**: este nível fornece protocolos que permitem elementos como servidores, armazenamento e rede participem de um ambiente grid unificado. Diversos protocolos e funções devem existir nesta camada para dar suporte a heterogeneidade de uma grid;
c) **Recursos**: este nível é constituído pelos diversos recursos que fazem parte da grid, como servidores primários e dispositivos de armazenamento;
d) **Redes**: no nível de rede estão disponíveis todos os recursos de conectividade da grid, como switches, roteadores e a infraestrutura necessária (DANTAS, 2005).

Com relação aos aspectos topológicos, podem ser consideradas três diferentes configurações, descritas abaixo, a Figura 10 ilustra a abrangência de cada topologia.

a) **Intragrid:** topologia que considera a utilização de recursos e serviços presentes dentro de uma mesma organização;
b) **Extragrid:** topologia que considera recursos alocados entre mais de uma instituição com interação, podendo possuir inúmeras organizações virtuais dentro de uma mesma instituição;
c) **Intergrid:** topologia composta por muitas instituições parceiras, compostas por centenas até milhares de organizações virtuais (JACOB; FUKUI; TRIVEDI, 2005).

Figura 10 - Topologias das grids.



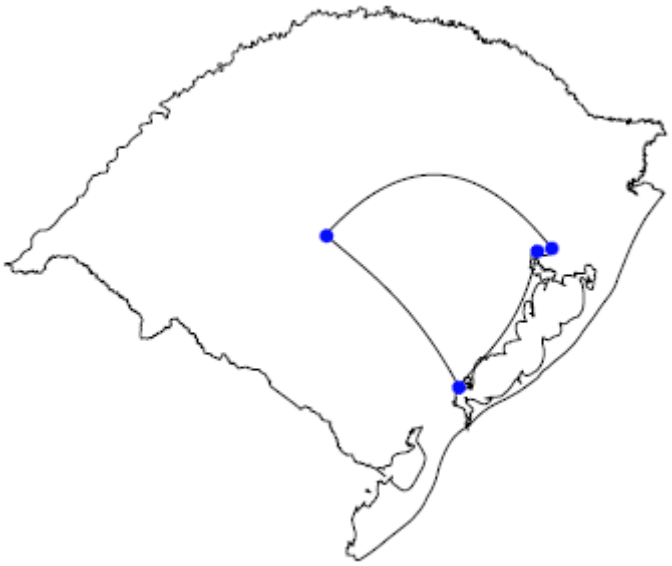
Fonte: JACOB; FUKUI; TRIVEDI (2005).

2.3.2 GridRS

Com o intuito de contornar o problema da baixa eficiência energética dos clusters convencionais, surgem projetos como a GridRS, uma grid de alto desempenho focada em aspectos de eficiência energética, seguindo o conceito de *green grid*.

O projeto conta com a colaboração de quatro universidades do Rio Grande do Sul, PUCRS, UFRGS, UFPel e UFSM (Figura 11). A estrutura é heterogênea, e conta com máquinas com processadores de alto desempenho, como o Xeon, e máquinas com processadores baixo desempenho e baixo consumo, como o ARM.

Figura 11 - Distribuição da GridRS.



Fonte: GRIDRS.

A grid conta atualmente com 34 nós, comunicando-se através de uma rede privada virtual (VPN) através da internet, armazenamento de dados realizados localmente em cada um dos sítios, recursos gerenciados através do software OAR, e monitoração dos recursos de forma visual através do software Ganglia. Cada instituição disponibiliza um portal de acesso, com uma base unificada de usuários.

A grid está disponível para todos os estudantes e pesquisadores que tiverem interesse em desenvolver algum projeto de cunho científico, sejam estas pessoas estudantes das instituições participantes ou não.

2.4 Softwares Utilizados

Nesta seção serão descritos os softwares utilizados na análise do desempenho e consumo do cluster desenvolvido. Primeiramente será apresentado um benchmark, logo após uma aplicação científica paralela, e por fim o software utilizado na medição de consumo.

2.4.1 LINPACK

O benchmark escolhido para avaliar o desempenho do cluster desenvolvido é o *High Performance* LINPACK (HPL), por ser amplamente utilizado, é possível encontrar comparativos para quase todos os sistemas computacionais, além de ser utilizado na análise de desempenho dos supercomputadores do TOP500.

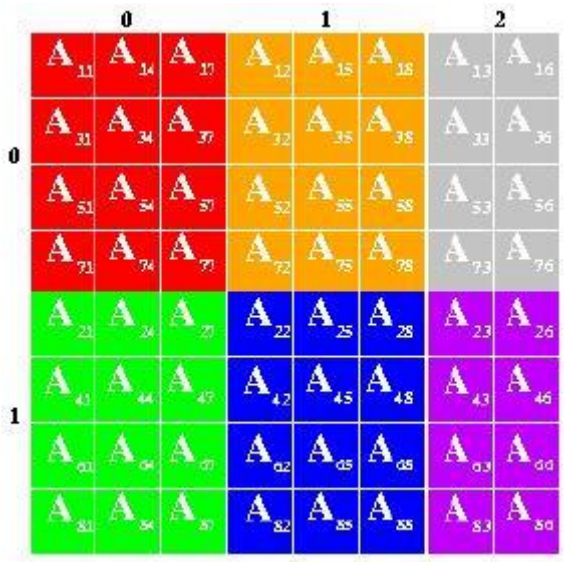
O LINPACK benchmark começou a ser desenvolvido por Jack Dongarra em meados de 1980, quando ele começou a coletar dados baseando-se na velocidade em que um sistema de matrizes lineares de 100 x 100 era resolvido, utilizando rotinas desenvolvidas em Fortran. Problemas desta magnitude, ainda podem ser utilizados para medir desempenho em um único nó de um cluster Beowulf, por exemplo, no entanto não é possível medir o desempenho de um cluster utilizando esta abordagem (STERLING, 2002).

Para realizar as medições neste tipo de sistema Dongarra implementou o *High Performance* LINPACK (HPL). Este benchmark resolve sistemas de equações lineares do tipo A.x=b, onde A é a matriz dos coeficientes, gerada estocasticamente pelo programa, possuindo tamanho N x N . x, e b possui dimensão N. Para resolver este sistema, o programa realiza a fatoração da matriz A utilizando um método

conhecido como decomposição LU, com pivoteamento parcial de linha. Esta fatoração resulta em A=L.U, onde L é a matriz triangular inferior (*lower*) e U a matriz triangular superior (*upper*). Para encontrar a solução, o algoritmo aplica sucessivamente os passos de solução triangular (BESERRA et. al., 2011).

Os dados são distribuídos em uma matriz bidimensional P x Q, de acordo com o esquema de blocos cíclicos, a fim de assegurar um bom balanceamento de carga e escalabilidade do algoritmo. A matriz de coeficientes de tamanho N x N + 1 é particionada em Nb x Nb blocos, que são ciclicamente processados nas duas dimensões da matriz P x Q. A Figura 12 demonstra a distribuição da matriz de coeficientes na grade, utilizando 6 processos. Cada elemento corresponde a uma submatriz de dimensão Nb x Nb (BESERRA et. al., 2011).

Figura 12 - Distribuição dos blocos lógicos da matriz na grade de processos.



Fonte: HPL.

Esta implementação é altamente portável, e possui como dependências o MPI, e uma implementação de um subprograma básico de álgebra linear (BLAS, do inglês *Basic Linear Algebra Subprograms*) ou uma biblioteca de processamento de sinais vetoriais e imagens (VSIPL, do inglês *Vector Signal and Image Processing Library*), os quais estão disponíveis para uma larga variedade de sistemas, incluindo os processadores ARM (*Advanced RISC Machine*). O HPL permite a configuração dos

valores P e Q mencionados anteriormente, o valor N da matriz, o valor Nb, entre outros parâmetros, através de um arquivo chamado HPL.dat.

Por mais de dez anos Jack Dongarra, Erich Strohmaier e Horst Simon vem suportando uma lista chamada TOP500, onde estão presentes as 500 máquinas mais poderosas do mundo, apresentada na seção 2.5.

A principal desvantagem da utilização de sistemas benchmark como o LINPACK, tanto em sua versão monoprocessada, quanto na paralela, é a tendência a superestimar o desempenho do sistema. Isto deve-se ao fato do cálculo ser realizado em matrizes densas, as quais possuem uma alocação de dados muito favorável, ao contrário das aplicações executadas normalmente. Não raramente, o LINPACK atinge valores 30% ou mais de desempenho de pico teórica, enquanto aplicações científicas comuns atingem por volta de 10% (STERLING, 2002).

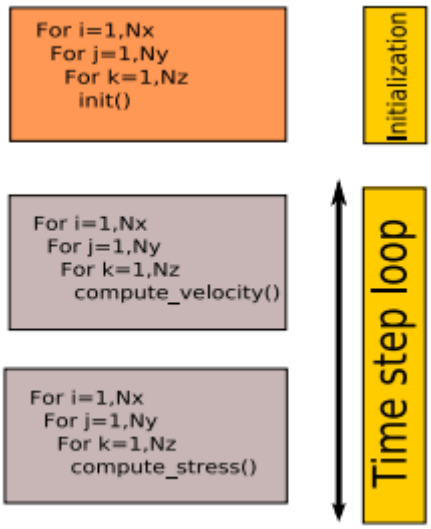
2.4.2 Ondes3D

O Ondes3D, uma aplicação paralela desenvolvida pela empresa francesa BRGM (do francês *Bureau de Recherches Géologiques et Minières*) para simular a propagação de ondas sísmicas em um meio geológico tridimensional utilizando o método das diferenças finitas (DUCELLIER A. et. al., 2009). A aplicação foi escolhida pelo seu embasamento científico, além de apresentar tanto rotinas de cálculo quanto de manipulação de memória alocada de forma dinâmica, representando mais fielmente o comportamento de uma aplicação real, ao contrário do LINPACK.

Os parâmetros da aplicação são definidos em quatro arquivos distintos, e um arquivo principal de entrada. Neste arquivo principal chamado de option.h estão definidos o comportamento do material (elástico ou inelástico) além do método de cálculo que será utilizado. Os outros quatro arquivos de configuração definem principalmente o número de pontos – inicial e final – em cada direção (x, y e z), o número de passos que a simulação deve ter, um arquivo com o modelo geológico, um arquivo com histórico dos tremores, contendo o valor do impacto, a inclinação, a profundidade e o instante de tempo em que ocorreu (BRGM, 2011).

A Figura 13 ilustra os três principais passos da aplicação, a alocação dos dados, a inicialização dos dados, e os cálculos de propagação das ondas. O primeiro laço de repetição trata do processo de alocação e inicialização dos dados. Posteriormente são apresentados os dois grandes laços de cálculo que determinam a propagação do sismo. Todos os laços de repetição possuem o mesmo padrão de acesso dos dados (RIBEIRO, 2008).

Figura 13 - Ilustração do funcionamento do Ondes3D



Fonte: RIBEIRO (2008).

2.4.3 LabVIEW

LabVIEW (do inglês *Laboratory Virtual Instruments Engineering Workbench*) é uma linguagem de programação visual, desenvolvida pela National Instruments em 1986, e é amplamente utilizada por engenheiros e cientistas, tanto em projetos acadêmicos quanto em industrias (FARACO, 2007).

Um programa desenvolvido com LabVIEW é chamado de Virtual Instrument (VI), que parte do conceito de instrumentação virtual, ou seja, a combinação de hardware e software para criar uma solução de instrumentação definida pelo usuário. A National Instruments também fornece plug-ins de hardware utilizados para aquisição de dados (DAQ, do inglês *Data Aquisition*), facilitando a criação de sistemas

customizados para diversos segmentos, como automação industrial e medição de consumo (NATIONAL INSTRUMENTS, 2007).

Um VI é composto de três partes principais, um diagrama de blocos, um painel frontal e os conectores. O diagrama de blocos consiste no fluxo de execução do programa, as operações realizadas entre as entradas e saídas, também chamado de executável do programa. O painel frontal é a parte interativa de um VI, onde serão apresentados os gráficos, botões de controle e possíveis valores de entrada definidos pelo usuário. Por fim, os conectores são os responsáveis por representar graficamente onde as entradas e saídas estarão conectadas dentro do diagrama de blocos. (FARACO, 2007).

2.5 TOP500

Em 1993, pela primeira vez, uma lista dos 500 mais poderosos supercomputadores foi disponibilizada na web. O TOP500 fornece estatísticas detalhadas, contendo o número de núcleos, desempenho (flops/s), consumo energético, localização geográfica, fabricante, entre outras informações (DONGARRA et. al., 1997).

Estas informações são compiladas em uma lista, com a ajuda de especialistas em computação de alto desempenho, cientistas da computação e fabricantes. A lista é publicada duas vezes por ano junto com *a International Supercomputing Conference*, e a ACM/IEEE *Supercomputing Conference*, nos meses de junho e novembro, respectivamente. A classificação dos supercomputadores é de acordo com o seu desempenho no benchmark HPL, citado na seção anterior.

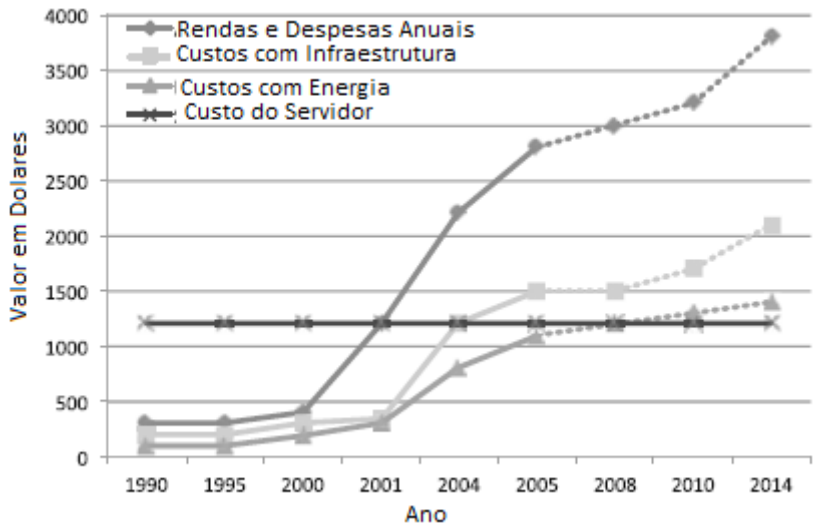
Através do histórico de atualizações e substituições dos sistemas presentes na lista, Strohmaier e Meuer (2004) determinaram a dinamicidade do mercado de supercomputadores. Cerca de 160 sistemas são substituídos ou atualizados a cada meio ano na lista do TOP500, ou seja, a máquina que atualmente está na posição de número 100, estará fora da lista nos próximos 2 ou 3 anos.

2.6 Green500

A corrida das grandes fabricantes de supercomputadores, focando apenas no desempenho trouxe uma era de máquinas "*power hungry*" consumindo enormes quantidades de energia, necessitando estruturas de alto custo para refrigeração e operação (SCOGLAND; SUBRAMANIAM; FENG, 2012).

O custo de um servidor rack em 2001 excedia o custo anual com infraestrutura e energia deste mesmo servidor. Em 2004 o custo anual da estrutura igualou o custo do servidor, e em 2008 o custo anual da energia necessária para manter o servidor funcionando igualou o custo do mesmo (Figura 14). Buscando trazer o problema deste alto consumo à tona e incentivar a competição por supercomputadores mais eficientes em 2007 Wu-chun Feng propôs formalmente a utilização de uma nova métrica de classificação destas máquinas, o número de operações de ponto flutuante por Watt (flops/Watt), dando origem a lista Green500 (SCOGLAND; SUBRAMANIAM; FENG, 2012).

Figura 14 - Custo de infraestrutura, energia e do servidor em relação ao tempo.



Fonte: SCOGLAND; SUBRAMANIAM; FENG, (2012).

A lista apresenta informações relacionadas à arquitetura das máquinas, além da localização, desempenho e consumo energético. A última lista publicada (Junho de 2013), é liderada por dois sistemas heterogêneos baseados em aceleradores NVIDIA Kepler K20, e utilizam processadores Xeon E5-2687W. O desempenho das

máquinas é de 3.208,83 e 3.179,88 Mflops/Watt para a primeira e segunda colocadas respectivamente. A Tabela 3 apresenta a lista dos 10 primeiros colocados no Green500, além das respectivas posições no TOP500.

Tabela 3 - Top 10 Supercomputadores do Green500 e sua colocação no TOP500.

Nome	Ranking		Eficiência		
	Green500	TOP500	Rmax	Rpico	Mflops/Watt
Eurora	1	467	98511	175667	3208,827362
Aurora Tigon	2	NC	98640	167782	3179,883946
Beacon	3	397	110500	157547,52	2449,567723
SANAM	4	52	421200	1098000	2351,102428
BlueGene/Q, Power BQC	5	169	188967	209715	2299,148315
Cetus	6	167	188967	209715	2299,148315
CADMOS BG/Q	7	168	188967	209715	2299,148315
BlueGene/Q, Power BQC	8	170	188967	209715	2299,148315
Vesta	9	166	188967	209715	2299,148315
BlueGene/Q, Power BQC	10	171	188967	209715	2299,148315

Fonte: Do autor, com base em DENG et. al.. (2013).

2.7 Processadores ARM

Segundo a Forbes em novembro de 2012 os processadores ARM eram utilizados em aproximadamente 90% dos dispositivos móveis, como smartphones, tablets e netbooks (TEAM, 2012).

Por atuar neste nicho de mercado, os processadores ARM precisam apresentar alta eficiência energética, o que faz com que eles sejam uma das alternativas na busca pelo Exascale. Projetos como o Mont-Blanc Zero já utilizam processadores ARM na construção de nós computacionais, a fim de avaliar sua eficiência energética em aplicações de alto desempenho (RAJOVICY; PUZOVIC; RAMIREZ, 2012).

Neste trabalho, serão utilizados processadores da família ARMv7, mais especificamente o SoC (System on Chip) Sitara AM3358, baseado no processador Cortex-A8, por estarem presentes nas placas BeagleBone Black, utilizadas no cluster.

Primeiramente serão apresentados conceitos básicos para a compreensão do funcionamento dos processadores ARM, para então apresentar o processador Cortex-A8.

2.7.1 RISC x CISC

Ao contrário das máquinas com conjuntos de operações complexos (CISC, do inglês *Complex Instruction Set*), a arquitetura RISC (*Reduced Instruction Set*) utiliza um conjunto reduzido de instruções, que possuem um tamanho fixo, permitindo tanto a operação a ser realizada, quanto aos operandos acesso simultâneo, por estarem em posições conhecidas da memória, consumindo assim menos energia (AROCA et. al., 2012).

Em contrapartida, por serem mais simples são necessárias diversas instruções RISC para executar a mesma função de uma instrução CISC, fazendo com que programas RISC sejam mais extensos. Este tipo de abordagem apresentava problemas em máquinas com memória limitada, não sendo mais significativo atualmente (AROCA et. al., 2012).

Segundo Aroca (2012), atualmente processadores CISC converte suas instruções em microcódigos, similares a instruções RISC, fazendo com que sejam arquiteturas híbridas RISC/CISC. Este tipo de arquitetura possuí uma sobrecarga desnecessária, tornando o processamento mais complexo e "*power hungry*".

2.7.2 System on Chip

Um System on Chip (SoC) é definido pela empresa de pesquisa de mercado Dataquest como um dispositivo com mais de cem mil portas lógicas que possua pelo menos um núcleo programável e uma memória no chip.

SoC incorporam diversos componentes que estariam dispostos separadamente em placas convencionais, como processadores de sinais, codificadores, filtros ativos, protocolos e amplificadores operacionais. Chvojka Júnior (2005) elencou as principais vantagens desta abordagem como sendo:

a) Aumento da velocidade de operação do sistema, devido ao fato do fluxo de dados entre o processador e os outros componentes ocorrer no mesmo chip;
b) Redução da potência consumida, devido ao alto grau de integração e uso de tensões menores;
c) Redução de tamanho, atribuído a redução do uso de componentes adicionais;
d) Redução no uso de trilhas nas placas de circuito impresso, aumentando a confiabilidade do sistema.

Este tipo de chip está cada vez mais presente em dispositivos como smartphones, tablets e outros que necessitam de economia de energia. A larga adoção por parte do mercado de dispositivos móveis, aliada com os fatores apresentados acima, faz com que os SoC estejam atingindo desempenho e eficiência energética cada vez maiores. (RAJOVICY; PUZOVIC; RAMIREZY, 2012).

2.7.3 Os processadores ARM

ARM é um acrônimo de *Advanced RISC Machine*, que como o próprio nome sugere, são processadores que utilizam um conjunto de instruções RISC. Os processadores ARM se tornaram populares devido a sua utilização em SoC, apresentados anteriormente.

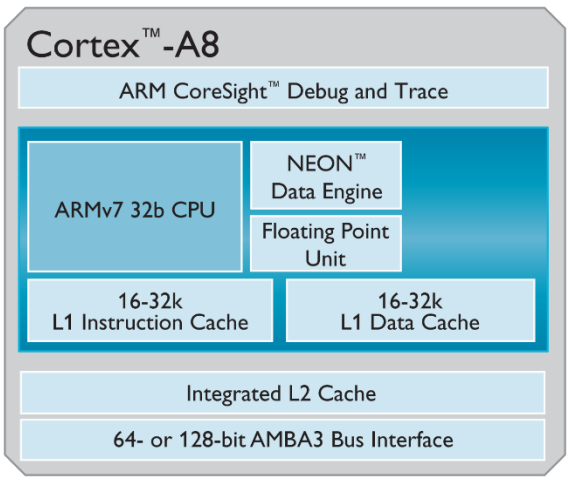
A empresa ARM não fabrica os processadores, apenas os projeta e licencia a sua fabricação para outras empresas. Atualmente existem diversos SoC no mercado de dispositivos móveis que fazem uso da propriedade intelectual dos chips ARM como o Qualcomm Scorpion/Dragonfly, Samsung Exynos, Texas Instruments OMAP, Freescale i.MX, e o Marvell Armada.

Os chips da família Cortex-A estão despertando interesse da comunidade para aplicações científicas, principalmente devido ao início do suporte a operações com ponto flutuante nativamente, suporte a instruções SIMD, multiprocessamento simétrico (a partir do Cortex-A9), e por serem os primeiros chips ARM a atingirem 1 GHz ou mais (a partir do Cortex-A8). Estes fatores combinados com a sua eficiência energética e a utilização da área do chip faz com que processadores ARM sejam uma das principais apostas para a chegada ao Exascale (GÖDDEKE et. al., 2013).

2.7.3.1 Cortex-A8

O Cortex-A8 (Figura 15) é um processador, como os demais processadores da família ARMv7-A, de 32-bit, que utiliza um conjunto de instruções RISC, possui 16 registradores e arquitetura de memória Harvard. Pode operar em frequências entre 600 Mhz e 1 Ghz, consumindo menos de 300mW de energia. O A8 ainda conta com dois pipelines, um de inteiros de 13 estágios, e outro de 10 utilizando NEON, uma extensão para SIMD de 128-bit, muito útil para aceleração multimídia e processamento de sinais. Os processadores desta família (ARMv7-A) ainda implementam a tecnologia Jezelle-RCT, que permite compilação *just-in-time* de linguagens que utilizam *bytecode*, como o Java, e a tecnologia Thumb-2 que reduz o tamanho do código enquanto mantém a equivalência de desempenho, pois permite instruções de 16-bit coexistirem com instruções de 32-bit (ROBERTS-HOFFMAN, 2009).

Figura 15 - Processador ARM Cortex-A8.



Fonte: Datasheet do processador ARM Cortex-A8.

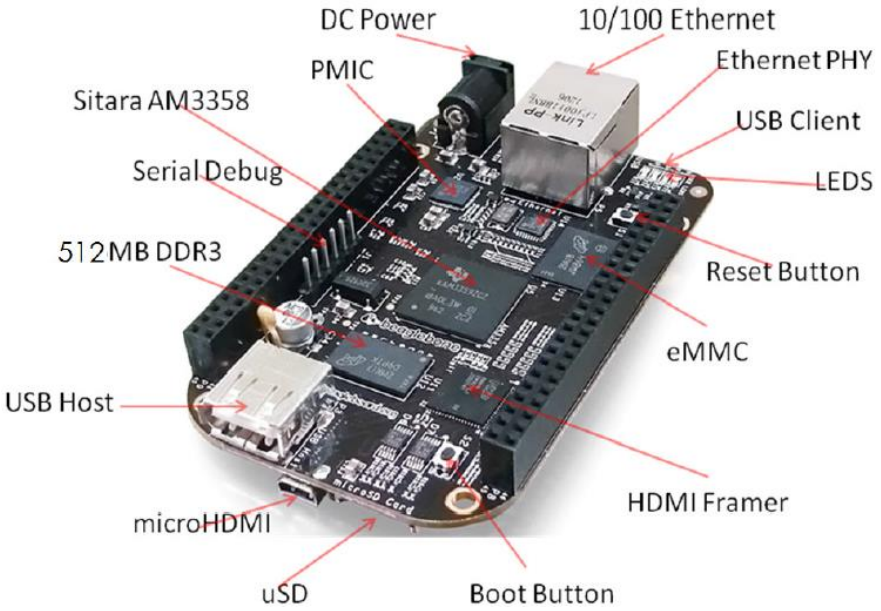
2.8 BeagleBone Black

A BeagleBone Black é um *Single-board Computer* (computador de placa única) de baixo custo com processador ARM, a placa mais atual produzida pela empresa BeagleBoard.org. Ela é designada para a comunidade *open source*, e por este motivo possui todas as especificações de hardware abertas à comunidade.

Os principais componentes da placa estão dispostos na Figura 16, e serão detalhados conforme apresentado abaixo:

a) DC *Power*: Entrada 5 Volts DC (corrente contínua) principal, utilizada para alimentação da placa;
b) PMIC (*Power Management Integrated Circuits*): Circuito integrado de gerência de energia, fornece energia para os demais componentes da placa, detalhado na seção 2.8.1.1;
c) Sitara AM3358: SoC baseado no ARM Cortex-A8, detalhado na seção 2.8.1.5;
d) 512MB DDR3: Memória RAM utilizada pelo processador, detalhada na seção 2.8.1.2;
e) uSD: Soquete para cartão micro SD (*Secure Digita*l);
f) eMMC: Memória embarcada MMC (do inglês, *MultiMediaCard*) *onboard* de 2GB, apresentada na seção 2.8.1.3;
g) LEDS: Quatro LEDs (do inglês, *Light-emitting diode*) controláveis pelo usuário;
h) Ethernet PHY: Controladora de conexão de rede, detalhada na seção 2.8.1.4;
i) 10/100 Ethernet: Conector da interface de rede.

Figura 16 - BeagleBone Black.



Fonte: beagleboard.org.

2.8.1.1 PMIC

O circuito de gerência de energia utilizado é o TPS65217C, um chip único, que consiste em um circuito de alimentação de entrada dupla linear, três conversores do tipo *step-down*, e quatro reguladores LDO (do inglês, *low-dropout*). O chip pode ser alimentado pela fonte externa de 5V DC, ou através da porta USB. Os três conversores *step-down* de 2.25MHz são responsáveis por prover a energia principal da placa, energia para o processador e para a memória (COLEY, 2013).

2.8.1.2 Memória RAM DDR3

A BeagleBone Black utiliza uma única memória de modelo MT41K256M16HA-125 de 512MB. Esta memória uma DDR3 de baixo consume, fabricado pela Micron.

A interface com o processador é através de 16 vias de dados, 16 vias de endereço e 14 vias de controle (COLEY, 2013).

2.8.1.3 Memória MMC

A eMMC utilizada na BeagleBone Black é do tipo NAND de 2 GB, modelo MTFC2GMTEA-0F_WT, fabricada pela empresa Micron, do tipo não volátil, ou seja não precisa de energia para que os dados sejam mantidos, com um controlador de 11 vias, de acordo com as especificações de sistemas MMC.

Como principais vantagens, memórias MMC não apresentam componentes móveis, ou seja são resistentes a vibrações, além de ser otimizada para a controladora da placa, utilizando o modo 8-bit ao invés de 4-bit dos cartões SD, resultando em um tempo de boot reduzido pela metade (COLEY, 2013).

2.8.1.4 Controladora Ethernet

A controladora Ethernet utilizada é uma PHY (*Physical Layer*) LAN8710A de baixo consumo, full-duplex, que opera na faixa de 100 Mbps.

Esta controladora implementa auto negociação a nível de hardware, determinando automaticamente a melhor velocidade possível, e modo duplex de operação. Além de suporte Auto-MDIX, que permite a conexão entre placas no modo cross-over (COLEY, 2013).

2.8.1.5 Sitara AM3358

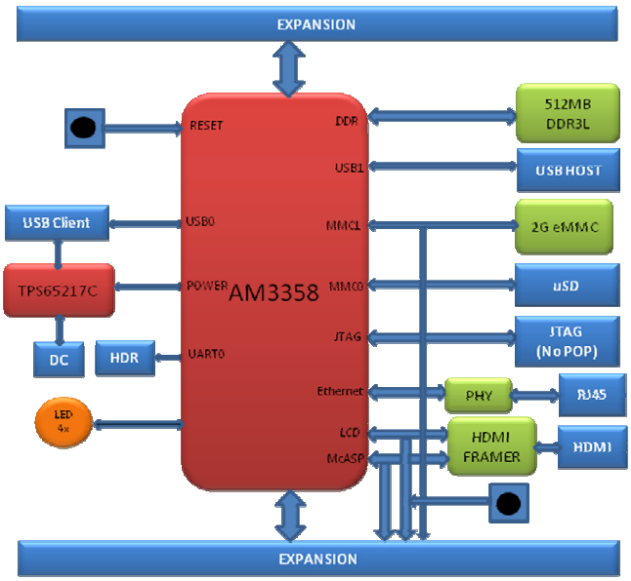
O Sitara AM3358 é um SoC baseado no processador ARM Cortex-A8, com adição de componentes de processamento gráfico, interface para periféricos e interfaces industriais. Segundo Coley (2013) o AM3358 contém os seguintes subsistemas:

a) Unidade de processamento baseado no ARM Cortex-A8;
b) Acelerador gráfico POWERVR SGX para aceleração 3D e suporte a display externo;
c) PRU-ICSS (*Programmable Real-Time Unit and Industrial Communication Subsystem*), que prove núcleos programáveis,

possibilitando a implementação de interfaces de comunicação industrial (EtherCAT, PROFINET, EtherNet/IP, PROFIBUS, e mais) com frequência de operação independente, tornando-a mais eficiente e flexível;

Na Figura 17 é possível observar a conexão entre o SoC Sitara AM3358, memória RAM, eMMC, cartão micro SD, interface HDMI e demais componentes da BeagleBone Black.

Figura 17 - Digrama de blocos do SoC Sitara AM3358.



Fonte: beagleboard.org.

2.9 Medição de Consumo Energético

A medição de consumo em sistemas computacionais pode ocorrer de diversas formas e em diversas partes deste sistema. O consumo pode ser medido entre a fonte de alimentação e a entrada de energia do sistema, conforme Figura 18, entre a placa mãe e a fonte de alimentação, assim como entre partes individuais do sistema (KRITIKAKOS, 2011).

Figura 18 - Medição de consumo entre a fonte de alimentação e o nó computacional

Fonte: GE (2007)

Como o objetivo deste trabalho é a medição do consumo de todo o sistema, serão focadas as formas de medição entre a fonte de alimentação e a entrada de energia dos nós. A forma mais comum para realizar este tipo de medição é através da utilização de medidores de consumo como o Watts up?, apresentado no manual de medição de consumo do Green500(GE, 2007). O problema na utilização deste medidor, e de outros similares é o intervalo entre as amostras, que geralmente é de 1 segundo. De acordo com o manual do fabricante do Watts up?, quando não é utilizada uma interface para transferir os dados para um computador, como apresentado na figura anterior, e o tempo de amostragem for maior que 34 minutos, o tempo entre as amostras passará para 4 segundos. Ou seja, qualquer variação de consumo que ocorre neste intervalo de tempo será desconsiderada pelo medidor (WATTS UP METERS, 2003).

Outra prática bastante comum, e também sugerida pelo manual de medição de consumo do Green500, é assumir que o consumo é similar em todos os nós do cluster, ou seja, o consumo é medido em apenas um nó, durante a realização de um benchmark por exemplo, e o resultado obtido é multiplicado pelo número de nós que compõe o cluster. A acurácia desta pratica foi analisada, e os resultados serão apresentados na seção 5.2.

3 TRABALHOS RELACIONADOS

3.1 Projeto Mont-Blanc

O projeto Mont-Blanc, desenvolvido no *Barcelona Supercomputing Center* (BSC), teve início em Outubro de 2011 com o intuito de desenvolver uma nova arquitetura de nós computacionais capaz de atingir o *Exascale* utilizando 15 a 30 vezes menos energia além de obter desempenho similar ao líder do TOP 500 até 2017, e liderar o Green500 até 2014.

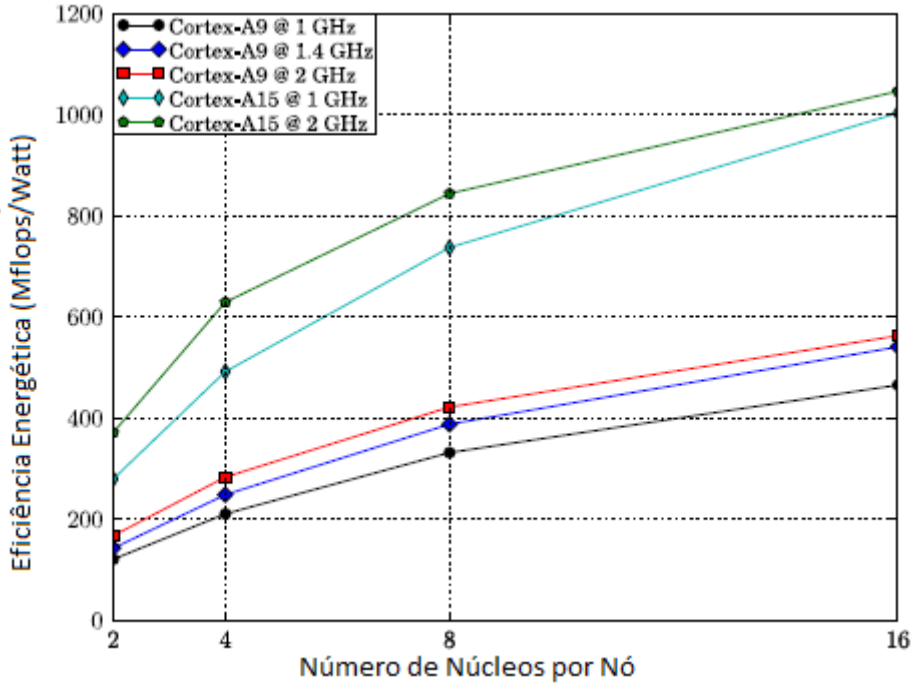
O projeto Mont-Blanc Zero, coordenado por Mateo Valero buscava a construção do primeiro supercomputador utilizando processadores ARM. O objetivo inicial era a utilização de processadores Dual-Core ARM Cortex-A9 com o intuito de obter o desempenho de 200 Petaflops consumindo 10 MW (RAJOVICY; PUZOVIC; RAMIREZ, 2012).

O primeiro cluster experimental elaborado foi chamado de Tibidabo contendo 128 nodos, utilizando chips NVIDIA Tegra2, e processadores Dual-Core ARM Cortex-A9. Mesmo com o intuito de apenas provar um conceito, este cluster inicial já obteve eficiência energética de 120 Mflops/w, competindo com clusters baseados em processadores Xeon X5660 e AMD Opteron 6128, no entanto ainda sendo 21 vezes menor que clusters baseado em processadores Intel Xeon Phi, primeiro colocado no Green500 em novembro de 2012, data em que o protótipo foi elaborado (RAJOVICY; PUZOVIC; RAMIREZ, 2012).

Para melhorar a eficiência energética do cluster, foram realizados diversos testes de desempenho, com outros modelos processadores ARM aumentando o

número de núcleos por nodo. Estes testes demonstraram melhoras significativas na utilização do ARM Cortex-A15, o qual permitiu atingir uma eficiência energética de até 1046 Mflops/W a 2GHz (Gráfico 1), uma melhora de 8,7 vezes.

Gráfico 1 - Comparativo entre processadores ARM Cortex-A



Fonte: RAJOVICY; PUZOVIC; RAMIREZ, (2012).

Em novembro de 2012 o BSC colaborou com a NVIDIA e a SECO no desenvolvimento da plataforma KAYLA, a primeira plataforma hibrida ARM + CUDA (*Compute Unified Device Architecture*). Esta nova plataforma foi colocada à prova em Junho de 2013 quando o BSC anunciou o seu novo protótipo, o primeiro cluster híbrido com processadores ARM Cortex-A9 e GPU (*Graphics processing unit*) Nvidia Tesla K20. O novo supercomputador foi nomeado Pedraforca.

O Pedraforca utiliza a interface Mellanox QDR InfiniBand para estabelecer comunicação direta entre as diversas GPU Tesla K20 além de 4 placas Nvidia Tegra 3. Os processadores ARM não são utilizados para computar os problemas, apenas para rodar o Sistema Operacional, e estabelecer a comunicação das GPU entre os nós (VERRY, 2013).

O protótipo conta com 64 nós, cada nó contendo o seguinte hardware:

a) Uma placa portadora Mini-ITX;
b) Um módulo Q7 (contendo o processador ARM e o módulo de memória - atualmente utilizando o processador Tegra 3 @ 1.3GHz e 2GB DDR2;
c) Uma placa aceleradora NVIDIA Tesla K20 (1170 Gflops);
d) Uma interface InfiniBand 40Gb/s (Mellanox ConnectX-3);
e) Um disco de estado sólido de 2.5" SATA 3 MLC de 250GB.

3.2 Cluster baseado em Raspberry PI

Em maio de 2013 Joshua Kiepert publicou um artigo sobre a construção de um cluster Beowulf utilizando o *single-board Computer* Raspberry PI (RPI) modelo B durante sua tese de doutorado.

A placa possui um SoC BCM2835, que integra GPU, DSP, SDRAM, USB, além de um processador ARM1176JZF-S pertencente à família ARM11 e 512MB de memória RAM, compartilhada com a GPU. Um cluster formado por este tipo de hardware, além de possuir um baixo custo (cerca de 45 dólares cada nó), possui uma série de interfaces externas de baixo nível fazendo com que ele seja ideal também para testar o comportamento de sistemas embarcados em larga escala (KIEPERT, 2013). O cluster desenvolvido por Kiepert possuía 32 nós comunicando-se através de MPI, utilizando o sistema operacional Raspbian "*wheezy*" (baseado no Ubuntu 11.10). Não foi realizado nenhum tipo de teste relacionando consumo e desempenho das placas.

3.3 BeagleBoard x PandaBoard

Um grupo da Universidade Federal do Rio Grande do Sul (UFRGS) analisou o desempenho e a eficiência energética de clusters baseados em processadores ARM para computação de alto desempenho. Foram utilizados dois tipos de plataforma para

realização dos testes, a PandaBoard, e a BeagleBoard (versão anterior da placa utilizada neste trabalho), ambas apresentadas na Tabela 4.

Tabela 4 - PandaBoard e BeagleBoard.

	PandaBoard	**BeagleBoard**
Processador	ARM Cortex-A9	ARM Cortex-A8
SoC	OMAP4430	OMAP3530
Clock	1 GHz	600 MHz
Memória	1 GB RAM DDR2	120 MB RAM DDR

Fonte: Elaborado pelo autor, com base em Padoin et. al.. (2012).

As plataformas foram submetidas a testes utilizando o Benchmark LINPACK apresentado na seção 2.4.1 o consumo instantâneo foi mensurado utilizando o analisador de consumo Dranetz 4300 (PADOIN et. al., 2012). Um dos principais problemas com relação as medições realizadas, é a taxa em que as leituras foram realizadas, a cada segundo, um intervalo muito grande, podendo omitir picos de consumo.

Outra contribuição deste trabalho foi a utilização de diversas flags de compilação, a fim de obter melhor desempenho. Com a utilização das flags corretas, eles demonstraram que é possível otimizar até 5 vezes a velocidade da aplicação. Os testes realizados com relação ao consumo das placas, apresentam uma eficiência de 19,749 Mflops/W utilizando a BeagleBoard, e 92,041 Mflops/W na PandaBoard, conforme a Tabela 5. Boa parte deste consumo, é atribuído ao fato destas placas serem destinadas ao desenvolvimento de aplicações embarcadas, possuindo assim uma série de portas de entrada e saída, tais como HDMI, VGA, DVI, áudio e USB, que não estariam presentes em um cluster convencional, podendo assim, ser possível obter uma eficiência energética ainda melhor (PADOIN et. al., 2012).

Tabela 5 - Eficiência energética dos equipamentos testados.

Equipamento	Beagle	Panda
Desempenho (MFlops)	23,56	755,20
Consumo Instantâneo (W)	1,193	8,205
Eficiência Energética (MFlops/W)	19,749	92,041

Fonte: PADOIN et. al.. (2012).

Em outro trabalho, este mesmo grupo de pesquisados realizou um comparativo entre a PandaBoard, e maquinas utilizando processadores XeonX7 e XeonE5, analisando o tempo e a energia utilizados na solução de um problema. A descrição detalhada dos processadores utilizados pode ser observada na Tabela 6 (PADOIN et. al., 2012).

Tabela 6 - Especificações técnicas dos processadores utilizados.

	XeonE5	XeonX7	ARMa9
Arquitetura	Nehalem	Nehalem	ARM Cortex A9
Modelo	Xeon E5530	Xeon X7550	OMAP 4430
Clock (GHz)	2,4	2,0	1,0
Processadores	2	4	1
Núcleos/Processadores	4	8	2
TDP (W)	80	130	0,25

Fonte: PADOIN et. al.. (2012).

Para realização dos testes foram utilizados 6 testes diferentes do pacote NAS *Parallel Benchmarks* (NPB). Os resultados obtidos são apresentados na Tabela 7, onde estão listados os diversos benchmarks realizados, e a energia consumida para chegar a solução em cada caso e a relação entre os processadores Xeon e o ARM.

Tabela 7 - Energia para solução e relação entre os processadores.

	Energia para a solução (Wh)			**Relação**	
Benchmark	**XeonE5**	**XeonX7**	**ARMa9**	**ARMa9/XeonE5**	**ARMa9/XeonX7**
BT	4,293	9,038	5,100	1,19	0,56
CG	1,551	2,905	2,271	1,46	0,78
EP	0,678	1,468	0,614	0,91	0,42
FT	0,875	2,070	17,800	20,35	8,60
MG	0,149	2,784	0,271	1,82	0,10
SP	3,336	13,556	3,904	1,17	0,29

Fonte: PADOIN et. al.. (2012).

Na tabela é possível observar que a energia gasta para chegar a solução de um problema utilizando processadores ARM é inferior na maioria dos casos quando comparado com o XeonX7, e equivalente ao XeonE5. O único teste onde o ARM apresenta uma grande desvantagem é o FT, devido a utilização intensa de cálculos com ponto flutuante.

4 MATERIAIS E METODOS

Esta seção está dividida em três subseções, a primeira apresenta os materiais que compõe o cluster, bem como a forma que os nós foram alimentados, os elementos de comunicação entre eles e os materiais utilizados na medição de consumo. A segunda seção descreve os métodos utilizados para a medição de consumo e a forma como os dados foram armazenados, e por fim, a terceira seção apresenta os experimentos que foram realizados e os seus objetivos.

4.1 Materiais utilizados

4.1.1 Cluster

O cluster é composto por 10 placas BeagleBone Black, apresentadas na seção 2.8, interconectadas através de suas interfaces Fast Ethernet. Primeiramente foi realizada uma avaliação do desempenho do sistema operacional, e logo após uma configuração padrão inicial de todos os nós.

4.1.1.1 Sistema Operacional

As placas BeagleBone Black, por padrão, possuem o sistema operacional Angstrom, uma distribuição Linux disponibilizada pelo fabricante. Devido ao baixo suporte fornecido pela comunidade, optou-se pela realização de testes de desempenho a fim de determinar qual a melhor distribuição a ser utilizada. Para tal,

foram utilizados dois benchmarks, uma versão do LINPACK sequencial codificada em C, apresentado na seção 2.4.1, e o Whetstone, um benchmark sintético de ponto flutuante utilizado comumente para medir o desempenho em computação científica, tendo como unidade de medida o número de instruções Whetstone por segundo (WIPS do inglês *Whetstone Instructions per Second*), também desenvolvido utilizando a linguagem C.

Os dois sistemas operacionais testados foram o Angstrom versão 2012.05, que utiliza o Kernel 3.8.13 do Linux e o Ubuntu Precise 12.04.3 LTS, utilizando a mesma versão do Kernel Linux, tratados a partir deste momento como Angstrom e Ubuntu respectivamente. Ambos os sistemas foram instalados na memória eMMC para melhor desempenho, conforme apresentado na seção 2.8.1.3.

O compilador utilizado foi o gcc, no Ubuntu a versão 4.7.3-1ubuntu1 e no Angstrom a versão 4.7-2013.02-01. Os parâmetros utilizados para compilar ambos benchmarks foram iguais, sem a utilização de nenhuma *flag* de otimização para que a comparação dos resultados fosse realizada sob as mesmas condições. Para maior precisão dos testes, foram executadas 10 iterações sob cada benchmark, utilizando como parâmetro no LINPACK uma matriz de 500x500 e no Whetstone 10000 iterações. A média destes resultados é apresentada na Tabela 8.

Tabela 8 - Resultado dos benchmarks.

	LINPACK (Kflops)	**Whetstone (WIPS)**
Angstrom	11731,75	133,5
Ubuntu	30576,5	291,65

Fonte: Elaborado pelo autor.

Como pode ser observado, o sistema operacional Ubuntu obteve desempenho 2,6 vezes maior no benchmark LINPACK e 2,18 vezes maior no Whetstone, tornando-se o sistema escolhido.

4.1.1.2 Configuração dos nós

Com o sistema operacional já instalado, cada placa recebeu um endereço de IP fixado em um servidor DHCP, e um nome – beagle01 a beagle10 – a fim de facilitar a sua identificação. Para suprir as dependências dos softwares utilizados para os testes, e permitir a comunicação entre os diversos nós, foi realizada a instalação da biblioteca MPI versão 1.6 (estável) apresentado na seção 2.1.3.1, e uma biblioteca BLAS, neste caso, o ATLAS (*Automatically Tuned Linear Algebra Software*) disponível no endereço math-atlas.sourceforge.net para resolução dos sistemas lineares do HPL. Também foi necessária a troca de chaves do SSH entre os nós, para que a comunicação fosse estabelecida sem a necessidade de senha.

Com o intuito de facilitar a gerência do cluster também foi instalado o software TakTuk, uma ferramenta utilizada na distribuição de pacotes e execução de comandos remotamente para um grande número de nós, utilizada em grids de grande porte como a Grid5000. No nó chamado de beagle01, utilizado como *frontend* de acesso aos demais nós, também foi realizada a configuração de um compartilhamento samba para disponibilizar eventuais pacotes ou arquivos de configuração com os outros nós. Para facilitar o transporte dos nós todos eles foram fixados a uma estrutura única. A forma final do cluster pode ser visualizada na Figura 19.

Figura 19 – Placas que compõe o cluster fixadas na estrutura

Fonte: Elaborado pelo autor.

4.1.2 Alimentação

A alimentação tanto da placa desenvolvida para medição de consumo quanto do cluster foi realizada com fontes de alimentação digital simétrica de modelo FA-3050. Segundo a INSTRUTHERM, cada uma das fontes possui duas saídas reguláveis independentes de 0 a 30V com corrente máxima de 5A em cada saída e precisão de ± 1,5%, além de uma saída fixa de 5V com corrente máxima de 3A. Todas as saídas possuem proteção contra curto-circuito, sobrecarga e inversão de polaridade.

Devido a utilização de amplificadores operacionais OP07, no filtro que será apresentado a seguir, foram necessárias tensões de -5 V e +5 V em sua alimentação, portanto, uma das fontes foi utilizada em configuração simétrica. A segunda fonte, utilizada na alimentação de todos os nós do cluster foi configurada no modo de saídas independentes, e por ser necessária uma corrente superior a 3A na alimentação dos 10 nós, utilizou-se uma das saídas reguláveis, que permite cargas de até 5A.

4.1.3 Rede

Conforme apresentado na seção 2.8 cada uma das placas possui um conector de rede ethernet de 10/100 Mbps (Fast Ethernet), portanto para interconectar as diversas placas e possibilitar a troca de mensagens entre elas foi utilizado um switch Fast Ethernet V2H124 de 24 portas da marca Enterasys, devido a sua superioridade em relação aos HUBs, conforme apresentado na seção 2.1.2.1. O switch foi dedicado exclusivamente para comunicação entre as placas BeagleBone Black, e a conexão foi realizada com cabos de par trançado Cat5e e conectores RJ45 seguindo o padrão EIA/TIA 568B de 1,5m de comprimento.

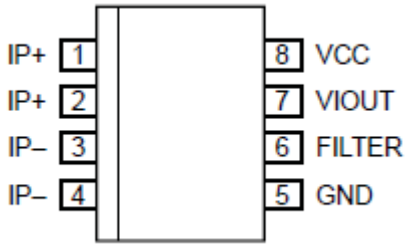
4.1.4 Medição de Consumo

Como a medição do consumo de forma precisa é parte importante deste trabalho, optou-se por desenvolver uma ferramenta de medição capaz de obter uma amostragem mais precisa e frequente. A solução encontrada foi a utilização de um sensor de efeito hall de modelo ACS712 desenvolvido pela Allegro MicroSystems, utilizado conforme apresentado na seção 4.1.4.1 e uma placa de aquisição de dados da National Instruments de modelo PCIe-6341 apresentada na seção 4.1.4.2.

4.1.4.1 Sensor e Filtragem do Sinal

O ACS712 é um sensor linear de efeito hall, seu funcionamento consiste na aplicação de uma corrente entre os terminais IP+ (pinos 1 e 2) e IP- (pinos 3 e 4) representados na Figura 20. A corrente, ao passar por estes terminais de cobre gera um campo eletromagnético, o qual é convertido para um valor de tensão proporcional. A resistência interna entre estes terminais é de 1,2 mΩ, proporcionando baixa perda de potência, além de possuir isolamento de 2,1kVRMS entre pinos 1 a 4 e os pinos 5 a 8, dispensando a utilização de componentes para isolação elétrica (ALLEGRO MICROSYSTEMS ACS712, 2012).

Figura 20 - Sensor de Efeito Hall ACS712

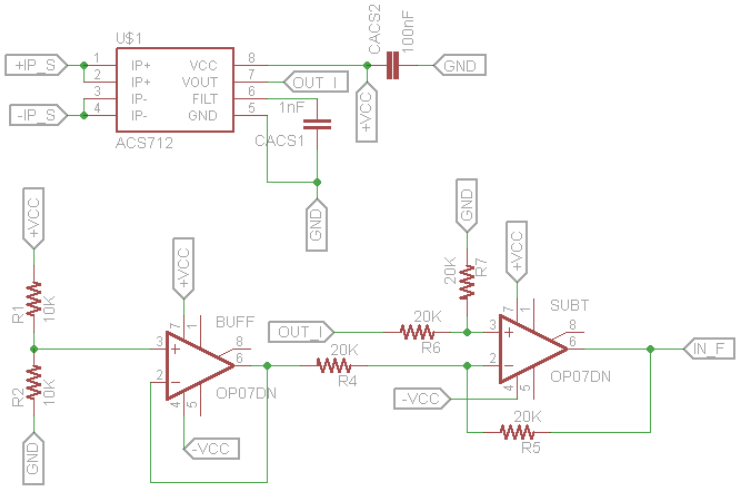


Fonte: Allegro MicroSystems (2012).

O circuito integrado (CI) utilizado foi o ACS712ELECTR-30A-T, que possibilita a medição de corrente contínua ou alternada, na faixa de -30 A a +30 A, gerando um valor de tensão no pino 7 (VIOUT) de 66 mV/A. Por funcionar também com corrente alternada, o valor inicial de tensão informado em sua saída é correspondente a metade da tensão de alimentação, ou seja, como o CI está sendo alimentado com uma tensão

de 5 V, o valor inicial de saída seria de 2,5 V. Como o objetivo é medir apenas corrente contínua, optou-se pela utilização de um amplificador operacional OP07 para subtrair metade da tensão de entrada fazendo com que o valor inicial seja 0 V, conforme o esquemático da Figura 21.

Figura 21 - Esquemático de ligação do sensor de Efeito Hall



Fonte: Elaborada pelo autor

Como a alimentação de todo o sistema foi através de fontes chaveadas, que poderiam gerar ruído nas leituras devido ao uso de interruptores eletrônicos (transistores operando na região de corte e saturação) cuja frequência de chaveamento ocorre em faixas superiores a 20 kHz (acima da audível), optou-se pelo projeto e montagem de filtro passa baixa (MEHL, 2012). Foi desenvolvido portanto, um filtro passa baixa Butterworth em topologia Sallen Key, de 8ª ordem, com 4 estágios e frequência de corte de 800 Hz. A atenuação das frequências indesejadas pode ser observada

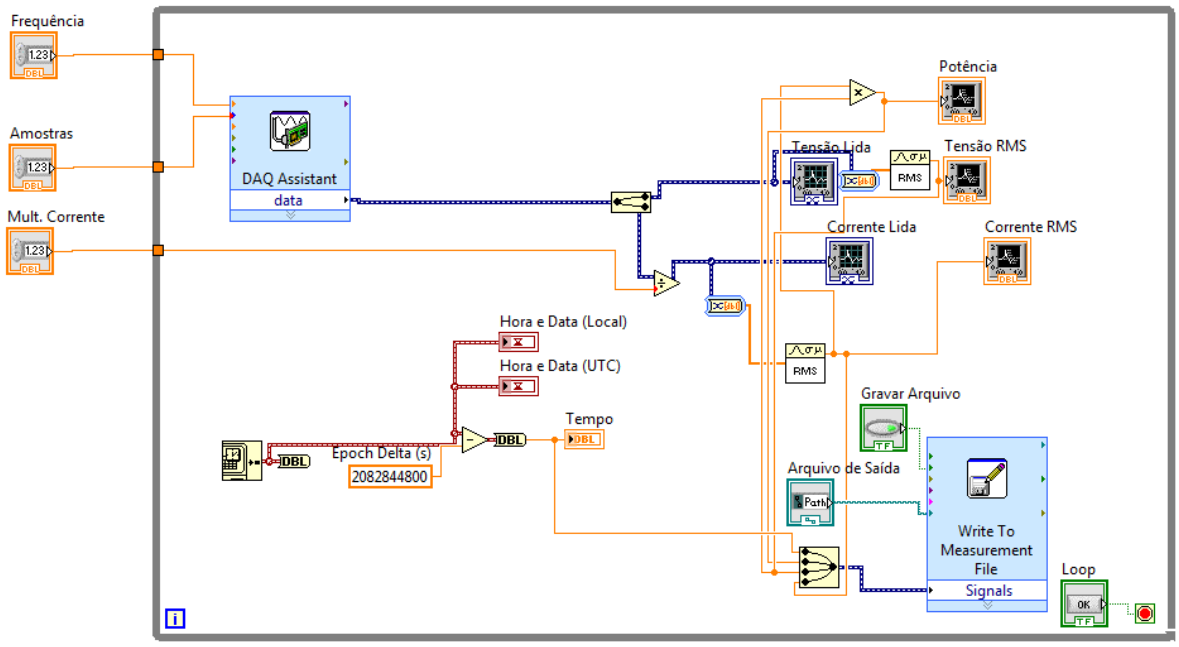
4.1.4.2 Aquisição de Dados

Para realizar a aquisição dos dados gerados pela placa desenvolvida, apresentada na seção anterior, optou-se pelo desenvolvimento de uma aplicação utilizando o LabVIEW, apresentado na seção 2.4.3, juntamente com uma placa de aquisição de dados.

O VI desenvolvido consiste em um laço de repetição com uma rotina de aquisição de dados, representada na figura abaixo pelo elemento DAQ *Assistant*, configurado para capturar os dados de forma contínua, e possuindo três parâmetros de controle, que serão aprofundados na apresentação do painel frontal. Os valores de tensão e corrente adquiridos são plotados em gráficos, bem como seu valor RMS (do inglês *Root Mean Square*), e a potência resultante da multiplicação dos valores RMS de tensão e corrente.

Para posterior análise e sincronização do início da aplicação com uma leitura correspondente gravada em um arquivo de log, foi necessária também a gravação de uma referência de data Epoch, que consiste em um valor em segundos decorrentes desde o dia primeiro de Janeiro de 1070 em sistemas Unix – utilizado nas placas. Como o LabView utiliza como referência o dia primeiro de Janeiro de 1904, padrão dos sistemas Macintosh, foi necessário realizar uma correção, representada no diagrama como Epoch Delta (s) (NATIONAL INSTRUMENTS, 2008).

Figura 22 - Diagrama de blocos do VI desenvolvido

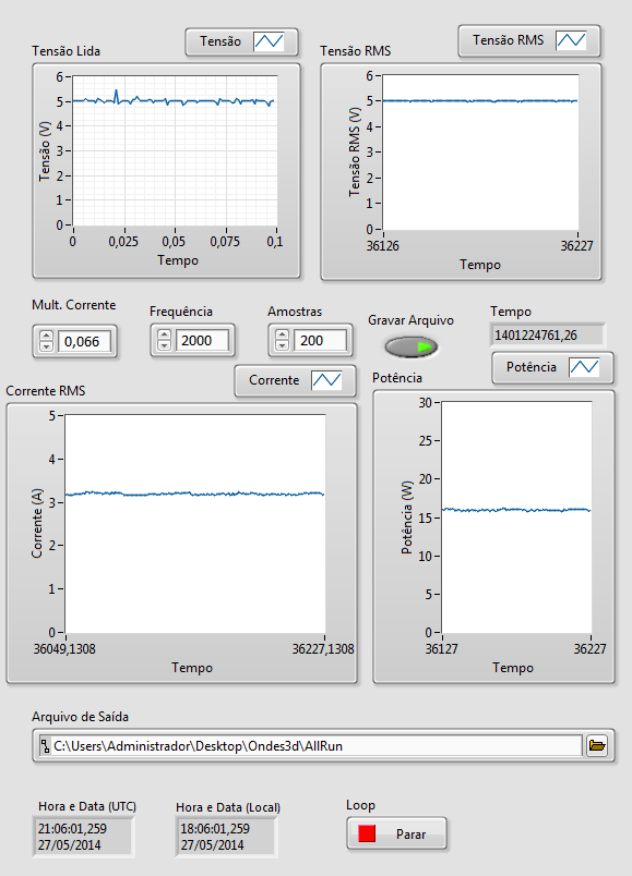


Fonte: Elaborado pelo autor

Os valores adquiridos são representados em elementos alocados em um painel chamado de painel frontal, apresentado na Figura 23. Estes elementos possibilitam a

visualização dos valores lidos de forma gráfica, bem como a configuração dos parâmetros utilizados na aquisição de dados, que são frequência de leitura em Hz, tamanho do buffer de leitura chamado de “Amostras”, e a relação da saída de tensão do sensor de corrente e a corrente passante (66 mV/A) chamado de “Mult. Corrente”. Além disso é possível habilitar ou desabilitar a gravação de arquivos, e alterar o nome do arquivo de saída.

Figura 23 - Painel frontal do VI desenvolvido



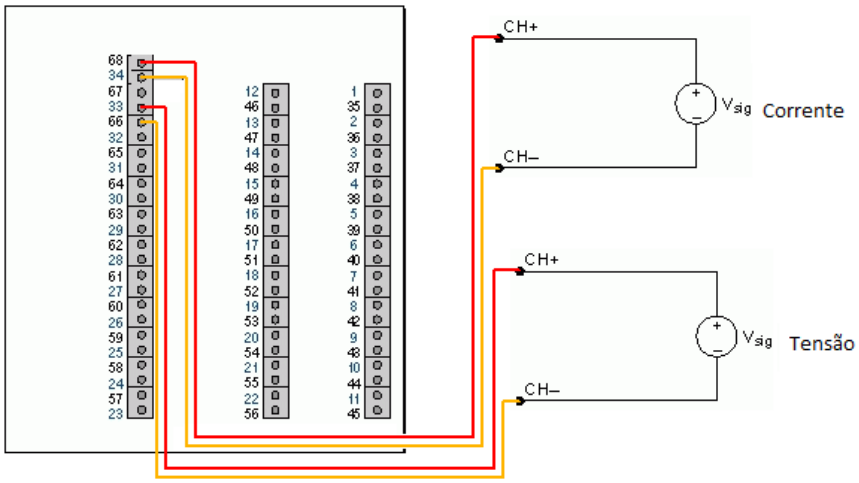
Fonte: Elaborado pelo autor

Para que o DAQ *Assistant*, presente no VI, realize as leituras é necessária uma placa de aquisição de dados. A placa utilizada durante os experimentos foi a PCIe-6341 desenvolvida pela National Instruments, que conta com entradas e saídas analógicas e digitais, além de contadores/temporizadores e gerador de frequência. No entanto, como foram utilizadas apenas as entradas analógicas, estes serão os únicos aspectos analisados.

A placa PCIe-6341 possui 16 entradas analógicas, com conversor analógico-digital com resolução de 16 bits. A taxa de amostragem máxima é de 500 kS/s e a faixa de tensão pode variar entre -10V e 10V. Para a leitura dos valores de tensão da alimentação do cluster foi utilizada uma faixa nominal de -10 V a 10 V, cujo erro segundo o fabricante é de 2,19 mV. Já para a leitura dos valores de corrente resultantes do sensor, a faixa nominal utilizada foi de -0,2 V a 0,2 V com erro de 60 µV (NATIONAL INSTRUMENTS, 2013).

A ligação entre a placa contendo o sensor, apresentada na seção 4.1.4.1, e a placa de aquisição de dados ocorre através de um bloco de conectores, neste caso, o SCB-68 da National Instruments, que por sua vez é conectado a placa de aquisição de dados através de um cabo blindado, com isolamento físico entre os diversos tipos sinais (analógico e digital) de modelo SHC68-68-EPM da mesma fabricante. As entradas foram utilizadas conforme a Figura 24.

Figura 24 - Conexões no bloco de conectores



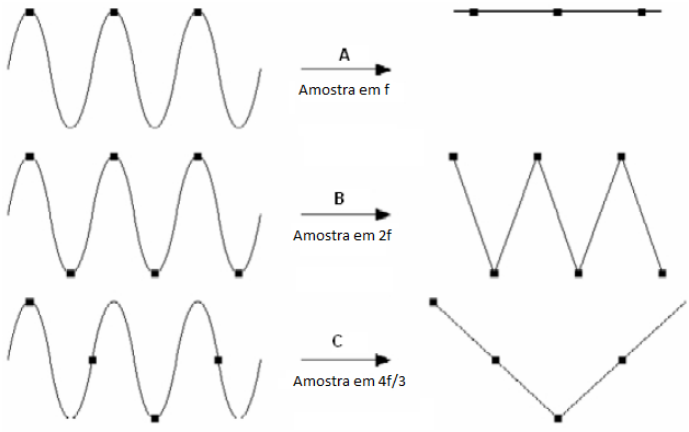
Fonte: Gerado pelo LabVIEW.

4.2 Método de Medição

O sistema de medição de consumo utilizado em todos os testes foi o mesmo, e os parâmetros para obtenção dos dados foram configurados de acordo com o teorema de Nyquist, onde a taxa de amostragem deve ser maior que duas vezes a componente

da maior frequência de interesse do sinal medido. Caso isso não ocorra, o sinal pode ser amostrado errado como demonstrado na Figura 25. Como o filtro passa baixa apresentado na seção 4.1.4.1 atenua frequências maiores que 800 Hz, a taxa de amostragem escolhida foi de 2 kHz, ou seja, acima de 2f, e um buffer de 200 amostras. (NATIONAL INSTRUMENTS, 2014).

Figura 25 - Efeitos do uso de diferentes taxas de amostragem em um sinal



Fonte: National Instruments (2014)

O arquivo de log gravado pelo VI possui uma leitura armazenada a cada 0,11 segundos, portanto, para estabelecer com exatidão o início e o término da execução da aplicação dentre as leituras realizadas, tanto a máquina que está executando o VI, quanto os nós do cluster foram sincronizados com o mesmo servidor NTP antes de iniciar qualquer um dos experimentos listados abaixo, e sempre que qualquer uma aplicação iniciava ou finalizava, este momento era armazenado em um arquivo.

4.3 Experimentos

Um experimento consiste em um ou mais testes realizados sobre um mesmo sistema ou processo, objetivando identificar quais variáveis têm influência sobre as respostas, ou determinar quais valores devem ser associados a uma determinável

variável de forma que a variabilidade das respostas seja minimizada (MONTGOMERY, 1999 apud FREITAS FILHO, 2008).

Os experimentos realizados a seguir, chamados de Experimento 1 a 4, seguem a estratégia de experimentação simples, também conhecida como projeto experimental, que segundo Freitas Filho (2008) deve apresentar a variação de apenas um fator, enquanto os demais permanecem com seus valores fixos. Quando este fator atingir um número de variações considerado aceitável, ele tornar-se-á fixo e um segundo parâmetro começará a sofrer variações controladas.

Os experimentos realizados, bem como seus objetivos, softwares utilizados, e parâmetros variados serão descritos nas seções seguintes. Cada um dos experimentos listados foi executado 5 vezes para cada mudança de parâmetro, a fim de aumentar a confiabilidade das leituras realizadas.

4.3.1 Experimento 1

O Experimento 1 utiliza o software Ondes3D, que possui como principal parâmetro o número de *Time Steps*, ou seja, o número de intervalos de tempo em que o software realizará a simulação de propagação do sismo. Por ser uma aplicação paralela que faz uso de MPI, outra possiblidade seria a variação do número de nós processando, e o número de processos.

Neste primeiro experimento, seguindo a estratégia de experimentação simples apresentada anteriormente, será variado apenas o número de *Time Steps* em apenas um nó, com um processo já que os nós possuem apenas um núcleo cada. O objetivo é avaliar mudança tanto no tempo de execução, como no consumo energético de acordo com o número de *Time Steps* processados, e avaliar se é possível encontrar um ponto que representar a melhor relação consumo/tempo, ou se existe uma relação direta entre o consumo e o tempo independentemente do tamanho do problema. Para analisar este comportamento, foram utilizados *Time Steps* variando de 10 até 100, com um incremento de 10 *Time Steps* a cada nova execução.

4.3.2 Experimento 2

O Experimento 2 utiliza mesmo software do anterior (Ondes3D), no entanto, o parâmetro que sofreu variações neste caso foi o número de nós processando, enquanto o número de *Time Steps* permaneceu fixo em 100 (maior valor utilizado no teste anterior) com intuito de fornecer um bom número de amostragens do consumo energético.

Neste experimento, serão mantidos ligados todos os 10 nós do cluster, independentemente de estarem ou não processando. Este experimento representa mais fielmente o consumo de um cluster real, uma vez que os nós ociosos em clusters como a GridRS, não são desligados mesmo estando ociosos. O objetivo é analisar o consumo da aplicação em um ambiente mais fiel a realidade atual, e posteriormente analisar a economia que poderia ser obtida caso os nós ociosos fossem desligados.

4.3.3 Experimento 3

O Experimento 3 também utiliza o Ondes3D, e como no teste anterior, o parâmetro que sofre variações é o número de nós processando, enquanto o número de *Time Steps* permaneceu fixo com mesmo valor do teste anterior (100).

Neste experimento, serão mantidos ligados apenas os nós que estiverem processando, ou seja, apesar do cluster possuir 10 nós, quando apenas 5 processos estiverem sendo executados apenas 5 nós permanecerão ligados. O objetivo é analisar o consumo apenas da aplicação, e a sua progressão, desconsiderando o consumo dos nós que estariam ociosos.

4.3.4 Experimento 4

Por fim, o Experimento 4 objetiva analisar o desempenho por Watt – métrica proposta pelo Green500 - do cluster através da utilização do benchmark HPL. O benchmark foi executado em um único nó e em todos os 10 nós. Para que o benchmark ficasse executando por um tempo mínimo de 10 minutos conforme recomendado no manual de medição de consumo do Green500, dois tamanhos de problema (Ns) diferentes foram utilizados.

Diversos parâmetros foram testados, dentro das limitações dos nós disponíveis (memória, número de nós e processadores). A Tabela 9 apresenta os principais dados dos arquivos de configuração (HPL.dat) que obtiveram o melhor desempenho (flops) em cada caso, sendo que Ps e Qs representam a quantidades de processos, NBs o número de blocos e Ns o tamanho da matriz, parâmetros detalhados anteriormente na seção 2.4.1.

Tabela 9 – Parâmetros do benchmark HPL

	1 Nó	**10 Nós**
Ns	5120	10000
Ps	1	1
Qs	1	10
NBs	128	128

Fonte: Elaborado pelo autor.

Além da análise de quantos flops por Watt o cluster é capaz de processar, também será verificado se a extrapolação do consumo de um nó pelo número de nós que compõe o cluster, conforme recomendado pelo Green500, condiz com as medições realizadas.

5 RESULTADOS

Este capítulo tem como objetivo apresentar a análise e discussão dos resultados obtidos nos experimentos listados no capítulo anterior. Primeiramente serão analisados os experimentos utilizando o software Ondes3D, e por fim será feita a análise do consumo e desempenho obtido através do benchmark HPL. Devido ao grande número de dados coletados, mais de 1,2 milhões de amostras, todos os dados foram processados através da utilização da linguagem estatística R. Todos os resultados que apresentam a média dos dados coletados, é possível observar o intervalo de confiança (estimado com uma distribuição t e 95% de confiança) que é mostrado no gráfico como uma sombra. Esta sombra pode estar invisível ao olho em alguns pontos indicando que o intervalo de confiança foi muito pequeno.

Os dados de consumo são apresentados em Joule, obtidos através da equação apresentada abaixo, onde P representa o consumo em Watts, e Δt o intervalo de tempo em segundos entre as amostras.

$$E(J) = \sum_{i=1}^{N} P(i) \times \Delta\mathrm{t(i)}$$

5.1 Experimentos Utilizando Ondes3D

Esta seção apresenta a análise dos experimentos 1 a 3, realizados com o software Ondes3D, bem como as deduções realizadas sobre os dados coletados.

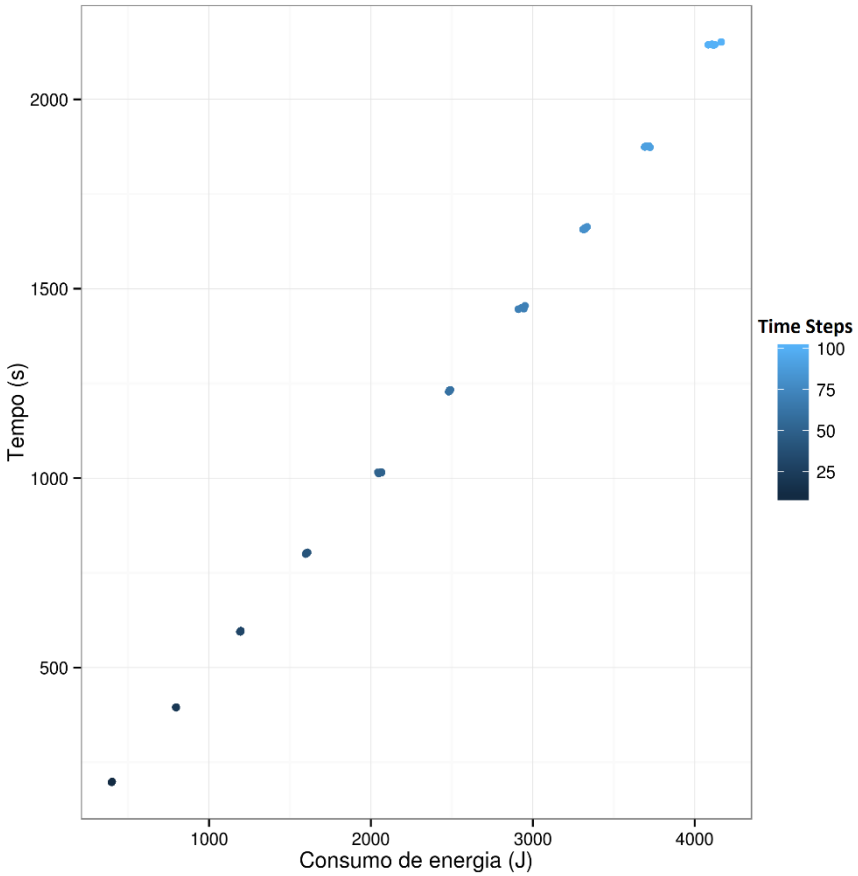
5.1.1 Experimento 1

O Experimento 1, como mencionado anteriormente busca avaliar as mudanças do consumo e do tempo com relação ao tamanho do problema a ser resolvido.

O Gráfico 2 apresenta a energia consumida em Joules e o tempo gasto em segundos, para chegar a solução, de acordo tamanho de problema. Como pode ser observado, existe uma linearidade na progressão tanto do consumo quanto do tempo, ou seja, quanto maior o número de *Time Steps*, maior será o consumo.

Sendo assim, o ponto ótimo para obter a melhor relação consumo/tamanho da entrada, é diretamente proporcional ao tempo, quanto mais tempo maior o consumo e vice-versa.

Gráfico 2 - Relação do consumo e do tempo com o número de *Time Steps* utilizando somente uma placa

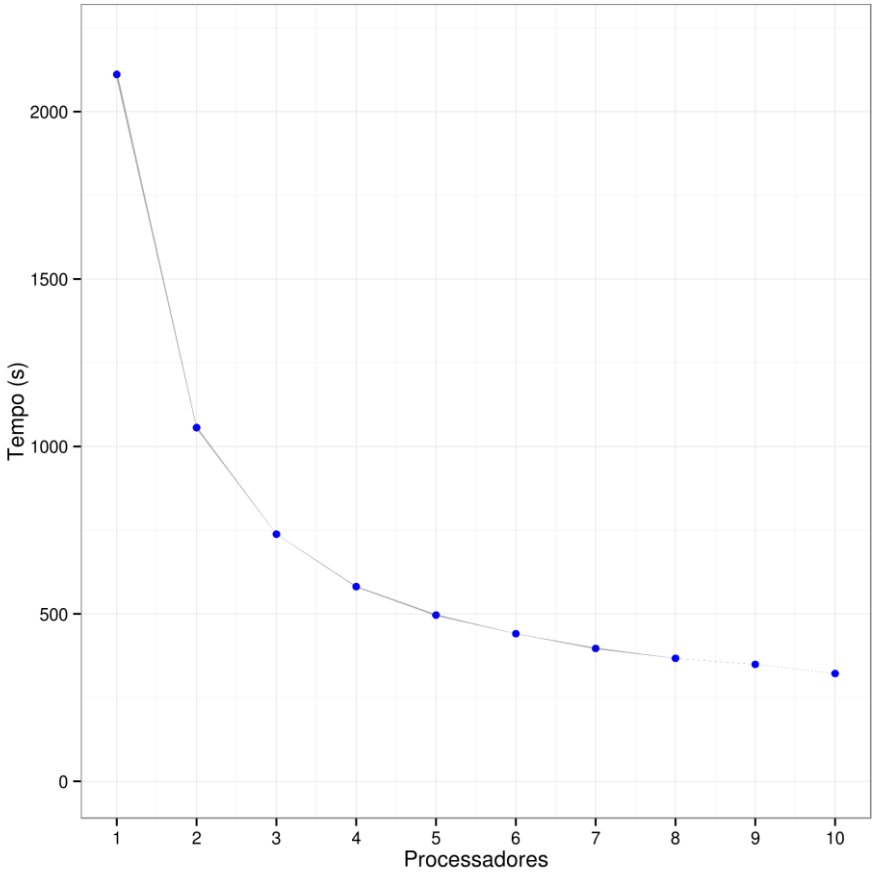


Fonte: Elaborado pelo autor.

5.1.2 Experimento 2

No Experimento 2, a entrada foi mantida fixa, o número de nós processando, foi variado, e todos os nós foram mantidos ligados. Como é possível observar no Gráfico 3, quanto maior o número de nós processando, menor será o tempo necessário para concluir a tarefa.

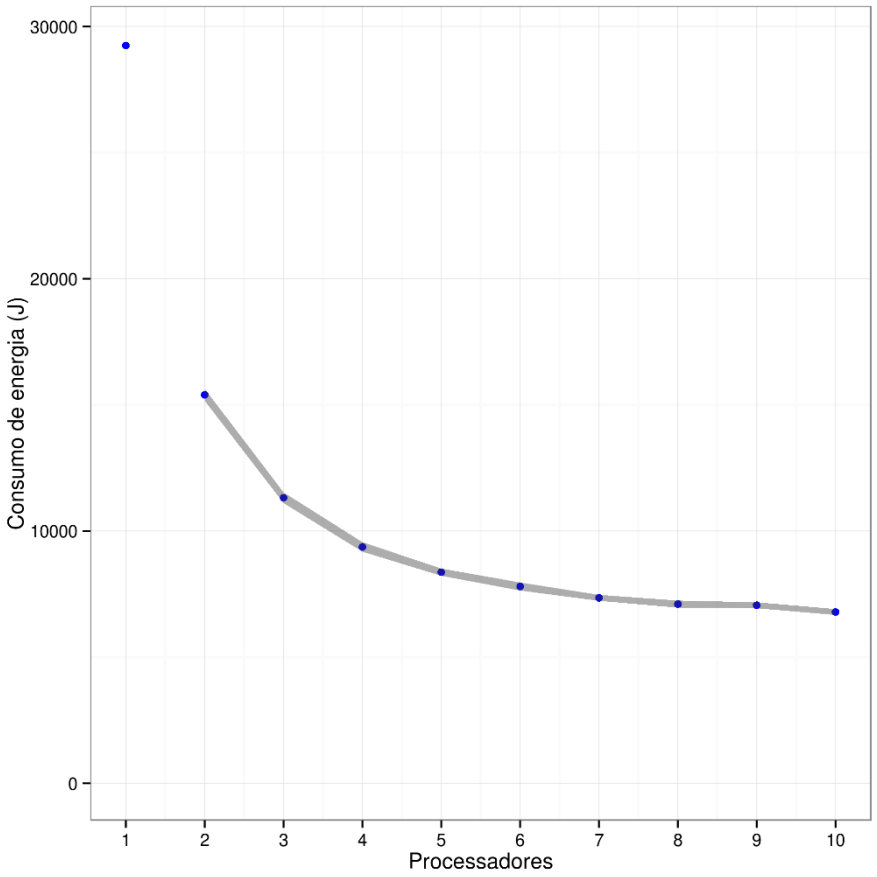
Gráfico 3 - Relação do tempo com o número de nós processando com todos os nós ligados



Fonte: Elaborado pelo autor.

Ao analisar o Gráfico 4 é possível observar um comportamento similar ao da curva de consumo, ou seja, quanto maior o número de máquinas processando, menor será o consumo.

Gráfico 4 - Relação do tempo com o número de nós processando mantendo todos os nós ligados



Fonte: Elaborado pelo autor

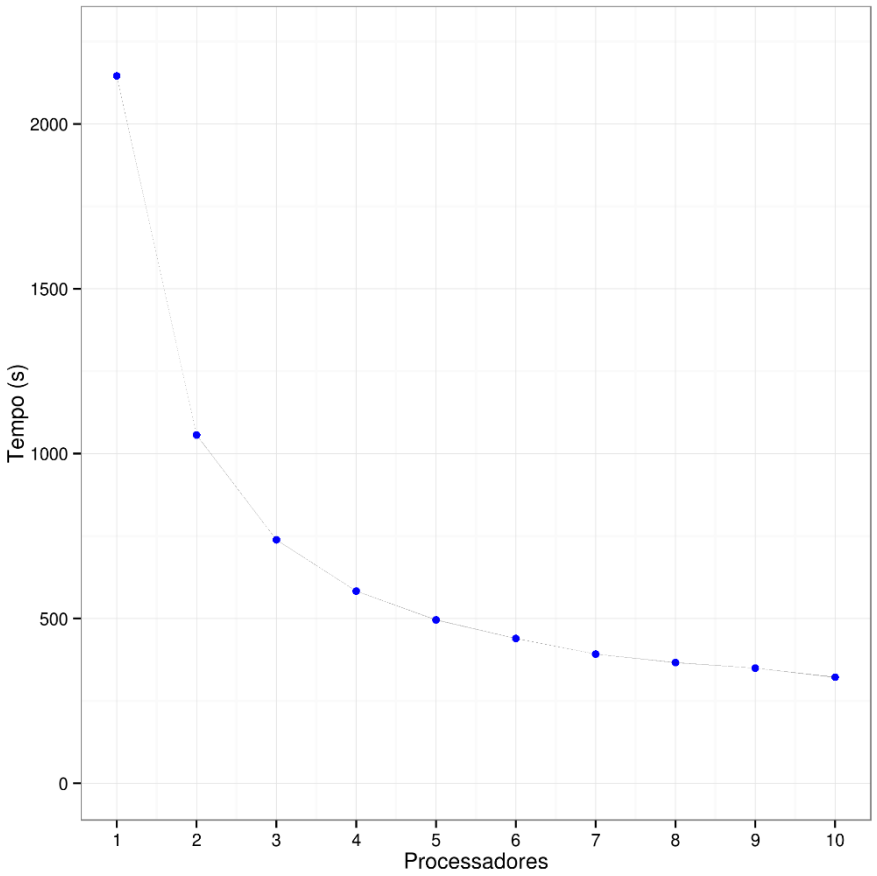
A relação de consumo e tempo pelo número de nós processando apresentada neste experimento representa mais fielmente as grids em funcionamento atualmente. Os nós estão ligados a maior parte do tempo, quer tenham tarefas alocadas para eles, ou não.

5.1.3 Experimento 3

No Experimento 3, a entrada foi mantida fixa e o número de nós, foi variado, sendo que apenas os nós que estavam processando foram mantidos ligados. Como é possível observar no Gráfico 5, quanto maior o número de nós processando

(Processadores) menor será o tempo necessário para concluir o processamento da aplicação, como ocorreu no experimento anterior.

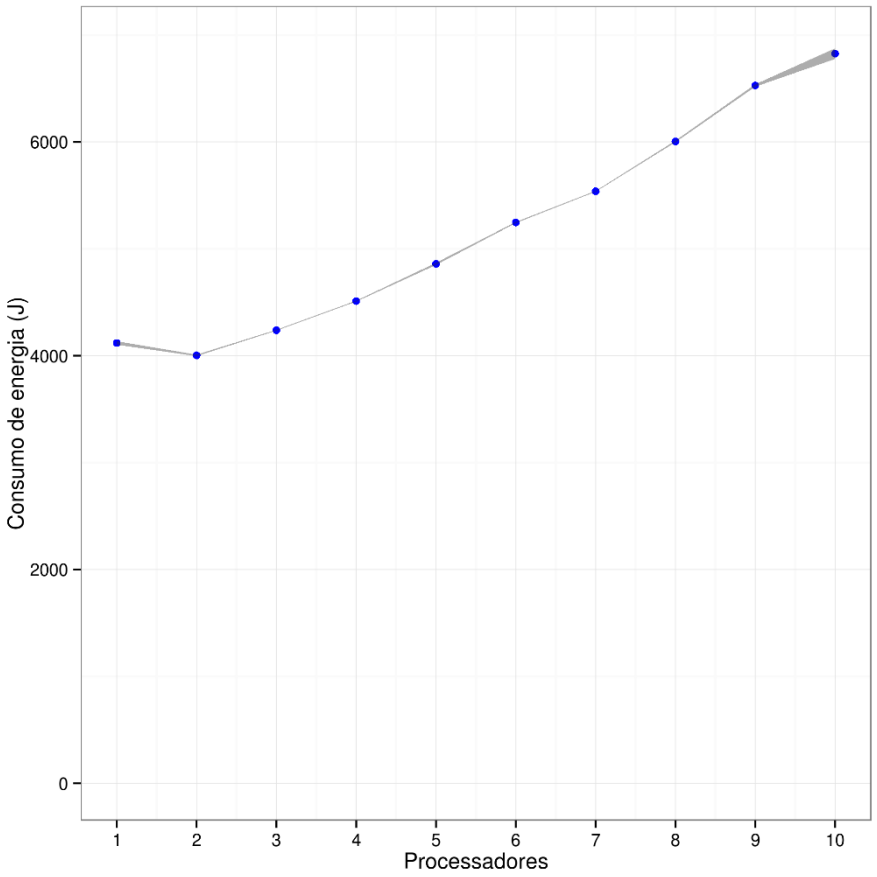
Gráfico 5 - Relação do tempo com o número de nós processando desligando os nós ociosos



Fonte: Elaborado pelo autor.

Em contrapartida, quanto maior o número de nós, maior será o consumo de energia, como pode ser observado no Gráfico 6. A única exceção ocorre quando apenas um nó está processando, isto deve-se ao fato do tempo de processamento ser 2,03 vezes maior que o tempo com dois nós executando. Ou seja, apesar de apenas uma placa estar ligada, o tempo excede o dobro fazendo com que o consumo para finalizar o processo seja maior.

Gráfico 6 - Relação do consumo de energia com o número de nós processando desligando os nós ociosos



Fonte: Elaborado pelo autor.

Vale ressaltar, que o que ocorre não é um aumento no consumo dos nós, mas sim a redução do consumo decorrente dos nós ociosos, ocasionando a inversão da relação de consumo. Neste gráfico fica mais claro o consumo da aplicação, desconsiderando o consumo gasto apenas para manter os demais nós ligados Na Tabela 10 é possível observar o percentual de energia que poderia ser economizado com relação ao experimento anterior. Vale ressaltar que esta economia não impacta no desempenho, pois os nós que foram desligados não estavam sendo necessários para a aplicação.

Tabela 10 - Percentual de economia de energia de acordo com o número de nós processando.

Nós	1	2	3	4	5	6	7	8	9	10
Consumo (%)	85,91	74,01	62,55	51,85	41,94	32,74	24,67	15,46	7,50	-

Fonte: Elaborado pelo autor.

Outro ponto importante observado, após analisar o consumo médio das execuções, é que apesar de não existir um ponto ótimo de consumo utilizando apenas uma placa, como demonstrado no Experimento 1, é possível determinar esse ponto utilizando mais nós. A Tabela 11 apresentar a média dos valores de consumo energético das diversas amostras, apontando que para este número de *Time Steps*, a melhor relação de consumo de energia pode ser obtida utilizando 2 nós.

Tabela 11 – Relação consumo de energia para solução pelo número de nós

Número de nós	**Consumo energético para a solução (J)**	**Economia com relação ao nó mais econômico (%)**
1	4118,882	2,9
2	4002,930	0
3	4238,214	5,88
4	4510,485	12,67
5	4858,918	21,38
6	5246,053	31,05
7	5537,877	38,34
8	6004,487	50
9	6527,813	63,07
10	6825,506	70,51

Fonte: Elaborado pelo autor.

5.2 Experimentos Utilizando HPL

Este experimento avaliou o desempenho do cluster utilizando o benchmark HPL em um nó e em todos os 10 nós do cluster. A média do desempenho obtido em um único nó foi de 67,06 Mflops, superior ao desempenho obtido na versão anterior da placa, testada por Padoin (2012), no entanto inferior a placa PandaBoard, ambas apresentadas na Tabela 5. Já na execução com 10 nós, a média das 5 execuções realizadas foi de 551,9 Mflops.

Como o objetivo é demonstrar a relação desempenho/consumo do cluster desenvolvido, foi necessária a conversão dos dados de consumo energético medido em Joules para consumo instantâneo medido em Watt, conforme recomendado pelo Green500. Como um Joule equivale a um Watt por segundo, esta conversão pode ser realizada a partir da equação apresentada abaixo.

$$Consumo\ Instantâneo\ (Watt) = \frac{Consumo\ Energético\ (Joules)}{Tempo\ de\ Execução\ (Segundos)}$$

Aplicando esta equação sobre os dados obtidos durante a execução dos testes, pode-se determinar quantos Mflops/Watt o cluster atinge, conforme apresentado na Tabela 12, bem como avaliar a técnica de extrapolação dos dados de consumo e desempenho normalmente utilizada para obtenção desta métrica em clusters de grande porte.

Tabela 12 - Dados do HPL e medição de consumo

	1 Nó	**10 Nós**
Desempenho (Mflops)	67,066	551,9
Energia (J)	2504,47	24617,5
Tempo (s)	1355,83	1219,25
Consumo Instantâneo (W)	1,847	20,191
Desempenho por Watt (Mflops/Watt)	36,311	27,33

Fonte: Elaborado pelo autor.

Como pode ser observado, neste caso, os dados coletados da execução nos 10 nós apresentam um desempenho 21,5% menor do que o valor que seria obtido com a extrapolação do desempenho de apenas um nó (670,66 Mflops). Um dos possíveis fatores responsáveis pelo não crescimento linear do desempenho é a carga da rede, devido a troca de mensagens entre os diversos nós, que pode atuar como um gargalo devido a sua baixa velocidade (100 Mbps). A extrapolação apenas do valor do consumo, fica mais próxima do valor medido, apresentando um consumo 9,3% menor. Caso ambos os valores (desempenho e consumo) fossem multiplicados pelo número de nós e aplicados sobre a equação apresentada anteriormente, o valor obtido seria 36,31 Mflop/Watt, um valor 32,86% maior do que o obtido durante as medições, superestimando assim a capacidade real do cluster.

6 CONCLUSÃO

Este documento apresentou o trabalho de conclusão realizado como um dos requisitos para obtenção do título de Bacharel em Engenharia da Computação. As principais contribuições foram a criação de um cluster composto por computadores de placa única baseados em SoC com processadores ARM, a análise do consumo destas placas, bem como a definição de uma forma mais eficiente de análise de consumo.

Este projeto tem uma ampla aplicação prática pois envolve a concepção de um cluster e também a análise de desempenho e consumo de processadores ARM. O trabalho também faz o uso de aplicações reais oriundas de grandes desafios científicos (e-ciência) como é o caso da sismologia na predição de terremotos. Neste contexto o trabalho tem importante contribuição científica. Um fator que comprova o teor científico do trabalho é a cooperação que foi estabelecida com a UFRGS e o professor Philippe O. A. Navaux através do uso da GridRS. E em escala internacional envolvendo o uso da aplicação científica Ondes3D, desenvolvida e mantida por Dr. Fabrice Dupros chefe de projeto na empresa estatal francesa BRGM.

Com base na análise dos resultados obtidos, é possível concluir que o uso apenas de processadores ARM, não apresenta uma solução para atingir o Exascale, no entanto, a combinação destes processadores com GPU pode ser, como já está sendo analisado pelo projeto Mont-Blanc Zero através do uso de placas NVIDIA Tesla K20.

Outras grandes contribuições são, a comprovação de que é possível obter um ponto com a melhor relação consumo/número de nós, de acordo com o tamanho do problema a ser resolvido, ou seja, um maior número de nós e um menor tempo para a solução não representam necessariamente o menor consumo energético. O

expressivo percentual de energia que poderia ser economizado se os nós de um cluster fossem ligados de acordo com a demanda de uma determinada tarefa. Além da demonstração do erro que é obtido com a extrapolação dos valores de consumo de um único nó pelo número de nós que compõe o cluster.

Como sugestão de trabalhos futuros, pode ser citado o desenvolvimento de uma ferramenta de alocação de tarefas que desligue os nós ociosos, e religue-os apenas quando alguma tarefa realmente necessite do recurso, através de pacotes *Wake-on-Lan* por exemplo. Outra possibilidade seria uma ferramenta que analisasse cada aplicação, e de acordo com a sua complexidade e o tamanho da entrada, e determinasse o número ideal de nós a serem alocados a fim de obter um consumo de energia otimizado. Ou ainda, uma plataforma de análise de consumo que processe os dados coletados em tempo real, a fim de adicionar o custo da energia ao custo de locação de uma grid, em projetos como o “rent-a-grid” da multinacional americana Amazon.

REFERÊNCIAS

ALLEGRO MICROSYSTEMS. **Datasheet ACS712**, 2012. Disponível em: <http://www.allegromicro.com/~/media/Files/Datasheets/ACS712-Datasheet.ashx>. Acesso em: 03 abr. 2014.

ALVES, Marcos J. P. **Construindo Supercomputadores com Linux**. Rio de Janeiro: BRASPORT, 2002. 179 p.

AROCA, Rafael Vidal; GONÇALVES, Luiz Marcos Garcia. **Towards green data centers: A comparison of x86 and ARM architectures power efficiency**. Journal of Parallel and Distributed Computing v. 72, 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0743731512002122>. Acesso em: 20 set. 2013.

BESERRA, David Willians S.C; SOUTO, Samuel Carlos Romeiro Azevedo; JOSÉ, Mariel; ANDRADE, Pimentel de; ARAUJO, Alberto Einstein Pereira de. **Comparativo de desempenho de um cluster virtualizado em relação a um cluster convencional sob condições equipotentes**. IX Workshop em Clouds, Grids e Aplicações, Campo Grande, UFMS, 2011. Disponível em: <http://sbrc2011.facom.ufms.br/files/anais/files/workshops/wcga/ST01_1.pdf>. Acesso em: 25 set. 2013.

BRGM. **ONDES3D: USER GUIDE**, 2011. 14 p.

BSC. **Building first supercomputer to combine ARM CPUs, GPU accelerators and InfiniBand**, 2013. Disponível em: <http://www.bsc.es/about-bsc/press/bsc-in-the-media/bsc-building-first-supercomputer-combine-arm-cpus-gpu-accelerators>. Acessado em: 10 nov. 2013.

CATALYUREK, U., STAHLBERG, E., FERREIRA, R., KURC, T., SALTZ, J.: **Improving performance of multiple sequence alignment analysis in multi-client environments**. International Workshop on High Performance Computational Biology (HICOMB), 2002. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1016584>. Acesso em: 15 nov. 2013.

CHVOJKA JÚNIOR, Vladimir. **Tecnologia SOC e o microcontrolador PSoC (Programmable System on Chip)**. São Paulo, Universidade São Judas Tadeu, 2005. Disponível em: <ftp://ftp.usjt.br/pub/revint/251_42.pdf>. Acesso em: 21 set. 2013.

COELHO, Sandro Athaide. **Introdução a Computação Paralela com o Open MPI**. São Pedro, MG: UFJF, 2012 Disponível em: <http://www.ufjf.br/getcomp/files/2013/03/Introdu%C3%A7%C3%A3o-a-Computa%C3%A7%C3%A3o-Paralela-com-o-OpenMPI.pdf>. Acesso em 15 set. 2013.

COLEY, Gerald. **BeagleBone Black System Reference Manual**. Dallas: Texas Instruments, v. A5.2, 2013. Disponível em: <http://www.farnell.com/datasheets/1701090.pdf>. Acesso em 13 nov. 2013.

CORDEIRO, Daniel; BRAGHETTO, Kelly R.; GOLDMAN, Alfredo; KON, Fabio **Da ciência a e-ciência: paradigmas da descoberta do conhecimento**. Revista USP, 97, 2013. Disponível em: <http://www.revistas.usp.br/revusp/article/view/61867/64710>. Acesso em: 5 nov. 2013.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed Systems Concepts and Design**.Terceira edição. Pearson, 2001. 757 p.

DANTAS, Mario. **Computação Distribuída de Alto Desempenho**. Rio de Janeiro: Axcel Books, 2005. 275 p.

DENG, Yuefan; ZHANG, Peng; MARQUES, Carlos; POWELL, Reid; ZHANG, Li. **Analysis of Linpack and power efficiencies of the world's TOP500 supercomputers**, Parallel Computing, v. 39, 2013. Disponível em:<http://dx.doi.org/10.1016/j.parco.2013.04.007>. Acesso em: 17 set. 2013.

DONGARRA, Jack J.; MEUER, Hans W.; SIMON, Horst D.; STROHMAIER, Erich. **Changing Technologies of HPC**, 1997. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.1826&rep=rep1&type=pdf>. Acesso em: 16 set. 2013.

DONGARRA, Jack. HPL - **A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers**. Departamento de ciência da computação, Univesidade do Tenesse, Tenesse, 2008 Disponível em: <http://netlib.org/benchmark/hpl/>. Acessado em 01 nov. 2013.

DORN, M., BURIOL, L.S., LAMB, L.C**.: A molecular dynamics and knowledge-based computational strategy to predict native-like structures of polypepti- des. Expert Systems with Applications** 20, 1–20, 2012. Disponível em: <http://dl.acm.org/citation.cfm?id=2388149>. Acesso em: 15 nov. 2013.

DUCELLIER, Ariane; AOCHI, Hideo; LEE-TIN-YIEN. Yohan. **Introducing anelasticity in a 3D Finite Differences code for the simulation of seismic wave propagation**, 2009. 62 p.

FARACO, Gemma; GABRIELE, Lorella. **Using LabVIEW for applying mathematical models in representing phenomena, Computers & Education**. Departamento de matemática da Universidade de Calabria, Itália, 2007. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360131505001909>. Acesso em: 07 mai. 2014.

FLYNN, Michael J. **Some Computer Organizations and Their Effectiveness. IEEE Computers Transaction** v. C–21 pg. 948–960, 1972. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=500907>1. Acesso em 10 set. 2013.

FREITAS FILHO, Paulo José. **Introdução a Modelagem e Simulação de Sistemas com Aplicações em Arena**. Segunda Edição. Visual Books, 2008. 372 p.

GE, R.; FENG, X.; PYLA, H.; CAMERON, K.; FENG, W.. **Power Measurement Tutorial for the Green500 List**, 2007. Disponível em: <http://www.green500.org/docs/pubs/tutorial.pdf>. Acesso em: 02 abr. 2014.

GNACIO, Aníbal Alberto Vilcapona; FERREIRA FILHO, Virgílio José Martins. **MPI: uma ferramenta para implementação paralela**. Rio de Janeiro, UFRJ. v. 22, n. 1, 2002. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382002000100007&lng=en&nrm=iso>. Acesso em: 17 set. 2013.

GÖDDEKE, Dominik, KOMATITSCH, Dimitri; GEVELER, Markus, RIBBROCK, Dirk; RAJOVIC, Nikola; PUZOVIC, Nikola, RAMIREZ, Alex. Journal of Computational Physics v. 237 pg. 132–150, 2013. **Energy efficiency vs. performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster**. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0021999112007115>. Acesso em 8 nov. 2013.

GREEN500. **About The Green500**. Disponível em: <http://www.green500.org/about>. Acessado em 01 nov. 2013.

GRIDRS. **The GridRS Platform**. Disponível em: <http://gridrs.lad.pucrs.br/> Acessado em: 05 nov. 2013.

HEY, Tony; TREFETHEN, Anne. **The Data Deluge: An e-Science Perspective. Grid Computing** – Making the Global Infrastructure a Reality, Wiley, 2003. Disponível em: <http://eprints.soton.ac.uk/257648/1/The_Data_Deluge.pdf>. Acesso em: 12 set. 2013.

HPL. **HPL Algorithm**. Disponível em: <http://www.netlib.org/benchmark/hpl/algorithm.html>. Acessado em 01 nov. 2013.

INSTRUTHERM. **Especificações da fonte de alimentação FA-3050**. Disponível em: <http://www.instrutherm.com.br/instrutherm/product.asp?template_id=60&old_template_id=60&partner_id=&tu=b2c&dept_id=2000&pf_id=03295&nome=Fonte+de+Alimenta%E7%E3o+Digital+Sim%E9trica&dept_name=Eletroeletr%F4nica>. Acesso em: 02 abr. 2014.

JACOB Bart; BROWN Michael; FUKUI Kentaro; TRIVEDI Nihar. **Introduction to Grid Computing**. Austin: IBM Redbooks, 2001. 261 p.

KIEPERT, Joshua. **Creating a Raspberry Pi-Based Beowulf Cluster**. Boise State University, 2013. Disponível em: <http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf>. Acesso em: 05 nov. 2013.

KOGGE, Peter; BERGMAN, Keren; BORKAR, Shekhar; CAMPBELL, Dan; CARLSON, William; DALLY, William; DENNEAU, Monty; FRANZON, Paul; HARROD, William; HILL, Kerry; HILLER, Jon; KARP, Sherman; KECKLER, Stephen; KLEIN, Dean; LUCAS, Robert; RICHARDS, Mark; SCARPELLI, Al; SCOTT, Steven; SNAVELY, Allan; STERLING, Thomas; WILLIAMS, Stanley R.; YELICK, Katherine. **ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems**, 2008. Disponível em: <http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf>. Acesso em: 10 nov. 2013.

KOPP, Ernani Maieski. **Construção de um cluster HPC para simulações de CFD**. Dissertação (Pós Graduação em Teleinformática e Redes de Computadores) – Departamento de eletrônica, Universidade Tecnológica Federal do Paraná, Curitiba, 2012. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/802>. Acesso em: 10 set. 2013.

KRITIKAKOS, Panagiotis. **Low-Power High Performance Computing**. Dissertação (Mestrado em Computação de Alto Desempenho), Universidade de Edimburgo, Escócia, 2011. Disponível em <https://www.epcc.ed.ac.uk/sites/default/files/Dissertations/2010-2011/PanagiotisKritikakos.pdf>. Acesso em: 14 mai. 2014.

LAMPORT, Leslie. **Time, clocks, and the ordering of events in a distributed system. Communications of the ACM** v.21, 1978. Disponível em: <http://research.microsoft.com/en-us/um/people/lamport/pubs/time-clocks.pdf>. Acesso em 10 set. 2013.

LINPACK. **The Linpack Benchmark**. Disponível em: <http://www.top500.org/project/linpack/>. Acessado em: 02 nov. 2013.

MACDONALD, Neil; MINTY, Elspeth; HARDING, Tim; BROWN, Simon. **Writing Message-Passing Parallel Programs with MPI**, Philadelphia 1987. Disponível em: <http://www.zib.de/zibdoc/mpikurs/mpi-course.pdf>. Acesso em: 16 set. 2013

MEHL, Ewaldo L. M.. **Fontes Chaveadas**. Universidade Federal do Paraná, Departamento de Engenharia Elétrica, Paraná, 2012. Disponível em: <http://www.eletrica.ufpr.br/mehl/downloads/FontesChaveadas.pdf>. Acesso em: 10 jun. 2014.

MICHÉA, D., KOMATITSCH, D.: **Accelerating a three-dimensional finite-difference wave propagation code using gpu graphics cards**. Geophysical Journal International, 2010. Disponível em: < http://dx.doi.org/10.1111/j.1365-246X.2010.04616.x>. Acesso em: 15 nov. 2013.

MISRA, Goldi; KURKURE, Nisha; DAS, Abhishek; DAS, Shweta; GUPTA, Abhinav. **HPC - A Benediction for Agriculture**. International Conference on Information Communication and Management, v. 16. 2011. Disponível em: <http://www.ipcsit.com/vol16/25-ICICM2011M057.pdf> Acesso em: 10 nov. 2013.

NATIONAL INSTRUMENTS. **Conceitos básicos da amostragem analógica**, 2014. Disponível em: <http://www.ni.com/white-paper/3016/pt/>. Acesso em: 06 jun. 2014.

NATIONAL INSTRUMENTS. **Getting Started with LabVIEW,** 2007. Disponível em: <www.ni.com/pdf/manuals/373427c.pdf>. Acesso em: 10 mai. 2014.

NATIONAL INSTRUMENTS. **LabVIEW Timestamp**, 2008. Disponível em: <http://www.ni.com/white-paper/7900/en/>. Acesso em: 09 jun. 2014.

NATIONAL INSTRUMENTS. **NI 6341/6343 Specifications**, 2013. Disponível em: <http://www.ni.com/pdf/manuals/370786d.pdf>. Acesso em: 08 jun. 2014.

PADOIN, Edson L.; OLIVEIRA Daniel A. G. de; VELHO, Pedro; NAVAUX Philippe O. A.. **Time-to-Solution and Energy-to-Solution: A Comparison Between ARM and Xeon**. 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2012), 2012. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6374752>. Acessado em: 06 out. 2013.

PADOIN, Edson L.; OLIVEIRA Daniel A. G. de; VELHO, Pedro; NAVAUX Philippe O. A.. **Evaluating Performance and Energy on ARM-based Clusters for High Performance Computing**. 41st International Conference on Parallel Processing Workshops (ICPPW), 2012.

PEREIRA FILHO, Nelio Alves. **Serviços de pertinência para clusters de alta disponibilidade**. 2004. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2004. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04102004-104700/>. Acesso em: 13 set. 2013.

RAJOVICY, Nikola; PUZOVIC, Nikola; RAMIREZ. Alex. **Tibidabo : Making the Case for an ARM Based HPC System**, 2012. Disponível em:< http://www.montblanc-project.eu/sites/default/files/publications/armhpc-sc.pdf>. Acesso em 10 set. 2013.

RIBEIRO, Christiane Pousa; MÉHAUT, Jean-François. **MAI: Memory Affinity Interface**. Centre de recherche INRIA, Grenoble, 2008. Disponível em: <http://hal.archives-ouvertes.fr/docs/00/49/20/05/PDF/RT-0359.pdf>. Acesso em: 05 mai. 2014.

ROBERTS-HOFFMAN, Katie; HEGDE, Pawankumar. **ARM Cortex-A8 vs. Intel Atom: Architectural and Benchmark Comparisons**. University of Texas, Dallas, 2009. Disponível em: <http://www.ee.unlv.edu/~meiyang/ecg700/readings/ARM%20Cortex-A8%20vs.%20Intel%20Atom.pdf>. Acesso em: 22 set. 2013.

SCHEPKE Claudio, DIVERIO Tiarajú A., NEVES Marcelo V., CHARÃO Andrea S. **Panorama de ferramentas para gerenciamento de clusters**. Instituto de Informática UFRGS, Porto Alegre, 2005. Disponível em:<http://www.inf.ufrgs.br/~cschepke/mestrado/wsppd05.pdf>. Acesso em: 08 set. 2013.

SCOGLAND, Tom; SUBRAMANIAM, Balaji; FENG, Wu-chun. **The Green500 list: escapades to exascale**. Virginia Tech, 2012. Disponível em: http://tom.scogland.com/pubs/pdf/scogland-green500-isc12.pdf>. Acesso em: 8 nov. 2013.

STERLING, Thomas. **Beowulf Cluster Computing with Linux**. Londres: MIT Press, 2002. 496 p.

STROHMAIER, Erich; MEUER, Hans W.. **Supercomputing: What have we learned from the TOP500 project?** Computing and Visualization in Science, v. 6, 2004. Disponível em: <http://link.springer.com/article/10.1007%2Fs00791-004-0132-5>. Acesso em: 17 set. 2013.

TANENBAUM, Andrew S.; STEEN, Maarten van. **Distributed Systems Principles and Paradigms**. New Jersey: Pretince-Hall, 2002. 785 p.

TEAM, Trefis. **Can Intel Challenge ARM's Mobile Dominance?** Forbes Investing, 2012. Disponível em: <http://www.forbes.com/sites/greatspeculations/2012/11/09/can-intel-challenge-arms-mobile-dominance/2/>. Acesso em: 20 set. 2013.

TOP500. **Introduction and Objectives**. Disponível em: <http://top500.org/project/introduction/>. Acessado em 01 nov. 2013.

VERRY, Tim. **Pedraforca Is A Power Efficient ARM + GPU Cluster For Homogeneous (GPU) Workloads**, GPU Technology Conference, 2013. Disponível em: <http://www.pcper.com/news/General-Tech/GTC-2013-Pedraforca-Power-Efficient-ARM-GPU-Cluster-Homogeneous-GPU-Workloads>. Acesso em: 10 nov. 2013.

WATTS UP METERS. **Watts up? operator manual**, 2003. Disponível em: <http://moss.csc.ncsu.edu/~mueller/cluster/arc/wattsup/metertools-1.0.0/docs/meters/wattsup/manual.pdf>. Acesso em: 05 abr. 2014.