UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

JORGE XIMENDES SILVA JR

Avaliação do consumo energético do benchmark NAS em processadores ARM

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Alexandre da Silva Carissimi

Porto Alegre
2015

RESUMO

Com a atual tecnologia construir um sistema computacional que alcance um desempenho Exascale seria necessário uma grande quantidade de energia. Assim uma tendência atual é a utilização de processadores de baixo consumo energético para construir tais sistemas. Neste trabalho será avaliado um cluster de processadores ARM de baixo consumo de energia como forma de verificar a possibilidade da construção de computadores de alto desempenho com essa arquitetura. Para avaliar o desempenho do cluster será utilizado a aplicação NAS em versão OpenMP e MPI.

Palavras-chave: ARM. computação verde. consumo de energia. computação de alto desempenho.

Evaluation of the energy consumption of the NAS benchmark in a cluster of ARM processors

ABSTRACT

With current technology to build a computational system that achieves an Exascale performance a large amount of energy would be required. So a current trend is the utilization of low power processors to build such systems. This work evaluates a cluster of ARM processors as a way to verify the possibility of high-performance computers construction with this architecture. To evaluate the performance of the cluster will be used to NAS application in OpenMP and Open-MPI version.

Keywords: ARM. green computing. energy consumption. high-performance computing.

LISTA DE ABREVIATURAS E SIGLAS

AC	Alternating Current
ACPI	Advanced Configuration and Power Interface
API	Application Programming Interface
APM	Advanced Power Management
CUDA	Compute Unified Device Architecture
CPU	Central Unit Processing
CVBS	Composite Video Baseband Signal
DC	Direct Current
DHCP	Dynamic Host Configuration Protocol
DSP	Digital Signal Processor
EUA	Estados Unidos Da América
EPA	United States Environmental Protection Agency
FPGA	Field-programmable Gate Arrays
GCC	GNU Compiler Collection
GPIO	General Purpose Input/Output
GPU	Graphic Processing Unit
HDMI	High-Definition Multimedia Interface
HPC	High-performance computing
HLT	Halt
I2C	Inter-Integrated Circuit)
I2S	Integrated Interchip Sound
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IR	Infrared

IRDA Infrared Data Association

LED Light Emitting Diode

LRADC Low Rate Analog-to-Digital Converter

MAC Media Access Control

MPI Message Passing Interface

NFS Network File System

NPB NAS Parallel Benchmarks

NUMA Non-Uniform Memory Access

OpenCL Open Computing Language

OpenMP Open Multi-Processing

OSPM Operating System-directed configuration and Power Management

OTG On The Go

PCI Peripheral Component Interconnect

PWM Pulse-Width Modulation

RMS Root Mean Square

RMON Remote Network MONitoring

SATA Serial ATA

SD Secure Digital

SSH Secure Shell

SNMP Simple Network Management Protocol

SOC System On Chip

SPDIF Sony/Philips Digital Interface Format

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

VGA Video Graphics Array

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A geração de energia no mundo hoje está baseada, na sua maior parte, em combustíveis fósseis. Tais fontes são poluentes e não renováveis. Com a crescente preocupação ambiental observada nas últimas décadas, a diminuição do consumo de energia e sua melhor utilização é um assunto em pauta em múltiplas áreas de estudo.

Na computação, o consumo de energia já, é há algum tempo, motivo de preocupações em sistemas embarcados e computadores portáteis. Preocupados com a duração das baterias e a dissipação de calor de tais sistemas, criaram-se diversas técnicas para se reduzir o consumo energético dessas plataformas. Contudo, a constante demanda por melhor desempenho, aliada ao aumento do custo energético propôs o problema do consumo de energia aos desktops e servidores, anteriormente alheios a ele.

O conceito de *Green Computing* (Computação Verde) foi introduzido em 1992, quando o EPA lançou o padrão *Energy Star* (RUTH, 2009). Tratava-se de uma certificação fornecida a equipamentos eletrônicos energeticamente eficientes. A partir de então, o *Energy Star* passou a ser uma certificação importante e reconhecida no mercado. Diversos fabricantes concentraram esforços no desenvolvimento de produtos energeticamente mais eficientes para responder a essa demanda de mercado. Um dispositivo energeticamente eficiente consome uma quantidade de energia proporcional à quantidade de trabalho que ele produz (GUNARATNE; CHRISTENSEN; NORDMAN, 2005). Dessa forma, todo o aumento na quantidade de potência dissipada por esse equipamento é justificada pelo aumento proporcional na quantidade de trabalho produzido por esse aparelho.

A eficiência energética e o consumo de energia tornaram-se uma das questões mais críticas em relação ao projeto e implantação de uma unidade de computação de alto desempenho (HPC - *High-performance computing*). Enquanto o consumo de energia combinado dos sistemas de HPC mundiais continua a ser pequeno em termos relativos (2% do total de emissões de CO_2), em termos absolutos, o custo já é extremamente elevado, cerca de 200-300 bilhões kWh (KOOMEY, 2011; BROWN et al., 2008). Essa tendência irá aumentar ainda mais no futuro próximo, e o custo de energia também está aumentando, que em breve pode levar a uma situação crítica devido aos limites de eficiência energética (KHAN; BOUVRY; ENGEL, 2012). Em 2005, o consumo total de energia para servidores e suas unidades de refrigeração foi projetado em 1,2% do consumo total de energia dos EUA e dobrando a cada cinco anos (KOOMEY, 2007; BROWN et al., 2008).

Uma análise dos atuais sistemas de HPC mostra que 40-60% do consumo de energia

pode ser atribuída aos nós de computação (processadores e memórias), 10% para os sistemas de interconexão e de armazenamento, e o restante (até 50%) é consumida pela própria infraestrutura, incluindo iluminação, fonte de alimentação, e acima de tudo, refrigeração (SCHÄPPI et al., 2009). Consequentemente, o maior retorno pode ser esperado na melhoria da eficiência energética dos nós de computação, o que também se traduzirá em redução nos requisitos de refrigeração.

Assim sendo, em 2008, especialistas alertaram através do relatório oficial DARPA (BERGMAN et al., 2008) que o consumo de energia aceitável para chegar ao exascale - computação exascale refere-se a sistemas de computação capazes de executar, pelo menos, 10^{18} operações de ponto flutuante por segundo - seria de 20 MW. Ao considerar essa recomendação, a eficiência energética dos futuros sistemas exascale tem um limite de 50 Gflops/W. Se sistemas exascale fossem construídos com a tecnologia atualmente empregada, seu consumo de potência alcançariam a ordem de GigaWatts. Isto é equivalente a toda a produção de uma usina de energia nuclear de tamanho médio (WEHNER; OLIKER; SHALF, 2009).

Existem formas para tentar alcançar o desempenho exascale com o uso de servidores com processadores de arquitetura convencional, normalmente usando arquitetura x86, e processadores de arquitetura não convencional, como processadores gráficos, fazendo assim uso de computação heterogênea. Nesse tipo de sistema os processos são distribuídos entre os processadores para maximizar o desempenho e minimizar o consumo de energia. Nessa abordagem tanto os processadores convencionais quanto os processadores não convencionais são utilizados em conjunto.

Outra abordagem é fazer o uso de processadores gráficos para computação de propósito geral através de bibliotecas que exploram o seu potencial computacional. Nessa abordagem somente o processador gráfico é utilizado para maximizar o desempenho.

Por exemplo, o Projeto Mont-Blanc (PROJECT.EU, 2015) é um dos primeiros a introduzir a idéia de um supercomputador baseado em ARM. As apostas do projeto estão em um SOC - *System-on-Chip* heterogêneo sendo uma combinação de um processador ARM e um processador gráfico para atingir alta velocidade de processamento com baixo consumo de energia. Outro motivo para apostar em sistemas com processadores ARM é o custo deles ser muito menor do que processadores com a arquitetura x86.

É Tamanha a preocupação com o consumo de energia que existe um ranking dos servidores que conseguem entregar a melhor relação entre desempenho e consumo de energia chamado *The Green500* (COMPUGREEN, 2015). Em sua última edição no ranking de novembro de 2014, os dez primeiros colocados fazem uso de computação heterogênea.

1.1 Objetivos deste trabalho

Neste trabalho, propõem-se a avaliar e comparar o consumo energético dos processadores ARM existentes nas placas *Cubieboard* e *Jetson*. Para fazer essa avaliação será usado o benchmark NPB nas suas versões *OpenMP* e *MPI*. Como objetivo final dessa avaliação entre estas placas, será definir qual delas é a melhor opção para montar um cluster de baixo custo e baixo consumo.

1.2 Organização do documento

Este trabalho é organizado em 5 capítulos além desta introdução. No capítulo 2 é abordado os principais conceitos de consumo de energia e processamento de alto desempenho. O capítulo 3 fornece as plataformas experimentais analisadas. No capítulo 4 discute os detalhes do avaliação experimental executada sobre as plataformas experimentais. No capítulo 5, a conclusão, onde se ressalta os principais desafios e contribuições deste trabalho. Por fim, as referências bibliográficas.

2 CONSUMO DE ENERGIA E PROCESSAMENTO DE ALTO DESEMPENHO

Neste capítulo, será visto como são expressas as aplicações paralelas e os principais paradigmas de programação, assim como os principais tipos de plataformas disponíveis para executar tais aplicações. Após será discutido o problema do consumo de energia e como definir, reduzir e medir o consumo.

2.1 Programação paralela

O objetivo da programação paralela é obter, em menor tempo que o sequencial, resultados que sejam satisfatórios ao problema apresentado. Em alguns problemas, ou casos, conseguir resolver um problema com entrada maior, ao dividi-lo em partes menores, e distribuí-lo entre várias unidades computacionais. Assim existem dois paradigmas principais para programação paralela.

O primeiro paradigma é a programação paralela com memória compartilhada. Nesse paradigma, a memória pode ser acessada simultaneamente por múltiplos fluxos de execução com a intenção de prover comunicação entre eles ou para evitar cópias redundantes. Ferramentas de programação incluem *POSIX Threads* (ANDREWS, 1999) e *OpenMP (Open Multi-Processing)* (CHANDRA, 2001).

POSIX threads é um padrão para *threads*, o qual define uma biblioteca padrão para criar e manipular *threads*. As bibliotecas que implementam a *POSIX threads* são chamadas *Pthreads*, sendo muito difundidas no universo *Unix* e outros sistemas operacionais semelhantes como *GNU/Linux* e *Solaris*.

O *OpenMP* é uma biblioteca de programação paralela em memória compartilhada disponível para múltiplas plataformas. Permite acrescentar paralelização aos programas escritos em C, C++ e Fortran sobre a base do modelo de execução *fork-join* e está disponível em muitas arquiteturas, incluindo as plataforma *Unix* e *Microsoft Windows*. O *OpenMP* é composto por um conjunto de diretivas de compilador, rotinas de biblioteca, e variáveis de ambiente.

O segundo paradigma é a programação paralela com memória distribuída. Nesse paradigma a comunicação entre processos é através de troca de mensagens. É uma forma de comunicação entre processos que consiste em enviar mensagens a destinatários. A transferência dos dados é uma tarefa cooperativa entre processos, pois para cada operação de envio deve haver uma operação de recebimento relacionada. A Ferramenta de programação inclui o *MPI* (GROPP et al., 1996).

O *MPI (Message Passing Interface)* é a especificação de um padrão para desenvolvimento de bibliotecas de troca de mensagens em ambientes de memória distribuída, criada no início de 1992 pelo *MPI Fórum*. O *MPI* suporta as linguagens de programação: *C* e *Fortran*. Por ser um padrão, as aplicações criadas são portáveis entre diferentes arquiteturas. Esse padrão define rotinas de comunicação ponto a ponto, rotinas de comunicação coletiva e rotinas de gerenciamento de processos.

Existem diferentes implementações do padrão *MPI*. Entre as de domínio público as mais conhecidas são: *CHIMP/MPI – Edinburgh Parallel Computing Center(EPCC)*; *LAM – Ohio Supercomputer Center*; *MPIAP – Australian National University*; *MPICH – Argonne National Laboratory Mississipi State University*; *MPI-FM – University Illinois*; *W32MPI – Universidade de Coimbra - Portugal*. Entre as implementações proprietárias estão as da *IBM* e da *HP*.

Outro paradigma utilizado atualmente é de programação heterogênea que consiste no uso de diferentes arquiteturas em um mesmo nó computacional a fim de se obter melhor desempenho das aplicações que executarão nesse nó. Entre as alternativas destacam-se arquiteturas como GPUs (*Graphic Processing Unit*) e FPGAs (*Field-programmable Gate Arrays*), ambos também conhecidos como aceleradores, em conjunto com processadores tradicionais.

Para poder utilizar as GPUs para processamento de propósito geral deve ser utilizada uma bilioteca específica. Entre as principais utilizadas estão o *CUDA (Compute Unified Device Architecture)* (NVIDIA, 2015) e o *OpenCL (Open Computing Language)* (KHRONOS, 2015).

O CUDA é a tecnologia da *NVIDIA* conhecida como Arquitetura de Computação Paralela de Propósito Geral. O *CUDA* trouxe um novo modelo de programação paralela e um conjunto de instruções, que permitem que aplicações paralelas sejam executadas em GPUs para a solução de problemas complexos mais eficientemente que em CPU. O *CUDA* permite que seus desenvolvedores utilizem C/C++ como linguagem de desenvolvimento. O CUDA explora somente o uso da GPU para executar as aplicações.

O *OpenCL* possui uma filosofia ligeiramente diferente de *CUDA*. Em *OpenCL*, a linguagem e seu *runtime* servem como uma camada de abstração ao hardware heterogêneo. Assim, um programa *OpenCL* tem o objetivo de aproveitar todos os dispositivos presentes na máquina, tais como processadores *multicore*, GPUs, DSPs, entre outros.

2.2 Plataformas de processamento paralelo e distribuído

As principais plataformas para executar uma aplicação paralela são máquinas de memória compartilhada (NUMA - *Non-Uniform Memory Access*), *cluster*, *grid* e *cloud*.

Uma arquitetura NUMA é uma arquitetura de memória compartilhada em computadores multiprocessados. Nessa arquitetura, o acesso à memória executado pelos processadores é não uniforme, o que significa que cada processador tem uma latência diferente ao acessar a memória do computador. Essa latência não-uniforme vem do fato de que cada processador, ou conjunto de processadores, tem os seus bancos de memória local. O conjunto dessas memórias locais de cada processador forma a memória principal do computador. Quando um processador requisita memória, mas não tem mais espaço em sua memória local, ele requisita espaço de memória ao processador vizinho, memória remota. Nesse caso, a latência será maior, por não estar no barramento local e necessário acesso a memória via barramento de comunicação externo e com isso a latência é maior. Nessa arquitetura a ferramenta de programação utilizada é o *OpenMP*.

A arquitetura de *Cluster* pode ser definida como um conjunto de nós computacionais, que cooperam entre si na execução de aplicações utilizando a troca de mensagens. Essa é uma forma de se obter o compartilhamento de recursos computacionais para prover um maior desempenho. Geralmente o acesso ao *cluster* é restrito e dedicado. A arquitetura é escalonável e flexível. Isso significa que nós podem ser adicionados ou removidos, sem a interferência no restante do conjunto. Na maioria das vezes, os nós de um *Cluster* são homogêneos, possuindo os mesmos recursos de memória, processamento, cache e rede. Um tipo famoso é o *cluster* da classe *Beowulf*, constituído por diversos nós escravos gerenciados por um só computador.

Já a computação em grade (*grid computing*) é um modelo computacional que visa a computação distribuída de tarefas pesadas. Nesse modelo, dividi-se os processos entre diversas máquinas que formam uma máquina virtual alcançando um alto poder computacional. As Máquinas de uso *grid* tendem a ser mais heterogêneas e geograficamente dispersas (portanto, não acoplados fisicamente), assim faz-se o uso de troca de mensagens. Embora uma única *grid* possa ser dedicada a uma aplicação particular, normalmente uma *grid* é utilizada para uma variedade de fins.

A computação em nuvem (*cloud computing*), ou simplesmente, *cloud*, está associada a um novo paradigma na área de computação. Basicamente, esse novo paradigma tende a deslocar a localização de toda a infraestrutura computacional para a Internet. Enquanto a computação em *grid* realiza um compartilhamento dos recursos entre os usuários, a computação em nuvem só aloca um recurso a um determinado usuário caso ele queira usá-lo. Logo, isso sugere a idéia

de que o recurso é totalmente dedicado àquele usuário.

Todas as plataformas discutidas anteriormente buscam alcançar o exascale, mas com um grande problema que é o aumento no consumo de energia delas.

2.3 Energia, potência e consumo energético

Energia é uma grandeza física que representa a capacidade de realizar trabalho e é medida em Joules (J), ou em quilowatt-hora (1 kWh = 3600 kJ). A taxa com que a energia é transferida durante a realização de um trabalho é definido como potência. A potência é medida em Watt (W = Joules por segundo). A relação entre energia e potência é dada por:

$$P = dE/dt \tag{2.1}$$

sendo *P* a potência, *E* a energia e *t* o tempo. Em sistemas elétricos, a potência pode ser definida como o trabalho realizado pela corrente elétrica em um determinado intervalo de tempo. Em um sistema de corrente contínua (DC), no qual a tensão (V), medida em Volts, e a corrente (I), medida em Ampères, se mantém constantes durante um dado período, a potência é dada por:

$$P = V * I \tag{2.2}$$

Sabendo que $V = R * I$, podemos chegar à expressão:

$$P = I^2 * R \tag{2.3}$$

Onde *R* é a resistência do circuito, medida em Ohm (Ω).

Para o caso de sistemas de corrente alternada (AC), a tensão é comumente expressa em V_{RMS}. A corrente, por sua vez, é I_{RMS} - RMS (do inglês *root mean square*) ou valor eficaz é uma medida estatística da magnitude de uma quantidade variável. Em sistemas desse tipo, a corrente *I(t)* – e, portanto, a potência instantânea – são funções dependentes do tempo, logo, a equação 2.3 precisa ser estendida para refletir esse fato. Se a função é periódica como ocorre, por exemplo, na rede elétrica pública de transmissão, a análise deve ser feita em relação à potência média dissipada ao longo do tempo. Logo:

$$P_{média} = I(t)^2 * R = R * I_{RMS}^2 \tag{2.4}$$

Ou seja, o valor RMS, I_{RMS}, da função *I(t)* é o valor constante que leva à mesma dissipação de potência que a dissipação de potência média ao longo do tempo da corrente I(t). Podemos, também, estender a relação para um sistema com tensão não constante, V(t), com valor RMS expresso por:

$$P_{média} = V_{RMS}^2/R \tag{2.5}$$

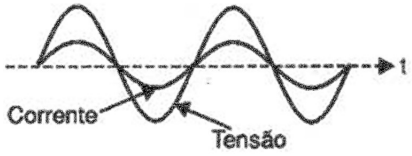
Ao igualar as equações 2.4 e 2.5, obtem-se:

$$P_{média} = I_{RMS} * V_{RMS} \tag{2.6}$$

É importante ressaltar que as equações aqui obtidas consideram apenas sistemas puramente resistivos (por exemplo: chuveiros, lâmpadas incandescentes, ferros de passar roupa), o que significa que não há cargas reativas, tais como capacitores e indutores, que são capazes de não só dissipar energia, mas também de armazená-la. Como exemplo de cargas indutivas, há os motores e os transformadores. Por sua vez, cargas capacitivas podem ser representadas por lâmpadas fluorescentes e computadores.

Quando uma tensão senoidal é aplicada numa carga resistiva, a corrente circulante pela carga acompanha instantaneamente as variações de tensão, como mostra a figura 2.1.

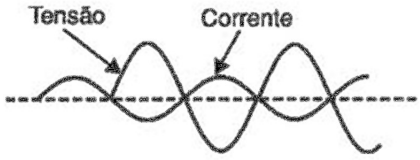
Figura 2.1: Corrente e tensão em carga resistiva.



Fonte: (BRAGA, 2014)

Contudo, na maioria dos casos reais, o sistema não se comporta como uma resistência pura. Adicionando-se componentes reativos ao sistema, o fluxo de potência flutua, ou seja, a corrente pode fluir tanto em um sentido quanto em outro ao longo do tempo. Nesse caso, a corrente não acompanha as variações de tensão instantaneamente, havendo um retardo ou um adiantamento, conforme ilustra a figura 2.2.

Figura 2.2: Corrente e tensão em cargas capacitivas e indutivas.

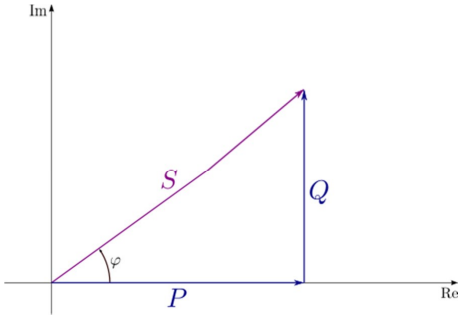


Fonte: (BRAGA, 2014)

Nesse caso, é necessário introduzir alguns conceitos como potência real, ou potência

ativa (P), potência reativa (Q), potência complexa (S) e potência aparente (|S|). A potência real (ativa), medida em Watt, representa a energia gasta em determinado espaço de tempo. A potência reativa, medida em volt-ampère reativo (var), representa a porção do fluxo de potência devido à energia armazenada que retorna à fonte a cada ciclo. Por fim, temos a potência complexa, medida em volt-ampère (VA), e a potência aparente, também medida em volt-ampère (VA), que é o valor absoluto da potência complexa e representa o produto da corrente pela tensão do circuito. Esses conceitos se relacionam como mostrado na figura 2.3.

Figura 2.3: Relação entre potência complexa (S), real (P) e reativa (Q).



Fonte: Jorge Ximendes

O ângulo φ formado pela potência complexa(S) e pela potência real(P) é usado para indicar o fator de potência (FP), que varia entre 0 e 1, e é dado por:

$$FP = cos(\varphi) \tag{2.7}$$

O melhor aproveitamento num sistema de corrente alternada ocorre quando o ângulo de defasagem é zero – ou seja, cosseno igual a 1. Nesse caso, 100% da energia aplicada é convertida em trabalho. Na prática, no entanto, equipamentos raramente atingem esse padrão. Um fator de potência baixo significa que energia reativa está sendo gerada e não é aproveitada (BRAGA, 2014). Com efeito, a medida de potência em Watt de um equipamento eletrônico determina a verdadeira potência que a companhia de energia elétrica deve fornecer para o dispositivo funcionar. Por outro lado, a grandeza volt-ampère (VA) é usada para dimensionar os fios e sistemas de proteção das instalações elétricas. A classificação Watt e VA é igual para alguns tipos de aparelhos, tais como, as lâmpadas incandescentes. No entanto, para equipamentos eletrônicos, essas grandezas podem se diferenciar significativamente, com a medida em VA sendo sempre igual ou maior que a medida em Watt. Em sistemas computacionais, como visto em [Konduri, 1999, apud CARISSIMI et al, 2010], a energia utilizada para realização de um programa

executado em um processador baseados em *switches* CMOS pode ser expressa como:

$$E_{op} = C * V_{dd}^2 * f * (N/f) + I_{fuga} * V_{dd} * (N/f) \quad (2.8)$$

Onde E_{op} é a energia, *C* é a capacitância de chaveamento, V_{dd} é a voltagem na qual o circuito está operando, f é a frequência de operação, I_{fuga} a corrente de fuga e *N* o número de ciclos requeridos para executar o programa. Observando a expressão acima, nota-se que, para diminuir o consumo de energia, é possível reduzir a voltagem de alimentação (V_{dd}). Contudo a frequência de processamento deve ser diminuída para que tal mudança seja possível. A redução da frequência faz com que o processador execute instruções em um ritmo menor, prejudicando o seu desempenho. Surge, então, o compromisso consumo energético versus desempenho.

2.4 Estratégias para redução do consumo de energia

Nesta seção serão descritas as principais estratégias em hardware e software para redução do consumo de energia.

2.4.1 Estratégias em hardware

A introdução da instrução HLT, no início da Era Pentium, foi uma das primeiras tentativas de se reduzir o consumo de processadores. Os projetistas da Intel valeram-se do fato de que o processador não é exigido em seu máximo desempenho todo o tempo. Dessa forma, através da instrução HLT, o sistema operacional, ao identificar que não há nada a executar, pode colocar o processador em um estado de baixo consumo quando isso for interessante.

Após a instrução HLT, os mecanismos em hardware para controle do consumo de energia evoluíram. Se destacando a APM (*Advanced Power Management*) e suas formas para controle do consumo de energia.

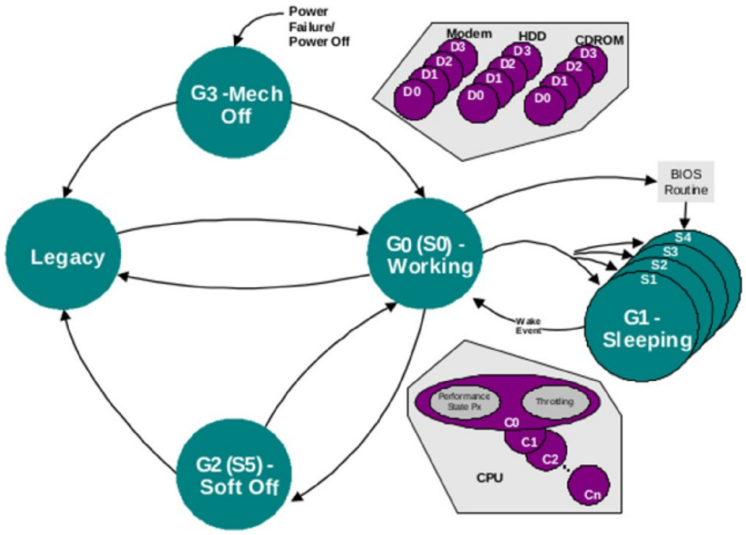
A APM, criada pela Intel em 1992, foi a primeira interface de gerenciamento de consumo entre o *firmware* e o sistema operacional. O APM utiliza uma abordagem em camadas para comunicar dispositivos com o sistema operacional. O hardware é controlado pela APM-*aware* BIOS que se comunica com um *driver* APM específico para o sistema operacional. A comunicação ocorre em ambos os sentidos: eventos de gerenciamento de energia são enviados a partir do BIOS para o *driver* APM, o *driver* APM envia informações e pedidos à BIOS através de chamadas de função. Através da APM, o sistema operacional pode obter informações

pertinentes à gerência de consumo e colocar o sistema em diferentes estados de menor atividade.

Lançada em dezembro de 1996, ACPI (*Advanced Configuration and Power Interface*) (HEWLETT-PACKARD et al., 2013) é o sucessor do APM. A ACPI dá ao sistema operacional toda a responsabilidade do gerenciamento de consumo, tendo em vista que é ele que está na melhor posição para avaliar o *status* de funcionamento do sistema.

A ACPI foi desenvolvida para estabelecer interfaces comuns para a indústria, permitindo um sistema para configuração dos dispositivos presentes na placa-mãe e o gerenciamento de energia desses dispositivos. A ACPI é o elemento-chave na OSPM (*Operating System-directed configuration and Power Management*). A ACPI evoluiu as interfaces de configuração da placa-mãe existentes para suportar arquiteturas avançadas de uma forma mais robusta, e potencialmente mais eficiente. De uma perspectiva de gerenciamento de energia, OSPM / ACPI promove o conceito de que os sistemas devem economizar energia através da transição dos dispositivos não utilizados para estados de menor consumo de energia, incluindo colocar todo o sistema em um estado de baixo consumo de energia quando possível. Para a ACPI conseguir esse objetivo, ela faz uso de diversos mecanismos implementados pelo hardware como mostrados na figura 2.4. Os quais serão discutidos nas próximas seções.

Figura 2.4: Estados do ACPI.



Fonte: (HEWLETT-PACKARD et al., 2013)

2.4.2 Global System State (Gx states)

Os *Gx states* aplicam-se a todo o sistema e são perceptíveis pelo usuários. A tabela 2.1 mostra o resumo dos *Gx states*. Esses estados são divididos em 4 fases:

- G3 *Mechanical Off*: A energia do computador foi totalmente removida por meio de um interruptor mecânico (como um botão na parte de trás da fonte).
- G2/S5 *Soft Off*: Um estado onde o computador consome uma quantidade mínima de energia. Não há, nesse caso, aplicações sendo executadas. Esse estado requer um grande tempo até voltar ao estado G0. O sistema operacional precisa ser reiniciado para voltar ao estado de funcionamento.
- G1 *Sleeping*: Um estado em que o computador consome uma pequena quantidade de energia, as aplicações do usuário não estão sendo executadas e o sistema operacional "parece" estar desligado. O tempo para retornar ao estado de trabalho varia de acordo com o *Sleeping State* selecionado antes da entrada nesse estado. O trabalho pode ser retomado sem reiniciar o sistema operacional.
- G0 *Working*: Um estado onde o sistema operacional está executando aplicações. Nesse estado, os periféricos tem seu estado de energia alterada dinamicamente. O usuário pode selecionar, por meio de uma API, várias características de desempenho/potência do sistema para que o software tenha seu desempenho otimizado ou a vida útil da bateria aumente.

Tabela 2.1: Resumo dos Global System State

Global system state	Software é executado	Latência	Consumo de energia	Necessário restart SO
G0 Working	Sim	0	Alto	Não
G1 Sleeping	Não	>0, varia com sleep state	Baixo	Não
G2/S5 Soft Off	Não	Alta	Muito próximo de zero	Sim
G3 Mechanical Off	Não	Alta	Bateria presente na placa-mãe	Sim

Fonte: (HEWLETT-PACKARD et al., 2013)

Conforme a figura 2.4, no estado G1 existem subestados que são os *Sleeping State (Sx states)*. Os *Sleeping State* são estados de repouso. Os Sx states são estados com baixa tempo de retorno ao estado G0 e eles são brevemente definidos a seguir:

- S1: Nesse estado, nenhum contexto do sistema - refere-se aos estados de operação pro-

cessador, memória cache e dispositivos - é perdido e o hardware mantém todo o contexto do sistema.

- S2: Esse estado é semelhante ao estado de repouso S1, exceto que o contexto da CPU e a cache do sistema são perdidos (o sistema operacional é responsável por manter os caches e o contexto da CPU).
- S3: Todo o contexto do sistema é perdido, exceto a memória do sistema. CPU, cache, e contexto do *chipset* são perdidos nesse estado. O hardware mantém o contexto de memória e restaura algum contexto para a CPU.
- S4: O estado de repouso S4 é o de mais baixo consumo de energia, mas com alto tempo de retorno ao estado G0. A fim de reduzir a um consumo mínimo de energia, assume-se que a plataforma de hardware tem todos os dispositivos desligados. O contexto da plataforma é mantido.
- S5 *Soft Off*: O estado S5 é semelhante ao estado S4, exceto que o sistema operacional não salva qualquer contexto. O sistema está em estado de *soft off* e requer um *boot* completo para ativação.

Assim como ocorre no estado G1, o estado G0 é composto por vários subestados denominados *Processor Power State (Cx states)*. Os *Processor Power State* são estados de consumo de potência e de gerenciamento térmico do processador. Eles possuem semântica de entrada e saída específicos e são brevemente definidos a seguir:

- C0: Enquanto o processador está nesse estado, ele executa as instruções.
- C1: Nesse estado de energia o processador tem a menor latência. A latência do hardware deve ser baixa o suficiente para que o sistema operacional não considere esse aspecto ao decidir se deve usá-lo. Além de colocar o processador em um estado de não execução, esse estado não tem outros efeitos visíveis ao software.
- C2: O estado C2 oferece maior economia de energia sobre o estado C1. A pior latência para esse estado é fornecida pela ACPI e o sistema operacional pode usar essa informação para determinar quando o estado C1 deve ser usado em vez do estado C2. Além de colocar o processador em um estado de não execução, esse estado não tem outros efeitos visíveis ao software.
- C3: O estado C3 oferece maior economia de energia que os estados C1 e C2. A pior latência para esse estado é fornecida pela ACPI e o sistema operacional pode usar essa informação para determinar quando o estado C2 deve ser usado em vez do estado C3. No estado C3, as caches do processador mantêm o estado, mas ignoram qualquer mudança.

O sistema operacional é responsável por garantir que as caches mantenham a coerência.

Ainda utilizando a figura 2.4 como referência, no subestado C0, do estado G0, ainda existem outros subestados que são os *Device and Processor Performance State (Px states)*. Os *Device and Processor Performance State* são estados de consumo de energia e de capacidade enquanto no estado ativo, C0 para processadores e D0 para dispositivos. Os *Px states* são brevemente definidos a seguir.

- P0: Enquanto um dispositivo, ou processador, está nesse estado, ele usa a sua capacidade máxima de desempenho e pode consumir sua potência máxima.
- P1: Nesse estado, o desempenho de um dispositivo, ou processador, é limitado abaixo do seu máximo desempenho e consome menos do que a potência máxima.
- Pn: Nesse estado, o desempenho de um dispositivo, ou processador, está no seu nível mínimo de desempenho e consome potência mínima, permanecendo num estado ativo. O estado *n* é um número máximo e depende do processador ou dispositivo. Os processadores e dispositivos podem definir o suporte para um número arbitrário de estados de desempenho desde que não exceda 16.

2.4.3 Device Power State

Os *Device Power State* são estados de consumo de energia específicos dos dispositivos. Geralmente não são perceptíveis pelo usuário. Por exemplo, alguns dispositivos podem estar no estado Desligado, embora o sistema como um todo está no estado de trabalho. Os estados de energia dispositivo são, genericamente, definidos a seguir. Esses estados são divididos em 5 fases:

- D3 (Off): A energia foi completamente removida do dispositivo. O contexto de dispositivo é perdido quando ele entra nesse estado, de modo que o sistema operacional irá reinicializar o dispositivo quando ligá-lo novamente. Dispositivos nesse estado tem o mais longo tempo de restauração. Todas as classes de dispositivos definem esse estado.
- D3hot: É definido para cada classe de dispositivos. Em geral, D3Hot é utilizado para economizar energia e, opcionalmente, preservar o contexto do dispositivo. Se o contexto do dispositivo é perdido quando ele entra nesse estado, o sistema operacional irá reinicializar o dispositivo quando houver transição para D0. Dispositivos nesse estado podem ter longo tempo de restauração. Todas as classes de dispositivos definem esse estado. O

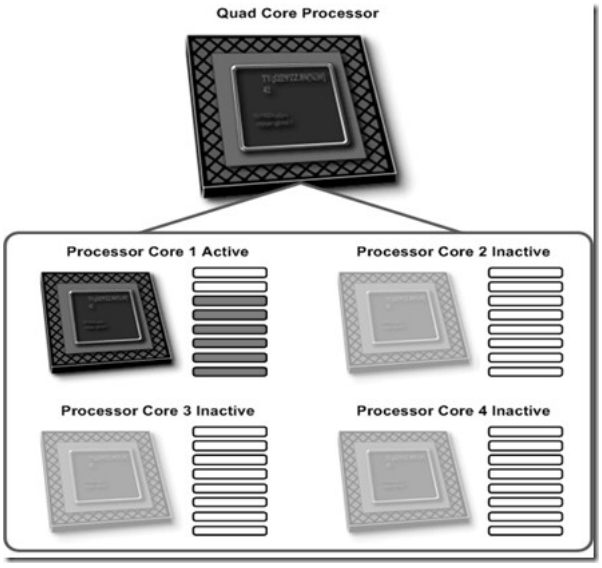
estado D3Hot difere do estado D3 em dois parâmetros distintos: O dispositivo ainda está energizado e software pode acessar um dispositivo em D3Hot.

- D2: É definido para cada classe de dispositivos embora, nem todos o fazem. Em geral, o estado D2 é utilizado para economizar energia e preservar menos contexto que o estado D1 ou D0. O estado D2 pode levar o dispositivo a perder algum contexto (por exemplo, através da redução de energia no barramento, forçando assim o dispositivo a desligar algumas de suas funções).
- D1: É definido para cada classe de dispositivos embora, nem todos o fazem. Em geral, o estado D1 deve economizar menos energia e preservar mais contexto que o estado D2.
- D0 (Fully-On): Esse estado é assumido como sendo o de mais elevado de consumo de energia. O dispositivo está completamente ativo.

2.4.4 Outros recursos

Outra técnica utilizada pelos fabricantes de processadores é utilizar o monitoramento do uso dos núcleos do processador para desligar aqueles que não são usados economizando energia e potência sem afetar o desempenho do sistema. A figura 2.5 mostra um processador com 4 núcleos com 3 deles desativados.

Figura 2.5: Exemplo desse recurso



Fonte: Jorge Ximendes

2.4.5 Estratégias em software

Em nível de núcleo de sistema operacional, a partir do núcleo 2.6.12, o Linux passou a oferecer uma infraestrutura chamada *cpufreq* (REDHAT, 2015). A ferramenta se baseia no conceito de governadores. Um governador é um algoritmo que calcula a frequência que ele considera adequada para a CPU em um determinado momento. Os diferentes tipos de governadores de *cpufreq* disponíveis estão descritos abaixo.

- *cpufreq_performance*: Força a CPU a usar a frequência de relógio mais alta possível. Como tal, esse governador em particular não oferece nenhum benefício de economia de energia.
- *cpufreq_powersave*: Força a CPU a usar a frequência de relógio mais baixa possível. Como tal, esse governador específico oferece economia máxima de energia, mas ao custo de menor desempenho de CPU.
- *cpufreq_ondemand*: É um governador dinâmico que permite que a CPU alcance a frequência máxima de relógio quando a carga do sistema for alta, e também uma frequência mínima de relógio quando o sistema estiver ocioso. Embora isso permita que o sistema ajuste o consumo de energia de acordo com o requisitado, quanto à carga do sistema, ele faz isso ao custo de latência maior entre a mudança de frequência. Como tal, a latência pode retirar qualquer benefício de economia de energia, se o sistema mudar entre ocioso e cargas de trabalho pesadas muito seguido.
- *cpufreq_userspace*: Permite que programas do usuário (ou qualquer processo que estiver executando como administrador) ajuste a frequência da CPU.
- *cpufreq_conservative*: Como o *cpufreq_ondemand*, o *cpufreq_conservative* também ajusta a frequência do relógio de acordo com o uso. No entanto, enquanto o *cpufreq_ondemand* o faz de forma mais agressiva, (ou seja, do máximo para o mínimo e de volta ao máximo), o *cpufreq_conservative* muda entre as frequências gradualmente. Isso significa que o *cpufreq_conservative* irá ajustar uma frequência de relógio que está destinada a suprir a carga, ao invés de simplesmente escolher entre o máximo e mínimo. Embora isso possa fornecer muita economia no consumo de energia, ele o faz em uma latência ainda maior do que o *cpufreq_ondemand*.

2.5 Medição do consumo de energia

Há vários equipamentos e formas para o medir o consumo de energia. Os wattímetros são dispositivos colocados em série com o sistema a ser medido e assim obtém o seu consumo. Tendem a ser precisos.

Os alicates de corrente alternada são dispositivos que medem a variação no campo magnético para medir a corrente alternada que passa por um condutor. Este tipo de equipamento não influencia na medição, pois não tem ligação física com o sistema a ser medido. São precisos mas só funcionam com corrente alternadas altas. Os multímetros são dispositivos que medem a tensão, ou a corrente do sistema, e, por serem de construção mais simples tendem a interferir na medição. O único problema de wattímetros, alicates de corrente alternada e multímetros é que todos eles medem o consumo do sistema como um todo.

Há casos em que o próprio sistema computacional tem recursos para medir o seu consumo de energia. Os dois principais recursos são: *Chips* dedicados para medir a tensão e corrente do sistema de forma precisa e com uma alta taxa de atualização dos dados medidos; Registradores especiais, internos aos processadores, sem que o próprio processador atualiza, conforme faz as suas operações, o consumo de energia dele de forma precisa.

Mesmo com os recursos já citados, há vezes em que não é possível medir o consumo de potência de um processador, seja pela falta de um chip para medir corrente e tensão ou de registradores internos no processador. Nessas situações, muitas vezes, é usado como métrica de consumo o TDP (*Thermal Design Power*) (INTEL, 2011) informado pelo fabricante. TDP, que em português, pode ser traduzido para "Energia Térmica de Projeto", serve para indicar a quantidade máxima de energia que um sistema de refrigeração deve dissipar. Ou seja, ao desenvolver o ventilador e o dissipador para uma CPU, as fabricantes devem criar produtos capazes de dissipar toda a energia gerada pelo processador em situações extremas.

Há ainda o ACP (*Average CPU Power*) (INTEL, 2011) que é bem recente, e só é encontrada nos processadores AMD. O ACP é diferente do TDP, o valor do ACP serve sim para fornecer um valor médio da potência gerada pelo processador em uso. Esse valor serve para que o próprio processador monitore o quanto de energia está sendo dissipada para que, quando necessário, um *overclock* possa ser realizado. A verificação é realizada através de uma comparação entre o ACP e o TDP, o que possibilita ao processador aumentar a frequência sem que seja preciso medir a temperatura.

São poucos os artigos que usam o TDP como métrica em seus textos como, por exemplo, (HUANG et al., 2008), pois não há entre os fabricantes de processadores um método padrão para

mensurar o TDP de seus produtos. O que leva os valores informados pelas fabricantes a serem alvo de grande controvérsia.

2.6 Considerações do capítulo

O uso de programação paralela é uma necessidade vital para alcançar um maior desempenho computacional e um melhor uso das tecnologias de processamento paralelo atuais. Mesmo com o melhor aproveitamento dos recursos computacionais que a programação paralela proporciona, o consumo de energia de tais plataformas computacionais aumenta exponencialmente o que torna esse fato um problema maior do que tentar aumentar o desempenho computacional da plataforma. Na tentativa de solucionar este problema foram criadas soluções como computação heterogênea e *cluster* baseados em processadores de baixo consumo energético. Baseado nessa última abordagem, no próximo capítulo serão definidas as plataformas de processadores ARM avaliadas neste trabalho e reconhecidas por seu baixo consumo.

3 PLATAFORMAS EXPERIMENTAIS

Neste capítulo, serão descritas as placas *Cubietruck* e *Jetson*. Serão definidas as características de um cluster *Beowulf*, bem como a construção de um cluster utilizando as placas *Cubietruck*, segundo a arquitetura *Beowulf*.

3.1 Placa Cubietruck

O SOC (*System-on-Chip*) presente na placa *Cubietruck*, mostrada na figura 3.1, possui o nome comercial de *Allwinner A20*. O processador ARM presente nesse SOC possui a arquitetura ARMv7 de 32 bits, também conhecido como ARM Cortex-A7. Esse processador conta com dois *cores*, cada *core* com *clock* de 1 GHz. A GPU que faz parte desse SOC é a *Mali-400 MP2* com *clock* de 350MHz. A placa conta com 2 GB de memória RAM DRR3, *clock* de 480MHz, e utiliza como armazenamento uma memória *flash NAND* de 8 GB. O preço praticado para a *Cubietruck* é de aproximadamente US$90 nos Estados Unidos.

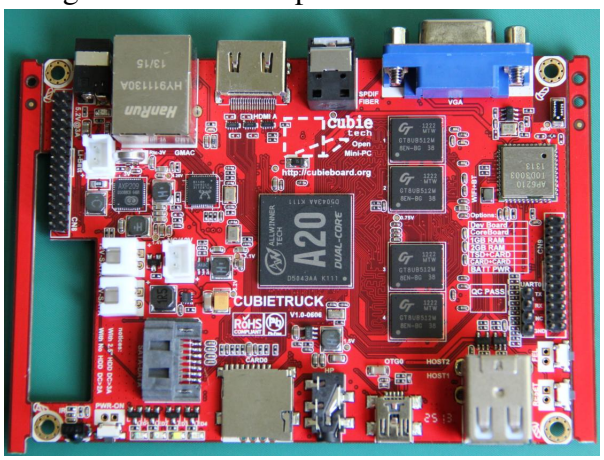
A placa possui diversas conexões, entre elas: rede *GigaEthernet* através do controlador RTL8211E, rede sem fio e bluetooth através do controlador BCM AP6210, HDMI, VGA, USB, interface SATA e GPIO. Também conta com 54 pinos de extensão (incluindo conexões como I2S, I2C, SPI, CVBS, LRADC, UART, PS2, PWM, TS/CSI, IRDA, LINEINandFMIN, TVIN). A *Cubietruck* possui o chip *AXP209* que monitora a tensão e corrente instantânea consumida por ela, auxiliando assim no monitoramento do consumo energético.

As principais distribuições suportadas pela *Cubietruck* são: *Cubieez*, *Lubuntu*, *Ferdora*, *Debian*, *Ubuntu*, *Android*. O sistema operacional pode ser instalado diretamente na memória de armazenamento da placa ou ser utilizado através de um cartão SD. A ordem de *boot* da placa é a seguinte: Cartão SD, memória de armazenamento interno, USB.

3.2 Placa Jetson

O SOC presente na placa *Jetson*, mostrada na figura 3.2 possui o nome comercial de *Tegra K1*. O processador ARM presente nesse SOC possui a arquitetura ARMv7-A de 32 bits também conhecido como ARM Cortex-A15. Esse processador conta com quatro *cores*, cada *core* com *clock* de 2,32 GHz. A GPU que faz parte desse SOC é a *Nvidia Kepler GK20a*, *clock* de 852MHz, e possui 192 *CUDA cores*. A placa conta com 2 GB de memória RAM do tipo

Figura 3.1: Visão superior da Cubietruck.



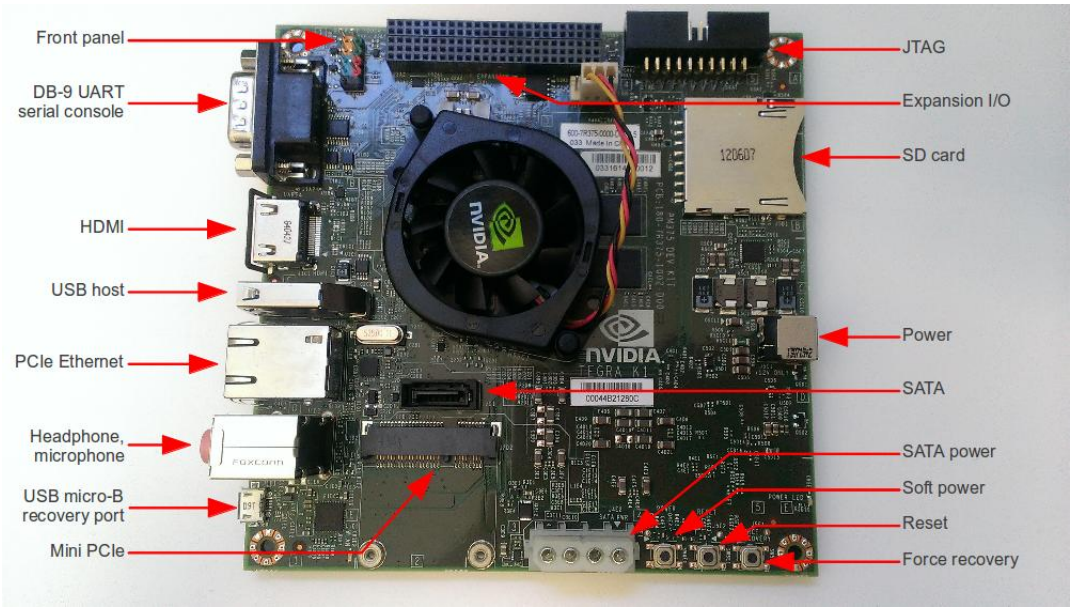
Fonte: (LEE, 2015)

DDR3L com *clock* de 933MHz e utiliza como armazenamento uma memória *fast eMMC* de 16 GB. O valor de potência instantânea estimada para ela é de 5W (EMAMI, 2015b) no pior caso, ou seja, com o uso dos quatro *cores*. A placa *Jetson* tem como preço praticado pela fabricante *Nvidia*, o valor de US$192.

A *Jetson* possui diversas conexões, entre elas: placa de rede *GigaEthernet* através do controlador RTL8111GS, HDMI, USB, interface SATA e mini-PCIe.

As principais distribuições suportadas pela *Jetson* são: *Linux4Tegra*, *Gentoo Linux*, *Busybox* e *Android*. O sistema operacional pode ser instalado diretamente na memória de armazenamento da placa ou ser utilizado através de um cartão SD. A ordem de *boot* da placa é a seguinte: Cartão SD, memória de armazenamento interno, USB, placa de rede.

Figura 3.2: Visão superior da Jetson.



Fonte: (EMAMI, 2015a)

Na tabela 3.1, estão destacadas as principais características dos processadores presentes nas placas *Cubietruck* (ARMv7) e *Jetson* (ARMv7-A).

Tabela 3.1: *Tabela comparativa ARMv7 X ARMv7-A*

	ARMv7	ARMv7-A
Processador	Allwinner A20 de 32 bits	Tegra K1 de 32 bits
Memória	DDR2/DDR3/DDR3L	DDR3L/LPDDR3
Cache	L1: 32 KB(Instruções)/32 KB(Dados) L2:256 KB (Compartilhado entre os dois *cores*)	L1: 32 KB(Instruções)/32 KB(Dados) L2:2 MB (Compartilhado entre os quatro *cores*)
GPU	Mali400 MP2	Kepler "GK20a"
Largura de banda da memória	2 Gb/s	17 Gb/s
Clock	1 GHz	2,32 GHz
Quantidade de *cores*	2	4

Fonte: Jorge Ximendes

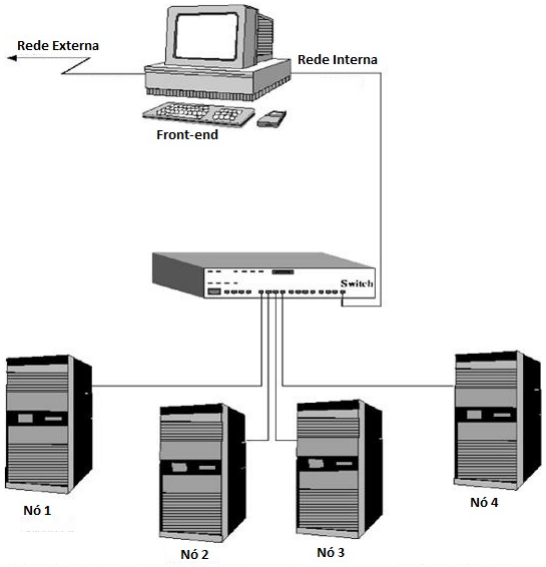
3.3 Cluster de processadores ARM

Uma arquitetura possível para ser utilizada na construção de um *cluster* é a arquitetura *Beowulf*, mostrada na figura 3.3. Essa arquitetura é composta por um computador que serve como nó servidor e um ou mais nós clientes. A conexão entre o nó servidor e os nós clientes utiliza uma rede isolada, isso é, sem conexão dos nós clientes com qualquer outra rede. Qualquer tipo de computador pode ser usado na construção dessa arquitetura. Como equipamento de rede utilizado na conexão entre o nó servidor e os nós clientes normalmente é utilizado um *switch*. O *software* utilizado é todo baseado em *software* livre e tem como componentes principais: sistema operacional *GNU/Linux*, servidor DHCP (*Dynamic Host Configuration Protocol*) e servidor de arquivos NFS (*Network File System*), assim como clientes DHCP e NFS. Os nós clientes não possuem dispositivos de entrada de dados, como *mouse* e teclado, nem saída de vídeo. O acesso ao nó clientes é feito via terminal utilizando a conexão SSH (*Secure Shell*).

3.4 Cubiecluster

Para auxiliar nos testes, foi construído um *cluster* de *Cubietrucks*, segundo a arquitetuta *Beowulf*. Esse *cluster* é composto por um total de oito placas *Cubietrcuks* e pode ser visto na

Figura 3.3: Esquema básico de uma rede local usada na construção de *cluster beowulf*.



Fonte: (BRAGA, 2015)

figura 3.4.

Figura 3.4: Visão frontal do cluster de Cubietrucks

Fonte: Jorge Ximendes

3.4.1 Nó servidor

O nó servidor funciona como um *front-end* entre a rede isolada e a rede externa, isto é, permite através dele acessar os nós clientes conectados na rede isolada. O nó servidor também funciona como um servidor DHCP distribuindo os endereços IP (*Internet Protocol*) para os nós clientes. O nó servidor também agrega um servidor de arquivos NFS permitindo a rápida distribuição de arquivos entre os nós clientes. Também auxilia na configuração dos nós clientes, pois permite acessar os nós clientes via terminal utilizando a conexão SSH. A distribuição Linux

utilizada no nó servidor é a Ubuntu 14.04, com *kernel* versão 3.13.0-53, e o *hardware* que o nó servidor possui esta descrito na tabela 3.2.

Tabela 3.2: *Hardware do nó servidor*

Processador	Core/Thread	Memória	Conexões	Armazenamento
Intel® Core™ i5-2537M CPU @ 1.40GHz	2/4	4 GB DDR3@1333MHz	Placa de rede Realtek RTL8111/8168/8411 PCI Express Gigabit Ethernet e Placa de rede sem fio Intel Corporation Centrino Advanced-N 6235	SSD 128 GB SanDisk SDSSDP128G

Fonte: Jorge Ximendes

3.4.2 Nós clientes

Os nós clientes são placas *Cubietrucks*. Esses nós clientes utilizam um cliente NFS para acessar o servidor NFS presente no nó servidor. Para a comunicação entre os nós clientes durante a execução das aplicações, que utilizam o *MPI*, utiliza-se a conexão SSH. O sistema operacional utilizado nos nós clientes é o *Debian wheezy* 3.3, com *kernel* versão 3.4.106, e está armazenado em um cartão SD. A imagem desse sistema operacional foi obtido da Internet (PEčOVNIK, 2013) e utilizada sem nenhuma modificação. Os nós clientes tiveram todas as conexões desabilitadas, com exceção da conexão de rede, com o auxílio de ferramentas presentes no sistema operacional.

3.4.3 Interconexão

A interconexão entre os nó servidor e os nós clientes é baseada em um *switch*. Na tabela 3.3 estão descritas as especificações do *switch* utilizado.

3.5 Considerações do capítulo

Neste capítulo, foram descritos as plataformas experimentais *Cubietruck* e *Jetson* e apresentado a definição de um *cluster Beowulf* e suas características. Ainda, foi mostrado a construção de *cluster* de *Cubietruck*, segundo a arquitetura *Beowulf*. No capítulo seguinte será mostrado os resultados da avaliação dessas plataformas.

Tabela 3.3: *Especificações do Switch 4250T*

Fabricante	Número de portas / velocidade	Capacidade de encaminhamento de pacotes por segundo	Tamanho da tabela de endereços MAC	Padrões	Protoclos de gerenciamento remoto	Modos de comunicação
3Com	48 Fast Ethernet (10BASE-T/100BASE-TX), 2 Gigabit Ethernet (10BASE-T/100BASE-TX /1000BASE-T)	10,1 milhões	8K	IEEE 802.1Q, IEEE 802.1p, IEEE 802.1w	RMON, SNMP	full-duplex, half-duplex

Fonte: Jorge Ximendes

4 AVALIAÇÃO EXPERIMENTAL

O objetivo deste trabalho é medir o desempenho computacional e o consumo de energia de processadores ARM e assim obter sua eficiência energética. Para medir o desempenho computacional será utilizado o benchmark NPB.

4.1 Benchmark, metodologia e critérios de avaliação

O NAS Parallel Benchmarks (NPB) (NASA, 2014) é um conjunto de *benchmarks*, derivados da dinâmica de fluidos computacional, bem reconhecido, para avaliar arquiteturas multicores atuais e emergentes. Neste trabalho serão utilizados tanto sua versão *OpenMP* quanto a versão *MPI*. O NPB é composto por oito aplicações diferentes, a saber:

- CG (Gradiente conjugado): É usado para calcular uma aproximação para o menor autovalor de uma grande, escassa, simétrica e positiva matriz definida. Esse *kernel* é típico de cálculos de grade não estruturados. O objetivo é testar o acesso irregular à memória e de comunicação.
- EP (Embaraçosamente paralelo): Fornece uma estimativa dos limites superiores alcançáveis para desempenho de ponto flutuante. Nesse caso não há comunicação significativa entre processadores sendo uma aplicação computacionalmente intensiva (*CPU bound*).
- FT (Transformada rápida de Fourier): Uma solução parcial de equação diferencial usando Transformada rápida de Fourier. Esse *kernel* executa a essência de muitos códigos espectrais. É um teste de desempenho de comunicação. Testa comunicação todos para todos.
- IS (Ordenação de inteiros): Esse *kernel* executa uma operação de classificação que é importante no método de códigos de partícula. Testa tanto a unidade de inteiros do processador quanto o desempenho de comunicação, e também o acesso randômico à memória.
- MG (Multigrid): Um *kernel* multigrid simplificado. Exige comunicação altamente estruturada e executa testes de comunicação de dados. Testa o acesso intensivo à memória e a comunicação de curta e longa distância.
- BT (Bloco Tri-diagonal) - Resolve três conjuntos de sistemas de equações desacoplados, primeiro em x, em seguida em y, e finalmente em z usando um solucionador de bloco Tri-diagonal.
- LU (Solução de equações lineares): Solucionador de Lower-Upper Gauss-Seidel.
- SP (Escalar penta-diagonal) - Resolve três conjuntos de sistemas de equações desaco-

plados, primeiro em x, em seguida em y, e finalmente em z usando um solucionador de escalar penta-diagonal.

O NPB é fornecido em classes para auxiliar na execução dos testes em diversas arquiteturas e hardwares com capacidades de cálculo e comunicação diferentes. Para arquiteturas simples, usa-se classes menores, como a S ou W. Para arquiteturas, ou hardware, intermediários, usa-se as classes que exigem mais, como A, B ou C. E para arquiteturas, ou hardwares, potentes, usa-se as classes mais pesadas, como D, E ou F. As classes possíveis para o NPB são:

- Classe S: pequeno para fins de teste rápido;
- Classe W: compatível com o poder de processamento de uma estação de trabalho;
- Classes A, B, C: problemas de teste de tamanho padrão, a necessidade computacional quadriplica de uma classe para a próxima;
- Classes D, E, F: grandes problemas de teste, a necessidade computacional aumenta dezesseis vezes de uma classe para a próxima.

Os testes executando as aplicações da suíte de aplicações do NPB serão efetuados utilizando a classe B, pois esta apresenta um tamanho suficientemente grande para testar o desempenho dos componentes da *cubietruck*.

Na versão *OpenMP* será variado o número de *threads* entre uma, duas e quatro. As aplicações do NPB serão executados com esses valores, pois será possível verificar o ganho de desempenho ao acrescentar as *threads* a execução delas. A execução com quatro *threads* também terá a função de verificar se o aumento de *threads* acima do número de *cores* afetará o desempenho.

Na versão *MPI* será variado o número de *processos* entre um e dezesseis. Partindo do mesmo princípio utilizado nos testes com *OpenMP*, os com um processo serão utilizados como referência, e os resultados com mais processos serão utilizados para verificar o aumento de desempenho das aplicações do NPB. Serão utilizados até dezesseis processos, pois este é o número máximo de *cores* de processamento presentes no cluster de placas *cubietruck*.

Serão efetuados tantos testes das aplicações do NPB quantos necessários para validar estatisticamente os resultados obtidos de acordo com um intervalo de confiança de 95%.

Os critérios a serem usados para avaliar o cluster são:

- Eficiência energética: O desempenho computacional dividido pelo consumo de energia gasto (medido em Mflops por Watts).
- Progressão do desempenho computacional: Para verificar se cada *thread*, ou processo, adicionada para executar os testes há um aumento de desempenho proporcional a essa

adição.

- Consumo energético de cada aplicação do NPB.
- Desempenho computacional de cada aplicação do NPB.

4.2 Consumo de potência na cubietruck

Como viso na seção 3.1, a placa *Cubietruck* apresenta um *chip* de nome *AXP209* que faz medições de tensão e corrente instantâneos da placa. Essas informações são acessadas através do comando GET no barramento I2C, presente na placa, nos endereços de E/S 57_{16} e 56_{16}, para obter a tensão instantânea, e nos endereços 59_{16} e 58_{16}, para obter a corrente instantânea. Para facilitar a obtenção dos valores monitorados pelo *chip* utilizou-se um *script* em *shell* para ler esses valores e facilitar a sua manipulação.

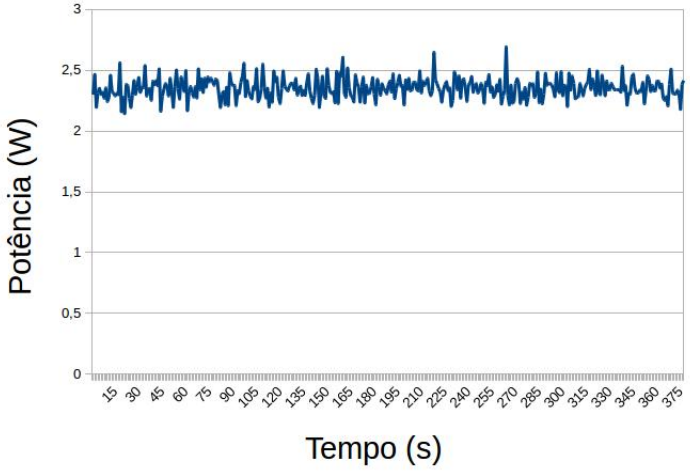
Foram efetuados experimentos para verificar a precisão dos valores obtidos pelo referido *chip* e então foi detectado uma variação muito grande dos valores obtidos. Desconfiou-se da influência e perturbação do *script* que lia esses valores por haver concorrência desse processo com as aplicações do NPB. Para validar se esta hipótese era correta usou-se um *core* para executar as aplicações do NPB e o outro *core* ficou responsável pela execução do *script* e assim os valores medidos de tensão e corrente estabilizaram. Através deste experimento ficou comprovado que para obter valores estáveis e precisos deve-se usar somente um *core* para processamento, caso contrário, os valores não são confiáveis.

As figuras 4.1 e 4.2 foram resultados da execução da aplicação EP, em sua versão *OpenMP*, com uma, e depois, duas *threads* em paralelo com a execução do *script* lendo os valores do *chip* em intervalos de um segundo. Comparando as duas figuras fica claro a interferência e a estabilidade dos valores lidos no chip *AXP209*.

Através da execução da aplicação EP, em sua versão *OpenMP*, com uma *thread* em paralelo, com a execução do *script* lendo os valores do *chip* em intervalos de um segundo, obteve-se como valor medido de potência consumida aproximadamente 2,4 W. Este valor foi o mesmo obtido em testes com as outras aplicações do NPB, tanto em versão *OpenMP*, quanto na versão *MPI*.

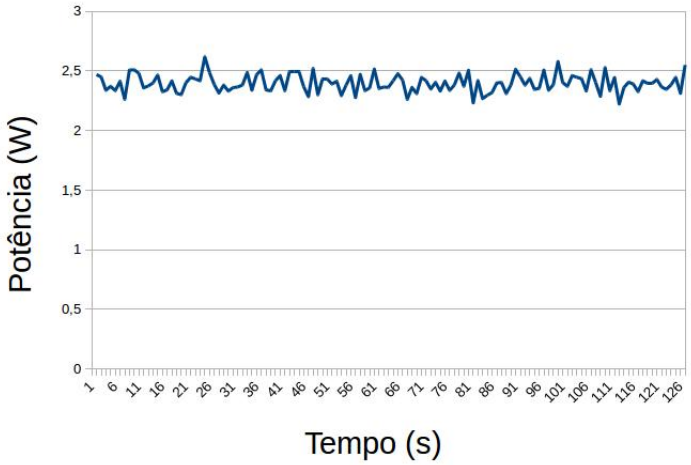
Como pretendia-se usar os dois *cores* para processamento, e não havia como utilizar o *chip* presente na placa, foi necessário fazer um experimento com um medidor externo. Este experimento foi executado no Departamento de Elétrica com uma fonte de tensão controlada que informava o valor instantâneo de corrente consumida, onde a placa *cubietruck* foi conectada

Figura 4.1: Potência medida com benchmark EP com 1 thread



Fonte: Jorge Ximendes

Figura 4.2: Potência medida com benchmark EP com 2 threads



Fonte: Jorge Ximendes

enquanto ela executava a aplicação EP com o uso de duas *threads* em sua versão *OpenMP*. Neste experimento, o valor de corrente foi de aproximadamente 650mA, resultando, assim, em um valor de potência estimado em 3,25 W. Este valor é muito próximo ao valor obtido nos experimentos realizados por (SMITH; HAMILTON, 2015).

4.3 Cenários de teste

Para avaliar a eficiência energética e o desempenho computacional do processador ARM, elaborou-se dois cenários de teste, em três configurações diferentes. O primeiro cenário consiste na execução do *OpenMP* em uma placa *cubietruck*. Nesse cenário, a configuração empregada foi usar uma, duas e quatro *threads*. O segundo cenário é a execução da versão *MPI* do NPB

com duas configurações distintas: um processo *MPI* por nó (placa *Cubietruck*) e dois processos *MPI* por nó.

4.3.1 Cenário I: execução do OpenMP

O objetivo deste teste é verificar se a utilização de mais de um *core*, traz uma eficiência energética proporcional a adição desse *core*. Para isso, cada *benckmark* será executado e será observado o tempo de execução até o seu término e comparado com a energia consumida. Essa comparação é baseada nas métricas *time-to-solution* e *energy-to-solution* introduzido por (PADOIN et al., 2012).

Primeiramente, será executada toda a suíte de aplicações do NPB com uma uma (1) *thread*. Posteriormente, será executada a suíte de aplicações com duas (2) *threads* e, finalmente, será executada a suíte de aplicações com quatro (4) *threads*. Cada aplicação do NPB fornece o tempo de execução e os MFlops por segundo da sua execução. Esses dados são apresentados na tabela 4.1.

Todas as aplicações do NPB foram executadas 32 vezes. Assim, os valores de tempo de execução apresentados na tabela 4.1, são as médias aritméticas dessas execuções. O desvio padrão foi próximo de 1% da média calculada.

A energia apresentada na tabela 4.1 foi obtida multiplicando o tempo de execução pelo valor médio da potência instantânea. No caso de uma *thread*, esse valor corresponde ao obtido pela leitura de tensão e corrente diretamente na placa (2,4 W), enquanto que para duas ou quatro *threads* empregou-se o valor estimado (3,25 W), valores apresentados na linha potência na tabela.

Os valores de eficiência energética, ou MFlops/W, apresentados na tabela 4.1, foram obtidos dividindo os MFlops por segundo pela potência medida quando executando uma *thread*, e pela potência estimada quando executando duas ou quatros *threads*.

Analisando a tabela 4.1, é possível ver que na maioria dos casos testados o uso de dois *cores* resultou em um tempo de execução praticamente a metade do caso com um só *core*, mas o consumo de energia não apresentou essa mesma proporção. Isso se reflete na eficiência energética (MFlops/Watt). Portanto, a adição de um *core* ao processamento provocou um gasto menor de energia, mas não foi proporcional a redução do tempo de execução. Isso porque dois *cores* funcionando consomem mais potência (3,25 W estimados) que apenas um *core* (2,4 W).

Cabe ressaltar que o *benchmark* FT foi o que apresentou menor incremento de desempenho. Isso é consequência do fato que no FT não há uma exploração do poder computacional

Tabela 4.1: *Resultados OpenMP*

Teste	Número de threads	1	2	4
	Potência(W)	2,4	3,25	3,25
CG	Tempo(s)	4819,76	2465,97	2468,18
	Energia(J)	11567,43	8014,41	8021,60
	MFlops/s	11,35	22,18	22,16
	MFlops/Watts	4,72	6,82	6,820
EP	Tempo(s)	602,47	304,55	306,39
	Energia(J)	1445,93	989,79	995,76
	MFlops/s	3,56	7,05	7,01
	MFlops/Watts	1,48	2,16	2,15
FT	Tempo(s)	1186,57	1005,29	1024,08
	Energia(J)	2847,78	3267,21	3328,26
	MFlops/s	77,58	91,57	89,89
	MFlops/Watts	32,32	28,17	27,65
IS	Tempo(s)	32,90	20,38	20,62
	Energia(J)	78,97	66,26	67,04
	MFlops/s	10,19	16,45	16,26
	MFlops/Watts	4,24	5,06	5,00
MG	Tempo(s)	210,73	115,99	117,26
	Energia(J)	505,75	376,98	381,11
	MFlops/s	92,35	167,77	165,96
	MFlops/Watts	38,48	51,62	51,06
LU	Tempo(s)	4245,30	2337,05	2388,78
	Energia(J)	10188,73	7595,42	7763,56
	MFlops/s	117,50	213,44	208,84
	MFlops/Watts	48,95	65,67	64,25
BT	Tempo(s)	4358,80	2530,06	2603,54
	Energia(J)	10461,13	8222,72	8461,52
	MFlops/s	161,04	277,59	269,74
	MFlops/Watts	67,10	85,41	82,99
SP	Tempo(s)	5326,94	2970,66	2989,84
	Energia(J)	12784,67	9654,67	9716,99
	MFlops/s	66,64	119,50	118,73
	MFlops/Watts	27,76	36,77	36,53

Fonte: Jorge Ximendes

dos *cores* devido a necessidade de sincronização imposta pela comunicação.

Por fim, observa-se que com a execução das aplicações com quatro *threads* houve pequena variação do desempenho obtido em relação a execução das aplicações com duas threads. Este resultado era esperado, pois a *cubietruck* conta com somente 2 *cores* no processador. Caso a *Cubietruck* possui-se a tecnologia *Hyperthreading*, como presente nos processadores da fabricante *Intel*, essa execução poderia ter aumentado o desempenho em vez de ter o comportamento apresentado.

4.3.2 Cenário II: execução do OpenMPI

O objetivo deste cenário de teste é verificar a influência da adição de nós de processamento na eficiência energética. Outro ponto a analisar é se o uso de mais de um *core* no mesmo nó apresenta um comportamento semelhante ao observado no *OpenMP*.

Para isso, os *benchmarks* foram executados considerando duas distribuições diferentes de processos nas placas *Cubietruck*. Na primeira distribuição, cada placa recebe no máximo um processo *MPI* enquanto que na segunda distribuição, uma placa recebe mais de um processo *MPI*. Isso faz com que se tenha a situação em que apenas um *core* por placa seja usado para cálculo ou que dois *cores* sejam utilizados.

Um ponto a ressaltar é o número de processos por *benchmarks* do NPB. Enquanto que os *benchmarks* CG, EP, FT, IS, LU e MG executam com uma quantidade de processos em potência de dois (1,2,4,8 e 16), os *benchmarks* BT e SP executam com uma quantidade de processos em quadrado perfeito (1,4 e 16).

Inicialmente, considerando a primeira distribuição, onde há apenas um processo *MPI* por placa. Os valores fornecidos na tabela 4.2 são a média de 32 execuções. Ainda, nesse caso, por haver apenas um processo por placa, o valor de potência instantâneo é aquele medido na placa com o auxílio do *script* (2,4 W). Essa potência é apresentada na linha potência na tabela, considerando o número de placas utilizadas para cada teste. A energia fornecida nessa tabela corresponde ao total das placas usadas no teste, por exemplo, no caso de dois processos *MPI*, o valor fornecido é igual ao consumo de duas placas.

Analisando os valores da tabela 4.2, é possível notar que, a exceção do IS, a adição de nós de processamento reduziu o tempo de execução. Em relação ao consumo de energia, é possível ver que no *benchmark* EP ficou constante porque, ao adicionar nós, se obteve um ganho de tempo proporcional a esta adição, mas, ao mesmo tempo, aumentou o consumo de energia na mesma proporção. Isso é um resultado relativamente esperado, já que o *benchmark* EP é basicamente uma aplicação que faz uso intensivo do processador. Ao se dobrar a quantidade de processadores, se tem uma execução pela metade do tempo. Porém, como cada processador adicionado, representa um aumento de consumo, a eficiência energética se manteve estável.

Para os *benchmarks* GC, BT, LU, MG, SP e FT o ganho de tempo não foi acompanhado por um ganho em eficiência energética.

Considerando agora a segunda distribuição, onde há dois processos *MPI* por placa. A tabela 4.3 apresenta os resultados obtidos nessa distribuição. Os valores presentes na tabela são o resultado da média calculada sobre 32 execuções. Como há dois processos por placa,

Tabela 4.2: *Resultados MPI com 1 processo por placa Cubietruck*

Teste	Número de processos	1	2	4	8
	Potência(W)	2,4	4,8	9,6	19,2
CG	Tempo(s)	5019,26	1843,11	1066,17	439,27
	Energia(J)	12046,23	8846,96	10235,25	8434,12
	MFlops/s	10,90	29,68	51,31	124,54
	MFlops/Watts	4,541	6,18	5,34	6,48
EP	Tempo(s)	595,60	299,40	150,07	75,36
	Energia(J)	1429,44	1437,13	1440,71	1447,01
	MFlops/s	3,60	7,17	14,31	28,49
	MFlops/Watts	1,50	1,49	1,49	1,48
FT	Tempo(s)	912,08	735,99	499,89	269,57
	Energia(J)	2189,01	3532,76	4798,99	5175,88
	MFlops/s	100,92	125,07	184,14	341,47
	MFlops/Watts	42,054	26,05	19,18	17,78
IS	Tempo(s)	28,89	48,62	40,42	22,78
	Energia(J)	69,33	233,37	388,11	437,50
	MFlops/s	11,61	6,90	8,30	14,72
	MFlops/Watts	4,83	1,43	0,86	0,76
MG	Tempo(s)	206,95	116,19	67,98	44,76
	Energia(J)	496,69	557,72	652,64	859,41
	MFlops/s	92,38	167,49	286,29	434,95
	MFlops/Watts	38,49	34,89	29,82	22,65
LU	Tempo(s)	4755,17	2468,46	1266,04	689,02
	Energia(J)	11412,41	11848,63	12154,05	13229,33
	MFlops/s	104,90	202,08	394,00	724,63
	MFlops/Watts	43,70	42,10	41,042	37,74
BT	Tempo(s)	4992,56	x	1290,07	x
	Energia(J)	11982,15	x	12384,73	x
	MFlops/s	140,64	x	544,29	x
	MFlops/Watts	58,60	x	56,69	x
SP	Tempo(s)	6098,45	x	1635,08	x
	Energia(J)	14636,29	x	15696,82	x
	MFlops/s	58,21	x	217,12	x
	MFlops/Watts	24,25	x	45,23	x

Fonte: Jorge Ximendes

o valor de potência instantânea usada é aquele estimado (3,25 W), obtido pelo experimento feito no Departamento de Elétrica. Essa potência é apresentada na linha potência na tabela, considerando o número de placas utilizadas para cada teste. Como na primeira distribuição, o valor de energia fornecido na tabela corresponde ao total de placas usadas no teste, por exemplo, no caso de oito processos *MPI*, o valor apresentado é igual ao consumo de quatro placas.

Analisando os valores apresentados na tabela 4.3, é possível visualizar um comportamento parecido com o visto na primeira distribuição. Ou seja, a adição de nós de processamento

Tabela 4.3: *Resultados MPI com 2 processo por placa Cubietruck*

Teste	Número de processos	2	4	8	16
	Potência(W)	3,25	6,5	13	26
CG	Tempo(s)	2526,58	1378,96	682,810	578,99
	Energia(J)	8211,40	8963,25	8876,53	15053,83
	MFlops/s	21,65	39,67	80,12	94,49
	MFlops/Watts	6,66	6,10	6,16	3,63
EP	Tempo(s)	301,55	151,37	75,48	38,04
	Energia(J)	980,05	983,94	981,28	989,27
	MFlops/s	7,12	14,18	28,45	56,44
	MFlops/Watts	2,19	2,18	2,18	2,17
FT	Tempo(s)	823,11	705,77	459,46	276,22
	Energia(J)	2675,10	4587,55	5973,06	7181,74
	MFlops/s	111,83	130,43	200,35	333,26
	MFlops/Watts	34,41	20,06	15,41	12,81
IS	Tempo(s)	21,54	44,47	35,57	28,63
	Energia(J)	70,01	289,09	462,44	744,43
	MFlops/s	15,57	7,54	9,43	11,72
	MFlops/Watts	4,79	1,16	0,72	0,45
MG	Tempo(s)	119,39	74,63	50,21	43,73
	Energia(J)	388,04	485,11	652,82	1137,04
	MFlops/s	165,05	260,78	387,69	445,21
	MFlops/Watts	50,78	40,12	29,82	17,12
LU	Tempo(s)	2622,34	1384,48	748,09	439,03
	Energia(J)	8522,62	8999,16	9725,19	11414,90
	MFlops/s	190,22	360,30	666,8	1136,19
	MFlops/Watts	58,52	55,43	51,29	43,69
BT	Tempo(s)	x	1364,28	x	489,39
	Energia(J)	x	8867,82	x	12724,29
	MFlops/s	x	514,68	x	1434,80
	MFlops/Watts	x	79,18	x	55,18
SP	Tempo(s)	x	1738,12	x	748,20
	Energia(J)	x	11297,83	x	19453,36
	MFlops/s	x	198,58	x	474,5
	MFlops/Watts	x	30,55	x	18,25

Fonte: Jorge Ximendes

reduz o tempo de execução, com a exceção do IS.

O *benchmark* EP teve um comportamento igual ao visto na primeira distribuição. Ou seja, a eficiência energética manteve-se constante, pois com o aumento de nós de processamento, o tempo de execução caiu proporcionalmente, mas o consumo de energia também subiu proporcionalmente.

Os comportamentos dos *benchmarks* CG, BT, LU, MG, SP e FT foram também parecidos com os obtidos na primeira distribuição. O ganho de desempenho não foi acompanhado

por uma melhora na eficiência energética.

Comparando as tabelas 4.2 e 4.3, nos casos de teste efetuados, usar dois processos *MPI* por placa, não demonstrou obter um desempenho em tempo de execução melhor que usar um processo *MPI* por placa. Entretanto, ao empregar a energia consumida como métrica de comparação, usar dois processos *MPI*, por placa consumiu menos energia do que quando utilizando um processo *MPI* por placa. Como resultado dessas conclusões, a eficiência energética ao utilizar dois processos *MPI* por placa é melhor do que quando utilizando um processo *MPI* por placa.

4.4 Comparação com a placa Jetson

O objetivo é fazer uma comparação para saber qual a melhor escolha para montar um cluster com processadores de baixo consumo: utilizar a placa *Jetson* ou a placa *Cubietruck*.

Para realizar esta comparação foram utilizados os *benchmarks* EP que testa somente o processador, IS que tem comportamento misto, isso é, testa tanto comunicação de dados quanto processamento, e o FT que faz uso de comunicação intensiva de dados. Foram utilizados tanto a versão *OpenMP* quanto a versão *MPI* desses *benchmarks*. A versão *OpenMP* foi executada com uma, duas e quatro *threads*. A versão *MPI* foi executada com dois, quatro, oito e dezesseis processos.

Para o cálculo de energia consumida foi utilizada para a *Cubietruck* o valor medido de 2,4W para o uso de um *core* e o valor estimado de 3,25W para o uso de dois *cores*. No caso da *Jetson* foi utilizado o valor estimado de 5W (EMAMI, 2015b) para todos os casos, pois não foi possível medir o consumo de potência, ou encontrar referências mais precisas sobre o consumo. Esses valores estão apresentado na linha potência nas tabelas, considerando o número de placas utilizadas para cada teste.

Tabela 4.4: *Resultados OpenMP Cubietruck*

Teste	Número de threads	1	2	4
	Potência(W)	2,4	3,25	3,25
EP	Tempo(s)	602,47	304,55	306,39
	Energia(J)	1445,93	989,79	995,76
IS	Tempo(s)	32,90	20,38	20,62
	Energia(J)	78,97	66,26	67,04
FT	Tempo(s)	1186,57	1005,29	1024,08
	Energia(J)	2847,78	3267,21	3328,26

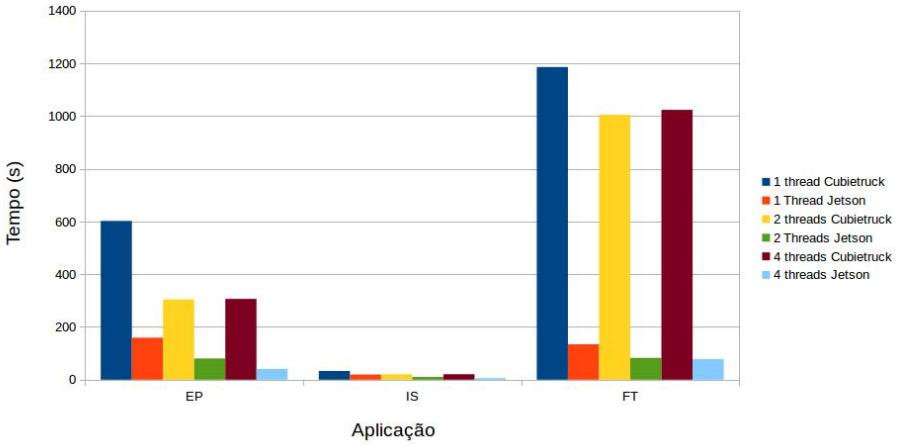
Fonte: Jorge Ximendes

Tabela 4.5: *Resultados OpenMP Jetson*

Teste	Número de threads	1	2	4
	Potência(W)	5	5	5
EP	Tempo(s)	159,45	80,43	40,78
	Energia(J)	797,25	402,15	203,9
IS	Tempo(s)	19,37	10,24	5,79
	Energia(J)	96,85	51,2	28,95
FT	Tempo(s)	134,33	82,76	78,25
	Energia(J)	671,65	413,8	391,25

Fonte: Jorge Ximendes

Figura 4.3: Comparação Jetson x Cubietruck OpenMP (Tempo(s))



Fonte: Jorge Ximendes

Pelas tabelas 4.4 e 4.5 e pelas figuras 4.3 e 4.4, a *Jetson* mostrou melhor desempenho melhor em tempo de execução nos testes efetuados usando *OpenMP*. Mesmo ela tendo tendo um consumo de potência maior, ainda assim ela consumiu menos energia para executar os testes.

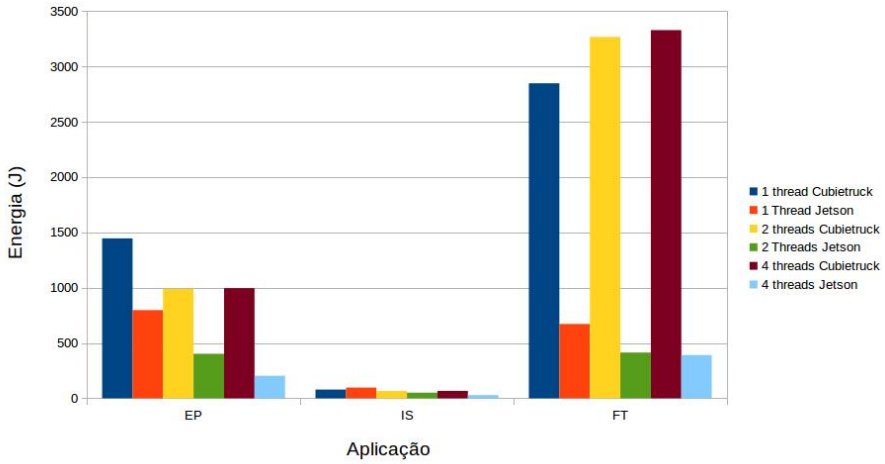
Tabela 4.6: *Resultados MPI Cubietruck*

Teste	Número de processos	2	4	8	16
	Potência(W)	4,8	9,6	19,2	26
EP	Tempo(s)	299,40	150,07	75,36	38,04
	Energia(J)	1437,13	1440,71	1447,01	989,27
IS	Tempo(s)	48,62	40,42	22,78	28,63
	Energia(J)	233,37	388,11	437,50	744,43
FT	Tempo(s)	735,99	499,89	269,57	276,22
	Energia(J)	3532,76	4798,99	5175,88	7181,74

Fonte: Jorge Ximendes

Pelas tabelas 4.6 e 4.7 e pelas figuras 4.5 e 4.6, a *Jetson* mostrou menor tempo de execução nos testes efetuados com o *MPI*. Mesmo a *Jetson* tendo tendo um consumo de potência

Figura 4.4: Comparação Jetson x Cubietruck OpenMP (Energia(J))



Fonte: Jorge Ximendes

Tabela 4.7: *Resultados MPI Jetson*

Teste	Número de processos	2	4	8	16
	Potência(W)	5	5	10	20
EP	Tempo(s)	82,27	41,25	20,92	11,11
	Energia(J)	822,7	825	418,2	222,2
IS	Tempo(s)	13,99	8,20	6,65	5,87
	Energia(J)	139,9	164	133	117,4
FT	Tempo(s)	110,22	74,36	53,85	47,80
	Energia(J)	1102,2	1487,2	1077	956

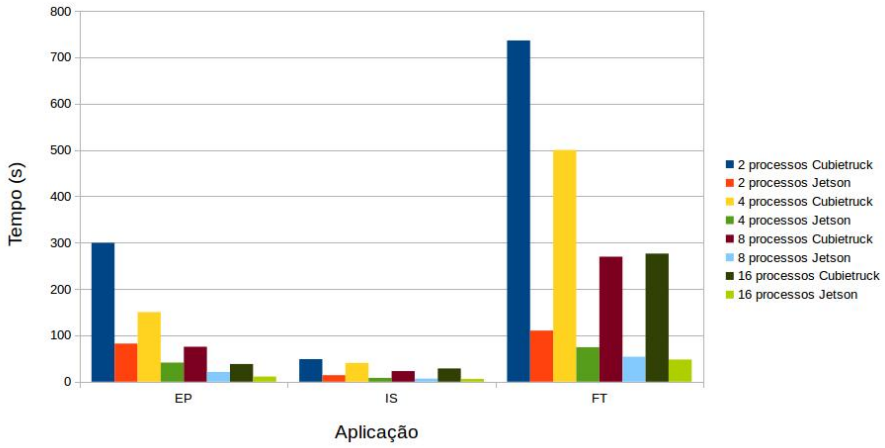
Fonte: Jorge Ximendes

maior, ainda assim ela consumiu menos energia para executar os testes.

Com base nesses resultados, como a *Jetson* apresentou o melhor desempenho tanto em tempo de execução, quanto em consumo de energia, conclui-se que as placas *Jetson* são mais adequadas que as *Cubietrucks* para comporem nós de um cluster baseado em processadores ARM.

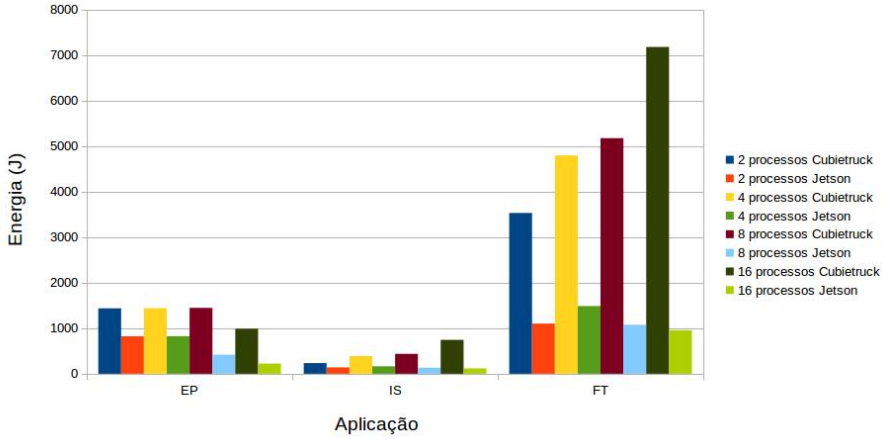
Cabe ressaltar que a diferença de arquiteturas entre o processador da *Jetson* (ARMv7-A) e o processador da *Cubietruck* (ARMv7) é um fator importante para os melhores resultados apresentados pela *Jetson*. A arquitetura do processador da *Jetson* em relação ao processador da *Cubietruck*, possui várias otimizações e novos recursos (MORIMOTO, 2012), tais como, três unidades de execução, execução fora de ordem, *pipeline* com profundidade de 15 estágios para ponto fixo e entre 17 a 25 estágios para ponto flutuante, melhorias no *branch-prediction* e na execução especulativa. Esse processador conta ainda com um controlador de memória de 64 bits com largura de banda de 17Gb/s (KLUG; SHIMPI, 2014), enquanto o processador da *Cubietruck* possui um controlador de memória de 32 bits com largura de banda de 2 Gb/s

Figura 4.5: Comparação Jetson x Cubietruck MPI (Tempo(s))



Fonte: Jorge Ximendes

Figura 4.6: Comparação Jetson x Cubietruck MPI (Energia(J))



Fonte: Jorge Ximendes

(OPTIMIZING. . . , 2013). Além dessas diferenças de arquitetura, a *Jetson* conta com um *clock* 130% maior que o *clock* da *Cubietruck* e o dobro cores no processador que a *Cubietruck*. A *Jetson* tem todas essas vantagens, mas com um consumo de potência apenas 35% maior que a *Cubietrcuk*, assim, a *Jetson* compensa, pois os ganhos em desempenho são muito maiores do que o aumento de consumo de potência.

Depois de considerar os aspectos técnicos, o custo para obter dezesseis *cores* com a *Jetson* é de aproximadamente US$ 768, pois é necessário o uso de quatro placas, e o custo para obter dezesseis *cores* com a *Cubietruck* é de aproximadamente US$ 720, pois é necessário o uso de oito placas. Assim, considerando a diferença de apenas US$ 48 entre o custo da *Jetson* e *Cubietruck*, compensa a compra da *Jetson* pelo desempenho entregue.

5 CONCLUSÃO

Uma análise dos atuais sistemas de HPC mostra que 40-60% do consumo de energia pode ser atribuída aos nós de computação (processadores e memórias). Consequentemente, o maior retorno pode ser esperado na melhoria da eficiência energética dos nós de computação. Assim, em 2008 especialistas alertaram que o consumo de energia aceitável para chegar ao exascale seria de 20 MW. Ao consider essa recomendação, a eficiência energética dos futuros sistemas de exascale tem um limite de 50 Gflops/W. Se sistemas exascale fossem construídos com a tecnologia atualmente empregada, seu consumo de potência alcançariam a ordem de GigaWatt. Para tentar minimizar esse problema, foram criadas abordagens como o uso de computação heterogênea, o uso de processadores gráficos para computação de propósito geral e, por exemplo, o Projecto Mont-Blanc (PROJECT.EU, 2015) que foi um dos primeiros a introduzir a idéia de um supercomputador baseado em ARM.

Assim, o objetivo deste trabalho foi avaliar se processadores ARM, disponíveis no mercado, em placas como *Raspberry PI*, *Cubietruck* e *Jetson* são alternativas para construir cluster de baixo custo e baixo consumo para processamento paralelo.

Essa avaliação foi feita com base no consumo energético que é obtido, no caso da *Cubietruck*, através da leitura das informações do *chip AXV209* com o uso de um *core* para processamento, e através do valor estimado para o uso de dois *core* para processamento. No caso da *Jetson*, foi usado o valor estimado para o pior caso como parâmetro.

Para avaliar o consumo energético foram usadas as aplicações do NPB em duas placas com processadores ARM: *Cubietruck* e *Jetson*. Foram utilizadas tanto a versão *OpenMP* quanto a versão *MPI* das aplicações do NPB. O número de *threads* ficou limitado a quantidade de *cores* dos processadores presentes nas placas, assim como o número de processos ficou limitado a quantidade de placas disponíveis.

Com os resultados obtidos tanto em tempo de execução quanto de consumo de energia, a *Jetson* apresentou um desempenho melhor nos quesitos avaliados em comparação a *Cubietruck*. Assim, a *Jetson* mostrou ser a melhor escolha para construção de um cluster de processadores ARM.

As placas *Jetson* e *Cubietruck* mostraram melhor escalabilidade que em um trabalho anterior (PADOIN et al., 2013) que foi feito usando placas *Panda Board*. O principal motivo dessa escalabilidade foi o uso de *chip Ethernet* nas placas *Jetson* e *Cubietruck*, enquanto na placas *Panda Board* é utilizado um *chip Ethernet* via barramento USB.

Um problema encontrado na realização deste trabalho foi a falta de mecanismos e equi-

pamentos para medir de forma adequada o consumo de energia. A intenção inicial era empregar o próprio *chip* exitente na *Cubietruck* para esta finalidade. Entretanto, tal abordagem se mostrou deficiente, porque ao realizar consultas ao *chip*, gerava-se uma interferência no sistema e, consequentemente uma perda de precisão quando os dois *cores* eram usados para processamento. Isso foi demonstrado ao se realizar medidas experimentais com apenas um *core* executando a aplicação e com dois *cores*. Portanto, o emprego dessa técnica para medir o consumo em placas *Cubietruck* não é eficaz e deve-se, em futuros trabalhos, buscar alternativas diferentes como, por exemplo, wattímetros externos.

O principal resultado deste trabalho foi identificar que duas versões distintas de processadores ARM apresentam desempenhos diferentes com um custo energético não proporcional a esse ganho. A placa com o processador ARMv7-A (*Jetson*) tem um consumo 35% superior a placa com o processador ARMv7 (*Cubietruck*), com ganhos que variam de duas vezes, no pior caso, a até dez vezes, no melhor caso. No entanto, seria necessário realizar mais testes para reforçar essa conclusão. Assim, sujere-se, como trabalhos futuros, a realização de testes com outros processadores ARM, com mais nós de processamento e comparar os processadores ARM com outra arquiteturas, como por exemplo, *Xeon Phi*. Outra possibilidade é utilizar o *benchmark Linpack*. Nesse caso, tem-se como parâmetro de comparação as máquinas ranqueadas no *Top Green500*.

REFERÊNCIAS

ANDREWS, G. R. **Foundations of parallel and distributed programming**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1999.

BERGMAN, K. et al. Exascale computing study: Technology challenges in achieving exascale systems. **Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep**, v. 15, 2008.

BRAGA, A. A. C. **Technical aspects of beowulf cluster construction**. 2015. Disponível em: <http://www.scielo.br/scielo.php?pid=S0100-40422003000300018&script=sci_arttext>. Acesso em: 18 maio 2015.

BRAGA, N. C. **Transformadores e fator de potência (EL104)**. 2014. Disponível em: <http://www.newtoncbraga.com.br/index.php/eletrotecnica/2193-el134.html>. Acesso em: 4 novembro 2014.

BROWN, R. et al. Report to congress on server and data center energy efficiency: Public law 109-431. **Lawrence Berkeley National Laboratory**, 2008.

CARISSIMI, A. et al. Energy-aware scheduling of parallel programs. In: **Conferência Latino Americana de Computação de Alto Rendimento (CLCAR). Gramado,:[sn]**. [S.l.: s.n.], 2010. p. 95–101.

CHANDRA, R. **Parallel programming in OpenMP**. [S.l.]: Morgan Kaufmann, 2001.

COMPUGREEN. **The Green500 List News And Submitted Items**. 2015. Disponível em: <http://www.green500.org/>. Acesso em: 7 maio 2015.

EMAMI, S. **Jetson TK1**. 2015. Disponível em: <http://elinux.org/Jetson_TK1>. Acesso em: 27 junho 2015.

EMAMI, S. **Jetson/Computer Vision Performance**. 2015. Disponível em: <http://elinux.org/Jetson/Computer_Vision_Performance>. Acesso em: 19 junho 2015.

GROPP, W. et al. A high-performance, portable implementation of the mpi message passing interface standard. **Parallel computing**, Elsevier, v. 22, n. 6, p. 789–828, 1996.

GUNARATNE, C.; CHRISTENSEN, K.; NORDMAN, B. Managing energy consumption costs in desktop pcs and lan switches with proxying, split tcp connections, and scaling of link speed. **International Journal of Network Management**, John Wiley and Sons, Ltd., v. 15, n. 5, p. 297–310, 2005. ISSN 1099-1190. Available from Internet: <http://dx.doi.org/10.1002/nem.565>.

HEWLETT-PACKARD et al. Advanced configuration and power interface specification revision 5.0 errata a. 2013.

HUANG, W. et al. Many-core design from a thermal perspective. In: ACM. **Proceedings of the 45th annual Design Automation Conference**. [S.l.], 2008. p. 746–749.

INTEL. **Measuring Processor Power**. 2011. Disponível em: <http://www.intel.com/content/dam/doc/white-paper/resources-xeon-measuring-processor-power-paper.pdf>. Acesso em: 1 julho 2015.

KHAN, S. U.; BOUVRY, P.; ENGEL, T. Energy-efficient high-performance parallel and distributed computing. **The Journal of Supercomputing**, Springer, v. 60, n. 2, p. 163–164, 2012.

KHRONOS, G. **OpenCL - The open standard for parallel programming of heterogeneous systems**. 2015. Disponível em: <https://www.khronos.org/opencl/>. Acesso em: 30 junho 2015.

KLUG, B.; SHIMPI, A. L. **The GPU - NVIDIA Tegra K1 Preview and Architecture Analysis**. 2014. Disponível em: <http://www.anandtech.com/show/7622/nvidia-tegra-k1/3>. Acesso em: 1 julho 2015.

KONDURI, G.; GOODMAN, J.; CHANDRAKASAN, A. Energy efficient software through dynamic voltage scheduling. In: IEEE. **Circuits and Systems, 1999. ISCAS'99. Proceedings of the 1999 IEEE International Symposium on**. [S.l.], 1999. v. 1, p. 358–361.

KOOMEY, J. Growth in data center electricity use 2005 to 2010. **A report by Analytical Press, completed at the request of The New York Times**, 2011.

KOOMEY, J. G. **Estimating total power consumption by servers in the US and the world**. [S.l.]: February, 2007.

LEE, a. **Cubietruck is put into trial production**. 2015. Disponível em: <http://cubieboard.org/2013/09/14/cubietruck-is-put-into-trial-production/>. Acesso em: 19 maio 2015.

MORIMOTO, C. E. **Cortex A15: o futuro?** 2012. Disponível em: <http://www.hardware.com.br/tutoriais/evolucao-dos-smartphones-parte3/cortex-a15-futuro.html>. Acesso em: 1 julho 2015.

NASA. **NAS Parallel Benchmarks**. 2014. Disponível em: <http://www.nas.nasa.gov/publications/npb.html>. Acesso em: 6 novembro 2014.

NVIDIA. **Parallel Programming and Computing Platform**. 2015. Disponível em: <http://www.scielo.br/scielo.php?pid=S0100-40422003000300018&script=sci_arttext>. Acesso em: 18 maio 2015.

OPTIMIZING system performance. 2013. Disponível em: <http://linux-sunxi.org/Optimizing_system_performance>. Acesso em: 1 julho 2015.

PADOIN, E. L. et al. Time-to-solution and energy-to-solution: a comparison between arm and xeon. In: IEEE. **Applications for Multi-Core Architectures (WAMCA), 2012 Third Workshop on**. [S.l.], 2012. p. 48–53.

PADOIN, E. L. et al. Análise de desempenho, escalabilidade e eficiência energética de mpsocs com processadores arm. In: **Conferencia Latino Americana de Computación de Alto Rendimiento (CLCAR)**. San José, Costa Rica: [s.n.], 2013.

PEčOVNIK, I. **Cubieboard / Cubietruck Debian SD image**. 2013. Disponível em: <http://www.igorpecovnik.com/2013/12/24/cubietruck-debian-wheezy-sd-card-image/>. Acesso em: 5 maio 2015.

PROJECT.EU montblanc. **Mont-Blanc Project Home Page**. 2015. Disponível em: <http://www.montblanc-project.eu/>. Acesso em: 7 Maio 2015.

REDHAT. **Usando os Governadores CPUfreq**. 2015. Disponível em: <https://access.redhat.com/documentation/pt-BR/Red_Hat_Enterprise_Linux/6/html/Power_Management_Guide/cpufreq_governors.html>. Acesso em: 6 novembro 2014.

RUTH, S. Green it more than a three percent solution? **Internet Computing, IEEE**, v. 13, n. 4, p. 74–78, July 2009. ISSN 1089-7801.

SCHÄPPI, B. et al. Energy efficient servers in europe–energy consumption, saving potentials and measures to support market development for energy efficient solutions, report. **Intelligent energy Europe project**, 2009.

SMITH, J.; HAMILTON, A. Massive affordable computing using arm processors in high energy physics. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2015. v. 608, n. 1, p. 012001.

WEHNER, M.; OLIKER, L.; SHALF, J. A real cloud computer. **Spectrum, IEEE**, IEEE, v. 46, n. 10, p. 24–29, 2009.

ANEXO A Primeira parte do trabalho de conclusão Avaliação da ferramenta EMonDaemon de Monitoramento de Consumo de Energia

Jorge X. S. Junior [1], **Alexandre Carissimi**[1]

[1]Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre - RS - Brasil

{jxsjunior,asc}@inf.ufrgs.br

Resumo. *O consumo de energia é um dos principais desafios para alcançar um desempenho Exascale. Pesquisadores neste tópico utilizam o consumo de energia de todo o sistema para as suas avaliações, devido à dificuldade de isolar a demanda de potência dos processadores. Embora os novos processadores possuam sensores que permitem medições precisas, eles fornecem diferentes interfaces para coleta de dados, o que torna difícil correlacionar o desempenho com consumo de potência/energia. Para superar esse problema, foi desenvolvida uma ferramenta independente de plataforma que coleta dados de potência e energia de sistemas homogêneos e heterogêneos. Essa ferramenta chamada EMonDaemon será comparada com ferramenta Performance Counter Monitor (PCM), criada pela Intel, usando aplicações do NAS Parallel Benchmarks e aplicações escritas em CHARM++.*

A geração de energia no mundo hoje está baseada, na sua maior parte, em combustíveis fósseis. Tais fontes são poluentes e não renováveis. Com a crescente preocupação ambiental observada nas últimas décadas, a diminuição do consumo de energia e sua melhor utilização é um assunto em pauta em múltiplas áreas de estudo.

Na computação, o consumo de energia já é há algum tempo, motivo de preocupações em sistemas embarcados e computadores portáteis. Preocupados com a duração das baterias e a dissipação de calor de tais sistemas, criaram-se diversas técnicas para se reduzir o consumo energético dessas plataformas. Contudo, a constante demanda por melhor desempenho, aliada ao aumento do custo energético propôs o problema do consumo de energia aos desktops e servidores, anteriormente alheios a ele.

O conceito de *Green Computing* (Computação Verde) foi introduzido em 1992, quando o US *Environmental Protection Agency* (EPA) lançou o padrão *Energy Star* [Ruth 2009]. Tratava-se de uma certificação fornecida a equipamentos eletrônicos energeticamente eficientes. A partir de então, o *Energy Star* passou a ser uma certificação importante e reconhecida no mercado. Diversos fabricantes concentraram esforços no desenvolvimento de produtos energeticamente mais eficientes para responder a essa demanda de mercado.

Um dispositivo energeticamente eficiente consome uma quantidade de energia proporcional à quantidade de trabalho que ele produz [Gunaratne et al. 2005]. Dessa forma, todo o aumento na quantidade de potência dissipada por esse equipamento é justificada por um aumento proporcional na quantidade de trabalho produzido por esse aparelho.

A eficiência energética e o consumo de energia tornaram-se uma das questões mais críticas em relação ao projeto e implantação de uma unidade de computação de alto desempenho (HPC - *High-performance computing*), ou um centro de dados em geral. Enquanto o consumo

de energia combinado dos sistemas de HPC mundial continua a ser pequeno em termos relativos (2% do total de emissões de CO_2), em termos absolutos, o custo já é extremamente elevado, cerca de 200-300 bilhões kW h [Koomey 2011, Brown et al. 2008]. Essa tendência irá aumentar ainda mais no futuro próximo, e o custo de energia também está aumentando, que em breve pode levar a uma situação crítica devido aos limites de eficiência energética [Khan et al. 2012].

Em 2005, o consumo total de energia para servidores e suas unidades de refrigeração foi projetado em 1,2% do consumo total de energia dos EUA e dobrando a cada cinco anos [Koomey 2007, Brown et al. 2008].

Uma análise dos atuais sistemas de HPC mostra que 40-60% do consumo de energia pode ser atribuída aos nós de computação (processadores e memórias), 10% para os sistemas de interconexão e de armazenamento, e o restante (até 50%) é consumida pela própria infraestrutura, incluindo iluminação, fonte de alimentação, e acima de tudo, refrigeração [Schäppi et al. 2009]. Consequentemente, o maior retorno pode ser esperado da melhoria da eficiência energética dos nós de computação, uma vez que qualquer melhoria também se traduzirá em requisitos de refrigeração reduzidos.

Assim sendo, em 2008 especialistas alertaram através do relatório oficial DARPA [Bergman et al. 2008] que o consumo de energia aceitável para chegar ao exascale - computação exascale refere-se a sistemas de computação capazes de, pelo menos, 10^{18} operações de ponto flutuante por segundo - seria de 20 MW. Se consideramos essa recomendação, a eficiência energética dos futuros sistemas de exascale tem um limite de 50 Gflops/W.

Se sistemas de escala exascale fossem construídos com a tecnologia atualmente empregada, seu consumo de potência alcançariam a ordem de GigaWatt. Isto é equivalente a toda a produção de uma usina de energia nuclear de tamanho médio [Wehner et al. 2009].

1. Conceitos Básicos

Energia é uma grandeza física que representa a capacidade de se realizar trabalho. Ela é medida em Joules (J) ou em quilowatt-hora (1 kWh = 3600 kJ). Potência é a taxa com que a energia é transferida, medida em Watt (W = Joules por segundo). A relação entre energia e potência é:

$$P = dE/dt \tag{1}$$

Sendo *P* a potência, *E* a energia e *t* o tempo. Em sistemas elétricos, a potência pode ser definida como o trabalho realizado pela corrente elétrica em um determinado intervalo de tempo. Em um sistema de corrente contínua (*Direct Current* – DC) no qual a tensão (V), medida em Volts, e a corrente (I), medida em Ampères, se mantém constantes durante um dado período, a potência transmitida é dada por:

$$P = V * I \tag{2}$$

Sabendo que $V = R * I$, podemos chegar à expressão:

$$P = I^2 * R \tag{3}$$

Sendo *R* a resistência do circuito, medida em Ohm(Ω).

Para o caso de sistemas de corrente alternada (*Alternating Current* – AC), a tensão é comumente expressa em V_{RMS}(*Root Mean Square*, em inglês). A corrente, por sua vez, é I_{RMS}. Em sistemas desse tipo, a corrente I(t) – e, portanto, a potência instantânea – são funções dependentes do tempo, logo, a equação 3 precisa ser estendida para refletir esse fato. Se a função é periódica como ocorre, por exemplo, na rede elétrica pública de transmissão, a análise deve ser feita em relação à potência média dissipada ao longo do tempo. Logo:

$$P_{média} = I(t)^2 * R = R * I_{RMS}^2 \tag{4}$$

Ou seja, o valor RMS, I_{RMS}, da função I(t) é o valor constante que leva à mesma dissipação de potência que a dissipação de potência média ao longo do tempo da corrente I(t). Podemos, também, estender a relação para um sistema com tensão não constante, V(t), com valor RMS expresso por:

$$P_{média} = V_{RMS}^2/R \tag{5}$$

Ao igualar as equações 4 e 5, obtem-se:

$$P_{média} = I_{RMS} * V_{RMS} \tag{6}$$

É importante ressaltar que as equações aqui obtidas consideram apenas sistemas puramente resistivos (por exemplo: chuveiros, lâmpadas incandescentes, ferros de passar roupa), o que significa que não há cargas reativas, tais como capacitores e indutores, que são capazes de não só dissipar energia, mas também de armazená-la. Como exemplo de cargas indutivas, há os motores e os transformadores. Por sua vez, cargas capacitivas podem ser representadas por lâmpadas fluorescentes e computadores.

Quando uma tensão senoidal é aplicada numa carga resistiva, a corrente circulante pela carga acompanha instantaneamente as variações de tensão, como mostra a figura 1.

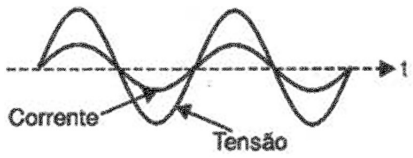


Figura 1. Corrente e tensão estão, normalmente, em fase em um dispositivo puramente resistivo alimentado por corrente alternada. Adpatado de [Braga 2014]

Contudo, na maioria dos casos reais, o sistema não se comporta como uma resistência pura. Adicionando-se componentes reativos ao sistema, o fluxo de potência flutua, ou seja, a corrente pode fluir tanto em um sentido quanto em outro ao longo do tempo. Nesse caso, a corrente não acompanha as variações de tensão instantaneamente, havendo um retardo ou um adiantamento, conforme ilustra a figura 2.

Nesse caso, é necessário introduzir alguns conceitos como potência real ou potência ativa (P), potência reativa (Q), potência complexa (S) e potência aparente (|S|). A potência real (ativa), medida em Watt, representa a energia gasta em determinado espaço de tempo. A potência reativa, medida em volt-ampère reativo (var), representa a porção do fluxo de potência devido à energia armazenada que retorna à fonte a cada ciclo. Por fim, temos a potência complexa, medida em volt-ampère (VA), e a potência aparente, também medida em volt-ampère

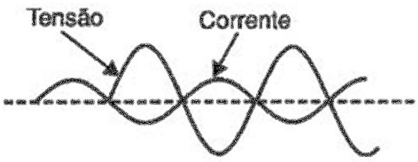


Figura 2. Corrente e tensão não estão em fase quando há presença de componentes reativos. Adpatado de [Braga 2014]

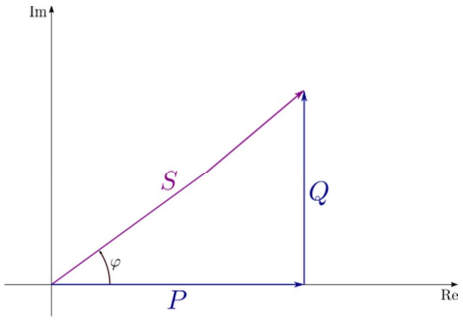


Figura 3. A potência complexa (S) é o vetor soma das potências real (P) e reativa (Q). A potência aparente (|S|) é a magnitude da potência complexa.

(VA), que é o valor absoluto da potência complexa e representa o produto da corrente pela tensão do circuito. Esses conceitos se relacionam como mostrado na figura 3.

O ângulo φ formado pela potência complexa(S) e pela potência real(P) é usado para indicar o fator de potência (FP), que varia entre 0 e 1, e é dado por:

$$FP = cos(\varphi) \tag{7}$$

O melhor aproveitamento num sistema de corrente alternada ocorre quando o ângulo de defasagem é zero – ou seja, cosseno igual a 1. Nesse caso, 100% da energia aplicada é convertida em trabalho. Na prática, no entanto, equipamentos raramente atingem esse padrão. Um fator de potência baixo significa que energia reativa está sendo gerada e não é aproveitada [Braga 2014]. Com efeito, a medida de potência em Watt de um equipamento eletrônico determina a verdadeira potência que a companhia de energia elétrica deve fornecer para o dispositivo funcionar. Por outro lado, a grandeza volt-ampère (VA) é usada para dimensionar os fios e sistemas de proteção das instalações elétricas. A classificação Watt e VA é igual para alguns tipos de aparelhos, tal como as lâmpadas incandescentes. No entanto, para equipamentos eletrônicos, essas grandezas podem se diferenciar significativamente, com a medida em VA sendo sempre igual ou maior que a medida em Watt. Em sistemas computacionais, como visto em [Konduri, 1999, apud CARISSIMI et al, 2010], a energia utilizada para realização de um programa executado em um processador baseados em switches CMOS pode ser expressa como:

$$E_{op} = C * V_{dd}^2 * f * (N/f) + I_{fuga} * V_{dd} * (N/f) \tag{8}$$

Onde E_{op} é a energia, C é a capacitância de chaveamento, V_{dd} é a voltagem na qual o circuito está operando, f é a frequência de operação, I_{fuga} a corrente de fuga e N o número de ciclos requeridos para executar o programa. Observando a expressão acima, notamos que, para diminuir

o consumo, podemos reduzir a voltagem de alimentação (V_{dd}). Contudo a frequência de processamento deve ser diminuída para que tal mudança seja possível. A redução da frequência faz com que o processador execute instruções em um ritmo menor, prejudicando o seu desempenho. Surge, então, um compromisso consumo energético versus desempenho.

2. Estratégias para Redução do Consumo de Energia

Nesta seção serão descritas as principais estratégias em hardware e software para redução do consumo de energia.

2.1. Estratégias em hardware

A introdução da instrução HLT (halt), no início da Era Pentium, foi uma das primeiras tentativas de se reduzir o consumo de processadores. Os projetistas da Intel valeram-se do fato de que o processador não é exigido em seu máximo desempenho todo o tempo. Dessa forma, através da instrução HLT, o sistema operacional, ao identificar que não há nada a executar, pode colocar o processador em um estado de baixo consumo quando isso for interessante.

Após a instrução HLT, os mecanismos em hardware para controle do consumo de energia evoluíram. Se destacando a APM e suas formas para controle do consumo de energia.

2.1.1. APM (Advanced Power Management)

A APM (*Advanced Power Management*), criada pela Intel em 1992, foi a primeira interface de gerenciamento de consumo entre o firmware e o sistema operacional. O APM utiliza uma abordagem em camadas para comunicar dispositivos com o sistema operacional. O hardware é controlado pela APM-*aware* BIOS que se comunica com um *driver* APM específico para o sistema operacional. A comunicação ocorre em ambos os sentidos: eventos de gerenciamento de energia são enviados a partir do BIOS para o *driver* APM, o *driver* APM envia informações e pedidos à BIOS através de chamadas de função.

Através da APM, o sistema operacional pode obter informações pertinentes à gerência de consumo e colocar o sistema em diferentes estados de menor atividade.

2.1.2. Advanced Configuration and Power Interface (ACPI)

Lançada em dezembro de 1996, ACPI [Hewlett-Packard et al. 2013] é o sucessor do APM. A ACPI dá ao sistema operacional toda a responsabilidade do gerenciamento de consumo, tendo em vista que é ele que está na melhor posição para avaliar o status de funcionamento do sistema.

A *Advanced Configuration and Power Interface* (ACPI) foi desenvolvida para estabelecer interfaces comuns para a indústria, permitindo um robusto sistema para configuração dos dispositivos presentes na placa-mãe e o gerenciamento de energia desses dispositivos. A ACPI é o elemento-chave na *Operating System-directed configuration and Power Management* (OSPM). A ACPI evoluiu as interfaces de configuração da placa-mãe existentes para suportar arquiteturas avançadas de uma forma mais robusta, e potencialmente mais eficiente. De uma perspectiva de gerenciamento de energia, OSPM / ACPI promove o conceito de que os sistemas devem economizar energia através da transição dos dispositivos não utilizados para estados de menor consumo de energia, incluindo colocar todo o sistema em um estado de baixo consumo de

energia quando possível. Para a ACPI conseguir esse objetivo ela faz uso de diversos mecanismos implementados pelo hardware como mostrados na figura 4. Os quais serão discutidos nas próximas seções.

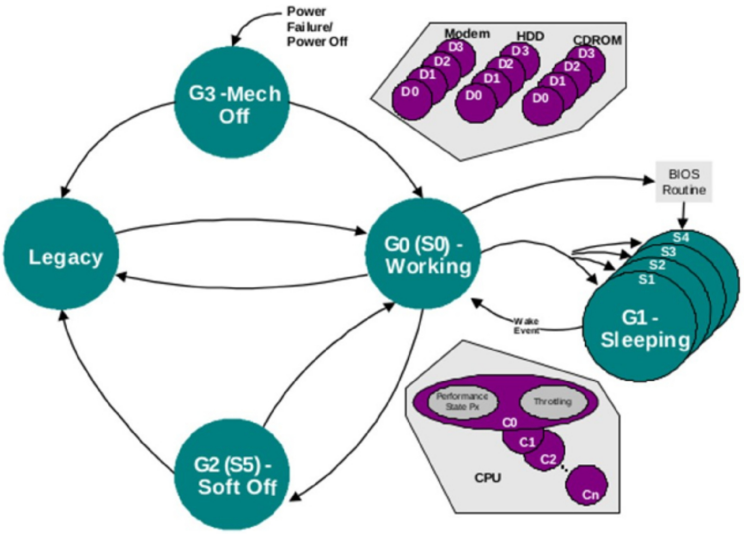


Figura 4. Estados possíveis do ACPI. Adaptado de [Hewlett-Packard et al. 2013]

2.1.3. Global System State (Gx states)

Os *Gx states* aplicam-se a todo o sistema e são perceptíveis pelo usuários. A tabela 1 mostra o resumo dos *Gx states*. Esses estados são divididos em 4 fases:

- G3 *Mechanical Off*: A energia do computador foi totalmente removida por meio de um interruptor mecânico (como um botão na parte de trás da fonte).
- G2/S5 *Soft Off*: Um estado onde o computador consome uma quantidade mínima de energia. Sem aplicações sendo executadas. Esse estado requer um grande tempo até voltar ao estado G0. O sistema operacional precisa ser reiniciado para voltar ao estado de funcionamento.
- G1 *Sleeping*: Um estado em que o computador consome uma pequena quantidade de energia, as aplicações do usuário não estão sendo executadas e o sistema operacional ”parece” estar desligado. O tempo para retornar ao estado de trabalho varia de acordo com o *Sleeping State* selecionado antes da entrada nesse estado. O trabalho pode ser retomado sem reiniciar o sistema operacional.
- G0 *Working*: Um estado onde o sistema operacional está executando aplicações. Nesse estado, os periféricos tem seu estado de energia alterada dinamicamente. O usuário pode selecionar, por meio de uma API, várias características de desempenho/potência do sistema para que o software tenha seu desempenho otimizado ou a vida útil da bateria aumente.

Conforme a figura 4, no estado G1 existem subestados que são os *Sleeping State (Sx states)*. Os *Sleeping State* são estados de repouso. Os Sx states são estados com com baixa tempo de retorno ao estado G0 e eles são brevemente definidos a seguir:

- S1: Nesse estado, nenhum contexto do sistema - refere-se aos estados de operação processador, memória cache e dispositivos - é perdido e o hardware mantém todo o contexto do sistema.
- S2: Esse estado é semelhante ao estado de repouso S1, exceto que o contexto da CPU e a cache do sistema são perdidos (o sistema operacional é responsável por manter os caches e o contexto da CPU).
- S3: Todo o contexto do sistema é perdido, exceto a memória do sistema. CPU, cache, e contexto do *chipset* são perdidos nesse estado. O hardware mantém o contexto de memória e restaura algum contexto para a CPU.
- S4: O estado de repouso S4 é o de mais baixo consumo de energia, mas com alto tempo de retorno ao estado G0. A fim de reduzir a um consumo mínimo de energia, assume-se que a plataforma de hardware tem todos os dispositivos desligados. O contexto da plataforma é mantido.
- S5 *Soft Off*: O estado S5 é semelhante ao estado S4, exceto que o sistema operacional não salva qualquer contexto. O sistema está em estado de *soft off* e requer um boot completo para ativação.

Assim como ocorre no estado G1, o estado G0 é composto por vários subestados denominados *Processor Power State (Cx states)*. Os *Processor Power State* são estados de consumo de potência e de gerenciamento térmico do processador. Eles possuem semântica de entrada e saída específicos e são brevemente definidos a seguir:

- C0: Enquanto o processador está nesse estado, ele executa as instruções.
- C1: Nesse estado de energia o processador tem a menor latência. A latência do hardware nesse estado deve ser baixa o suficiente para que o sistema operacional não considere esse aspecto ao decidir se deve usá-lo. Além de colocar o processador em um estado de não execução, esse estado não tem outros efeitos visíveis ao software.
- C2: O estado C2 oferece maior economia de energia sobre o estado C1. A pior latência para esse estado é fornecida pela ACPI e o sistema operacional pode usar essa informação para determinar quando o estado C1 deve ser usado em vez do estado C2. Além de colocar o processador em um estado de não execução, esse estado não tem outros efeitos visíveis ao software.
- C3: O estado C3 oferece maior economia de energia que os estados C1 e C2. A pior latência para esse estado é fornecida pela ACPI e o sistema operacional pode usar essa informação para determinar quando o estado C2 deve ser usado em vez do estado C3. Enquanto no estado C3, as caches do processador mantêm o estado, mas ignoram qualquer mudança. O sistema operacional é responsável por garantir que as caches mantenham a coerência.

Ainda utilizando a figura 4 como referência, no subestado C0 do estado G0 existem os subestados que são os *Device and Processor Performance State (Px states)*. Os *Device and Processor Performance State* são estados de consumo de energia e de capacidade enquanto no estado ativo, C0 para processadores e D0 para dispositivos. Os *Px states* são brevemente definidos a seguir.

- P0: Enquanto um dispositivo ou processador está nesse estado, ele usa a sua capacidade máxima de desempenho e pode consumir sua potência máxima.
- P1: Nesse estado, o desempenho de um dispositivo ou processador é limitado abaixo do seu máximo desempenho e consome menos do que a potência máxima.

- Pn: Nesse estado, o desempenho de um dispositivo ou processador está no seu nível mínimo de desempenho e consome potência mínima, permanecendo num estado ativo. O estado *n* é um número máximo e depende do processador ou dispositivo. Processadores e dispositivos podem definir o suporte para um número arbitrário de estados de desempenho desde que não deve exceda 16.

Tabela 1. Resumo dos Global System State

Global system state	Software é executado	Latência	Consumo de energia	Necessário restart SO
G0 Working	Sim	0	Alto	Não
G1 Sleeping	Não	>0, varia com sleep state	Baixo	Não
G2/S5 Soft Off	Não	Alta	Muito próximo de zero	Sim
G3 Mechanical Off	Não	Alta	Bateria presente na placa-mãe	Sim

2.1.4. Device Power State

Os *Device Power State* são estados de consumo de energia específicos dos dispositivos. Geralmente não são perceptíveis pelo usuário. Por exemplo, alguns dispositivos podem estar no estado Desligado, embora o sistema como um todo está no estado de trabalho. Os estados de energia dispositivo são genericamente definidos a seguir. Esses estados são divididos em 5 fases:

- D3 (Off): A energia foi completamente removida do dispositivo. O contexto de dispositivo é perdido quando ele entra nesse estado, de modo que o sistema operacional irá reinicializar o dispositivo quando ligá-la novamente. Dispositivos nesse estado tem o mais longo tempo de restauração. Todas as classes de dispositivos definem esse estado.
- D3hot: É definido para cada classe de dispositivos. Em geral, D3Hot é utilizado para economizar energia e, opcionalmente, preservar o contexto do dispositivo. Se o contexto do dispositivo é perdido quando ele entra nesse estado, o sistema operacional irá reinicializar o dispositivo quando houver transição para D0. Dispositivos nesse estado podem ter longo tempo de restauração. Todas as classes de dispositivos definem esse estado. O estado D3Hot difere do estado D3 em dois parâmetros distintos: O dispositivo ainda está energizado e software pode acessar um dispositivo em D3Hot.
- D2: É definido para cada classe de dispositivos embora nem todos o fazem. Em geral, o estado D2 é utilizado para economizar energia e preservar menos contexto que o estado D1 ou D0. O estado D2 pode levar o dispositivo a perder algum contexto (por exemplo, através da redução de energia no barramento, forçando assim o dispositivo a desligar algumas de suas funções).
- D1: É definido para cada classe de dispositivos embora nem todos o fazem. Em geral, o estado D1 deve economizar menos energia e preservar mais contexto que o estado D2.

- D0 (Fully-On): Esse estado é assumido como sendo o de mais elevado de consumo de energia. O dispositivo está completamente ativo.

2.2. Estratégias em software

Em nível de sistema operacional, a partir do núcleo 2.6.12, o Linux passou a oferecer uma infraestrutura chamada *cpufreq* [redhat 2014]. A ferramenta se baseia no conceito de governadores. Um governador é um algoritmo que calcula a frequência que ele considera adequada para a CPU em um determinado momento. Os diferentes tipos de governadores de *cpufreq* disponíveis estão descritos abaixo.

- *cpufreq_performance*: Força a CPU a usar a frequência de relógio mais alta possível. Como tal, esse governador em particular não oferece nenhum benefício de economia de energia.
- *cpufreq_powersave*: Força a CPU a usar a frequência de relógio mais baixa possível. Como tal, esse governador específico oferece economia máxima de energia, mas ao custo de menor desempenho de CPU.
- *cpufreq_ondemand*: É um governador dinâmico que permite que a CPU alcance a frequência máxima de relógio quando a carga do sistema for alta, e também uma frequência mínima de relógio quando o sistema estiver ocioso. Embora isso permita que o sistema ajuste o consumo de energia de acordo com o requisitado, quanto à carga do sistema, ele faz isso ao custo de latência maior entre a mudança de frequência. Como tal, a latência pode retirar qualquer benefício de economia de energia se o sistema mudar entre ocioso e cargas de trabalho pesadas muito seguido.
- *cpufreq_userspace*: Permite que programas do usuário (ou qualquer processo que estiver executando como administrador) ajuste a frequência da CPU.
- *cpufreq_conservative*: Como o *cpufreq_ondemand*, o *cpufreq_conservative* também ajusta a frequência do relógio de acordo com o uso. No entanto, enquanto o *cpufreq_ondemand* o faz de forma mais agressiva, (ou seja, do máximo para o mínimo e de volta ao máximo), o *cpufreq_conservative* muda entre as frequências gradualmente. Isso significa que o *cpufreq_conservative* irá ajustar uma frequência de relógio que está destinada a suprir a carga, ao invés de simplesmente escolher entre o máximo e mínimo. Embora isso possa fornecer muita economia no consumo de energia, ele o faz em uma latência ainda maior do que o *cpufreq_ondemand*.

2.3. Outros recursos

Outra técnica utilizada pelos fabricantes de processadores é utilizar o monitoramento do uso dos núcleos do processador para desligar aqueles que não são usados economizando energia e potência sem afetar o desempenho do sistema. A figura 5 mostra um processador com 4 núcleos com 3 deles desativados.

3. Processadores Intel e Consumo De Energia

Nesta seção serão descritos os recursos de hardware e software utilizados pelos processadores Intel para medir o consumo de energia.

3.1. Recursos em hardware

Os recentes processadores da Intel para servidores Xeon Série E5 e E7 fornecem informações sobre a energia consumida pela CPU (núcleos e memória cache) através de registradores disponíveis no *chip* do processador. Esses sensores são chamados *Running Average Power Limit*

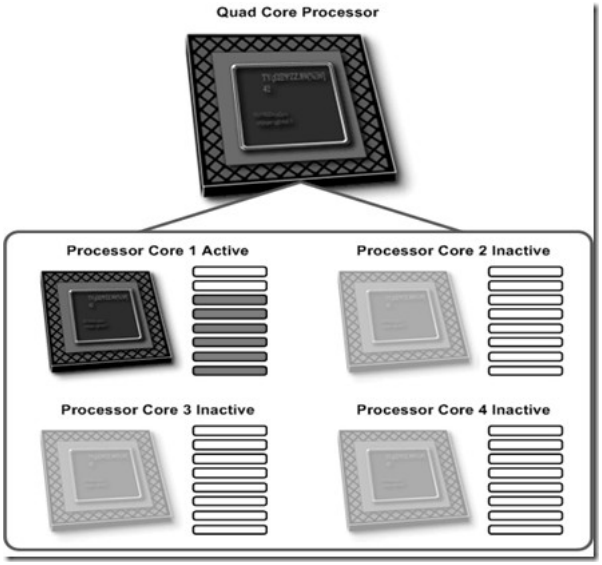


Figura 5. Exemplo

(RAPL). A Intel introduziu o RAPL com a microarquitetura Sandy Bridge. Os sensores que permitem medir o consumo de energia dos componentes são listados na Tabela 2.

O RAPL pode ser configurado e examinado por leitura nos registradores específicos de máquina (MSR - *Machine Specific Registers*). Na arquitetura Intel isso só é possível no modo privilegiado.

Tabela 2. Sensores RAPL disponíveis

RAPL_PKG	CPU Inteira
RAPL_PP0	Apenas núcleos de processador
RAPL_PP1	Um dispositivo específico fora do core
RAPL_DRM	Controlador de memória

3.2. Recursos em software

A Intel fornece uma ferramenta paa medir o consumo de energia denominada *Performance Counter Monitor* (PCM) [Willhalm 2014] que faz uso dos recursos descritos anteriormente. Ela tem o funcionamento baseado em duas fases:

- Na primeira fase é feita a descoberta do hardware, onde são recolhidas informações sobre a plataforma, tais como:
  - Número de processadores físicos,
  - Número de processadores lógicos,
  - Número de sockets,
  - Frequência nominal do processador,
  - Package thermal spec power, definida como a potência dissipada especificada pelo fabricante para o processador,
  - Package minimum power, definida como a potência mínima dissipada pelo processador,
  - Package maximum power, definida como a potência máxima dissipada pelo processador.
- Na segunda fase é feito o monitoramento do consumo de energia de cada um dos possíveis processadores presentes na plataforma. Para cada processador físico, indica qual foi o consumo de energia e de potência na execução da aplicação utilizada.

4. Energy Monitoring Tool: EMonDaemon

O EMonDaemon [Padoin et al. 2014], desenvolvida no grupo GPPD/INF-UFRGS pelo doutorando Edson L. Padoin, é uma solução para medir a potência instantânea e consumo de energia durante a execução de aplicações. Ele funciona em duas fases consecutivas, quando executado juntamente com as aplicações, tal como descrito abaixo.

- fase de descoberta: no início, a ferramenta obtém informações estáticas sobre a plataforma, como o fabricante do processador, modelo de processador, frequências de relógio disponíveis e frequência de relógio atual. Essas informações irám orientar as decisões sobre como os dados de potência e energia serão recolhidos.
- Fase de monitoramento: durante a execução da aplicação, EMonDaemon monitora o sistema para coletar informações sobre energia, ou potência, com uma periodicidade definida pelo usuário. Ele mantém alguns dados estatísticos, tais como a potência mínima, potência máxima e potência média, e o consumo de energia. Além disso, a ferramenta pode também registrar a potência instantânea e frequência de relógio dos processadores durante a execução.

5. Proposta

A ferramenta PCM é específica para os processadores fabricados pela Intel. O EMonDaemon tem como objetivo ser independente de plataforma. O objetivo deste trabalho é comparar a ferramenta EMonDaemon com o PCM quanto a precisão, consumo de energia e frequência de mensuração.

5.1. Metodologia

Para realizar a comparação entre as ferramentas PCM e EMonDaemon, serão medidos o consumo de cada ferramenta na execução de benchmarks tradicionalmente usados em HPC. Assim como serão utilizados benchmarks feitos utilizando uma abordagem mais atual.

Serão usados os seguintes benchmarks:

1. Aplicações feitas usando a linguagem CHARM++ [Parallel Programming Lab 2014, of Computer Science at University of Illinois 2014].
2. NAS Parallel Benchmarks (NPB) [NASA 2014]: O NPB é um conjunto de benchmarks derivados da dinâmica de fluidos computacional (CFD) bem reconhecidos para avaliar arquiteturas multicores atuais e emergentes. Serão usadas 9 aplicações diferentes, nomeados:
    - BT - Block Tri-diagonal solver
    - CG - Conjugate Gradient, irregular memory access and communication
    - EP - Embarrassingly Parallel
    - FT - discrete 3D fast Fourier Transform, all-to-all communication
    - IS - Integer Sort, random memory access
    - LU - Lower-Upper Gauss-Seidel solver
    - MG - Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
    - SP - Scalar Penta-diagonal solver
    - UA - Unstructured Adaptive mesh, dynamic and irregular memory access

    Classes possíveis para o NPB
    - Classe S: pequeno para fins de teste rápido
    - Classe W: tamanho da estação de trabalho
    - Classes A, B, C: problemas de teste padrão; quatro vezes é o aumento de tamanho de uma classe para a próxima
    - Classes D, E, F: grandes problemas de teste; dezesseis vezes é o aumento de tamanho de uma classe para a próxima

    Os testes utilizando o NPB serão efetuados usando a classe B.

5.2. Ambiente Experimental

A plataforma sobre a qual será efetuada o trabalho proposto, é:

- Sistema operacional utilizado: Ubuntu 12.04.5 LTS com kernel 3.8.13.13.
- Processador utilizado: Intel(R) Xeon(R) CPU E5-2650 @ 2.00GHz com 8 núcleos e 16 threads.

6. Cronograma

A Tabela 3 apresenta a proposta de cronograma para a execução deste trabalho de graduação.

Tabela 3. Cronograma de atividades para o trabalho de graduação

Tarefa	JAN	FEV	MAR	ABR	MAI	JUN
1. Instalação das ferramentas	•					
2. Execução dos Benchmarks	•	•				
3. Avaliação dos resultados			•	•		
4. Escrita da Monografia			•	•	•	
5. Entrega e Apresentação						•

Referências

Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hill, K., Hiller, J., et al. (2008). Exascale computing study: Technology challenges in achieving exascale systems. *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep*, 15.

Braga, N. C. (2014 (acessado Novembro 4, 2014)). *Transformadores e fator de potência (EL104)*. `http://www.newtoncbraga.com.br/index.php/eletrotecnica/2193-el134.html`.

Brown, R. et al. (2008). Report to congress on server and data center energy efficiency: Public law 109-431. *Lawrence Berkeley National Laboratory*.

Carissimi, A., Geyer, C. F., Maillard, N., Navaux, P. O., Cavalheiro, G. G., Pilla, M. L., Yamin, A. C., Charao, A. S., Stein, B., De Rose, C. A., et al. (2010). Energy-aware scheduling of parallel programs. In *Conferência Latino Americana de Computação de Alto Rendimento (CLCAR). Gramado,:[sn]*, pages 95–101.

Gunaratne, C., Christensen, K., and Nordman, B. (2005). Managing energy consumption costs in desktop pcs and lan switches with proxying, split tcp connections, and scaling of link speed. *International Journal of Network Management*, 15(5):297–310.

Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba (2013). Advanced configuration and power interface specification revision 5.0 errata a.

Khan, S. U., Bouvry, P., and Engel, T. (2012). Energy-efficient high-performance parallel and distributed computing. *The Journal of Supercomputing*, 60(2):163–164.

Konduri, G., Goodman, J., and Chandrakasan, A. (1999). Energy efficient software through dynamic voltage scheduling. In *Circuits and Systems, 1999. ISCAS'99. Proceedings of the 1999 IEEE International Symposium on*, volume 1, pages 358–361. IEEE.

Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*.

Koomey, J. G. (2007). Estimating total power consumption by servers in the us and the world.

NASA ((acessado Novembro 6, 2014)). *NAS Parallel Benchmarks*. `http://www.nas.nasa.gov/publications/npb.html`.

of Computer Science at University of Illinois, D. ((acessado Novembro 20, 2014)). *Parallel Programming Laboratory*. http://charm.cs.uiuc.edu/.

Padoin, E. L., Pilla, L. L., Boito, F. Z., Castro, M., Mehaut, J.-F., and Navaux, P. O. (2014). Performance/energy trade-off in scientific computing: The case of arm big.little and intel sandy bridge.

Parallel Programming Lab, Dept of Computer Science, U. o. I. ((acessado Novembro 20, 2014)). *Charm++:parallel programming framework*. http://charmplusplus.org/.

redhat (2014 (acessado Novembro 6, 2014)). *Usando os Governadores CPUfreq*. https://access.redhat.com/documentation/pt-BR/Red_Hat_Enterprise_Linux/6/html/Power_Management_Guide/cpufreq_governors.html.

Ruth, S. (2009). Green it more than a three percent solution? *Internet Computing, IEEE*, 13(4):74–78.

Schäppi, B., Przywara, B., Bellosa, F., Bogner, T., Weeren, S., Harrison, R., and Anglade, A. (2009). Energy efficient servers in europe–energy consumption, saving potentials and measures to support market development for energy efficient solutions, report. *Intelligent energy Europe project*.

Wehner, M., Oliker, L., and Shalf, J. (2009). A real cloud computer. *Spectrum, IEEE*, 46(10):24–29.

Willhalm, T. (2012 (acessado Novembro 6, 2014)). *Intel Performance Counter Monitor - A better way to measure CPU utilization*. https://software.intel.com/en-us/articles/intel-performance-counter-monitor-a-better-way-to-measure-cpu_-utilization.