UNIVERSIDADE FEDERAL DO PAMPA

Robson R. de Oliveira Gonçalves

Análise do Desempenho e Consumo de Energia de Coprocessadores Xeon Phi utilizando Diferentes Modelos de Programação

Alegrete

2018

Robson R. de Oliveira Gonçalves

Análise do Desempenho e Consumo de Energia de Coprocessadores Xeon Phi utilizando Diferentes Modelos de Programação

Dissertação de Mestrado apresentada ao Programa de Pós-graduação Stricto Sensu em Engenharia Elétrica da Universidade Federal do Pampa, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr Alessandro Girardi

Coorientador: Prof. Dr Claudio Schepke

Alegrete

2018

Robson R. de Oliveira Gonçalves

Análise do Desempenho e Consumo de Energia de Coprocessadores Xeon Phi utilizando Diferentes Modelos de Programação

Dissertação de Mestrado apresentada ao Programa de Pós-graduação Stricto Sensu em Engenharia Elétrica da Universidade Federal do Pampa, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Dissertação de Mestrado defendido e aprovado em 22 de JUNHO de 2018

Banca examinadora:

Prof. Dr Alessandro Girardi
Orientador

Prof. Dr Claudio Schepke
Co-orientador

Prof. Dr. Jumar Luis Russi
UNIPAMPA

Prof. Dra. Aline Vieira de Mello
UNIPAMPA

Prof. Dr. João Vicente Ferreira Lima
UFSM

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

G635a Gonçalves, Robson Romário de Oliveira
Análise do Desempenho e Consumo de Energia de Coprocessadores Xeon Phi utilizando Diferentes Modelos de Programação / Robson Romário de Oliveira Gonçalves.
100 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa, MESTRADO EM ENGENHARIA ELÉTRICA, 2018.
"Orientação: Alessandro Gonçalves Girardi".

1. Processamento de Alto Desempenho. 2. Xeon Phi. 3. Consumo de Energia. I. Título.

Agradecimentos

Primeiramente agradeço a Deus, por sempre ter escutado e aceito todos meus pedidos de orientação, principalmente dando muita força para enfrentar os momentos difíceis da vida.

Aos meus pais, Alberto (in memoriam) e Leda, meu eterno agradecimento. Sempre incentivaram e acreditaram em minha capacidade ensinando que o céu é o limite e não existe o impossível. Isto me fortaleceu e principalmente aprendi a enfrentar os desafios da vida e conquistar meus sonhos.

A minha esposa Darciela, por ser tão importante na minha vida. Pelo seu companheirismo, amizade, paciência, compreensão e amor, para que este trabalho pudesse ser concretizado.

A minha querida filha Alice, um anjo que Deus colocou em nossas vidas, onde aprendemos que o mais importante é amar e cuidar de nossa família com felicidade.

Aos meus irmãos, Anderson, Gerson, Jeferson e Ronaldo e também minhas queridas cunhadas, pois sempre se orgunharam de mim e confiaram em meu trabalho.

A prof. Márcia Cera (in memoriam), sempre exigindo e extraindo o máximo de minha capacidade pois sempre acreditou em meu potencial, ajudando e orientando o melhor caminho a seguir. Fico muito feliz e agradecido em conhece-la e poder trabalharmos juntos!

Ao meu orientador Alessandro Girardi, primeiramente por ter aceitado me orientar e dar continuidade ao trabalho independente das adversidades que surgiram. Exigindo maior dedicação de minha parte mas sempre orientando com sólida experiência acadêmica de forma que o trabalho tivesse grande destaque. Obrigado por exigir esforço e comprometimento na elaboração e produção dos artigos acadêmicos, graças a isto conseguimos um artigo com excelente qualis !! Você é grande referência profissional e pessoal. Muito obrigado !!

Ao meu co-orientador Cláudio Schepke, uma luz que veio num momento muito triste logo após a partida da prof. Márcia, onde assumiu a responsabilidade sobre seus os orientandos e principalmente me deu forças para continuar com este trabalho. Obrigado de coração por tudo que fizeste por mim!

E para finalizar um agradecimento ao meu colega e amigo Luciano Vargas pela sua motivação e muitas conversas positivas, onde sempre diz "Tudo que pedir com fé a Deus ele atenderá". Temos a certeza que voaremos muito alto um dia !!

"A felicidade não se resume na ausência de problemas,
mas sim na sua capacidade de lidar com eles."
(Albert Einstein)

Resumo

Com o crescimento computacional ocorrido nos último anos, surgem questões e problemas relacionados ao custo de energia principalmente em ambientes computacionais de grande porte como *data centers* ou centros de processamento de alto desempenho pois exigem grande demanda de energia para manter seu funcionamento. Na área da Computação de Alto Desempenho (HPC), acompanhando esta necessidade surgem novos paradigmas computacionais para suportar o aumento exponencial de núcleos de processadores, mudando de modelos e arquiteturas *multicore* onde centralizavam o processamento somente nas CPUs para novas arquiteturas heterogêneas e *manycore* que distribuem o processamento utilizando GPUs e coprocessadores. A relação desempenho com consumo de energia eficiente em arquiteturas *manycore* ainda é tema recente e existem muitos questionamentos que precisam ser explorados, pois juntamente com o aumento do número de cores surgiram vários modelos de programação. É preciso maior estudo sobre consumo de energia em modelos que usam e gerenciam cargas de trabalho parte em CPUs e parte em GPUs/coprocessadores. Nesse contexto, o objetivo geral deste trabalho é realizar um estudo e metodologia sobre a relação desempenho e consumo de energia em coprocessadores Intel Xeon Phi, analisando e identificando quais os principais fatores que afetam diretamente o consumo eficiente de energia em ambientes *manycore*. A avaliação baseada no comportamento e variação do consumo de energia durante execução dos benchmarks LINPACK, HPL 2.1 e HPCG sobre os modelos de programação *host*, *offload*, *native* e *simétrico* pela Intel. A metodologia utilizada consiste em processos de planejamento dos cenários, execução, monitoramento, coleta, medição e análise dos dados de consumo energético e desempenho sobre os resultados gerados e principalmente das definições dos critérios avaliados, quais métricas utilizadas e modelos de cenários utilizados. As principais contribuições deste trabalho são: Avaliar quais modelos de programação proporcionam melhores resultados com consumo eficiente sem comprometer o desempenho; Analisar e identificar se o comportamento de recursos e configurações de memória compartilhada, número de *nodes*, *cores*, processos e *threads* impactam diretamente na relação desempenho e consumo de energia. Resultados mostram que existe grande variação no desempenho e consumo de energia entre os modelos e a importância da escolha e configuração adequada dos fatores utilizados durante o processamento de aplicações paralelas. Considerando cenários de memória compartilhada sobre benchmark Linpack, os modelos *host* e *offload* apresentam aumento linear no desempenho para cargas de trabalho (size) até 10000 e leve incremento após este valor. Mas o modelo *nativo* apesar de menor desempenho nestes cenários, em contrapartida consome menos energia e principalmente mais favorável para aplicações com alto grau de paralelismo.

Palavras-chave: Processamento de Alto Desempenho, Xeon Phi, Consumo de Energia

Abstract

With computational growth occurring in the last few years, issues and problems related to energy costs arise mainly in large computing environments such as data centers or high-performance processing centers because they require high energy demand to maintain their operation. In the area of High Performance Computing (HPC), following this need arise new computational paradigms to support the exponential increase of processor cores, changing from models and multicore architectures where they centralized the processing only in the CPUs for new heterogeneous architectures and *manycore* that distribute processing using GPUs and coprocessors. The performance relationship with efficient energy consumption in manycore architectures is still a recent issue and there are many questions that need to be explored, because along with the increase in the number of colors have appeared several programming models. Greater study on power consumption is required in models that use and manage part workloads on CPUs and part on GPUs or coprocessors. In this context, the general objective of this work is to perform a study and methodology on the relation between performance and energy consumption in Intel Xeon Phi coprocessors, analyzing and identifying the main factors that directly affect the efficient consumption of energy in environments *manycore* . The evaluation will be based on the behavior and variation of the energy consumption during the execution of the LINPACK, HPL 2.1 and HPCG benchmarks on the programming languages *host*, *offload*, *native* and *symmetric* proposed by Intel. The methodology used consists of scenarios planning, execution, monitoring, collection, measurement and analysis of energy consumption data and performance on the results generated, mainly the definitions of the evaluated criteria, the metrics used and the scenario models used. The main contributions of this work are: Evaluate which programming models provide better results with efficient consumption without compromising performance; Analyze and identify if the behavior of shared memory resources and features, number of nodes, cores, processes and threads directly impact the performance and power consumption relationship. Results show that there is great variation in the performance and energy consumption between the models and the importance of the choice and proper configuration of the factors used during the parallel application processing. Considering shared memory scenarios on the Linpack benchmark, the *host* and *offload* models present a linear increase in performance for workloads (size) up to 10000 and a slight increase after this value. But the *native* model in spite of lower performance in these scenarios, in contrast consumes less energy and mainly more favorable for applications with a high degree of parallelism.

Key-words: High Performance Processing, Xeon Phi, Power Consumption, Performance Analysis

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas

BLAS Basic Linear Algebra Subprograms

COI Coprocessing Offload Infrastructure

EDP Energy-Delay Product

EEA Efficiently Energetic Acceleration

HPCG Gradientes de Conjugado de Alto Desempenho

HPL High Performance Linpack

HPP Heterogeneous Processing Platform

MIC Many Integrated Core

KNC Intel Xeon Phi Knights Corner

KNL Intel Xeon Phi Knights Landing

MKL Math Kernel Library

MPI Message Passing Interface

MPSS Manycore Platform Software Stack

NCC Núcleo de Computação Científica da UNESP

NUMA Nom-uniform Access Memory

PMU Performance Monitoring Unit

RAPL Intel Open Source Running Average Power Limit

SCIF Symmetric Communication Interface

SIMD Single Instruction Multiple Data

SMC System Management Controller

SWaP Space Wattage and Performance

TDP Thermal Design Power

VEth Virtual Ethernet

Sumário

1 INTRODUÇÃO

Na última década houve um crescimento exponencial na demanda por processamento computacional em escala de teraflops nas mais diversas áreas e setores. Isto foi possível devido ao avanço cada vez maior no uso de transistores integrados em chips para prover maior capacidade computacional, mas em contrapartida, o consumo de energia principalmente em *Data Centers* ou ambientes de grande porte elevaram custos com energia a patamares críticos, impactando diretamente nos custos financeiros, inviabilizando manter disponível ambientes escaláveis baseados em arquiteturas homogêneas e apenas multicore. Devido a este cenário preocupante, a indústria de processadores tem apresentado arquiteturas cada vez mais heterogêneas compostas por *hosts* com CPUs *multicore* juntamente com aceleradores gráficos ou coprocessadores *manycore* acoplados e conectados via PCIe, possibilitando processamento de operações ponto flutuante na ordem de teraflops com menor custo e com maior capacidade de paralelismo de tarefas. Além da escalabilidade na capacidade de processamento, outra grande necessidade é o consumo eficiente de energia principalmente em ambientes de alto processamento computacional. A preocupação pela eficiência no consumo de energia proporcionou iniciativas como o surgimento do ranking Green 500 a partir de 2007, criado pela *Stanford Performance Evaluation Corporation* com o objetivo de padronizar e mensurar o consumo eficiente de energia dos supercomputadores participantes do Top 500 (TOP500.ORG, 2017) e comunidade acadêmica, através de pesquisas e proposições específicas para suportar arquiteturas heterogêneas e *manycore* com ênfase em consumo eficiente de energia. Como exemplo do quanto o consumo de energia é elevado, é possível citar o supercomputador Sunway TaihuLight do National Supercomputing Center em Wuxi/China, primeiro na posição do ranking de novembro de 2017 do Top 500 e vigésimo no ranking do Green 500. Ele possui 10.649.600 núcleos de processamento, desempenho (Rmax) de 93.014,6 TFlops, com eficiência de energia de 6.051 GFlops/Watt e consumo de energia de 15.371 kW.

A Intel disponibilizou no mercado seus coprocessadores Intel Xeon Phi em meados de 2012, no qual a partir deste momento surge uma grande necessidade de trabalhos acadêmicos para explorar e apresentar diversos aspectos sobre arquiteturas heterogêneas em escala de teraflops. Até o presente momento existem poucos trabalhos que abordam consumo de energia em ambientes *manycore*, especificamente utilizando os coprocessadores Intel Xeon Phi e mais restritamente ainda abordando os modelos de programação paralela existentes, identificando quais os principais fatores e customizações devem ser consideradas para uma melhor relação desempenho e consumo de energia.

1.1 Objetivos

O objetivo geral deste trabalho é um estudo e avaliação sobre as principais configurações e ambientes que estão diretamente associados a melhoria ou perda na relação desempenho e consumo eficiente de energia em ambientes com coprocessadores Intel Xeon Phi. Como principal resultado deste trabalho é possível apresentar uma metodologia para medir e avaliar o desempenho e eficiência de energia destes coprocessadores. Para auxiliar este estudo e validar as avaliações, foram executados e analisados diversos comportamentos relacionados com desempenho e variação do consumo de energia durante execução dos benchmarks *Linpack*, *HPL 2.1* e *HPCG*.

Os objetivos específicos deste trabalho são:

a) Identificar quais modelos de programação paralela são mais eficientes no consumo de energia sem comprometer performance em ambientes de alto desempenho.

b) Analisar o impacto no consumo de energia nos ambientes de memória compartilhada e distribuída em arquiteturas Intel Xeon Phi.

c) Identificar através de cenários e simulações, qual o impacto existente customizando fatores de paralelismo como número de *nodes*, *cores*, processos e *threads*, na relação desempenho e consumo de energia durante o processamento.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

O Capítulo 2 apresenta fundamentos e conceitos relacionados com desempenho e consumo de energia em HPC. Apresenta a evolução das arquiteturas multicore para manycore, especificando em maiores detalhes a arquitetura Intel MIC e informações sobre coprocessadores Intel Xeon Phi. Também são apresentados neste capítulo aspectos relevantes sobre métricas e consumo de energia em processamento de alto desempenho, descrevendo as principais métricas utilizadas atualmente.

O Capítulo 3 aborda trabalhos já realizados pela comunidade acadêmica relacionados com o tema, de forma que também seja possível entender quais abordagens já foram exploradas sobre consumo de energia e arquiteturas *manycore* e quais diferenciais este trabalho contempla.

O Capítulo 4 descreve a metodologia utilizada neste estudo, que aborda a definição dos benchmarks e softwares, descrição do ambiente de simulação e também a estratégia dos cenários avaliados.

O capítulo 5 descreve os critérios e métricas utilizadas para medir desempenho e consumo de energia.

O Capítulo 6 apresenta em detalhes os resultados das simulações e testes realizados juntamente com as considerações encontradas sobre os questionamentos e objetivos deste trabalho.

Para finalizar, o capítulo 7 apresenta a conclusão sobre este trabalho e sugestões de trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta a evolução da indústria de processadores, também são descritos conceitos e fundamentos sobre arquiteturas *multicore* e *manycore*. Logo após apresenta os modelos de programação paralela proposto pela Intel para melhor produtividade e ganho de desempenho em ambientes que exigem paralelismo. Na segunda parte são apresentados detalhes sobre consumo de energia em ambientes de alto desempenho descrevendo informações sobre as métricas para energia e desempenho utilizados atualmente.

2.1 Arquiteturas Multicore e Manycore

Esta seção apresenta informações sobre arquiteturas *multicore* e *manycore* mais especificamente arquitetura Intel Many Integrated Core (MIC) e coprocessadores Intel Xeon Phi da família *x100* denominado Intel Xeon Phi Knights Corner (KNC).

2.1.1 Evolução das arquiteturas

Arquiteturas computacionais estão se tornando cada vez mais heterogêneas denominadas HPP e compostas por CPUs com núcleos *multicore* integrados através de conexões *PCIe* como coprocessaores Intel Xeon Phi e aceleradores gráficos como GPUs NVIDIA conforme apresentado na figura 1. Estes dispositivos são compostos por núcleos *manycore* e memória local para processar cargas de trabalho que exigem alto processamento computacional. Nos últimos anos houve um grande avanço tornando-se o principal veículo para atender altas demandas por desempenho escalável e eficiência energética principalmente em ambientes que exigem alto desempenho computacional (GIEFERS et al., 2016).

A Intel criou o primeiro supercomputador desenvolvido para o Laboratório Sandia National, o qual era composto por 9000 CPUs e mais de 1000 processadores Pentium PRO e consumo de um megawatt (MATTSON, 2010). Passados dez anos, em 2007 surge o primeiro processador com codinome *"Polaris"* para processamento em escala de *terabytes*, com dimensão de 275mm2 e composto por 1 CPU, 80 cores e consumo de 97 watts.

2.1.2 Arquitetura Intel Xeon Phi

Em 2009, a Intel apresenta uma potencial arquitetura para supercomputação demonstrando um protótipo com processamento de 1 *teraflops* executando o HPC Benck-

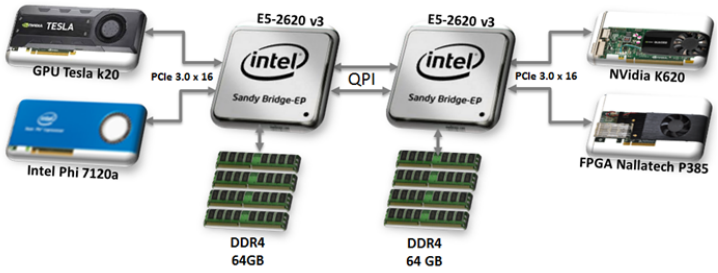
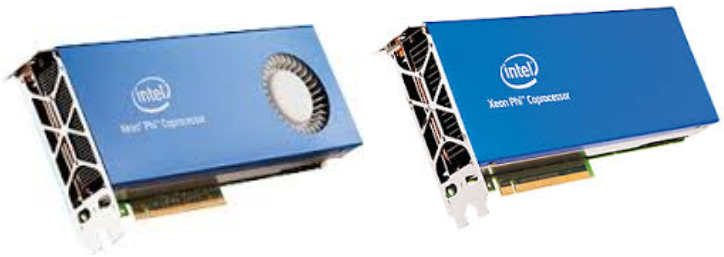


Figura 1 – Arquitetura HPP.

Fonte: (KRISTA, 2015).

mark (SGEMM). Ainda neste ano foi apresentado à imprensa o chip com codinome *Rock Creek* contendo 48 *cores* conectados via uma rede de malha. Em 2010 a Intel descreve esta arquitetura como a primeira de múltiplos núcleos de uso geral pela indústria, descrevendo os primeiros detalhes sobre a MIC como uso do padrão de programação e memória de arquitetura *x86* contendo muito mais *threads*. Surgem, assim, as primeiras versões de coprocessadores Intel Xeon Phi denominados *Knights Ferry* e *Knights Corner* (Rupert Goodwins, 2010). A figura 2 apresenta a versão dos coprocessadores *Knights Corner*.



(a) cooled (b) passive

Figura 2 – Versões dos coprocessadores Intel Xeon Phi *Knights Corner*.

Fonte: (INTEL et al., 2009).

O processador Intel Xeon Phi é um processador host inicializável que oferece paralelismo e vetorização maciça para suportar aplicativos de computação de alto desempenho. A arquitetura integrada e eficiente em energia oferece significativamente mais poder computacional por unidade de energia consumida contra outras plataformas comparáveis que proporcionam um custo bem mais elevado.

Estes coprocessadores são placas adicionais padrão PCI Express que funcionam em sinergia com os processadores Intel Xeon para permitir ganhos de desempenho significativos para código com alto nível de paralelismo até 1,2 teraflops de precisão dupla por coprocessador. Fabricado com a tecnologia de 22 nm da Intel com transistores Tri-Gate 3-D, cada coprocessador apresenta mais núcleos, mais segmentos e unidades de execução de vetor maiores que um processador Intel Xeon. O alto grau de paralelismo compensa

a velocidade mais baixa de cada núcleo para oferecer maior desempenho agregado para cargas de trabalho altamente paralelas (Intel Xeon Phi, 2015). A figura 3 apresenta a microarquitetura dos coprocessadores Intel da família *x100*

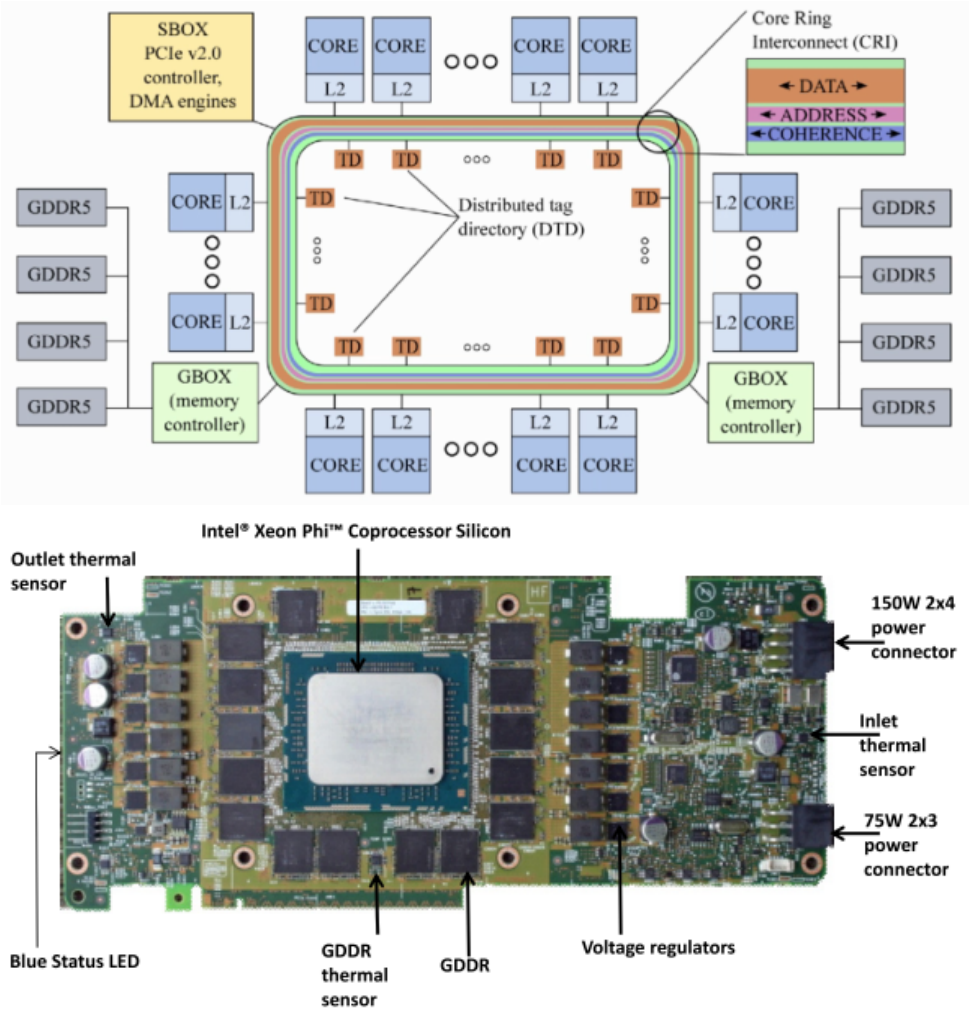


Figura 3 – Arquitetura Intel Xeon Phi Familia x100.

Fonte: (Intel Xeon Phi, 2015).

Os coprocessadores Intel Xeon Phi foram criados para otimizar a relação desempenho por watt para aplicações com cargas de trabalho (*workloads*) em ambientes com elevado paralelismo de cores. A Intel disponibiliza um conjunto de ferramentas de software e arquitetura *manycore* com suporte Single Instruction Multiple Data (SIMD). A versão KNC suporta até 61 cores de 1,2GHz com 4 *threads* físicas por *core* proporcionando a execução de até 244 *threads*. Executa instruções SIMD com tamanho de 512 *bits* com 32 registradores nativos no qual suporta desempenho máximo teórico com precisão dupla em até 1 *teraflop/s*. Suporta maior largura de banda de memória comparado à família de processadores para servidores modelo Intel E5. Sua arquitetura possui dois *pipelines* sendo uma unidade de processamento escalar com suporte a processadores antecessores baseados em *Pentium* e outra unidade vetorial para suporte SIMD (CHRYSOS, 2012).

A tabela 1 apresenta as versões disponíveis na família *x100*.

Modelo	**C/T**	**Frequência**	**L2 Cache**	**Memoria**	**TDP (W)**	**Refrigeração**
3110X	61 (244)	1053 MHz	30.5 MB	6/8 GB	300	Bare Board
3120A	57 (228)	1100 MHz	28.5 MB	6 GB	300	Fan/Heatsink
3120P	57 (228)	1100 MHz	28.5 MB	6 GB	300	Passive Heatsink
31S1P	57 (228)	1100 MHz	28.5 MB	8 GB	270	Passive Heatsink
5110P	60 (240)	1053 MHz	30 MB	8 GB	225	Passive Heatsink
5120D	60 (240)	1053 MHz	30 MB	8 GB	245	Bare Board
SE10P	61 (244)	1100 MHz	30.5 MB	8 GB	300	Passive Heatsink
SE10X	61 (244)	1100 MHz	30.5 MB	8 GB	300	Bare Board
7110P	61 (244)	1250 MHz	30.5 MB	16 GB	300	Passive Heatsink
7110X	61 (244)	1250 MHz	30.5 MB	16 GB	300	Bare Board
7120A	61 (244)	1238 MHz	30.5 MB	16 GB	300	Fan/Heatsink
7120D	61 (244)	1238 MHz	30.5 MB	16 GB	270	Bare Board
7120P	61 (244)	1238 MHz	30.5 MB	16 GB	300	Passive Heatsink
7120X	61 (244)	1238 MHz	30.5 MB	16 GB	300	Bare Board

Tabela 1 – Versões de Coprocessadores Intel Xeon Phi *x100*.

Fonte: (Intel Corp, 2015).

O coprocessador Intel Xeon Phi é uma placa PCIe que foi projetada para ser instalada em plataformas Intel. A figura 4 apresenta um exemplo típico de configuração de uma *workstation* Intel contendo duas CPUs no *host* e dois coprocessadores Xeon Phi.

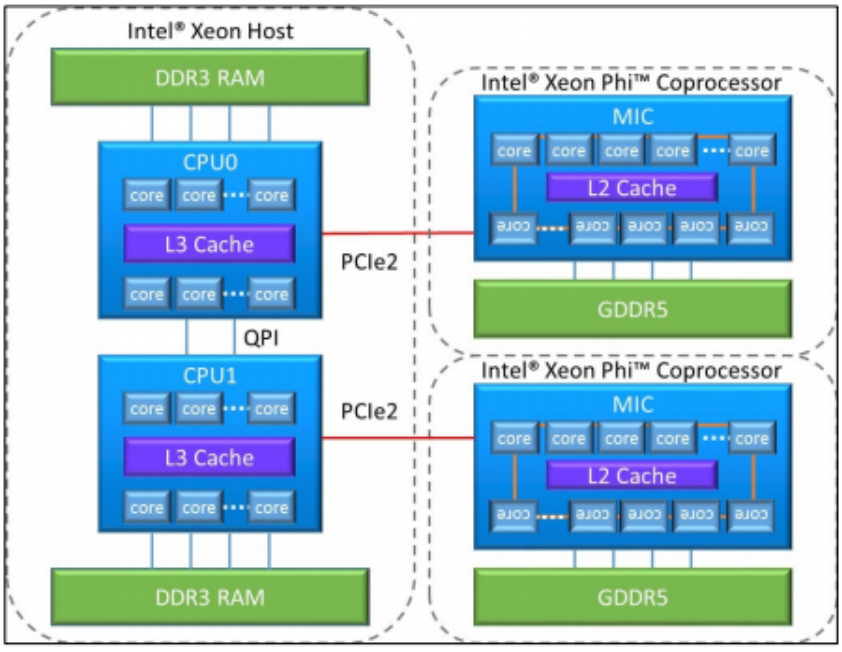


Figura 4 – Configuração típica de uma *workstation* Intel Xeon Phi.

Fonte: (MPSS, 2014a).

Quando um ou mais *hosts* são conectados através de canais InfiniBand, tal como Intel True Scale HCAs, então coprocessadores podem se comunicar em alta velocidade com outros processadores ou coprocessadores localizados em outros *nodes*, formando um

cluster. Inclusive coprocessadores podem se comunicar *peer-to-peer* sem intervenção do *host* que está hospedado. A figura 5 apresenta este modelo baseado em *cluster*.

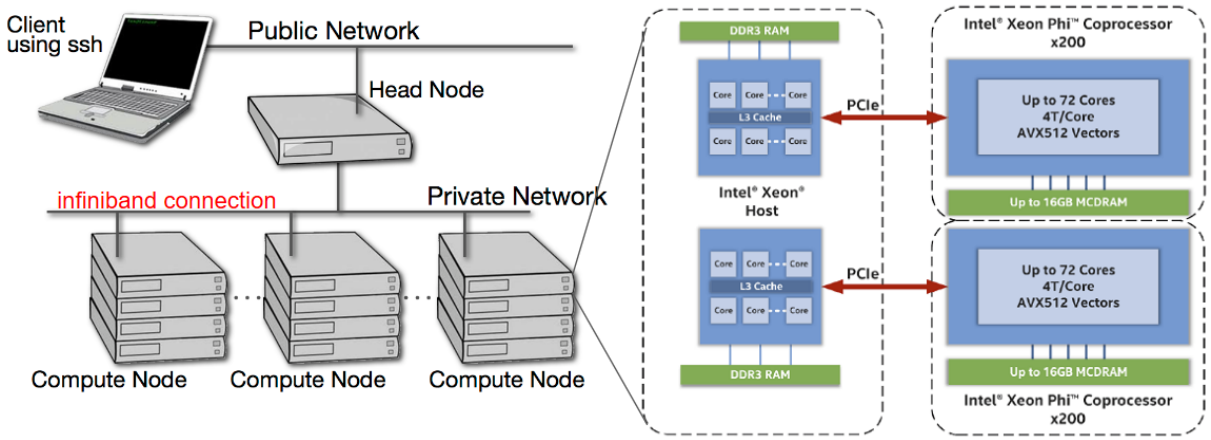


Figura 5 – Configuração típica de um *node* Intel Xeon Phi em Cluster.

Fonte: (MPSS, 2014a).

No ambiente de simulações utilizado neste trabalho o modelo de coprocessador Intel Xeon Phi utilizado foi o 7120P. Este modelo é um *Knights Corner* do tipo *server* com litografia 22nm que possui: 61 núcleos físicos e 244 threads físicas, frequência de 1,24GHz e turbo max de 1,33GHz com 16GB de memória do tipo GDDR5.

Fisicamente os recursos mecânicos do coprocessador Intel Xeon Phi são compatíveis com a placa eletrônica PCI Express 225W / 300W *High Power Card Electromechanical Specification 1.0*. Sob o aspecto elétrico e térmico do Intel Xeon Phi, existe o System Management Controller (SMC), sensores térmicos e também um ventilador. A figura 6 apresenta a especificação da placa e arquitetura de comunicação entre o processador, memória e o mecanismo de refrigeração.

A regulagem de tensão é feita pelos reguladores de tensão (VRs) alimentados pela placa-mãe através dos conectores PCI Express e um conector de alimentação auxiliar de 2x4 (150W) e 2x3 (75W) na borda da placa. Além destes conectores, também são precisos conectores *SKUs* de 300W tamanhos 2x4 e 2x3 para serem alimentado pelas fontes de energia do sistema. O *SKU* 225W pode ser alimentado apenas através do conector PCI Express e Conector 2x4 (INTEL, 2015).

Os parâmetros disponíveis para monitoramento térmico do Xeon Phi estão apresentados na tabela 2. O $T_{control}$ é o ponto de ajuste no qual os ventiladores do sistema devem funcionar para manter a temperatura do coprocessador. Ele é requisito para que o sistema BMC use comandos IPMB para consultar a SMC para um valor do $T_{control}$ preciso e possa variar somente entre 80 e 84 graus *celsius*. Quando a temperatura de junção do coprocessador $T_{junction}$ atingir $T_{throttle}$, o SMC vai acionar o ventilador térmico, que deverá reduzir a frequência ao menor valor suportado e reduzir o consumo total de energia do

Parâmetro	**Especificação**
T_{RISE}	$10°C$
Max T_{INLET}	$45°C$
Max T_{EXAUST}	$70°C$
Tcase(processor) min, max	$5°C, 95°C$
$T_{control}$	$82°C$
$T_{throttle}$	$104°C$
$T_{thermtrip}$	$(T_{throttle} + 20°C)^3$

Tabela 2 – Especificação térmica do coprocessador Intel Xeon Phi.

Fonte: (INTEL, 2015).

coprocessador. Se a temperatura do coprocessador atingir $T_{thermtrip}$, o sistema operacional do coprocessador tomará medidas para desligar a placa através do desligamento dos VRs, para evitar maiores danos no coprocessador.

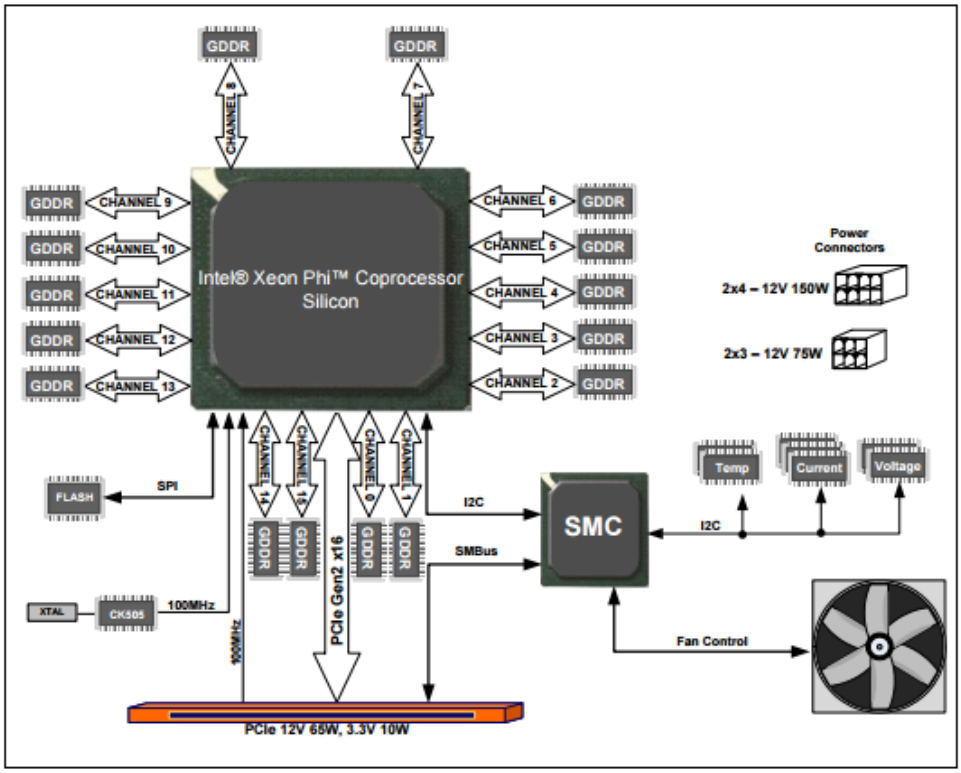


Figura 6 – Especificação da placa do coprocessador Intel Xeon Phi.

Fonte: (INTEL, 2015).

2.1.3 Modelos de Programação Paralela

Os Modelos de Programação Paralela da Intel estão divididos em dois grupos. O primeiro grupo suporta modo explícito usando memória virtual compartilhada e basicamente por chamadas usando *OpenMP* e através do uso de *runtime offload libraries* existentes na *MPSS*. Neste modo o modelo de programação paralela disponível é o *offload*. O segundo grupo permite a execução de aplicações *standalone* (compiladas para arquite-

tura *mic*) ou aplicações distribuídas através de processos MPI que utilizam conexões *ssh* e protocolos TCP/IP entre a comunicação *host* e coprocessadores Intel Xeon Phi. Neste modo os modelos disponíveis são o *nativo*, *simétrico* e *host*.

A figura 7 apresenta a transição dos modelos de programação paralela (*host*, *Offload*, *simétrico* e *nativo*) baseados em arquiteturas *multicore* homogêneas evoluindo para modelos baseados em *manycore*.

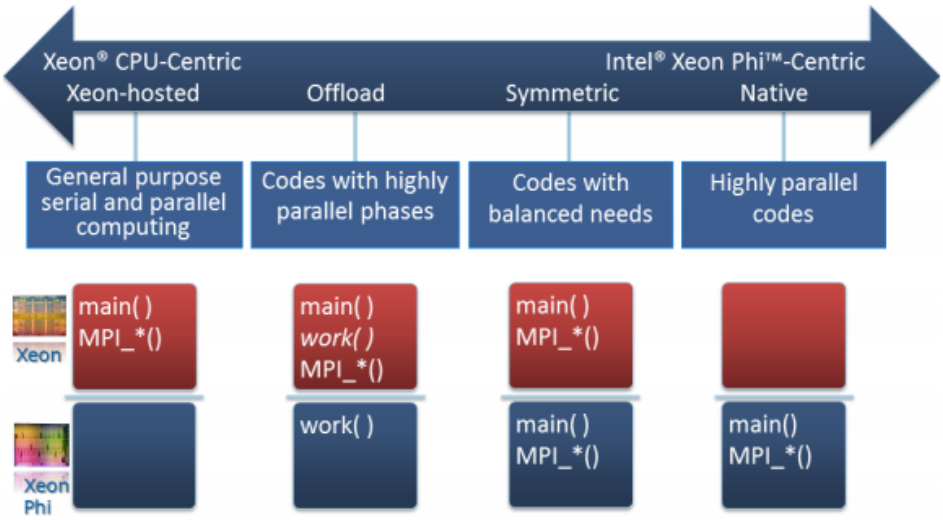


Figura 7 – Modelos de Programação Paralela da Intel.

Fonte: (KARPUSENKO, 2013).

1. Modelo ***host*** - Este modelo é utilizado para processamento serial ou paralelismo centralizado no *host* cujo comportamento demanda maior latência de memória.

   A figura 8 apresenta um exemplo de código em linguagem C que pode ser compilado para execução no *host* ou *nativo* sem nenhuma alteração de código, apenas diretivas e parâmetros durante a compilação.

```c
#include <stdio.h>
#include <unistd.h>
int main(){
  printf("Hello world! I have %ld logical cores.\n",
  sysconf(_SC_NPROCESSORS_ONLN ));
}
```

Figura 8 – Exemplo de Código em Linguagem C para execução *standalone*.

Fonte: (KARPUSENKO, 2013).

2. Modelo ***offload*** - O modelo *offload* também demanda maior latência de memória e barramento e possui a característica de somente terminar a execução quando não houver mais blocos de cargas de trabalho (*workloads*) em execução nos coprocessadores. Este modelo é adequado para aplicações com parte serial e blocos paralelos

sob demanda. Como exemplo, um ou mais processos de uma aplicação são iniciados em um ou mais processadores em um *host* Intel. Estes processos, representados por *main()* na figura 7, podem compartilhar a carga de trabalho, representada por *work()* nesta figura, com um ou mais coprocessadores conectados, aproveitando melhor o uso de outros núcleos de processamento com suporte a unidades vetoriais e larga banda de memória. Para os casos de aplicações com mais de um processo, os processos geralmente se comunicam usando mensagens via Message Passing Interface (MPI), representado na figura por *MPI_*()*, no *host*. Este processo de compartilhar carga de trabalho pode ser também programado explicitamente em códigos C/C++ ou Fortran, pois são suportados pelos compiladores (MPSS, 2014a).

A figura 9 apresenta um exemplo em C de chamada *pragma* para coprocessadores de forma geral sem especificar qual coprocessador utilizar.

```c
#include <stdio.h>
int main(int argc, char * argv[] ) {
  printf("Hello World from host!\n");
#pragma offload target(mic)
    {
      printf("Hello World from coprocessor!\n"); fflush(0);
    }
  printf("Bye\n");
}
```

Figura 9 – Exemplo de Código em Linguagem C para execução *offload* explícita.

Fonte: (KARPUSENKO, 2013).

3. Modelo ***simétrico*** - Neste modelo, ambos *host* e coprocessadores Intel Xeon Phi executam a mesma aplicação HPC composta por vários processos, no qual utilizam comunicação padrão através do uso de processos MPI. Neste modelo, as cargas de trabalho não são compartilhadas através de chamadas *pragma* mas sim permanecem dentro de cada processo que contém a aplicação. As aplicações devem ser compiladas uma versão para *host* e outra para *nativo* para coprocessadores Xeon Phi (MPSS, 2014a), (KARPUSENKO, 2013).

4. Modelo ***nativo*** - Este modelo é uma variante do modelo simétrico, que permite que aplicação execute um ou mais processos somente no coprocessador não dependendo do *host*, dependendo apenas dos *drivers* Symmetric Communication Interface (SCIF) e Virtual Ethernet (VEth) (MPSS, 2014a). Para isso a aplicação precisa ser compilada com suporte nativo do Intel Xeon Phi executado *standalone* diretamente no sistema operacional linux existente no coprocessador ou através de processos MPI pelo *host*.

Toda integração e comunicação entre *host* e coprocessadores Intel Xeon Phi ocorre através da Manycore Platform Software Stack (MPSS). A comunicação neste modo entre

host é chamado de SCIF na camada de aplicação e Coprocessing Offload Infrastructure (COI) via barramentos PCIe na camada de sistema e hardware. Esta plataforma de comunicação ocorre através do protocolo TCP/IP entre o *host* e os coprocessadores Xeon Phi utilizando SSH. Isto é possível pois internamente no coprocessador existe uma instância do sistema operacional Linux nativo no dispositivo. A figura 10 apresenta em detalhes as camadas e comunicação entre o *host* e Xeon PHI (MPSS, 2014b).

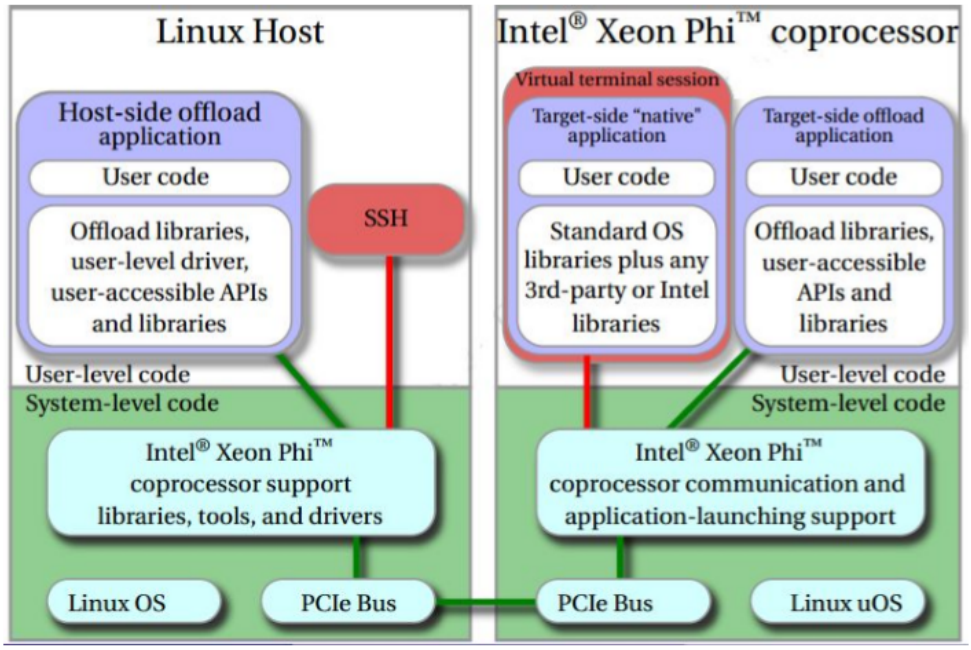


Figura 10 – Arquitetura Heterogênea CPU + Coprocessador da Intel.

Fonte: (MPSS, 2014b).

2.2 Métricas de Desempenho e Energia

Esta seção apresenta um levantamento e classificação das principais métricas relacionadas com desempenho e energia atuantes em HPC. Também descreve o processo de medição de consumo de energia utilizado atualmente pelo Green 500, órgão referência em HPC para avaliação de eficiência em energia assim como o Top 500 para desempenho.

2.2.1 Consumo de Energia em Ambientes de Alto Desempenho

A *Stanford Performance Evaluation Corporation* descreve vários aspectos relacionados com recursos computacionais que devem ser considerados quando se deseja medir o consumo de energia em HPC. O documento *Energy Efficient High Performance Computing Power Measurement Methodology* criado de forma colaborativa por Green 500, Top 500 e o grupo de trabalho Energy Efficient High Performance Computing Working Group, apresenta uma metodologia detalhada para consumo eficiente e métricas direcionadas a

computação de alto desempenho considerando o benchmark HPL Linpack (GREEN500, 2010). Nesta metodologia estão definidos os níveis de qualidade do L1 (adequado) até o L3 (ótimo) e também se a medição realizada é internamente nos componentes ou externamente abstraindo o sistema e seus dispositivos (subsistemas). O principal documento gerado por esta metodologia é o relatório de consumo de energia e deve estar disponível no *ranking* do Green 500.

2.2.2 Monitoramento Energético Externo

Existem diversos produtos e também trabalhos acadêmicos como (BARRACHINA et al., 2013), (ROSTIROLLA et al., 2015) e (IGUAL et al., 2015), que apresentam soluções para medição de energia com *power meters*, que são equipamentos físicos acoplados externamente aos componentes para medir toda energia diretamente consumida pela fonte de alimentação de energia.

Este trabalho não abordará avaliação de consumo de energia através de medidores externos, pois o objetivo é avaliar restritamente via software quais fatores de arquitetura e ambiente internos em processadores, CPUs e memória no sistema de processamento, que possam influenciar ou otimizar o consumo de energia.

2.2.3 Flops por Watt

Conforme descrito por (SHARMA; HSU, 2006), por décadas a noção de desempenho era sinônimo de velocidade e medida através da métrica "Flops", no qual seu objetivo é medir a quantidade de operações de ponto flutuante por segundo.

O foco por maior desempenho tem levado ao surgimento de supercomputadores com elevado consumo de energia e também necessitando de instalações com maior estrutura de refrigeração. Aliado a esta demanda para medição de energia e também outras métricas que não a velocidade, a Top 500 e outros órgãos desenvolveram um esforço e sensibilidade para melhor avaliar ambientes HPC e auxiliar principalmente no aumento significativo de custos relacionados com energia. Com isto surge a necessidade de disponibilizar uma lista no qual além do interesse na métrica de performance também se considera a eficiência no consumo de energia, propondo a lista Green500 e uma nova linha para discussões de potenciais métricas que podem ser utilizadas para sistemas de supercomputadores.

A métrica *Flops por Watt* proposta por (SHARMA; HSU, 2006) é a mais utilizada para medir desempenho por consumo de energia. No entanto, outras métricas foram definidas. Por exemplo, a Sun Microsystems define a métrica *Space Wattage and Performance (SWaP)* que considera a medida do espaço para calcular a eficiência energética de um computador (BALLADINI et al., 2011). Outra métrica disponível é Energy-Delay

Product (EDP) que resulta o montante de energia consumida durante em determinado período de tempo (SUBRAMANIAM; FENG, 2010).

A equação 2.1 é e definição física para energia.

$$E = \int P(t)\ dt \tag{2.1}$$

(ROSTIROLLA et al., 2015) também descreve através da interpretação da equação do consumo de energia sobre o tempo conforme a figura 11:

$$E = \sum_{i=1}^{N} P(i)\ *\ \Delta t(i) \tag{2.2}$$

No qual:

- i é o número de elementos
- N é o número de CPUs

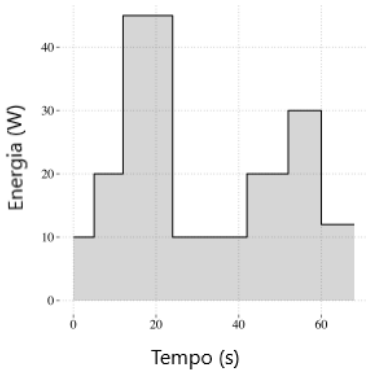


Figura 11 – Exemplo de Energia por Tempo.

Baseado na equação 2.1, a equação de Desempenho por Energia (*Performance per Watt*) é:

$$PPW = \frac{Floating\ Point\ Operations/Second}{Joules/Second} = \frac{FLOPS}{W} \tag{2.3}$$

Por exemplo, se considerarmos o resultado de uma execução do benchmark *Linpack* que é em GFlops com valor de 4000 GFlop/s e que durante a execução da carga de trabalho consumiu 400W então teremos:

$$PPW = \frac{4000\ GFLOPS}{400\ W} = 10 GFLOPS/W \tag{2.4}$$

2.2.4 Potência Ociosa (Idle Power)

Um processador de computador é descrito como ocioso quando não está sendo usado por nenhum programa. Todo programa ou tarefa que é executado em um sistema de computador ocupa uma certa quantidade de tempo de processamento na CPU. Se a CPU tiver concluído todas as tarefas, ela ficará inativa(*idle*). Processadores modernos usam o tempo ocioso para economizar energia. Métodos comuns estão reduzindo a velocidade do clock junto com a tensão da CPU e enviando partes do processador para um estado de suspensão. Em processadores que têm uma instrução de interrupção que interrompe a CPU até que ocorra uma interrupção, como a instrução HLT do x86, ela pode economizar quantidades significativas de energia e aquecimento se a tarefa ociosa consistir em um loop que executa repetidamente instruções HLT. Um coprocessador Intel Xeon Phi geralmente dissipa uma média de 94.05W quando está no estado ocioso (BALLADINI et al., 2011).

2.2.5 Desempenho Máximo de Processamento

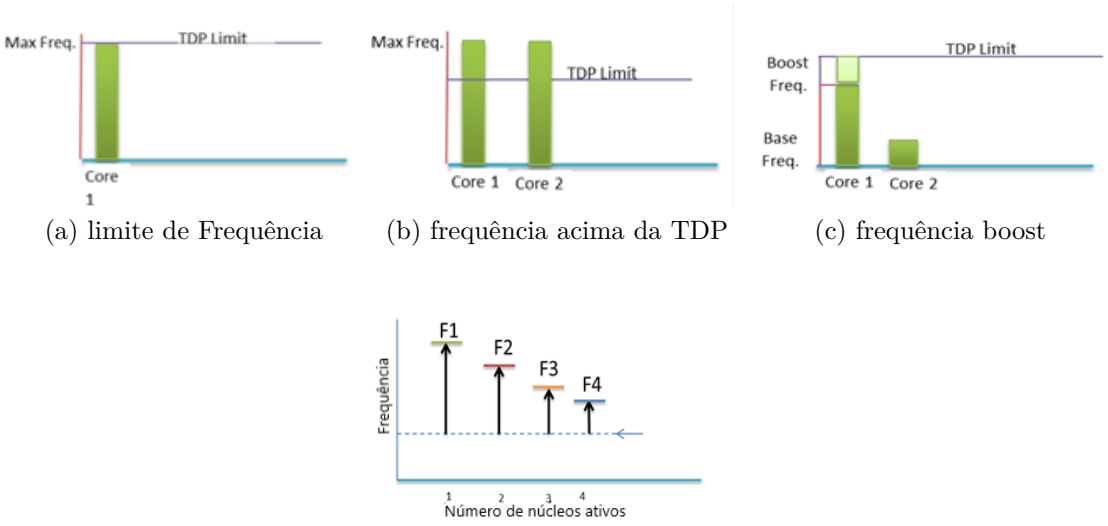
Na computação de alto desempenho, $R_(max)$ e $R_(peak)$ são pontuações usadas para classificar supercomputadores com base em seu desempenho usando o *Linpack* Benchmark. A pontuação $R_(max)$ de um sistema descreve o desempenho máximo alcançado; A pontuação $R_(peak)$ descreve seu desempenho máximo teórico. Os valores para ambos os escores são geralmente representados em *Flops*.

2.2.6 Thermal Design Power (TDP)

O Thermal Design Power (TDP), é a quantidade máxima de calor permitida para um processador ou componente como uma CPU ou coprocessador dissipar durante seu uso. Em vez de especificar a dissipação real da CPU, o TDP serve como referência de um valor limite nominal sugerido pelos fabricantes, por exemplo, para um processador com TDP de 35W, a Intel informa através de OEM que, se implementar um sistema de resfriamento capaz de dissipar calor até este limite o chip funcionará normalmente. É possível o processador consumir mais energia do que o limite suportado pelo TDP por um curto espaço de tempo sem que seja "termalmente significativo", pois o calor levará algum tempo para se propagar não violando necessariamente o TDP (Intel RAPL, 2014).

A figura 12 apresenta a relação do TDP com frequência, desempenho e consumo de energia. Na figura 12a, com uma CPU com único núcleo (*core*), uma aplicação demandando desempenho e utilizando toda capacidade e frequência até os limites permitidos pela TDP. Mas conforme apresentado nas figuras 12b e 12c, observa-se que se mais um núcleo for adicionado então a carga de trabalho *multi-thread* poderá exigir desempenho máximo em ambos núcleos excedendo o limite do TDP e consumindo mais energia. Para evitar isso, conforme apresentado na figura 12d, a CPU usará frequências máximas dife-

rentes dependendo do número de núcleos ativos (Intel RAPL, 2014).



(a) limite de Frequência (b) frequência acima da TDP (c) frequência boost

(d) diferentes freq máximas

Figura 12 – Influência do TDP no desempenho e consumo de energia.

Fonte: (Intel RAPL, 2014).

2.3 Considerações Finais

Este capítulo apresentou conceitos e fundamentos relacionados com computação de alto desempenho, evolução das arquiteturas multicore e manycore, detalhes da arquitetura Intel MIC e quais os modelos de programação da Intel a serem considerados neste trabalho. Também neste capítulo foram apresentadas informações sobre a metodologia de avaliação de consumo de energia do Green500 e fundamentos das métricas de desempenho e energia utilizados neste trabalho.

3 Trabalhos Relacionados

Este capítulo apresenta os trabalhos mais relevantes relacionados com estudos, proposições e técnicas para consumo de energia eficiente. Os trabalhos citados estão agrupados em tópicos para melhor estruturação e identificação por contexto.

Durante esta etapa do trabalho, a metodologia utilizada foi inicialmente exploratória, na qual o objetivo principal foi investigar utilizando procedimentos de pesquisa bibliográfica para encontrar e classificar os trabalhos realizados e relacionados com o tema deste trabalho nos últimos anos, e posteriormente descritiva, relatando as características e fatos dos trabalhos mais relevantes buscando comparar resultados e conexões com este trabalho.

Até o momento a indústria de processadores mais especificamente a Intel, não disponibiliza um modelo ou metodologia eficiente que contemple claramente quais os principais fatores que influenciam no consumo de energia eficiente sobre arquitetura Intel Xeon Phi. De certa forma isto também veêm incentivando a comunidade acadêmica na realização de trabalhos com proposições ou caracterização sobre este tema, pois o custo de energia ainda é elevado em ambientes de alto desempenho.

3.1 Trabalhos realizados e relacionados com este tema

A caracterização do consumo de energia em GPUs ou coprocessadores é importante, pois identifica e classifica os fatores relacionados ao consumo. O trabalho de (SHAO; BROOKS, 2013) têm como objetivo demonstrar um modelo detalhado de consumo de energia a nível de instrução em processadores Intel Xeon Phi. As principais contribuições de seu trabalho são: A caracterização de energia por instrução (EPI), utilizando um conjunto de micro benchmarks especializados que analisam diferentes categorias de instruções considerando variados comportamentos de memória, número de cores ativos e número de threads por core; Construir um modelo de energia baseado em nível de instrução utilizando a caracterização EPI ao longo de contadores estatísticos de desempenho para capturar a carga de trabalho (*workload*) das atividades executadas. Este modelo prevê uma média de erros abaixo de 5%. Disponibilizar aos desenvolvedores a oportunidade de melhorar eficiência energética em seus códigos com a redução de energia podendo chegar acima de 10% devido a operações *prefetch* redundantes em implementação customizada do *Linpack*.

Arquiteturas heterogêneas cresceram muito com o surgimento de *manycore*. Elas têm como objetivo o compartilhamento de recursos e processamento distribuído entre *no-*

des, *host* e coprocessadores. Já o modelo híbrido, combina além de recursos distribuídos também o uso e combinação de múltiplos modelos de comunicação como MPI e OpenMP ou entre diferentes modelos de programação como CUDA para GPUs e Pthreads, C++, TBB entre outros para Intel Xeon Phi. O trabalho realizado por (LAKOMSKI; ZONG; JIN, 2015) apresenta um estudo detalhado para compreender o melhor balanço entre desempenho e consumo de energia em aplicações computacionais híbridas. Um dos principais problemas identificado neste trabalho é o desperdício de ciclos computacionais e também energia quando CPUs estão ociosas aguardando a finalização dos processos em execução nos dispositivos como por exemplo Xeon Phi ou outro acelerador. Isto ocorre em grande parte devido a prática geral de uso do modelo de programação *offload* que apesar de paralelizar o código para aceleradores permanece a principal execução na CPU. Um exemplo pode ser visto na figura 13, no qual a CPU desperdiçou 916 joules de energia enquanto aguardava a *Matrix Multiplication* (MM) e 1035 *joules* enquanto aguardava *Fractal*(F) serem finalizados no Xeon Phi. (LAKOMSKI; ZONG; JIN, 2015) também apresenta nesta figura que o consumo de energia é inferior quando executado em GPU e também usando *OpenCL*.

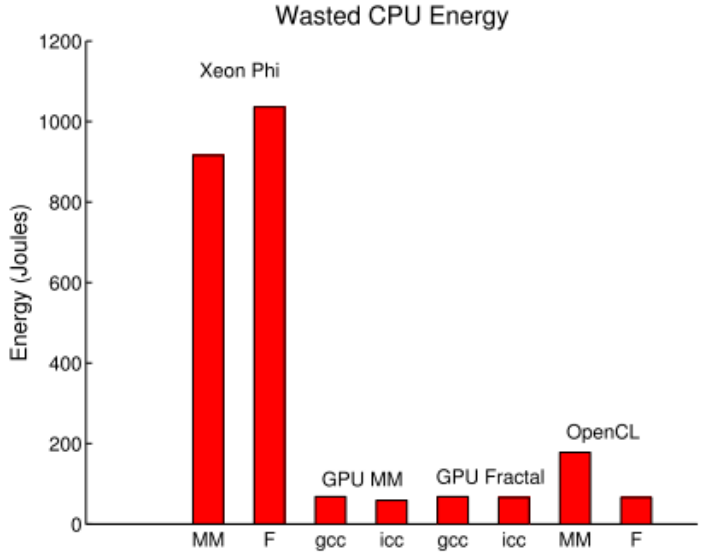
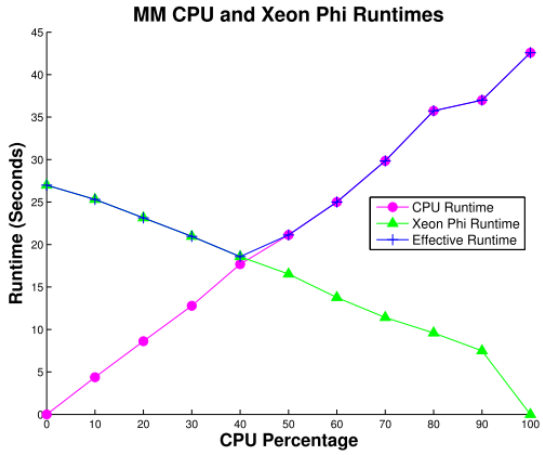


Figura 13 – Desperdício de Energia pela CPU aguardando dispositivos.

Fonte: (LAKOMSKI; ZONG; JIN, 2015).

Para remediar este problema (LAKOMSKI; ZONG; JIN, 2015) propõe um modelo de programação híbrida permitindo a CPU (*host*) executar blocos de código enquanto aceleradores (GPU ou Xeon Phi) também execute outros blocos de código de forma independente. Desta forma permitirá além do aumento de desempenho também melhorar a eficiência no consumo de energia. No experimento foram customizadas as aplicações *Matrix Multiplication* e *Fractal* para executar de forma concorrente e suportar o modelo híbrido considerando sistemas heterogêneos CPU + GPU ou CPU + Xeon Phi. A

definição do percentual de divisão entre CPU ou dispositivo é através de parâmetro de execução. Para todos testes realizados a técnica de medição utilizada foi através da coleta dos dados de consumo de energia da CPU via interface do módulo Marcher, uma extensão do RAPL. A coleta dos dados de energia do Xeon Phi foi realizada via interface Intel micsmc. A figura 14 apresenta o gráfico de execução da MM sobre CPU/Xeon Phi que utiliza como parâmetro 40% para CPU e 60% para Xeon Phi o melhor ponto da relação desempenho e consumo de energia. Como conclusão (LAKOMSKI; ZONG; JIN, 2015) descreve que a relação entre desempenho e consumo de energia são objetivos conflitantes e dependem muito das características das aplicações, sendo altamente dependente de suas implementações de código.



CPU %	CPU Runtime [Seconds]	CPU Energy [Joules]	PHI Runtime [Seconds]	PHI Energy [Joules]	Effective Runtime [Seconds]	Total Energy [Joules]
100	42.59	7918	0*	0*	42.59	7918
90	37.00	6909	7.49	1151	37.00	8090
80	35.74	6395	9.59	1539	35.74	7935
70	29.84	5670	11.41	1859	29.84	7529
60	25.00	4798	13.76	2327	25.00	7126
50	21.12	3956	16.52	2939	21.12	6896
40	17.69	3356	18.57	3297	18.57	6564
30	12.79	2873	20.96	3692	20.96	6566
20	8.62	2412	23.14	4207	23.14	6620
10	4.38	1991	25.31	4623	25.31	6614
0	0	1429	26.99	4795	26.99	6224

*Application is not active on Xeon Phi

Figura 14 – Resultados da Matrix Multiplication sobre CPU/Xeon Phi.

Fonte: (LAKOMSKI; ZONG; JIN, 2015).

Outro trabalho relacionado com arquiteturas híbridas e ambientes *manycore* é o de (SI et al., 2014). Apresenta uma proposta da MT-MPI, uma implementação para coordenação de *multithreads* internamente em MPI em conjunto com OpenMP. Demonstra os benefícios do uso eficiente de paralelismo interno durante processamento MPI sob vários aspectos, incluindo comunicação de memória compartilhada, tipos de dados derivados e operações de I/O pela rede. Seu trabalho também considera especificamente arquiteturas Intel Xeon Phi, com aplicações executando nativamente em coprocessadores.

Existem inúmeros trabalhos relacionados ou propostas de algoritmos no uso de frequência e voltagem dinâmicos (DVFS) para redução do consumo de energia. (HSU; KREMER, 2003) apresenta um compilador para otimizar programas de forma a considerar DVFS para reduzir seu consumo de energia. Seu algoritmo identifica regiões no qual a CPU pode ser reduzida sem comprometer perda de desempenho. Outro trabalho é o (BALLADINI et al., 2011), que avalia diferentes frequências de clock de CPUs para eficiência no consumo de energia sobre paradigmas de programação. Em seu trabalho utiliza OpenMP em memórias compartilhadas e MPI para memória distribuída. Trabalhos mais recentes estão relacionados a eficiência energética nos processadores Intel Xeon Phi

Knights Landing (KNL). O trabalho de (LAWSON et al., 2016), pretende estabelecer os limites de energia de acordo com as características da carga de trabalho e o desempenho da aplicação. Seu trabalho enfatiza como a frequência de operação varia no Xeon Phi influenciando dentro dos limites permitidos. O modelo de consumo de energia utilizado define o consumo baseado no consumo estático e o número de núcleos físicos é o fator que varia de acordo com a carga de trabalho.

Uma das técnicas muito utilizadas para otimização de desempenho é o uso de *Thread Affinities*, pois faz o balanceamento de carga de trabalho e permite uso eficiente de *threads* entre os *cores* durante o processamento computacional. Um dos trabalhos pioneiros no uso desta abordagem para avaliar consumo de energia foi realizado por (LAWSON; SOSONKINA; SHEN, 2014). Como principal objetivo, ele investiga os efeitos variando opções existentes em *thread affinities* sobre coprocessadores Intel Xeon Phi considerando a redução de utilização de *cores* para avaliar o consumo de energia e tempo de execução. Existem seis possibilidades em thread affinities para o mapeamento de threads por core: *balanced, compact, scatter, none, disabled* e *explicit*. As opções *disabled* e *explicit* não foram consideradas neste trabalho. A opção *balanced* eventualmente distribui threads entre todos cores disponíveis. A opção *scatter* distribui as threads sob a forma de rodizio entre os cores. A opção *compact* distribui considerando o número máximo de threads por core que neste caso são quatro. O benchmark utilizado é o *NASA Advanced Supercomputing (NAS) Parallel Benchmarks* customizado para executar nativamente em coprocessadores Intel Xeon Phi. O benchmark *NBP* disponibiliza um conjunto de aplicações com variados esquemas computacionais e de comunicação. Estes esquemas estão classificados através de categorias de classes como A, B, C, D e E em ordem ascendente conforme o tamanho do problema. Neste trabalho (LAWSON; SOSONKINA; SHEN, 2014) utiliza os benchmarks denominados *EP* (operações com ponto flutuante), *CG* (vetor e matrix de multiplicação), *FT* (comunicação de dados), *IS* (velocidade de operações com inteiros) e *MG* (comunicação de dados em curtas distâncias) especificamente para refletir maior processamento similar ao utilizado no mundo real em aplicações científicas ou de engenharias. As métricas utilizadas foram: 1) Para desempenho, foram extraídas de contadores e eventos de hardware disponibilizados através da ferramenta Intel VTune Amplifier XE 2013. Particularmente estas métricas são a média de CPU por *Thread*, Intensidade de Vetorização e Largura de banda. 2) Para Consumo de energia, foram obtidas através de energia consumida registrado e disponibilizado em intervalos de tempo pelas ferramentas de monitoramento da Intel *micsmc* e também com o uso de dados coletados através de dispositivo via Performance Monitoring Unit (PMU) disponível em cada core independentemente.

Conforme apresentado na figura 15, os resultados obtidos por (LAWSON; SOSONKINA; SHEN, 2014) demonstram que o modo *compact* apresenta pouco desempenho até 180 threads, mas acima disto possui excepcional ganho. Os modos *balanced, scatter* e

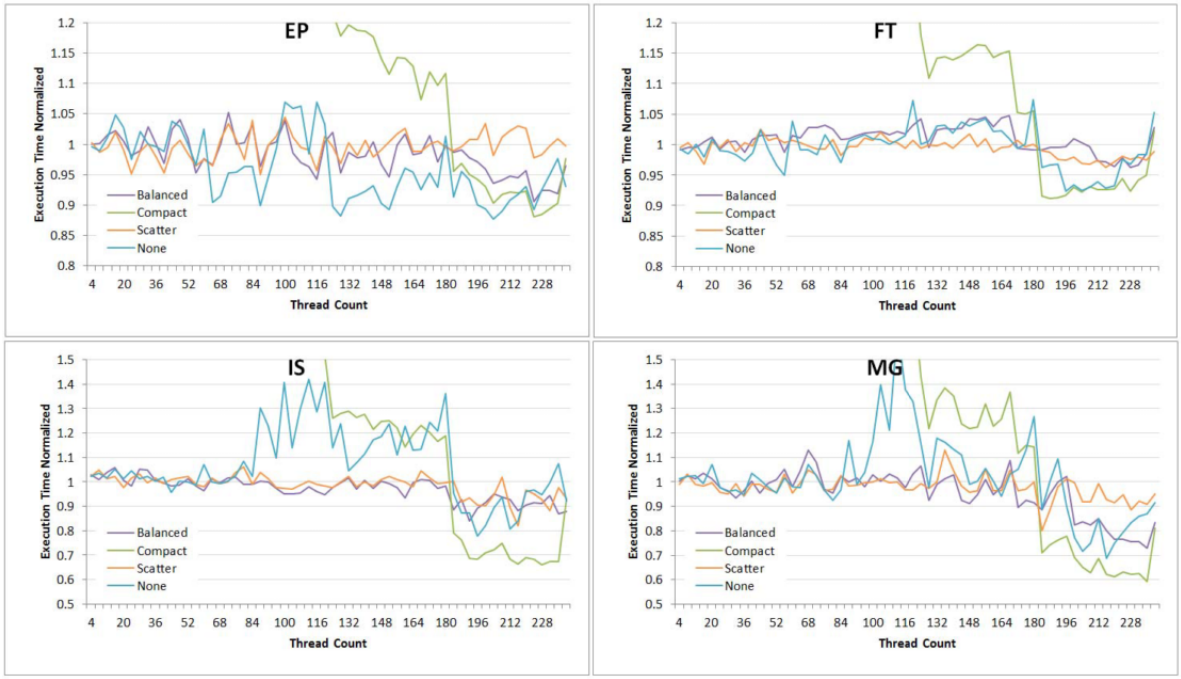


Figura 15 – Resultados de threads por tempo de execução para os benchmarks EP, FT, IS e MG com diferentes modos de thread affinities.

Fonte: (LAWSON; SOSONKINA; SHEN, 2014).

none oscilam de forma caótica entre os valores maiores e menores que o normal. De forma geral, na figura acima é possível perceber que o uso de thread affinities pode reduzir o consumo de energia para uso acima de 180 threads.

O trabalho de (LORENZON; CERA; BECK, 2015) baseado na métrica EDP, avalia e apresenta diferentes níveis de consumo de energia em sistema embarcado de propósito geral sobre interfaces de programação paralela. Este trabalho destaca o consumo de energia durante a execução dos benchmarks *Game of Life* (GL), *Gram-Schmidt* (GS), *LU-Decomposition* (LU), *Matrix Multiplication* (MM), *Calculation of the PI Numbe*r (PI), *Mandelbrot Set* (MS), *Dijkstra* (DJ) e *Similarity of Histograms* (HS) sobre APIs de programação paralela utilizando *Posix Threads*, OpenMP, *Message Passing Interface* (MPI). A relação com este trabalho é relevante pois em seu trabalho avalia o quanto de desempenho e consumo de energia é consumido em memória(RAM e cache) e processamento(instruções executadas) conforme o comportamento e uso de *threads*.

Existem diversos trabalhos direcionados no comparativo entre Intel Xeon Phi com outros aceleradores gráficos. Igualmente, existem muitos estudos acadêmicos sobre NVIDIA GPU e principalmente avaliando seu desempenho e modelos de programação baseados em CUDA. Entre os trabalhos comparativos mais recentes podemos citar (HENAO et al., 2016), nele descreve-se uma proposta de um esquema chamado Efficiently Energetic Acceleration (EEA) para aplicações científicas de grande escala executadas em arquiteturas heterogêneas. Através de dois monitores denominados *enerGyPU* (para GPU) e

enerGyPhi (para Xeon Phi) que realizam a captura e controle dos dados em tempo real enquanto executam as aplicações. Posteriormente também possibilita consultar através de gráficos os resultados de consumo de energia. A métrica PPW utilizada no Green 500 na qual é medido o consumo de energia sobre um determinado instante de tempo conforme apresentado pela equação de 2.3 de energia, considerando que neste trabalho houve mais de um node a equação proposta para atender a cada componente (CPU, GPU, RAM) é $Power_{Node}(i) = \sum_{j=1}^{nc} P_{CPU}^{j}(i) + \sum_{j=1}^{ng} P_{GPU}^{j}(i) + \sum_{j=1}^{nm} P_{RAM}^{j}(i)$. Os benchmarks utilizados foram *HPL 2.0* otimizado para GPU Tesla 20-*series* e o outro o *HPL 2.1* otimizado para Intel Xeon Phi. Os 12 experimentos aplicados no ambiente *Phi-Server* considerando quatro matrizes de tamanhos {20480, 30720, 40960, 56320} com quatro blocos de tamanhos {512, 768, 1024} demonstraram que a compactação por *thread affinities* proporcionou melhor desempenho e consumo eficiente de energia dos recursos computacionais como no caso da resolução da matriz tamanho {20480, 30720, 40960} na divisão por blocos de tamanho {1024}.

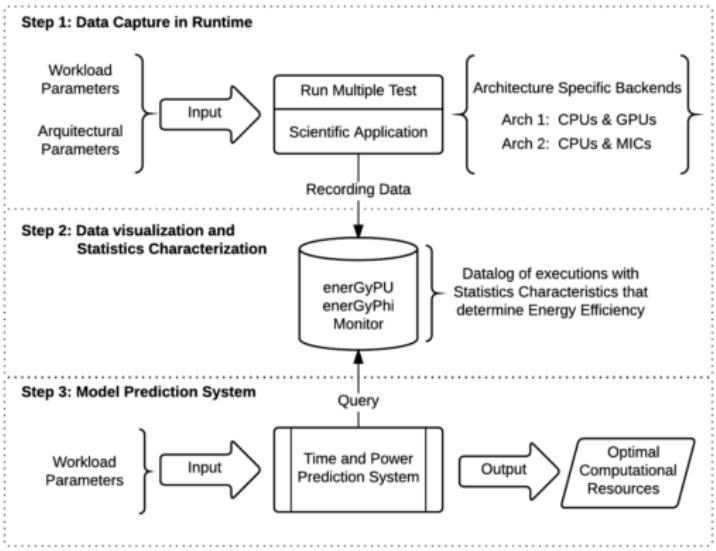


Figura 16 – Esquema de Energia Eficiente proposto por (HENAO et al., 2016).

Fonte: (HENAO et al., 2016).

Também existem outras propostas de ferramentas ou métodos para monitoramento e registro de energia durante o processamento. Entre elas pode-se citar (SMEJKAL et al., 2017). Sua proposta é um software chamado E-Team que é um mecanismo baseado em RAPL para monitoramento e registro baseado em RAPL para contabilizar o consumo de energia de aplicações e ambientes multi-core, podendo ser utilizado como um serviço com a possibilidade de iniciar e parar quando necessário.

O monitoramento externo é muito utilizado para monitoramento de consumo de energia elétrica em ambientes de *data centers* ou ambientes de alto desempenho. Existem vários trabalhos em comparações de medidores de potência externos também chamados

de *power meters*. Entre eles, podemos citar (ROSTIROLLA et al., 2015), no qual propõe uma estrutura externa de componentes para uso em ambientes HPC. Giefers em seu trabalho (GIEFERS et al., 2016), fez um estudo que envolve os coprocessadores NVIDIA GPU, Intel Xeon Phi e FPGA para abraçar a grande maioria dos operadores de hardware aplicados em sistemas HPC modernos. O trabalho de (IGUAL et al., 2015) aborda o consumo de energia em aceleradores gráficos baseados em PCIe realizando um comparativo entre Xeon Phi, NVIDIA GPU utilizando como *powermeter* o DC Power Analyzer modelo Agilent N67058 e o framework de software *pmlib*. Este trabalho é importante pois valida que a ferramenta *micsmc* da Intel registra resultados semelhantes ao obtido pelo *powermeter*, garantindo que mesmo via software os valores de consumo de energia e temperatura coletados são válidos.

Authors	Context
(SHAO; BROOKS, 2013)	Characterization and Energy Model by Instruction (EPI) Benchmark: *Customized Microbenchmarks*
(LAKOMSKI; ZONG; JIN, 2015) (SI et al., 2014)	Load Balancing between *CPU* e *GPUs* Hybrid Programming Model with OpenMP Benchmark: *NAS(MG)/Graph500/One-Sided Microbenchmark*
(LAWSON; SOSONKINA; SHEN, 2014)	Energy Evaluation with *Thread Affinities* Benchmark: *NAS(EP,FT,IS,MG)*
(BALLADINI et al., 2011)	Impact of DVFS and Parallel Programming on HPC systems Benchmark: *NAS(IS,EP,MG)*
(HENAO et al., 2016)	Energy monitor on *Xeon Phi* e *GPU* Benchmark: *Linpack* e *HPL 2.1*
(LORENZO et al., 2015)	Monitoring different thread numbers Benchmark: *PARSEC* e *SPLASH-2X*
(LORENZON; CERA; BECK, 2015)	Different Multi-Threading Interfaces Benchmark: *Memory: GL/GS/LU/MM* e *CPU: PI/MS/DJ/HS*
(GIEFERS et al., 2016)	Energy-efficiency comparative of GPU,Xeon Phi and FPGA Benchmark: *Sparse Matrix Multiplication*
(LAWSON et al., 2016)	Power Limiting of Parallel Applications on Intel Xeon Phi Benchmark: *GAMESS, CoMD and NAS*

Tabela 3 – Resumo dos trabalhos relacionados.

Realizando um comparativo entre este trabalho e os trabalhos já realizados no meio acadêmico, abaixo estão alguns diferenciais:

a) ***Avaliação dos Modelos de Programação da Intel*** - Os trabalhos analisados não abrangem avaliação de desempenho e consumo de energia entre os modelos de programação da Intel.

b) ***Caracterização dos fatores influenciadores*** - Os trabalhos abordam escopo e cenários limitados (geralmente 1 coprocessador Xeon Phi) em casos mais específicos enquanto que neste trabalho o ambiente é composto de multi coprocessadores garantindo amplo número de cenários *offload* e *nativo*, proporcionando resultados mais completos. Também avalia através de grande volume de *ranges* a influência dos fatores como *size*, número de *nodes*, número de *cores*, *threads*, processos.

c) ***Memória compartilhada e Distribuída*** - A maioria dos trabalhos abordam uso de memória distribuída. Neste trabalho os cenários abordam resultados do benchmark *Linpack* para memória compartilhada e *HPL 2.1* e *HPCG* para memória distribuída. Todos estes benchmarks tradicionalmente são referência para classificação dos resultados no *ranking* internacional como Top 500.

3.2 Considerações Finais

Neste capítulo foram apresentados os trabalhos realizados no meio acadêmico mais relevantes relacionados diretamente com o contexto que está sendo abordado neste documento. A tabela 3 apresenta um resumo dos trabalhos descritos anteriormente e também destaca a as características e contribuições que foram considerados neste trabalho.

4 Metodologia

Este capítulo apresenta a metodologia de avaliação utilizada neste trabalho. Inicialmente foram analisados vários trabalhos acadêmicos já realizados relacionados com o tema com o objetivo de identificar quais abordagens e técnicas foram utilizadas e seus resultados sobre desempenho e consumo eficiente de energia. Logo após, foram avaliadas várias ferramentas e técnicas para registro, monitoramento e coleta dos dados de consumo de energia e resultados das execuções dos benchmarks. Dando seguimento, foram implementados um conjunto de *scripts* e aplicações para automatizar todos processos necessários para geração de dados de consumo de energia e desempenho a serem analisados posteriormente. Para a apresentação em gráficos e análise dos resultados de forma confiável e objetiva foi desenvolvida uma aplicação web. também apresenta em detalhes os critérios e planejamento dos cenários de testes considerados neste trabalho para avaliação de desempenho e consumo de energia de arquiteturas Intel Xeon Phi.

4.1 Benchmarks Utilizados

Durante o levantamento bibliográfico foram identificados características e critérios relevantes dos *benchmarks* utilizados em outros trabalhos acadêmicos. Baseado neste levantamento e também nos fatores de compatibilidade com o ambiente de simulações e arquiteturas Intel juntamente com os requisitos listados abaixo, foram escolhidos os benchmarks *Linpack*, *HPL 2.1* e o *HPCG*, os quais atendem aos seguintes requisitos:

a) Grau de relevância perante Ranking Top 500 e comunidade acadêmica;

b) Suporte à memória compartilhada em ambientes heterogêneos;

c) Suporte à memória distribuída utilizando MPI e OpenMP;

d) Suporte à arquitetura Intel Xeon e Xeon Phi utilizando os seguintes modelos:

- Processamento somente no *host*,
- Processamento *offload* (host + coprocessadores)
- Processamento *nativo* (somente nos coprocessadores Xeon Phi)
- Processamento *simétrico* (com suporte Xeon Phi)

4.1.1 *Linpack*

O benckmark *Linpack* surgiu através do projeto do software *Linpack* originalmente concebido para dar aos usuários a capacidade de verificar quanto tempo era consumido

durante a resolução de certos problemas com matrizes. As referências foram crescendo e em 1979 foi publicado o guia de usuário do *Linpack*. É utilizado como métrica de desempenho computacional considerando sistemas de ponto flutuante para medir o quão rápido o computador pode resolver sistemas de equações lineares $Ax=b$ como uma tarefa normal de trabalho de processamento (DONGARRA, 2015).

O *Linpack* utiliza basicamente o pacote Basic Linear Algebra Subprograms (BLAS) que disponibiliza um conjunto de rotinas para trabalhar com operações de álgebra linear como adição de vetores, multiplicação escalar, produto cartesiano, combinações lineares e matriz de multiplicação. Está consolidada pela maioria da comunidade acadêmica e indústria computacional como padrão uma API para implementações de instruções em ponto flutuante para hardware como vetor de registradores ou SIMD (Netlib BLAS, 2017) e (Wiki BLAS, 2017).

O arquivo de entrada (INPUT *file*) do *Linpack* necessita dos parâmetros citados na tabela 4, além dos outros customizáveis para otimização.

Variável de Ambiente	Linha de comando	Descrição
N	-n	Tamanho do problema *n by n*
ALIGN	–	4 (*single* 32 bits) 8 (*double* 64bits)

Tabela 4 – Parâmetros de execução de aplicação ou pelo arquivo INPUT *file* do *Linpack*.

Neste trabalho foi utilizado o *Linpack* para simulações sobre os modelos de programação *host*, *offload* e *nativo* sobre uso de memória compartilhada.

4.1.2 *HPL 2.1*

O benckmark High Performance Linpack (HPL), também chamado de *MP Linpack* na biblioteca Math Kernel Library (MKL) da Intel, é uma versão para cluster do *Linpack*. Também é um pacote de software baseado em *BLAS* nível 3 para resolução de sistemas lineares densos com precisão dupla (*double*/64bits) (Netlib HPL, 2017).

O arquivo de entrada (HPL.dat) do *HPL 2.1 Linpack* possibilita a configuração de alguns parâmetros padrão para execução, mas também é possível configurar outros parâmetros que afetam diretamente o ganho ou perda de desempenho conforme citados na tabela abaixo.

As tabelas 5 e 6 apresentam alguns parâmetros de execução e otimização respectivamente do *HPL 2.1 Linpack*.

A tabela 7 apresenta alguns parâmetros da arquitetura Intel MIC importantes para execução do HPL Linpack.

Variável de Ambiente	Linha de comando	Descrição
N	-n	Tamanho do problema
NB	-b	Tamanho de bloco(block size)
P	-p	dimensão linha(row)
Q	-q	dimensão coluna(column)
ALIGN		4(single) 8 (double)

Tabela 5 – Parâmetros de execução de aplicação ou pelo arquivo INPUT *file* do *HPL 2.1 Linpack*.

Fonte (DONGARRA, 2015).

Variável de Ambiente	Linha de comando	Descrição
MPI_PROC_NUM	-np	Processos MPI
MPI_PER_NODE	-perhost ou -n	Processos por socket
KMP_AFFINITY	-genv KMP_AFFINITY *<valor>*	Tipo de afinidade
OMP_NUM_THREADS	-genv OMP_NUM_THREADS *<valor>*	Número de threads

Tabela 6 – Parâmetros de execução de aplicação ou pelo arquivo HPL.dat do HPL Linpack.

Fonte (Netlib HPL, 2017).

Variável de Ambiente	Linha de comando	Descrição
NUMMIC	-genv NUMMIC *<valor>*	Qtd mics
HPL_PNUMMICS	-genv HPL_PNUMMICS *<valor>*	Qtd mics
HPL_MIC_DEVICE	-genv HPL_MIC_DEVICE *<valor>*	Quais mics
HPL_MIC_CORE	-genv HPL_MIC_CORE *<valor>*	CPUs core
HPL_MIC_NODE	-genv HPL_MIC_NODE *<valor>*	NUMA node
MIC_OMP_NUM_THREADS	-genv MIC_OMP_NUM_THREADS *<valor>*	Threads
HPL_HOST_CORE	-genv HPL_HOST_CORE *<valor>*	Cores utilizar
HPL_MIC_NUMCORES	-genv HPL_MIC_NUMCORES *<valor>*	Qtd cores

Tabela 7 – Parâmetros MIC de execução de aplicação HPL Linpack.

Fonte (Intel MKL Variables, 2017).

Para obter maior otimização do HPL Linpack também deve-se considerar as proposições abaixo:

a) Para melhor desempenho, habilitar Nom-uniform Access Memory (NUMA) no sistema e configurar execução para um processo MPI para cada socket NUMA.

b) Considerar MPI_PROC_NUM igual ao valor de P x Q. No qual P e Q são coluna e linha do número de processos no grid.

c) Utilizar os valores para NB (tamanho de bloco): 1 coprocessador=960, 2 coprocessadores=1024 e 3 coprocessadores=1200

d) A estimativa e escolha do tamanho do problema (N) devem ser considerados sobre

o seguinte cálculo:

$$N = \frac{\sqrt{(F * P * Q * M)}}{8} \tag{4.1}$$

No qual:

- F é o fator de utilização de memória
- M é o tamanho de memória em bytes

O fator de utilização de memória estima-se 0,7 para ambientes homogêneos e 2,5 para heterogêneos. Quando maior o fator, maior a distribuição de memória entre os *nodes* (Intel MKL Heterogeneous, 2017).

4.1.3 *HPCG*

O benchmark de Gradientes de Conjugado de Alto Desempenho (HPCG) é um esforço para criar uma nova métrica para a classificação de sistemas HPC. *HPCG* destina-se como um complemento ao HPL, atualmente utilizado para classificar os sistemas de computação TOP 500. Os padrões computacionais e de acesso a dados de HPL ainda são representativos de algumas aplicações escaláveis importantes, mas não de todas. *HPCG* é projetado para exercer padrões computacionais de acesso a dados que mais se aproximam de um conjunto diferente e amplo de aplicações importantes e para incentivar os designers de sistemas de computador a investir em capacidades que terão impacto sobre o desempenho coletivo dessas aplicações (DONGARRA; LUSZCZEK; HEROUX, 2017).

4.2 Ferramentas para Monitoramento e Coleta dos Dados

Os procedimentos de monitoramento e coleta de dados durante a execução das simulações deste trabalho utilizaram as ferramentas MPSS, Intel Open Source Running Average Power Limit (RAPL) e *Power Cap Framework*.

4.2.1 Intel MPSS

A Intel MPSS é uma plataforma de componentes de software para disponibilizar ferramentas de software para desenvolvimento, ferramentas e utilitários, bibliotecas *middleware*, módulos e *daemons* necessários para uso da arquitetura Intel MIC e dos coprocessadores Intel Xeon Phi. Subjacente a todos estes recursos computacionais, está o Intel Xeon Phi Linux Kernel, uma versão reduzida e específica para arquitetura MIC (MPSS, 2014a).

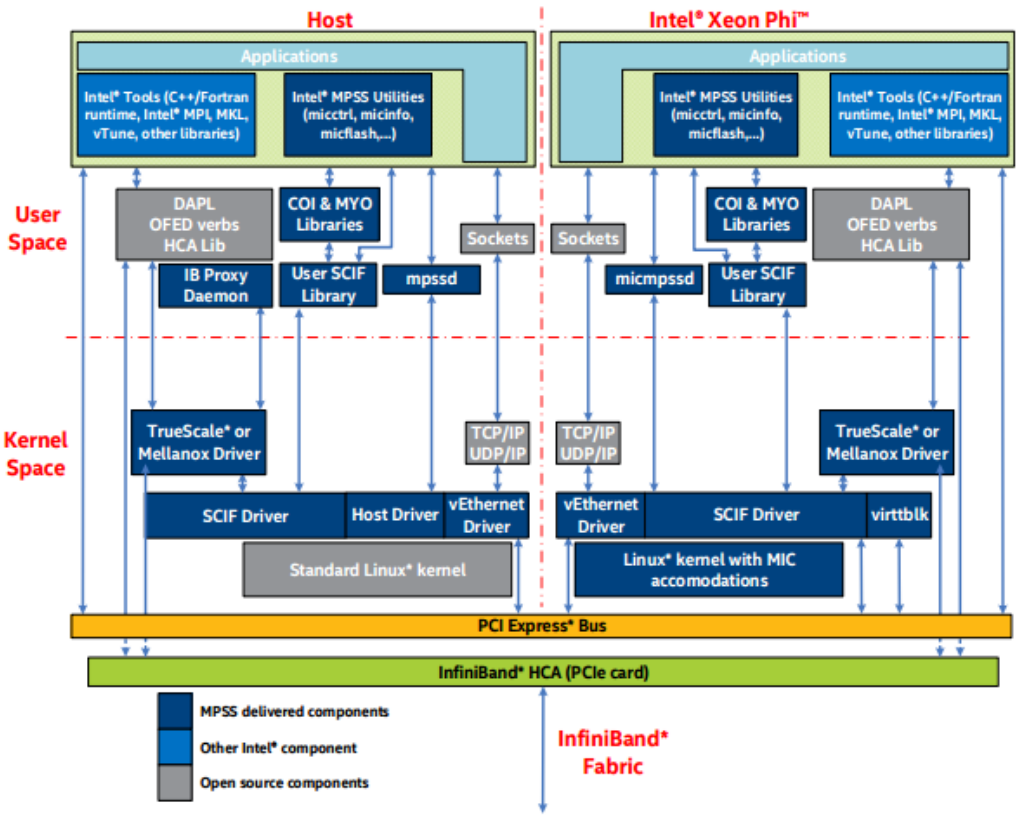


Figura 17 – Arquitetura Intel *MPSS*.

Fonte: (MPSS, 2014a).

Os utilitários para gerenciamento dos coprocessadores Xeon Phi são:

a) ***micctrl*** - Pode ser utilizado para controlar o sistema operacional como por exemplo boot, shutdown e reset. além de oferecer inúmeras opções para simplificar o processo de configuração de cada processador.

b) ***micinfo*** - Apresenta informações detalhadas sobre os coprocessadores instalados.

c) ***micsmc*** - Utilitário para monitorar parâmetros físicos dos coprocessadores tais como: memória, temperatura, frequência dos cores, consumo de memória etc.

d) ***miccheck*** - Utilitário para checklist e testes de diagnósticos para verificar a configuração e status corrente dos componentes da MPSS sobre o coprocessador.

e) ***micflash*** - Utilizado para atualizar imagens flash e apresentar informações sobre versão em uso.

Neste trabalho, todo monitoramento de desempenho e consumo de energia dos coprocessadores foi realizado pelo utilitário *micsmc*.

Os parâmetros são atualizados a cada milésimo de segundos pela MPSS e consultados durante a execução dos processos. As figuras 18, 19, 20 apresentam os parâmetros de temperatura, frequência e memória respectivamente do coprocessador Xeon Phi.

```
mic0 (temp):
Cpu Temp: ..... 48.00 C
Memory Temp: ..... 38.00 C
Fan-In Temp: ..... 33.00 C
Fan-Out Temp: ..... 38.00 C
Core Rail Temp: ..... 37.00 C
Uncore Rail Temp: ..... 39.00 C
Memory Rail Temp: ..... 39.00 C
```

Figura 18 – Informações de temperatura do *mic0* no log gerado pelo *micsmc*.

Fonte: (MPSS, 2014a).

```
mic0 (freq):
Core Frequency: ..... 1.24 GHz
Total Power: ..... 101.00 Watts
Low Power Limit: ..... 315.00 Watts
High Power Limit: ..... 375.00 Watts
Physical Power Limit: .... 395.00 Watts
```

Figura 19 – Informações de frequência do *mic0* no log gerado pelo *micsmc*.

Fonte: (MPSS, 2014a).

```
mic0 (mem):
Free Memory: ..... 14976.32 MB
Total Memory: ..... 15513.17 MB
Memory Usage: ..... 536.84 MB
```

Figura 20 – Informações de memória do *mic0* no log gerado pelo *micsmc*.

Fonte: (MPSS, 2014a).

4.2.2 Intel *RAPL Power Meter* e *Power Cap Framework*

O RAPL é uma ferramenta para Linux que implementa um software para medição de energia usando *driver power capping sysfs* disponível a partir do Linux kernel 3.13 release. (*Intel Open Source - RAPL Power Meter site*).

O ***Power Cap Framework*** disponibiliza uma interface entre kernel e ambiente de usuário permitindo acesso aos contadores de energia (*power capping drivers*) para manipulação de consumo de energia.

Existem três formas de acessar registros da RAPL usando o *kernel* linux. A primeira é lendo arquivos localizados em /sys/class/powercap/intel-rapl/intel-rapl:0 usando

powercap interface. Este modo não requer acesso restrito e foi utilizado neste trabalho. A segunda, pode ser usando o *perf*, interface de eventos disponível a partir da versão kernel 3.14. Este modo requer acesso *root* ou *sudo perf stat -a -e ̈power/energy-cores//̈bin/ls*. Eventos do perf em */sys/bus/event_source/devices/power/events/*. A terceira opção é usando *raw-access* sobre contadores *MSRs* em */dev/msr* (VWEAVER, 2017).

A figura 21 apresenta-se a árvore de contadores de kernel disponibilizado pela *RAPL* e *Power cap*.

```
/sys/devices/virtual/powercap
??? intel-rapl
??? intel-rapl:0
?   ??? energy_uj
?   ??? intel-rapl:0:0
?   ?   ??? energy_uj
?   ?   ??? max_energy_range_uj
?   ?   ??? power
?   ??? intel-rapl:0:1
?   ?   ??? energy_uj
?   ?   ??? max_energy_range_uj
?   ?   ??? power
?   ??? max_energy_range_uj
?   ??? max_power_range_uw
?   ??? power
??? intel-rapl:1
?   ??? energy_uj
?   ??? intel-rapl:1:0
?   ?   ??? energy_uj
?   ?   ??? max_energy_range_uj
?   ??? intel-rapl:1:1
?   ?   ??? energy_uj
?   ?   ??? max_energy_range_uj
?   ??? max_energy_range_uj
?   ??? max_power_range_uw
?   ??? power
```

Figura 21 – Estrutura de registros *sysfs* do *power cap*.

Fonte: (VWEAVER, 2017).

Os atributos utilizados foram:

a) *energy_uj* - Registro corrente de energia em micro joules.

b) *fax_energy_range_uj* - Limite do contador de energia citado acima, também em micro-joules.

c) *power_uw* - Registro corrente de energia em micro-watts.

d) *max_power_range_uw* - Limite do contador de energia citado acima, também em micro-watts.

É possível em alguns ambientes ter ambos *power* e *range* contadores, porém somente um é obrigatório.

A figura 22 apresenta um exemplo de consulta ao contador da "*cpu0*".

```
$ cd /sys/class/powercap/
$ cat intel-rapl/intel-rapl\:0/energy_uj
124973008911
$ cat intel-rapl/intel-rapl\:1/energy_uj
124973008911
$ cat intel-rapl/subsystem/intel-rapl\:0\:0/energy_uj
62564240661
```

Figura 22 – Exemplo de consulta aos contadores do *Power Cap*.

Fonte: (VWEAVER, 2017).

4.3 Ambiente de Simulação

Todos testes e avaliações foram realizados no ambiente *GridUNESP*[1] de alto desempenho do Núcleo de Computação Científica da UNESP (NCC). Além dos nós usuais de clusters também existem três nós específicos com recursos Intel MIC (*phi01*, *phi02* e *phi03*). As características do ambiente *phi03.ncc.unesp.br* estão apresentadas na tabela 8:

CPU	Modelo da CPU	2x E5-2699v3
	Nro de Cores por CPU	18
	Frequência	2.3GHz
	Memória RAM	128GB
Coprocessadores	Modelo do Coprocessador	Intel Xeon Phi 7120P
	Nro de Coprocessadores	4
	Nro de Cores por Coprocessador	61
	Frequência	1.2GHz
	Memória Coprocessador	16GB

Tabela 8 – Configurações do ambiente de testes e simulações do NCC da UNESP.

Fonte (NCC da UNESP, 2017).

Um ponto muito importante é que somente foi possível avaliarmos neste trabalho o desempenho e consumo de energia entre os modelos de programação (*host,offload,nativo*)

[1] Agradecimento especial a equipe do NCC da UNESP pelo fornecimento e suporte técnico

devido ao node *phi03* estar utilizando o mesmo modelo *7120P* nos quatro coprocessadores instalados. Neste caso seria inviável e incompatível avaliarmos, por exemplo, consumo de energia entre um coprocessador desde modelo e outro coprocessador de modelo diferente.

4.4 Processo para Medição e Análise de Consumo de Energia

O processo de medição e análise de desempenho e consumo de energia tem como objetivo utilizar métodos através de um conjunto de aplicações desenvolvidas para automatizar todos testes e simulações realizados neste trabalho, monitorando, registrando e armazenando dados confiáveis para apresentação e análise dos resultados. Basicamente este processo é realizado em três etapas. A figura 23 apresenta o fluxo macro deste processo.

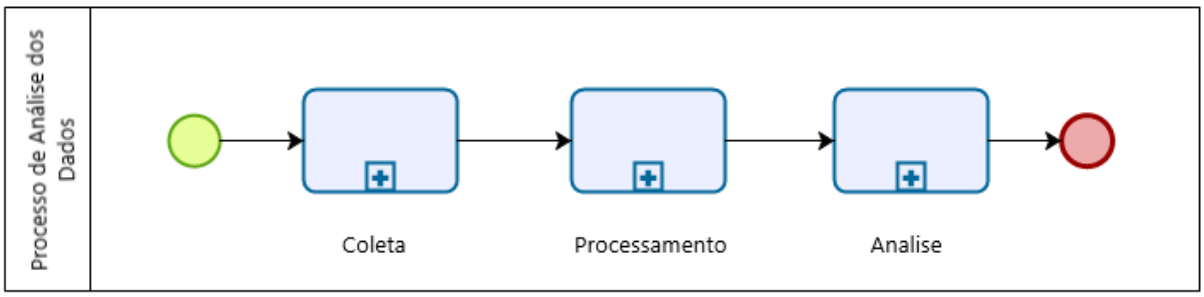


Figura 23 – Processo de Análise de Consumo de Energia.

Procedimentos realizados no processo:

a) *Passo 1* - A primeira etapa é responsável pela execução dos cenários juntamente com o monitoramento dos registros de energia da CPU, DRAM e coprocessadores Intel Xeon Phi. Todos procedimentos nesta etapa são realizados por *scripts shell* e os arquivos de cenários são gerados através de planilha de dados *LibreOffice*. Ao final do processo estão disponíveis os arquivos de resultados da execução do *Benchmark*, *Logs* de variáveis de ambiente utilizadas na execução, *Logs* de monitoramento de consumo de energia em *joule* das CPUs e RAM e para finalizar a temperatura em *celsius* dos coprocessadores.

b) *Passo 2* - Na segunda etapa, após a geração e coleta dos dados de logs dos benchmarks, todos arquivos e dados são validados através de uma aplicação Java. Esta aplicação consulta cada arquivo gerado, valida os atributos e dados e caso estes estejam consistentes então exporta para uma base de dados *mysql*.

c) *Passo 3* - Na terceira etapa através de uma aplicação *php web* é um conjunto de relatórios para consulta de todos os dados de forma gráfica. Todos relatórios e grá-

ficos contemplam e aplicam procedimentos estatísticos como desvio padrão, média entre outros filtros para garantir confiabilidade dos resultados apresentados.

4.4.1 Coleta dos dados

O processo de coleta de dados tem como objetivo varrer todos cenários e para criar um processo no sistema operacional para a execução do benchmark e outro processo para monitorar os registros de consumo de energia de todo ambiente. A figura 24 apresenta o fluxo básico deste processo.

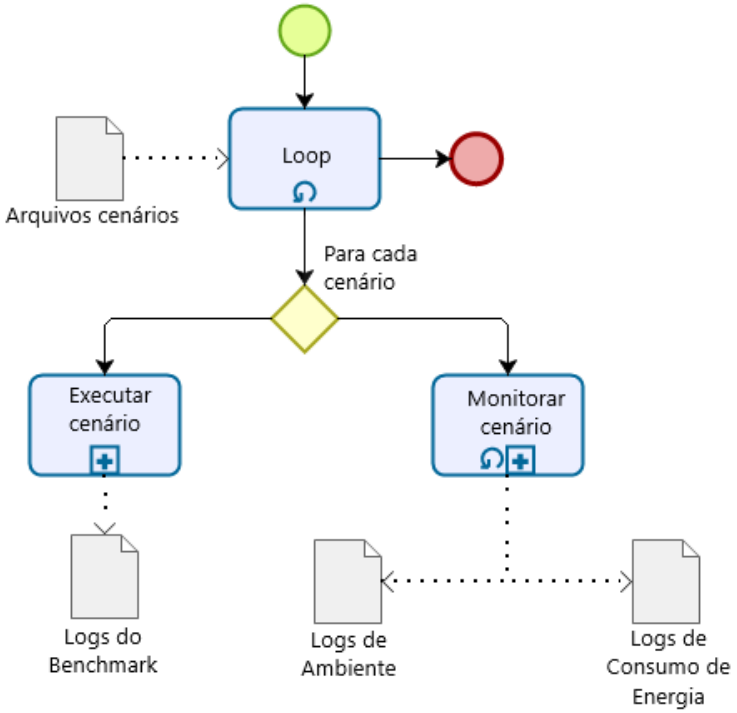


Figura 24 – Coleta dos Dados.

Procedimentos realizados no processo de coleta dos dados:

a) *Passo 1* - A primeira ação é varrer (loop) no arquivo de cenários todos que devem ser executados, atribuindo cada parâmetro informado para a respectiva variável ou parâmetro de processamento. O processo é sequencial e somente inicia o processamento de um novo cenário caso o cenário anterior já tenha finalizado. Para cada cenário são criados dois processos no sistema operacional. O primeiro para execução do benchmark e o segundo processo para monitoramento dos processos criados pelo primeiro.

b) *Passo 2* - O primeiro processo é responsável por preparar variáveis de ambiente baseado nos parâmetros passados ao cenário e disparar a execução do benchmark definido no cenário. Os resultados, juntamente com os logs do benchmark são armazenados em diretório para posterior coleta.

c) *Passo 3* - No segundo processo, o processo de monitoramento fica executando em intervalos definidos nos parâmetros de cenário, até que a execução do benchmark tenha finalizado. Os logs de consumo de energia e variáveis de ambiente são armazenados em diretório para posterior coleta.

4.4.2 Processamento dos dados

O processamento dos dados conforme apresentado na figura 25 é uma etapa importante pois realiza a importação de todos arquivos gerados durante a coleta e após uma validação dos resultados dos benchmarks e também dos outros arquivos de logs de energia realiza a inclusão em banco de dados para que os dados possam ser melhor manipulados para apresentação. É importante salientar que em momento algum os dados são manipulados de forma que fiquem diferentes dos arquivos originais. Este processo utiliza uma aplicação elaborada para ler todos arquivos e processar a inclusão no banco de dados.

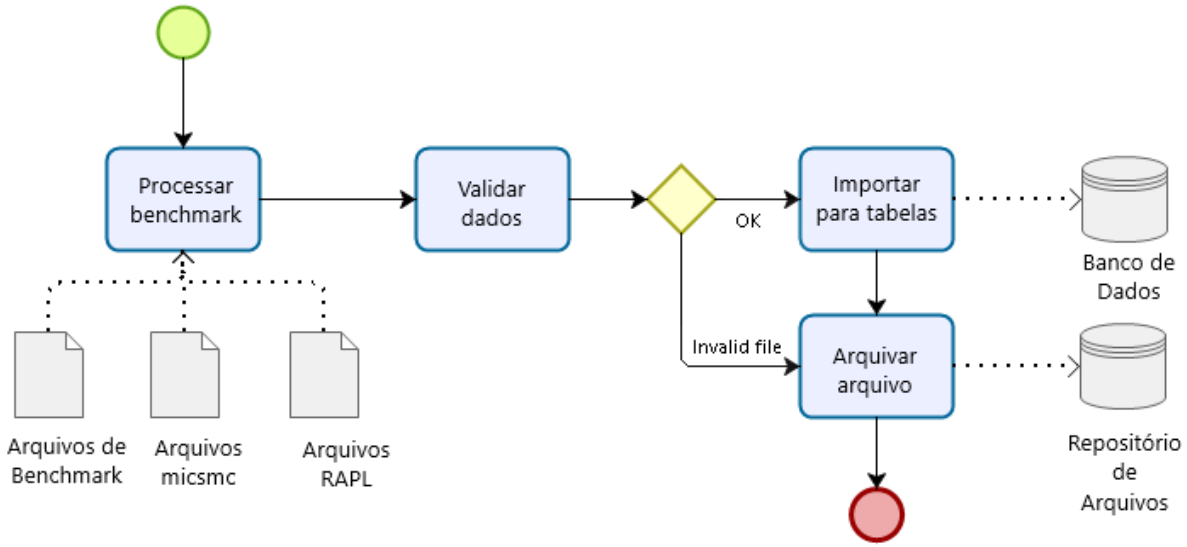


Figura 25 – Processamento dos Dados.

Procedimentos realizados no processamento dos dados:

a) *Passo 1* - A aplicação java faz a varredura de todos arquivos de resultados de benchmarks juntamente com os arquivos de consumo de energia gerados tanto para coprocessadores quanto para as CPUs e memória DRAM. Para cada arquivo realiza os seguintes passos.

b) *Passo 2* - O passo seguinte ao de leitura dos arquivos é a validação dos dados de forma a garantir que somente sejam importados os resultados com status *passed* dos benchmarks e os respectivos arquivos de logs de energia registrados na execução deste benchmark.

c) *Passo 3* - Os arquivos corretos, são importados em respectivas tabelas em banco de dados mysql. Posteriormente o arquivo é movido para um diretório para posterior consulta se necessário.

4.4.3 Análise dos dados

Os resultados são consultados dinamicamente através do ambiente web. O fluxo básico e processo de consulta estão apresentados na figura 26. Este ambiente foi extremamente necessário pois o volume de cenários e simulações realizados foi grande, necessitando que os dados estejam disponíveis com fácil acesso para consultas através de ferramentas como sql e componentes gráficos dinâmicos. Através do ambiente web é possível parametrizar a geração dos gráficos conforme atributos existentes na base de dados armazenada pelos processos anteriores (coleta e processamento).

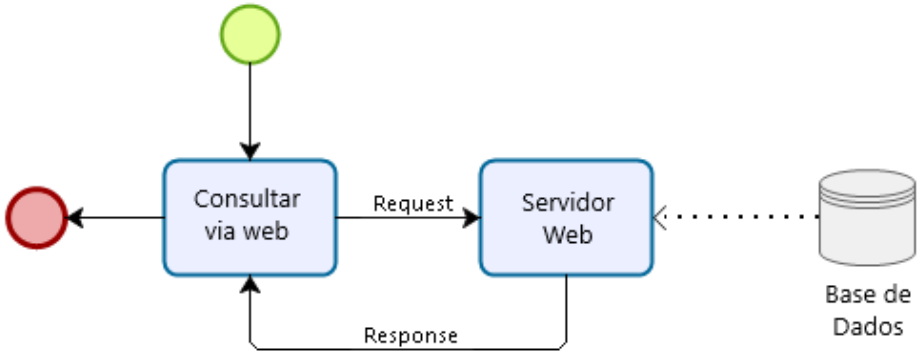


Figura 26 – Apresentação dos Dados.

Procedimentos realizados no processo de apresentação dos dados:

a) O usuário seleciona um dos gráficos disponíveis e submete ao servidor web a requisição.

b) O servidor web recebe a solicitação e processa conexões com banco de dados para consulta nos dados e retorna ao usuário o gráfico com os dados adequadamente filtrados e tratados.

4.5 Critérios que influenciam no Desempenho e Energia

Nos capítulos anteriores, foram apresentados estudos e levantamentos sobre trabalhos acadêmicos relacionados com o tema deste trabalho. Observou-se que diversos elementos e critérios podem influenciar direta ou indiretamente na relação de desempenho e consumo eficiente de energia. Porém os trabalhos estão direcionados a proposições ou alternativas sobre contextos específicos com cenários bem restritos e consequentemente

não abordando ou detalhando quais características devem ser consideradas para se obter o modelo de programação mais eficiente para ser utilizado por aplicações de uso geral.

Conforme relatado no capitulo de trabalhos relacionados, ainda existem dúvidas e questionamentos sobre quais são as principais características e critérios base para serem considerados na escolha de um modelo de programação que impactam diretamente na relação desempenho e energia. Este trabalho busca classificar e apresentar uma visão através da figura 27 de como é importante conhecermos e otimizarmos corretamente não só internamente em códigos e técnicas de programação aplicadas mas principalmente externamente ao sistema e considerando o ambiente de execução.

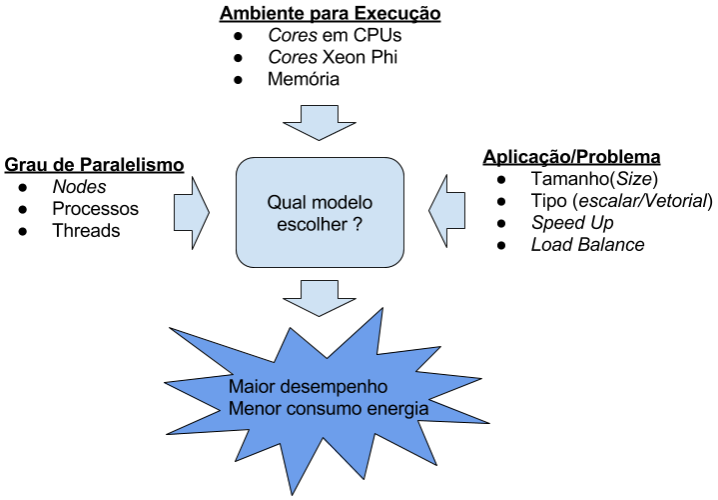


Figura 27 – Critérios que influenciam nos modelos de programação paralelos.

Um dos eixos básicos é o *Ambiente de Execução*, que disponibiliza recursos computacionais como *cores*, CPUs, memória e coprocessadores Xeon Phi. Outro eixo importante é o *Grau de Paralelismo*, responsável pela definição do número de processos e *threads* em uso e principalmente pela forma e distribuição das cargas entre os processos. Finalmente a *Aplicação*, que através das características como tamanho de uso de memória e comportamentos internos da aplicação também exigem que o ambiente esteja otimizado.

4.6 Parâmetros de Avaliação

Baseado nos critérios e características apresentados no tópico anterior, os parâmetros escolhidos neste trabalho para avaliação do desempenho e consumo de energia estão discriminados nos próximos tópicos.

4.6.1 Ambiente de Execução

Os parâmetros relacionados ao ambiente de execução são:

a) ***align*** - no qual determina o tamanho da instrução em arquiteturas. Neste trabalho foram avaliados valores 4 (*32 bits*) e 8 (*64 bits*). Esta variável é definida como parâmetro ALIGN durante a execução dos benchmarks e depende diretamente da capacidade da Instrução .

b) ***thread affinities*** - para determinar distribuição de carga de trabalho entre cores. Nas variáveis KMP_AFFINITY e MIC_KMP_AFFINITY foram considerados os valores *compact*, *balanced* e *scatter* (Robert Reed, 2017).

c) ***page size*** e ***large page*** - que gerencia o buffer e consequentemente otimização em problemas que exigem memória.

d) ***cores*** - que possibilita a definição do uso da capacidade de processamento disponível no ambiente, abrangendo tanto os *cores* da CPU quanto dos coprocessadores a utilizar. Neste trabalho foi utilizado a variável KMP_PLACE_THREADS e MIC_KMP_PLACE_THREADS, considerando valores {1, 2, 4, 8, 12, 16, 24, 32, 64, 72, 244} para *cores* (*exemplo KMP_PLACE_THREADS=72c*) dependendo dos cenários em uso (Intel Thread Affinity, 2017).

4.6.2 Grau de Paralelismo

Os parâmetros escolhidos relacionados ao *Grau de Paralelismo* foram:

a) ***threads***, é um dos parâmetros de desempenho mais utilizados nos trabalhos acadêmicos e indústria. Neste trabalho foram utilizadas as variáveis OMP_NUM_THREADS e MIC_OMP_NUM_THREADS, considerando valores {8, 12, 16, 24, 32, 64, 72, 244}. Importante lembrar que estes parâmetros possuem relação com OpenMP.

b) ***processos***, considerando cenários sobre memória distribuída e *benchmarks HPL 2.1* e *HPCG*, foram utilizadas as variáveis MPI: MPI_PER_NODE e MPI_PROC_NUM. Neste trabalho, devido ao ambiente possuir 2 CPUs com 18 cores físicos cada e mais 4 coprocessadores Intel Xeon Phi, os valores utilizados para MPI_PER_NODE foram {1, 2, 4}, e para MPI_PROC_NUM foram {2, 4, 8, 16}.

4.6.3 Contexto da Aplicação

Os parâmetros escolhidos relacionados ao *Contexto da Aplicação* foram:

a) ***dimensionamento do problema***, para validar o impacto dos outros critérios de ambiente e paralelismo é importante considerar como eles se comportam em todas as faixas de tamanho de problema. Com isso será possível garantir quais

critérios impactam diretamente em quais faixas de problema. Neste trabalho o dimensionamento do problema está vinculado diretamente às capacidades mínima e máxima de processamento e memória do ambiente de simulação.

b) ***tempo de execução***, o tempo gasto na execução é muito relevante pois apresenta na prática o quanto a aplicação está desperdiçando de energia e recursos computacionais para uma determinada carga de trabalho.

4.7 Métricas Utilizadas

4.7.1 Desempenho

O tópico Desempenho Máximo de Processamento da seção 2.2 Métricas de Desempenho e Energia apresenta os conceitos sobre *Flops*, $R_(max)$ e $R_(peak)$ utilizados pelo *Linpack*. Neste trabalho a métrica em uso para avaliar desempenho será a *GFlops/s* conforme padrão usado pelo Top 500.

4.7.2 Consumo de Energia

Considerando a equação 2.1 criada pelo Green 500 e atualmente em uso para avaliação do ranking dos ambientes HPC mais eficientes em consumo de energia, também foi definido que o consumo de energia deve ser baseado na média do total de energia consumido em um determinado período de tempo.

Detalhando a equação do Green 500 e considerando o ambiente avaliado neste trabalho, as partes diretamente a serem avaliadas no consumo de energia devem ser CPU, Memória e Coprocessadores Xeon Phi, nos quais o consumo total de energia é obtido através da soma parcial da energia consumida destes componentes. Neste contexto a equação 4.2 proposta por (HENAO et al., 2016) pode ser perfeitamente aplicada para avaliar os componentes Intel Xeon Phi.

$$Energia_{Node}(i) = \sum_{j=1}^{nc} P^{j}_{CPU} + \sum_{j=1}^{ng} P^{j}_{XeonPhi} + \sum_{j=1}^{nm} P^{j}_{RAM} * \ \Delta t(i) \tag{4.2}$$

O consumo de energia por node é a soma geral do consumo de cada componente representado na equação 4.2 por P^{j}_{CPU} para CPU, $P^{j}_{XeonPhi}$ para Xeon Phi e P^{j}_{RAM} para Memória.

4.8 Cenários Avaliados

O planejamento dos cenários de execução foi definido baseado no cruzamento e matriz dos valores definidos para cada critério. Por exemplo, considerando 10 cargas de

trabalho com tamanhos diferentes, 3 critérios de ambiente de execução sobre 4 modelos de programação gera-se um total de 120 cenários somente considerando ambiente de execução. Outras características também foram consideradas, como por exemplo memória compartilhada e distribuída, em cenários com memória distribuída o processamento é baseado em processos MPI e não utiliza a variável OMP_NUM_THREADS e o contrário, na compartilhada não utiliza MPI_PER_NODE e MPI_PROC_NUM.

O número de execuções também foi considerado, somente deve-se considerar cenários com um número de execuções maior que 30 vezes.

Para e elaboração dos cenários de teste foram considerados os seguintes elementos:

a) Ambiente de execução

Os parâmetros de variáveis de ambiente de execução estão listados na tabela 9.

Descrição	Parâmetro	Valores
align	ALIGN	{4,8}
cores	KMP_PLACE_THREADS	{1,2,4,8,12,16,24,32,64,72,244}c
thread affinity	KMP_AFFINITY	{compact,scatter,balanced}

Tabela 9 – Elementos de *Ambiente de Execução* dos cenários.

b) Grau de Paralelismo

Os parâmetros de variáveis de paralelismo de execução estão listados na tabela 10.

Descrição	Parâmetro	Valores
processos	MPI_PROC_NUM	{2,4,8,16}
threads	OMP_NUM_THREADS	{8,12,16,24,32,64,72}

Tabela 10 – Elementos de *Grau de Paralelismo* dos cenários.

c) Contexto da Aplicação

Os parâmetros de variáveis de aplicação estão listados na tabela 11

Descrição	Valores
size(N)	{1000,..,40000}
threads	{2,4,8,12,16,24,32,64}

Tabela 11 – Elementos do *Contexto da Aplicação* dos cenários.

A definição das possibilidades de cenários sobre os modelos de programação é apresentada na tabela 12. Esta definição é a base deste trabalho pois todos os cenários devem gerar resultados com o objetivo do comparativo entre estes possíveis modelos.

hosted	{*CPUs*}
offload com 1 xeon phi	{*CPUs + mic0*}
offload com 2 xeon phi	{*CPUs + mic0 + mic1* }
offload com 3 xeon phi	{*CPUs + mic0 + mic1 + mic2*}
offload com 4 xeon phi	{*CPUs + mic0 + mic1 + mic2 + mic3*}
nativo 1 xeon phi	{*mic0*}

Tabela 12 – Modelos suportados pelo ambiente de simulação.

4.8.1 Configurações para Linpack

O benchmark *Linpack*, utiliza um arquivo *INPUT* de configuração como entrada para parametrização das suas respectivas variáveis, conforme detalhes apresentados no tópico *Linpack* da seção 4.1.

A tabela 13 apresenta a parametrização das variáveis necessárias para que o benchmark *Linpack* seja executado contemplando os cenários (modelos) pré-definidos nos tópicos anteriores deste trabalho:

Variáveis	CPU	OF1D	OF2D	OF3D	OF4D	N1D
Variáveis Host						
OMP_NUM_THREADS	max 72	max 72	max 72	max 72	max 72	max 72
KMP_AFFINITY	compact	compact	compact	compact	compact	compact
OFFLOAD_DEVICES:	empty	0	1	2	3	(0,1,2,3)
NUMMIC:	empty	1	2	3	4	(1,2,3,4)
Variáveis MIC						
MIC_OMP_NUM_THREADS	empty	max 240	max 240	max 240	max 240	max 240
MIC_KMP_AFFINITY	empty					
HPL_PNUMMICS:	empty	1	2	3	4	(1,2,3,4)
HPL_MIC_DEVICE:	empty	0	1	2	3	(0,1,2,3)
HPL_LARGEPAGE	empty					
HPL_NUMWAPS	empty					

Tabela 13 – Variáveis utilizadas para o *Linpack*.

Os modelos de arquivo de cenários do Linpack utilizam a tabela 14 como layout, possibilitando informar os dados necessários para a execução do benchmark. A tabela 15 apresenta um exemplo de arquivo de configuração de cenário necessário para execução do Linpack.

#	**Valor**
1	Cenario
2	Time
3	N
4	Dimensão
5	ALIGN
6	OMP_NUM_THREADS
7	MIC_OMP_NUM_THREADS
8	KMP_AFFINITY

Tabela 14 – Parâmetros de configuração dos cenários do *Linpack*.

LP_CPU_ONLY:1:1024:1024:4:8::nowarnings,compact,1,0,granularity=fine:
LP_OFFLOAD_1_DEVICE:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
LP_OFFLOAD_2_DEVICES:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
LP_OFFLOAD_3_DEVICES:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
LP_OFFLOAD_4_DEVICES:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
LP_NATIVO_1_DEVICE:1:1024:1024:4::240:nowarnings,compact,1,0,granularity=fine:

Tabela 15 – Exemplo de arquivo de cenário do *Linpack*.

4.8.2 Configurações para HPL 2.1

O benchmark HPL, utiliza um arquivo *HPL.dat* de configuração como entrada para parametrização das suas respectivas variáveis, conforme detalhes apresentados no tópico *HPL 2.1* da seção 4.1.

A tabela 16 mostra a parametrização das variáveis necessárias para que o benchmark *HPL 2.1* seja executado contemplando os cenários(modelos) pré-definidos nos tópicos anteriores deste trabalho:

Variáveis	CPU_ONLY	OF1D	OF2D	OF3D	OF4D	N1D
Variáveis Host						
OMP_NUM_THREADS	max 72	max 72	max 72	max 72	max 72	max 72
KMP_AFFINITY	compact	compact	compact	compact	compact	compact
OFFLOAD_DEVICES:	empty	0	1	2	3	(0,1,2,3)
NUMMIC:	empty	1	2	3	4	(1,2,3,4)
Variáveis MIC						
MIC_OMP_NUM_THREADS	empty	max 240	max 240	max 240	max 240	max 240
MIC_KMP_AFFINITY	empty					
HPL_PNUMMICS:	empty	1	2	3	4	(1,2,3,4)
HPL_MIC_DEVICE:	empty	0	1	2	3	(0,1,2,3)
HPL_LARGEPAGE	empty					
HPL_NUMWAPS	empty					

Tabela 16 – Variáveis utilizadas para o *HPL 2.1*.

Os modelos de arquivo de cenários do HPL utilizam a tabela 17 como layout, possibilitando informar os dados necessários para a execução do benchmark. A tabela 18 apresenta um exemplo de arquivo de configuração de cenário necessário para execução do HPL.

4.9 Tratamento dos Resultados

Durante o processo de coleta, tratamento dos dados e armazenamento em banco de dados, todas informações estão armazenadas exatamente conforme capturado nos arquivos de logs. Isto possibilita a garantia que não houve manipulação e conversões dos dados durante o processo.

O processo de monitoramento e coleta de dados durante a execução está baseado num parâmetro de configuração de intervalo de tempo de coleta definido nos cenários. Isto é necessário pois os registros do kernel do sistema operacional são em milésimos de

#	**Valor**
1	Cenário
2	Time
3	N
4	NB
5	P
6	Q
7	ALIGN
8	MPI_PROC_NUM
9	OMP_NUM_THREADS
10	MIC_OMP_NUM_THREADS
11	MPI_PER_NODE
12	NUMMIC
13	HPL_PNUMMICS
14	HPL_MIC_DEVICE

Tabela 17 – Parâmetros de configuração dos cenários do *HPL 2.1*.

MP_CPU_ONLY:5:10000:256:1:2:4:2:2:240:1:0:0:
MP_OFFLOAD_1_DEVICE:5:1024:960:4:4:4:16:32:240:2:1:1:0
MP_OFFLOAD_2_DEVICES:5:1024:1024:4:4:4:16:32:240:4:2:2:0,1
MP_OFFLOAD_3_DEVICES:5:1024:1200:4:4:4:16:32:240:2:3:2:0,1,2
MP_OFFLOAD_4_DEVICES:5:1024:1200:4:4:4:16:32:240:4:3:2:0,1,2,3

Tabela 18 – Exemplo de arquivo de cenário do *HPL*.

segundos e se coletarmos numa fração de segundo o número de coletas fica muito grande. Para isso foi criado um parâmetro de *intervalo de tempo de coleta*, para definir no cenário o intervalo de coleta baseado no tamanho do problema. Exemplo, para um problema com tamanho 1000, o intervalo é 1 segundo, gerando média de 10 coletas. Mas para um tamanho 10000 de problema, 1 segundo deverá gerar muitos dados "repetidos"de coleta. Neste caso, o valor do intervalo está definido para 10 segundos, para gerar na média 10 a 20 coletas.

Conforme apresentado na seção Intel *RAPL Power Meter* e *Power Cap Framework* do capítulo 4, os dados de energia foram coletados através dos contadores *energy_uj* para CPU e DRAM, e para os coprocessadores Xeon Phi utilizado pela ferramenta *micsmc*.

Os valores armazenados para os coprocessadores Xeon Phi estão em *Graus Celsius*. A Intel não registra os valores de consumo de energia em *joules* ou *watts*. Por isso, conforme a definição de conversão pela *International System of Units*, foi feita de Graus Celsius para Joules por segundo conforme a equação:

$$0.000526565076466\ C^{\circ} = 1\ J \tag{4.3}$$

Logo, foi feito a multiplicação do valor armazenado em grau celsius por 0.000526565076466 totalizando o valor correspondente em *joules*.

4.9.1 Configurações para HPCG

Apesar do benchmark HPCG possuir a finalidade de avaliar a capacidade de uso de memória entre outros recursos em ambientes de memória distribuída, a definição de seus cenários é muito semelhante ao HPL pois também utiliza MPI.

Portanto, neste trabalho os cenários para HPCG seguem a mesma abordagem utilizada para HPL, mudando apenas o *range* de tamanho de problema.

4.10 Considerações Finais

Este capítulo apresentou informações sobre a metodologia aplicada para coleta, processamento e apresentação dos resultados das simulações sobre os cenários planejados. Também foram apresentadas informações sobre os benchmarks utilizados, tecnologias e frameworks utilizados para monitoramento e coleta dos registros de energia durante as execuções e processamento. Também apresentou informações sobre o método de avaliação utilizado neste trabalho, juntamente com as definições sobre os critérios de avaliação utilizados.

5 Resultados

Este tópico apresenta inicialmente detalhes sobre tratamento e manipulação dos resultados gerados durante o processamento e testes. E logo após apresenta os resultados obtidos através da execução dos cenários criados pelas combinações dos parâmetros e modelos de programação apresentados no capítulo anterior. Através desses resultados é possível avaliar e identificar quais modelos são mais eficientes na relação desempenho e consumo de energia.

5.1 Desempenho

O comparativo de desempenho entre os modelos de programação da *Intel* é muito relevante pois apresenta a capacidade de processamento realizado na prática pelos modelos de programação e o quanto dos limites especificados pelo fabricante estão sendo utilizados para cada modelo.

A grande diferença de desempenho entre o modelo *nativo* e outros ocorre devido a diferença na frequência dos processadores, onde no primeiro, somente no Xeon Phi, é 1.2GHz enquanto que no *host* que possui duas CPUs o clock do processador é 2.3GHz cada.

As seções a seguir apresentar em detalhes os resultados de desempenho obtidos durante a execução e processamento dos cenários.

5.1.1 Desempenho por Carga de Trabalho (*Linpack*)

O desempenho baseado em carga de trabalho (*workloads*) é muito utilizado na comunidade acadêmica, pois permite extrair o comportamento, considerando do mínimo ao máximo da capacidade computacional disponível no ambiente. Os resultados dos benchmarks Linpack apresentados a seguir foram gerados considerando o número de *threads* igual a 64, *thread affinity* com valor *compact* e *align* igual a 4.

Durante a execução da Linpack em memória compartilhada apresentado na figura 28, é possível notar que os modelos *host* e *offload* (OF1D, OF2D, OF3D e OF4D) apresentam um aumento linear de desempenho para problemas com tamanhos menores que 10.000. A partir deste valor, o desempenho tende a estabilizar. Não há ganhos de desempenho para diferentes configurações no modelo *offload*. No modelo *nativo* (N1D), o comportamento de desempenho segue o mesmo padrão, mas com um valor menor do que os outros modelos.

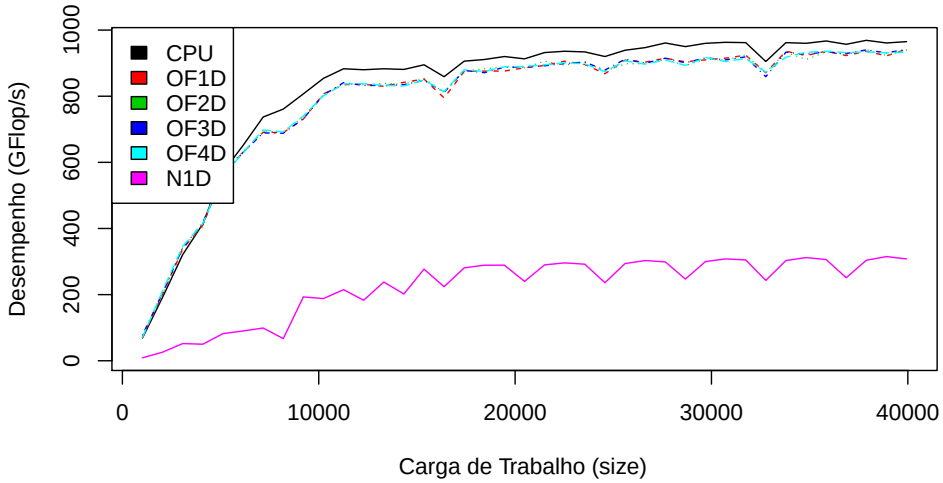


Figura 28 – Desempenho utilizando memória compartilhada (Linpack).

Os modelos *host* e *offload* demonstram grande semelhança nos resultados, isso ocorre porque a capacidade de memória do host é compartilhada com os coprocessadores em uso. Já no modelo *nativo*, como existe a limitação de apenas 16 GB em cada coprocessador usado no processamento, entende-se que o uso deste modo de processamento é adequado para execuções altamente paralisadas ou vetoriais, mas não requer grande uso de memória.

Os resultados apresentados figura 28 foram obtidos através da parametrização de variáveis conforma segue descrito na tabela abaixo:

	CPU	**OF1D**	**OF2D**	**OF3D**	**OF4D**	**N1D**
Parâmetros de Contexto da Aplicação						
Valores de N(Size)	{1000,...,30720}					
Dimensão	{1000,...,30720}					
Alinhamento(Align)	4					
Parâmetros de Paralelismo						
OMP_NUM_THREADS	{12,16,24,32,72}					
MIC_OMP_NUM_THREADS	-	{12,16,24,32,72}				
KMP_AFFINITY	compact					
MIC_KMP_AFFINITY	compact					
Parâmetros de Ambiente(Host)						
OFFLOAD_DEVICES:	-	0	0,1	0,1,2	0,1,2,3	0
NUMMIC:	-	1	2	3	4	1
Parâmetros de Ambiente(MIC)						
HPL_PNUMMICS:	-	1	2	3	4	1
HPL_MIC_DEVICE:	-	0	0,1	0,1,2	0,1,2,3	0
HPL_LARGEPAGE	-					
HPL_NUMWAPS	-					
Benchmark Executado	*Linpack*					

Tabela 19 – Parâmetros - Desempenho por carga de trabalho (*Linpack*)

5.1.2 Desempenho por Carga de Trabalho (*HPL 2.1*)

Outro importante cenário é a verificação e comparativo de desempenho entre os modelos mas considerando ambiente com memória distribuída. O processamento sobre o modelo *offload* utiliza MPI para comunicação e gerenciamento dos processos. Os gráficos no HPL benchmark também foram gerados usando o parâmetro P igual a 1, Q igual a 4 e tamanho de bloco da seguinte forma: (OF1D = 960, OF2D = 1024 e OF3D = 1024).

Através dos resultados obtidos para a execução do benchmark HPL e apresentados na figura 29, é possível mostrar que os modelos baseados *host* e *offload* foram executados de forma semelhante, com crescimento linear de acordo com o tamanho do problema. Para problema com tamanho 10000, a execução da Linpack obteve mais de 750 GFlops, enquanto que em HPL o resultado foi inferior a 150 GFlops. Semelhante ao trabalho de (HENAO et al., 2016), o melhor desempenho foi obtido usando afinidade de thread com valor *compact*.

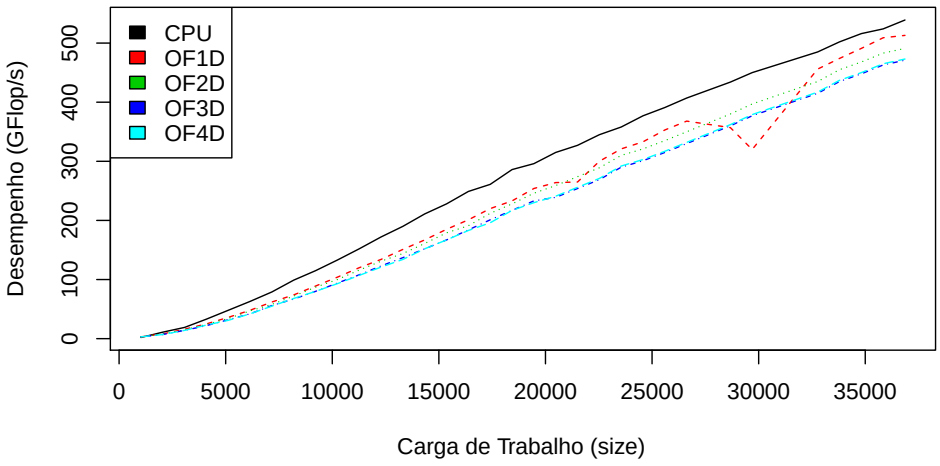


Figura 29 – Desempenho utilizando memória distribuída (HPL 2.1)

Os resultados apresentados na figura 29 foram obtidos através da parametrização de variáveis conforme segue descrito na tabela 20

5.2 Consumo de Energia

Além do desempenho, o consumo de energia gasto nos modelos de programação também é outro fator chave deste trabalho. Esta seção aborda o consumo de energia nos resultados obtidos durante a execução dos benchmarks Linpack e HPL.

	CPU	OF1D	OF2D	OF3D	OF4D	N1D
Parâmetros de Contexto da Aplicação						
Valores de P	1					
Valores de Q	4					
Valores de N(Size)	{1024,...,31744}					
Valores de NB(Block Size)	960	960	1024	1200	1200	960
Alinhamento(Align)	8					
Parâmetros de Paralelismo						
MPI_PER_NODE	4					
MPI_PROC_NUM	4					
OMP_NUM_THREADS	{72}					
MIC_OMP_NUM_THREADS	-	{72}				
KMP_AFFINITY	compact					
MIC_KMP_AFFINITY	compact					
Parâmetros de Ambiente(Host)						
OFFLOAD_DEVICES:	-	0	0,1	0,1,2	0,1,2,3	0
NUMMIC:	-	1	2	3	4	1
Parâmetros de Ambiente(MIC)						
HPL_PNUMMICS:	-	1	2	3	4	1
HPL_MIC_DEVICE:	-	0	1	2	3	0
HPL_LARGEPAGE	-					
HPL_NUMWAPS	-					
Benchmark Executado	*HPL 2.1*					

Tabela 20 – Parâmetros - Desempenho por carga de trabalho (*HPL 2.1*)

5.2.1 Consumo de Energia durante o Processamento (*Linpack*)

O tempo de execução de uma aplicação é muito importante e também é um bom indicador desta relação de desempenho e consumo de energia. Uma questão relevante é entendermos que, se aumentando o tamanho do problema ou o número de *threads* pode aumentar ou reduzir o tempo de execução. O consumo de energia gasto pela CPU e memória RAM é significativo comparado aos coprocessadores Xeon Phi.

Os resultados apresentados no trabalho de (LAKOMSKI; ZONG; JIN, 2015) também apresentam este comportamento, nele, utilizando os benchmarks Matrix de Multiplicação e Fractais, no modelo *offload* demonstra que aumentando a porcentagem de utilização de CPU através de um parâmetro de configuração o consumo de energia aumenta também. Ainda em seu trabalho, no entanto com o Linpack no Xeon Phi ocorre o oposto, quanto mais o tempo de execução o consumo médio diminui. Em comparação com este trabalho, analisando as figuras 30 e 34 é possível verificar que realmente aumentando a utilização da CPU aumenta o seu consumo de energia, mas o tempo de execução é maior e consequentemente consome mais energia no processamento total.

Os resultados do consumo de energia, considerando a memória compartilhada da Linpack, são apresentados na figura 30. O modelo de descarga com 4 Xeon Phis (OFD4) demonstra maior consumo de energia concentrado no *CPUs* e muito pouco nos coprocessadores. O consumo total de energia cresce quadraticamente com o tamanho do problema.

O modelo *offload* não é eficiente na relação desempenho e consumo de energia em ambientes de memória compartilhada pois requer uma forte dependência da CPU no host para gerenciar o processamento principal. No modelo *nativo*, como mostrado na figura 31, o consumo de energia é significativamente menor em comparação com os outros

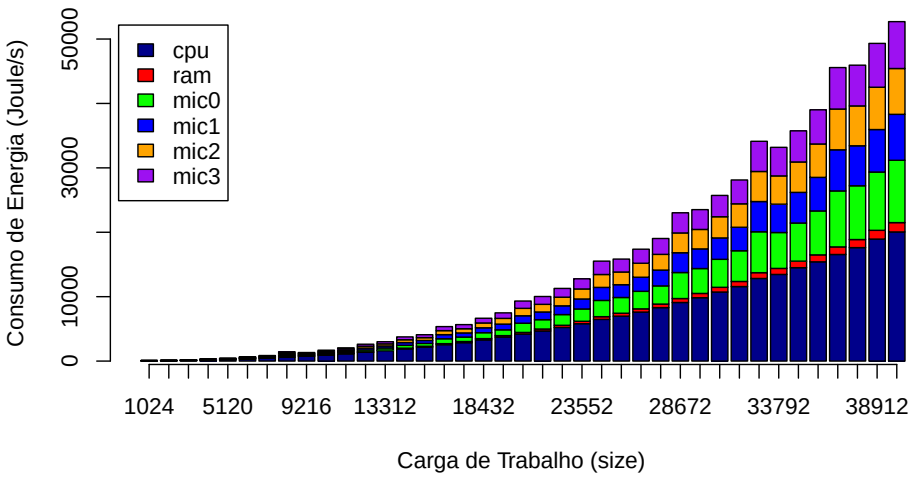


Figura 30 – Energia em modelo *offload* com 4 Xeon Phis (Linpack)

modelos e também é proporcional à carga de trabalho. Uma questão importante é que, neste modelo, o consumo de energia também está concentrado na CPU e muito pouco na memória do coprocessador, demonstrando que, em geral, as execuções do Linpack em memória compartilhada requerem mais poder de processamento que a memória.

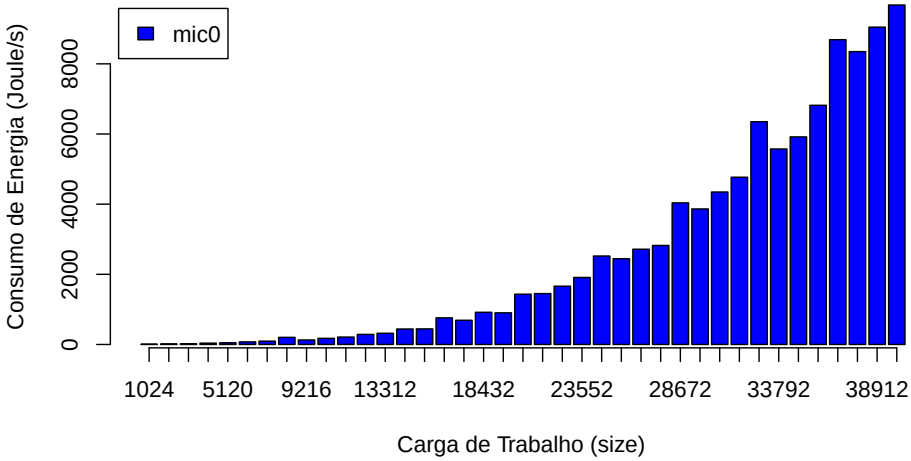


Figura 31 – Energia em modelo *nativo* com 1 Xeon Phi (Linpack).

5.2.2 Consumo de Energia em memória compartilhada (*Linpack*)

Considerando o número de threads e analisando os resultados obtidos com a Linpack na memória compartilhada na figura 32, é possível verificar que nos modelos *host* (CPU) e *offload* (OF1D, OF2D, OF3D e OF4D), o consumo de energia é similar, independentemente do número de threads. O modelo *nativo* (N1D), com o aumento do número de

threads, apresenta redução no consumo de energia e com 64 threads significativos reduzem o consumo em comparação com os outros modelos.

O consumo de energia com threads superiores a 64 foi inferior e demonstraram que a aplicabilidade dos coprocessadores Xeon Phi é recomendada para aplicação com alto nível de paralelismo e baixo uso de processamento de memória.

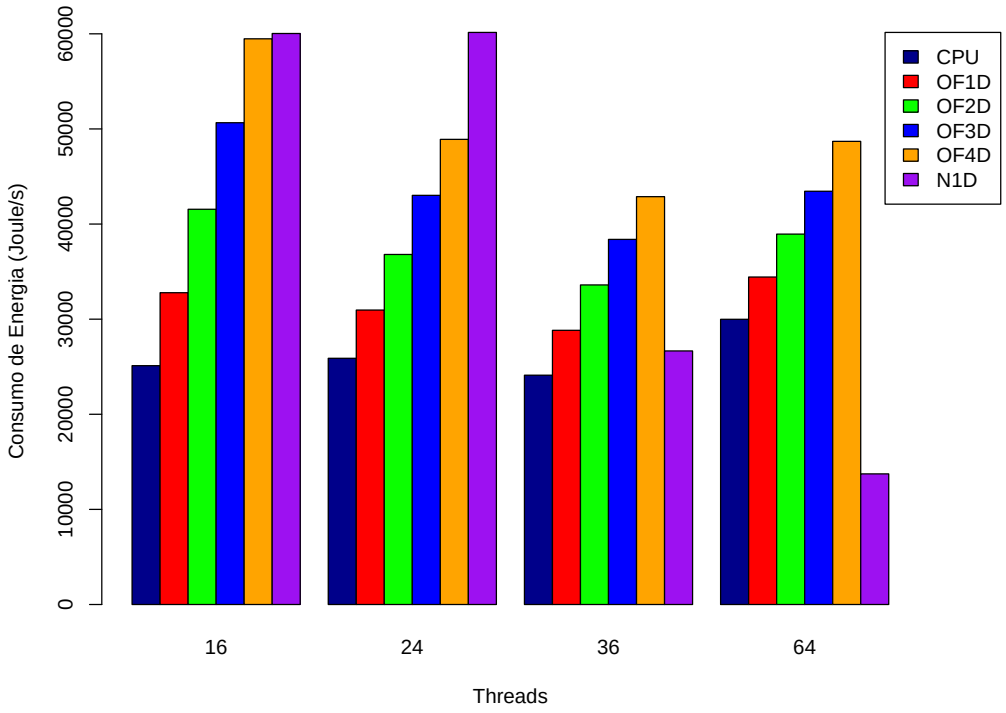


Figura 32 – Consumo de energia em diferentes números de threads (Linpack).

5.2.3 Consumo de Energia em memória distribuída (*HPL 2.1*)

Os resultados obtidos com o benchmark HPL na memória distribuída, apresentados na figura 33, mostram que o consumo de energia dos modelos *offload* é muito maior do que outros. O modelo *offload* OF3D apresentou maior consumo e até maior que o OF4D. Isso demonstra que a adição de coprocessadores não aumentará necessariamente o desempenho e o consumo de energia. Outra característica relevante é que no modelo *offload*, os cenários com 3 Xeon Phis (OF3D), com 16 ou 32 threads, obtiveram um padrão no consumo de energia, mas com 64 threads foi maior em comparação com os cenários de 1, 2 e 4 Xeon Phis. O modelo *host* foi o mais eficiente de energia durante as execuções de benchmark HPL em todos os cenários, mesmo considerando diferentes números de threads.

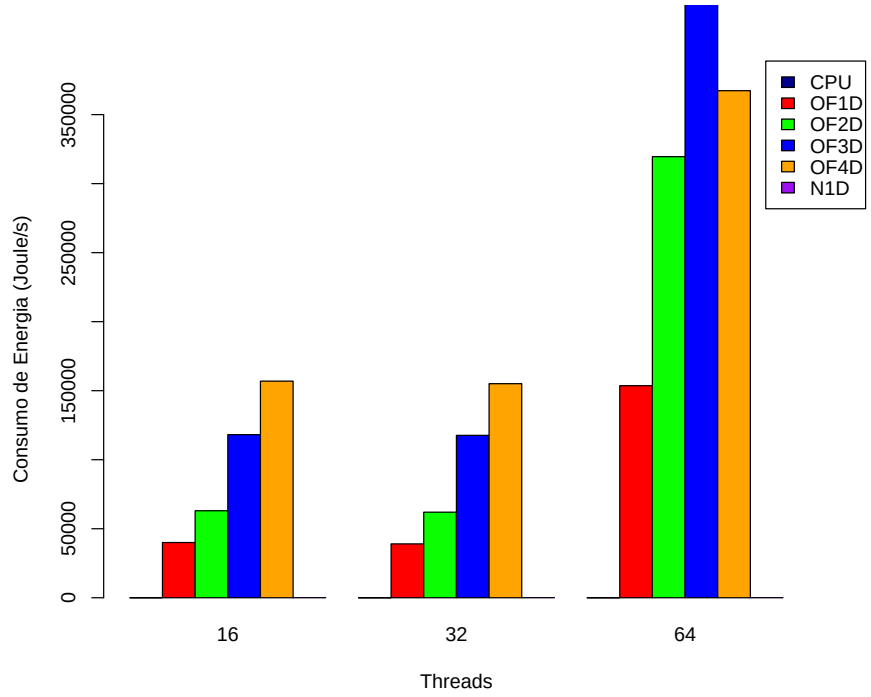


Figura 33 – Consumo de energia em diferentes números de threads (HPL).

5.3 Tempo

Na comparação geral do consumo de energia entre os resultados da Linpack e HPL, percebe-se que, devido à grande diferença de tempo de execução entre eles, gerou alto consumo de energia nas execuções HPL. Na comparação entre as figuras 32 e 33, é possível identificar que houve um aumento no consumo de energia ao aumentar o número de coprocessadores.

5.3.1 Tempo de Execução no consumo de energia

O tempo de execução é um dos fatores que influenciam o desempenho e o consumo de energia. Nos resultados Linpack apresentados na figura 34, é possível verificar que, seguindo o mesmo comportamento apresentado na figura 28, os modelos *host* e *offload* possuem tempos de execução muito semelhantes, enquanto no modelo nativo (N1D) o tempo de execução é muito maior e aumenta de acordo com o tamanho do problema. Esse comportamento de tempo de execução mais longo no modelo *nativo* também foi obtido nos resultados obtidos pelo benchmark HPL.

Comparando o consumo de energia entre os resultados dos benchmarks, a partir da figura 33 é possível verificar que o consumo de energia para 64 threads na HPL começou a partir de valores acima de 100000 J/s, bem acima do máximo consumo obtido em Linpack (figura 30), que foi perto de 50000 J/s. Isso ocorre principalmente devido ao alto tempo de execução do benchmark HPL.

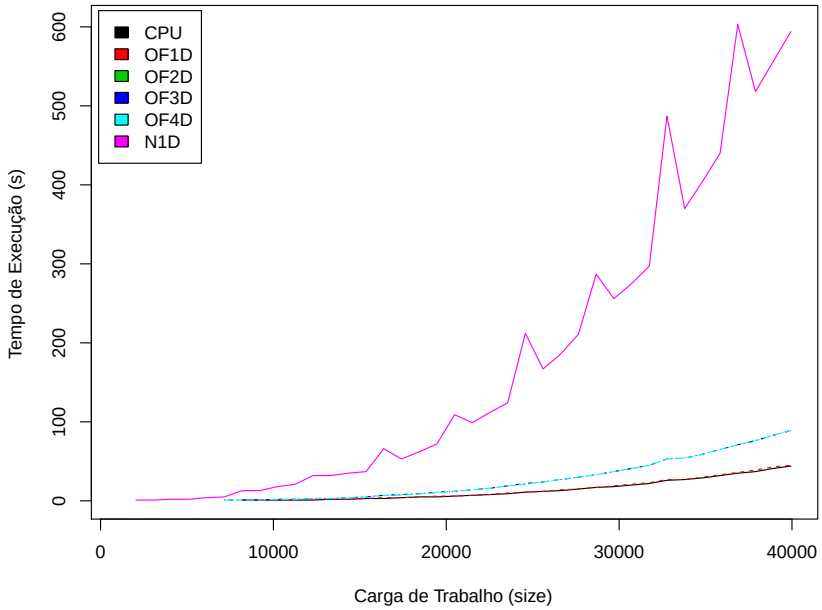
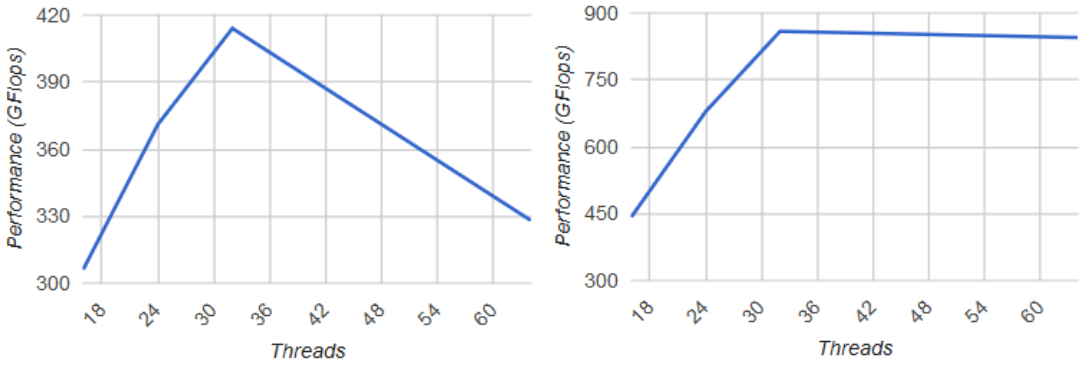


Figura 34 – Tempo de execução (Linpack).

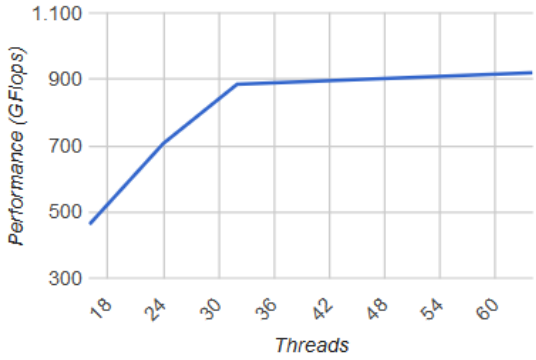
5.3.2 Consumo de Energia por número de threads (*HPL 2.1*)

O número de threads deve estar relacionado principalmente à carga de trabalho a ser executada. Conforme mostrado na figura 35, pequenos problemas devem usar um número menor de threads, enquanto problemas maiores precisam expandir o número de threads para melhorar o desempenho.

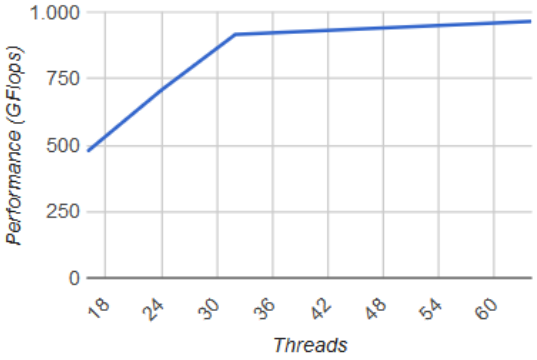


(a) Size = 4096

(b) Size = 16384



(c) Size = 24576



(d) Size = 39936

Figura 35 – Desempenho por número de threads (Linpack)

5.3.3 Consumo de Energia de CPU e RAM (*Linpack*)

As CPUs e DRAM demandam bem maior consumo de energia se comparado com os coprocessadores Intel Xeon Phi. Por isto, os modelos *host* e *offload* mesmo em estado *idle* sem nenhuma execução já consomem mais energia que o modelo *nativo* pois necessitam dos recursos de CPU e memória do *host* enquanto que o *nativo* utiliza apenas processamento do Xeon Phi.

As figuras 36 e 37 apresentam as diferenças na média de consumo de energia entre os modelos da *Intel* diferenciando o uso de energia por recursos de processamento. Os resultados foram obtidos através da execução do *Linpack* com 72 *threads* e configurações iguais para todos os modelos.

Verificando os gráficos é possível identificarmos que:

a) O consumo de energia das duas CPUs se mantém regular nos modelos *host* e *offload*.

b) O consumo de energia da DRAM demonstrou grande variação entre problemas próximos, demonstrando a relação com o multiplicador que o problema utiliza.

c) O modelo *nativo* por não utilizar CPU do *host*, consome muito menos energia que os modelos que utilizam.

d) Percebe-se que no modelo *offload* independente de uso de 1,2,3 ou 4 coprocessadores *xeon phi*, mantém regular o consumo de energia da CPU e DRAM. E também mantém um consumo mínimo de energia dos coprocessadores.

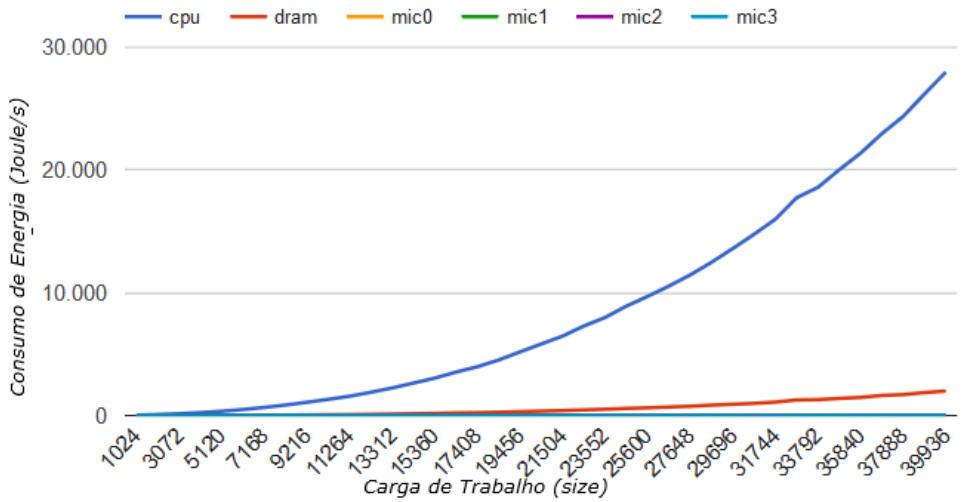


Figura 36 – Consumo de Energia no modelos que usam CPUs

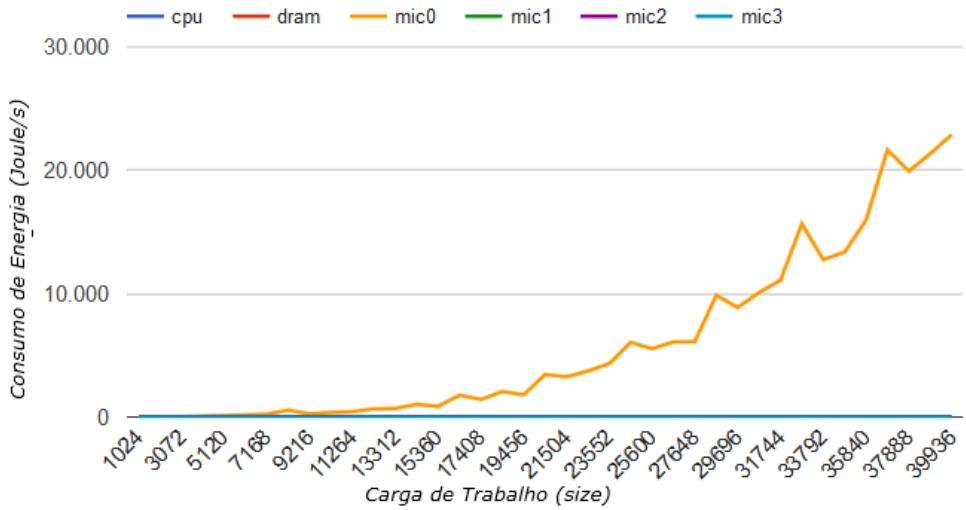


Figura 37 – Consumo de Energia no modelo *nativo*

As figuras 38, 39, 40 e 41 apresentam o consumo de energia das duas CPUs e DRAM para o modelo *offload*.

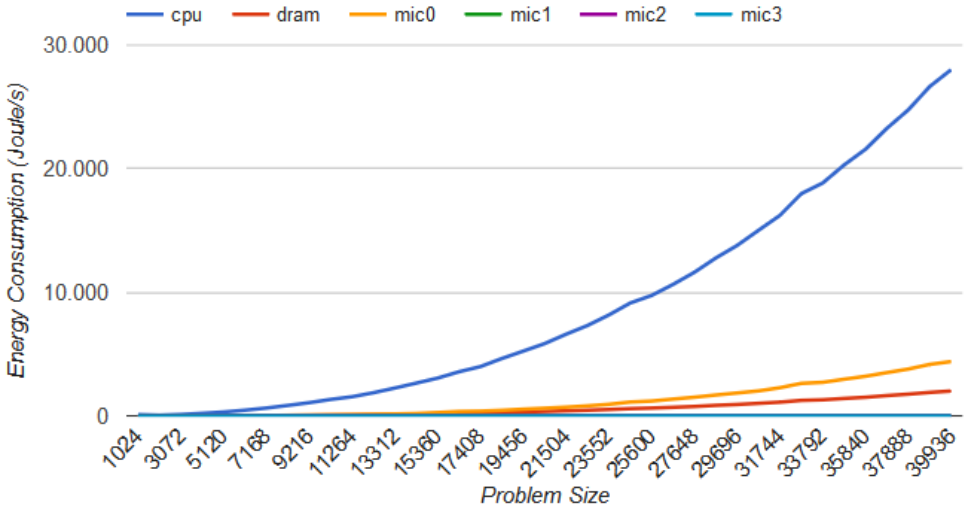


Figura 38 – Consumo de Energia no modelo *offload* 1 xeon phi

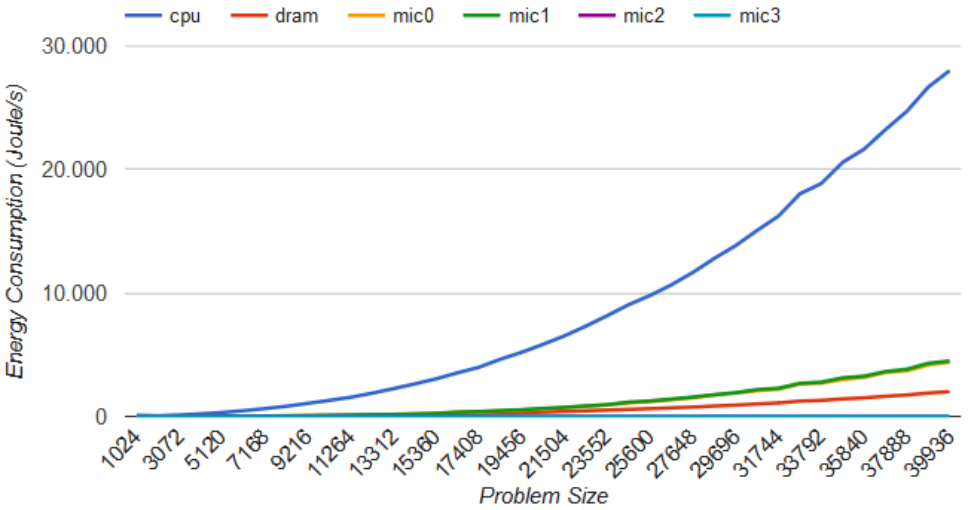


Figura 39 – Consumo de Energia no modelo *offload* 2 xeon phi

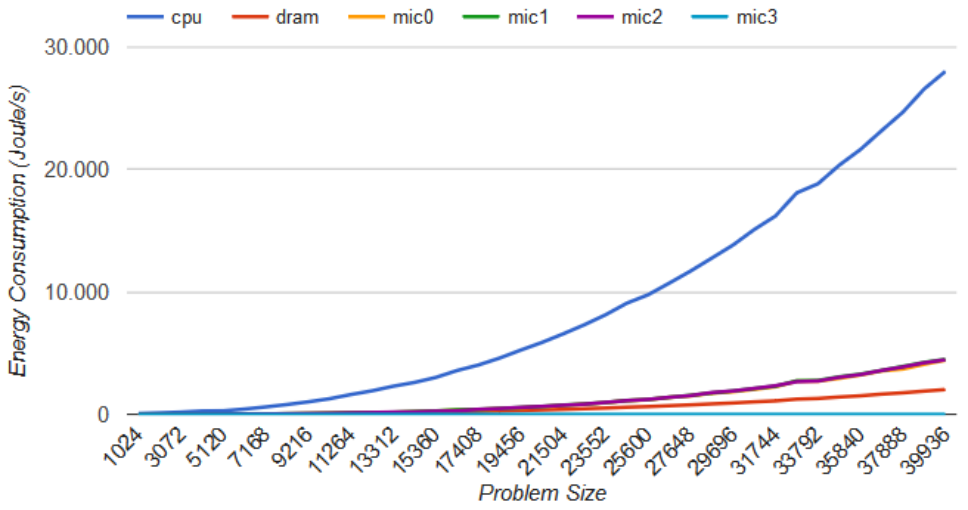


Figura 40 – Consumo de Energia no modelo *offload* 3 xeon phi

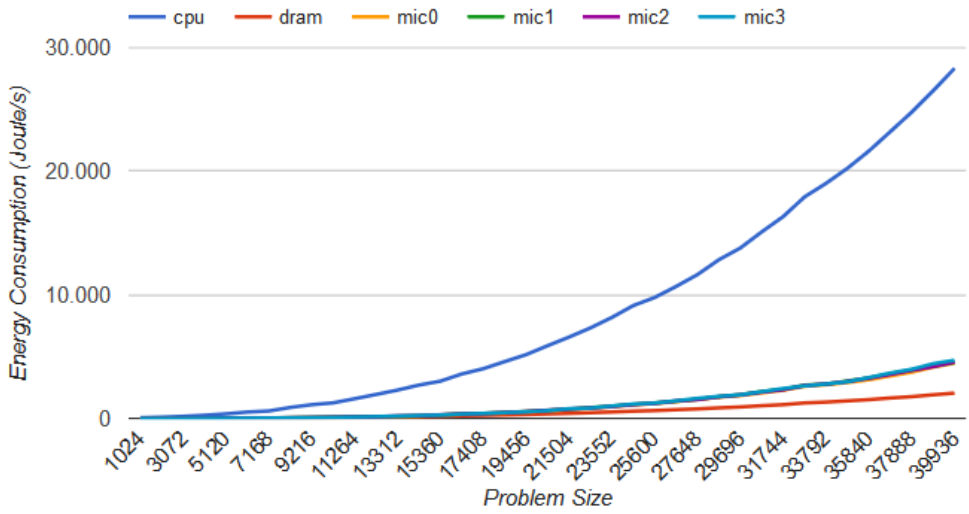


Figura 41 – Consumo de Energia no modelo *offload* 4 xeon phi

5.4 Desempenho por Watt

5.4.1 Desempenho por Watt (*Linpack*)

Aplicando a métrica *performance per watt* (figura 42), é possível identificar que o modelo *nativo* (N1D) demonstra menor valor em comparação com outros modelos sobre os resultados de benchmarks da Linpack .

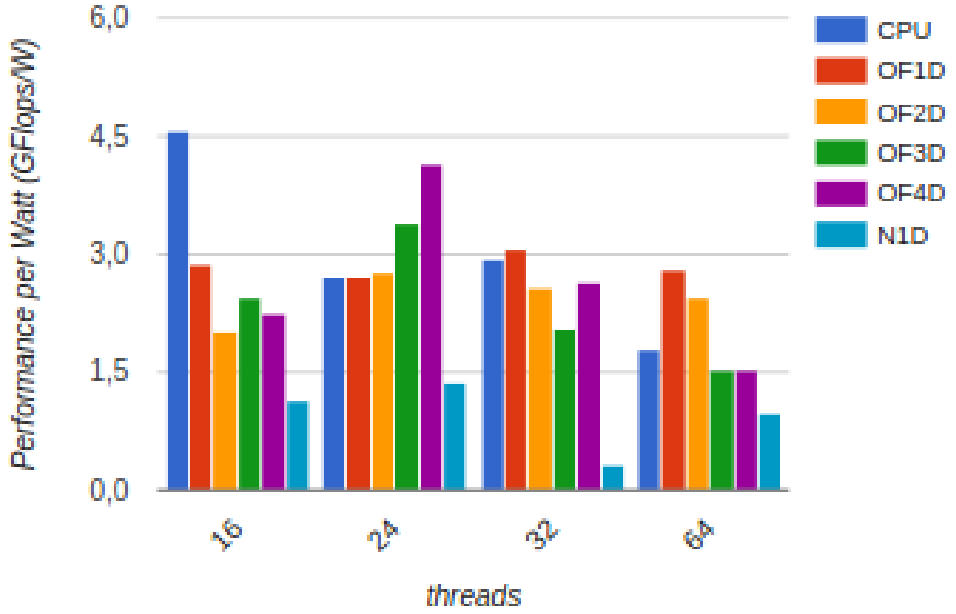


Figura 42 – Desempenho por Watt em diferentes números de threads (Linpack).

Entre os cenários sobre modelos *offload* (OF1D, OF2D, OF3D e OF4D), o intervalo foi entre 1,5 e 4 com variações semelhantes usando 24 threads. Mas o modelo *host* (CPU) usando 16 threads foi o cenário de maior valor atingindo 4,5. O modelo *host* (CPU)

usando 16 threads obteve a melhor relação PPW com um valor de 4.5. Este valor é um excelente resultado em comparação com o ranking Green 500 de junho de 2017, onde apenas 24 supercomputadores estão com PPW maior do que 4.5. O modelo *nativo* obteve a pior relação com valores menores que 1,5 em todos os cenários independentemente do número de threads.

5.4.2 Desempenho por Watt (*HPL 2.1*)

A figura 43 mostra os valores de PPW muito maiores obtidos nas execuções de benchmark HPL. Esse comportamento é muito importante porque demonstra que as execuções HPL apenas no *host*, mesmo usando uma abordagem distribuída, são mais eficientes em relação ao desempenho e ao consumo de energia do que executar o Linpack para memória compartilhada.

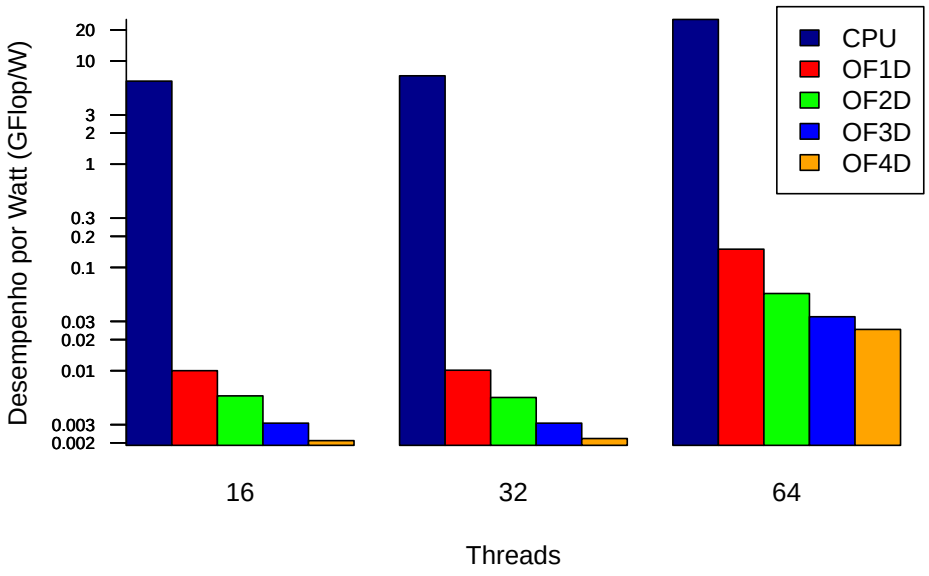


Figura 43 – Desempenho por Watt em diferentes números de threads (HPL).

6 CONCLUSÃO

Este trabalho apresentou um estudo e metodologia para avaliação das principais características relacionadas com desempenho e consumo de energia aplicados sobre modelos de programação disponíveis em arquiteturas *manycore* utilizando coprocessadores Intel Xeon Phi.

6.1 Considerações Finais

Nos primeiros capítulos foram abordados conceitos e fundamentos sobre arquiteturas Intel MIC(Many Integrated Core) e coprocessadores Xeon Phi relevantes a este trabalho. Também foram abordados informações sobre métricas e técnicas para avaliação de desempenho e consumo de energia em ambientes de alto desempenho. Foram analisados diversos trabalhos acadêmicos mais recentes e relevantes com o tema apresentando suas principais abordagens e objetivos e principalmente quais os diferenciais comparado com este trabalho. Os capítulos restantes abordaram a metodologia aplicada para a organização e condução deste trabalho juntamente com as ferramentas utilizadas, finalizando com a apresentação dos resultados preliminares.

Os resultados apresentados foram obtidos através da execução e monitoramento de diversos cenários utilizando os *benchmarks Linpack*, *HPL 2.1* e *HPCG*, que demonstraram grande variação no desempenho e consumo de energia entre os modelos de programação e também entre os critérios e características relacionadas e avaliadas.

Com relação aos modelos de programação mais eficientes em desempenho e energia, os modelos *host* e *offload* demonstraram bastante semelhança na maioria das simulações realizadas tanto no desempenho quanto consumo de energia, ao contrário do modelo *nativo*, que apresentou menor desempenho mas em contrapartida significativo consumo eficiente de energia.

Sobre consumo de energia entre memória compartilhada e distribuída, os modelos *host* e *offload* demonstraram-se mais adequados para uso com memória compartilhada devido ao controle do processamento e uso de memória centralizado no host.

A escolha e configuração adequada dos critérios influencia diretamente na otimização do desempenho e consumo de energia. No ambiente de Execução, o critério *align* obteve maior desempenho com valor 8(*double precision*). O *page size* deve ser dimensionado para execuções com uso maior que 100Mb de memória RAM. A definição do número de *cores* influencia diretamente tanto no desempenho quanto no consumo de energia juntamente com a delimitação do número de CPUs a considerar no processamento

da aplicação.

Para grau de paralelismo, o número de *threads* na maioria das execuções também influenciou diretamente no desempenho e energia, e quanto maior o número threads confirmou maior o desempenho. Mas houve algumas exceções, nos modelos *host* e *offload*, os cenários com uso de mais de 200 *threads* tiveram o mesmo desempenho utilizando apenas 72. A escolha de 24 threads demonstrou a pior média de desempenho e maior consumo de energia perante os demais simulados.

No contexto da aplicação, o *dimensionamento do problema* e *tempo de execução* são critérios que devem ter seus limites aceitáveis bem definidos pelo usuário, pois através para processamento com tamanhos de problemas pequenas deve-se considerar o uso do modelo *nativo* que não utiliza nenhum recurso computacional da CPU, reduzindo bastante o consumo de energia com desempenho satisfatório. E para grandes cargas de trabalho que exigem maior uso de processamento e memória, utilizar os modelos *host* e *offload*.

Uma das principais contribuições deste trabalho é a possibilidade de identificar a relevância dos critérios para otimização de desempenho e consumo de energia salientando quais características e aspectos devem ser considerados na escolha do modelo de programação a ser utilizado sobre arquiteturas Intel Xeon Phi.

6.2 Sugestões para Trabalhos Futuros

Este trabalho abordou o desempenho e consumo de energia em arquiteturas da Intel Xeon Phi considerando os coprocessadores da família x100. Durante este trabalho, em 2017, a Intel lançou novos processadores da família x200. Agora definidos como processadores e não mais coprocessadores, estes novos modelos possuem uma arquitetura diferente da família x100 com proposições diretamente alinhadas para muito maior desempenho e ainda menor consumo de energia mas que também poderiam sofrer uma avaliação conforme a metodologia apresentada neste trabalho. Avaliação destes novos modelos Xeon Phis seria muito interessante e também poderia incluir o impacto do uso de cache e memória nestes novos modelos pois possuem grande capacidade e possuem maior variedade de parametrizações.

Referências

BALLADINI, J. et al. Impact of parallel programming models and CPUs clock frequency on energy consumption of HPC systems. *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, p. 16–21, 2011. ISSN 21615322. Citado 4 vezes nas páginas 36, 38, 43 e 47.

BARRACHINA, S. et al. An Integrated Framework for Power-Performance Analysis of Parallel Scientific Workloads. *ENERGY 2013, The Third International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, n. c, p. 114–119, 2013. Disponível em: <http://www.thinkmind.org/index.php?view=article{&}articleid=energy{_}2013{_}5>. Citado na página 36.

CHRYSOS, G. Intel Xeon Phi Coprocessor - the Architecture. *Hot Chips Conference 2012*, n. Figure 3, p. 1–8, 2012. Citado na página 29.

DONGARRA, J. *Frequent Asked Questions on the LINPACK Benchmark*. 2015. Disponível em: <http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html>. Citado 2 vezes nas páginas 50 e 51.

DONGARRA, J.; LUSZCZEK, P.; HEROUX, M. *HPCG Benchmark*. 2017. Disponível em: <http://www.hpcg-benchmark.org/index.html>. Citado na página 52.

GIEFERS, H. et al. Analyzing the energy-efficiency of sparse matrix multiplication on heterogeneous systems: A comparative study of GPU, Xeon Phi and FPGA. In: *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2016. p. 46–56. ISBN 978-1-5090-1953-3. Disponível em: <http://ieeexplore.ieee.org/document/7482073/>. Citado 2 vezes nas páginas 27 e 47.

GREEN500. Energy Efficient High Performance Computing Power Measurement Methodology. 2010. Citado na página 36.

HENAO, J. A. G. et al. enerGyPU and enerGyPhi Monitor for Power Consumption and Performance Evaluation on Nvidia Tesla GPU and Intel Xeon Phi. n. May, 2016. Citado 6 vezes nas páginas 13, 45, 46, 47, 63 e 71.

HSU, C.-H.; KREMER, U. The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction. *ACM SIGPLAN Notices*, v. 38, n. 5, p. 38, 2003. ISSN 03621340. Citado na página 43.

IGUAL, F. D. et al. A power measurement environment for PCIe accelerators. *Computer Science - Research and Development*, Springer Berlin Heidelberg, v. 30, n. 2, p. 115–124, 2015. ISSN 1865-2034. Disponível em: <http://link.springer.com/10.1007/s00450-014-0266-8>. Citado 2 vezes nas páginas 36 e 47.

INTEL. Intel Xeon Phi Coprocessor x100 Product Family. n. February, 2015. Citado 2 vezes nas páginas 31 e 32.

Intel Corp. *Intel Xeon Phi Product Family: Product Brief*. 2015. Disponível em: <https://www-ssl.intel.com/content/www/us/en/high-performance-computing/high-performance-xeon-phi-coprocessor-brief.html>. Citado na página 30.

Intel MKL Heterogeneous. *Heterogeneous Support in the Intel Distribution for LINPACK Benchmark | Intel Software*. 2017. Citado na página 52.

Intel MKL Variables. *Environment Variables | Intel® Software*. 2017. Disponível em: <https://software.intel.com/en-us/mkl-linux-developer-guide-environment-variables>. Citado na página 51.

Intel RAPL. *Running Average Power Limit – RAPL | 01.org*. 2014. Disponível em: <https://01.org/blogs/2014/running-average-power-limit-\T1\textendash-rapl>. Citado 2 vezes nas páginas 38 e 39.

INTEL, T. et al. History of Many-Core Leading to Intel ® Xeon Phi ™. 2009. Citado na página 28.

Intel Thread Affinity. *OpenMP* Thread Affinity Control | Intel Software*. 2017. Disponível em: <https://software.intel.com/en-us/articles/openmp-thread-affinity-control>. Citado na página 62.

Intel Xeon Phi. *Coprocessadores Intel*. 2015. Disponível em: <https://www.intel.com.br/content/www/br/pt/products/processors/xeon-phi/xeon-phi-coprocessors.html>. Citado na página 29.

KARPUSENKO, V. Parallel Programming and Optimization with Intel Xeon Phi Coprocessors Introduction Parallel Programming and Optimization with Intel Xeon Phi Coprocessors Handbook on the Development and Optimization of Parallel Applications for Intel Xeon ® Process. *Colfax Workshop*, v. 1, n. 2, 2013. Disponível em: <http://www.colfax-intl.com/nd/xeonphi/book.aspx>. Citado 2 vezes nas páginas 33 e 34.

KRISTA. *Introduction to the HPP - Heterogeneous Process... | NDN Hub*. 2015. Disponível em: <https://community.cmc.ca/docs/DOC-2196>. Citado na página 28.

LAKOMSKI, D.; ZONG, Z.; JIN, T. Optimal Balance between Energy and Performance in Hybrid Computing Applications. 2015. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=7393>. Citado 4 vezes nas páginas 42, 43, 47 e 72.

LAWSON, G.; SOSONKINA, M.; SHEN, Y. Energy Evaluation for Applications with Different Thread Affinities on the Intel Xeon Phi. *Workshop on Applications for Multi-Core Architectures (WAMCA)*, p. 54–59, 2014. Citado 3 vezes nas páginas 44, 45 e 47.

LAWSON, G. et al. Runtime Power Limiting of Parallel Applications on Intel Xeon Phi Processors. In: *2016 4th International Workshop on Energy Efficient Supercomputing (E2SC)*. IEEE, 2016. p. 39–45. ISBN 978-1-5090-3856-5. Disponível em: <http://ieeexplore.ieee.org/document/7830507/>. Citado 2 vezes nas páginas 44 e 47.

LORENZO, O. G. et al. Power and Energy Implications of the Number of Threads Used on the Intel Xeon Phi. v. 2, p. 55–65, 2015. Citado na página 47.

LORENZON, A. F.; CERA, M. C.; BECK, A. C. S. Performance and energy evaluation of different multi-threading interfaces in embedded and general purpose systems. *Journal of Signal Processing Systems*, v. 80, n. 3, p. 295–307, 2015. ISSN 19398115. Citado 2 vezes nas páginas 45 e 47.

MATTSON, T. The Future of Many Core Computing : A tale of two processors. n. January, p. 79, 2010. Citado na página 27.

MPSS, I. Intel Manycore Platform Software Stack (Intel MPSS) | Intel Software. 2014. Disponível em: <https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss>. Citado 6 vezes nas páginas 30, 31, 34, 52, 53 e 54.

MPSS, I. Intel Manycore Platform Software Stack (Intel MPSS) | Intel Software. 2014. Disponível em: <https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss>. Citado na página 35.

NCC da UNESP. *GridUNESP*. 2017. Disponível em: <http://www.unesp.br/gridunesp/>. Citado na página 56.

Netlib BLAS. *BLAS (Basic Linear Algebra Subprograms)*. 2017. Disponível em: <http://www.netlib.org/blas/>. Citado na página 50.

Netlib HPL. *HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*. 2017. Disponível em: <http://www.netlib.org/benchmark/hpl/>. Citado 2 vezes nas páginas 50 e 51.

Robert Reed. *Best Known Methods for Using OpenMP* on Intel Many Integrated Core (Intel MIC) Architecture | Intel Software*. 2017. Disponível em: <https://software.intel.com/en-us/articles/best-known-methods-for-using-openmp-on-intel-many-integrated-core-intel-mic-architecture>. Citado na página 62.

ROSTIROLLA, G. et al. GreenHPC: A novel framework to measure energy consumption on HPC applications. *2015 Sustainable Internet and ICT for Sustainability, SustainIT 2015*, 2015. Citado 3 vezes nas páginas 36, 37 e 47.

Rupert Goodwins. *Intel unveils many-core Knights platform for HPC | ZDNet*. 2010. Disponível em: <http://www.zdnet.com/article/intel-unveils-many-core-knights-platform-for-hpc/>. Citado na página 28.

SHAO, Y. S.; BROOKS, D. Energy characterization and instruction-level energy model of Intel's Xeon Phi processor. In: *International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2013. p. 389–394. ISBN 978-1-4799-1235-3. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6629328>. Citado 2 vezes nas páginas 41 e 47.

SHARMA, S.; HSU, C.-H. Making a Case for a Green500 List *. 2006. Citado na página 36.

SI, M. et al. MT-MPI: Multithreaded MPI for Many-Core Environments. *Proceedings of the 28th ACM international conference on Supercomputing - ICS '14*, 2014. Citado 2 vezes nas páginas 43 e 47.

SMEJKAL, T. et al. E-Team : Practical Energy Accounting for Multi-Core Systems E-Team : Practical Energy Accounting for Multi-Core Systems. *Atc*, 2017. Citado na página 46.

SUBRAMANIAM, B.; FENG, W.-c. Statistical Power and Performance Modeling for Optimizing the Energy Efficiency of Scientific Computing. *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, p. 139–146, 2010. Citado na página 37.

TOP500.ORG. Green500 | TOP500 Supercomputer Sites. 2017. Disponível em: <https://www.top500.org/green500/>. Citado na página 23.

VWEAVER. *Various Research Projects*. 2017. Disponível em: <http://web.eece.maine.edu/{~}vweaver/project>. Citado 2 vezes nas páginas 55 e 56.

Wiki BLAS. *Basic Linear Algebra Subprograms*. 2017. Disponível em: <https://en.wikipedia.org/wiki/Basic{_}Linear{_}Algebra{_}Su>. Citado na página 50.

Anexos

ANEXO A – Resultados

A.1 Cenários

Os arquivos de cenários são necessários para execução do benchmark Linpack e HPL. Nele contém os parâmetros de configurações como a definição dos recursos de hardware utilizar e também de otimização de desempenho.

A.1.1 Cenários para *linpack*

Abaixo estão 1) a descrição dos parâmetros e 2) um exemplo de arquivo de cenário para execução do Linpack:

Descrição dos parâmetros no arquivo:

```
0) script file
1) cenario
2) monitor time interval
3) problem size
4) dimension
5) align 4 or 8
6) OMP_NUM_THREADS (max 72 CPU)
7) MIC_OMP_NUM_THREADS (max 244 Xeon Phi) (empty for CPU cenarios!!)
8) KMP_AFFINITY(compact, balanced or scatter)
```

Exemplo de arquivo:

```
#CPU_ONLY
linpack.sh:LP_CPU_ONLY:1:1024:1024:4:8::nowarnings,compact,1,0,granularity=fine:
linpack.sh:LP_CPU_ONLY:2:2048:2048:4:8::nowarnings,compact,1,0,granularity=fine:
#OFFLOAD_1_DEVICE
linpack.sh:LP_OFFLOAD_1_DEVICE:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
linpack.sh:LP_OFFLOAD_1_DEVICE:2:2048:2048:4:32:240:nowarnings,compact,1,0,granularity=fine:
#OFFLOAD_2_DEVICES
linpack.sh:LP_OFFLOAD_2_DEVICES:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
linpack.sh:LP_OFFLOAD_2_DEVICES:2:2048:2048:4:32:240:nowarnings,compact,1,0,granularity=fine:
#OFFLOAD_3_DEVICES
linpack.sh:LP_OFFLOAD_3_DEVICES:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
linpack.sh:LP_OFFLOAD_3_DEVICES:2:2048:2048:4:32:240:nowarnings,compact,1,0,granularity=fine:
#OFFLOAD_4_DEVICES
```

```
linpack.sh:LP_OFFLOAD_4_DEVICES:1:1024:1024:4:32:240:nowarnings,compact,1,0,granularity=fine:
linpack.sh:LP_OFFLOAD_4_DEVICES:2:2048:2048:4:32:240:nowarnings,compact,1,0,granularity=fine:
#NATIVE_1_DEVICE
linpack.sh:LP_NATIVE_1_DEVICE:1:1024:1024:4::240:nowarnings,compact,1,0,granularity=fine:
linpack.sh:LP_NATIVE_1_DEVICE:2:2048:2048:4::240:nowarnings,compact,1,0,granularity=fine:
```

A.1.2 Cenários para *hpl linpack*

Abaixo estão 1) a descrição dos parâmetros e 2) um exemplo de arquivo de cenário para execução do HPL:

Descrição dos parâmetros no arquivo:

```
0):nome do script de execucao
1):cenario
2):  time
3):  N - problem size 8*N*N/(P*Q) definir no max 70% da memory
4):  NB - Numero de block size .  OF1D=960 OF2D=1024 OF3D=1200
5):  P - numero de rows P<=Q
6):  Q - numero de columns
7):  Align:  4 ou 8
8):  MPI_PROC_NUM onde deve ser igual a P*Q
9) OMP_NUM_THREADS (max 72 CPU)
10) MIC_OMP_NUM_THREADS (max 244 Xeon Phi) (empty for CPU cenarios!!)
11):  MPI_PER_NODE(-perhost)=numero sockets no system - 2 ou 4 é o padrªo
12):NUMMIC - numero de mics por node utilizado ignorado se HPL_MIC_DEVICE setado - HPL_PNUMICS=0
nao usa MICs
13):HPL_PNUMMICS numero de mics utilizado ignorado se HPL_MIC_DEVICE setado - HPL_PNUMICS=0 nao
usa MICs
14):HPL_MIC_DEVICE - quais mics serªo executados
```

Exemplo de arquivo:

```
#CPU_ONLY
mp_linpack.sh:MP_CPU_ONLY:5:10000:256:1:2:4:2:2:240:1:0:0:
mp_linpack.sh:MP_CPU_ONLY:5:11024:256:1:2:4:2:2:240:1:0:0:
#OFFLOAD_1_DEVICE
mp_linpack_offload_1_device:MP_OFFLOAD_1_DEVICE:5:1024:960:4:4:4:16:32:240:2:1:1:0
mp_linpack_offload_1_device:MP_OFFLOAD_1_DEVICE:5:2048:960:4:4:4:16:32:240:2:1:1:0
#OFFLOAD_2_DEVICES
mp_linpack_offload_2_devices:MP_OFFLOAD_2_DEVICES:5:1024:1024:4:4:4:16:32:240:4:2:2:0,1
```

```
mp_linpack_offload_2_devices:MP_OFFLOAD_2_DEVICES:5:2048:1024:4:4:4:16:32:240:4:2:2:0,1
#OFFLOAD_3_DEVICES
mp_linpack_offload_3_devices:MP_OFFLOAD_3_DEVICES:5:1024:1200:4:4:4:16:32:240:2:3:2:0,1,2
mp_linpack_offload_3_devices:MP_OFFLOAD_3_DEVICES:5:2048:1200:4:4:4:16:32:240:2:3:2:0,1,2
#OFFLOAD_4_DEVICES
mp_linpack_offload_4_devices:MP_OFFLOAD_4_DEVICES:5:1024:1200:4:4:4:16:32:240:4:3:2:0,1,2,3
mp_linpack_offload_4_devices:MP_OFFLOAD_4_DEVICES:5:2048:1200:4:4:4:16:32:240:4:3:2:0,1,2,3
```

A.2 Logs

Os arquivos de logs são gerados durante a execução dos benchmarks.

A.2.1 Logs do *micsmc*

Abaixo está um exemplo de arquivo de log gerado pelo micsmc da Intel. Nele contém dados de temperatura, frequência e memória dos coprocessadores e também o percentual de uso de cada núcleo dos coprocessadores:

```
        mic0 (info):
Device Series:  ...........  Intel(R) Xeon Phi(TM) coprocessor x100 family
Device ID: ..............  0x225c
Stepping:  ..............  0x2
Substepping:  ............  0x0
Coprocessor OS Version:  ..  2.6.38.8+mpss3.6.1
Flash Version:  ...........  2.1.02.0391
Host Driver Version:  .....  3.6.1-1 (root@phi03.ncc.unesp.br)
Number of Cores:  .........  61


        mic0 (temp):
Cpu Temp:  ..............  49.00 C
Memory Temp:  ............  38.00 C
Fan-In Temp:  ............  34.00 C
Fan-Out Temp:  ...........  38.00 C
Core Rail Temp:  .........  38.00 C
Uncore Rail Temp:  ........  40.00 C
Memory Rail Temp:  ........  40.00 C


        mic0 (freq):
Core Frequency:  .........  1.24 GHz
```

```
Total Power:  .............  100.00 Watts

Low Power Limit:  .........  315.00 Watts

High Power Limit:  ........  375.00 Watts

Physical Power Limit:  ....  395.00 Watts


        mic0 (mem):

Free Memory:  .............  15023.55 MB

Total Memory:  ............  15513.17 MB

Memory Usage:  ............  489.61 MB


        mic0 (cores):

Device Utilization:  User:  0.00%, System:  0.06%, Idle:  99.94%

Per Core Utilization (61 cores in use)

Core #1:  User:  0.00%, System:  0.24%, Idle:  99.76%

Core #2:  User:  0.00%, System:  0.24%, Idle:  99.76%

Core #3:  User:  0.00%, System:  0.24%, Idle:  99.76%

Core #4:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #5:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #6:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #7:  User:  0.00%, System:  0.24%, Idle:  99.76%

Core #8:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #9:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #10:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #11:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #12:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #13:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #14:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #15:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #16:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #17:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #18:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #19:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #20:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #21:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #22:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #23:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #24:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #25:  User:  0.00%, System:  0.00%, Idle:  100.00%

Core #26:  User:  0.00%, System:  0.00%, Idle:  100.00%
```

```
Core #27:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #28:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #29:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #30:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #31:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #32:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #33:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #34:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #35:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #36:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #37:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #38:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #39:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #40:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #41:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #42:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #43:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #44:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #45:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #46:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #47:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #48:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #49:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #50:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #51:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #52:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #53:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #54:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #55:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #56:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #57:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #58:  User:  0.00%, System:  0.00%, Idle:  100.00%
Core #59:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #60:  User:  0.00%, System:  0.24%, Idle:  99.76%
Core #61:  User:  0.00%, System:  0.71%, Idle:  99.29%
```

A.2.2 Logs do RAPL e Powercap

Abaixo está um exemplo de arquivo de saída para registro de consumo de energia(em microjoules) das CPUS e DRAM do servidor.

Exemplo de arquivo de log do powercap:

```
Arquivo:  mp_linpack-MP_CPU_ONLY-21504-0106165905-20170106-MP.log:
20170106165905-262143999938-133254066101-262143999938-207828408996-262143999938-227067413879
20170106165914-262143999938-134076375488-262143999938-208504897705-262143999938-227264660766
20170106165923-262143999938-135124925781-262143999938-209557431945-262143999938-227516351440
20170106165932-262143999938-136329221008-262143999938-210687190429-262143999938-227773721984
20170106165941-262143999938-137586725402-262143999938-211839366882-262143999938-228038999389
20170106165950-262143999938-138827495239-262143999938-212995765258-262143999938-228292125488
20170106165958-262143999938-140072476806-262143999938-214161224914-262143999938-228551483642
20170106170007-262143999938-141303306518-262143999938-215301313720-262143999938-228813712219
20170106170016-262143999938-142563106323-262143999938-216500320434-262143999938-229070375488
20170106170025-262143999938-143796224060-262143999938-217661649841-262143999938-229327343200
20170106170034-262143999938-145032167480-262143999938-218790502563-262143999938-229582016723
20170106170043-262143999938-146277847900-262143999938-219971892578-262143999938-229834647094
20170106170052-262143999938-147515031066-262143999938-221126711303-262143999938-230092082885
20170106170100-262143999938-148748549133-262143999938-222267377685-262143999938-230352474182
20170106170109-262143999938-149996348693-262143999938-223430750915-262143999938-230601820922
20170106170118-262143999938-151217239562-262143999938-224565766479-262143999938-230854033508
20170106170127-262143999938-152488171264-262143999938-225740827270-262143999938-231113999938
20170106170136-262143999938-153739536437-262143999938-226891174194-262143999938-231359827880
20170106170145-262143999938-154966843383-262143999938-228033358764-262143999938-231618196594
20170106170154-262143999938-156217314575-262143999938-229191442138-262143999938-231865477783
20170106170202-262143999938-157443393066-262143999938-230311237854-262143999938-232118117004
20170106170211-262143999938-158691396545-262143999938-231467681884-262143999938-232372666870
20170106170220-262143999938-159932363220-262143999938-232593197265-262143999938-232611471984
20170106170229-262143999938-161178560241-262143999938-233755678039-262143999938-232871497253
20170106170238-262143999938-162421826354-262143999938-234888379638-262143999938-233116678466
20170106170247-262143999938-163681463378-262143999938-236063633117-262143999938-233360308105
20170106170256-262143999938-164913850341-262143999938-237200649902-262143999938-233612402709
20170106170304-262143999938-166142349731-262143999938-238309142211-262143999938-233840123474
20170106170313-262143999938-167406141113-262143999938-239467654968-262143999938-234099795776
20170106170322-262143999938-168637190551-262143999938-240586277648-262143999938-234344390197
20170106170331-262143999938-169878284423-262143999938-241736932983-262143999938-234589722717
```