ADICIONANDO A INTERFACE THREADING BUILDING BLOCKS EM BENCHMARK DE AVALIAÇÃO DE DESEMPENHO E CONSUMO ENERGÉTICO

Henrique Fan da Silva [1]

Claudio Schepke [2]

Resumo:

O objetivo desse trabalho é agregar mais uma interface de programação paralela para medir o desempenho e consumo de energia de diferentes arquiteturas multinúcleo em um benchmark desenvolvido pelo estudante do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da Universidade Federal do Pampa (UNIPAMPA) Adriano Marques Garcia, que compara diferentes interfaces de programação paralela. O benckmark é composto por 13 aplicações, que são utilizadas em seu trabalho. Nesta pesquisa as aplicações estão sendo desenvolvidas utilizando a interface de programação paralela Intel Threading Building Blocks (Intel TBB). Tais aplicações serão comparadas com as outras interfaces de programação já implementadas, como Pthreads, OpenMP, MPI-1 e MPI-2. Será feita a análise da interface de programação paralela Intel Threading Building Blocks, e verificar se ela é uma boa alternativa para medir o desempenho e o consumo de energia de arquiteturas multinúcleos. Para atender esses objetivos, aplicações que estão sendo implementadas serão comparadas com as outras interfaces já usadas. A partir daí pode-se ter uma conclusão se a interface é uma boa alternativa para medir o desempenho e o consumo de energia.

Palavras-chave: BENCHMARK THREADING BUILDING BLOCKS CONSUMO ENERGÉTICO DESEMPENHO

Modalidade de Participação: Iniciação Científica

ADICIONANDO A INTERFACE THREADING BUILDING BLOCKS EM BENCHMARK DE AVALIAÇÃO DE DESEMPENHO E CONSUMO ENERGÉTICO

[1] Aluno de graduação. fanhenrique@gmail.com. Autor principal

[2] Docente. claudioschepke@unipampa.edu.br. Orientador

ADICIONANDO A INTERFACE THREADING BUILDING BLOCKS EM BENCHMARK DE AVALIAÇÃO DE DESEMPENHO E CONSUMO ENERGÉTICO

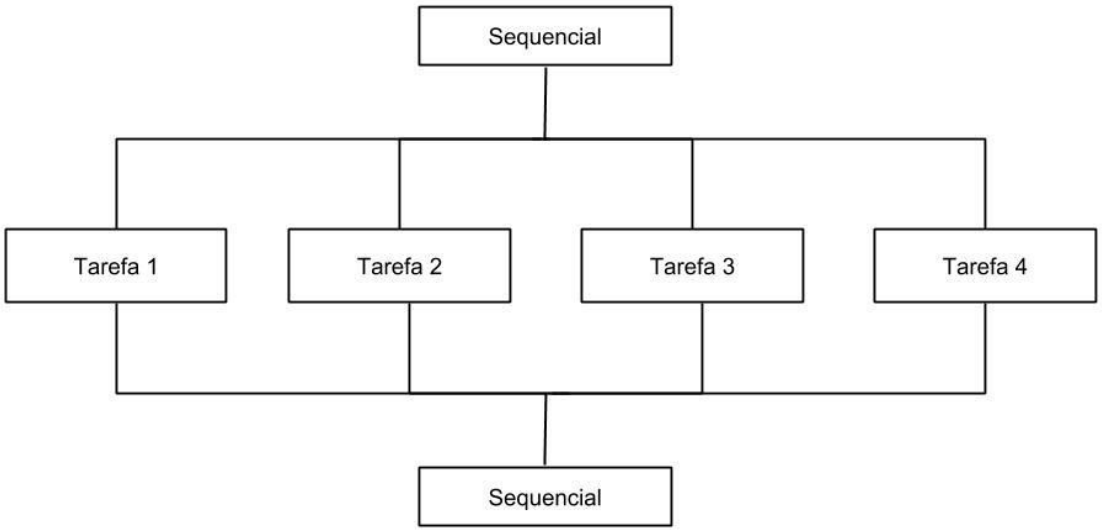
1 INTRODUÇÃO

Atualmente as arquiteturas dos processadores possuem vários núcleos de processamento. O sistema operacional trata cada núcleo como se cada um fosse um processador diferente. Cada um tem seus próprios recursos. Esses processadores multinúcleo permitem a execução de tarefas (ou threads) concorrentes, ou seja, cada aplicação vai estar rodando em um núcleo diferente, tendo melhor aproveitamento do processador.

Os processadores multicore surgiram com a dificuldade de resfriar processadores single core, que possuíam frequências cada vez mais altas. Como possuíam um grande número de transistores, produziam muito calor. Embora os multicores tenham uma frequência menor, o sua utilização se dá pelo fato de trabalharem melhor em um ambiente multitarefas.

A maioria das aplicações não utilizam todos os núcleos de um processador. Geralmente são executadas em apenas um núcleo. Para ter um melhor desempenho, deve-se paralelizar as aplicações, ou seja, o programa deve rodar em dois ou mais núcleos. Abaixo, na Figura 1, segue exemplo da paralelização de um programa:

Figura 1- Programa sequencial sendo dividido em tarefas.

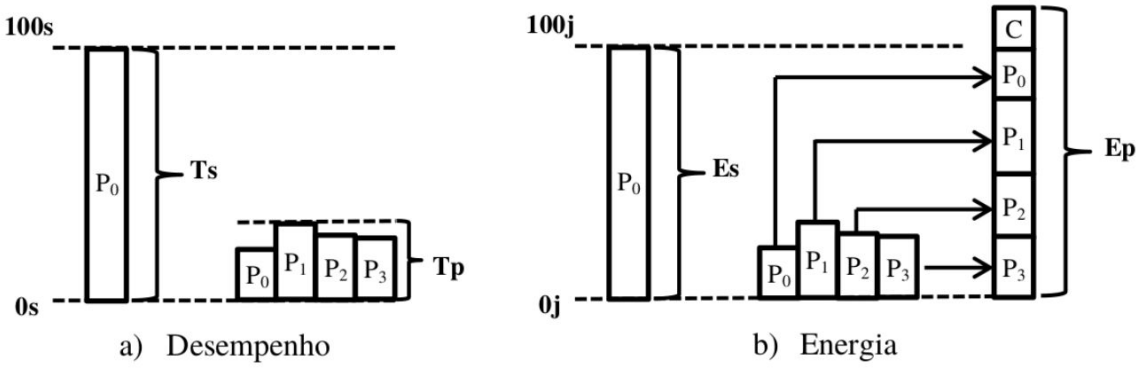


Fonte: do autor, 2018.

A Figura 1, acima, demonstra um programa sequencial sendo paralelizado em determinada parte do programa em quatro tarefas, ou seja, cada tarefa estará sendo executada em um núcleo distinto do processador.

Uma aplicação paralelizada geralmente tem um melhor desempenho em comparação à sua versão sequencial, pois o tempo total de execução é dividido entre núcleos do processador. A Figura 2 abaixo compara a versão sequencial e paralela de um programa.

Figura 2- Versão sequencial e paralela de um programa.



Fonte: Lorenzon, 2014.

A Figura 2a demonstra o desempenho de um programa em sua versão sequencial e paralela. Na versão sequencial a aplicação levou 100 segundos, enquanto a versão paralela levou 30 segundos, que foi o tempo de execução do núcleo P1 do processador. A Tarefa que estará rodando no P1, provavelmente é a que possui maior carga de trabalho, ou seja, é ela que vai determinar o tempo final da aplicação.

A Figura 2b está demonstrando o consumo de energia da aplicação sequencial e paralela. Percebe-se que a versão paralela tem com consumo total de energia (Ep) maior que a sequencial (Es), isso se dá, pelo consumo de energia gasto na comunicação entre os núcleos, representada por (C). Essa comunicação é feita através da memória compartilhada, que geralmente está hierarquicamente mais distante do processador (memória cache L3 e memória principal), que possuem acesso mais lento e maior consumo de energia, quando comparado com as memória privada de cada processador (memória cache L1 e L2) [KORTHIKANTI, AGHA, 2010].

O objetivo desse trabalho é agregar mais uma interface de programação paralela para medir o desempenho e consumo de energia de diferentes arquiteturas multinúcleo em um benchmark desenvolvido pelo estudante do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da Universidade Federal do Pampa (UNIPAMPA) Adriano Marques Garcia, que compara diferentes interfaces de programação paralela.

O benckmark é composto por 13 aplicações, que são utilizadas em seu trabalho. Nesta pesquisa as aplicações estão sendo desenvolvidas utilizando a interface de programação paralela Intel Threading Building Blocks (Intel TBB). Tais aplicações serão comparadas com

as outras interfaces de programação já implementadas, como Pthreads, OpenMP, MPI-1 e MPI-2.

2 METODOLOGIA

Existem várias formas de paralelizar um aplicação. [RAUBER, T, RÜNGER G. 2010] propõem três etapas fundamentais para a paralelização de um aplicação sequencial:

- **Decomposição da Computação:** o programa deve ser decomposto em tarefas suficientes para utilizar todos os núcleos do processador.

- **Atribuição das tarefas para threads:** As tarefas devem ter bom balanceamento de carga. As threads devem ter aproximadamente o mesmo número de cálculos a serem executados.

- **Mapeamento das threads em núcleos de processador:** é realizada pelo sistema operacional, com a finalidade de obter uma utilização equilibrada dos núcleos e mantendo a menor comunicação possível entre os núcleos do processador.

As 13 aplicações foram classificadas em três grupos as que utilizam vetores, matrizes e as que não operam em estruturas de dados. Abaixo segue a tabela com as 13 aplicações.

Tabela 1- Apresentação das aplicações.

Utilizam Vetores	Utilizam Matrizes	Não Utilizam Estruturas de Dados
Ordenação Par-Ímpar; Série Harmônica;	Dijkstra; Método de Jacobi; Multiplicação de Matrizes; Similaridade entre Histograma; Jogo da Vida; Gram-Schmidt; Turing Ring; Transformada Discreta de Fourier;	Cálculo do Pi; Produto Escalar; Integração Numérica;

Fonte: do autor, 2018.

Para paralelizar as aplicações que utilizam estruturas de dados em uma única dimensão (vetores), será utilizada a decomposição de domínio em blocos de uma dimensão. Cada thread fica responsável por computar única e exclusivamente cada bloco. A distribuição dos blocos será definida com estática para granularidades grossas e dinâmica para granularidades finas.

Nos programas que possuem matrizes como estruturas de dados, a decomposição do domínio é feito em blocos de duas dimensões. Apenas o laço for mais externo será paralelizado, ou seja, o laço que controla as linhas também irá controlar as threads. Assim como no caso das aplicações com vetores, na paralelização com Intel TBB será utilizada a distribuição em blocos estáticos para granularidade grossa o dinâmica para granularidade fina.

As aplicações que não computam sobre estruturas de dados possuem implementação bastante simples, com apenas um laço que faz toda a computação. Nesse caso a paralelização irá ser através da divisão do número de iterações pela quantidade de threads de forma estática. No final do laço os resultados de cada thread devem ser unificados.

3 RESULTADOS e DISCUSSÃO

Esse trabalho tem como pretensões colaborar com o trabalho do Adriano Marques Garcia, fornecendo às 13 aplicações utilizando Intel TBB. As mesmas estão sendo desenvolvidas em colaboração com ele. Assim que tiverem funcionando perfeitamente serão testadas e comparadas com as outras interfaces de programação Pthreads, OpenMP, MPI-1 e MPI-2, que já foram implementadas anteriormente.

As estratégias de paralelização foram adotadas levando em contas as estratégias já utilizadas nas implementações das outras interfaces. Essas implementações que estão sendo realizadas levam em conta as características de cada aplicação, ou seja, qual o tipo de estrutura que as aplicações serão computadas.

4 CONSIDERAÇÕES FINAIS

Neste trabalho deseja-se prover a análise da interface de programação paralela Intel Threading Building Blocks, e verificar se ela é uma boa alternativa para medir o desempenho e o consumo de energia de arquiteturas multinúcleos. Para atender esses objetivos, aplicações que estão sendo implementadas serão comparadas com as outras interfaces já usadas. A partir daí pode-se ter uma conclusão se a interface é uma boa alternativa para medir o desempenho e o consumo de energia.

Como trabalhos futuros, pretende-se integrar outras interface além das já utilizadas no trabalho do mestrando Adriano Marques Garcia, para que o trabalho se torne mais abrangente.

REFERÊNCIAS

GARCIA, A. M. Classificação de um Benchmark Paralelo para Arquiteturas Multinúcleo. 2016. Dissertação (Graduação em Ciência da Computação) - Universidade Federal do Pampa, Alegrete, 2016.

KORTHIKANTI, V. A.; AGHA, G. Towards optimizing energy costs of algorithms for shared memory architectures. In: ACM. Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures. [S.l.], 2010. p. 157–165.

LORENZON, A. F. Avaliação do desempenho e consumo energético de diferentes interfaces de programação paralela em sistemas embarcados e de propósito geral. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2014.

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, Springer, v. 80, n. 3, p. 295–307, 2014.

RAUBER, T.; RÜNGER, G. Parallel programming: For multicore and cluster systems. [S.l.]: Springer Science & Business Media, 2010.