Usando as Interfaces de Programação Paralela OpenMP e TBB em um Benchmark

Henrique Fan, Jian Furquim, Claudio Schepke

Centro Tecnológico de Alegrete – Universidade Federal do Pampa (UNIPAMPA)
97.546-550 – Alegrete – RS – Brazil

fanhenrique@.gmail.com,jian_mf@hotmail.com,
claudioschepke@unipampa.edu.br

Abstract. *This paper aims to analyze the parallel programing interfaces OpenMP and Intel Threading Building Block (Intel TBB) in a new benchmark composed by a set of applications. The execution times obtained in five applications were compared using the two interfaces, along with the sequential versions of the algorithms. The paper evaluates the parallelization strategy of each interface. The results obtained showed that in the majority of tests performed OpenMP had a better result than TBB. The TBB interface obtained results very close to those of OpenMP in most applications.*

Resumo. *Este artigo tem como objetivo analisar as interfaces de programação paralela OpenMP e Intel Threading Building Blocks (Intel TBB) em um novo benchmark composto por um conjunto de aplicações. Foram comparados os tempos de execução obtidos em cinco aplicações, utilizando as duas interfaces, junto com as versões sequenciais dos algoritmos. O artigo avalia a estratégia de paralelização de cada interface. Os resultados obtidos mostraram que em maioria dos teste realizados OpenMP teve um resultado melhor que TBB. A interface TBB obteve na maioria das aplicações resultados muito próximos aos de OpenMP.*

1. Introdução

Um novo benchmark está em processo de criação para validar diferentes interfaces de programação paralela considerando aspectos de desempenho e consumo energético GARCIA, A. M. (2016). Atualmente este benchmark já possui 4 interfaces de programação sendo usados em todas as 13 aplicações que cobrem diferentes tipos de problemas. Atualmente começamos a transcrever o código das aplicações para TBB, a fim de se ter mais uma interface de programação a disposição dos usuários. Neste trabalho apresentamos a implementação de apenas 5 aplicações e alguns resultados preliminares em termos de desempenho. As 5 aplicações são apresentadas a seguir:

Cálculo do PI: O Pi é um valor irracional que estabelece uma relação numérica entre o perímetro de uma circunferência e seu diâmetro, e.g. [Roy 1990]. Nesta aplicação, foi utilizado o método de Gregory-Leibniz, e.g. [Andrews e Roy 1999]. A equação de Gregory-Leibniz fica num laço. No final do laço temos um valor aproximado de Pi, quanto mais iterações o laço tiver mais preciso é o valor.

Integração Numérica: A integral de uma função serve para determinar a área sob uma curva no plano cartesiano através da aproximação, e.g. [Stewart 2001]. O método de aproximação envolvido em nossa aplicação é de quadratura numérica.

Produto Escalar: O produto escalar é uma operação entre dois vetor que tem como resultado um escalar, e.g. [Callioli, Domingues e Costa 2007]. No algoritmo o produto escalar é calculado entre uma sequência de valores ordenados e outra de ordenação inversa.

Ordenação Par-Ímpar: A ordenação par-ímpar ordena um vetor através da comparação dos pares de índices (Par-Ímpar) dos elementos. Se um par estiver na ordem errada, ou seja, o primeiro é maior que o segundo, os elementos são trocados. Repete-se isso até que todos os elementos estejam ordenados.

Multiplicação de Matrizes: Existem muitas formas para realizar a multiplicação de matrizes. O método usado consiste da multiplicação dos elementos das linhas da matriz A pelos elementos das colunas da matriz B.

2. Paralelização das Aplicações

As aplicações do Cálculo do Pi, Integração Numérica e Produto Escalar possuem uma implementação bastante simples não operando sobre estruturas de dados. A computação principal é feita em apenas um laço. Na paralelização a quantidade de iterações do laço é dividida entre o número de threads de forma estática, ou seja, cada threads terá aproximadamente o mesmo quantidade de iterações para computar. Após a conclusão do laço, foi usado a operação de redução para que os resultados parciais computados separadamente por cada thread sejam unificados no resultado final.

A aplicação Ordenação Par-Ímpar opera em um vetor. Para paralelizar essa aplicação, cada thread computa sobre diferentes elementos do vetor, que é dividido em blocos de iterações. Novamente, como nas aplicações que não operam sobre estrutura de dados, as iterações são divididas de forma estática.

Na aplicação Multiplicação de Matrizes a decomposição dos blocos é em duas dimensões. A paralelização é feita apenas no laço mais externo, ou seja, no laço que controla as linhas da matriz resultante. Assim como nos casos anteriores a divisão é feita de forma estática.

3. Resultados

Para validar a paralelização e comparar as interfaces de programação, cada aplicação foi executada em suas versões sequencial e paralelizadas, em OpenMP e Intel TBB, variando o número de threads em 4, 8, 16 e 32. Foram feitas 20 execuções para cada uma das versões e número de threads, com 3 entradas distintas. Assim obteve-se a média de tempo de execução em segundos conforme descritas na Tabela 1, Tabela 2. Os melhores resultados estão em negrito. A máquina usada na execução dos algoritmos possui um processador Intel Xeon E5-2650 a 2,0 GHz de 32 núcleos e 126 GB de memória RAM. Com as respectivos tamanhos nas caches L1, L2 e L3, sendo 32K, 256K e 20480K.

Tabela 1. Médias das aplicações Cálculo do Pi, Integração Numérica e Produto Escalar.

	Cálculo do Pi			Integração Numérica			Produto Escalar		
Iterações	**4 bilhões**	**8 bilhões**	**12 bilhões**	**2 bilhões**	**4 bilhões**	**8 bilhões**	**5 bilhões**	**15 bilhões**	**25 bilhões**
Sequencial	63,29	126,36	189,49	31,59	63,17	126,39	14,92	44,70	74,66
OpenMP 4 Threads	**15,93**	**31,82**	**47,73**	**8,08**	**16,09**	**32,27**	**3,63**	**10,93**	**18,07**
OpenMP 8 Threads	**8,86**	**17,73**	**26,68**	**4,48**	**8,97**	**18,02**	**2,02**	**6,13**	**10,07**
OpenMP 16 Threads	**4,70**	**9,31**	**13,90**	**2,39**	**4,81**	**9,37**	**1,09**	**3,22**	**5,30**
OpenMP 32 Threads	**4,65**	**9,27**	**13,88**	**2,35**	**4,65**	**9,27**	**0,88**	**2,55**	**4,20**
TBB 4 Threads	16,01	31,98	47,95	8,95	17,51	34,15	5,91	17,67	29,46
TBB 8 Threads	8,92	17,83	26,71	5,07	10,08	19,59	3,30	9,84	16,41
TBB 16 Threads	5,01	9,59	14,29	3,00	5,55	10,70	2,08	5,46	8,95
TBB 32 Threads	4,67	9,28	13,91	2,36	4,68	9,31	1,56	4,59	7,64

Tabela 2. Médias das aplicações Ordenação Par-Ímpar e Multiplicação de Matrizes.

	Ordenação Par-Ímpar			Multiplicação de Matrizes		
Iterações	**100000**	**150000**	**250000**	**512x512**	**1024x1024**	**2048x2048**
Sequencial	30,61	68,83	191,26	1,42	12,91	162,79
OpenMP 4 Threads	**9,33**	**20,93**	**58,13**	0,37	3,40	**34,98**
OpenMP 8 Threads	**5,25**	**11,84**	**32,44**	**0,21**	**1,85**	**18,87**
OpenMP 16 Threads	**2,96**	**6,33**	**17,24**	**0,12**	**0,98**	**10,39**
OpenMP 32 Threads	**2,76**	**6,01**	**16,34**	**0,11**	0,87	8,72
TBB 4 Threads	12,50	27,69	77,14	**0,36**	**3,30**	36,07
TBB 8 Threads	7,49	16,23	44,01	0,22	1,86	19,90
TBB 16 Threads	5,94	10,56	49,87	0,17	1,17	10,51
TBB 32 Threads	5,34	10,45	32,04	0,12	**0,86**	**8,18**

Os resultados esperados eram que os tempos de execuções decaísse com o aumento do número de threads, o que ocorreu em todas as aplicações, menos no teste com 16 threads em TBB na Ordenação Par-Ímpar. O teste teve como resultado 49,87 segundos, sendo que com 8 threads alcançamos um resultado menor de 44,01 segundos.

Em relação aos resultados obtidos, pode se perceber que OpenMP teve valores

médios de tempo de execuções menores em quase todos os testes. Apenas na aplicação da Multiplicação de Matrizes que TBB obteve resultados menores em quatro ocasiões, sendo com 4 threads com matrizes de 512x512 e 1024x1024 e 32 threads com matrizes de 1024x1024 e 2048x2048.

Embora OpenMP tenha alcançado um desempenho superior. Os resultados alcançados usando a interface TBB ficaram bem próximos de OpenMP nas aplicações do Cálculo do PI, Integração Numérica e Multiplicação de Matrizes. Já nos programas do Produto Escalar e da Ordenação Par-Ímpar TBB teve uma queda de desempenho.

Podemos perceber que o ganho de desempenho de 16 para 32 threads, tanto com OpenMP e TBB, foi muito pouco, praticamente nulo. Isso ocorre, pois o processador usado nos teste possui apenas 16 threads. Este pequeno ganho de desempenho se dá pelo fato do processador possuir Hyper-Threading, possibilitando dois segmentos de processamento por núcleo.

4. Conclusão e Trabalhos Futuros

Neste trabalho foram apresentados os resultados da comparação das interfaces de programação paralela OpenMP e TBB em algumas aplicações. Com base nos resultados, podemos observar que OpenMP alcançou um desempenho superior de forma geral que TBB. Mesmo assim TBB ficou com resultados muito próximos em três das cinco aplicações. Como trabalhos futuros pretende-se validar todas as aplicações da biblioteca. Para que possamos verificar se a nova interface TBB possa ser incluída no benchmark.

5. Referências

ROY, R. (1990) “The Discovery of the Series Formula for PI by Leibniz, Gregory and Nilakantha”. Mathematics Magazine, Mathematical Association of America.

STEWART, J. (2001) “Cálculo”, vol. 1. Pioneira Thomson Learning.

CALLIOLI, C. A.; DOMINGUES, H. H.; COSTA, R. C. F. (2007) “Álgebra linear e aplicações”.

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

GARCIA, A. M.; SCHEPKE, C.; GIRARDI, A. G.; SILVA, S. A. (2018) “A New Parallel Benchmark for Performance Evaluation and Energy Consumption”. 13th International Meeting on High Performance Computing for Computational Science (VECPAR), São Pedro.

GARCIA, A. M.; SCHEPKE, C.; GIRARDI, A. G.; SILVA, S. A. (2018) “Power Consumption of Parallel Programing Interfaces in Multicore Architectures”. Simpósio de Sistemas Computacionais de Alto Desempenho (WSCAD), São Paulo.