Solução do Problema de Difusão de Calor em Processadores Multicore

Marcelo André Pedrazzani, Guilherme Galante

[1]Colegiado de Informática, UNIOESTE
Rua Universitária 2069, Cascavel-PR

{marcelopedrazzani,gui.galante}@gmail.com

1. Introdução

Com a crescente demanda por poder de processamento e com o limite físico na fabricação de transistores em camadas de silício cada vez menos espessas, busca-se novas tecnologias para a fabricação de circuitos integrados e microprocessadores. Uma das alternativas para suprir a demanda por processamento é o uso de arquiteturas paralelas, sendo um dos seus representantes as arquiteturas que empregam a tecnologia *multicore*.

A tecnologia *multicore* (múltiplos núcleos) consiste em dois ou mais processadores no interior de um único chip. O sistema operacional trata esses núcleos como se cada um fosse um processador diferente, com seus próprios recursos de execução. Isto possibilita que aplicações sejam designadas para núcleos diferentes, aumentando seu desempenho e proporcionando um ambiente multitarefa mais eficiente. Para que isso aconteça, a aplicação deve ser desenvolvida de modo a explorar esse paralelismo eficientemente [BINSTOCK 2006].

Para explorar o paralelismo em processadores *multicore*, algumas bibliotecas como OpenMP e Pthreads podem ser utilizadas. Assim o objetivo do trabalho é comparar a solução do problema da difusão de calor empregando estas duas bibliotecas.

2. Resolução da Equação da Difusão de Calor

Todo e qualquer problema teórico que se deseja modelar, ou seja, demonstrar como as grandezas físicas atuam e afetam o sistema físico das quais foram extraídas, deve seguir um conjunto de etapas, as quais permitem extrair modelos que possam ser representados computacionalmente. A Figura 1 apresenta um fluxograma, adaptado de [BORGES and PADOIN 2006], demonstrando as principais etapas da modelagem matemática.

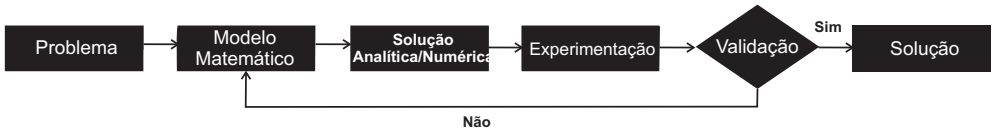

Figura 1. Etapas da Modelagem Matemática

O método de solução da equação de difusão de calor, proposto neste trabalho, é composto por duas etapas:

1. Montagem do sistema de equações;
2. Resolução do sistema de equações através do Gradiente Conjugado (GC).

A montagem do sistema de equações se da através do método das diferenças finitas. Para o resolver o sistema foi utilizado o método numérico Gradiente Conjugado. Pelo fato das operações que formam o método do GC serem de caráter vetorial e matricial, a estratégia adotada nas implementações da solução será a decomposição por dados, que permite segmentar uma estrutura de dados em diversos blocos e distribuí-la entre as unidades de execução, para serem processadas simultaneamente.

2.1. Pthreads

Na paralelização utilizando Pthreads em cada chamada de função de cálculo matricial e/ou vetorial cria-se um conjunto de threads. Cada thread executa um segmento da estrutura de dados; assim, não há dependência de dados entre as threads, com exceção no cálculo do produto escalar, onde há a necessidade de acesso a uma mesma variável por todas as threads, e sua consistência é garantida utilizando técnicas de controle de seção crítica.

2.2. OpenMP

A API OpenMP explora o paralelismo em máquinas com memória compartilhada, tendo como principais componentes as diretivas de compilação (*pragmas*), cláusulas, bibliotecas de serviços e variáveis de ambiente. Da mesma forma utilizada na implementação com Pthreads, a paralelização com o OpenMP foi realizada nas operações matriciais e/ou vetoriais.

3. Resultados

Na Tabela 1 são apresentadas quatro máquinas onde os testes foram realizados, os quais foram desenvolvidos em linguagem C e compilados com o gcc-4.3, utilizando sistema operacional Debian Linux.

Tabela 1. Máquinas utilizadas

Nome	Processador	Cache	Núcleos
caporal	AMD Opteron(tm) Processor (242) 1,60GHz	1024 KB	2
cohiba	Intel(R) Xeon(TM) CPU 3,60GHz	2048 KB	2
macalan	Dual-Core AMD Opteron(tm) 2220 2,80GHz	1024 KB	4
dalmore	Intel(R) Xeon(R) CPU E5345 2,33GHz	4096 KB	8

Nas Figuras 2, 3 e 4 apresenta-se os tempos de execução obtidos nas execuções do método utilizando as três bibliotecas. Utilizou-se domínios de 10x10 células, 50x50 células e 80x80 células.

Pode-se notar que as soluções paralelas no domínio 10x10 consumiram mais tempo que a implementação seqüencial; isto acontece devido à sobrecarga de criação, destruição e sincronização das threads, em OpenMP e Pthreads. Como matrizes com dimensões pequenas têm baixo custo de processamento, o tempo gasto com o gerenciamento das threads/processos não é compensado.

Entretanto, com o aumento do domínio amostrado, há um maior custo de processamento. Pode-se notar que as implementações paralelas foram mais rápidas que a implementação seqüencial. Como o tempo de gerenciamento, utilizando ferramentas de

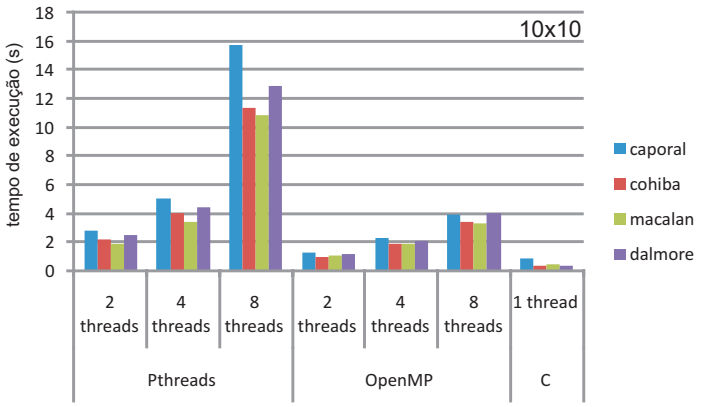

Figura 2. Tempo de execução para domínio 10x10

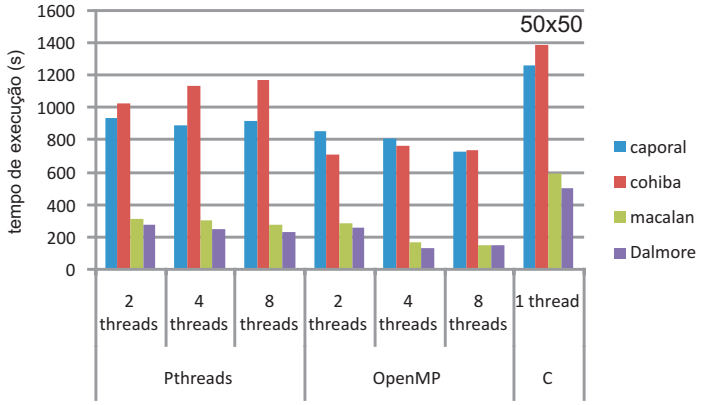

Figura 3. Tempo de execução para domínio 50x50

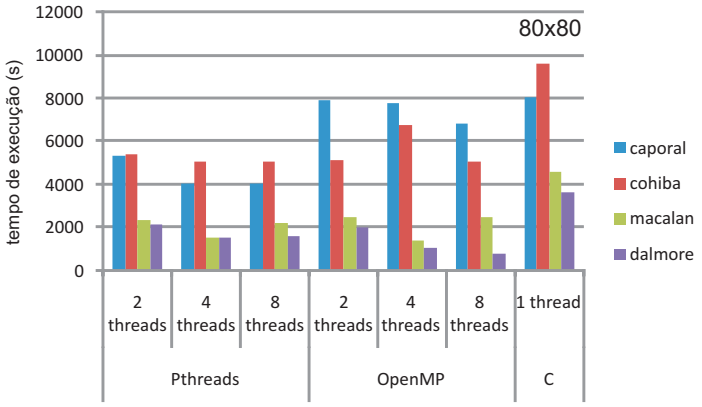

Figura 4. Tempo de execução para domínio 80x80

memória compartilhada, são relativamente constantes, quanto maior o domínio a ser calculado melhores ganhos de desempenho poderão ser obtidos.

O resultado de uma simulação usando um domínio 25x25 é mostrado na Figura 5.

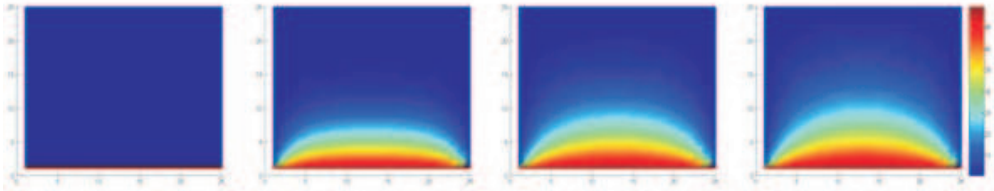

Figura 5. Amostras (0, 500, 1000 e 1500 segundos) da evolução da solução da difusão de calor, domínio 25x25 com tempo de simulação de 1500 segundos

4. Conclusões

O uso de múltiplos núcleos já é uma tendência na construção de processadores. No entanto para aproveitar todo o potencial que esta arquitetura oferece, deve-se utilizar as ferramentas apropriadas para a construção do software.

Nestre trabalho as implementações utilizando OpenMP e Pthreads apresentaram desempenhos satisfatórios, porém observa-se que sua eficiência cresceu de acordo com o tamanho do domínio, pois o custo de criar, sincronizar e destruir as threads não são compensados pelo baixo custo computacional na solução de pequenos domínios.

5. Agradecimentos

Agradecimentos especiais à UFPR por ter cedido as máquinas para a execução dos testes deste trabalho.

Referências

BINSTOCK, A. (2006). Multi-core processor architecture explained. *Intel Software Network*.

BORGES, P. A. P. and PADOIN, E. L. (2006). Exemplos de métodos computacionais aplicados a problemas na modelagem matemática. In *Anais ERAD*.