Análise do algoritmo correlation paralelo utilizando UPC

Felipe Diniz Tomás - RA:110752

***Abstract*—O presente estudo realiza uma análise do algoritmo *correlation* presente na coleção *PolyBench*, paralelizado utilizando UPC, com o objetivo de analisar e mensurar a diferença entre os modelos udp, smp e mpi. Dessa forma, foi coletado o tempo de execução de determinados testes, buscando calcular o *speedup*[1] de cada caso. Os resultados não mostraram diferenças significativas entre os modelos de API, performando de forma similar entre elas. O trabalho permitiu entender melhor quanto a utilização do UPC, seu funcionamento e sua aplicação.**

I. Introdução

Unified Parallel C (UPC) é uma extensão da linguagem de programação C projetada para computação de alto desempenho em máquinas paralelas de grande escala, incluindo aquelas com um espaço de endereço global comum (SMP e NUMA) e aquelas com memória distribuída (por exemplo. clusters).

O ambiente consiste de um único espaço de endereço particionado compartilhado, onde as variáveis podem ser lidas e escritas diretamente por qualquer processador, mas cada variável está fisicamente associada a um único processador.

UPC usa um único programa, modelo de dados múltiplos (SPMD) de computação no qual a quantidade de paralelismo é fixada no tempo de inicialização do programa, normalmente com um único thread de execução por processador.

Dessa forma, a UPC permite gerar modelo de códigos udp, smp e mpi, logo o objetivo deste trabalho é gerar o código de cada modelo e compará-los conforme o desempenho.

II. Descrição do problema

A correlação calcula os coeficientes de correlação, que nada mais é que a covariância normalizada. Leva o seguinte como entrada:

- N×M matriz que representa N pontos de dados, cada um com M atributos.

e dá a seguinte saída:

- $M \times M$ matriz onde i, j-ésimo elemento é o coeficiente da correlação entre i e j. A matriz é simétrica.

III. Metodologia

As execuções realizadas foram feitas no seguinte ambiente computacional:

- Processador Intel® Core™ i5-4210U
  - Número de núcleos: 2;
  - Número de threads: 4;
  - Frequência baseada em processador: 1.70 GHz;
  - Frequência turbo max[2]: 2.70 GHz;
  - Cache: 3 MB Intel® Smart Cache[3]
    - Tamanho da memória cache:
      - L1 - 128 KB;
      - L2 - 512 KB;
      - L3 - 3 MB.
- 8 Gb de memória RAM DDR3
- Sistema Operacional: Ubuntu, sendo ele:
  - Ubuntu 18.04 LTS;
  - Kernel: 4.15.0.43;
  - Compilador: gcc 9.3.0.

A. Casos de testes

Foi utilizada o mesmo tamanho de matriz para todos os modelos, udp, smp e mpi. Uma matriz de 2800×2800. Para cada teste foi realizado um total de 6 execuções, sendo a primeira delas descartada para evitar *compulsory miss*[4], e calculado a média de tempo das demais.

IV. Resultados

O tempo do código sequencial foi de 451 segundos para a matriz de 2800×2800.

Os resultados de tempo do código paralelo, considerando os modelos pedidos, foram:

TABLE I: Tempo em segundos do código paralelo.

Threads	smp	udp	mpi
1	448s	415s	435s
2	309s	270s	330s
3	282s	246s	227s
4	231s	221s	218s

Com o tempo de execução de cada base de dados, tanto do código paralelo como sequencial, é aplicado a fórmula abaixo para o cálculo de *Speedup*.

$$S(t) = \frac{Tempo\ de\ execução\ Sequencial}{Tempo\ de\ execução\ com\ t\ Threads}$$

A figura 1 a seguir indica o *Speedup* alcançado para cada base de dados executadas pelo algoritmo UPC paralelo.

[1]O *speedup* é um valor que mede a melhoria na velocidade de execução de um processo executado em duas arquiteturas semelhantes com recursos distintos.

[2]Frequência turbo máxima é a frequência máxima de núcleo único, à qual o processador pode funcionar, usando a Tecnologia Intel® Turbo Boost.

[3]Cache inteligente refere-se à arquitetura que permite que todos os núcleos compartilhem dinamicamente o acesso ao cache de último nível.

[4]O *compulsory miss* é uma falha que ocorre quando um bloco é trazido pela primeira vez a memória cache.

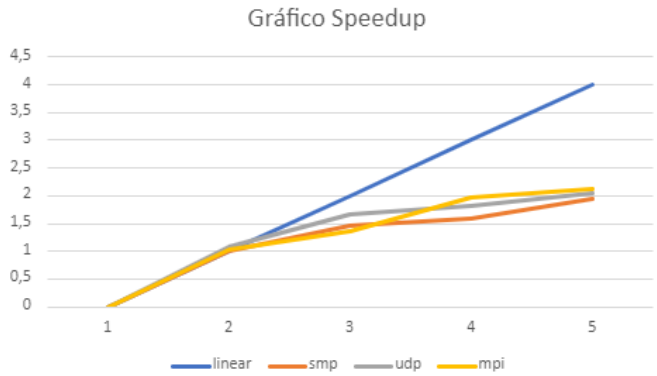

Fig. 1: Gráfico de *Speedup* do algoritmo correlation paralelo.

TABLE II: Valores da métrica de Speedup.

Threads	smp	udp	mpi
1	1,00	1,08	1,03
2	1,45	1,66	1,36
3	1,59	1,82	1,97
4	1,94	2,03	2,13

Como podemos observar o desempenho não chegou ao ideal em nenhum dos modelos após passar de duas threads, lembrando que o ambiente de testes tem apenas dois núcleos físicos para processamento. A diferença entre os modelos não foram grandes, na verdade se comportaram de forma similar quanto ao desempenho.

Segundo a documentação, o MPI pode ser mais lento do que usar um dos outros tipos de API, já o UDP pode performar melhor em sistemas com comunicação via Ethernet, principalmente comparado ao MPI.

Upc permite a implementação em um único espaço de endereço particionado e compartilhado, como já dito anteriormente, as variáveis são lidas e escritas por qualquer processador, porém cada variável está fisicamente associada a um único processador. As primitivas de destaque usadas na implementação, foram o tipo shared, cujo o elemento final é qualificado por compartilhamento entre as threads, e o upc forall que como definido pela documentação, é uma operação coletiva em que, para cada execução do corpo do loop, a expressão de controle e a expressão de afinidade são de valor único.

Além disso a linguagem também possui primitivas de alocação de memória, que se bem sucedidas, retornam um ponteiro para shared.Também possui primitivas de de sincronização quando necessário, como barreira e lock, além de outras ferremantas para implementação de programas paralelos [1]

A utilização de UPC mostra-se versátil, a documentação e versões são bem atualizadas e completas, porém é difícil encontrar publicações pela comunidade, principalmente em português, em comparação a outras alternativas de programação paralela vistas no curso. Provavelmente é usado por instalações de computação de alto desempenho, além de laboratórios de pesquisa, portanto é interessante conhecer e se integrar que a linguagem é uma opção de uso.

V. Conclusão

É muito importante se integrar e conhecer os métodos que permitem a implementação da programação paralela, o upc é uma dessas ferramentas que se mostra interessante para este tipo de tarefa. A capacidade de permitir diferentes APIs aplicando modelos distintos, proporciona um versatilidade interessante quanto a sua usabilidade. Em suma, o trabalho permitiu entender melhor quanto a utilização do UPC, seu funcionamento e sua aplicação.

Quanto ao código, necessita de um estudo mais aprofundado acerca da linguagem, já que em suma a implementação não atingiu o ideal de desempenho e é básica quando aos recursos da linguagem.

Para trabalhos futuros seria interessante pesquisa mais extensa sobre a utilização da linguagem e seus recursos, além de uma aplicação prática ao código buscando otimizá-lo, realizando testes em ambientes mais complexos interconectados para que sejam testados os recursos de comunicação entre processadores.

References

[1] U. Consortium, D. Bonachea, and G. Funck, “Upc language and library specifications, version 1.3,”