Paradigmas de Computação Paralela

Relatório de Desenvolvimento

24 de Novembro de 2019

Conteúdo

0.1 Introdução

Este trabalho, realizado no âmbito da Unidade Curricular de Paradigmas da Computação Paralela, tem como principal objetivo avaliar a aprendizagem do paradigma de programação baseado em memória partilhada (OpenMP).

O algoritmo a ser estudado em concreto é o da multiplicação de matrizes esparsas por vetores, matrizes essas que se diferem das restantes pelo facto de possuírem uma grande quantidade de elementos cujo valor é nulo. Estas matrizes possuem aplicações em várias áreas como a Física (por exemplo, o método das malhas para resolução de circuitos elétricos) e a Engenharia (por exemplo, em Machine Learning).

Nas multiplicações, os valores do vetor são multiplicados por valores da linha da matriz e o resultado da soma dessas multiplicações é colocado numa posição de um vetor resultado. Isto faz com que haja inúmeras repetições do mesmo cálculo, sendo portanto útil a utilização da paralelização.

0.2 Descrição do Problema

0.2.1 Representação em formato COO

O caso de estudo, em particular, tem como base matrizes quadradas (N x N) e, como referido anteriormente, matrizes esparsas são caracterizadas pela grande quantidade de valores nulos que possuem,tendo surgido várias alternativas para a sua representação, sendo uma delas o Formato COO (Coordinate List Format), a qual será também aqui usada.

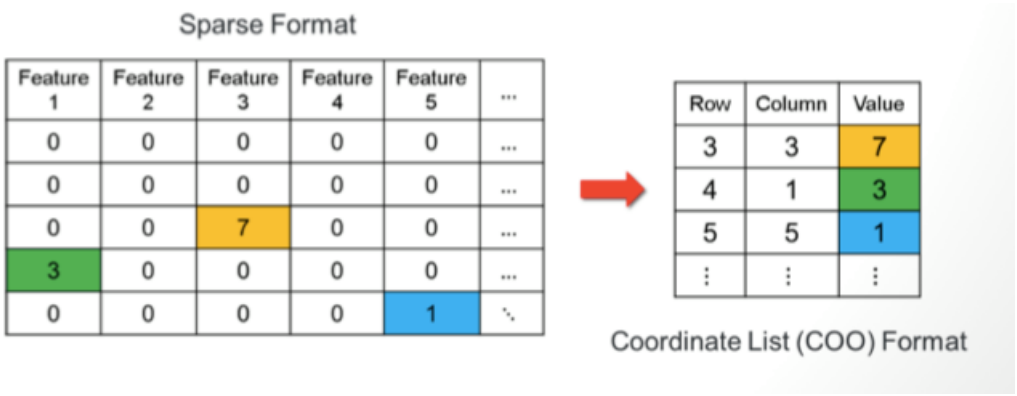

Figura 1: Representação em COO de matriz esparsa

Neste tipo de representação são utilizados 3 vetores: val, col e row, em que para cada posição i, o elemento value[i] corresponde ao elemento da posição (row[i], column[i]), sendo descartados os elementos cujo valor é nulo.Ao invés de 3 vetores, o grupo optou pela utilização de uma matriz 3 x tam (sparsa) , em que tam representa o número de elementos diferentes de zero na matriz original e 3 porque são três linhas na representação em COO, determinadas pela matriz original: na posição 0 é a linha, na posição 1 a coluna e na posição 2 os valores diferentes de zero dessa linha e coluna.

0.2.2 Algoritmo Sequencial

Assim, após todas as modificações efetuadas o algoritmo para a multiplicação é a seguinte:

```
for (unsigned i = 0; i < t; i++){
    resultado[sparsa[0][i]] += x[sparsa[1][i]] * sparsa[2][i]; //Hotspot
}
```

Da análise deste algoritmo pode verificar-se que apesar de fazer o que é pretendido, existem certos aspetos que se devem ter em atenção aquando da paralelização do mesmo.

Primeiramente, num ambiente de memória partilhada, caso não haja nenhum tipo de sincronização, como é o caso, várias threads podem tentar aceder,simultanemanete, à mesma posição de memória sendo pelo menos um dos acessos para escrita, fenómeno designado por **Data Race**. Este fenómeno leva a que hajam possíveis erros de computação no resultado final,já que a ordem de acesso à memória é não-determinística, levando a uma degradação da performance.

Para além disso, o processador ao requerer acesso a uma posição de memória irá copiar um bloco inteiro para a cache, com o propósito de tirar partido da localidade espacial nos acessos, reduzindo assim o número de vezes que é necessário ir buscar dados à memória. Em problemas resolvidos com recurso a paralelismo podemos ter duas threads a tentar escrever em valores que se encontram na mesma linha de cache invalidando toda a linha de cache, obrigando uma das threads a pedir novamente os dados à memória, fenómeno que é conhecido como **False Sharing**.

0.3 Implementação do algoritmo paralelo

Após a identificação do **hotspot**, foi feita uma primeira tentativa de paralelização do algoritmo, tendo em conta as adversidades referidas anteriormente. Para isso foi utilizada a diretiva *critical* do OpenMP que atribui um lock específico a cada thread, fazendo com que apenas uma thread efetue o código identificado pelo hotspot. Apesar de resolver os problemas, esta solução leva a uma degradação da performance, no sentido em que se torna ainda mais lento que a versão sequencial, devido à serealização que é feita bem como ao overhead inicial na criação das threads.

Para resolver este problema, criou-se um array auxiliar para armazenar os resultados de cada thread, o que irá possibilitar a escrita sem desordem dos seus cálculos.

Depois de efetuado o trabalho de cada uma, é feita a redução dos vários arrays auxiliares no array resultado final. É usada a diretiva **atomic** para resolver a concorrência entre os acessos pois a mesma permite que apenas uma thread de cada vez adicione os seus resultados dos cálculos no array resultado. A utilização desta, ao invés da **critical** deve-se ao facto da primeira utlizar recursos do hardware para ser mais eficiente.

0.4 Demonstração e Análise de Resultados

Após o desenvolvimento da solução para o problema, foram efetuados vários testes para perceber melhor o comportamento do algoritmo, em termos de melhoria de performance em relação à versão sequencial. Foram efetuados testes em várias máquinas , porém os resultados apresentados são do nodo 662, com 24 cores, 48 com Hyperthreading, devido ao facto de ser o único que apresenta frequência fixa, levando a resultados mais viáveis.

0.4.1 Apresentação de Resultados

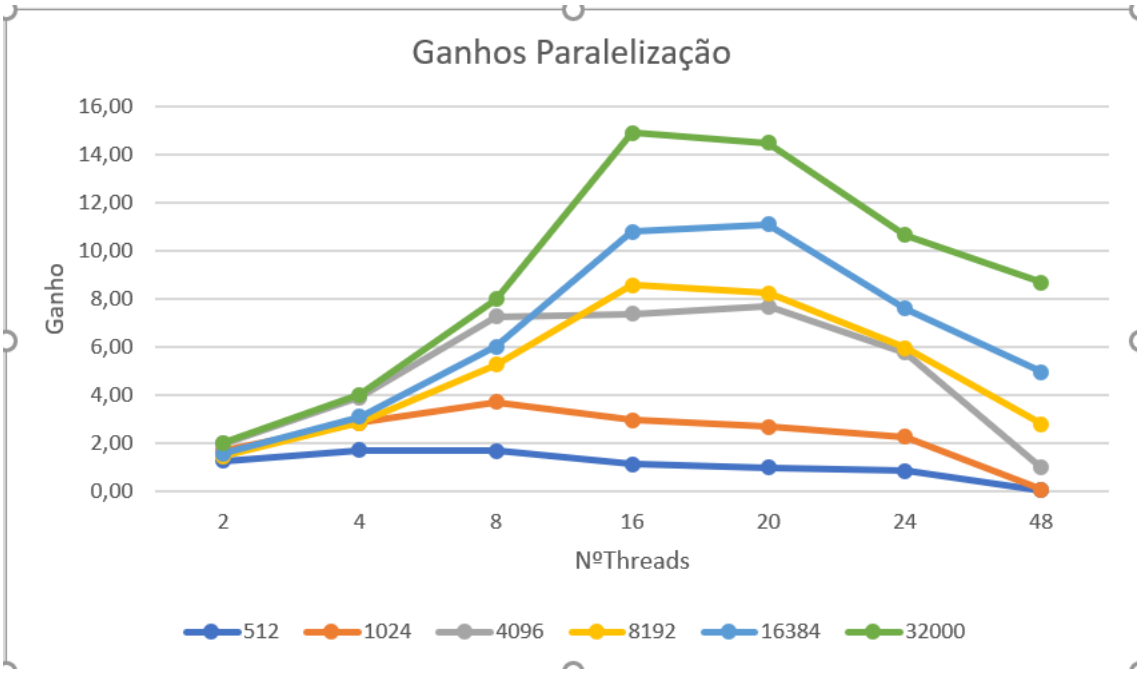

Figura 2: Ganho para diferentes tamanhos de matriz

Para a realização dos testes levou-se em consideração alguns aspetos:

- A Cache foi limpa entre cada multiplicação.
- A dispersão da matriz é de aproximadamente 60%
- Para cada tamanho da matriz e cada thread foram realizados 10 testes, sendo o resultado usado a mediana dos mesmos.
- O tempo utilizado foi em mili segundos.

0.4.2 Análise de Resultados

Para começar tem que se ter em conta que existem certos aspetos que dificultam a escalabilidade deste algoritmo,independentemente do tamanho do input. Primeiramente,existe o overhead associado à criação de threads, sendo este mais notável em inputs pequenos. Para além disso , apesar de a diretiva atomic ser mais eficiente que a critical, a serealização continua a existir, o que degrada a performance. Por fim, também a sincronização das threads no final da zona paralela tem o seu impacto na performance do programa.

Matriz com tamanho 512 e 1024

Para os tamanhos 512 e 1024 podemos observar pelo gráfico que não é possível obter um ganho significativo, embora se note uma ligeira melhoria no mesmo de uma matriz com tamanho 512 para 1024. À medida que se aumenta o número de cores existe perda da performance. Isto acontece devido ao overhead que o array auxiliar provoca, assim como a respetiva redução dos vários arrays no array resultado.

Matriz com tamanho 4096

Para tamanho 4096 da matriz verificou-se melhorias nos tempos de execução até aos 20 cores.Apesar do ganho obtido, este não é proporcional ao número de cores o que se traduz na degradação da performance em cache e respetiva redução final no array resultado.

Matriz com tamanho 8192

A matriz de tamanho 8192 apresenta uma curva de ganho semelhante a 4096 em termos de formato na qual até 16 cores existem melhorias nos tempos obtidos. A melhoria em relação a 4096 é mínima e tal deve-se a uma melhor utilização do array auxiliar a cada thread que compensa o overhead que este cria pois possui muitos mais dados para analisar. A partir dos 16 cores, como são precisos grande quantidade dados da memória, observa-se uma perda da performance.

Matriz com tamanho 16384 e 32000

Tanto matrizes de tamanho 16384 como 32000 possuem um comportamento parecido.Observa-se um ganho máximo de 11.1 e 14.9 para 20 cores e 16 cores respetivamente. Verifica-se também que a partir desse número de cores a performance sofre degradação com o aumento dos mesmos. Concluímos que o elevado número de acessos à memória faz com que se atinja o máximo de performance nesses cores. No entanto,especula-se que a partir desse ponto, os cálculos estão dependentes de idas à memória, o que afeta a performance resultante.

0.5 Conclusão

Com a realização deste trabalho aprofundamos os nossos conhecimentos sobre computação paralela em memória partilhada recorrendo à ferramenta *OpenMP*.

O grupo verificou o trabalho implicado na produção de um algoritmo paralelo assim como os fatores a ter em conta para a não escalabilidade do mesmo, como é o caso. Concluimos que apesar de ser possível identificar alguns dos fatores,existem outros de difícil explicação e que podem ser de uma grande variedade de causas.

Em suma, ficámos satisfeitos com o conhecimento e experiência adquiridos nesta área, que sem dúvida irão ser úteis em trabalhos futuros.

0.6 Autores

- Adriana Meireles a82582
- Shazhod Yusupov a82617

Capítulo 1

Anexos

N	Medições/NºThreads	Sequential:	2	4	8	16	20	24	48
	1	0,671	0,536	0,39	0,39	0,6	0,691	0.807	45,717
	2	0,666	0,563	0,398	0,407	0,61	0,728	0,821	47,039
	3	0,704	0,528	0,411	0,39	0,608	0,694	0,811	41,723
	4	0,666	0,532	0,398	0,436	0,599	0,7	0,808	49,794
512	5	0,7	0,532	0,41	0,409	1,087	0,7	0,817	46,343
	6	0,672	0,542	0,43	0,446	0,61	0,717	0,816	32,68
	7	0,702	0,539	0,382	0,407	0,598	0,707	0,8	44,858
	8	0,671	0,527	0,394	0,409	0,578	0,731	0,805	44,847
	9	0,67	0,537	0,396	0,406	0,605	0,728	0,8	41,348
	10	0,671	0,548	0,381	0,369	0,593	0,704	0,815	46,78
	Mediana	0,671	0,5365	0,398	0,408	0,6065	0,712	0,811	45,2875
	Ganho	1	1,25	1,69	1,649	1.113	0.951	0.827	0.015

N	Medições/NºThreads	Sequential:	2	4	8	16	20	24	48
	1	2,6513	1,56959	0,968203	0,764472	0,910174	0,996009	1,2261	43,8523
	2	2,8084	1,55636	0,936337	0,721335	0,91461	0,993448	1,10996	44,274
	3	2,6485	1,56449	0,949726	0,702588	0,887808	1,04682	1,15537	45,5682
	4	2,6503	1,56551	0,890147	0,721284	0,882784	0,995638	1,19774	48,2324
1024	5	2,6657	1,5636	0,979527	0,810804	0,892864	1,02206	1,17437	47,0752
	6	2,6573	1,57131	0,938876	0,717587	0,936189	0,964447	1,18139	48,3821
	7	2,807	1,56148	0,918778	0,726393	0,900873	1,00277	1,18715	49,1023
	8	2,6551	1,56802	0,918252	0,751448	0,916981	1,0237	1,20253	43,283
	9	2,7485	1,57066	0,915131	0,70652	0,886177	1,01047	1,20094	42,6103
	10	2,6589	1,54552	0,974035	0,699004	0,917047	0,985109	1,19204	49,4376
	Mediana	2,6573	1,566765	0,9376065	0,723864	0,9055235	1,00662	1,192445	46,3217
	Ganho	1	1,698	2,835	3,685	2,935	2,66	2,234	0.,057

N	Medições/NºThreads	Sequential:	2	4	8	16	20	24	48
	1	42,4606	21,5024	10,9754	5,9989	5,74925	5,48729	7,48019	47,6137
	2	42,4629	21,5709	10,9308	5,86119	5,64862	5,56224	7,31929	43,1847
	3	44,6209	21,4874	11,0116	5,88829	5,83372	5,52449	7,39129	44,1
	4	42,6577	21,4972	10,9757	5,85873	5,73918	5,62179	7,60269	45,7461
4096	5	42,5303	21,6347	10,9995	5,8486	5,99296	5,55034	6,5914	49,8313
	6	44,6849	21,5525	10,9518	5,83868	5,94646	5,55647	7,48248	49,2071
	7	42,5199	21,4182	10,9631	5,81746	5,81253	5,66483	7,31435	41,8855
	8	42,3672	21,6069	11,0511	5,85037	5,72775	5,55492	6,06806	45,4041
	9	42,4251	21,5156	11,0237	5,79345	5,72511	5,54837	7,38582	42,6462
	10	42,566	21,3975	10,9739	5,88026	5,91803	5,48483	7,43573	7,12606
	Mediana	42,5199	21,509	10,97555	5,85455	5,78089	5,555695	7,388555	45,5751
	Ganho	1	1,977	3,875	7,264	7,356	7,659	5,756	0,95

N	Medições/#Threads	Sequencial	2	4	8	16	20	24	48
8192	1	127,90	90,66	46,47	23,74	15,41	15,35	21,67	43,24
	2	130,06	89,33	44,35	27,24	14,45	15,47	20,88	45,19
	3	127,82	85,53	46,02	24,94	14,67	15,81	21,69	51,06
	4	127,82	86,68	48,24	24,92	16,72	16,03	22,16	47,09
	5	127,82	89,47	48,41	23,34	14,89	15,22	21,44	52,67
	6	127,82	85,55	44,24	22,91	14,95	15,22	21,26	45,65
	7	127,82	91,18	44,46	27	15,95	15,22	21,63	43,72
	8	127,82	90,37	43,63	23,74	14,54	15,22	21,5	45,42
	9	127,82	86,77	48,26	23,17	14,71	15,22	21,16	41,57
	10	127,82	90,43	43,95	26,47	15,65	15,22	21,38	47,42
	Mediana	127,82	89,85	45,24	24,33	14,92	15,2218	21,44	45,42
	Ganho		1,43	2,83	5,26	8,57	8,22	5,96	2,76

N	Medições/#Threads	Sequencial	2	4	8	16	20	24	48
16384	1	679,37	345,23	174,1	89,72	49,64	49,67	69,75	108,15
	2	714,6	342,95	174,48	88,24	50,55	48,35	71,2	111,6
	3	680,29	345,6	171,94	87,67	49,58	48,7	73,83	102,92
	4	521,36	343,88	171,91	91,44	50,81	47,79	68,56	108,47
	5	510,14	344,72	173,95	90,88	47,43	48,68	50,52	107,52
	6	536,65	345,05	171,4	89,19	50,1	48,29	67,14	111,29
	7	515,29	343,18	168,84	89,52	49,30	47,9	72,25	108,95
	8	554,34	340,89	174,91	87,13	50,11	48,18	71,46	105,43
	9	521,68	342,97	171,61	86,36	49,88	48,15	71,99	102,59
	10	536,13	344,75	173,12	90,92	47,33	48,70	67,53	111,07
	Mediana	536,39	344,3	172,53	89,355	50,1	48,29	71,2	108,47
	Ganho		1,56	3,09	6	10,78	11,1	7,6	4,95

N	Medições/#Threads	Sequencial	2	4	8	16	20	24	48
32000	1	2597,69	1316,43	653,24	328,06	179,8	140,97	185	282,14
	2	2596,2	1290,71	650,49	327,46	175,04	177,83	205,22	302,47
	3	2598,57	1293,16	646,83	327,18	173,84	175,19	187,41	304,36
	4	2593,93	1289,54	654,87	325,27	173,28	190,26	264,19	307,5
	5	2594,1	1291,59	648,28	325,42	172,98	177,07	244,07	292,52
	6	2725,05	1299,46	654,62	329,01	176,06	174,5	254,5	272,29
	7	2724,87	1291,33	645,95	325,15	182,39	176,28	263,58	304,43
	8	2596,16	1289,92	646,88	325,49	172,56	181,58	243,39	305,87
	9	2596,9	1303,75	652,83	325,44	174,62	181,031	260,48	297,15
	10	2594,45	1300,81	651,08	326,51	174,38	182,3	204,28	295,52
	Mediana	2596,18	1291,59	650,49	325,49	174,62	177,45	244,07	302,47
	Ganho		2	3,99	7,97	14,9	14,5	10,65	8,68