Universidade do Minho

Mestrado em Engenharia Informática

Algoritmos Paralelos - 2014/2015

Paralelização OpenMP e PThreads em Multiplicação de Matrizes

10 de Abril de 2015

Fábio Gomes pg27752

Índice

Introdução

O problema que nos foi apresentado está relacionado com o algoritmo de multiplicação de Matrizes comparando os tempos e os misses da versão Paralela do PDF, com *OpenMP* e *PThreads*.

Os ficheiros foram compilados com flag -O3 e os Misses simulados com o CacheGrind do Valgrind.

Versão Paralela do PDF

A versão disponibilizada indicou os seguintes resultados:

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	**Time**	**Eff.**	**Time**	**Eff.**	**Time**	**Eff.**
1	0,322	1	0,264	1	0,333	1
2	0,219	0,735	0,189	0,698	0,3	0,555
4	0,141	0,571	0,119	0,555	0,303	0,275

Versão Paralela OpenMP

Com OpenMP a parte da rotina de multiplicação ficou assim

```
#pragma omp parallel for num_threads(thread_count) default(none) private(i, j)
shared(A, x, y, m, n)
   for (i = 0; i < m; i++){
    y[i] = 0.0;
    for (j = 0; j < n; j++)
    y[i]+= A[i*n+j]*x[j];
   }
```

Testando 3 vezes obtive os seguintes resultados:

Teste 1

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,094	1	0,076	1	0,097	1
2	0,057	0,825	0,049	0,773	0,071	0,683
4	0,039	0,600	0,034	0,555	0,051	0,474

Teste 2

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,093	1	1,000	1	0,095	1
2	0,055	0,857	0,583	0,857	0,070	0,681
4	0,038	0,616	0,406	0,616	0,053	0,445

Teste 3

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,094	1	0,076	1	0,095	1
2	0,057	0,819	0,048	0,798	0,070	0,675
4	0,039	0,599	0,035	0,547	0,051	0,468

Resultando na seguinte média

Teste Média

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,094	1	0,384	1	0,095	1
2	0,056	0,833	0,227	0,847	0,070	0,680
4	0,039	0,604	0,158	0,606	0,052	0,462

Versão Paralela PThreads

Devido ao modo como o PThreads funciona ficou uma versão diferente obrigando a calcular as linhas que cada Thread trabalha. Como se fazia com os processos em MPI.

```
for (thread = 0; thread < thread_count; thread++)
      pthread_create(&thread_handles[thread], NULL,
         Pth_mat_vect, (void*) thread);

   for (thread = 0; thread < thread_count; thread++)
      pthread_join(thread_handles[thread], NULL);



...

void *Pth_mat_vect(void* rank) {
   long my_rank = (long) rank;
   int i;
   int j;
   int local_m = m/thread_count;
   register int sub = my_rank*local_m*n;
   int my_first_row = my_rank*local_m;
   int my_last_row = (my_rank+1)*local_m - 1;

#  ifdef DEBUG
   printf("Thread %ld > my_first_row = %d, my_last_row = %d\n",
         my_rank, my_first_row, my_last_row);
#  endif

   for (i = my_first_row; i <= my_last_row; i++) {
      y[i] = 0.0;
      for (j = 0; j < n; j++)
          y[i] += A[sub++]*x[j];
   }

   return NULL;
}
```

Testando 3 vezes obtive os seguintes resultados:

Teste 1

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,085	1	0,054	1	0,036	1
2	0,073	0,579	0,055	0,492	0,028	0,631
4	0,084	0,252	0,073	0,186	0,038	0,236

Teste 2

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,084	1	0,066	1	0,033	1
2	0,091	0,465	0,055	0,600	0,028	0,583
4	0,010	2,108	0,069	0,240	0,038	0,215

Teste 3

	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,084	1	0,053	1	0,035	1
2	0,069	0,607	0,046	0,574	0,026	0,662
4	0,085	0,247	0,058	0,231	0,036	0,239

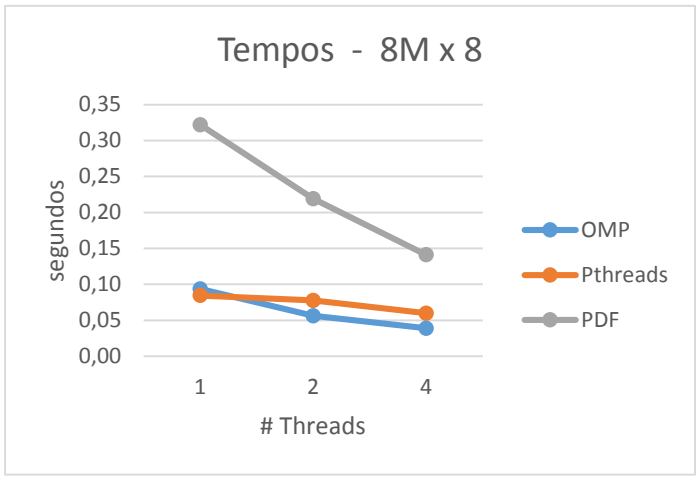
Resultando na seguinte média

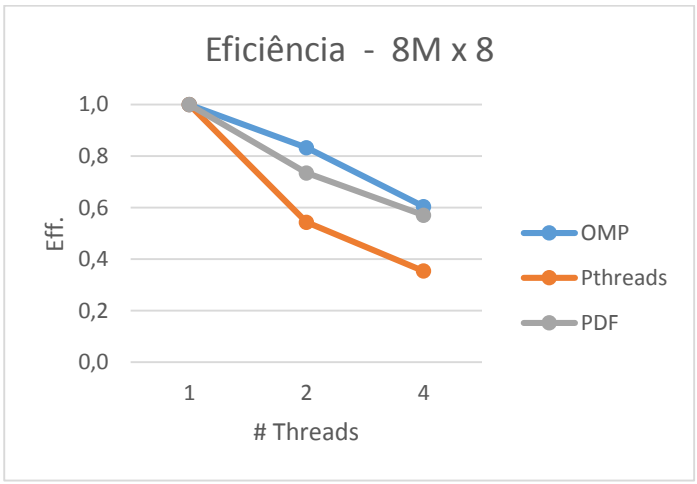
Teste Média

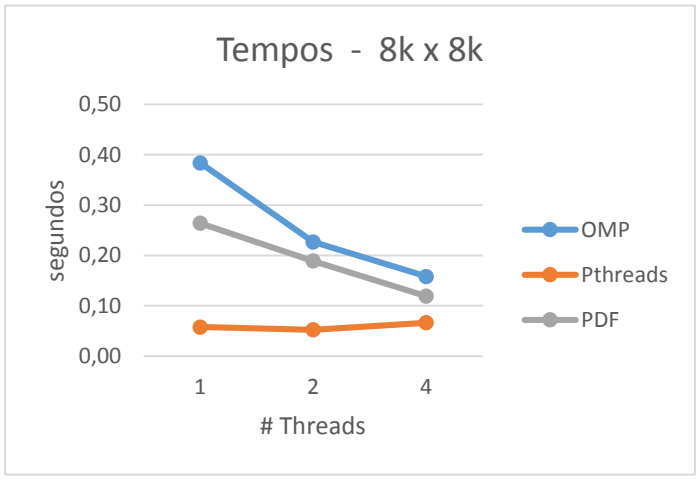
	Matrix Dimension					
	8,000,000x8		8000x8000		8x8,000,000	
Threads	Time	Eff.	Time	Eff.	Time	Eff.
1	0,084	1	0,058	1	0,034	1
2	0,078	0,544	0,052	0,554	0,028	0,624
4	0,060	0,354	0,066	0,218	0,037	0,230

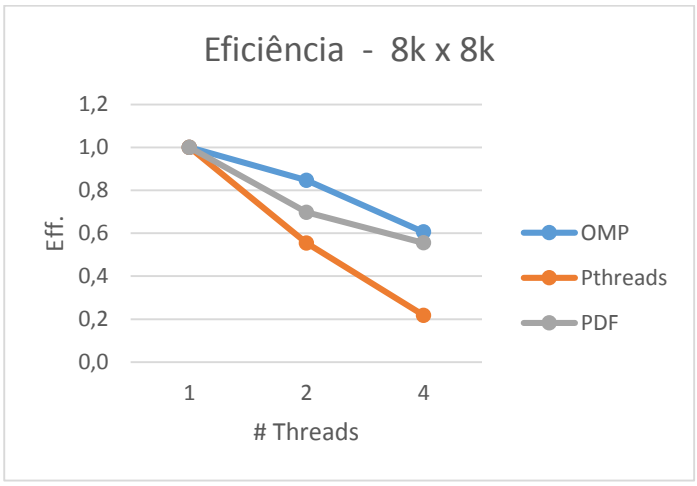
Comparação de Tempos

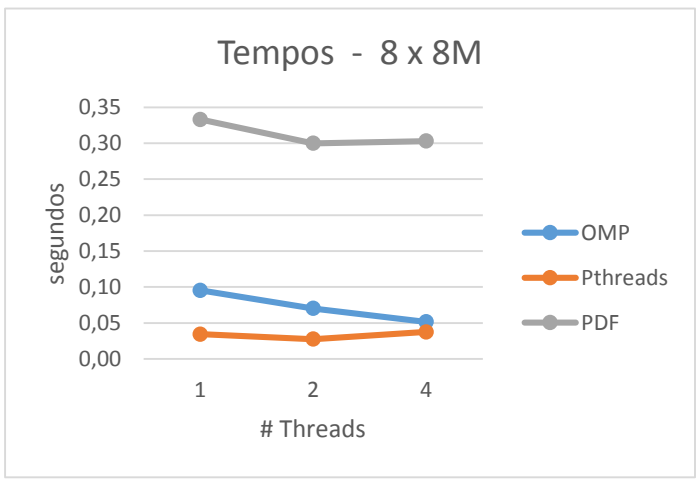
Comparando os Tempos com as 3 implementações, na maior parte das vezes a versão PThreads foi a melhor. Como os Tempos descem mais em relação às outras a eficiência também desce sendo a versão com menor eficiência devido à forma como é calculada.

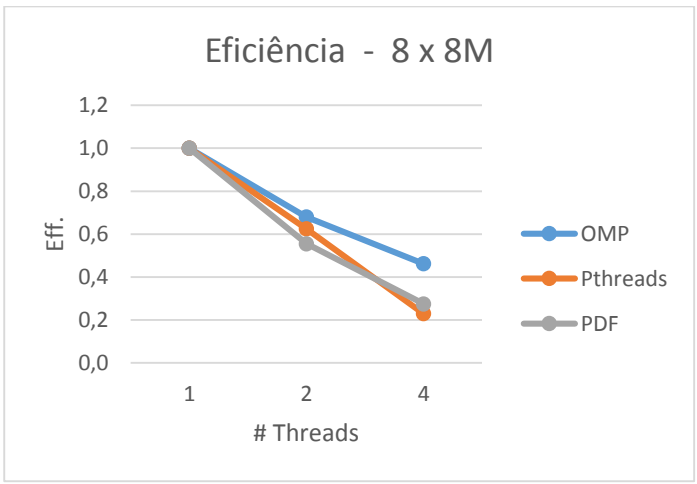






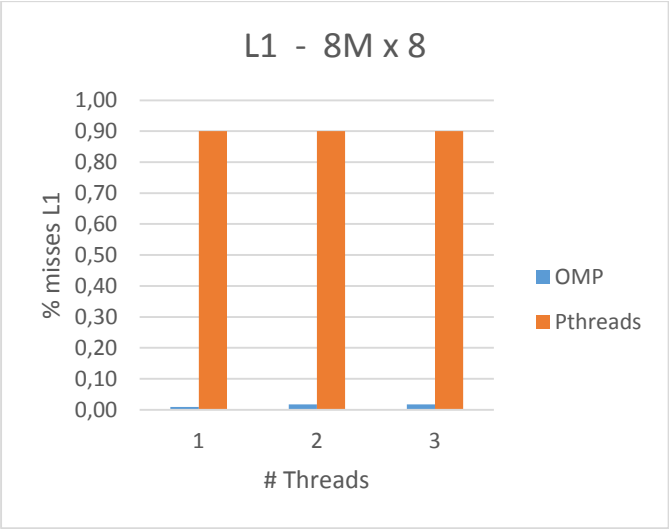


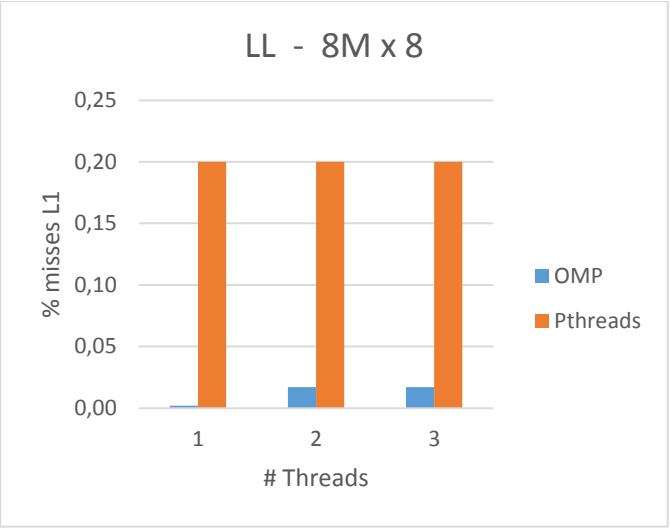


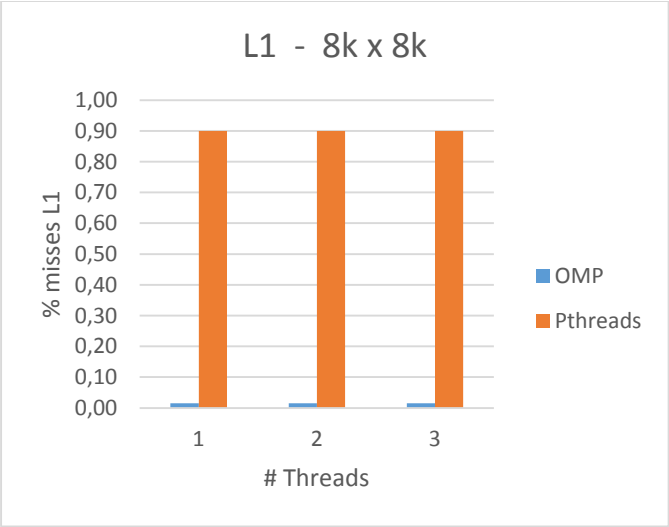


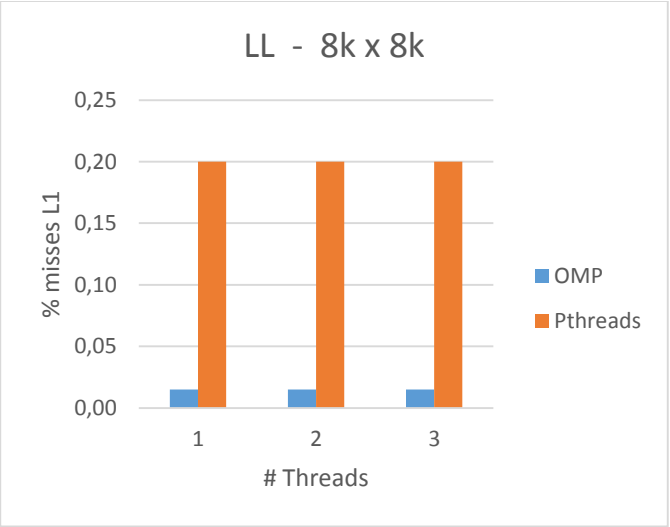
Comparação de Misses

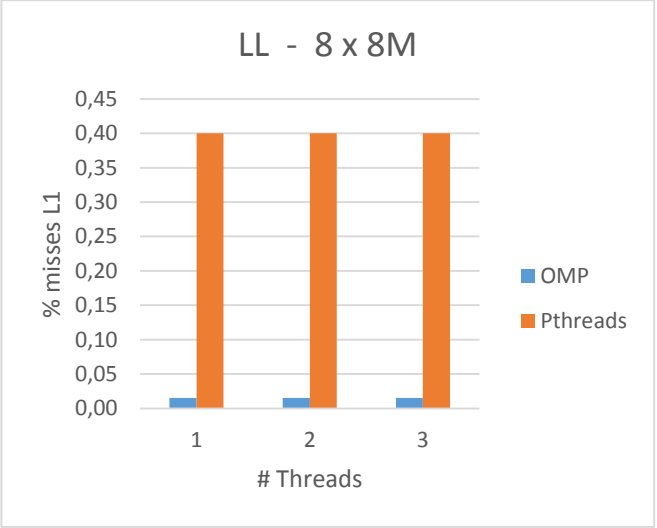
Apesar da versão com PThreads ser a mais rápida, os Misses são muito maiores que a OpenMP.

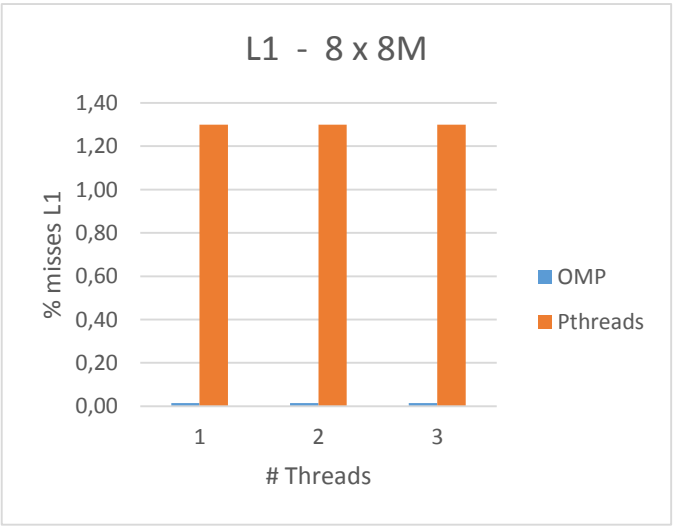












Conclusão

Pela segunda vez consecutiva a versão PThreads conseguiu ser a melhor implementação apesar de ser mais difícil implementa-la pois implica mudanças na estrutura do código enquanto a OpenMP pode ser resolvida com alguns *pragmas* e pequenas alterações apenas no método/função em questão.