Relatório de Entrega de Trabalho
Disciplina de Programação C – Prof. Filipo Mór

Aluno: Fulano da Silva / Ciclano Pereira
Exercício: Relatório 2 **Entrega:** DD/MM/AAAA

1) Implementação

Foi implementado um código C MPI onde o processo mestre executa a leitura de um arquivo contendo valores inteiros separados por vírgula e armazena estes valores na forma de um vetor em memória. Em seguida, o processo mestre divide este vetor em *k* segmentos, sendo *k* o número de processos escravos disparados. Cada segmento do vetor contém *q* posições, sendo *q = n/k* (número de elementos do vetor dividido pelo número de processos escravos). Caso a divisão possua resultado não exato, então o valor de *q* será configurado para o valor absoluto (sem resto) da divisão. Neste caso, porém, o último processo escravo (que contém o *process ID* mais alto) receberá um segmento de vetor com maior quantidade de elementos a serem somados do que os processos escravos anteriores. Em seguida, o processo mestre envia os segmentos correspondentes a cada um dos processos escravo, que por sua vez efetuam a soma dos valores dos elementos do seu segmento e retornam o valor da soma para o mestre. Finalmente, o mestre efetua a soma dos resultados intermediários retornados pelos escravos, exibindo o valor final da soma através de um comando *printf*. A topologia da aplicação é mostrada na Figura 1.

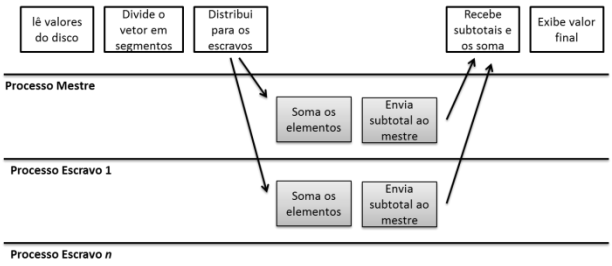


Figura 1- topologia da aplicação desenvolvida.

2) Dificuldades encontradas

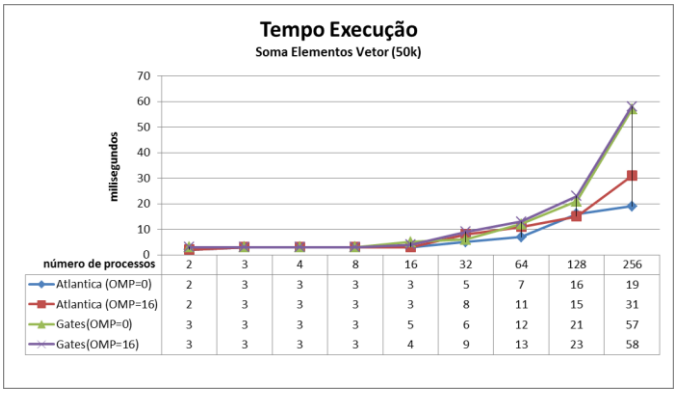
Depuração do código MPI; sintaxe no C para alocação de memória junto aos comandos MPI.

3) Testes

Foram executados testes com alocação de 2, 3, 4, 8, 16, 32, 64, 128 e 256 processos nos clusteres atlantica (dois nodos alocados) e gates (dezesseis nodos alocados) do LAD e em duas versões do código. A primeira versão com inclusão de diretivas OpenMP de 16 threads no processo escravo (no loop de soma dos valores do segmento recebido) e uma versao sem uso de OpenMP. O arquivo de entrada continha 50.000 valores (gerados automaticamente no *range* 1..49.999). Desta forma, o resultado final pode ser facilmente verificado aplicando-se a fórmula $\sum_{i=1}^{n} i = \frac{n(n-1)}{2}$.

4) Análise do Desempenho

Verificou-se que para pequenas quantidades de processos disparados (entre 2 e 8 processos) o desempenho da aplicação praticamente não foi alterado. No entanto, ao utilizar-se quantidades a partir de 16, o desempenho foi degradando de forma crescente e não linear, conforme verifica-se na Figura 2. Este comportamento se deve, provavelmente, ao *overhead* gerado pela concorrência das threads durante a execução, uma vez que a quantidade de núcleos é limitada em cada nodo (2 na gates e 16 na atlantica). Já o uso de OpenMP parece ter gerado uma pequena diferença no desempenho da aplicação quando executada na máquina atlantica, que conta com mais núcleos a disposição em cada nodo. Este comportamento sugere que o uso de OpenMP pode ser interessante para *n* maiores em máquinas com maior quantidade de núcleos. Porém, para máquinas com poucos núcleos, a utilização de OpenMP parece ser desnecessária, pois o desempenho não foi afetado pela sua utilização.



número de processos	2	3	4	8	16	32	64	128	256
Atlantica (OMP=0)	2	3	3	3	3	5	7	16	19
Atlantica (OMP=16)	2	3	3	3	3	8	11	15	31
Gates(OMP=0)	3	3	3	3	5	6	12	21	57
Gates(OMP=16)	3	3	3	3	4	9	13	23	58

Figura 2 - Resultados.

5) Observações Finais

Os testes realizados permitiram a identificação de uma tendência de degradação no desempenho da aplicação quando aumentando-se a quantidade de processos na execução. No entanto, provavelmente o volume de dados utilizado nos testes tenha sido insuficiente para que fosse possível se chegar a um mapeamento seguro deste comportamento. Portanto, para trabalhos futuros, sugere-se o aumento do tamanho do vetor. Para tal, o código atual deverá ser atualizado para utilizar variáveis *long int* ao invés de *int*, a fim de permitir a alocação de vetores significantemente maiores.

Fontes no LAD: */home/ppd05/PP/trabalho1/ somaVetor_v2_OMP.c* e *somaVetor_v2.c (ambos enviados para entregas@filipomor.com)*