DECOMPOSIÇÃO RECURSIVA DE TAREFAS E MAP REDUCE

João Victor de Mesquita Cândido dos Santos **RA:** 102028
Raphael Ribeiro Faria **RA:** 104120

Turma: Noturno
Unidade Curricular: Programação Concorrente e Distribuída
Docente: Dr. Álvaro Luiz Fazenda

DESCRIÇÃO DO PROBLEMA 1:

Desenvolver um programa, utilizando JavaThreads, C+PThreads ou C+OpenMP, para fazer ordenação de valores numéricos dispostos em um vetor de tamanho: N = 10^7, com dados do tipo *Double*. Deve-se utilizar necessariamente o algoritmo de *MergeSort*, fazendo a decomposição de tarefas de forma recursiva definindo um limite para a quantidade de *Threads* abertas ao mesmo tempo.

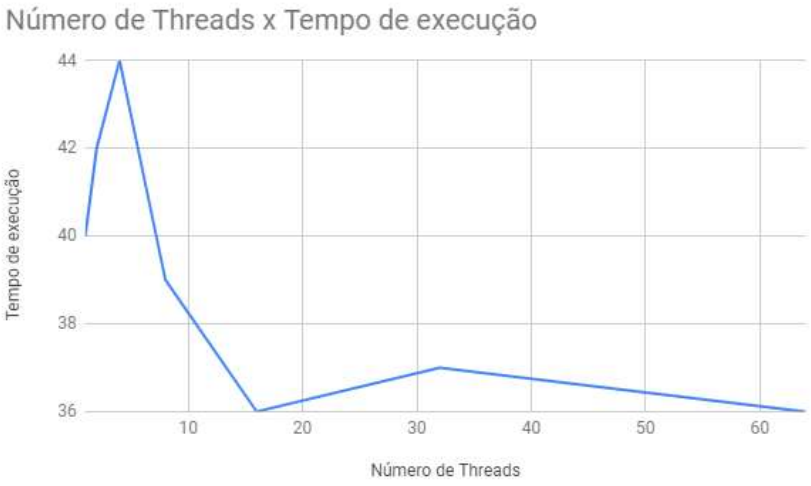
SOLUÇÃO:

Para resolver o problema proposto, primeiramente foram gerados valores aleatórios para realizar o procedimento, e armazenados em um arquivo de texto. Para a ordenação desses valores, foi criado um algoritmo de Map-Reduce que distribui os elementos criados em vários arquivos de texto, sendo que cada arquivo contém um número igual de elementos. Para cada arquivo, foi aplicado o algoritmo de ordenação proposto (MergeSort), fazendo com que cada arquivo esteja com os seus valores ordenados corretamente.

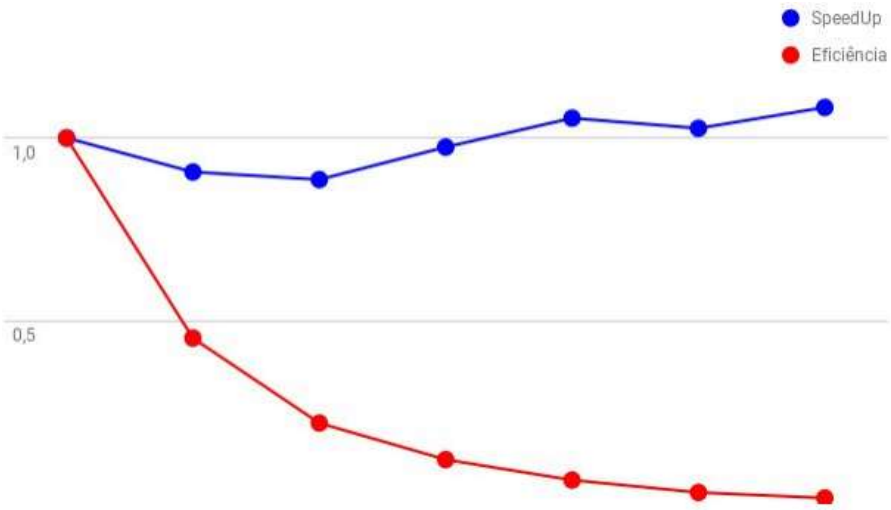
Após realizada a ordenação de forma individual, o algoritmo utilizou o método reduce para analisar todos os arquivos e inserir os valores em ordem crescente em um único arquivo, obtendo assim, a ordenação completa de todos os elementos aleatórios gerados.

A medição de tempo se concentrou apenas no processamento da ordenação, desconsiderando o tempo gasto para alocação e preenchimento do vetor, gerando um gráfico de Speedup e Eficiência utilizando 1, 2, 4, 8, 16, 32 e 64 threads.

Tempo de execução(s)	Número de Threads
40	1
42	2
44	4
39	8
36	16
37	32
36	64



- **Gráfico do Speedup e eficiência:**



DESCRIÇÃO DO PROBLEMA 2:

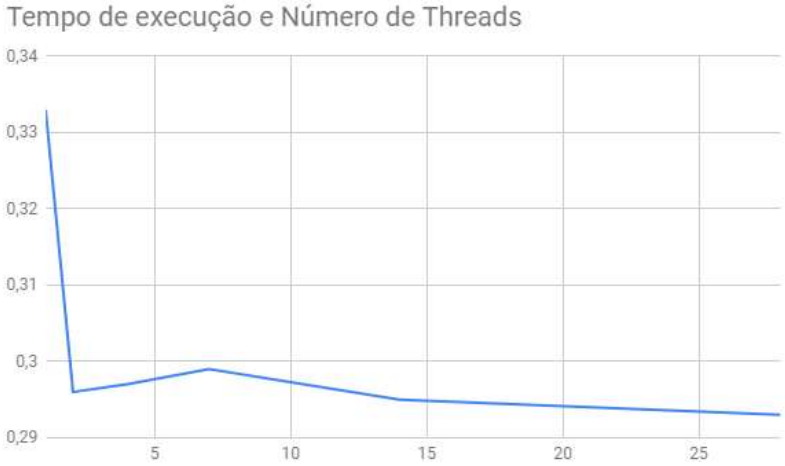
A partir do código-fonte exemplo *"mapreduce_serial.c"*, implementar uma versão paralela em *OpenMP* que conte a quantidade de ocorrências de cada código de gastos (*numSubCota*) do arquivo *csv* fornecido. O código-fonte exemplo conta ocorrências do arquivo *"data.txt"* (também fornecido), que contém apenas números, desta forma, deve-se adaptar ou criar um novo código-fonte para realizar o que se pede.

SOLUÇÃO:

Para resolver o problema proposto, foi criado um código em OpenMP que lê todos os dados do arquivos separados por tokens, e assim, foi pego o token desejado e verificado a quantidade de vezes em que um determinado código de gastos foi encontrado.

As medições de tempo foram feitas utilizando-se 1, 2, 4, 7, 14 e 28 threads.

Tempo de execução(ms)	Número de Threads
0,333	1
0,296	2
0,297	4
0,299	7
0,295	14
0,293	28



- **Gráfico do Speedup e eficiência:**