OddEvenSort Paralelo, QuickSort Paralelo e QuickSort Sequencial em MPI e seus Desempenhos.

Victor Lelis Soares

28 de outubro de 2021

1 Introdução

A ordenação de elementos é um dos problemas mais amplamente estudado na computação, tanto algoritmos sequenciais quanto paralelos, a ordenação pode ser baseada em comparação ou não baseada em comparação.

2 Descrição dos Algoritmos

Com exceção do QuickSort sequencial, todos os outros algoritmos foram executados em ambiente MPI, tirando proveito da troca de mensagens e de sua memória distribuída. A saída de ambos os algoritmos, são os tempos de execução em cada processador, e o tempo de execução do algoritmo QuickSort sequencial.

2.1 OddEvenSort

Esta implementação, primeiramente, recebe um arquivo de entrada pela linha de comando que contém, respectivamente: a quantidade n de valores inteiros e na linha seguinte n elementos, após isso, é feita a separação dos elementos entre os p processadores, de tal forma que cada um possua n/p elementos(assumindo que n seja divisível por p).O algoritmo de fato começa com a ordenação local dos n/p elementos de cada processador(usando o QuickSort sequencial), com o critério que foi especificado no documento do trabalho.
Após essa ordenação local, na fase par, cada processador de índice par faz a operação troca-e-separa com seu respectivo processador de índice impar, da mesma forma, na fase impar, o mesmo vale para os processadores de índice impar que realizam a comunicação com os processadores de índice par. Dessa forma, localmente entre os processadores que fizeram essa comunicação, os elementos estarão localmente ordenados, e isso segue até que esteja totalmente ordenado.

2.2 QuickSort Sequencial

O quicksort usa a estratégia de divisão e conquista. A estratégia consiste em rearranjar as chaves de modo que as chaves "menores"precedam as chaves "maiores". Em seguida o quicksort ordena as duas sublistas de chaves menores e maiores recursivamente até que a lista completa se encontre ordenada.

2.3 QuickSort Paralelo

Esta implementação, primeiramente, recebe um arquivo de entrada pela linha de comando que contém, respectivamente: a quantidade n de valores inteiros e na linha seguinte n elementos, após isso, é feita a separação dos elementos entre os p processadores, de tal forma que cada um possua n/p elementos(assumindo que n seja divisível por p).
O processador p_0 compartilha com todos os outros processadores do grupo o seu elemento $nlocal/2$ via Broadcast, e após isso, cada processador separa seu elementos em: uma lista l que são seus elementos menores ou iguais que o pivô e em uma lista u com os elementos maiores que o pivô. Feito isso, os processadores que se encontram na parte mais baixa($p_i < processadores/2$), vão emparelhar com um processador da parte mais alta, enviando para a parte mais alta os maiores elementos(u) e recebendo os menores elementos(l) do processador emparelhado na parte mais alta, após isso, podemos ver que todos os elementos menores que o pivô estão na parte inferior da máquina e todos os elementos maiores que o pivô estão na parte superior. O processo acima recursivamente dividindo os processadores em subgrupos até que cada processador tenha sua própria lista local, que é ordenada localmente utilizando o QuickSort sequencial.

3 Ambiente dos Experimentos

O presente experimento foi executado em um computador com as seguintes características:

- Processador: Intel® Core™ i5-4570 CPU @ 3.20GHz × 4.
- GPU: Intel® HD Graphics 4600 (HSW GT2).
- Disco rígito: HD Seagate BarraCuda, 1TB, 3.5, SATA 3.
- Memória RAM: 4GB DDR3 1666Mhz.
- Sistema Operacional: Arch Linux 64 Bits, Release: 2021.10.01, Kernel: 5.14.8.
- Versão GCC: 9.3.0.
- Versão OpenMPI: 4.1.1.

4 Experimentos

Para executar este experimento e chegar ao presente resultado, fiz um grande número de execuções para cada situação e fazendo fazendo uma amostragem, chegando em resultados justos para os algoritmos descritos anteriormente.
O SpeedUp dos algoritmos foi calculado da seguinte maneira:

$$SpeedUp = \frac{TempoDoalgoritmoSequencial}{TempodoAlgoritmoParalelo}$$

Perceba que, a medida em que aumento o número de processadores, o desempenho dos algoritmos paralelos começa a decair, mesmo que pouco graficamente, isso acontece pois estou executando em uma máquina com 4 núcleos, então, com mais do que 4 processos, começa a ter concorrência de núcleos entre os processos, ou seja, temos ao menos 2 processos disputando um núcleo, o que diminui o desempenho dos algoritmos paralelos.

Figura 1: Gráfico Comparativo entre os algoritmos, utilizando 2 processadores.

Entrada	QuickSortParalelo(pior caso)	OddEvenParalelo(pior caso)	QuickSequencial	p =2
100000	0,003405	0,009354	0,18879	
1000000	0,055505	0,102162	0,190568	
10000000	0,129324	1,230599	2,281856	
speed'up em relação ao quick sequencial	14,13779657	1,982850948	-	

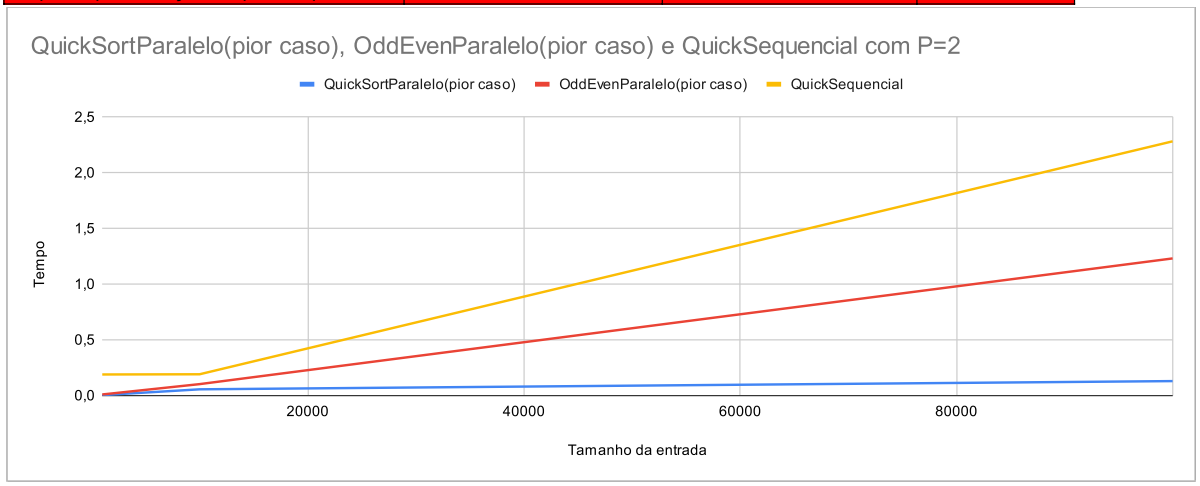


Figura 2: Gráfico Comparativo entre os algoritmos, utilizando 4 processadores.

Entrada	QuickSortParalelo(pior caso)	OddEvenParalelo(pior caso)	QuickSequencial	p = 4
100000	0,00159	0,002563	0,016937	
1000000	0,012087	0,036688	0,191089	
10000000	0,134728	0,419956	2,240135	
speed'up em relação ao quick sequencial	16,49648597	5,331279793	-	

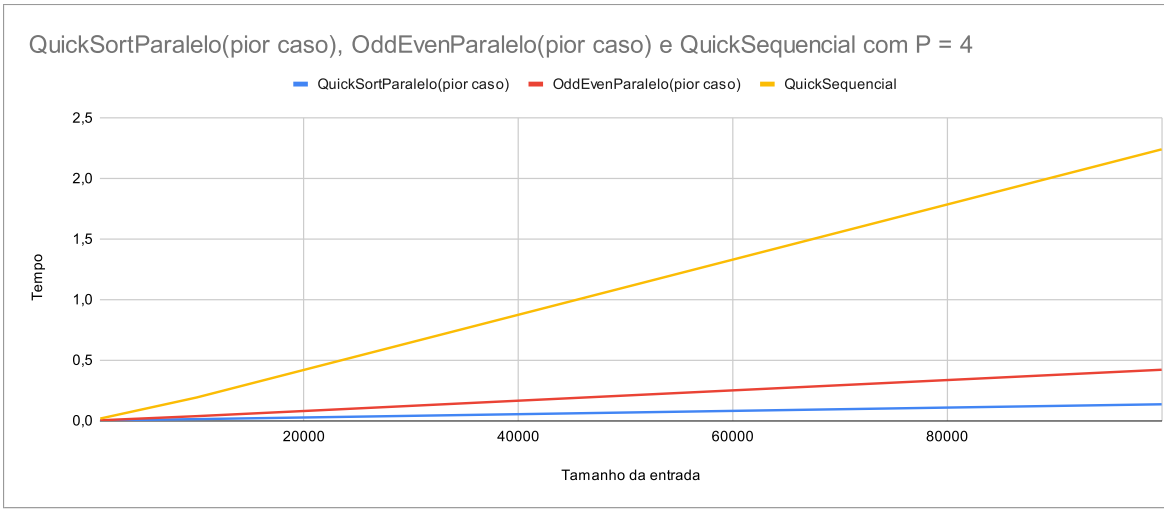


Figura 3: Gráfico Comparativo entre os algoritmos, utilizando 8 processadores.

Entrada	QuickSortParalelo(pior caso)	OddEvenParalelo(pior caso)	QuickSequencial	p = 8
100000	0,07003	0,231772	0,0297	
1000000	0,125516	0,242238	0,398596	
10000000	0,951177	1,056678	4,50763	
speed'up em relação ao quick sequencial	4,304375163	3,224645388	-	

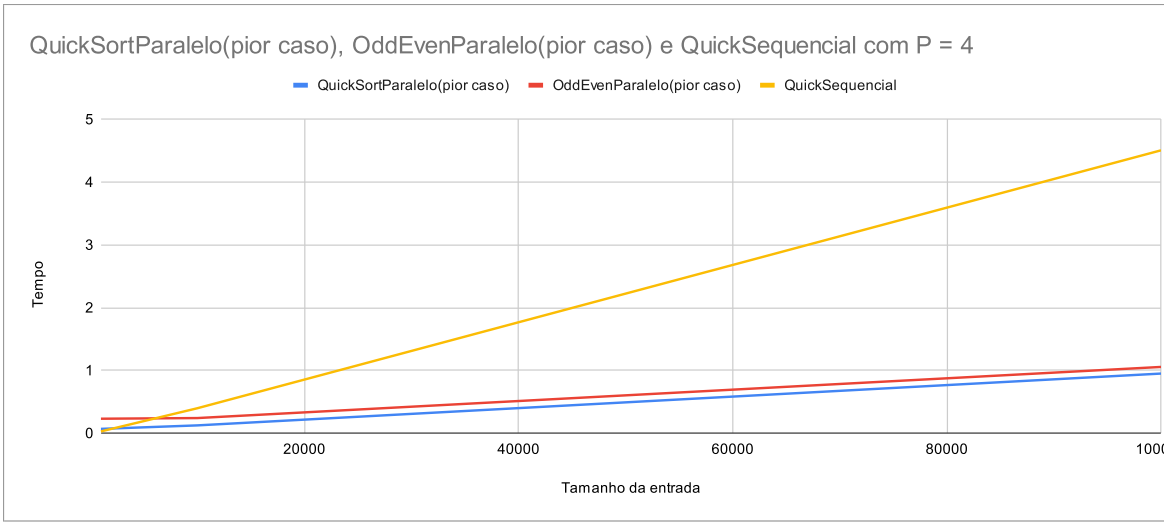
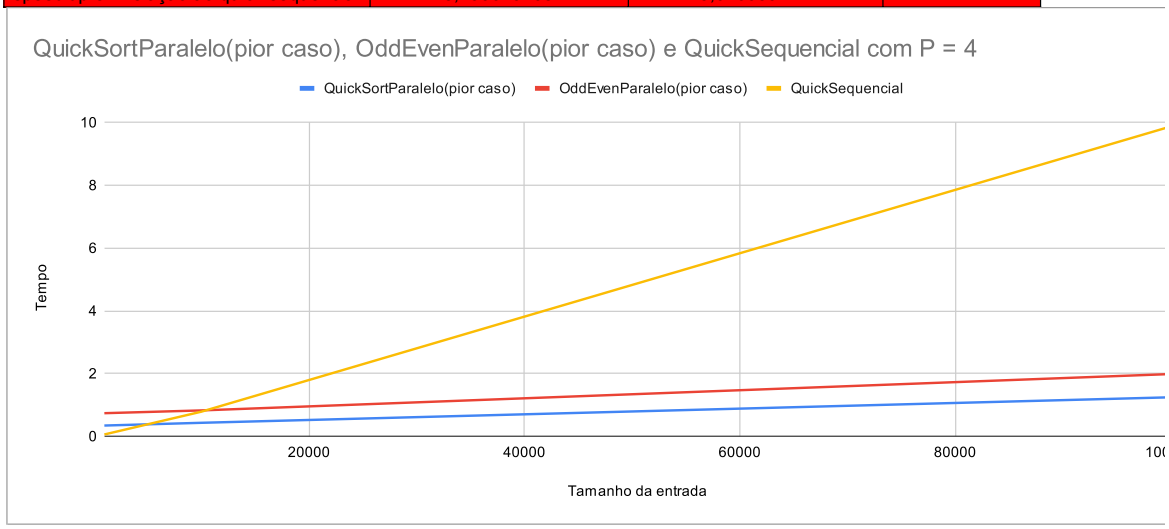


Figura 4: Gráfico Comparativo entre os algoritmos, utilizando 16 processadores.

Entrada	QuickSortParalelo(pior caso)	OddEvenParalelo(pior caso)	QuickSequencial	p = 16
100000	0,346528	0,740058	0,062264	
1000000	0,433738	0,828218	0,787475	
10000000	1,246511	1,987001	9,873187	
speed'up em relação ao quick sequencial	5,290629408	3,016059227	-	



5 Como Executar?

No diretório do arquivo em que se encontra este relatório, seguem todos os códigos fontes, inclusive, um makefile, para compilar os algoritmos OddEven Paralelo e o QuickSort Paralelo. Basta apenas compilar utilizando:

- make all

Após isso, vão ser gerados dois executáveis, oddeven e quick. Para executar com o MPI basta apenas digitar a seguinte linha de comando:

- mpirun -np p –hostfile arquivohost ./executavel arquivoentrada

Caso queira apagar todos os executáveis do diretório, só precisar digitar:

- make clean

Obs: No diretório do arquivo, também há um programa para gerar números aleatórios, chamado de generatenumbers, para auxiliar nos casos de teste, execute-o digitando quantos números deseja gerar e redireciona a saída através do terminal para o arquivo de sua preferência.