

KHALIFA UNIVERSITY

MSC COMPUTING AND INFORMATION SCIENCE

MUHAMMAD MUNEEB - 100052975

Advance Cloud Computing Assignment no 1

March 31, 2021

Contents

A	MPI based sorting	3
A.1	Approach 1 - Producer and Consumer 1	3
A.2	Approach 2 - Binary Tree based merging 2	4
A.2.1	Input Format and parameters	4
B	Without MPI	5
B.1	File system based approach	5
B.1.1	Input Format and parameters	6
C	Some important points	6

List of Figures

1	Method 1	3
2	Method 2	5
3	Input file. First-line represents the size and the number of the processors (chunks). Last invokes the merge functionality. and the remaining files are the name of the files.	6
4	Config file. This file will read each line from the input.txt file and assign parameters to each job.	6
5	Method 1 = Producer-Consumer scenario with chunk size 16, Method 2 = Producer-Consumer scenario with chunk size 32, Method 3 = Binary tree scenario, Method 4 = File system	7
6	Method 1 = Producer-Consumer scenario with chunk size 16, Method 2 = Producer-Consumer scenario with chunk size 32, Method 3 = Binary tree scenario, Method 4 = File system	8

List of Tables

A MPI based sorting

A.1 Approach 1 - Producer and Consumer 1

This is just like a consumer-producer. The producer will produce the chunks and the consumer will sort them. After sorting it will be returned to the producer and the producer will merge it with already sorted chunks.

- Number of the Chunks
- Number of the Process

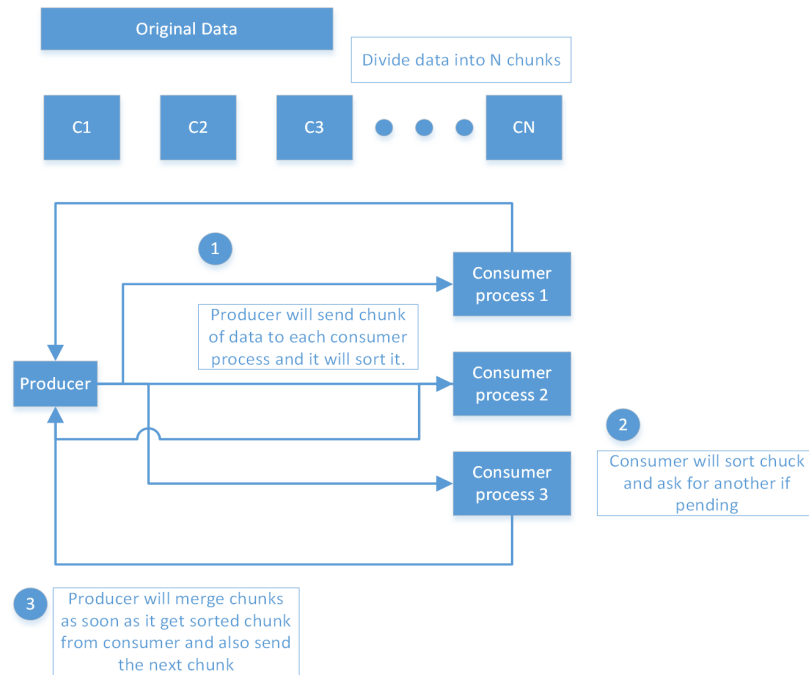


Figure 1: Method 1

Size of the Array	Process	Chunks	Time
100000	4	16	0.627904518
200000	4	16	1.197910529
300000	4	16	1.837172734
400000	4	16	2.47362824
500000	4	16	3.132573469

Size of the Array	Process	Chunks	Time
100000	8	16	0.564916335
200000	8	16	1.193286223
300000	8	16	1.798713388
400000	8	16	2.404711638
500000	8	16	2.964826135

Size of the Array	Process	Chunks	Time
100000	4	32	0.665319508
200000	4	32	1.541839508
300000	4	32	2.112342428
400000	4	32	3.260899819
500000	4	32	3.743955862

Size of the Array	Process	Chunks	Time
100000	8	32	0.743805749
200000	8	32	1.432077572
300000	8	32	2.138749711
400000	8	32	3.270000938
500000	8	32	3.732136792

A.2 Approach 2 - Binary Tree based merging 2

A.2.1 Input Format and parameters

This is the config.sh command which is used for MPI based second approach.
`mpixec -n 8 python mpiapproach2.py 8 100000`

8 is the number of the processor and 100000 is the size of the array list. Same arguments must be used in the config.sh file for this method. Each process will gets its sub-array and sort it and then merge will take place in binary tree-based topology.

Size of the Array	Process	Time
100000	4	0.541922037
200000	4	1.207622157
300000	4	1.750779647
400000	4	2.269990774
500000	4	2.999329958

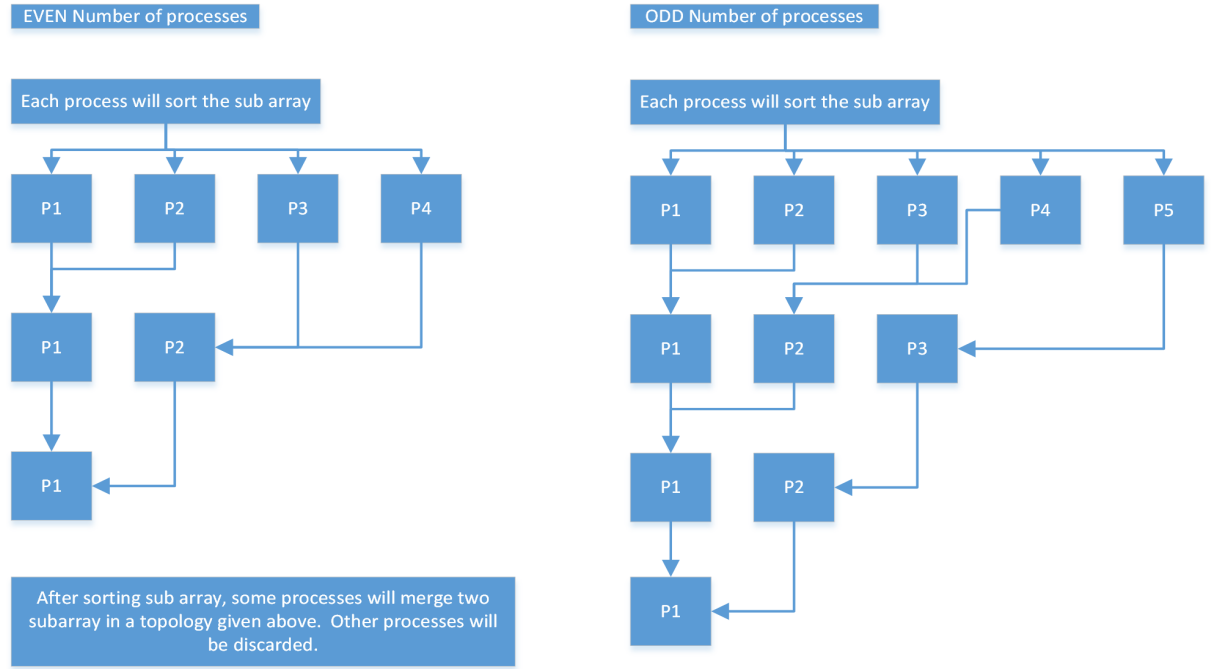


Figure 2: Method 2

Size of the Array	Process	Time
100000	8	0.611630967
200000	8	1.193906015
300000	8	2.115327124
400000	8	2.784451479
500000	8	3.386796344

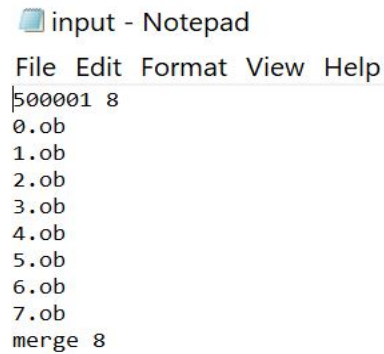
B Without MPI

B.1 File system based approach

In this approach, there is one master node. It will generate the data, divide it into several processors, and store it on disk. Each processor will sort the array and store it on the disk. In the end, there is one more node called merger and it will read data from every file and merge it. Time is calculated from data generation till the final list is not produced. It is important to note that processing nodes or jobs will wait until files are available for processing.

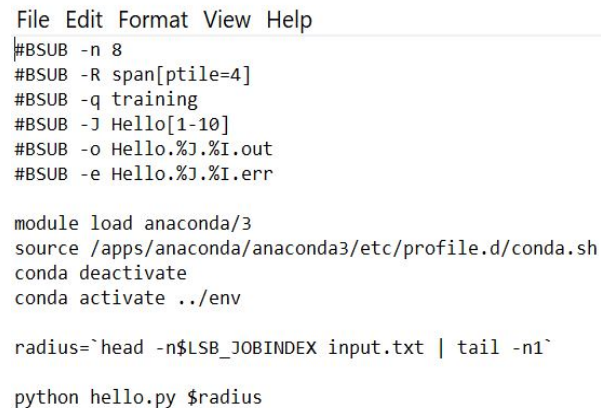
B.1.1 Input Format and parameters

There are three files CODE.py, input.txt, and config.sh. Code file contains the python code and all of the functionality, different functionality will be active depending on the input.txt data.



```
input - Notepad
File Edit Format View Help
500001 8
0.ob
1.ob
2.ob
3.ob
4.ob
5.ob
6.ob
7.ob
merge 8
```

Figure 3: Input file. First-line represents the size and the number of the processors (chunks). Last invokes the merge functionality. and the remaining files are the name of the files.



```
File Edit Format View Help
#BSUB -n 8
#BSUB -R span[ptile=4]
#BSUB -q training
#BSUB -J Hello[1-10]
#BSUB -o Hello.%J.%I.out
#BSUB -e Hello.%J.%I.err

module load anaconda/3
source /apps/anaconda/anaconda3/etc/profile.d/conda.sh
conda deactivate
conda activate ../env

radius=`head -n$LSB_JOBINDEX input.txt | tail -n1`

python hello.py $radius
```

Figure 4: Config file. This file will read each line from the input.txt file and assign parameters to each job.

C Some important points

- Sometimes running job on HPC for small array size can yield large time.

Size of the Array	Process	Time
100000	4	0.19415715
200000	4	0.700092193
300000	4	1.864884157
400000	4	1.525402008
500000	4	2.25743481

Size of the Array	Process	Time
100000	8	0.246226402
200000	8	0.474980811
300000	8	0.613643847
400000	8	0.807208514
500000	8	1.090195287

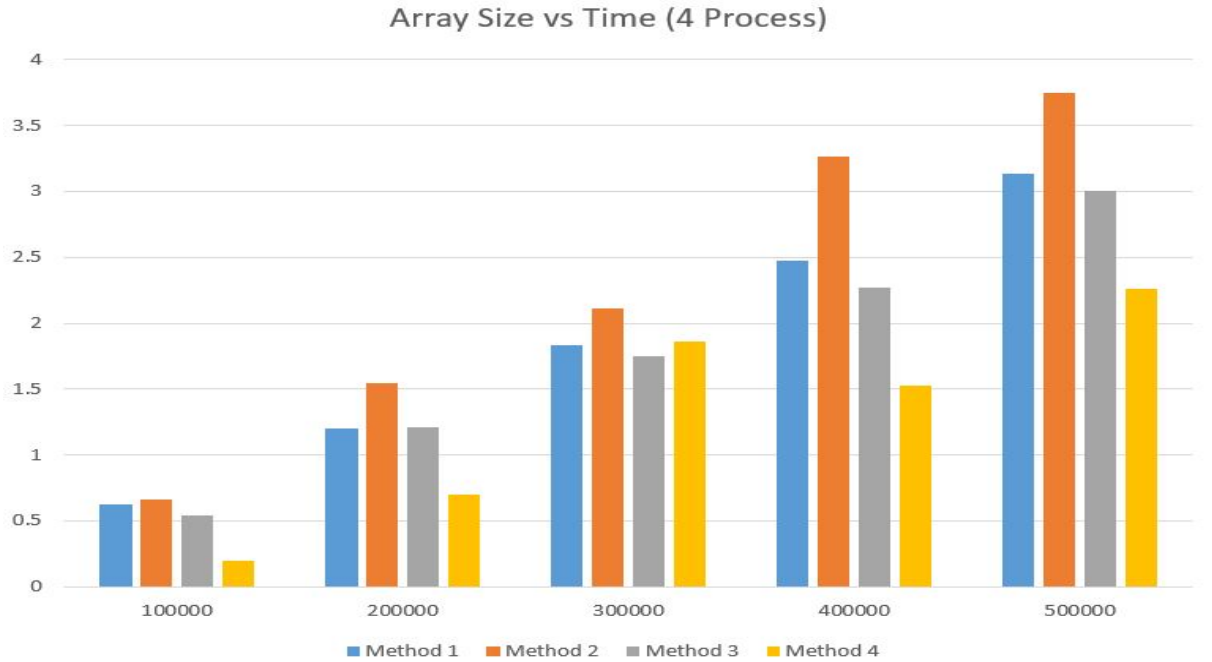


Figure 5: Method 1 = Producer-Consumer scenario with chunk size 16, Method 2 = Producer-Consumer scenario with chunk size 32, Method 3 = Binary tree scenario, Method 4 = File system

- In the fourth method (File system approach) the way we will store the files can affect the performance.
- All the approaches are parameterized and can be run for any number of processes, chunks, and different array sizes (best to my knowledge.)

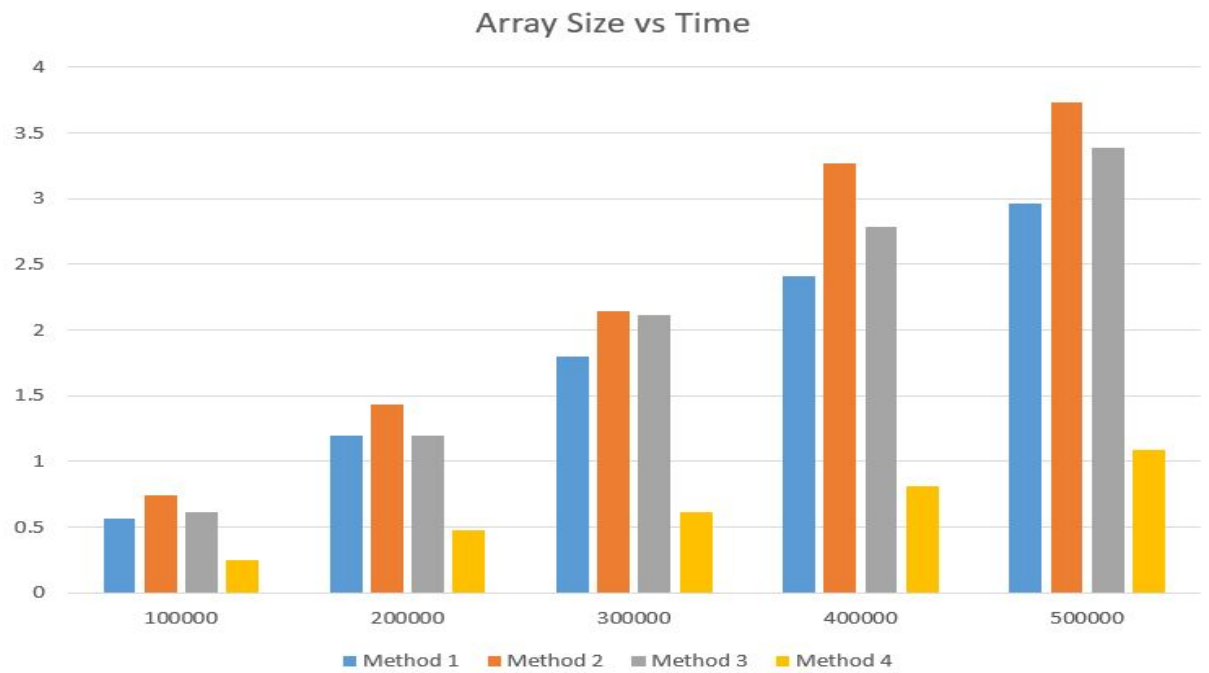


Figure 6: Method 1 = Producer-Consumer scenario with chunk size 16, Method 2 = Producer-Consumer scenario with chunk size 32, Method 3 = Binary tree scenario, Method 4 = File system

-
-