Data Analysis using MPI

CSE 702 Fall ’19
Instructor - Dr. Russ Miller

Presented by Niranjan Mirashi

Contents

- What is Data Analysis?
- Problem Definition
- Data Collection
- Data Cleaning and Text Processing
- How MapReduce works
- Word Count Algorithm in MR
- Serial Execution
- Word Count using MPI
- Algorithm
- Results
- Observations
- References

What is Data Analysis?

- **Data analysis** is the process of evaluating **data** using analytical and statistical tools to discover useful information and aid in decision making.
- With so much data being generated every second, there is always some useful information that can be extracted and used for analysis.

Problem Definition

- Data collection, cleaning and text processing using multiple processors.
- Simulation of a Spark environment using MPI.
- Simulation of “Word count” algorithm (MapReduce).

Data Collection

- Data collection using Twitter API, NYT, CommonCrawl (a public data repo).
- Wrote a program to generate random sentences out of a given word corpus.
- Each processor collects data corresponding to their keyword set in parallel.

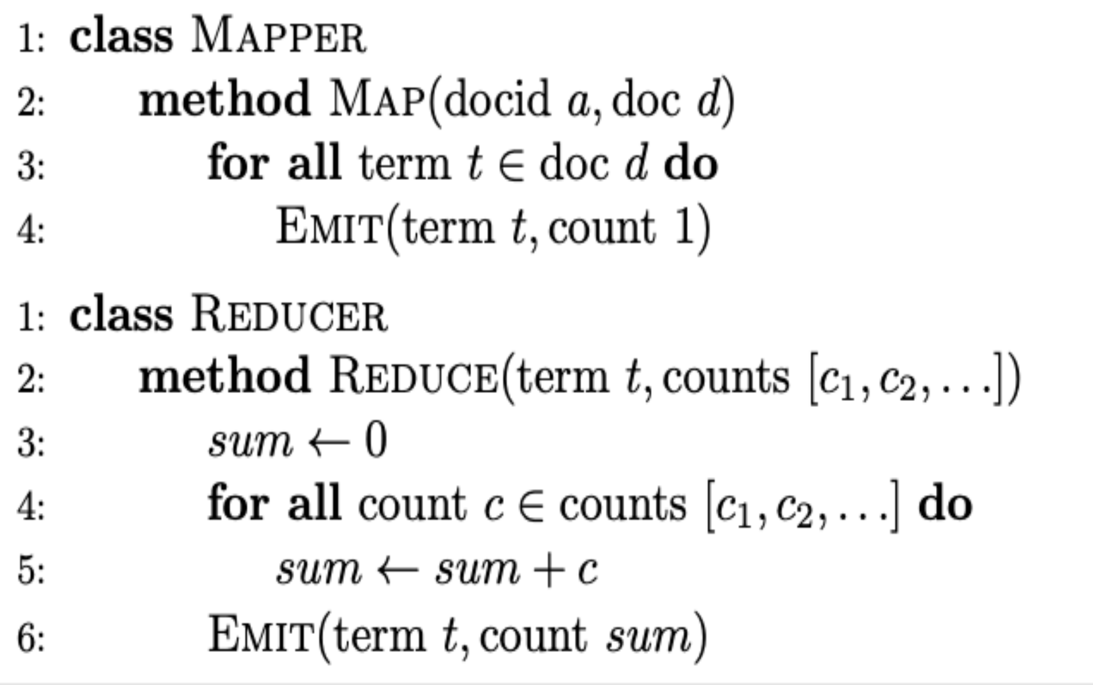
Data Cleaning and Text Processing

- Data Cleaning :
  - Getting rid of html tags and links.
  - Removing non UTC-8 characters.
  - Removing punctuation marks, unnecessary spaces and twitter tags like @rt, etc.
- Text Processing :
  - Lemmatization .
  - Stemming.
  - Removing stop words.

Working of MapReduce

- Divide and Conquer.
- Uses multiple processors.
- Phases of MapReduce –
    - Mapping
    - Shuffling
    - Combining
    - Reducing

Word Count Algorithm in MapReduce



```
class Mapper
    method Map(docid $a$, doc $d$)
        for all term $t \in$ doc $d$ do
            Emit(term $t$, count 1)

class Reducer
    method Reduce(term $t$, counts $[c_1, c_2, \ldots]$)
        $sum \leftarrow 0$
        for all count $c \in$ counts $[c_1, c_2, \ldots]$ do
            $sum \leftarrow sum + c$
        Emit(term $t$, count $sum$)
```

Serial Execution

- Test Parameters :
  - Max data = 138 MB
  - Max number of words = 2,11,74,415
  - Serial Execution time = 402.54 s

Input Size (in MB)	Time (in seconds)
4	12.47
9	26.12
18	52.6
35	102.22
70	210.33
138	402.54

Word Count using MPI

- Mapping Phase – Processors emit (store) a count = 1 for each word in a key-pair format.
- Shuffling Phase – The processors will send the intermediate mapper output to the reducers. But in this case, the processors act as both mappers and reducers. So we skip this phase.
- Combine Phase – Also known as a sub-reducing phase, where each processor will compute total word count for it's respective map.
- Reduce Phase – The local counts are reduced to one global count list.

Algorithm

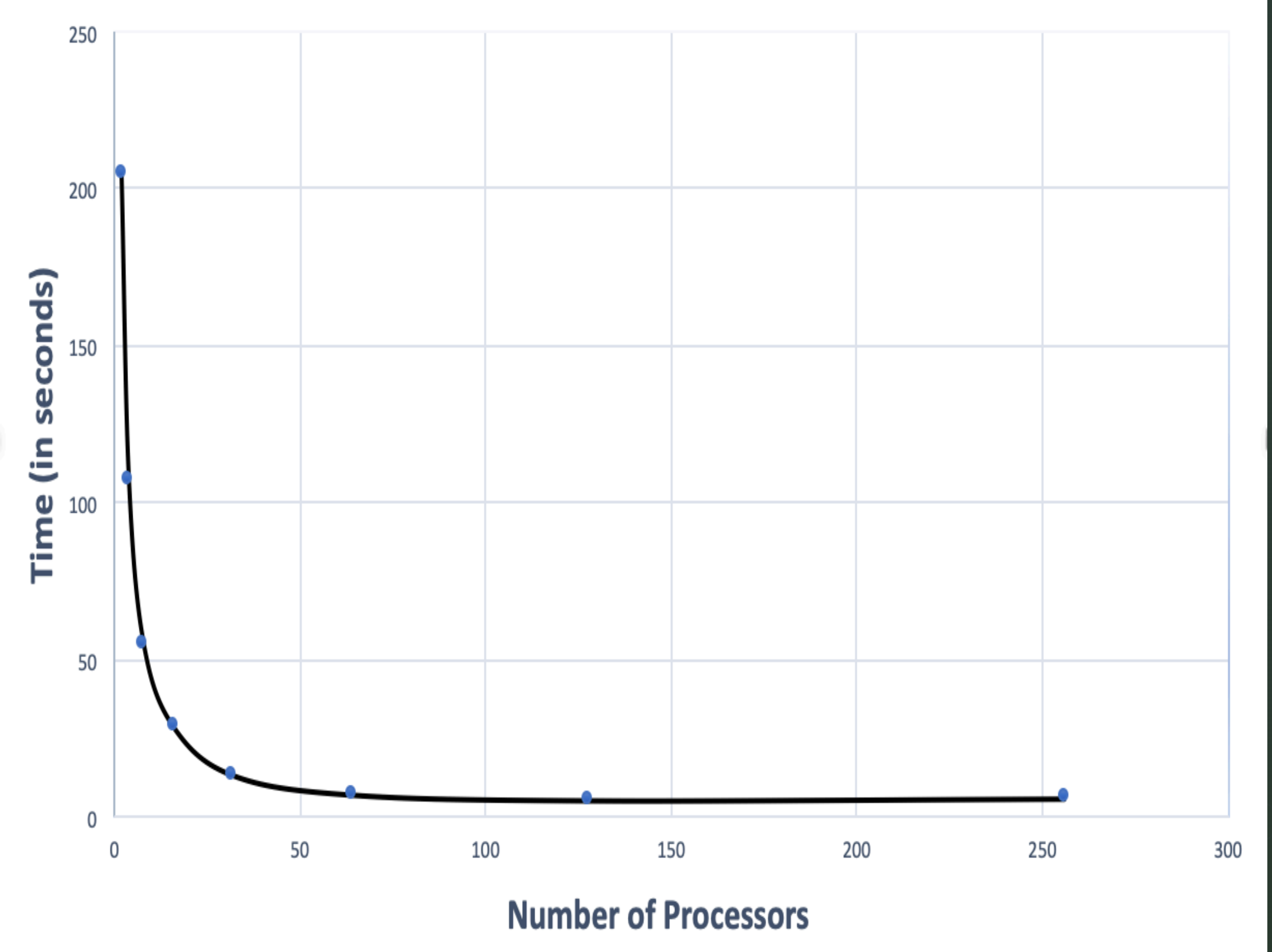
1. Scatter list of words to all processors. Each processor is responsible for collecting data corresponding to it’s local word corpus.
2. Perform data pre-processing and cleaning tasks.
3. Map phase - Emit (Store) all words as keys and values as count = 1.
4. Combine phase - Using mapper output, combine all keys and add their corresponding values.
5. Reduce phase – In this phase, all processors have a local word count.
    1. Using Recursive Halving – One processor, in this case, P0 gets one large dictionary with all keys and values.
    2. Using MPI Gather – All processors send their local dictionaries to P0 and P0 combines them.

Parallel Execution Results

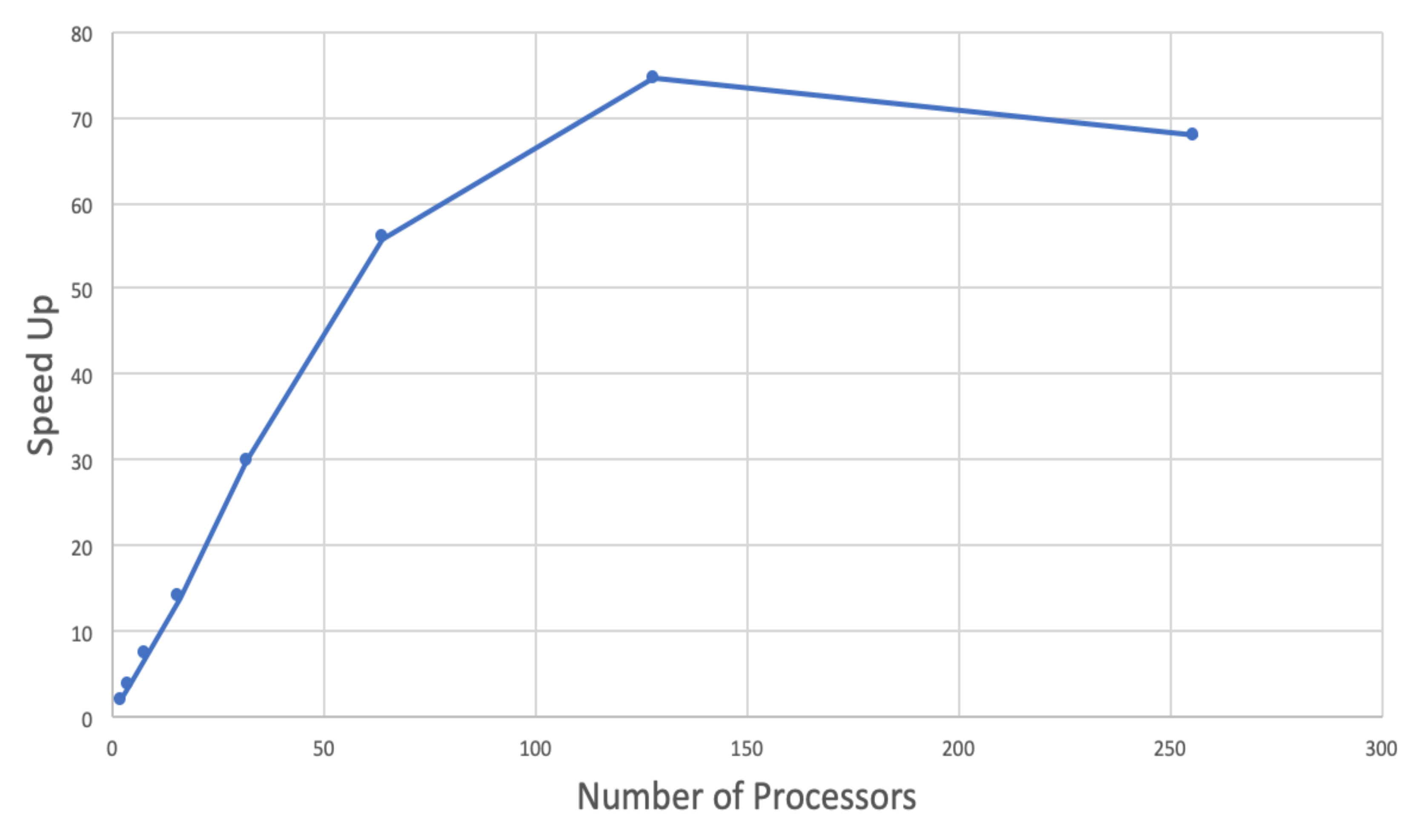
Evaluating Amdahl’s Law

Data size = 138 MB
Number of words = 2,11,74,415

No. of Processors	Time (in seconds)
2	205.1
4	107.57
8	55.28
16	28.92
32	13.44
64	7.21
128	5.4
256	5.94



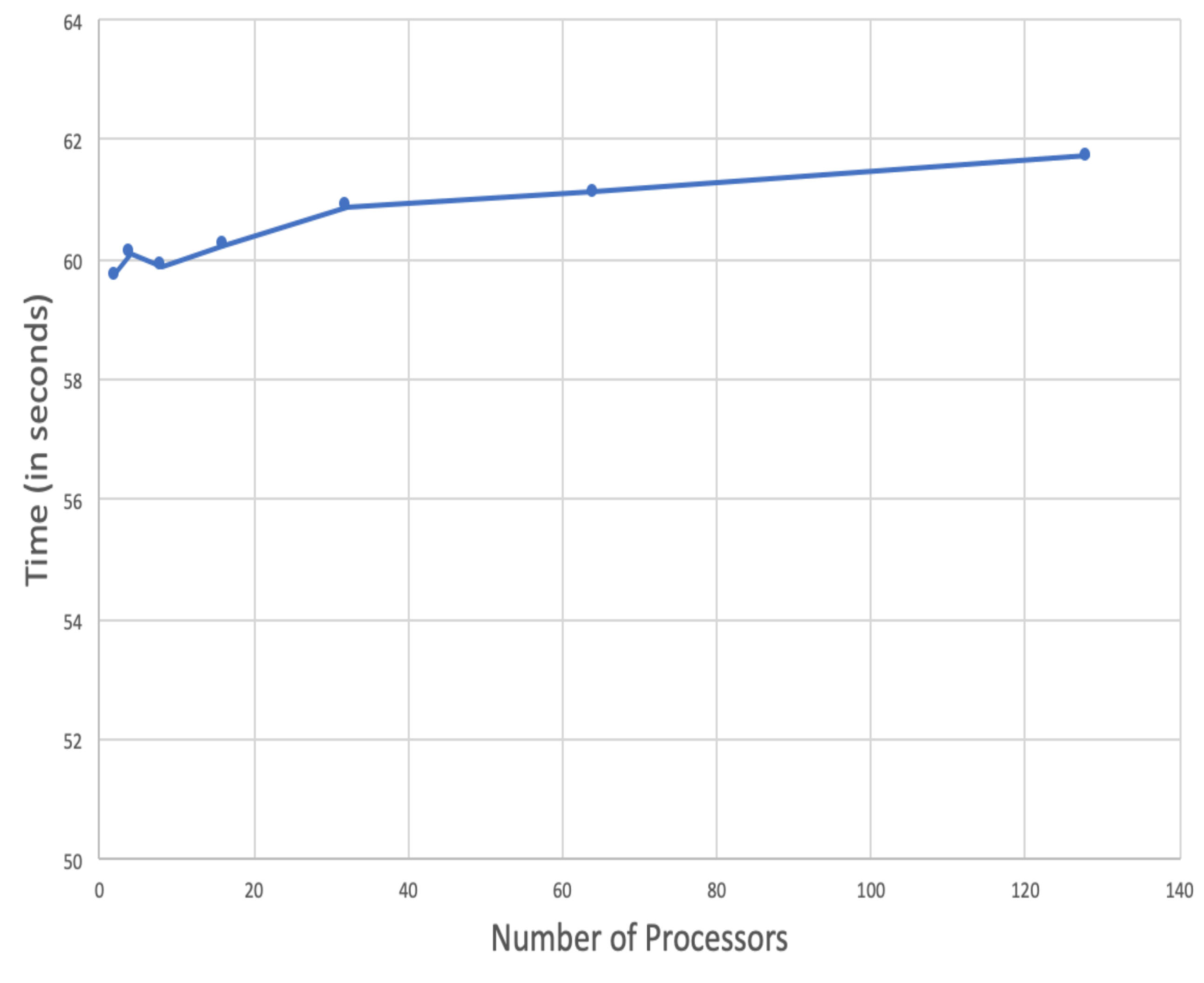
Speed up


Speed Up
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
Number of Processors

Evaluating Gustafson’s Law

Fixed Data per Processor = 20 MB

No. of Processors	Time (in seconds)
2	59.74
4	60.1
8	59.89
16	60.23
32	60.87
64	61.12
128	61.72



Observations

- Speedup was observed significantly up to 64 processors.
- For the data size used, using 64 processors is optimum.
- There was a slight increase in the processing time for 256 processors, indicating increase in communication time.
- When we have fixed data per processor, slight increase in running time is observed as we increase the number of processors since the cost of communication increases.

References

- Algorithms Sequential & Parallel: A Unified Approach (Dr. Russ Miller, Dr.Laurence Boxer)
- https://netjs.blogspot.com/2018/02/word-count-mapreduce-program-in-hadoop.html
- https://mpi4py.readthedocs.io/en/stable/intro.html
- https://ubccr.freshdesk.com/support/solutions

Thank you.