MapReduce

Vraj Patel • Kishen Gowda • Rushil Shah • Saumitra Sharma • Mrinal Anand

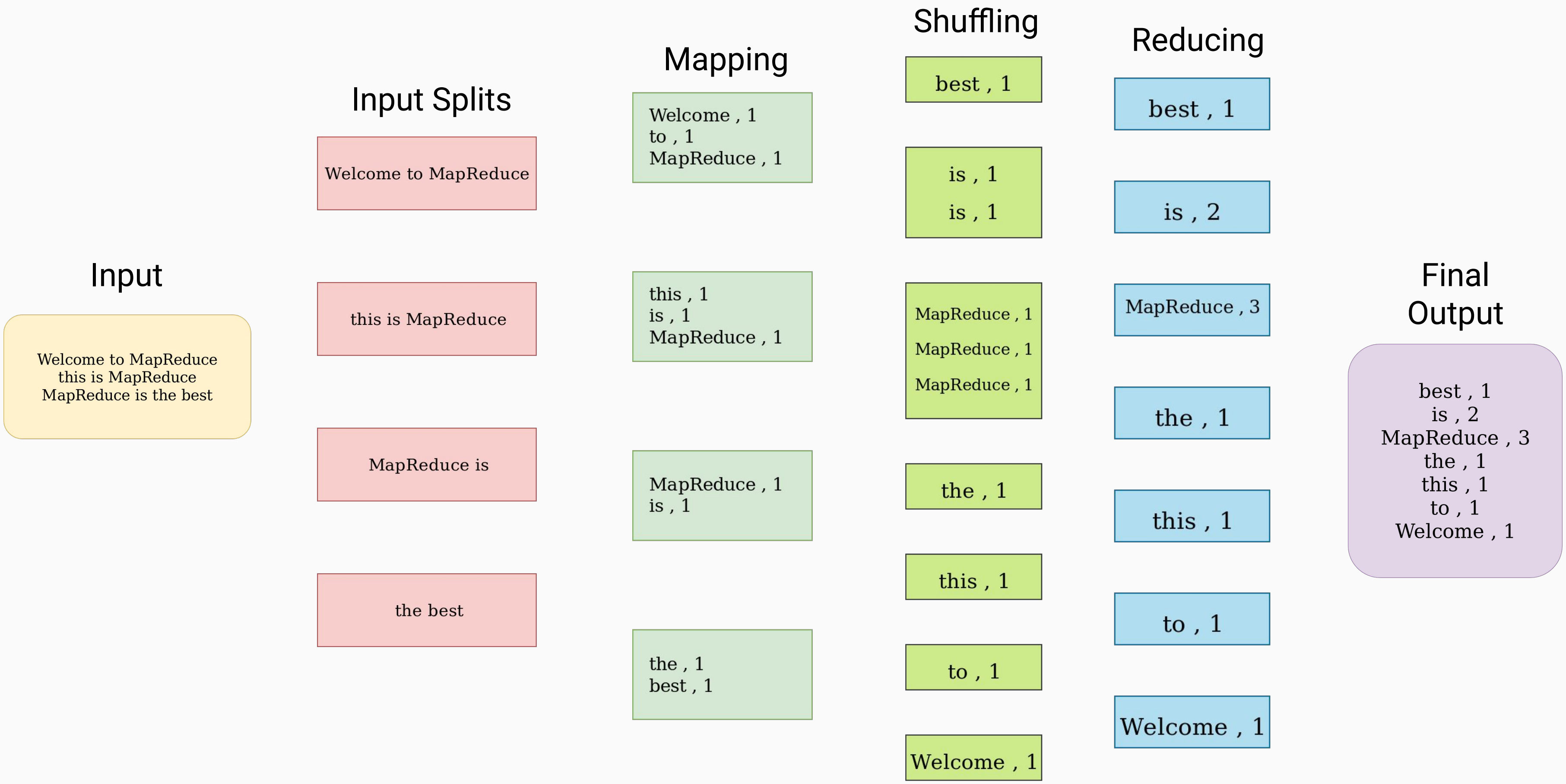
What is MapReduce and How does it work?

- It is a programming model suitable for processing huge data parallelly.
- It works in two phases:
    - Map Phase
    - Reduce Phase
- Input to each of the above phase are key-value pairs.
- The four steps of execution: splitting, mapping, shuffling, and reducing.

Now,

An example . . .

SAY "WORD COUNT"
ONE MORE TIME...
imgflip.com



Input

Welcome to MapReduce
this is MapReduce
MapReduce is the best

Input Splits

Welcome to MapReduce

this is MapReduce

MapReduce is

the best

Mapping

Welcome , 1
to , 1
MapReduce , 1

this , 1
is , 1
MapReduce , 1

MapReduce , 1
is , 1

the , 1
best , 1

Shuffling

best , 1

is , 1
is , 1

MapReduce , 1
MapReduce , 1
MapReduce , 1

the , 1

this , 1

to , 1

Welcome , 1

Reducing

best , 1

is , 2

MapReduce , 3

the , 1

this , 1

to , 1

Welcome , 1

Final Output

best , 1
is , 2
MapReduce , 3
the , 1
this , 1
to , 1
Welcome , 1

(key, value) ?

(“word”, count)

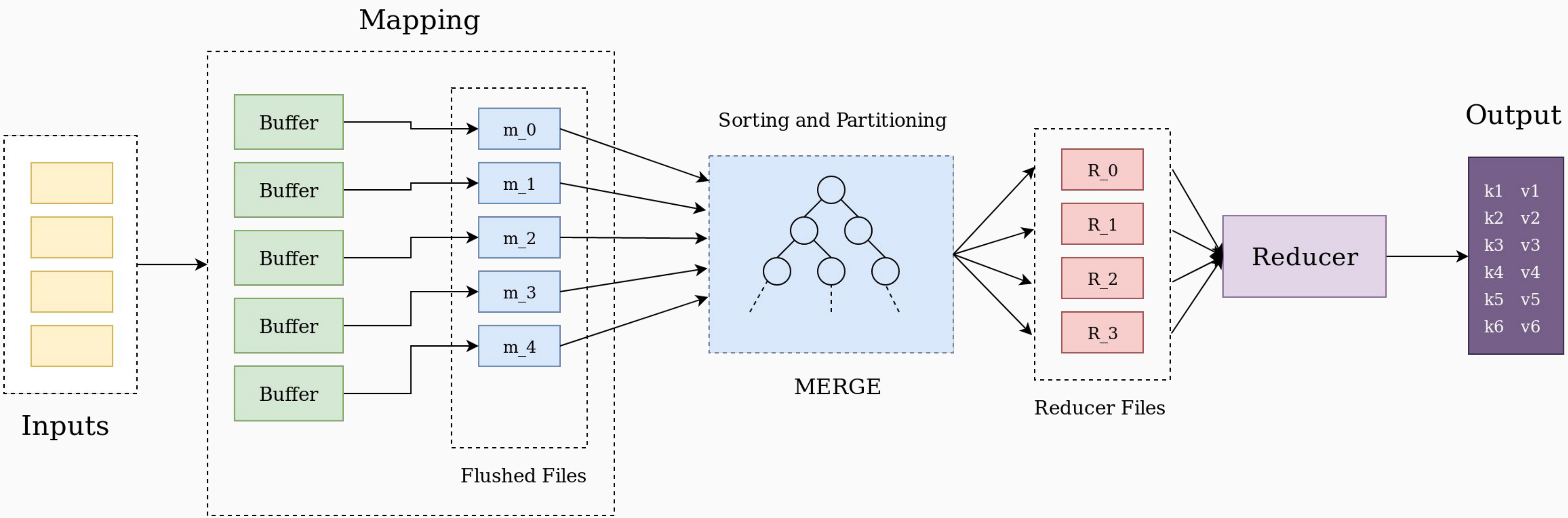
(“MapReduce”, 1)

(“MapReduce”, 1)

(“MapReduce”, 1)

(“MapReduce”, 3)

Architecture



Mapping

- Every mapper thread has a corresponding buffer and set of files it will map
- When MR_EMIT is called -> (key, value) stored in buffer
- When buffer is full -> sort and flush

Mapping - Scheduling Policies

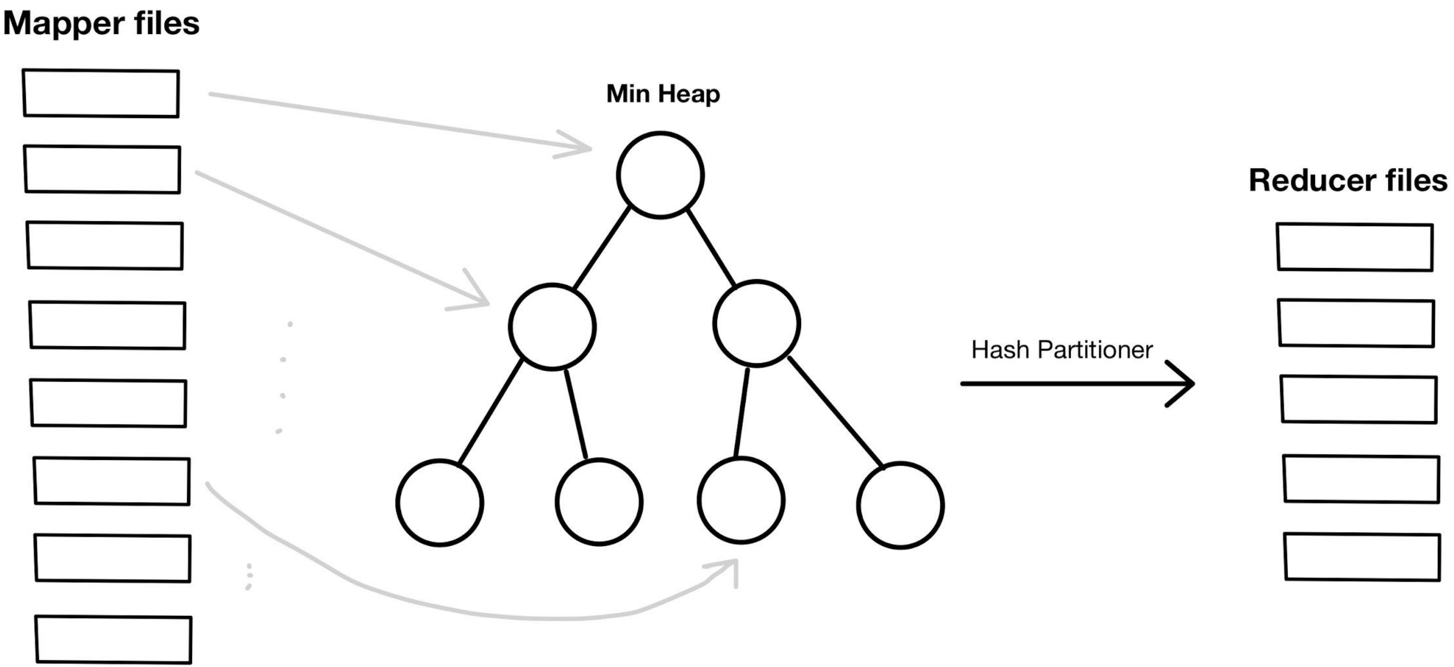
- Allocate as per given order in Round Robin fashion
- Allocate as per given order in Sorted Double Round Robin fashion
- Makespan Minimization - Longest Processing time (LPT) [4/3 Approx.]

Performance on Uneven workload:

Scheduling Policy	**Average time (in s)**
Round Robin	88
Sorted Double Round Robin	86
LPT	76

Sorting And Partitioning

- External Sorting Algorithm using Min-Heap
- Partition the (key, value) pair based on user-defined Partition function to corresponding reducer files

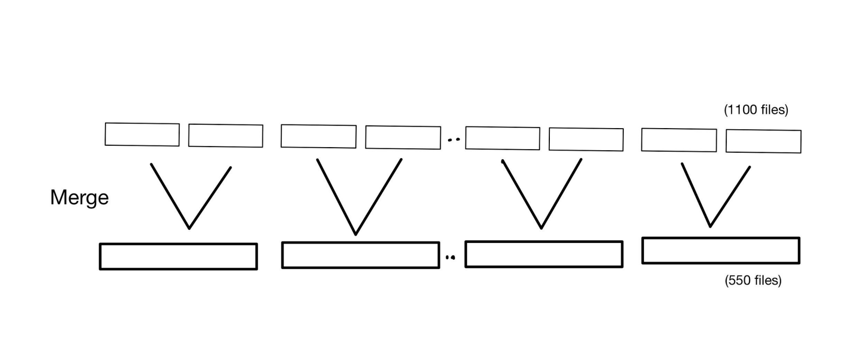

Mapper files
Min Heap
Hash Partitioner
Reducer files

Complexity Analysis

- Sorting and Partitioning
    - No. of chunks = **n/b**
    - Time for heap construction = **O(n/b)**
    - Time to find minimum= **O(1)**
    - Time for insertion = **O(log(n/b))**
    - Total time required = **O(n*log(n/b))**

Sorting and Partition - Problem

- Limit on number of open files per process:
  - Linux : 1024
  - Windows : 512
- Solve by merging pair of files till number of files become less than the limit
- Conduct the merging concurrently, use semaphores to limit the number of open files while merging



Benchmarks

- Word Count
- Mutual Friends
- Matrix Multiplication

Matrix Multiplication

Matrix

(m,n)

Does not fit
in the main memory

Vector

$(n,1)$

Fits in the
main memory

(key, value) ?

(i, vec_j * m_{ij}
)

$$(i, vec_j * m_{ij})$$

The i^{th} row of the matrix **m**

$$(i, vec_j * m_{ij})$$

The j^{th} element of i^{th} row of the matrix **m**

$$\left(i,\ vec_j * m_{ij} \right)$$

The jth element of vector **v**

But what happens after reduction?

- All the values with same value of **i,** i.e., with same row number, collect together and add up.

- That’s exactly what we want!

Mutual Friends

(key, value) ?

0 : [1, 4, 5]
1 : [0, 3, 5]

01 : [1, 4, 5]
04 : [1, 4, 5]
05 : [1, 4, 5]

01 : [0, 3, 5]
13 : [0, 3, 5]
15 : [0, 3, 5]

Keys

01 : [1, 4, 5]

10 : [0, 3, 5]

Keys

01 : [1, 4, 5]

01~~10~~ : [0, 3, 5]

We only consider them in
the sorted order

Values

01 : [1, 4, 5]

01 : [0, 3, 5]

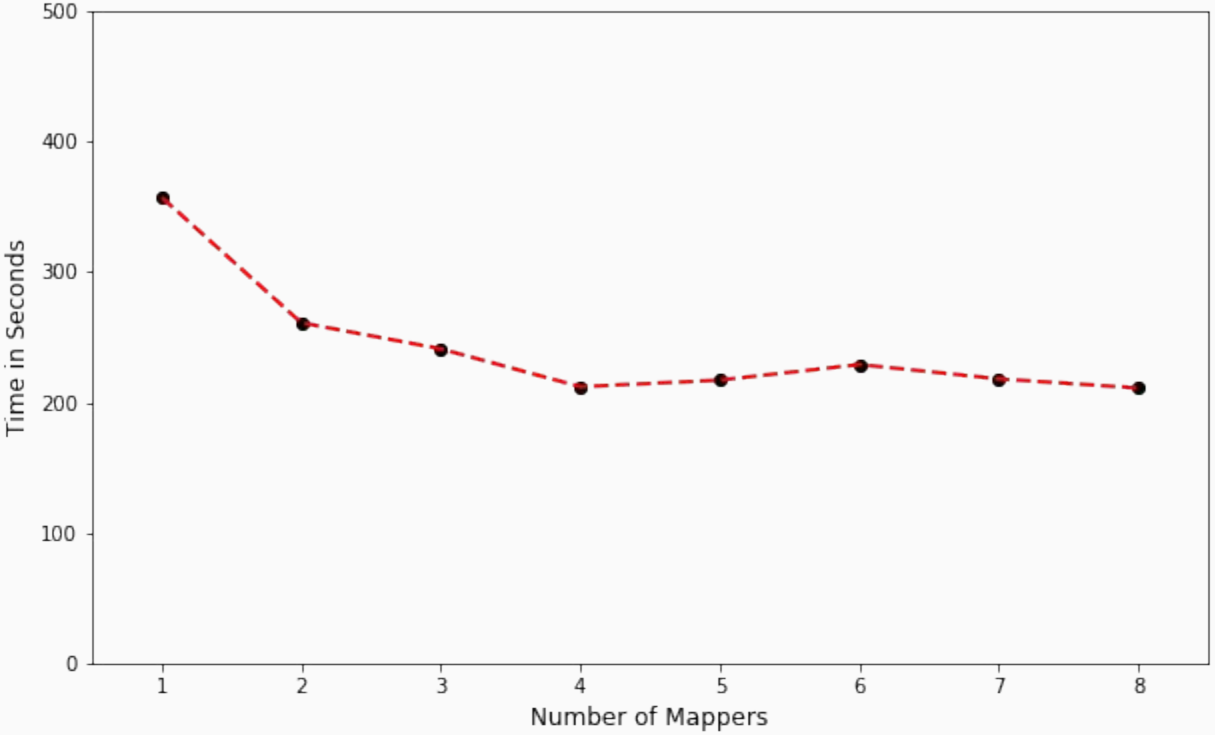
Mutual Friend

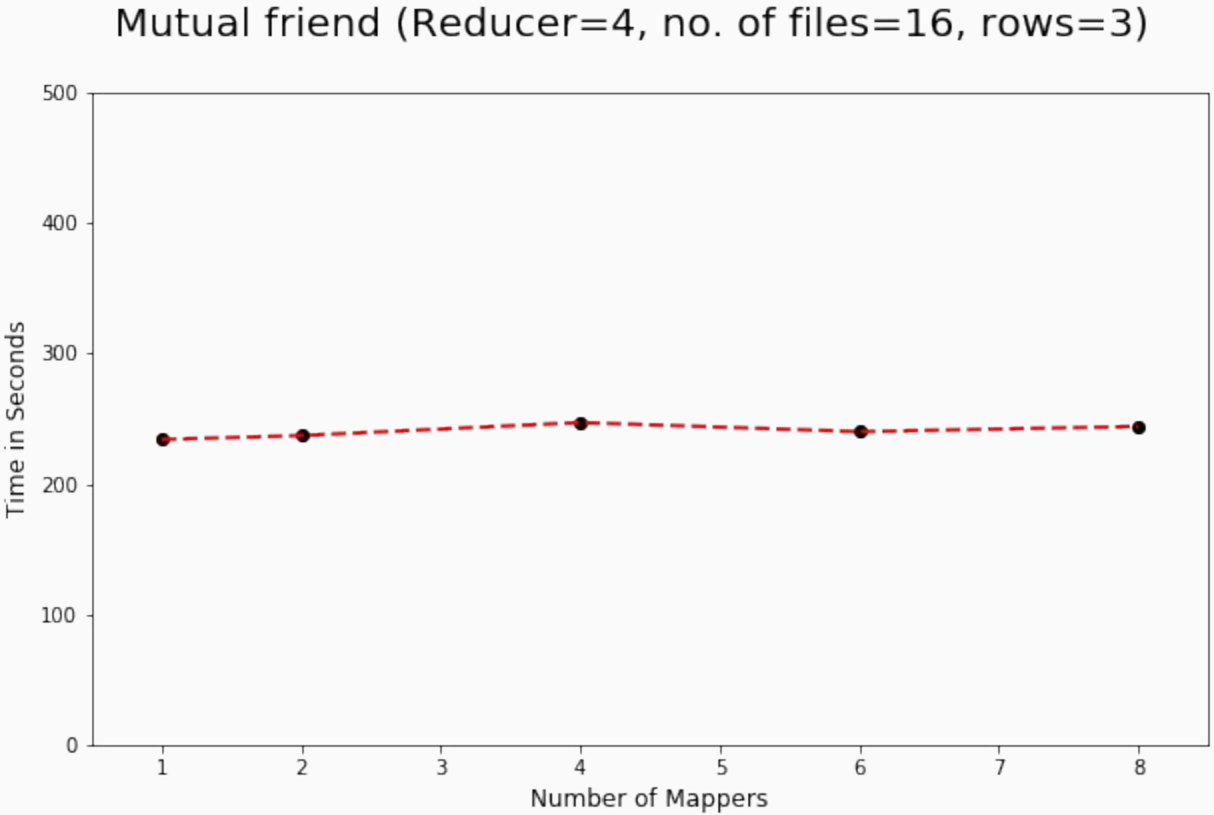
01 : [1, 4, 5]

01 : [0, 3, 5]

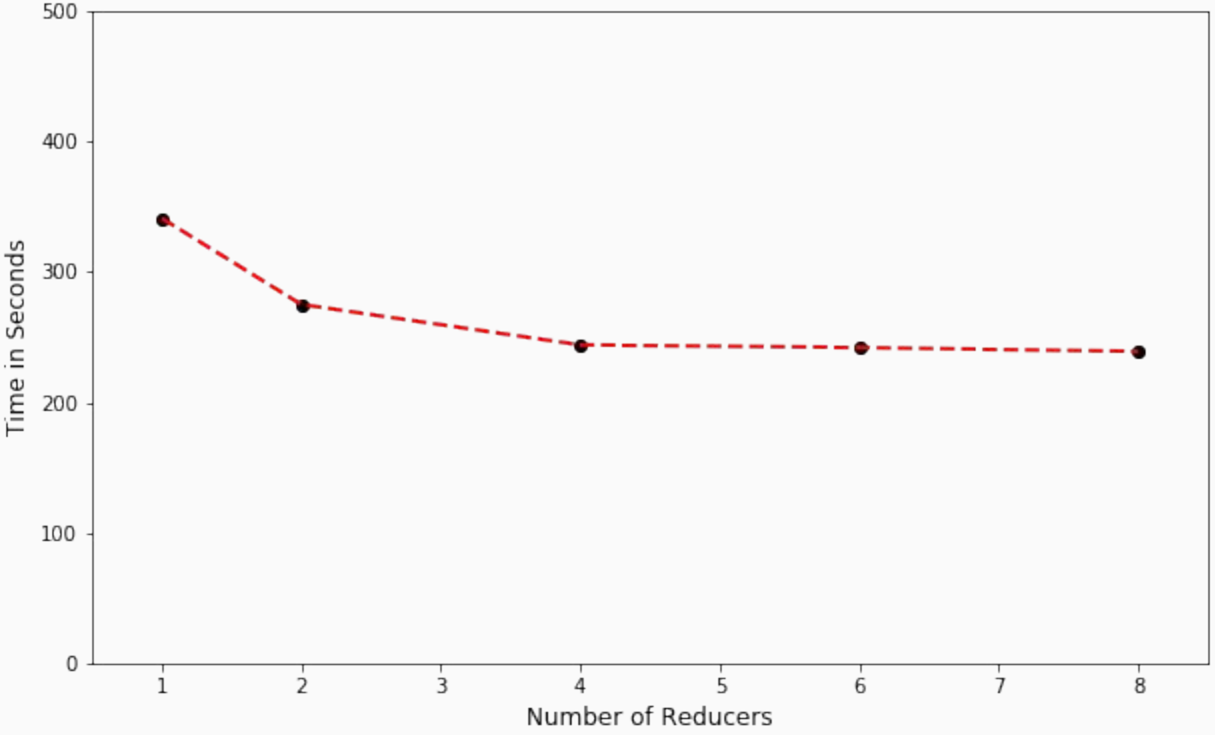
Evaluation of MapReduce on Benchmarks

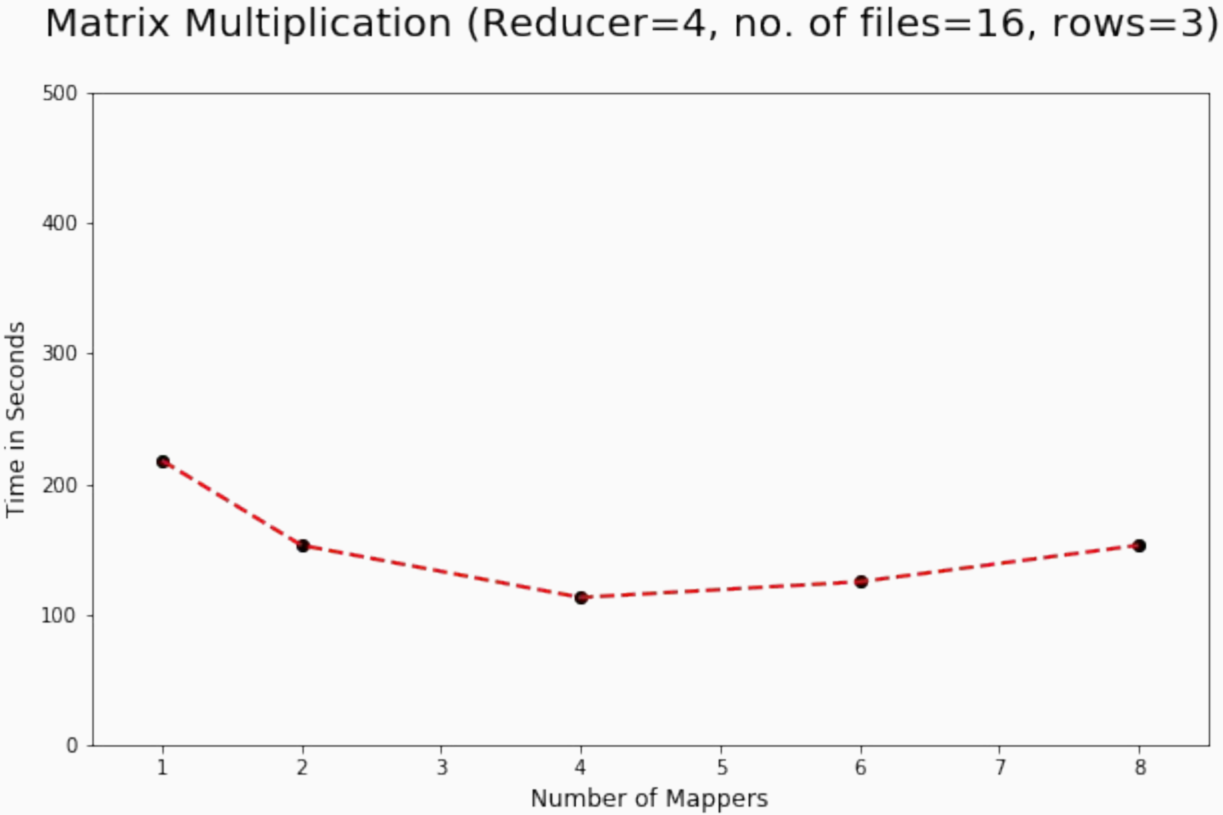
Word-Count (Reducer = 8, no. of files=16, rows=3)



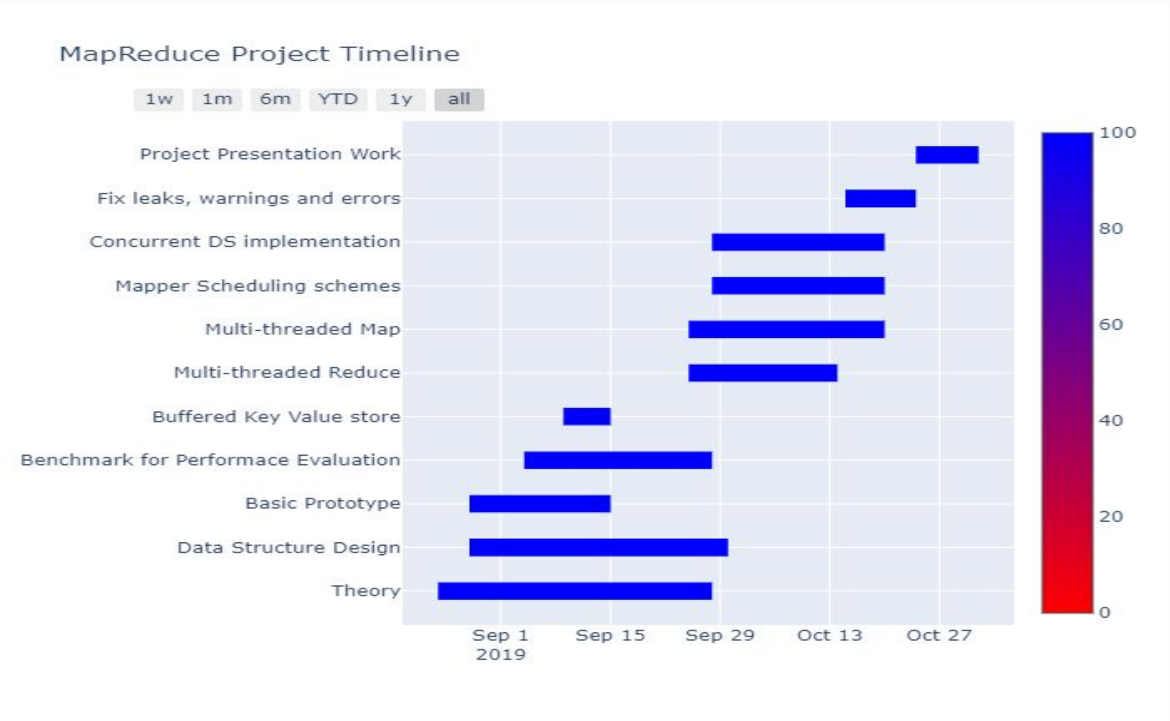

Mutual friend (Reducer=4, no. of files=16, rows=3)
Time in Seconds
0
100
200
300
400
500
Number of Mappers
1
2
3
4
5
6
7
8

Mutual friend (Mapper=8, no. of files=16, rows=3)




Matrix Multiplication (Reducer=4, no. of files=16, rows=3)
Time in Seconds
500
400
300
200
100
0
1
2
3
4
5
6
7
8
Number of Mappers

Gantt Chart


MapReduce Project Timeline
1w
1m
6m
YTD
1y
all
Project Presentation Work
Fix leaks, warnings and errors
Concurrent DS implementation
Mapper Scheduling schemes
Multi-threaded Map
Multi-threaded Reduce
Buffered Key Value store
Benchmark for Performace Evaluation
Basic Prototype
Data Structure Design
Theory
Sep 1
2019
Sep 15
Sep 29
Oct 13
Oct 27
100
80
60
40
20
0

Work Division

Vraj	Mapreduce.h -> Multithreaded Mapper, quicksort, Model
Kishen	Sorter.h -> External Sort, Files Compressor, Scheduling policies
Mrinal	Mapreduce.h -> Multithreaded Reducer, Metrics, Plots
Saumitra	Complete Word Count Benchmark, Matrix Multiplication Benchmark
Rushil	Complete Mutual Friends Benchmark (3 iterations), Model

References

1. Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.
2. https://github.com/remzi-arpacidusseau/ostep-projects/tree/master/concurrency-mapreduce
3. http://pages.cs.wisc.edu/~remzi/OSTEP/Educators-Slides/Andrea/lecture24-mapreduce.pdf
4. Operating Systems: Three Easy Pieces, Remzi H. Arpaci-Dusseau, and Andrea C. Arpaci-Dusseau, Arpaci-Dusseau Books, August 2018 (Version 1.00)
5. http://stevekrenzel.com/finding-friends-with-mapreduce
6. https://userweb.ucs.louisiana.edu/~vvr3254/CMPS598/Notes/Matrix-Vector%20Multiplication%20by%20MapReduce-v2.pdf