

Enhancing MapReduce using MPI and an optimized data exchange policy

Hisham Mohamed and Stéphane Marchand-Maillet

Viper group, CVML Laboratory, University of Geneva
September 10, 2012



Fifth International Workshop on Parallel Programming Models and
Systems Software for High-End Computing (P2S2), 2012



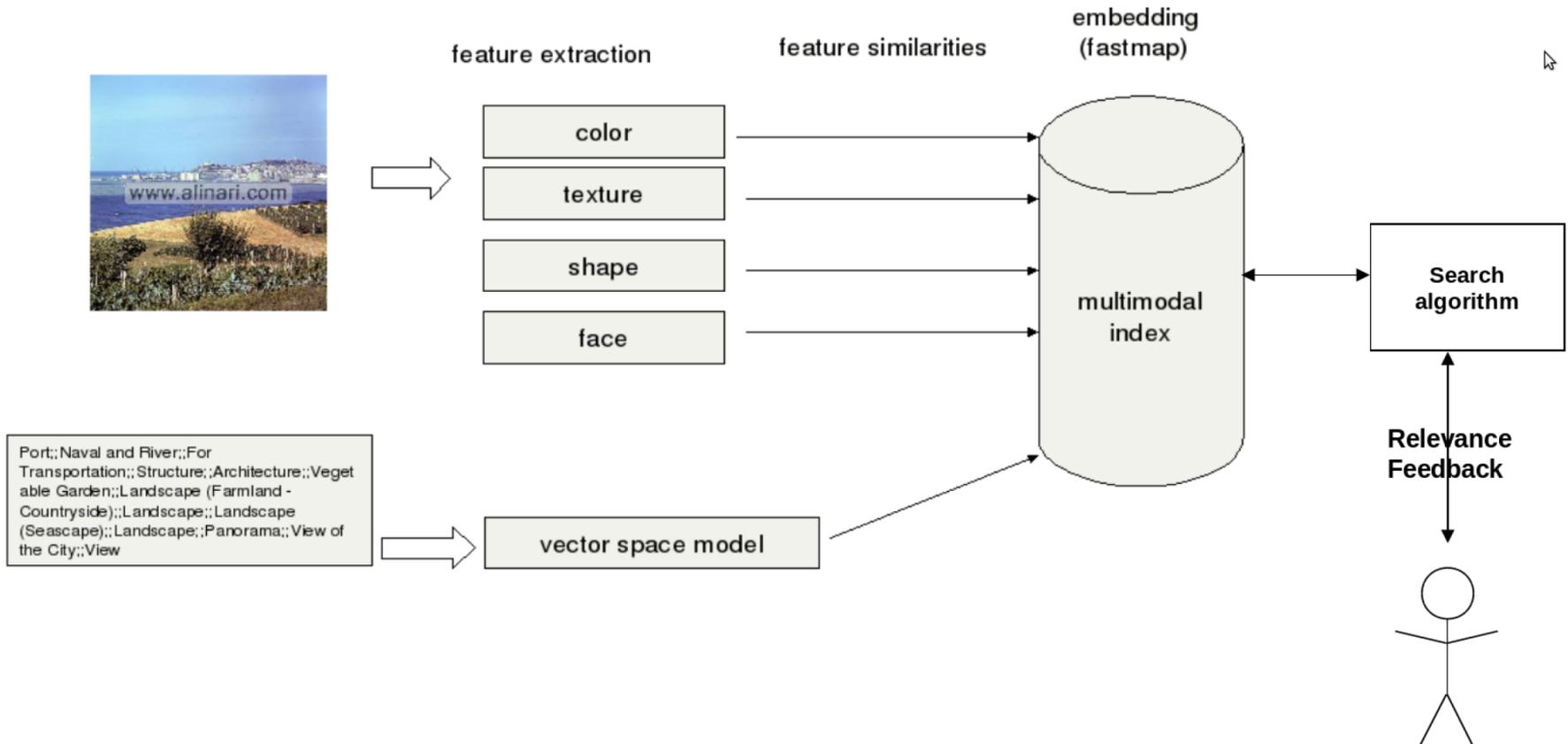
**UNIVERSITÉ
DE GENÈVE**

Outline

- Motivation
- MapReduce
- MapReduce overlapping using MPI (MRO-MPI)
- Experiments
 - Wordcount
 - Distributed inverted files.
- Conclusion

Motivation

- Cross Modal Search Engine (CMSE)



Motivation

- Scalability
 - Multimedia Data increases rapidly.
 - Indexing
 - Searching
 - High dimensional data.

Our proposed solution

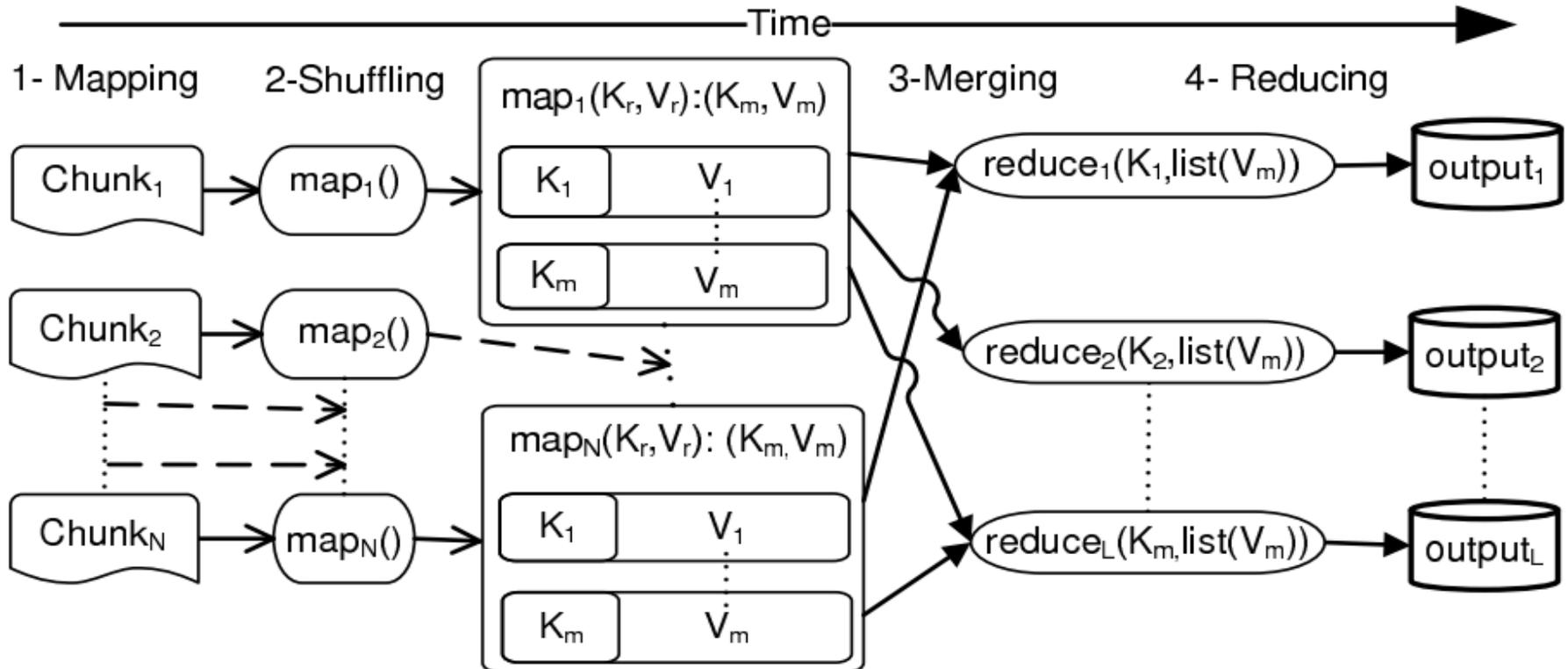
- In CMSE, we need data and algorithm parallelization.
- MapReduce overlapping using MPI (MRO-MPI)
 - C/C++ implementation of MapReduce using MPI.
 - Improving the MapReduce Model.
 - Maintain the usability of the Model.

Outline

- Motivation
- MapReduce
- MapReduce overlapping using MPI (MRO-MPI)
- Experiments
 - Wordcount
 - Distributed inverted files.
- Conclusion

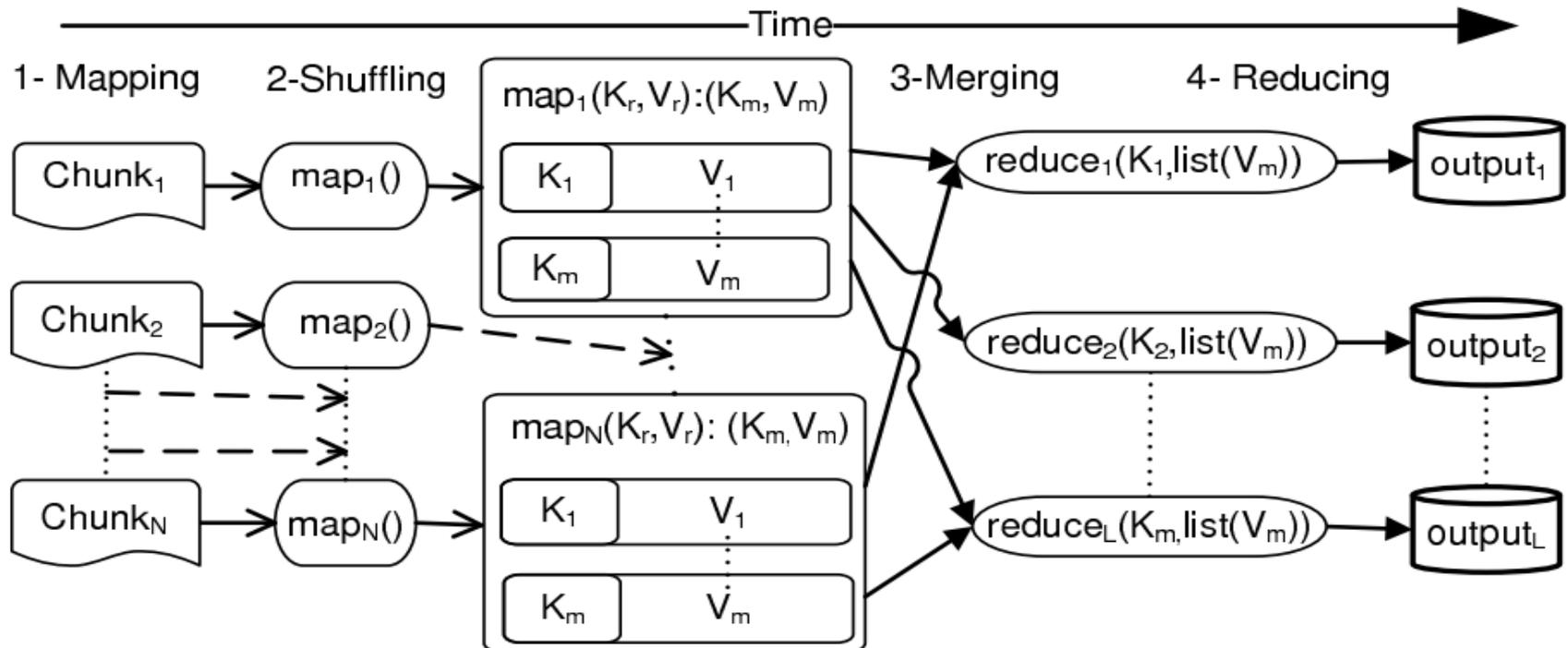
MapReduce

- MapReduce brings a simple and powerful interface for data parallelization, by keeping the user away from the communications and the exchange of data.



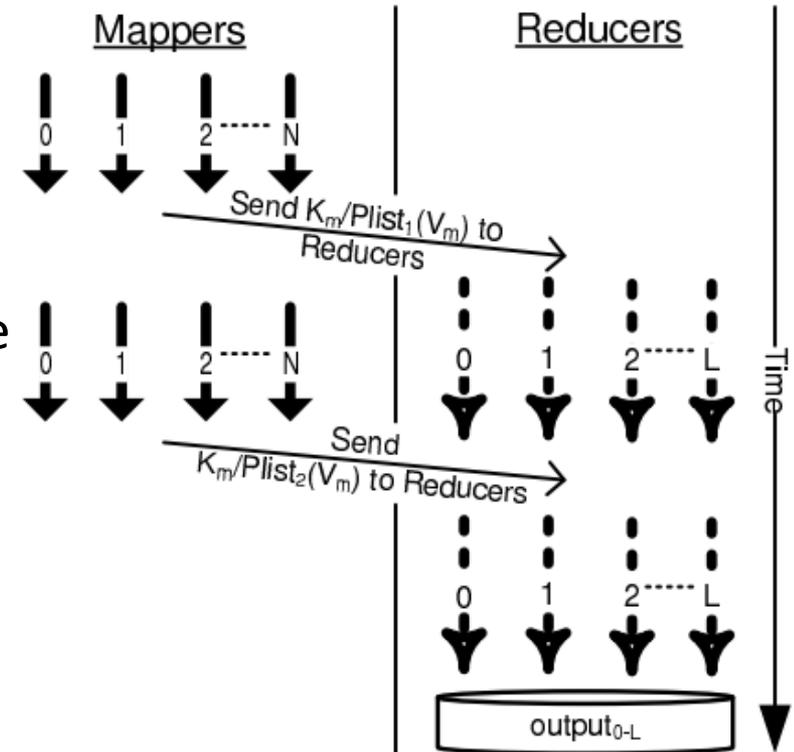
MapReduce

- The current model for MapReduce has at least three bottlenecks:
 - Dependence.
 - Multiple Disk access.
 - All-to-All communication.



MapReduce Overlapping (MRO)

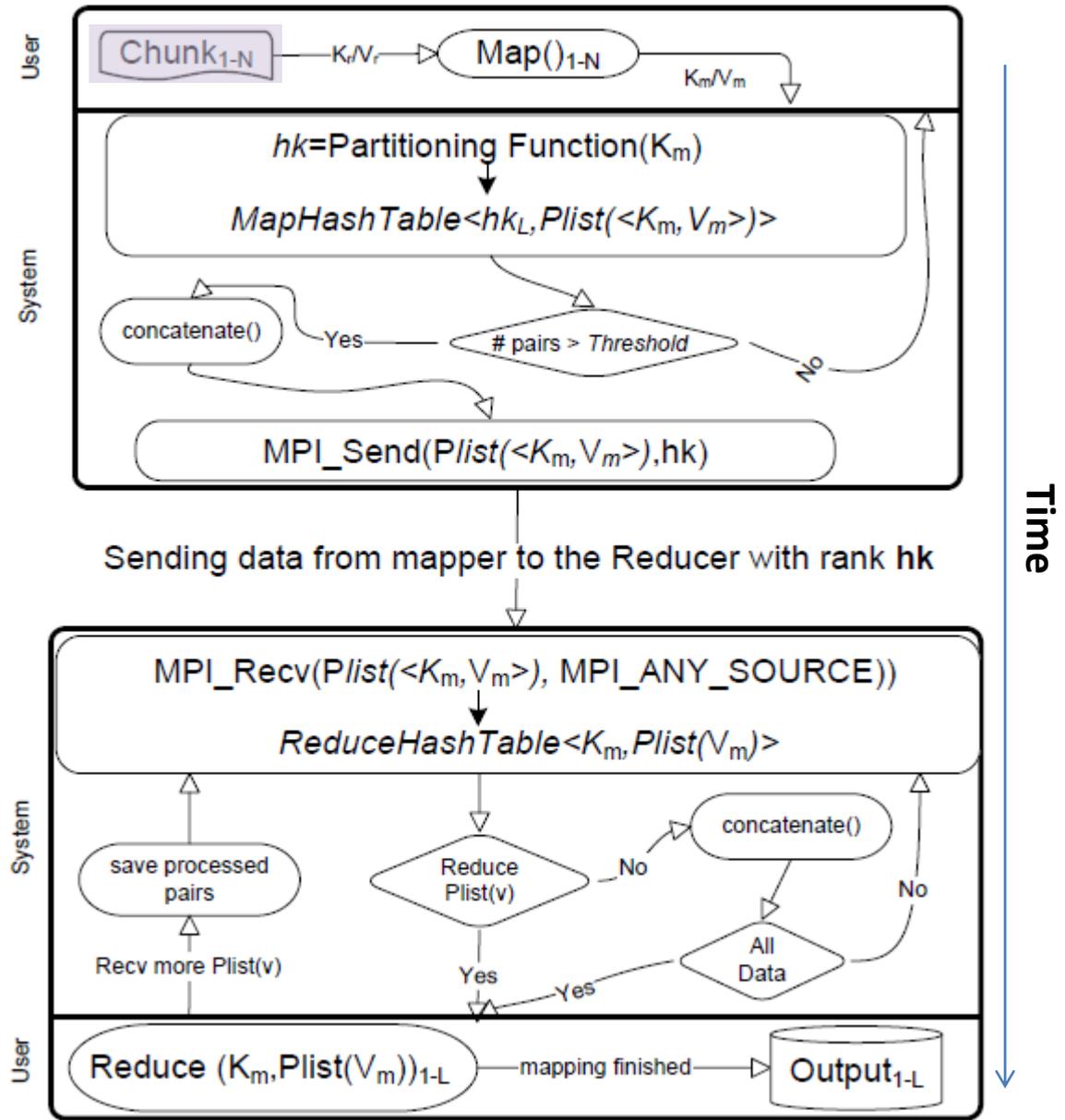
- Send partial intermediate (K_m , V_m) pairs to the responsible reducers.
- We rule out:
 - The multiple read/write.
 - Shuffling phase is merged with the mapping phase.
 - Reducers do not wait until the mappers finish their work.
- Difficulties:
 - Rate of sending data between Mappers and Reducers.
 - The ratio between the Mappers and Reducers



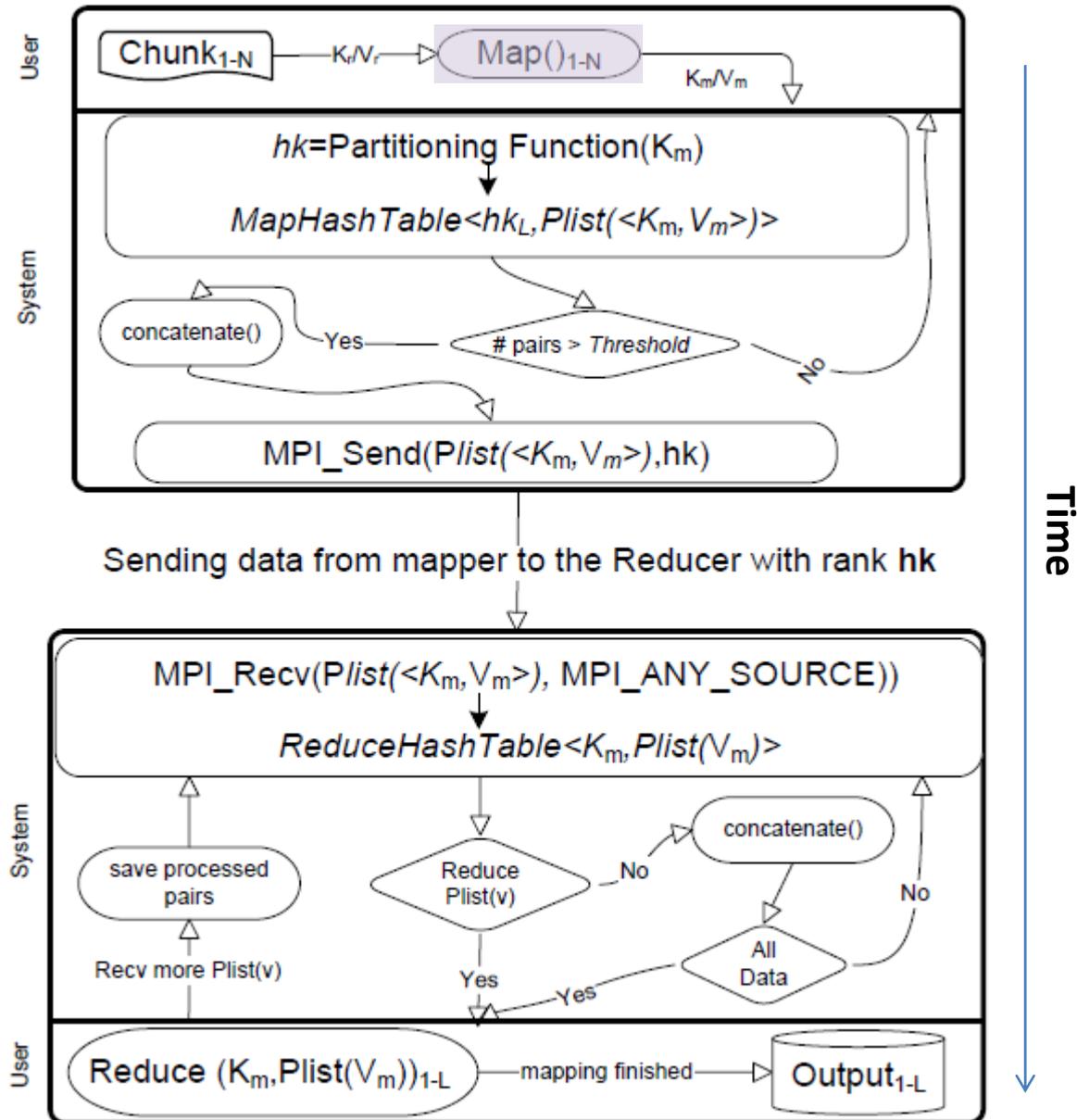
MapReduce Overlapping using MPI (MRO-MPI)

- MapReduce
 - Data parallelization.
- Message Passing Interface (MPI)
 - Separate processes with a unique *rank*.
 - MPI supports point-to-point, one-to-all, all-to-one and all-to-all communications.
 - Communication between processes.
- MapReduce-MPI
 - Based on the original MapReduce Model

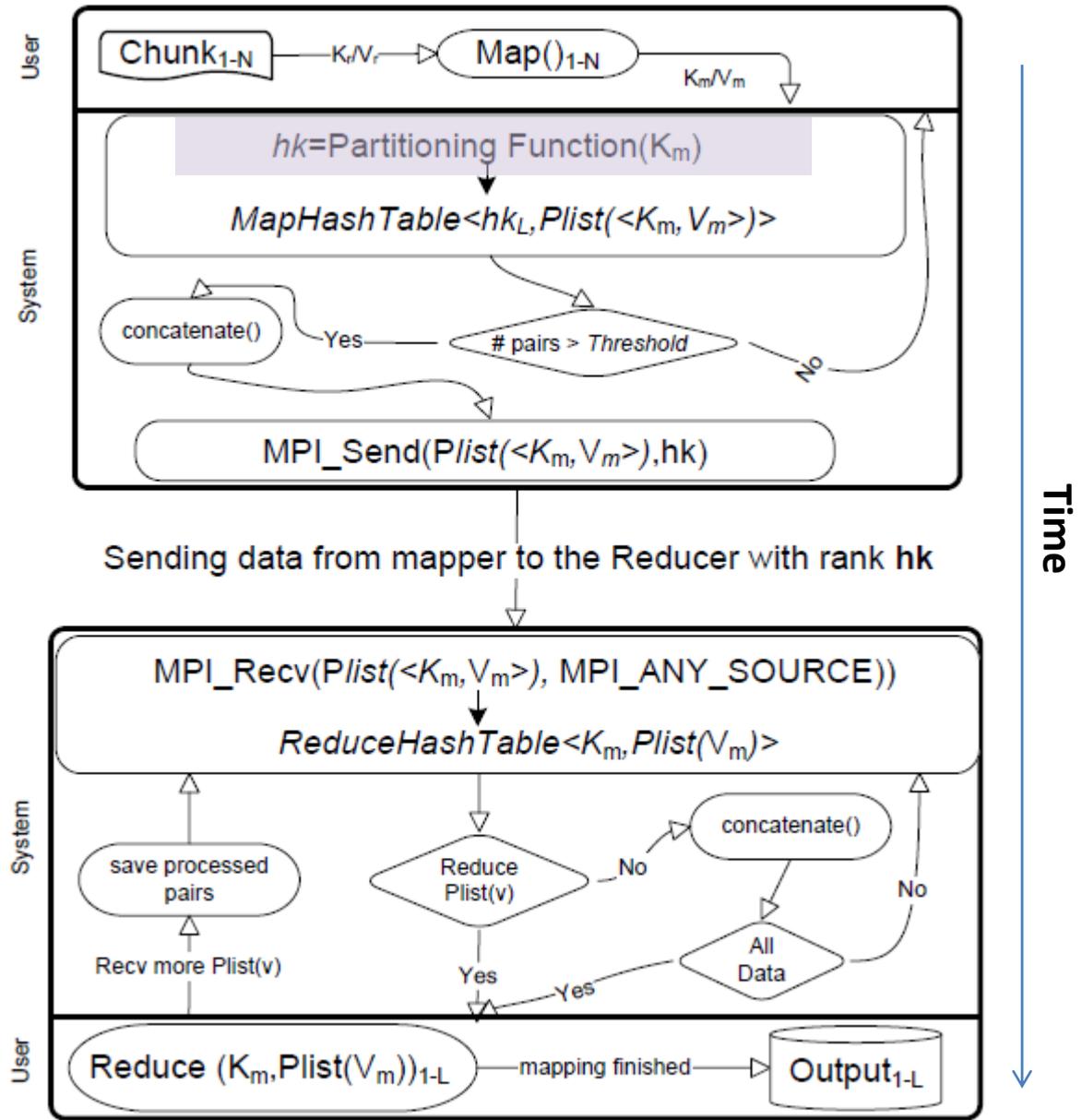
MRO-MPI



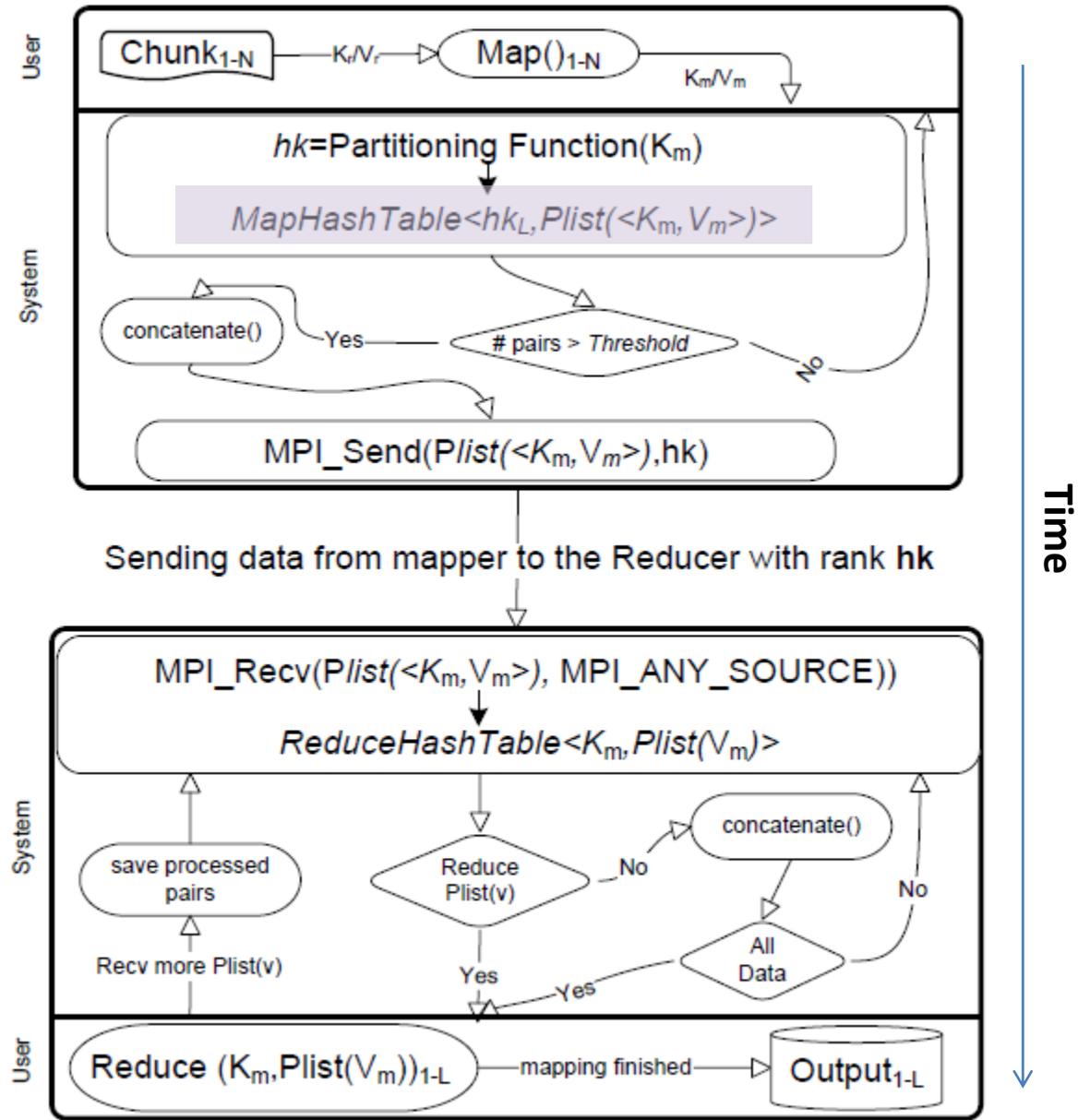
MRO-MPI



MRO-MPI

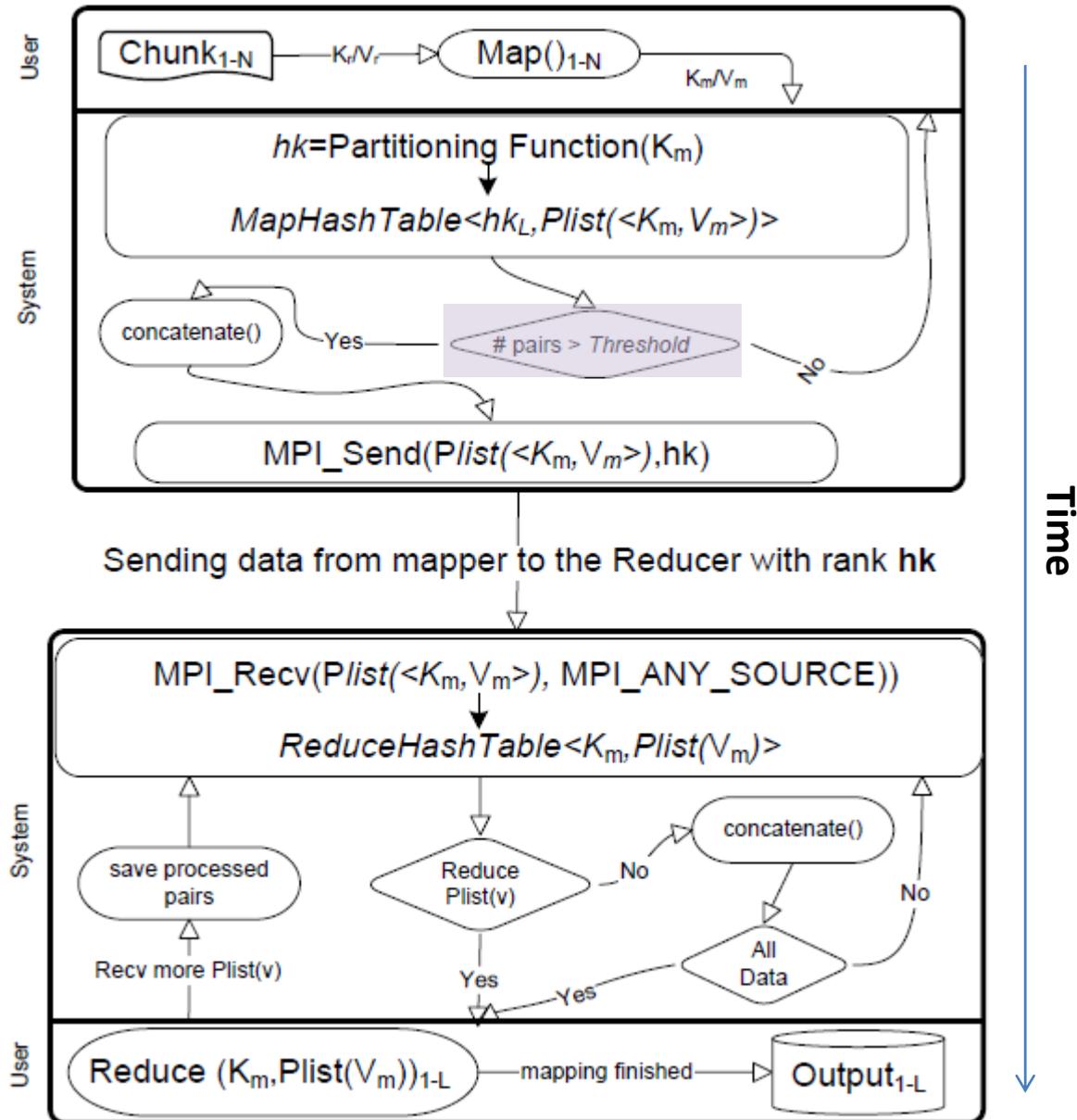


MRO-MPI

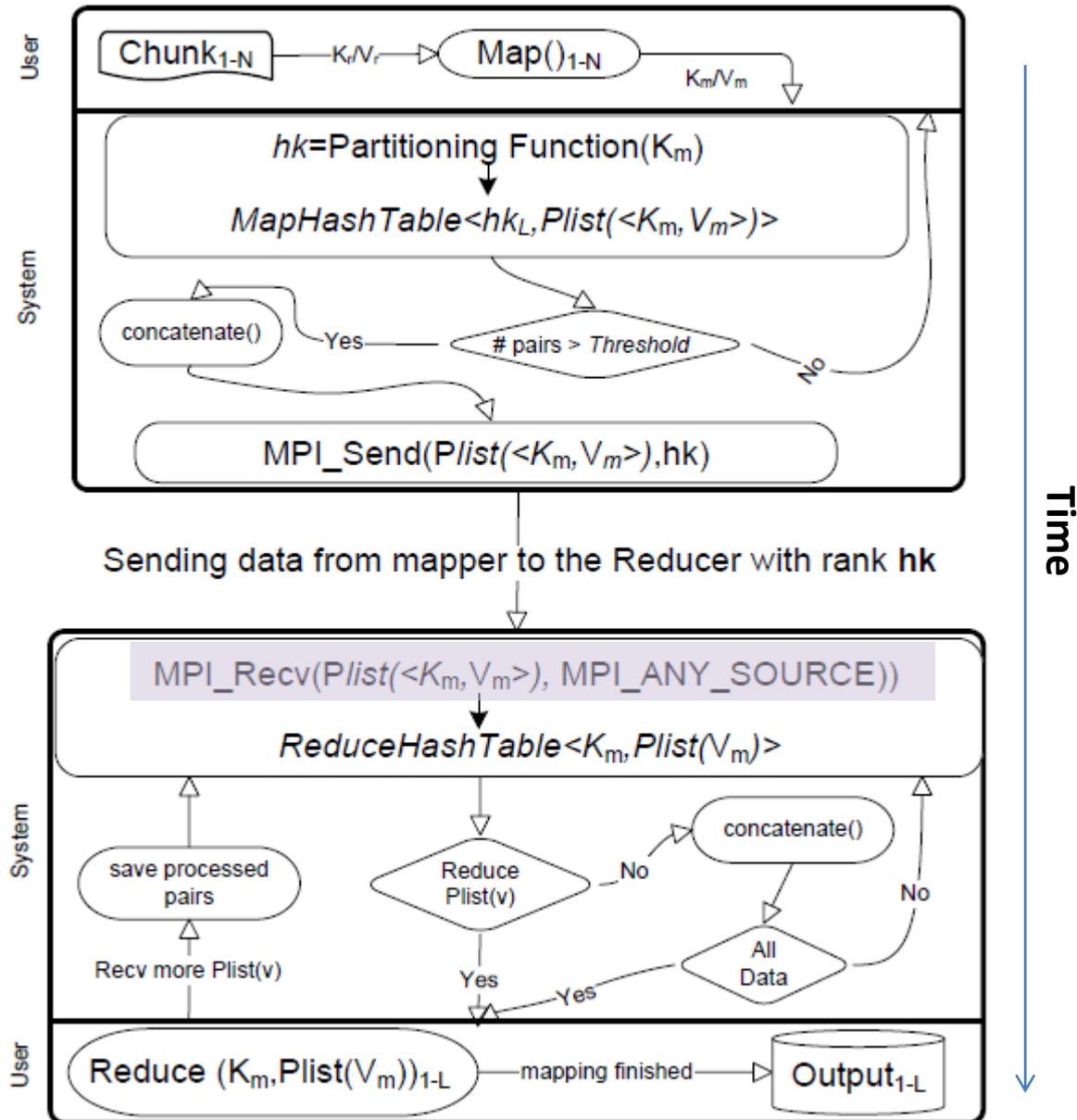


MRO-MPI

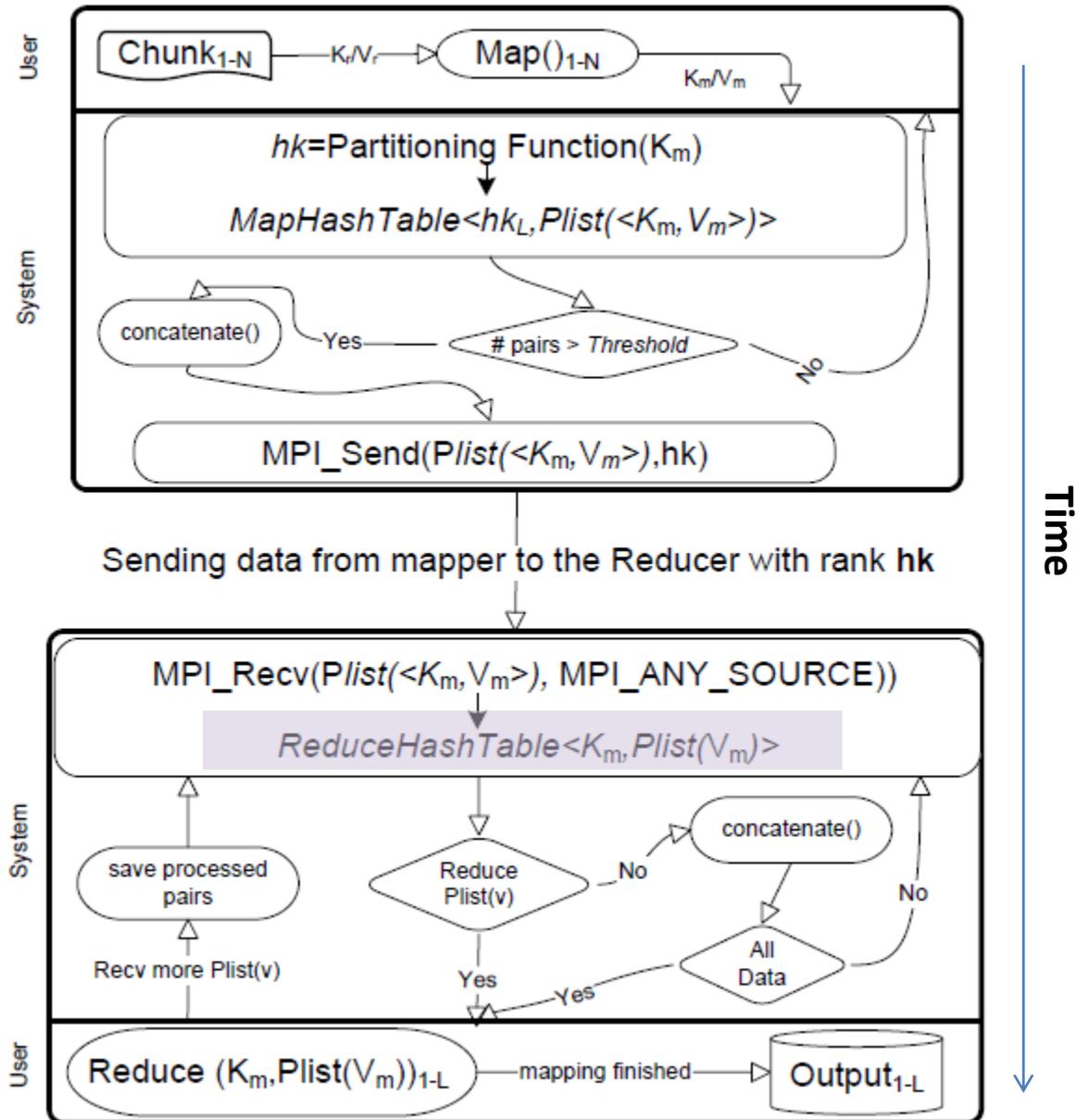
- Rate of Sending the data



MRO-MPI



MRO-MPI

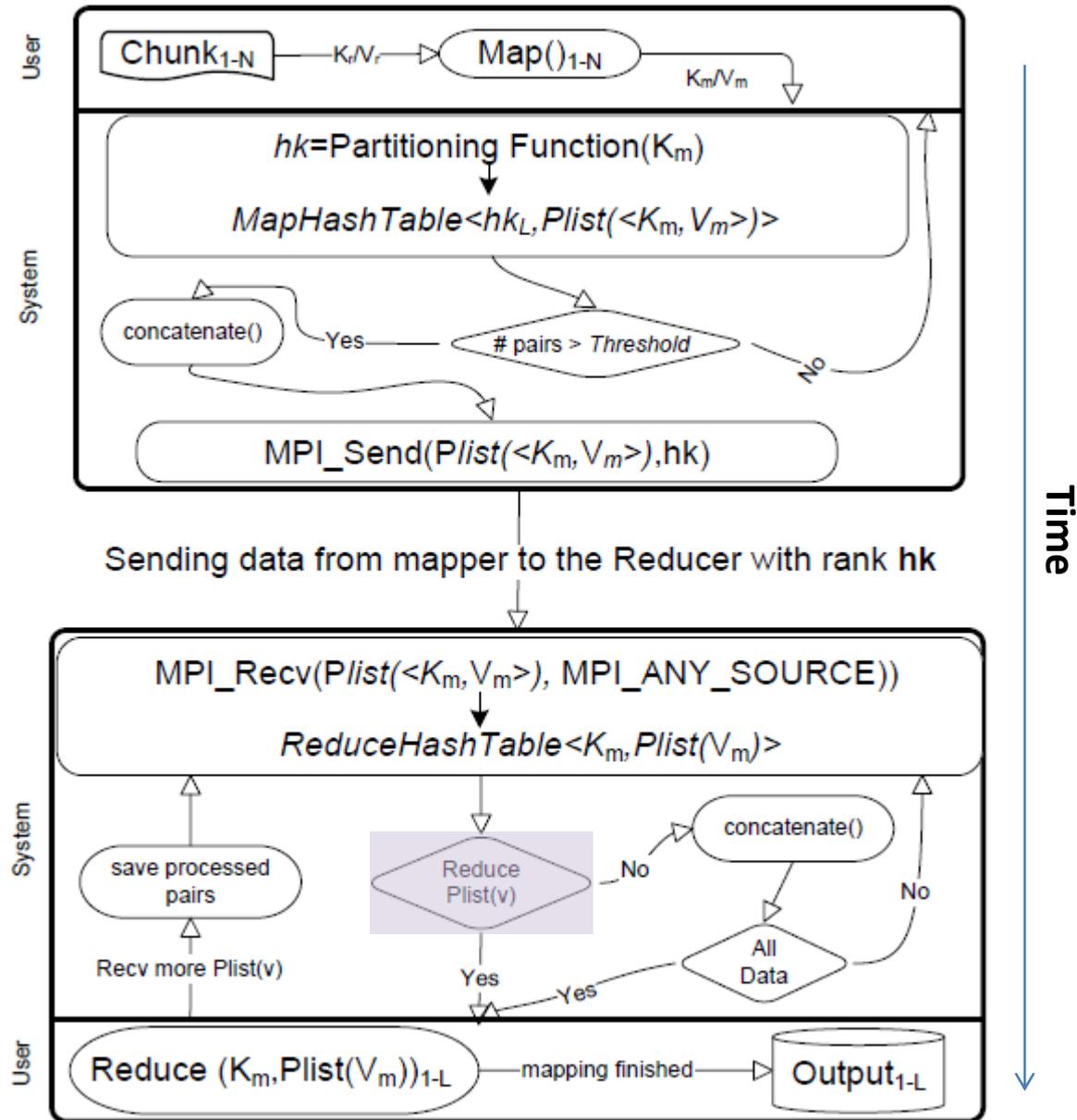


MRO-MPI

– Same simple interface

– Extra parameters:

- Rate of sending data.
- Number of Mappers to Reducers.
- Data type.



Outline

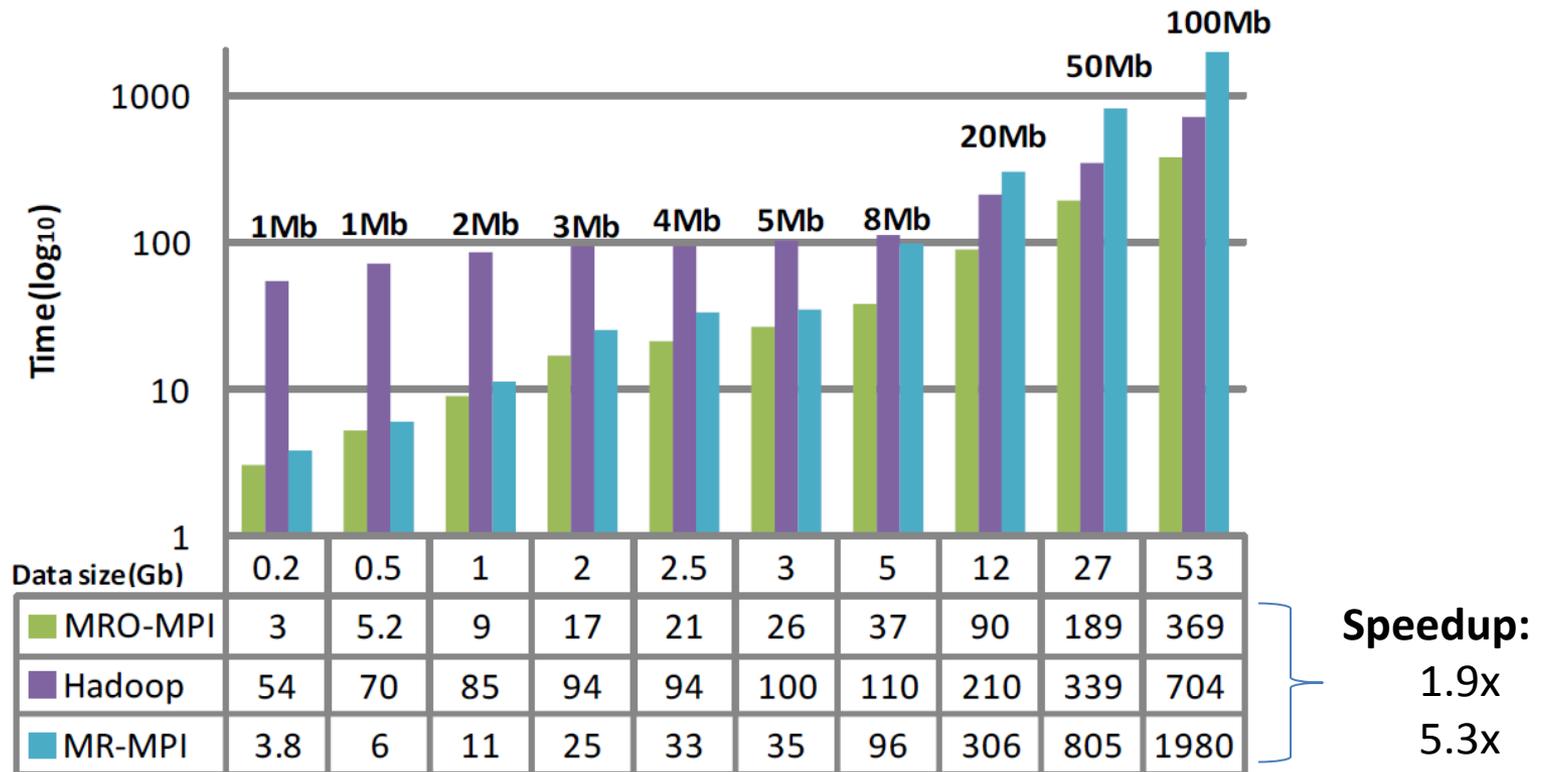
- Motivation
- MapReduce
- MapReduce overlapping using MPI (MRO-MPI)
- Experiments
 - Wordcount
 - Distributed inverted files.
- Conclusion

WordCount

- WordCount:
 - Reads text files and counts how often words occur.
 - Input data size varies from 0.2Gb to 53Gb from project Gutenberg.

WordCount

- MRO-MPI: 24 as mappers and 24 as reducers.
- MR-MPI: 48 cores are used as mappers then as reducers.
- Hadoop: 48 reducers and the number of mappers varies according to the number of partial input files.



X-axis: Data size in gigabytes. Y-axis: log 10 of the running time. Values in the table show the running time in seconds. Values above the columns shows the size of each chuck. ²¹

Outline

- Motivation
- MapReduce
- MapReduce overlapping using MPI (MRO-MPI)
- Experiments
 - Wordcount
 - Distributed inverted files.
- Conclusion

Inverted Files

- **Inverted Files** is an indexing structure composed of two elements: the *vocabulary* and the *posting lists*.
 - **Vocabulary**
 - **Posting lists**

Name= Doc1 #id=1

Computer security known as information security as applied to computers and networks.....

Name= Doc1 #id=2

MapReduce has been used as a framework for distributing larger corpora.....

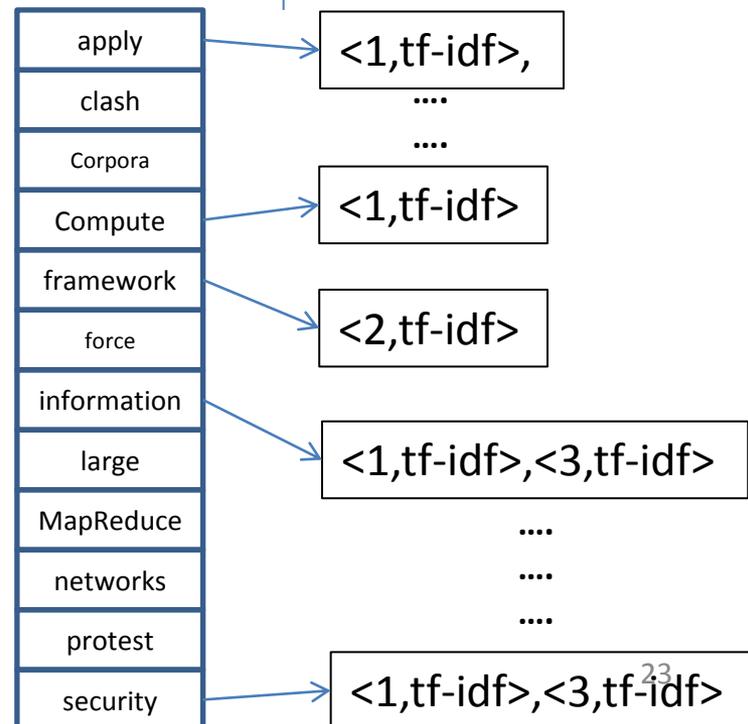
Name= Doc1 #id=3

Protesters have been clashing with security forces. No information.....

.....
.....
.....
.....

Vocabulary

Posting Lists



Inverted Files – tf-idf

- **tf-idf** - weighting scheme (SMART system, 1988):

- Used to evaluate how important a word in a document with respect to other documents in the corpus.

- **Term Frequency (tf):**

$$tf_{ij} = \frac{freq_{ij}}{\max_i freq_{ij}}$$

- $freq_{ij}$: number of occurrence of term t_i in document d_j .

- **Inverse Document Frequency (idf):**

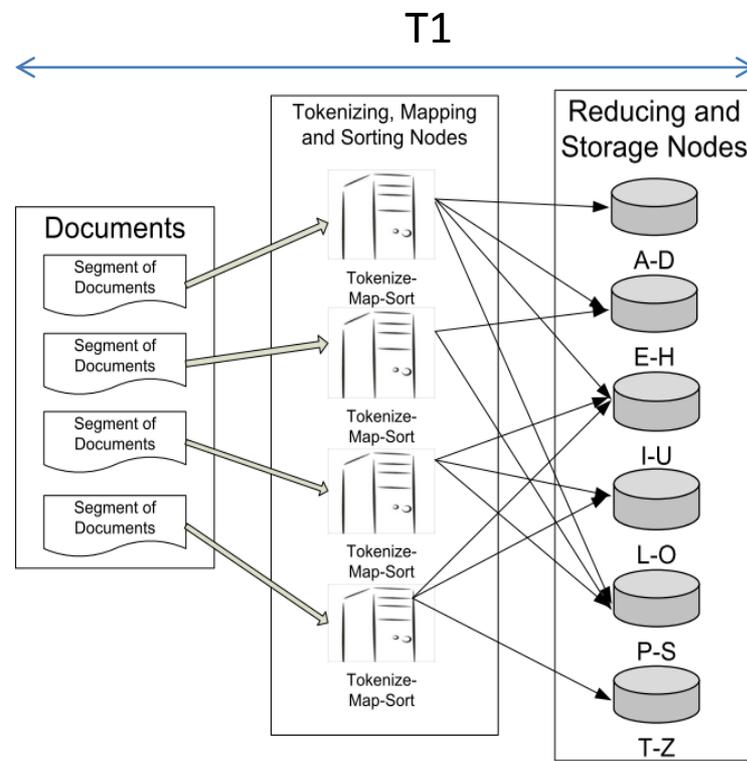
$$idf_i = \log \frac{N}{n_i}$$

- n_i : number of documents where t_i appears.
- N : total number of documents.

$$W_{ij} = tf_{ij} \times idf_i$$

MRO-MPI for inverted files

- Mappers:
 - $(K_m, V_m) = (\text{term}, (\text{document name}, \text{tf}))$.
- Reducers:
 - Distributes the data based on their lexicographic order, each reducer being responsible for a certain range of words.
 - Similar terms are saved into the same database, reducer nodes can calculate the correct tf-idf value.



Distributed inverted files

- 9,319,561 text (XML) excerpts related to 9,319,561 images from 12 million ImageNet corpus.
- Data size: 36GB of XML data.
- Hadoop: 40 minutes with 26 Reducers.
- Double speedup because of sending the data while the map functions is working.
- The best ratio between the mappers and reducers is found to be:

$$2M \geq R \geq M.$$

Map/Reduce	0	4	13	26
0	81.6	-	-	-
4	-	39.2	46.6	51.7
10	-	41.5	39.8	39.2
13	-	66	28.8	24
20	-	111.9	19.6	18.5
22	-	113.9	16.45	14.2
44	-	118.5	125.2	45.8

Conclusion

- We proposed MRO-MPI for intensive data processing.
- Maintain the simplicity of MapReduce.
- High speedup with the same number of nodes.

Questions ?