ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 1, January 2018

Improving Hadoop Performance by Decouple Shuffle and Reduce with MapReduce

Sindhuja.R[1], Priyadharshini.P[2]

P.G. Student, Department of Computer Engineering, K.Ramakrishnan College of Engineering, Kariyamanikam Road, Samayapuram, Trichy, Tamilnadu, India[1]

Assistant Professor, Department of Computer Engineering, K.Ramakrishnan College of Engineering, Kariyamanikam Road, Samayapuram, Trichy, Tamilnadu, India[2]

ABSTRACT: Hadoop parallelizes job execution with map and reduce tasks. Shuffle, the all-to-all input data fetch phase in a reduce task can remarkably affect job performance. To attribute the delay in job completion to the coupling of the shuffle phase and reduce tasks, fails to address data distribution skew among reduce tasks, and makes task scheduling inefficient. In this work, a proposal is made to decouple shuffle from reduce tasks and convert it into a platform service provided by Hadoop. To present iShuffle, a user-transparent shuffle service that pro-actively pushes map output data to nodes via a novel shuffle-on-write operation and flexibly schedules reduce tasks considering workload balance.

KEYWORDS: MapReduce, iShuffle, Data distribution skew, Task scheduling.

I.INTRODUCTION

Hadoop is a popular open-source implementation of the MapReduce programming model for processing large amounts of data in parallel.Each job in Hadoop consists of two dependent phases,which contains multiple user-defined map or reduce tasks.There are many existing studies focusing on improving the performance of map tasks.Work has been done to preserve locality via map scheduling or input replication as the data locality is critical to map performance.Existing work tries to overlap shuffle with map by proactively sending map output or fetching map output in a globally sorted order[1].Unfortunalety,the coupling of coupling of shuffle and reduce phases in reduce task presents challenges to attaining high performance in multi-tenant environments.

First, the fairnessbetween multiple jobs is enforced by limiting the number ofconcurrenttasks of each job, which means the reduce tasksmay not be able to run at the same time. The shuffle phase will not start untill the correspondingreduce task is scheduled to run, and only the first wave ofreduce can be overlapped with map, leaving the potentialparallelism in shuffle unexploited. Second, tasks schedulingin Hadoop is oblivious of the data distribution skew amongreduce tasks.

iShuffle features a number of keydesigns: 1) proactive and deterministic pushing shuffleddata from map to Hadoop nodes when map output files arematerialized to local file systems, shuffle-on-write. 2)automatic predicting reduce execution time based on the input partition size and placing the shuffled data to mitigatethe partition skew and to avoid hotspots. 3) binding reducetasks with data partitions only when reduce is scheduled torealize the load balancing enabled by the partition placement.4) preemptive reduce scheduling to ensure fairnessbetween the reduce tasks from different MapReduce jobs.

II.RELATED WORK

In [2] authors use Shuffle Watcher, a new multi-tenant Map Reduce scheduler that shapes and reduces Shuffle traffic to improve cluster performance (throughput and job turn-around times), while operating within specified fairness constraints. Shuffle Watcher employs key techniques. It curbs intra-job Map-Shuffle concurrency to shape Shuffle

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 1, January 2018

traffic by delaying or elongating a job's Shuffle based on the network load. In [3] authors address Map Reducer's poor performance on heterogeneous clusters. In [4] authors proposedMap Reduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. In [5] authors review the techniques used by such systems, and surveys current commercial and research systems.Highly parallel database systems are beginning to displace traditional mainframe computers for the largest database and transaction processing tasks. In [6] authors proposed the new architecture introduced decouples the programming model from the resource management infrastructure, and delegates many scheduling functions (e.g., task fault-tolerance) to per-application components.

III. PROPOSED SYSTEM

The proposed system uses the shuffle and reduce phases are decoupled.iShuffle is implemented in this design. iShuffle is a job independent shuffle service that pushes the map output to its designated reduce node. It decouples the shuffle and reduce and allows the shuffle to be performed independently from reduce.Shuffle-on-write operation and flexibly schedules reduce tasks considering workload balance. A user-transparent shuffle service that overlaps the data shuffling of any reduce task with the map phase, addresses the input data skew in reduce tasks, and enables efficient reduce scheduling. Binding reduce tasks with data partition for scheduling load balancing. iShuffle reduces the job completion time.

The ishuffling of MapReduce has modules like

- Load Data
- Clustering
- Job Files
- Partition combine
- iShuffle and Data skew

A.Load Data

In this module the user makes to choose the file from the system to load data. The uploaded file will be stored in a database. The dataset will be selected for further process. Then the dataset will be chosen. Finally, all the datasets are inserted into the database table.

B. Clustering

Clustering is the grouping of a particular set of objects based on their characteristics, aggregating them according to their similarities. Regarding to data mining, this methodology partitions the data implementing a specific join algorithm, most suitable for the desired information analysis. In this data to be clustered. Then it can be inserted into the database.

C. Job Files

Job file is used by the Scheduler to run as background tasks. In this module clustered data are considered as job files. These job files are send to server for map reduce. The job files are sent. The socket can be used for the design. There are two or more programs using the same network. The authentication process will be done. The job files are sent and receive the data. The authentication message will be sent. Then click ok to receive the files.

D. Partition Combine

Binding reduce tasks with data partitions only when reduce is scheduled to realize the load balancing enabled by the partition placement. The partition placement algorithm has two heuristics. First is the largest partition first for picking partitions. Second is the less workload first for picking destination nodes. It sorts the partitions in the descending order of size. It repeats until all the partitions are assigned.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 1, January 2018

E. iShuffle and Data Skew

iShuffle is to decouple shuffle from reduce tasks and allows shuffle to be performed independently from reduce. It is a job independent shuffle service that pushes the map output to its designated reduce node.iShuffle reduces the job completion time. Data skew is performed when the process gets workload, it is the process of storing the data into another server based on capacity.

IV. SYSTEM ARCHITECTURE

The data processing in MapReduce model is expressed as two functions: map and reduce. The map function takes an input pair and produces a list of intermediate key/value pairs. A reduce task consists of two phases: shuffle and reduce. The shuffle phase fetches the map outputs associated with reduce task from multiple nodes and merges them into one reduce input. The decoupling of shuffle and reduce phases in a reduce task presents challenges to attaining high performance in multi-tenant environments.
iShuffle consists ofthree components: shuffler, shuffle manager, and task scheduler.The shuffler is a background thread that collects intermediatedata generated by map tasks and predicts the size ofindividual partitions to guide the partition placement. Theshuffle manageranalyses the partition sizes reported by allshufflers and decides the destination of each partition. Theshuffle manager and shufflers are organized in a layeredstructure which is similar to Hadoop's JobTracker andTaskTrackers. The task scheduler extends existingHadoop schedulers to support flexible scheduling of reducetasks.

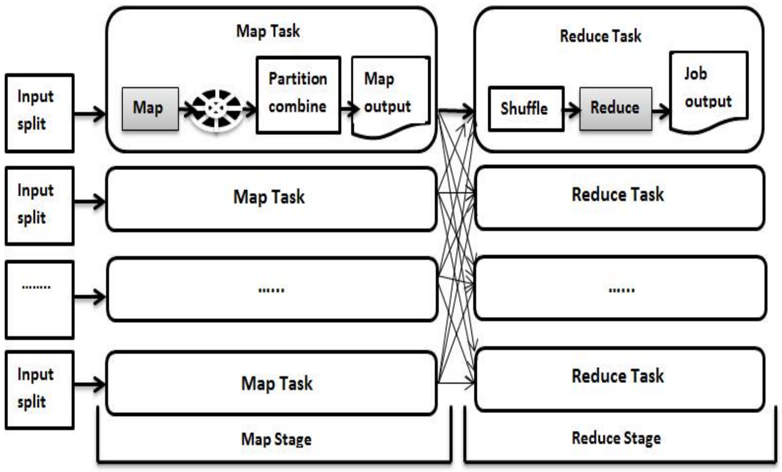


Fig 1. System Architecture

V. RESULT AND OUTCOME

The result is that the shuffle and reduce are decoupled so the performance increased.TheFig 2 represents the data set loading/collective message module.The dataset will be chosen for further process. The dataset will be uploaded.

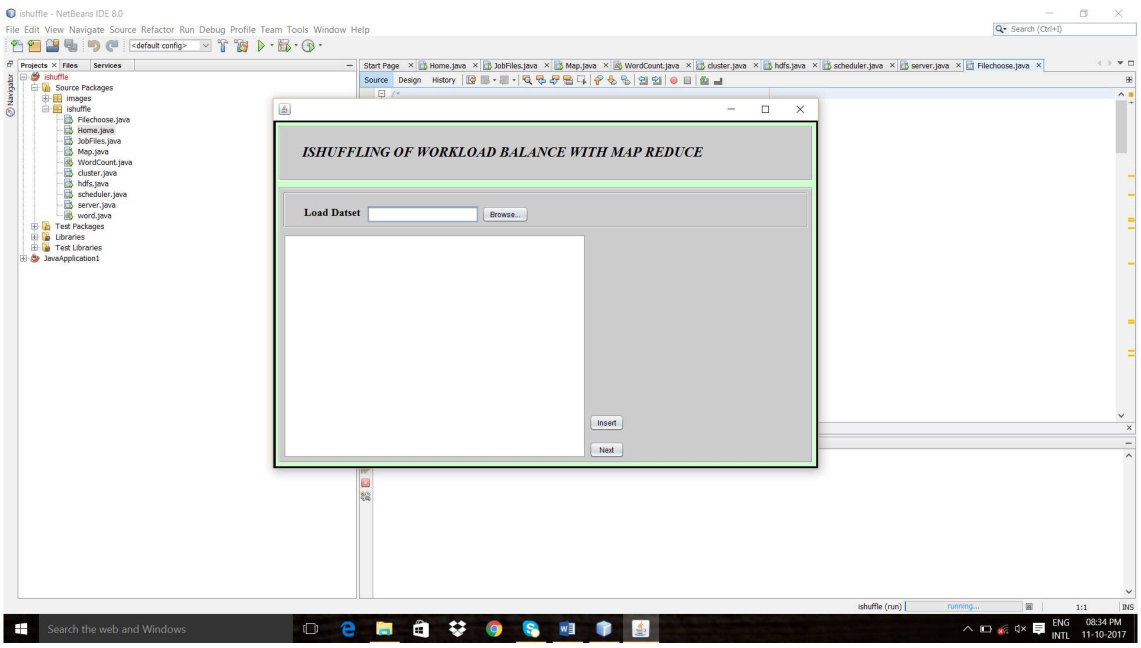


Fig 2. Load Dataset

The Fig 3. represents the Map and Reduce. The MapReduce splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. The input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

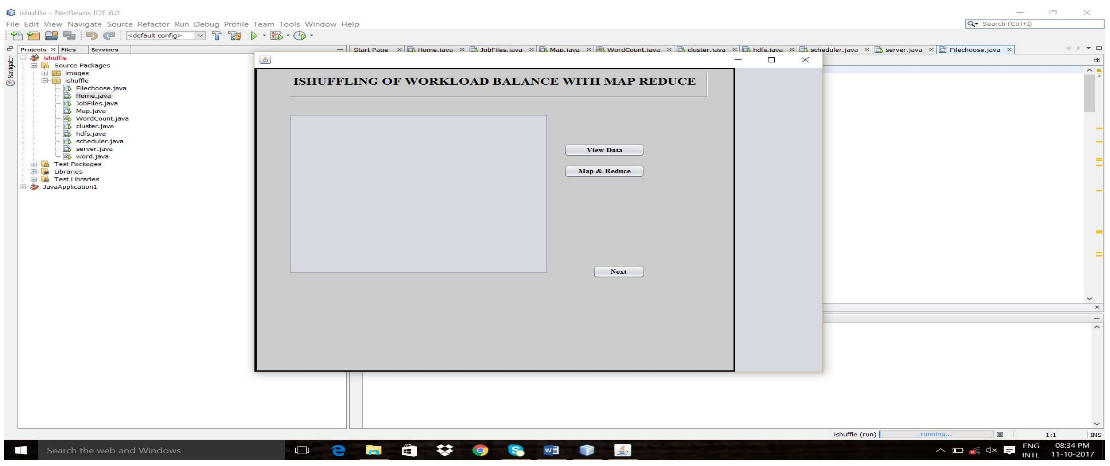
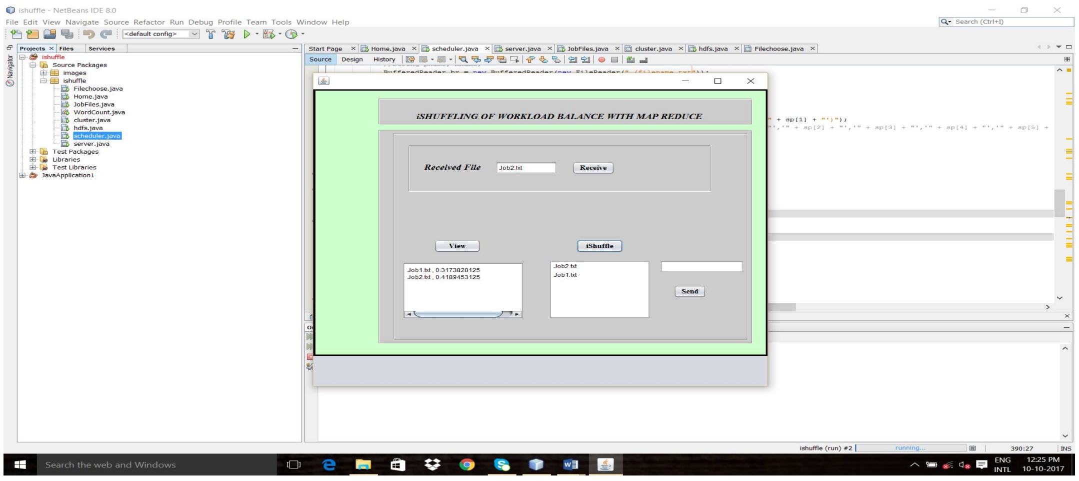


Fig 3. Map Reduce

The Fig 4. represents the iShuffling of Data. It decouples shuffle from reduce tasks. It sorts in descending order.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

Vol. 6, Issue 1, January 2018

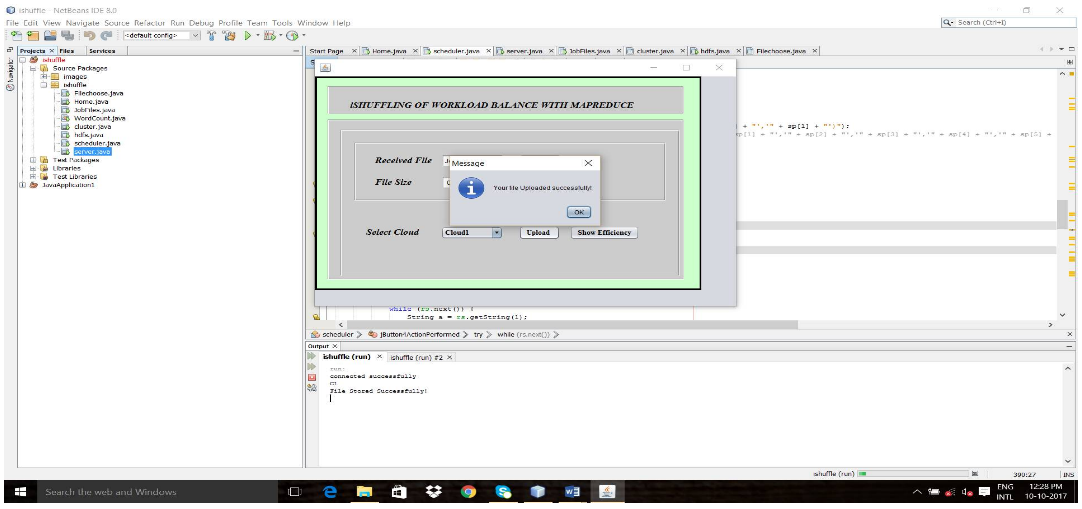


```
Job1.txt , 0.3173828125
Job2.txt , 0.4189453125
```

```
Job2.txt
Job1.txt
```

Fig 4. iShuffle

The Fig 5. represents uploading of files.The files are uploaded in the cloud storage server. It improves the performance.



Your file Uploaded successfully!

```
run:
connected successfully
C1
File Stored Successfully!
```

Fig 5.Uploading of Files

VI. CONCLUSION AND FUTURE WORK

iShuffle, a user-transparent shuffle service that provides optimized data shuffling to improve job performance. It decouples shuffle from reduce tasks and proactively pushes data to be shuffled to Hadoop node via a novel shuffle-on-write operation in map tasks. iShuffle further optimizes the scheduling of reduce tasks by automatic balancing workload on multiple nodes with flexible reduce dispatching and preemptive reduce scheduling.iShuffle also significantly improves job performance in a multi-user Hadoop cluster. The Future enhancement follows the proposed system by utilizing cloud storage as Job files are stored in cloud. The cloud storage can be simulated by using CloudSim Toolkit. CloudSim is a simulation tool that allows cloud developers to test the performance of their

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 1, January 2018

provisioning policies in a repeatable and controllable environment. In the simulated environment, based on the size of the job files, it will be scheduled to the VM.

REFERENCES

1. Ahmad F., Chakradhar S.T., Anand R., and Vijaykumar T.N. (2014), "ShuffleWatcher: Shuffle-aware scheduling in multi-tenant map-reduce clusters," in Proc. USENIX Conf. USENIX Annu. Techn. Conf., pp. 1–12.
2. Ahmad F., Chakradhar S.T., Raghunathan A., and Vijaykumar T.N. (2012), "Tarazu: Optimizing Map Reduce on heterogeneous clusters," in Proc. Int Conf. Archit. Support Program. Languages Operating Syst.,p. 61–74.
3. Ananthanarayanan G., et al. (2011), "Scarlett: Coping with skewed con-tent popularity in Map Reduce clusters," in Proc. ACM Eur. Conf. Comput. Syst., pp. 287–300.
4. Dean J. and Ghemawat S. (2004), "Map Reduce: Simplified data processing on large clusters," in Proc. 6th Conf. Symp. Opearting Syst. Des. Implementation, pp. 10–10.
5. DeWitt D. and Gray J. (1992), "Parallel database systems: The future of high performance database systems," Commun. ACM, vol. 35,no. 6, pp. 85–98.
6. Gandhi R., Xie D., and Hu Y.C. (2013), "PIKACHU: How to rebalance load in optimizing Map Reduce on heterogeneous clusters," in Proc. USENIX Conf. Annu. Tech. Conf., pp. 61–66.
7. Ghodsi A., Zaharia M., Hindman B., Konwinski A., Shenker S., and Stoica I. (2011), "Dominant resource fairness: Fair allocation of multiple resource types," in Proc. USENIX Symp. Netw. Syst. Des. Implementation, pp. 323–336.
8. Kwon Y., Balazinska M., Howe B., and Rolia J. (2012), "Skew Tune: Mitigating skew in Map Reduce applications," in Proc. ACM SIGMOD Int. Conf. Manage. Data,pp. 25–36.
9. Ramakrishnan S. R., Swart G., and Urmanov A. (2012), "Balancing reducer skew in Map Reduce workloads using progressive sampling," in Proc. ACM Symp. Cloud Comput., Art. no.16.
10. Rao S., Ramakrishnan R., Silberstein A., Ovsiannikov M., and Reeves D. (2012), "Sailfish: A framework for large scale data processing," in Proc. ACM Symp. Cloud Comput., Art. no. 4.
11. Shah M. A., Hellerstein J. M., and Brewer E. (2004), "Highly available, fault-tolerant, parallel data flows," in Proc. ACM SIGMOD Int Conf. Manage. Data, pp. 827–838.
12. Tan J., et al.(2014), "DynMR: Dynamic Map Reduce with reduce task interleaving and map task backfilling," in Proc. ACM Eur. Conf. Comput. Syst., Art. no. 2.