Map Reduce Jobs for Distributed Scheduler in Big Data Applications

Shalini. A[1], Latha.P[2], Rajesh.P[3]
[1,2,3] Assistant Professor, CSE, Kingston Engineering College, Vellore, India
Email - shalinibenny@kingston.ac.in, raji.maghudan08@gmail.com, sreelathagtec@gmail.com

Abstract: Most of the data intensive applications executed by data centers are based on Map Reduce or its open-source implementation, Hadoop. Such applications are executed on large clusters requiring large amounts of energy, making the energy costs a considerable fraction of the data centre's overall costs. To minimize the energy consumption when executing each Map Reduce job is a critical task for data centers. we propose a framework for improving the energy efficiency of MapReduce applications.We first model the problem of Energy-Aware scheduling of a single Map Reduce job as an Integer Program. We then propose two heuristic algorithms, called energy-aware Map Reduce scheduling algorithms (EMRSA-I and EMRSA-II), that finds the assignment of map and reduce tasks to the machine slots in order to minimize the energy consumed when executing the application.

Key Words: Map Reduce, Big data, energy-aware scheduling, SLA, EMRSA.

1. INTRODUCTION:

One of the major challenges of processing data intensive applications is minimizing their energy costs. Big data applications run on large clusters within data centers, where their energy costs make energy efficiency of executing such applications a critical concern. For scheduling multiple Map Reduce jobs, Hadoop originally employed a FIFO scheduler. To overcome the issues with the waiting time in FIFO, Hadoop then employed the Fair Scheduler. In most of the cases, processing big data involves running production jobs periodically. For example, Facebook processes terabytes of data for spam detection daily. Such production jobs allow data centers to use job profiling techniques in order to get information about the resource consumption for each job. One of the major challenges of processing data intensive applications is minimizing their energy costs. Big data applications run on large clusters within data centers, where their energy costs make energy efficiency of executing such applications a critical concern. For scheduling multiple Map Reduce jobs, Hadoop originally employed a FIFO scheduler. To overcome issues with the waiting time in FIFO, Hadoop then employed the Fair Scheduler. In most of the cases, processing big data involves running production jobs periodically. For example, Facebook processes terabytes of data for spam detection daily. Such production jobs allow data centers to use job profiling techniques in order to get information about the resource consumption for each job.

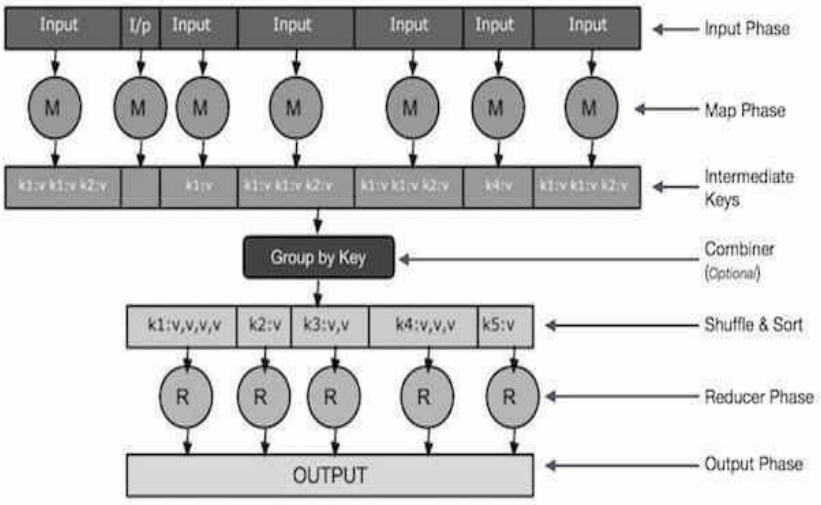


Fig.1

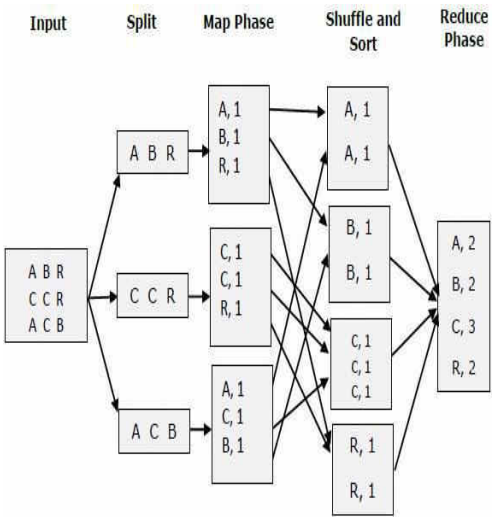
2. MAP REDUCTION:

A classic Hadoop cluster includes a single master node and multiple slave nodes. The master node runs the JobTracker routine which is responsible for scheduling jobs and coordinating the execution of tasks of each job. Each slave node runs the TaskTracker daemon for hosting the execution of MapReduce jobs. The concept of "slot" is used to indicate the capacity of accommodating tasks on each node. In a Hadoop system, a slot is assigned as a map slot or a reduce slot serving map tasks or reduce tasks, respectively. At any given time, only one task can be running per slot. Scheduler will dispatch the jobs to task tracker for map reducing. The number of available slots per node indeed provides the maximum degree of parallelization in Hadoop.

A.Input Phase − Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.

B.Map − Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.

C.Intermediate Keys − The key-value pairs generated by the mapper are known as intermediate keys.



D.Combiner − A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.

E.Shuffle and Sort − The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.

F.Reducer − The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.

G. Output Phase − In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

3. EMRSA-X

1. Create an empty priority queue Q^m
2. Create an empty priority queue Q^r
3. for all j ∈ A do
4. $ecr^m_{j=min}\ \forall_i \in_M \frac{e_{ij}}{pij}$ for EMRSA-I; or

$ecr^{m}_{j=}\frac{\sum_{\forall i\in M}\frac{e_{ij}}{pij}}{M}$,for EMRSA-II

5. $Q^{m.}$.enqueue(j,ecr^{m}_{j})
6. For all j∈ B do
7. $ecr^{r}_{j=min}\ \forall_i\in R\ \frac{e_{ij}}{pij}$ for EMRSA-I; or

$ecr^{r}_{j=}\frac{\sum_{\forall i\in R}\frac{e_{ij}}{pij}}{R}$,for EMRSA-II

8. $Q^{r.}$.enqueue(j,ecr^{r}_{j})
9. $D^{m}\leftarrow\infty$;$D^{r}\leftarrow\infty$
10. While Q^{m} is not empty and Q^{r} is not empty do
11. j^{m} = Q^{m}. extraMin()
12. jr = Qr. extraMin()
13. f= $\frac{\sum_{\forall i\in M}\ pijm}{\sum_{\forall i\in R}\ Pijr}$
14. T^{m}:sorted unassigned map tasks i∈M based on pij^{m}
15. T^{r}:sorted unassigned reduce tasks i∈ R based on pij^{r}
16. if $T^{m=\varphi}$ and $T^{r}=\varphi$ then break
17. ASSIGN –LARGE()
18. ASSIGN –SMALL()
19. If $D^{m}=\infty$ then
20. D^{m}=D-p^{r}
21. D^{r}=p^{r}
22. If $T^{m}\neq\varphi$ or $T^{r}\neq\varphi$ then
23. No feasible schedule
24. Return
25. OUTPUT:X,Y

4. CONCLUSION:

In this paper, we described Energy aware scheduling of Map Reduce-based framework for big data processing in hadoop. Increasing the needs for big data processing and the implementation Hadoop for such processing, improving MapReduce performance with energy saving objectives can have a significant impact in reducing energy consumption in data centers. In this projectr, we show that there are significant optimization opportunities within the MapReduce framework in terms of reducing energy consumption. We proposed two energy-aware MapReduce scheduling alorithms, EMRSA-I and EMRSA-II, that schedule the individual tasks of a MapReduce job for energy efficiency while meeting the application deadline. Both proposed algorithms provide very fast solutions making them suitable for execution in real-time settings.

REFERENCES:

1. J. Koomey, “Growth in data center electricity use 2005 to 2010,” vol. 1. Oakland, CA, USA: Analytics Press, Aug. 2011.
2. J. Hamilton, “Cooperative expendable micro-slice servers (CEMS): Low cost, low power servers for internet-scale services,” in Proc. Conf. Innovative Data Syst. Res., 2009.
3. J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” in Proc. 6th USENIX Symp. Oper. Syst. Des. Implementation, 2004, pp. 137–150.
4. Hadoop. (2014) [Online]. Available: http://hadoop.apache.org/[5] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Job scheduling for multi-user MapReduce clusters,” Electrical Eng. Comput. Sci. Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-55, Apr. 2009.
5. Apc. (2014) [Online]. Available: http://www.apc.com/

6. J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguad_e, "Resource-aware adaptive scheduling for mapreduce clusters," in Proc. 12th ACM/IFIP/USENIX Int. Middleware Conf., 2011, pp. 187–207.
7. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: Automatic resource inference and allocation for MapReduce environments," in Proc. 8th ACM Int. Conf. Autonomic Comput., 2011, pp. 235–244.

A. Verma, L. Cherkasova, and R. H. Campbell, "Two sides of a coin: Optimizing the schedule of MapReduce jobs to minimize their makespan and improve cluster performance," in Proc. IEEE 20th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.,2012, pp. 11–18.
B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: Localityaware resource allocation for MapReduce in a cloud," in Proc. Conf. High Perform. Comput., Netw., Storage Anal., 2011.
C. Moseley, A. Dasgupta, R. Kumar, and T. Sarl_os, "On scheduling in map-reduce and flow-shops," in Proc. 23rd Annu. ACM Symp.

8. Parallelism Algorithms Archit., 2011, pp. 289–298.
9. H. Chang, M. S. Kodialam, R. R. Kompella, T. V. Lakshman, M. Lee, and S. Mukherjee, "Scheduling in MapReduce-like systems for fast completion time," in Proc. IEEE 30th Int. Conf. Comput. Commun., 2011, pp. 3074–3082.
10. F. Chen, M. S. Kodialam, and T. V. Lakshman, "Joint scheduling of processing and shuffle phases in MapReduce systems," in Proc. IEEE Conf. Comput. Commun., 2012, pp. 1143–1151. [14] Y. Zheng, N. B. Shroff, and P. Sinha, "A new analytical technique for designing provably efficient MapReduce schedulers," in Proc.IEEE Conf. Comput. Commun., 2013, pp. 1600–1608.
11. T. J. Hacker and K. Mahadik, "Flexible resource allocation for reliable virtual cluster computing systems," in Proc. Int. Conf. High Perform. Comp., Netw., Storage Anal., 2011, pp. 1–12.
12. B. Palanisamy, A. Singh, and L. Liu, "Cost-effective resource provisioning for MapReduce in a cloud," IEEE Trans. Parallel Distrib. Syst., no. 1, pp. 1, PrePrints, 2014.