Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey on Adaptive MapReduce Scheduling in Shared Environments

Shalini K. C
Dept. of Computer Science and Engineering
VTU, PG Center
Mysuru, India

Dr. Thippeswamy K
Prof. and Hod Dept. of Computer Science and Engineering
VTU, PG Center
Mysuru, India

Abstract— Workloads are the processes which may consume data for analytics need, and some other may act as just data generators in a shared environment. This scenario increases the need of resource-utilization much more in data centre where workload consolidation is considered for improving resource utilization. This case is more challenging if the interaction between workloads which are of from different world, which compete for limited resources. We have proposed a MapReduce scheduler to improve the resource utilization among machines while fulfilling the completion time of certain jobs, this scheduler also keep track of resources that are needed for non-MapReduce workloads. This paper proposes management of MapReduce jobs in the presence of variable resource availability, increasing the accuracy of the estimations made by the scheduler, thus improving completion time goals without an impact on the fairness of the scheduler.

Keywords— MapReduce, Analytics, Transactional, Shared environment, Resource management

I. INTRODUCTION

MapReduce 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.

The MapReduce workloads are introduced in order to deal with the management huge amount of data, providing support at the same time for both data analytics and transactional workloads. Running these workloads in different machines would lead to underutilized resources; hence different workloads are merged to work in shared environment.

In shared environment each workloads are divided and executed simultaneously at any point of time. Because of this nature, there exist a possibility of higher variability and thus the performance of these systems less predictable.

The Reverse Adaptive Scheduler is a scheduler which is well suited to allow the integrated management of data processing structure along with transactional and analytics workloads. This scheduler deals with dynamic resource availability while still being guided by completion time goals.

The resource management is the topic which has been studied in MapReduce environments; this paper has focused on shared environments with transactional workloads.

II. REVERSE-ADAPTIVE SCHEDULER

The driving principles of the scheduler are resource availability awareness and continuous job performance management.

The former is used to decide task placement on tasktrackers over time, while the latter is used to estimate the number of tasks to be run in parallel for each job in order to meet performance objectives, expressed in the form of completion time goals. Job performance management has been extensively evaluated and validated in our previous work, presented as the Adaptive Scheduler.

In this paper we extend the resource availability awareness of the scheduler when the MapReduce jobs are collocated with other timevarying workloads. One key element of our proposal in this paper is the variable Sfit, which is an estimator of the minimal number of tasks that should be allocated in parallel to a MapReduce job to keep its chances to reach its deadline, assuming that the available resources will change over time as predicted by f(t).

Notice that the novelty of this estimator is the fact that it also considers the variable demand of resources introduced by other external workloads. Thus, the main components of the Reverse-Adaptive Scheduler, as described in the following sections, are:

- Sfit estimator
- Utility function that leverages Sfit used as a per-job performance model.
- Placement algorithm that leverages the previous two components.

III. REVERSE FITTING ALGORITHM TO FIND Sfit

Require: J: List of Jobs in the system; sj pend: Number of pending map tasks for each job; Γj and Ωtt: Resource demand and capacity for each job and tasktracker correspondingly, as used by the auxiliary function fit


```
1: for j in J do
2: sjfit = sjpend
3: end for
4: P = []
5: Sort J by completion time goal
6: for j in J do
7: a = Tnext(J)goal // deadline for the next job in J
8: b = Tjgoal // deadline for j
9: for p in P do
10: if spfit > 0 then
11: spfit = spfit − fit(p, a, b)
12: end if
13: end for
14: if sjfit > 0 then
15: sjfit = sjfit − fit(j, a, b)
16: end if
17: Add j to P
18: end for
19: return sjfit for each job in J
```

Algorithm 1

Algorithm 1 shows how this estimation takes place. We first start assuming that for each job j, sj fit equals the number of pending tasks sjpend (lines 1-3), and then proceed to subtract as many tasks as possible beginning from the job with the last deadline to the job with the earliest deadline (lines 5-8), and as long as they fit within the availble amount of resources (lines 9-17).

The algorithm uses the fit() function, which given a job j and two points in time a and b returns the amount of tasks from job j that can be assigned between time a and b, taking into consideration the profile and resource requirements of said job. Notice also how on every iteration we try to fit tasks between the two last deadlines (lines 7-8), and try to assign tasks from jobs with the latest deadlines first as long as they still have remaining tasks left (lines 9-13).

IV. JOB PLACEMENT ALGORITHM

Given an application placement matrix P, a utility value can be calculated for each job in the system. The performance of the system can then be measured as an ordered vector of job utility values, U. The objective of the scheduler is to find a new placement P of jobs on tasktrackers that maximizes the global objective of the system, U(P), which is expressed as follows:

$$\max \min_{j} u_j(P) \quad (4)$$

$$\min \Omega_{tt,r} - \_tt(\_jP_{j,tt}) * \Gamma_{j,r} \; \forall r \quad (5)$$

such that

$$\forall tt \forall r \; (\_jP_{j,tt}) * \Gamma_{j,r} \leq \Omega_{tt,r} \quad (6)$$

and

$$\Omega_{n,r} = \Omega_{tt,r} + \Omega_{ds,r} \quad (7)$$

V. SYSTEM ARCHITECTURE

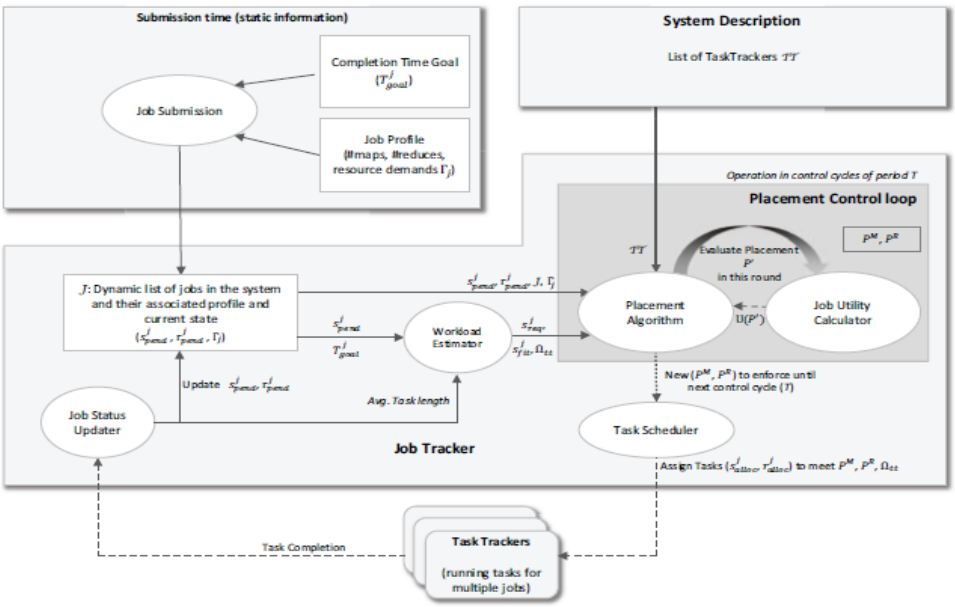

Fig 1 System architecture of the reverse-adaptive scheduler

Above figure 1 illustrates the architecture and operation the scheduler. The system consists of five components: Placement Algorithm, Job Utility Calculator, Task Scheduler, Job Status Updater and Workload Estimator. Each submission includes both the job's completion time goal (if one is provided) and its resource consumption profile.

In this work we concentrate on the estimation of the parameter Sjfit that feeds the Placement Algorithm, as well as the performance model used by the Job Utility Calculator. The major change in this architecture compared to the scheduler presented in is the introduction of the Workload Estimator, that not only estimates the demand for MapReduce tasks as did in previous work, but also provides estimates for the data-store resource consumption, derived from the calculation of f(t).

VI. CONCLUSIONS

In this paper we have presented the Reverse-Adaptive Scheduler, which introduces a novel resource management and job scheduling scheme for MapReduce when executed in shared environments along with other kinds of workloads. Our scheduler is capable of improving resource utilization and job performance. The model we introduce allows for the formulation of a placement problem which is solved by means of a utility-driven algorithm. This algorithm in turn provides our scheduler with the adaptability needed to respond to changing conditions in resource demand and availability of resources.

The scheduler works by estimating the need of resources that should be allocated to each job, but in a more proactive way than previously existing work, since the estimation takes into account the expected availability of resources. In particular, the proposed algorithm consists of two major steps: reversing the execution of the workload and generating the current placement of tasks.

ACKNOWLEDGMENT

I express deepest gratitude to my project guide Dr. Thippeswamy K., Professor and Head, Department of Computer Science and engineering, VTU, PG Centre, Mysuru who has been a source of perpetual inspiration to me throughout my this curriculum.

Finally, I would also like to thank all the faculties of the Department of Computer Science and engineering, non-teaching faculties and my friends who directly or indirectly supported me during my work.

REFERENECES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processingon large clusters," in OSDI'04: Proceedings of the 6th Symposium onOperating Systems Design and implementation. San Francisco, CA:USENIX Association, December 2004, pp. 137–150.

[2] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach,M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: Adistributed storage system for structured data," ACM Trans. Comput.Syst., vol. 26, no. 2, pp. 4:1–4:26, Jun. 2008.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman,A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo:amazon's highly available key-value store," in Proceedings of twentyfirstACM SIGOPS symposium on Operating systems principles, ser.SOSP '07. NY, USA: ACM, 2007, pp. 205–220.

[4] L. A. Barroso, J. Clidaras, and U. H¨olzle, "The datacenter as acomputer: An introduction to the design of warehouse-scale machines,second edition," Synthesis Lectures on Computer Architecture, vol. 8,no. 3, pp. 1–154, 2013.

[5] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi, "Dynamicestimation of cpu demand of web traffic," in Proceedings of the 1st international conference on Performance evaluation methodolgies andtools, ser. valuetools '06. New York, NY, USA: ACM, 2006.

[6] J. Polo, D. Carrera, Y. Becerra, M. Steinder, and I. Whalley,"Performance-driven task co-scheduling for MapReduce environments,"in Network Operations and Management Symposium, NOMS, Osaka,Japan, 2010, pp. 373–380.

[7] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder,J. Torres, and E. Ayguad´e, "Resource-aware adaptive scheduling formapreduce clusters," in Middleware 2011, ser. Lecture Notes in ComputerScience, vol. 7049. Springer Berlin Heidelberg, 2011, pp. 187–207.

[8] A. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: Automatic ResourceInference and Allocation for MapReduce Environments," in 8th IEEE International Conference on Autonomic Computing, Karlsruhe,Germany., June 2011

[9] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysisand demand prediction of enterprise data center applications," inWorkload Characterization, 2007. IISWC 2007. IEEE 10th InternationalSymposium on, Sept 2007, pp. 171–180.

[10] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalableapplication placement controller for enterprise data centers," in Procs.of the 16th intl. conference on World Wide Web, ser. WWW '07. NY,USA: ACM, 2007, pp. 331–340.

[11] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "A methodologyfor understanding mapreduce performance under diverse workloads,"EECS Department, University of California, Berkeley, Tech. Rep.UCB/EECS-2010-135, Nov 2010.

