ISSN:2320-0790

An International Journal of Advanced Computer Technology

A SYNCHRONIZED PROCESS BASED SCHEDULING TO IMPROVE MAP/REDUCE EXECUTION STRATEGY

N. Barathi, R. Dinesh Kumar

[1]Department of Computer Science and Engineering, Bharathiyar College of Engineering and Technology, karaikal/Pondicherry University, India

[2]Department of Computer Science and Engineering, Bharathiyar College of Engineering and Technology, karaikal/Pondicherry University, India

Abstract: MapReduce is a widely used parallel computing framework for large scale data processing. The two major performance metrics in MapReduce are job execution time and cluster throughput. They can be seriously impacted by straggler machines. Speculative execution is a common approach for dealing with the straggler problem by simply backing up those slow running tasks on alternative machines. To improve speculative execution strategies MCP (Maximum Cost Performance) can identify slow task and EWMA (Exponentially Weighted Moving Average) to predict process speed and calculate task completion time. Multiple speculative execution strategies have been proposed, but there is a pitfall: incoming jobs are allocated to nodes present in server and fail to schedule process type allocate to node for processing. To overcome this process we proposed a new scheduling based speculative execution strategy. For scheduling we first calculate number of name node residing in server, minimum threshold of resources allocated to name node. We use minimum to avoid huge interaction among the name node when the competition for resources arises. To choose a proper work node for computing the task, we take both time scheduling and ability of work node to compute the task.

Keywords: MapReduce, hadoop virtual setup

I. INTRODUCTION

The main objective of this project is to improve the effectiveness of speculative execution significantly, accurately and promptly identify stragglers.

By developing a new strategy, MCP (Maximum Cost Performance), this improves the effectiveness of speculative execution significantly. We have to improve the job execution time and cluster throughput. Use both the progress rate and the process bandwidth within a phase to select slow tasks.

The main challenge in enabling resource management in Hadoop clusters stems from the resource model adopted in MapReduce. Hadoop expresses capacity as a function of the number of tasks that can run concurrently in the system. To enable this model the concept of typed-`slot' was introduced as the schedulable unit in the system. `Slots' are bound to a particular type of task, either reduces or map and one task of the appropriate type are executed in each slot.

Pioneer implementations of Map Reduce have been designed to provide overall system goals (e.g., job throughput). Thus, support for user-specified goals and resource utilization management have been left as secondary considerations at best. We believe that both capabilities are crucial for the further development and adoption of large-scale data processing.

On one hand, more users wish for ad-hoc processing in order to perform short-term tasks. Furthermore, in a Cloud environment users pay for resources used. Therefore, providing consistency between price and the quality of service obtained is key to the business model of the Cloud. Resource management, on the other hand, is also important as Cloud providers are motivated and hence require both high levels of automation and resource utilization while avoiding bottlenecks.

II. FRAMEWORK

The frameworks of A synchronized process based scheduling to improve map/reduce execution strategy is shown below:

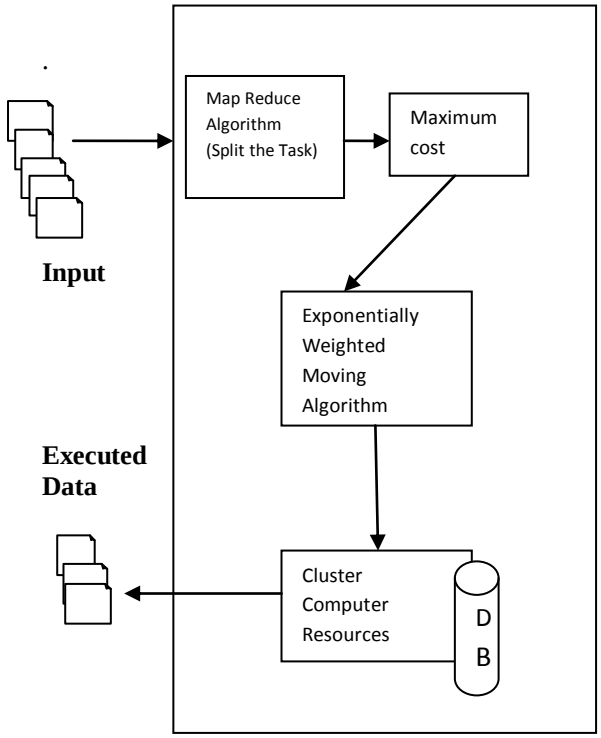

Fig. 1 A framework of a synchronized process based scheduling to improve map/reduce execution strategy.

Use average progress rate to identify slow tasks while in reality the progress rate can be unstable and misleading. In a typical MapReduce job, the master divides the input files into multiple map tasks, and then schedules both map tasks and reduce tasks to worker nodes in a cluster to achieve parallel processing. The main difference between LATE and Mantriis that Mantri uses the task's process bandwidth to calculate the task's remaining time. To use EWMA to predict the process speed of tasks in order to find slow tasks or slow nodes in time.e run this experiment in heterogeneous environments with and without straggler nodes. The scenarios which affect the performance of those strategies: data skew, tasks that start asynchronously, improper configuration of phase percentage and abrupt resource competitions.

III. RELATED WORK

Hadoop Virtual Setup

Hadoop virtual setup comprises a formation of the set of name node and the set of the data load. Name node act as the each processor and it will do the job what we assigned. Data load means how much data are loaded for the name node (i.e.) it describes the storage of the each name node. Compute the processing speed, capacity for each name node in order to assign the job. Allot the job for each name node based upon those requirements

Job Submission

When a user submits a job to the system, this job places in a waiting queue until the scheduler allocates proper resources for its execution. ISM can employ this proposed model and benefit from predictor results in order to accept or reject jobs such that a significant improvement occurs in system utilization and throughput.

In fact, the main goal of ISM is to decrease the average of jobs wait time by giving a low chance to the job which is likely to fail, especially when it has bad effect on a great number of waiting jobs. Furthermore, 1SM gives a high chance to the job which has effect on the start time of a small number of jobs. This behavior causes a decrease in harmful effects of predictor in accuracy and an increase in the numbers of jobs which are rejected mistakenly.

Load Prediction

The short-term forecast requires knowledge of the load from one hour up to a few days. Information derived from the short-term load forecasts are vital to the system as operations in terms of short-term unit maintenance work, weekly, daily, and hourly load scheduling of generating units, and economic and secure operation of power systems.

Time series can be defined as a sequential set of data measured over time, such as the hourly, daily or weekly peak load. The basic idea of forecasting is to first build a pattern matching available data as accurate as possible, then obtains the forecasted value with respect to time using established model. Generally, series are often described as having following characteristics.

$$X(t) = T(t) + S(t) + R(t) \quad t = \ldots -1,0,1,2,\ldots$$

Here, *T (t)* is the trend term, *S (t)* the seasonal term, and *R (t)* is the irregular or random component (which can be generated using Matlab ***normrnd()*** command). We do not consider the cyclic terms since these fluctuations can have a duration from two to ten years or even longer which is not applicable to short-term load forecasting.

We have such assumptions to make things a little easier for the moment:

1) The trend is a constant level;
2) The seasonal effect has period ***s***, that is, it repeats after ***s*** time periods. Or the sum of the seasonal components over a complete cycle or period is zero.

$$\sum_{j=1}^{s} S(t+j) = 0$$

Schedule Monitor:

Given an optimal plan, our objective is to monitor its continued optimality, electing to replan only in those cases where continued execution of the plan will either not achieve the goal, or will do so sub-optimally.

Given an optimal plan, our objective is to monitor its continued optimality, electing to replan only in those cases where continued execution of the plan will either not achieve the goal, or will do so sub-optimally. , a plan continues to be valid if, according to the action theory and the current situation, the precondition of every action in the plan will be satisfied, and at the end of plan execution, the goal is achieved.

A number of systems have been developed for monitoring plan validity (cf. Section 6) which all implicitly takes the following similar approach. The planner annotates each step of the plan with a sufficient and necessary condition that confirms the validity of the plan. During plan execution these conditions are checked to determine whether plan execution should continue.

We formally characterize the annotation and its semantics as goal regression. The provision of such a characterization enables its exploitation with other planners, such as very effective heuristic forward search planners.

IV. CONCLUSION

Speculative execution is a common approach for dealing with the straggler problem by simply backing up those slow running tasks on alternative machines. To improve speculative execution strategies MCP (Maximum Cost Performance) can identify slow task and EWMA (Exponentially Weighted Moving Average) to predict process speed and calculate task completion time. Multiple speculative execution strategies have been proposed, but there is a pitfall: incoming jobs are allocated to nodes present in server and fail to schedule process type allocate to node for processing.

A new scheduling based speculative execution strategy process incoming jobs. For this we first calculate number name node residing in server, minimum threshold of resources allocated to name node. We use minimum to avoid huge interaction among the name node when the competition for resources arises. To choose a proper work node for computing the task, we take both time scheduling and ability of work node to compute the task. By this we evaluate the failure node effectively and avoid the slow task loss by choosing the best nodes to complete the task.

REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008.

[2] "Apache hadoop, http://hadoop.apache.org/."

[3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *Proc. of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, ser. EuroSys '07, 2007.

[4] K. Avi, K. Yaniv, L. Dor, L. Uri, and L. Anthony, "Kvm : The linux virtual machine monitor," *Proc. of the Linux Symposium, Ottawa, Ontario, 2007*, 2007.

[5] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou, "Scope: easy and efficient parallel processing of massive data sets," *Proc. VLDB Endow.*, vol. 1, pp. 1265–1276, August 2008.

[6] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *Proc. of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08, 2008.

[7] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, "Dryadlinq: a system for general-purpose distributed dataparallel computing using a high-level language," in *Proc. of the 8th USENIX conference on Operating systems design and implementation*, ser. OSDI'08, 2008.

[8] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang, "Moon: Mapreduce on opportunistic environments," in *Proc. of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10, 2010.