IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 14, Issue 4 Ver. 2 (July-Aug. 2019), PP 74-76
www.iosrjournals.org

Elimination of Outliers: an Intensive task in parallel Processing

Sudhansu Bisoyi[1], Sachinanadan Mohany[2]
1(Department of Computer Science & Engineering, Gandhi Engineering College, India)
2(Department of Computer Science & Engineering, Gandhi Institute For Technology, India)

***Abstract**: In clusters with heterogeneous systems the progress of data intensive task is estimated using mapreduce framework. But it is not suitable for computational intensive task due to biased estimation of task progress, traditional frameworks cannot timely cut off outliers and therefore largely prolong execution time. Here proposed new framework No Outlier with the instrumentation and outlier clustering techniques for identification of outliers. Since dynamic instrumentation is more precise than static instrumentation, the exact outlier can be identified at runtime and speculative task execution is taken place with average CPU usage.*
***Key Word**: Instrumentation, Outliers Clustering, Scheduler, Parallel processing.*

I. Introduction

The clusters with large number of servers handle the computational intensive task to support users' applications (e.g., information retrieval, analytical or scientific computation). For this large scale infrastructure an efficient processing framework is needed.

The observation is that most existing computing frameworks are suitable for data-intensive tasks but lack consideration for compute-intensive ones. The main reason behind involves the important scheduling component in these frameworks. Many previous investigations have shown that outliers constitute a notorious performance killer in massive task processing. Outliers progress much more slowly than peer tasks and therefore dramatically delay the completion time of the whole job. Many common and unexpected factors may lead to outliers, like uneven task assignment, hardware/software problems, and network congestion. In order to address this issue, computing frameworks heavily rely on a smart scheduler, which is capable of identifying outliers, timely aborting them, and re-executing replica tasks on other healthy nodes.

In this paper, focus on the frameworks that achieve parallelism via SPMD (single program, multiple data) and support tasks with minimal mutual communication, rather than the message-passing tasks (e.g., MPI programs). A job is the unit that a user submits to the framework, and tasks refer to what a job is split into when executed on a large multi-node infrastructure. Here proposed a new framework with a proactive scheduler which schedule the task to various nodes for parallel execution. The task can be dynamically instrumented by including instrumenting code for the functions in an application program before sent for execution. Later gather result of instrumentation from all nodes. On applying the k means clustering algorithm to this resultant data the identification of exact outlier is possible and accordingly cut off outliers.

II. Literature Survey

MapReduce is one of the most succussful and representative frameworks, they lack a mechanism to estimate the progress of compute-intensive tasks and are consequently incapable of action to outliers. For example, MapReduce[2] assigns a specific amount of data to a task and naturally uses the processed data amount or proportion as the indicator of the task's progress. This approach embodies an assumption that the task progress is approximately linear with data I/O. Furthermore, main subsequent improvements on the scheduler of MapReduce also hold this assumption. However, it is not the case in compute-intensive tasks. When performing this kind of tasks, data I/O activities mostly take a small portion of time and are skewed within a small interval. Therefore the traditional mechanism of progress estimation and outlier identification becomes problematic if directly applied to compute-intensive tasks. Therefore a new proposed framework for computational intensive task with following challenges:

- We design and implement an easy-to-use, efficient and robust framework for parallel execution of massive compute-intensive tasks.
- We create an instrumentation-based technique to estimate the progress of compute-intensive tasks.
- At runtime, we design a scheduling policy based on the k-mean clustering method to identify outliers.

III. System Design

The proposed system which identify the outliers in clusters with heterogeneous systems. The proposed No Outlier(NO) system has two components. One is instrumenter and another one is outlier clustering method. The instrumenter will dynamically instrument the computational intensive task by including the code snippet at the entry and exit point of the functions. This task will be distributed to various node for the parallel execution by the proactive scheduler using policy. Later, with some time interval, result dynamical instrumented tracing statistical data is collected from all parallel execution of the application program is taken place. On applying the kmeans clustering algorithm on tracing statistical data to identify the exact outlier. Now policy generator will speculate[8] the outlier to another healthy node by rescheduling the tasks.

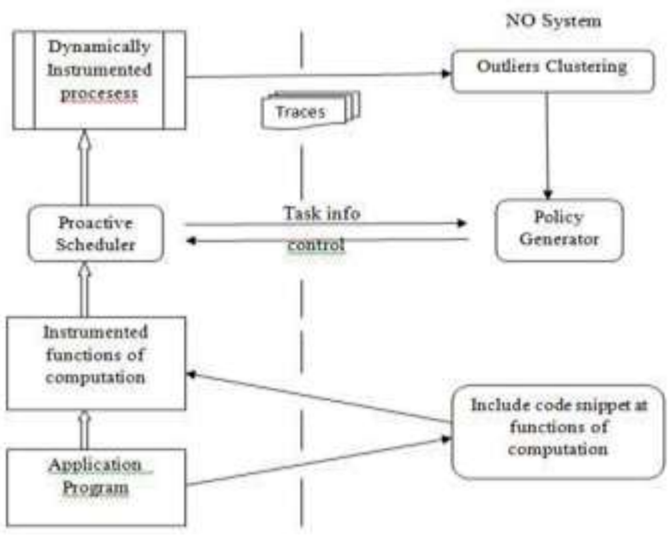
NO System
Dynamically Instrumented procesess
Outliers Clustering
Traces
Proactive Scheduler
Task info
control
Policy Generator
Instrumented functions of computation
Include code snippet at functions of computation
Application Program

Figure:1. Architecture

IV. Proposed Algorithm

The speculative algorithm is as shown below
Input: Instance of job and sleep interval
Output: The job state

```
1.  While job is not finished due to outliers
2.  Assign jobs as, blacklist.job=job.state
3.  Select exact outlier.job from blacklist using k mean clustering method
4.  For all jobs in blacklist sort by high priority
5.  Execute the outlier.job in order
6.  Sleep for interval
7.  for all jobs in blacklist
8.  If blacklist.job is finished
9.  Remove it from the blacklist or kill the job
10. Now job.state is not in blacklist
```

The speculation algorithm of tasks scheduling for a job with the rule is to assign a higher speculation priority to a worse outlier. The goal of speculation is to complete the job early. Earlier and faster speculation means better opportunity.

V. Conclusion

As most existing computing frameworks are oriented to data-intensive tasks, we propose an efficient and robust framework, NO, for massive compute intensive tasks. Our framework is based on Instrumentation, ProActive scheduler and kmeans clustering techniques to identify outliers. In a large-scale computing framework, a key responsibility of the scheduler is to avoid outliers that may severely prolong the job completion.

An extension will focus on identify the waiting queue for the speculative task else which leads to more resource usage of CPU for execution with increased overhead. So, proper identification of queue length leads to choose healthy node for execution.

References

[1]. M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, “Improving mapreduce performance in heterogeneous environments,” in Proceedings of the 8th USENIX conference on Operating systems design and implementation, ser. OSDI’08. Berkeley, CA, USA: USENIX Association, 2008, pp. 29–42.

[2]. G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E.Harris, "Reining in the outliers in mapreduce clusters using mantri," in Proceedings of the 9th USENIX conference on Operating systems design and implementation, ser.OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–16.

[3]. I. Ahmad and Y. kwong Kwok, "On exploiting task duplication in parallel program scheduling," IEEE Transactions on Parallel and Distributed Systems, vol. 9,pp. 872–892, 1998.

[4]. E. B. Nightingale, P. M. Chen, and J. Flinn, "Speculative execution in a distributed file system," ACM Trans. Comput. Syst., vol. 24, no. 4, pp. 361–392, Nov. 2006.

[5]. L. Carrington, A. Snavely, and N. Wolter, "A performance prediction framework for scientific applications," Future Gener. Comput. Syst., vol. 22, no. 3, pp. 336–346, Feb. 2006.

[6]. J. Dean and S. Ghemawat, "Mapreduce: Simplied data processing on large clusters," in Operating Systems Design and Implementation, 2004, pp. 137–150.

[7]. S. N. Srirama, P. Jakovits, and E. Vainikko, "Adapting scientific computing problems to clouds using mapreduce," Future Gener. Comput. Syst., vol. 28, no. 1,184–192, Jan. 2012.

[8]. NO2: Speeding Up Parallel Processing of Massive Compute-Intensive Tasks, Yongwei Wu, Weichao Guo, Jinglei Ren, Xun Zhao, and Weimin Zheng, Member, IEEE.

[9]. Nilay Narlawar and Ila Naresh Patil, "A Speedy Approach: User-Based Collaborative Filtering with Mapreduce", International journal of Computer Engineering & Technology (IJCET), Volume 5, Issue 5, 2014, pp. 32 - 39, Priya Deshpande and Sunayna Giroti, "Priority Based Dynamic Adaptive Checkpointing Strategy in Distributed Environment", International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 6, 2013,pp. 378 - 385, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.

[10]. Paulo J. G. Lisboa, Huda Naji Nawaf and Wesam S. Bhaya, "Recommendation System Based on Association Rules Applied to Consistent Behavior Over Time",International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 4, 2013, pp. 412 - 421, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.