International Journal of Research in Engineering, Science and Management
Volume-2, Issue-2, February-2019
www.ijresm.com | ISSN (Online): 2581-5792

Comparative Analysis of Outlier Mitigation Techniques

Sonali Dudhal[1], S. B. Deshmukh[2]
[1]*M.E. Student, Department of Information Technology, Pune Institute of Computer Technology, Pune, India*
[2]*Professor, Department of Information Technology, Pune Institute of Computer Technology, Pune, India*

***Abstract*: It is believed that the Outliers have a big one Impact on the performance of the Big Data. However, the reason for which discharges are caused is complicated. The precedents work mainly focus on detecting Outliers, optimizing the level of programming and Analysis of the root cause. These methods cannot provide Valuable information to help users optimize their programs. In this paper, propose new approach, a general method that incorporates Framework and characteristics of the system for the analysis of the root cause of outliers in the big data system. Big Roots considers the features from the Big Data pane, like the random read / write bytes and time to collect useless JVM data, as well as system resources use as CPU, I / O and network, which is able to detect Internal and external causes of Outliers. The experimental results prove that this technique is effective in identifying the root cause of Outliers and provides useful indications for performance optimization.**

***Keywords*: Big Data, root cause analysis, spark, Outlier.**

1. Introduction

Today's computer services are purchased all over the world through the use of data centers in the cloud. These are based on the Internet Virtual computing environments are distributed. Systems composed of hundreds and thousands of interconnected nodes and are essential for consumer compliance Quality of service and business objectives. Data centers in the cloud strongly exploit the virtualization to be formed groups of computers able to distribute effectively parallelizable frames such as MapReduce and Spark all this requires large amounts of calculation Power and storage capacity to operate on a large scale. This has subsequently, it promoted huge consumer acceptance for Cloud Based.. This promoted the formation of cloud data centers composed of Thousands of nodes and millions of virtualizes based on the cloud services, leading to a larger scale and to the complexity of the system among the components that interact. Then manifestation of the previously invisible emerging system the behavior has arisen in these distributed systems, represents a significant threat to effective supply Performance of the virtualized service.

In this paper, we propose, a general method for outlier detection. Incorporating the features of both the framework and the system the analysis of the root cause of the outlier which covers a larger one spectrum of causes and provides intuitive guide to performance improvement. The idea behind this approach is to compare the characteristics of the outliers with the normal tasks in The same scenario If the value of a delay function is deviated for much of the normal task, we treat this feature like the root cause of the delay This method outliers solves the drawbacks. Of previous work and provides a useful guide for the future performance optimization. Furthermore, the statistical rules are applied to different characteristics to reduce false positive. For example, we can filter the characteristics that represent the blocking time are many less than the duration of the activity. The reason is that if the time elapsed in such characteristics it is insignificant, so these characteristics would not do it the performance of the task is strongly affected.

2. Motivation

Data processing framework are dividing the data in small pieces and perform the operation in parallel. Only when each activity ends within a level the application can proceed to the next phase. If certain tasks are slower than the rest in the same phase, the execution of the whole application is slowed down by these activities called Outliers.

A. Objectives

- To find outliers in specific jobs.
- To detect both internal and external root causes of outliers.
- To improve performance.

3. Related work

Literature survey is the most important step in any kind of research. Before start developing we need to study the previous papers of our domain which we are working and on the basis of study we can predict or generate the drawback and start working with the reference of previous papers.

A. Slower Nodes or Outliers

Outlier nodes are the nodes on which a task takes much longer than normal to finish. Outlier nodes increase execution time and reduce cluster throughput. Outlier nodes degrade the cluster throughput. Outlier tasks are a hurdle in getting faster completion of applications on modern frameworks.

B. Causes of Outlier

1) Internal factors

- Resource capacity of worker nodes is heterogeneous.
- Tasks running on same worker node compete for resources among themselves.

2) External factors

- Co-hosted applications compete for resources.
- Input data may be skewed.
- Unacceptably slow speed of remote input/output source.
- Hardware fault.

In this section, we briefly review the related work on outliers detection system and their different techniques.

1) MapReduce: Simplified Data Processing on Large Cluster: MapReduce is a data processing approach, in which a single machine acts as a master, assigning map / reduce activities to all the other machines connected in the cluster. Technically, it could be considered as a programming model, which is applied to generate, implement and generate large data sets. The map reduction feature was adopted and implemented in several organizations such as Google, Apache Hadoop and Riak. Programmers can easily learn MapReduce without having any previous experience. Eliminates load balancing, fault tolerance, serialization and parallelization needs.

2) Outliers Root-Cause and Impact Analysis for Massive-scale Virtualized Cloud Data centres: This paper presents an empirical analysis of two large-scale virtualized cloud data centers to determine the impact and root cause of stragglers; Emerging phenomena found within distributed systems on a scale. The results were used to guide the development of a detection system for late laggards by combining off-line analysis and agent-based monitoring.

3) MrHeter: improving MapReduce performance in heterogeneous environments: We analyze the causes of poor MapReduce performance in heterogeneous groups, and the most important is the unreasonable assignment of tasks between nodes with different computational capabilities. On this basis, we propose to MrHeter, which separates the MapReduce process in the phase of random mixing of the map and the reduce operation phase, then builds the optimization model separately for them and obtains a different assignment of tasks for heterogeneous nodes according to the capacity of calculation.

4) Detection of Performance Anomalies in Web-based Applications: Performance management and reliability are two of the critical problems of business-critical applications. The ability to detect the occurrence of failures and performance anomalies has attracted the attention of researchers in recent years. In reality, this is a difficult problem, since a visible change in performance can be due to natural causes (for example, changes in workload, updates) or due to an internal anomaly or an error that can end in a performance error or application error.

5) Online Anomaly Prediction for Robust Cluster Systems: The anomaly forecasting scheme generates early alarms for imminent system anomalies and suggests possible causes of anomaly. Specifically, we use methods of Bayesian classification to capture various symptoms of abnormality and to infer causes of anomaly. Markov models are introduced to capture changing patterns of different measurement metrics. More importantly, our scheme combines Markov models and Bayesian classification methods to predict when an anomaly of the system will appear in the near future and what are the possible causes of the anomaly. To our knowledge, our work provides the first flow-based mining algorithm to predict system anomalies.

6) Proactive Straggler Avoidance using Machine Learning: This paper predicts the task execution time Facebook trace analysis using the Pearson's correlation coefficient. Numbers of relevant parameters are collected from the Facebook traces which are based on the input. System flow is as follows: Firstly, it collects the job history per node i.e. what resources are been utilized by the job and the time required for its execution. Second it extracts the feature from job history log like start time, end time etc. And finally schedules the job as per the policy i.e. the machine learning algorithm being utilized. Resource utilization parameters used are as follows: CPU Usage at Entry i.e. CPU usage at particular moment, physical Memory at Entry i.e. Physical memory utilization like virtual Memory At Entry, local Read Rate At Entry, local Write Rate At Entry, remote Read Rate At Entry, remote Write Rate At Entry, Max number of total tasks executing simultaneously which is called Physical memory peak. Max virtual memory utilization at any point etc., using which straggler detection becomes easier.

7) Insight and Reduction of Map Reduce Stragglers in Heterogeneous Environment: This paper proposes an approach for problem solving based on the relationship between system parameter. This approach adjusts the amount of task slots of task of node dynamically to match the processing power, according to current task progress wrangler avoid the resource wastage by removing the need of replication of tasks. In case of over resource utilized scenario, the execution time of the task is prolonged, so using the resource utilization we can identify which task is performing slower and we can consider it as straggler.

8) Heterogeneity and Dynamicity of Clouds at Scale Google Trace Analysis: This paper analyses the first monitoring data publicly available from a large multi-purpose group for better understanding of the challenges in developing effective cloud-based resource developers. The most important feature of the workload is the heterogeneity: in the types of resources (for example, core: RAM per machine) and their use (for example, the duration and resources required). This heterogeneity reduces the effectiveness of traditional slot and core programming. Furthermore, some activities are limited in terms of the type of machines they can use, which increases the complexity of resource allocation and complicates the migration of activities.

9) BigRoots: An Effective Approach for Root-cause Analysis

of Stragglers in Big Data System: This paper presents the general method for straggler detection, which takes into consideration both the system and the framework features called as the root-cause analysis in big data systems. Features like system resource utilization such as CPU, I/O and network and big data framework logs, are used which is able to detect both internal and external root causes of stragglers. Data locality is one of the features of big data log which helps in identifying from where the data needs to be fetched and time require for fetching it from local or remote node. System features used are CPU utilization, I/O or disk utilization and network utilization are being considered.

10) Wrangler: Predictable and Faster Jobs using Fewer Resources: This paper proposes the system that predicts stragglers using an interpretable linear modeling technique. Wrangler avoid the need replication of tasks to avoid the resource wastage. It uses the Node level features like CPU, Network (CPU idle time, system and user time and speed of the CPU), Disk (Number of bytes sent and received, statistics of remote read and write, statistics of RPCs, etc.) and, Memory utilization (Amount of virtual, physical memory available, amount of buffer space, cache space, shared memory space available, etc.), System-level features like states of thread i.e. waiting, running, terminated, blocked, etc .

11) Reducing Late-Timing Failure at Scale: Straggler RootCause Analysis in Cloud Datacenters: This paper proposes a method to for identifying straggler root-causes by analyzing key parameters within large-scale distributed systems, and also to determine the correlation between straggler occurrence and factors including resource contention, task concurrency, and server failures. Features used for the straggler detection are Data skew, high resource utilization (CPU, memory, disk), network package loss.

12) Reining in the Outliers in Map-Reduce Clusters using Mantri: This paper proposes a system for detecting the stragglers which concerns about the global resources. It estimates the remaining time for completion of task and time to run the new copy of data and avoids the redundant copies of data. Following are the features considered for straggler detection: Data Skew, Crossrack Traffic, CPU utilization. But the disadvantage associated with this technique is that Mantri needs large amount of time to detect stragglers and leading to wastage of time resulting in job delay.

13) Straggler Root-Cause and Impact Analysis for Massive-scale Virtualized Cloud Datacenters: This paper presents an empirical analysis of two production large-scale virtualized Cloud datacenters to ascertain the impact and root-cause of stragglers. Features used for straggler detection are: High CPU utilization, unhandled operational access request, Network package loss, Hardware faults. But the disadvantage associated with this technique is that it uses correlation & diagnosis method to identify the root causes of stragglers. However, their approach is at job level and does not provide specific reason to the cause of stragglers.

14) Improving Resource Utilization in MapReduce: For improve resource utilization, this paper propose a technique called as resource stealing which enables the running tasks to steal resources reserved for idle slots and provide them back when new tasks are assigned. Resource stealing takes care that the wasted resources are fully utilized without interfering with normal job scheduling. It measures the user-perceivable job execution time which helps to provide the deterioration or improvement of resource utilization. For the experimental purpose it utilizes the word count which is best example for I/O intensive application.

15) Spark: Cluster Computing with Working Sets: We propose a new framework called Spark that supports these applications while maintaining MapReduce scalability and fault tolerance. To achieve these goals, Spark introduces an abstraction called Resilient Distributed Datasets (RDD). An RDD is a read-only collection of partitioned objects in a set of machines that can be rebuilt if a partition is lost. Spark can exceed 10-fold Hadoop in iterative machine learning processes and can be used to interactively query a 39 GB data set with a response time of less than one second.

Table 1
Comparative study

S. No.	Paper Title	Authors	Feature Used for detection/ causes of outliers
1	BigRoots: An Effective Approach for Root-cause Analysis of Stragglers in Big Data System	Honggang Zhouy, Yunchun Liy, Hailong Yangy, Jie Jia, Wei Liy	CPU utilization, disk utilization, network utilization, Framework Features like Spark logs file, resource features and time features.
2	Straggler Root-Cause and Impact Analysis for Massive-scale Virtualized Cloud Datacenters	Peter Garraghan, Xue Ouyang, Renyu Yang, David McKee, Jie Xu.	High CPU utilization, Unhandled operational access request, Network package loss, Hardware faults.
3	Straggler Detection in Parallel Computing Systems through Dynamic Threshold Calculation	Xue Ouyang, Peter Garraghan, David Mckee, Paul Townend, Jie Xu	Service QoS (specifically timing constraints), task execution progress, and the cluster resource usage.
4	Reining in the Outliers in Map-Reduce Clusters using Mantri	Ganesh Ananthanarayanan, Srikanth Kandula, Albert Greenberg, Ion Stoica, Yi Lu, Bikas Saha, Edward Harris	Data Skew, Crossrack Traffic, CPU utilization.
5	Wrangler: Predictable and Faster Jobs using Fewer Resources	Neeraja J. Yadwadkar, Ganesh Ananthanarayanan, Randy Katz	Node level features like CPU, Network, Disk and, Memory utilization, System-level features like states of thread.
6	Reducing Late-Timing Failure at Scale: Straggler Root-Cause Analysis in Cloud Datacenters	Xue Ouyang, Peter Garraghan, Renyu Yang, Paul Townend, Jie Xu	Data skew, high resource utilization (CPU, memory, disk), network package loss.

4. Existing approaches

Lot of work has been done in this field because of its extensive usage and applications. In this section, some of the approaches which have been implemented to achieve the same purpose are mentioned. These works are majorly differentiated by the algorithm for outliers detection system.

Many existing works have proposed speculative execution for Mitigating the impact of outliers. Current Big Data Frames They have already adopted the speculative execution method, starting a replicated task on another machine if an activity is much slower than others. Google says so Speculative execution improves work response time a 44% offer the longest Approximate completion time speculative strategy for heterogeneous cluster. They are based on the rate of progress to detect possible laggards and start only speculative activities in fast nodes with high rate of progress. Focus on the small it works and introduces Dolly that clones all the tasks of a small job. To avoid waiting during speculation. Dolly achieves significant results Accelerate for small jobs and consume only an additional 5% However, all these speculative methods have the same shortage. Speculation consumes additional resources as a result, production groups perform many jobs at the same time the speculative execution will contend the resources with normal work. Speculative tasks are not necessary and waste resources in production environment.

Existing techniques used for Outlier Detection or Task Scheduling:

a) Dolly
b) Mantri
c) Wrangler
d) LATE
e) SAMR
f) ESAMR
g) Multi-task learning
h) Blacklisting
i) Speculative Execution

5. Proposed approach

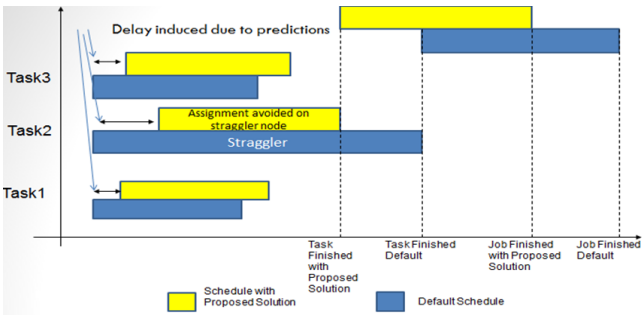

Fig. 1. Proposed approach

The proposed architecture tries to identify that which worker node should be assigned the task; this is done through distributed machine learning algorithm. It requires some time to carry out this processing, if the algorithm identifies whether the node is performing the task with slow rate then that node is blacklisted for short period of time.

6. Conclusion

Big data consists of the large variety of data that may be diverse in nature. When this data is divided into small pieces for the processing, then that pieces are called tasks. Big data is the data which contains both structured and unstructured data, for handling such huge amount of data, popular frameworks have been developed which are MapReduce, Dryad, Spark which require vast amounts of compute power and storage capacity to operate. These frameworks are useful for extracting useful information. They divide the job into small pieces or chunks called as tasks. In certain situations, it happens that, one task gets slower as compared to another task, this phenomenon is called as Outliers, as a result the execution of the entire application is slowed down by these tasks. Outliers adversely affect the performance of big data systems. Many a times High CPU utilization, High disk utilization, Network package loss, Hardware faults, Data skew, etc. are the factors responsible for the happening of outliers. So as to overcome the outliers, BigRoots, Mantri, Dolly, Wrangler, etc. techniques are attempted and provides guidance for performance optimization. The idea behind this approach is to compare the features of outliers with normal tasks in the same stage and if value of a outliers feature deviates greatly from that of normal task, then this feature is treated as the root cause of outliers. In this project, we will try propose the system that will analyze more features that results in outliers, and avoid the same, which will help to optimize the performance of the Big Data Systems.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *Conference on Symposium on Operating Systems Design & Implementation*, 2008, pp. 10–10.
[2] P. Garraghan, X. Ouyang, R. Yang, D. McKee, and J. Xu, "Outlier root-cause and impact analysis for massive scale virtualized cloud datacenters," *IEEE Transactions on Services Computing*, 2017.
[3] Xiao Zhang, Yanjun Wu, Chen Zhao, "MrHeter: improving MapReduce performance in heterogeneous environments," in *Springer Science Business Media*, New York 2016.
[4] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing. ACM*, 2012, p. 7.
[5] J. P. Magalhaes and L. M. Silva, "Detection of performance anomalies in web-based applications," in *Network Computing and Applications (NCA), 2010 9th IEEE International Symposium on. IEEE*, 2010, pp. 60–67
[6] X. Gu and H. Wang, "Online anomaly prediction for robust cluster systems," in *Data Engineering, 2009. ICDE'09, IEEE 25th International Conference on*. IEEE, 2009, pp. 1000–1011.
[7] Arefin, V. K. Singh, G. Jiang, Y. Zhang, and C. Lumezanu, "Diagnosing data center behavior flow byflow," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on. IEEE*, 2013, pp. 11–20.
[8] D. J. Dean, H. Nguyen, and X. Gu, "Ubl: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems," in *Proceedings of the 9th international conference on Autonomic computing. ACM*, 2012, pp. 191–200.

[9] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. ACM, 2007, pp. 59–72.

[10] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." *Hot Cloud*, vol. 10, no. 10-10, p. 95, 2010.