Modeling and Performance Evaluation of MapReduce in Cloud Computing Systems Using Queueing Network Model

Guzlan Miskeen

Dept of Computing, Faculty of Education Brack

University of Sebha

Sebha, Libya

Guz.Miskeen@Sebhau.Edu.Ly

Received:00 April 2018 / Accepted: 00 May 2018

A B S T R A C T

MapReduce is a two -stage information processing technique and it is common concept for big data. Map and Reduce procedures are distributed among some processors within a cluster in the cloud. The performance modeling and analysis of MapReduce execution times has been a challenging task. Analytic performance models provide reasonably accurate job response time estimation with significantly lower cost compared with experimental experiments. Queueing theory is one the modeling and analysis tools of such systems since it enables efficient analysis of the performance, availability and some other key metrics of a data processing system. In this paper, an M/G/1/K performance model with first come first serviced (FCFS) discipline of MapReduce is proposed. More specifically, it will present a queueing model with two stages hypoexponential service time and finite queue. This model has a cloud server with two stages to investigate the performance of the MapReduce technique subject to heavy traffic conditions. The system is analyzed via discrete-event simulation (DES). Key numerical examples are presented for varying number of mappers, reducers and the mean arrival rates to assess their effect on the system mean response time, loss probability and mean queue length. The results are expected to be useful for predicting MapReduce under various workloads and operating conditions of big data processing.

Keywords: Cloud computing, MapReduce, Performance Modeling, Queuing Theory, Hypoexponential distribution

1 Introduction

MapReduce is a well- known programming model that process in parallel large data on cloud clusters [1]. This model is composed of map and reduce functions, "Map" function processes a key/value pair to generate a set of intermediate key/value and a "Reduce" function merges all intermediate values associated with the same intermediate key [2].

In big data cluster, a MapReduce job is divided into several tasks that are executed on parallel on multiple virtual machines (VMs), which significantly reduce the job execution time. The operating concept of a MapReduce is depicted in Figure 1. The Hadoop Distributed File System (HDFS) is an open source System that is responsible for storing replicated data fashion and run in a distributed way on a cluster of servers [1].

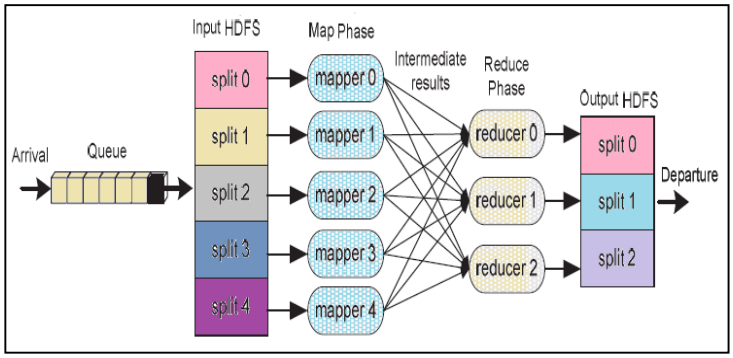

Figure 1: The *operating concept of MapReduce process (adopted from [1])*

For a cloud MapReduce cluster, a cloud node is a VM which can have several mappers and reducers [1]. MapReduce functionality can be described as following: upon arrival, MapReduce jobs are queued for processing at the cloud cluster which has hundreds of nodes. The job is scheduled by the load balancer which plays an important role in dispatching, monitoring, and tracking the availability of nodes at the cloud datacenter. Then, the input data of a MapReduce job is split into multiple data set. As a result, the map phase is initiated where each data set is processed by one mapper node to produce intermediate key/value pairs or results. After all data sets have received the required service, the reduce phase starts where each reducer can process and aggregate the intermediate results to form the final output results [3].

The ratio of the total mappers to reducers can be specified by the user and the job tracker, with additional controlling functionality, is responsible for provisioning the correct number of slave nodes (which host the mappers and reducers) to meet the QoS conditions. There is a tradeoff between cloud cluster's performance and the cloud usage cost since allocating fewer resources than required will affect the cloud's performance while allocating more nodes will increase the cost to the cloud user due to the over- provisioning[1]. Therefore, knowing the correct number of nodes can be implicitly made via determining the number of mappers and reducers needed to execute a MapReduce job and then resolving the cloud MapReduce performance-cost tradeoff.

The rest of the paper is organized as follows. Section 2 reviews the key studies on the performance evaluation of map reduce technique. Section 3 presents the proposed queueing mode to capture the MapReduce node's behaviour. Section 4 presents DES and numerical

examples to show how to utilize the proposed model in predicting the performance of the node. Feasible extension of the proposed model is presented in Section 5 and finally Section 6 concludes the study and suggests directions for future work.

2. Related work

There has been some prior work on the performance of MapReduce node. A queueing network model with hypoexponential service time and finite queue was proposed to study and analysis the performance of MapReduce and multi-stage big data processing [4]. In [5] MapReduce model behaviour was captured via a Triple-Queue Scheduler based where MapReduce workloads were classified into three types based on their CPU and I/O utilization under heterogeneous workloads. A network of queues model was proposed in [6] to model MapReduce and it was evaluated via simulation. Only the execution time of MapReduce jobs with varying cluster size was estimated. While a closed queuing networks model was proposed in [7] to model the map phase. More specifically, a mathematical model was constructed for predicting the execution time of the map phase of MapReduce single class jobs. The model results were validated by experiments on a single as well as a 2-node Hadoop environment.

The work in [3] presented an analytical model based on finite queueing system M/G/1/K to model MapReduce algorithm and to determine, at any time and under current workload conditions, the minimal number of cloud resources needed to satisfy the Service Level Objective SLO response time. The queueing model' server has three stages in tandem, namely: "job scheduling" delay centre, parallel n delay centre "VMs worker", and "result aggregating" delay centre. An analytic solution and a DES were developed to solve the system and the work considered only light- to- medium traffic. The work in [3] was extended in [1] where the model has a three- phase service time namely: delay centre scheduler (load balancer), parallel m delay centre mappers and parallel n delay centre reducers. An analytic solution and a DES were developed to solve the system that has three phases where the second and third phases are with m and n servers with exponential service rate respectively.

The above mentioned studies did not take into consideration the MapReduce operation under heavy traffic conditions. Moreover, to the best of the author's knowledge, the heavy traffic approximation for multiple server queueing system was not utilized to simplify the queueing model analysis. The paper aims to simplify a model similar to those proposed in [1] and [3] according to this theory. In this context, i.e., at heavy traffic, multiple server queueing systems can be approximated by single server queues with total mean service rate, $\mu_t = C\mu$. Using a heavy traffic approximation, multiple server systems can be approximated to single server queueing systems, as approved in [8],[9].

3. The Proposed Model

3.1 The Queueing Model

A queueing model for executing big data MapReduce tasks is proposed, as depicted in Figure 2. Three performance metrics are considered for the mean service time, mean queue length and the loss probability. In order to simplify the simulation of the MapReduce node model proposed in [1], the approximation based on the theory of heavy traffic condition is adopted where both parallel m and n delay centres were replaced in the proposed model by single delay server with $n\mu$ and $m\mu$ rates respectively.

The model assumptions and the analysis methods are justified as following: arrivals are assumed to have Poisson distribution, since it was shown that arrival of HTTP requests for documents under a heavy load closely follow the Poisson process (According to [1] and [3]). Service times are hyperexponentially distributed, as in reality, service times are not always exponential, but they are generally distributed. In this case, these models become difficult to be analytically solved when considering bursty traffic and non- Poisson arrivals. Therefore, simulation is an effective alternative to capture the system behaviour.

It is worth mentioning that the impact of buffer size variation on the node's performance was not taken into consideration in this study.

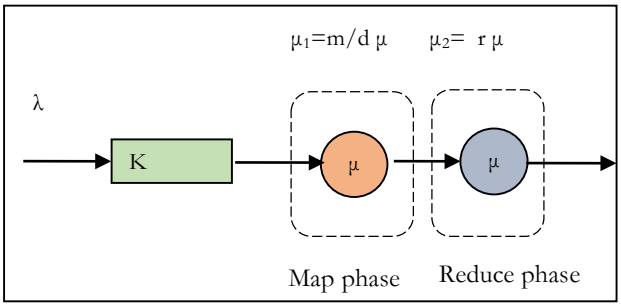

Figure 2: *M/G/1/K model for a big data server running m mappers and r reducers*

3.2 Performance Metrics

The performance metrics adopted in this study are briefly described below[10]:

1. Mean Response time (W)

It is the time a server takes to process a job (i.e., it is the time between receiving a job at the server node and its departure from the node).

2. Loss Probability (LP)

This is the percentage of jobs that get lost on arrival when they found the server queue full.

3. Mean queue length (Wq)

It is the number of jobs waiting in the queue to be served by the server.

These performance metrics are chosen to explicitly reflect the system behaviour when the MapReduce is overloaded in the event of queue saturation (at heavy traffic conditions).

4. Simulation and Numerical Results

4.1 Simulation Analysis

Discrete-event simulation was implemented using a java package to simulate M/G/1/K /FCFS for MapReduce process with m mappers and r reducers where m > r. The performance was assessed in terms of the MapReduce node's mean response time, mean queue length and loss probability. These performance metrics were compared to assess the effect of increasing the number of m and r on the performance.

where the service time has hypoexponential distribution. The simulation is implemented according to [11]. The built-in pseudorandom number generator was used to generate uniformly distributed random variables, U on [0,1] interval. (RVs) which were employed to generate exponential and hypoexponential RVs, are expressed in equations 1 and 2:

$$\text{Exponential RV} = \frac{-1}{\lambda} \cdot \ln(U) \quad (1)$$

$$\text{Hypoexpoential} = \sum_{i=1}^{n} \frac{-1}{\mu_i} \ln(U_i) \quad (2)$$

where λ is the mean rate of the exponential RV, μ_i is the mean rates for the hypoexponential stages of the RVs and U is a uniform RV.

since hypoexponential RVs are the sum of n exponential RVs. The rates of exponential RVs can be equal or different from each other. In the context of this work, the rates are assumed to be different and the simulation algorithm of 2-sytages hypoexponential RV is depicted in Figure 3.

Algorithm:
Generating 2-stages hypoexponential RV, X, using the inverse transform method to generate two exponential RVs, the following steps as followed:
Begin
Step 1: Input the value of the mean rate λ_1, λ_2 of the Exponential RV;
Step 2: Generate two uniform RVs $U_1[0,1]$, $U_2[0,1]$ and
Step 3: Let $X = \sum_{i=1}^{2} [-\frac{1}{\lambda_i} \ln (U_i[0,1])]$;
End.

Figure 3: *Algorithm of generating 2-stages hypoexponential RV*

In order to improve the accuracy of simulation output, the number of the simulated events was made 10^6.. The values of the simulation parameters are listed in Table 1.

Table 1: *The Parameters of The Simulation Experiments*

Parameter	***Value***
K	100
Λ	200-1400 job /sec
μ	1200 job /sec
μ_1	μ_1=m/d μ
μ_2	μ_2= r μ
D	5
M	6,9,12,15
R	2,3,4,5

4.2 Numerical Results

Figures 4-7 show the relations between the adopted performance metrics for the MapReduce as a function of the mean arrival rate in order to check the effect of increasing of mappers and reducers subject to heavy traffic conditions. While Figures 6 and 7 show mean response time and loss probability against the number of both mappers and reducers. Figure 4 to 6 illustrate the effect of the number of mappers and reducers on the MapReduce performance metrics.

Figure 4 depicts the mean response time. The higher the mappers and reducers the better the performance will be (with the lowest mean response time). Clearly, this improvement is achieved because more workers operate in parallel so that any incoming job that finds the first CPU core is busy will be more probably to receive service by other workers. Figure 5 shows the performance comparison in terms of loss probability. It is verified that the loss probability for m=15 and r=5 is much smaller than that of a m=6 and r=2. This is expected as the server capacity is around twice as the original one. As a result, the queue will have less number of jobs and this will reduce the possibility of being full that causing job loss. Figure 6 illustrates the mean queue length. It is obvious that the increase of m and r will delay the full occupation of the queue till the moment when the mean arrival rate = 800 job/sec and this is almost close to the theoretical value when the server utilization ρ=1. On the contrary, when m=6 and r=2 the queue is more probable to be full. Due to the low service rate which is a function of both m and r.

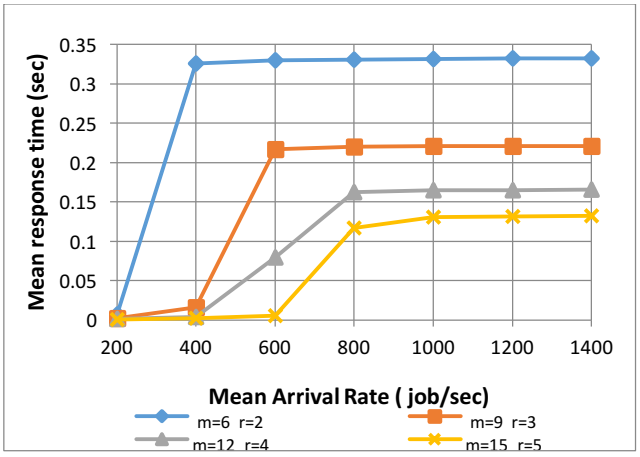

Figure 4: *Mean response time Vs mean arrival rate with varying mappers(m) and reducers (r)*

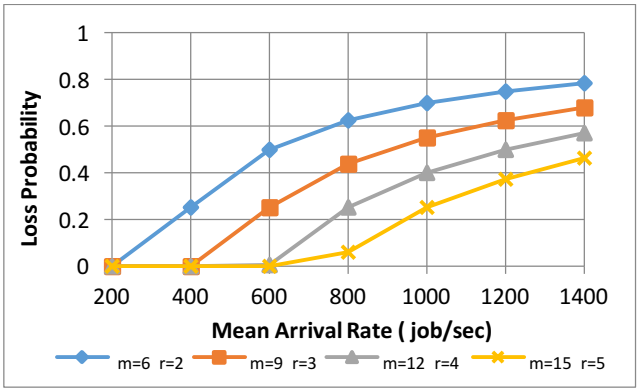

Figure. 5: *Loss probability Vs mean arrival rate with varying mappers(m) and reducers (r)*

In order to make a decision on the number of mappers and reducers according to specific workload conditions, Figure 7 can be employed for this purpose by taking the mean response time as a key performance metric. When the sum of m and r are equal to 12, for example, the corresponding mean response time is around 0.22 sec.

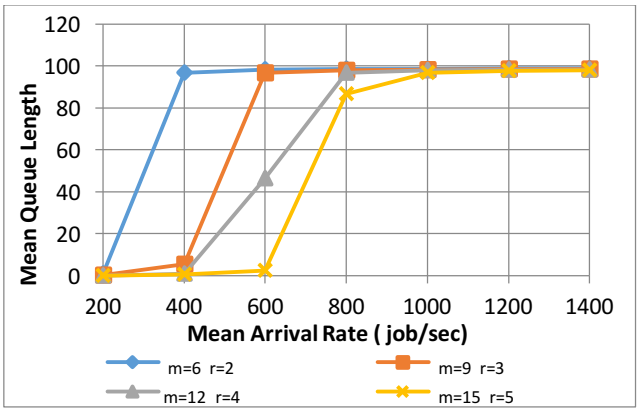

Figure. 6: *Mean queue length Vs mean arrival rate with varying mappers(m) and reducers (r)*

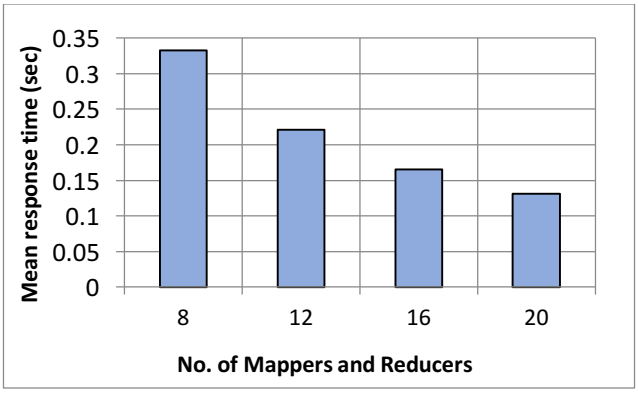

Figure. 7: *Mean response time Vs. the sum of mappers and reducers (m+r) (when λ =1200 jobs/ sec)*

5. A Model Extension Employing a QNM with Multiple Servers

In this section, a more general queueing model, depicted in Figure 8, is suggested to examine the performance of Map-reduces algorithm for a cluster has N servers utilizing the model proposed in [12]. This involves the use of the universal maximum entropy (ME) algorithm for arbitrary open QNMs with finite capacity (c.f., [8],[9]).

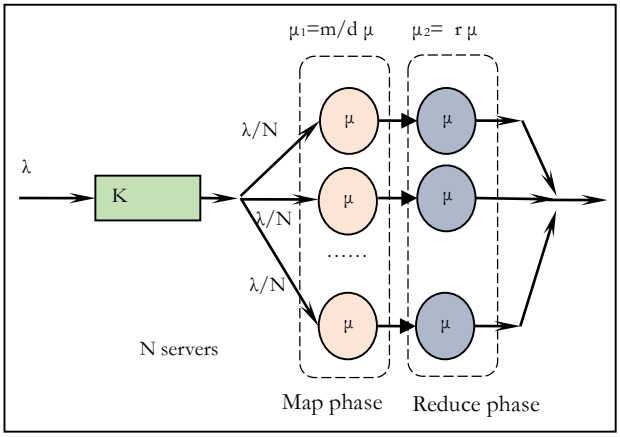

Figure. 8: *Possible extension of the proposed model*

6. Conclusions

The performance for big data map reduce process is investigated by assessing the impact of the number of mappers and reduces on the MapReduce system performance, in terms of the node mean response time, mean queue length and the loss probability, when fixing the buffer size. The results showed that the increase of the mappers and reducers in big data cluster node improve the overall performance. This improvement was quantified via DES.

The proposed model can be used as an effective tool to determine the number of mappers and reducers to meet specific operating conditions. This study is an attempt towards investigating the performance of MapReduce procedure using a simple QNM as a quantitative

tool for the design and possible development of MapReduce process under heavy traffic workloads.

Extensions of the work may address the modeling of a big data cluster that composed of N servers running MapReduce on parallel. In this context, one or more classes can be taken into consideration to reflect realistic operating conditions. The accuracy of the proposed queueing model can be improved by taking into account the scheduling delay at the load balancer, as suggested in [1],[3].

References

[1] K. Salah, J. M. A. Calero, "*Achieving Elasticity for Cloud MapReduce Jobs*", IEEE International Conference on Cloud Networking (CloudNet 2013), 2013, pp 195-199.

[2] J. Dean and S. Ghemawat, "*MapReduce: Simplified Data Processing on Large Clusters*", 2004, pp. 1-13.

[3] K. Salah, "A Queueing Model to Achieve Proper Elasticity for Cloud Cluster Jobs," Proceedings of the 2013 IEEE Cloud Conference, Santa Clara, CA, June 27, 2013, pp. 755-761.

[4] S. Zapechnikov, N. Miloslavskaya and A. Tolstoy "Analysis *of Hypoexponential Computing Services for Big Data Processing*", 2015 3rd International Conference on Future Internet of Things and Cloud, pp 579-584.

[5] C. Tian, H.Zhou,Y. He , L. Zha, "*A Dynamic MapReduce Scheduler for Heterogeneous Workloads*", 2009 Eighth International Conference on Grid and Cooperative Computing

[6] S. Ahn and S. Park, "An *Analytical Approach to Evaluation of SSD Effects under MapReduce Workloads*", Journal of Semiconductor Technology And Science, Vol.15, No.5, October, 2015 Issn(Print) 1598-1657.

[7] S. Bardhan, D. A. Menasce, "Queuing Network Models to Predict the Completion Time of the Map Phase of MapReduce Jobs", In the Proc. of International Computer Measurmenr Group Conference, Las Viga, NV, 3-4th of December 2012, PP. 146-153.

[8] D. D. Kouvatsos, I. U Awan, "*Entropy Maximization and Open Queueing Networks with Priorities and Blocking*", Elsevier, Performance Evaluation, 51, 2003,pp.191-227.

[9] Y. Li, "*Performance Modelling and Evaluation of Cellular Networks*", Phd thesis, University of Bradford, UK (2005)

[10] D. D., Chowdhury, "*High Speed LAN Technology Handbook*", Springer, USA (2000).

[11] A. M. Law , W. D. Kelton, "*Simulation Modelling and Analysis*", Mc Grow-Hill , 3rd ed., 2000

[12] S. El Kafhali, K. Salah Stochastic," *Modelling and Analysis of Cloud Computing Data Cente"*, IEEE, 2017.