OPEN ACCESS

Global Journal of Research in Humanities & Cultural Studies
ISSN: 2583-2670 (Online)
Volume 03| Issue 02 | March-April | 2023
Journal homepage: https://gjrpublication.com/gjrhcs/

Original Research Article

MapReduce Model: A Paradigm for Large Data Processing

*Kabiru Dahiru Ibrahim[1], Ibrahim Mahdi Muhammad[2], Yusuf Idris[3], Adamu Bello[4], Kassim Sulaiman Abubakar[5]

[1, 2, 3] Department of Computer Engineering, school of technology Kano State Polytechnic, Nigeria
[4,5] Department of Electrical Engineering, school of technology Kano State Polytechnic, Nigeria

DOI: 10.5281/zenodo.7819069 Submission Date: 28 March 2023 | Published Date: 12 April 2023

Corresponding author: Kabiru Dahiru Ibrahim
Department of Computer Engineering, school of technology Kano State Polytechnic, Nigeria

Abstract

MapReduce is a programming paradigm that enables massive processing of large amount of data over several machines in a cluster of commodity computers. It is fault tolerant and scalable hence suitable for cloud computing applications. This paper come up with a second order nonlinear model of the MapReduce using experimental data collected from Grid5000 experimental tested accessed from the local machine using Linux Secure Socket Shell protocol (SSH) as a command line interphase. System identification was performed on the collected data using MATLAB toolbox. The nonlinear model obtained was linearized and discretized using numeric optimization techniques to obtain the continuous transfer function. Within the limits of operating points, the model shows a perfect tracking and good representation of the dynamics of the real system and hence can be suitable for applying control laws.

Keywords: Mapreduce, Model, System identification, cloud computing, Hadoop

INTRODUCTION

MapReduce is a programing model and an associated implementation for processing and generating large data sets [1]. MapReduce handles the need for parallelization of a very large amount of date over several machines that are in a large cluster. It is a massive processing paradigm (Zhang, Cheng, & Boutaba, 2010) [2],[3] for the parallel processing of data which is distributed over a commodity cluster. MapReduce achieved the processing ability of vast amount of data over shorter period of time. Its mode of operation is based on, a map function and a reduced function [1]. MapReduce large data processing ability offers developers a means to transparently handle data partitioning, replication, task scheduling and fault tolerance computing on a cluster of commodity computers [4].

MapReduce process a very large amount of data, as a result, failures are recurrent scenarios in such process. MapReduce have a high performance (Khezr & Navimipour, 2017) [5] and also proved to be fault-tolerant [6]. It works on a master slave (worker) form. Supposed a map task is to be executed by a particular slave (X) but failed, it will then be assigned later to another slave (Y) by the master. All other slaves executing the reduced task will be updated about the re-execution of the failed task. Reduced task that could not be read from X as a result a failure will now be read from Y. Figure 2.1 shows the execution overview (Iosup, et al., 2017) [6]. The basic action performed by the map script is to transform the data into key/value pairs while the reducer basically accumulates (Zhang, Cheng, & Boutaba, 2010) [2].

The input key/value pairs [1] split the data into different segments based on the assigned keys and values that are sent into different machines in parallel over the cluster, each of the machines runs the data mapped to it, the immediate output generates two scripts; the map script and the reduced script [3]. The MapReduce architecture have three phases; Mapper, Reducer, and Shuffler (Khezr & Navimipour, 2017) [5].

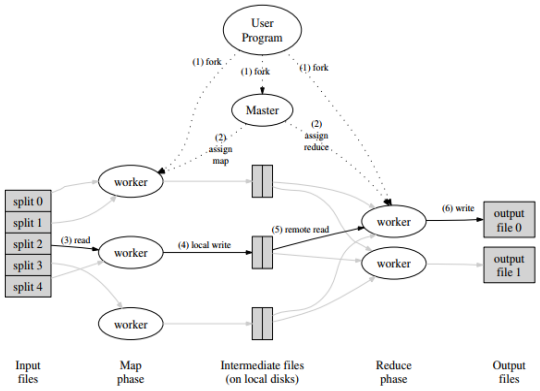

Figure 1.1: Execution Overview

Map functions process the input key/values pairs assigned to it to a set of immediate key/value pairs. Shuffling is the mapping of the immediate output of the map task to where they are required by reducers. Finally, a set of immediate values having a common key are reduced into a single reducer in the reducer phases [3].

The world largest companies (Yahoo, Facebook, Amazon, and IBM) use the MapReduce model as a tool for large data processing by implementing Hadoop, produced by Apache software foundation which is an open-source implementation of MapReduce (Khezr & Navimipour, 2017) [5]. Hadoop allow applications to run on clusters [3]. It has the ability to store large amount of data, replicate the data into blocks and assign it to multiple host thereby enhancing its reliability. It is built up of Hadoop Distributed File System (HDFS) that consist of NameNode (to manage file system metadata) and a DataNode that stores the actual data (Khezr & Navimipour, 2017) [5]. The architecture is shown in figure 1.2. HDFS therefore store the input, intermediate data and the output data of Hadoop MapReduce jobs [4].

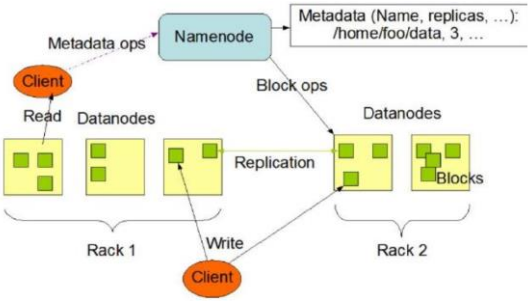

Figure 1.2: HDSF-Architecture [5].

The MRBS aims at some key objectives such as Multi-Criteria analysis that focus on the performance metrics of the MapReduce, Diversity that allows it to run on different types of applications and Usability which makes it user friendly [4]. The workloads emulated by MRBS are designed to cover five application domains: recommendation systems, business intelligence (BI), bioinformatics, text processing and data mining (Berekmeri, Serrano, Bouchenak, Marchand, & Robu, 2014) [7]. The MRBS execution statistically displayed the experimental runtime analysis as, among which include concurrent clients, warm up time, run time, throughput, cost, phase start, phase end and response time. The architectural representation of the MRBS is shown in figure 1.3.

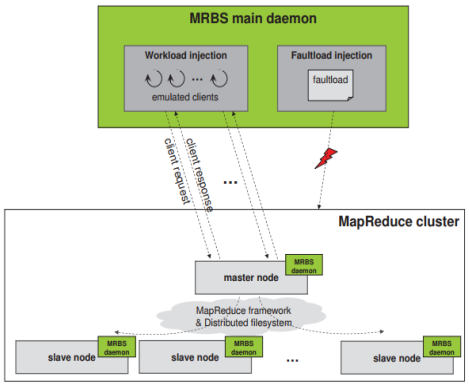

Figure 1.3 MRBS architecture [4]

Jeffrey Dean and Sanjay Ghemawat highlighted the applications of MapReduce in large cluster commodity machines for parallel computing [1]. Khezr and Navimipour used MapReduce in the optimization algorithms, presenting its application to ant colony algorithm, cuckoo search, and particle swarm optimization (PSO) [8]. Berekmeri in his work was able to come up with a first order model of the MapReduce paradigm [1] using system identification, which is a simple linear model with delay. In this paper, a second order non-linear model of the MapReduce was proposed taking into consideration the availability and fast response time as applied to cloud services. The proposed model shows a good tracking and representation of the real system and at the same time, it gives a higher order compared to the first order model identified in (Berekmeri, Serrano, Bouchenak, Marchand, & Robu, 2014) (Cerf, Berekmeri, Robu, & Nicolas Marchand, 2017) [7], [9], [10], hence more robustness to disturbance and uncertainties in the system is expected.

Experimental Cloud Environment

The experiment is deployed on-line over the Grid5000 cluster: a large-scale and highly reconfigurable grid experimental testbed for large data cloud computing. The nodes used for the experiments have specifications as highlighted table 2.1:

Quad Core	RAM	Disk	Network
CPU 2.53 GHz	15GB	298 GB	InfinitBand 20GB

Table 2.1: Nodes Configuration

The open source implementation of MapReduce framework applied is Apache Hadoop and the high level MRBS. The client interaction emulates a typical business-oriented query run over the cloud with data as large as 10GB. A data intensive Business Intelligent workload is selected. The BI benchmark consists of a decision support system. Apache Hive is deployed on top of Hadoop, which converts SQL like queries to a series of MapReduce jobs and subsequently generate the client interactions. To minimize network skews, the required nodes were reserved in the cluster on the same switch.

The nodes were accessed from the local machine using the SSH internet protocol. The experimental set-up is as shown in figure 2.1. The SSH connects the local computer to the remote server via the master node of the MapReduce which executes the MRBS benchmark tool.

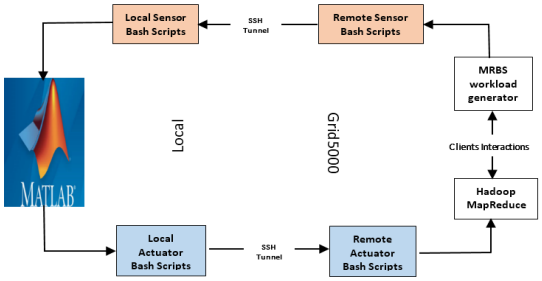

Figure 2.1 Experimental set-up [7]

Software lacks physical laws and mathematical equations and at the same time subjected to frequent updates, it was hence assumed that the captured dynamics are of high-level performance metrics that are not affected during such updates. The map function (mappers) and reduce function (reducers) were considered. The dynamic model of the MapReduce is as shown in figure 3.1 (Berekmeri, Serrano, Bouchenak, Marchand, & Robu, 2014) [7],[9].

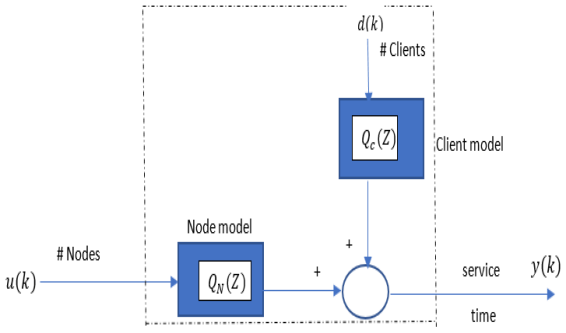

Figure 3.1: MapReduce Dynamic Model [9]

The output, average service time $y(k)$ measured at the k^{th} interval of time has effect on both mappers and reducers. The control input $u(k)$, represents the number of nodes in the cluster. The variable $d(k)$, represented by the number of clients, which changes in the course of running the experiments on the MapReduce job, is considered as the measured disturbances. The output is represented as shown in equation 3.1 for linear time invariant systems using the discretized form the nonlinear model and the superposition theorem, taking into account the approximations at the operating points.

$$y(k) = Q_N(Z)u(k) + Q_c(Z)d(k) \quad 3.1$$

$Q_N(Z)$ and $Q_c(Z)$ represent the discrete time model between service time and the number of nodes and clients respectively. Equation 3.2 gives the linearized form of the discretized transfer function for the second order model identified with delays that prevents oscillations due to unnecessary nodes additions while at the same time taking care of the Hadoop job tracker's action.

$$y(Z) = Z^{-T_C}\frac{p_x}{(Z+m_n)}d(K) + Z^{-T_N}\frac{p_y}{(Z+n_n)}U(K) \quad 3.2$$

The discrete time model of the transfer function as represented in equation 3.2 above have coefficients p_x, m_n, p_y and n_n,which represents the mathematical model parameters returned automatically from the system identification toolbox of MATLAB. The inputs were considered by varying the on the number of nodes and clients in the course of running the experiment to measure the corresponding output data (service time). The second-order nonlinear model identified as is a continuous transfer function is discretized taking a sampling period of 30 seconds using Tustin bilinear transformation. The desired models identified have an algorithm whose structures take the form;

$$y(k+1) = -\sum_{i=1}^{m} x_i.y(k+1-i) + \sum_{i=0}^{n} y_i.u(k-T-i) \quad 3.5$$

Taking the unknown parameters within the vector Θ, the equation is vectorially represented as;

$$\Theta = [x_1, x_2, x_3, \dots, x_m \;\; y_1, y_2, y_3, \dots, y_n] \quad 3.6$$

$$\emptyset(K) = [-y(k), \dots, y(k+1-n) \;\; u(k-T), \dots, u(k-T-m)] \quad 3.7$$

$$y(k+1) = \Theta^T.\emptyset \quad 3.8$$

To minimize the objective function, applying prediction error algorithms (PEM), the Quasi Newton method for numeric optimization technique was used with cost function;

$$J = \min_{\Theta} \sum_{k=1}^{N} e^2(k) \quad 3.9$$

The error (e) between the predicted output $y_n(k)$ and the measured output $y(k)$ for N size of data is given by;

$$e = y_n(k) - y(k) = y_n(k) - \Theta^T.\emptyset(k-1) \quad 3.10$$

Within the limits of the operating regions in the compensatory path $G(Z^{-1})$, varying the number of nodes as the takes control input to measure the service time, the identified model with coefficients x_i, y_j and the delay T e the form;

$$y_N(k) = -\sum_{i=1}^{N} x_i\, y_N(k-i) + \sum_{j=0}^{N} y_j\, u(k-T-j) \quad 3.11$$

For the direct path (Z^{-1}) which introduces the measured disturbance by varying the number of clients to measure the service time, the identified model coefficientsx_q, y_r and a delay T have the form;

$$y_N(k) = -\sum_{q=1}^{N} x_q\, y_C(k-q) + \sum_{r=0}^{N} y_r\, d(k-T-r) \quad 3.12$$

The overall transfer function of the second order model taking into account the delays for both the direct and the compensatory paths is given in the equation below,

$$D(Z^{-1}) = z^{-4}\frac{0.1927Z^{-1}}{1-1.629Z^{-1}+0.629Z^{-2}}d(k) \quad 3.13$$

3.13

$$G(Z^{-1}) = z^{-4}\frac{-0.5641+Z^{-1}-o.4353Z^{-2}}{1-1.836Z^{-1}+0.3262Z^{-2}}u(k) \quad 3.14$$

RESULTS

The Disturbance Scenarios

The disturbance scenarios taken into considerations shows the relationship between the clients and nodes variation against the service time as seen in the figure 3.3, and it represent the data used for the model identification. It can be seen that

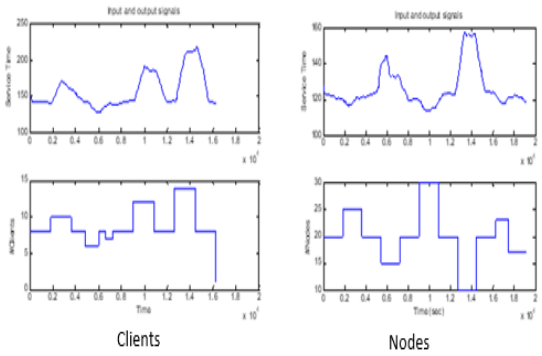

Figure 4.1: Disturbance Scenarios

As seen from the client's variation scenario, the change in clients is limited within the range of 0 to 15, while for the node's variation is within the limit of 30 as the maximum value so as maintain the validity of the model around the operating point defined by the number of nodes and clients also taking our threshold into consideration.

Clients Variations Model

The nonlinear ARX model identification from the MATLAB toolbox was used, several permutations of the input unit, output unit and delay were taken into considerations until a good model was obtained for both the self-validation and cross validation of data.

- **Self-Validation**

As seen from figure …. The nonlinear ARX model for the self-validation data fits to 75.48% showing a good tracking of the real system. It is also observed that the dynamics of the actual system is well represented, hence the model can be assumed to be a good model subject to verification on a different set of data for cross validation.

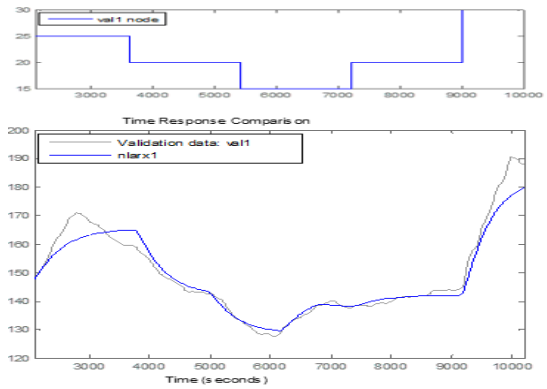

Figure 4.2: Client Self Validation.

Cross Validation

As seen from figure …, after cross validation, the nonlinear ARX model was able to show a good tracking having a fit to working data of 50.25% with an exact representation of the real. It can therefore be concluded that the nonlinear ARX model is a good model can the represent (within the limits of operating regions defined by the number of nodes and clients) the real system. A direct relationship is respected between the service time and the disturbance. At maximum disturbance of 14 clients, the service time increases to as high as 218 seconds.

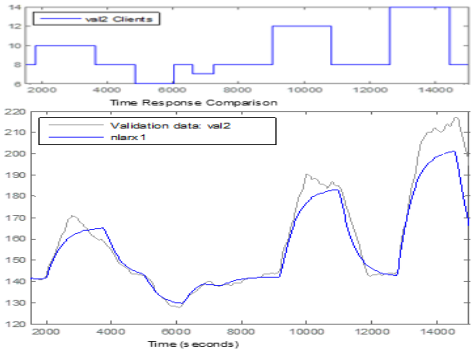

Figure 4.3: Clients Cross Validation

Nodes variation model

A similar approach was adopted for the clients' model identification, in these scenarios, the Hammerstein Wiener nonlinear model with input dead zone and output piecewise form of nonlinearities gives the best model.

- **Nodes self-validation**

The self-validation model fits to 76.57% at the same time captured the dynamics of the real system. It can be seen that as more resources are introduced (nodes=12), the service time significantly reduces to as low as 122 seconds, thereby respecting the expected inverse relationship.

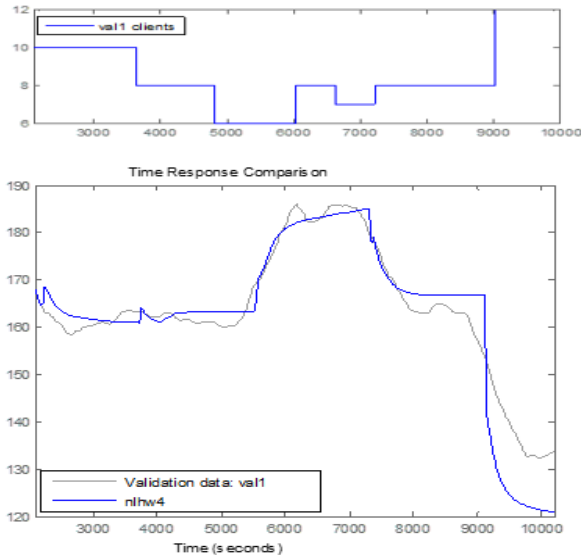

Figure 4.4: Node Self-Validation.

- **Node Cross Validation**

After cross validation on a different set of data for the node disturbance, the Hammerstein Wiener nonlinear model fits to 71.46% capturing the system dynamics with reasonable tracking leading to conclude on the validity of the model.

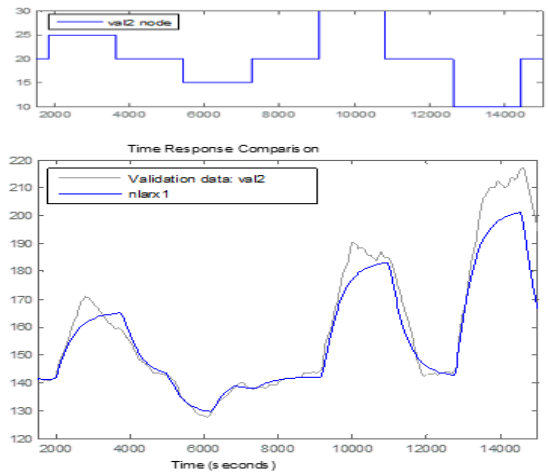

Figure 4.5: Node Cross Validation.

CONCLUSION

This paper studied the model of MapReduce as applicable to large scale distributed cloud computing through the system identification toolbox in MATLAB. First, experimental data were collected from the Grid5000 flatform using Linux, the nonlinear system identification tool box was used to propose a second order nonlinear model. The model was linearized around the operating region (defined by the number of nodes and clients) and discretized using the numeric optimization techniques (Quasi Newton method). The model obtained captured the dynamics of the real system with a good tracking performance and hence can be chosen to develop control laws.

Acknowledging Grid5000.

Experiments presented in this work were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations
(see https://www.grid5000.fr)

REFERENCES

1. J. Dean and S. Ghemawat, “MapReduce : Simplified Data Processing on Large Clusters,” pp. 1–13.
2. Qi Zhang · Lu Cheng · Raouf Boutaba "Cloud computing: state-of-the-art and research challenges".
3. D. Maclean, “A Very Brief Introduction to MapReduce,” pp. 2–4, 2011.
4. S. Bouchenak, “MRBS : A Comprehensive MapReduce Benchmark Suite Amit Sangroya , Dami ´ an Serrano , Sara Bouchenak LIG Technical Report (RR-LIG-024) Grenoble , France , 2012,” 2012.
5. S. Nima, K. Nima, and J. Navimipour, “MapReduce and Its Applications , Challenges , and Architecture : a Comprehensive Review and Directions for Future Research,” no. August, 2017.
6. H. S. Bhosale and P. D. P. Gadekar, “A Review Paper on Big Data and Hadoop,” vol. 4, no. 10, pp. 1–7, 2014.
7. Alex Iosup, Xiaoyun Zhu, Arif Merchant, Eva Kalyvianaki, Martina Maggio, Simon Spinner, TarekAbdelzaher, Ole Mengshoel, Sara Bouchenak, " Self-Awareness of Cloud Applications".
8. M. Berekmeri, “Modeling and control of cloud services : application to MapReduce performance and dependability To cite this version : HAL Id : tel-01278177 La Modélisation et le Contrôle des services BigData : Application à la Performance et la Fiabilité de MapReduce,” 2016.
9. S. N. Khezr and N. J. Navimipour, “MapReduce and its application in Optimization Algorithms: A comprehensive Study,” Majlesi Journal of Multimedia processing, vol. 4, no. 3, pp 1-33, 2015.
10. M. Berekmeri et al., “A Control Approach for Performance of Big Data Systems To cite this version : HAL Id : hal-00980372 Big Data Systems,” 2015.
11. S. Cerf, S. Berekmeri, and S. Berekmeri, “ScienceDirect Adaptive Feedforward and Feedback Control for Adaptive Feedforward and Feedback Control for Adaptive Feedforward Feedback Control for,” IFAC-PapersOnLine, vol. 50, no. 1, pp. 5504–5509.
12. S. Rachad, B. Nsiri, and B. Bensassi, “System Identification of Inventory System Using ARX and ARMAX Models,” vol. 8, no. 12, pp. 283–294, 2015.

CITE AS
Kabiru D. Ibrahim, Ibrahim M. M, Yusuf Idris, Adamu Bello, & Kassim S. A. (2023). MapReduce Model: A Paradigm for Large Data Processing. Global Journal of Research in Humanities & Cultural Studies, 3(2), 1–7.
https://doi.org/10.5281/zenodo.7819069