Applying MVC Data Model on Hadoop for Delivering the Business Intelligence

Walisa Romsaiyud
Graduate School of Information Technology,
Siam University, 38 Phetkasem Rd.,
Bangkok, 10160, Thailand.
walisa.rom@siam.edu

Abstract— **Model-View-Controller (MVC) pattern is very useful and popular design pattern that separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes. The model acts as the domain that software is built around, the view as the visual representation of a model. It is usually the resulting markup that the framework renders to the browser and the controller responsible for processing input, acting upon the model, and deciding on what action should be performed, such as rendering a view or redirecting to another page for supporting a business models. Hadoop MapReduce paradigm applied for collected the several datasets for generating the large volume of data in batch processing. In this paper enhanced the Hadoop MapReduce for processing large scale and different data set across platform with MVC pattern. In general, the system consists of five mainly methods: 1.) Collected data sources from different platforms such as the registrar system, web mail, web blogs, news and the English Wikipedia that are included in the data file system at local system, 2.) Generate the data into a map slot from input format, 3.) calculate the jobs from the business logic, 4.) Create a data model based on data entity framework, and 5.) Display an user interface for supporting the business interfaces.**

Keywords – Hadoop; MapReduce; Model-View-Controller (MVC); Data Entity Framework;

I. INTRODUCTION

Model–view–controller (MVC) [1], [2], [3] is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. In [4], they proposed a novel approach and an algorithm for exposing MVC functionality via SOAP Web services and implementing the application on a PHP MVC framework. From [5], they applied MVC architecture on a distributed to design of connector in distributed MVC system to support the loosely coupled, in [6] they applied a MVC for the teaching course "software design and development" using the methods of case analysis and group discussion, then they showed the case on ATM simulation-based system. In [7], they integrated platform of education development according to a network teaching platform's problem of current and proposed a new three major MVC structure and network integration platform's system structure by using functional block diagram for representation include teaching organization management, online communication, online assessment, online management and download center. However, the PHP programming applied in many researches such as in [8], introduced a lightweight Model-View-Controller-liked (MVC like) for optimizing the programming of PHP on abstracted simple document of classes such as model, view and controller function and tested and verified to be feasible when plug-in. In [9], they proposed an adaptive IC testing on "Model-View-Controller (MVC) for increasing the potential and improve the engineering productivity for analysis. As present, the data file size is increasing and making it compliant to process and execute. Then many researches or papers applied a Hadoop. Hadoop [1], [10]-[11] is a framework that enables the distributed batch processing of large volume of datasets on clusters of commodity hardware. It can be to scale up from a single machine to multiple machines. The Hadoop Distributed File System (HDFS) [11] as a part of Hadoop designed for split large data file into smaller blocks which are managed by different nodes in the cluster. In addition to this each block is replicated across several machines, so that a single machine failure does not result in any data being unavailable. Furthermore, blocks fit well with replication for providing fault tolerance and availability. MapReduce is the programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop. Many new researches and products based on Hadoop and MapReduce technologies that have emerged in recent years. The sector and associated programming framework or Sphere proposed in [12] for design and implementation of a distributed file systems to improve the execution time for processing a very large data sets in data locality. In [13], they proposed second algorithms for increasing OC (IOC) algorithm for feature extraction and second algorithm is an orthogonal centroid feature selection (OCFS) method for increasing the effectiveness and efficiency when compared with others. In [14], they used Hadoop, Hbase, Hive on data warehouse for supporting OLAP analysis. However, in this paper, we applied the collections of several datasets that large volume of data and loaded into data file system based on Hadoop. In the Hadoop, we used MapReduce functions for calculating the functions from users' demand. On top or in the final part, using MVC pattern for generating the results likes dynamic data model and user interface depended on user requirements and business logic in each departments or organizations.

The remainder of this paper is organized as follows; in the following section II presents a brief overview of related work. Subsequently, in section III describes an overview of the architecture and its components. Then, in section IV

978-1-4799-8027-7/14/$31.00©2014 IEEE

discusses the experimental results of the evaluation. Finally, concludes the study in section V.

II. BACKGROUND

This section presents the three cores that related with our work is as follows: The MVC pattern, Hadoop MapReduce Framework and Repository Pattern.

1.1 The MVC pattern

The Model-View-Controller or (MVC) pattern [15]-[17] originated in the Smalltalk development community in the 1970s, although it was popularized for use on the web with the advent of Ruby on Rails in 2003. The main elements of MVC consist of three components as following;

1) The model—the domain that software is built around. In some contexts, the term model might refer to a view-specific model—a representation of the domain for the specific purpose of being displayed in the user interface.
2) The view—the visual representation of a model, given some context. It's usually the resulting markup that the framework renders to the browser, such as the HTML representing the blog post.
3) The controller—the coordinator that provides the link between the view and the model. The controller is responsible for processing input, acting upon the model, and deciding on what action should be performed, such as rendering a view or redirecting to another page. For the blog example, the controller might look up the most recent comments for a post (the model) and pass them to the view for rendering.

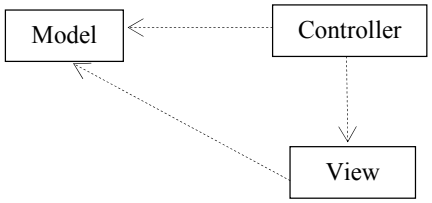

Figure 1. The components of the MVC pattern.

In Fig. 1, the components of the MVC pattern [15]-[16] are illustrated. The controller receives the user input, constructs the appropriate model, and then passes it to the view. Both the controller and the view have a dependency on the model, but the model itself is kept ignorant of the controller and view.

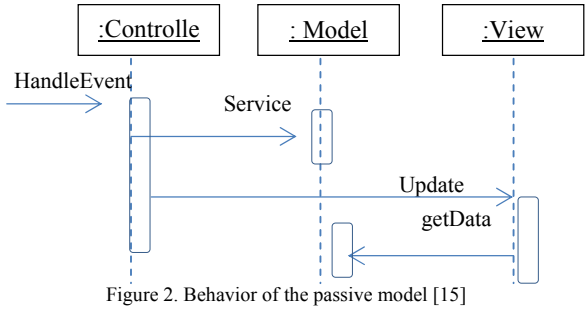

Figure 2. Behavior of the passive model [15]

As illustrated in Fig. 2, in this paper we employed one controller in such a way that it manipulates the model exclusively. The controller modifies the model and then informs the view that the model has changed and should be refreshed. The model in this scenario is completely independent of the view and the controller, which means that there is no means for the model to report changes in its state. The browser displays the view and responds to user input, but it does not detect changes in the data on the server. Only when the user explicitly requests a refresh is the server interrogated for changes.

1.2 Hadoop MapReduce Framework

The Apache Hadoop software library [18] is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures as shown in Fig. 3 and Fig. 4.

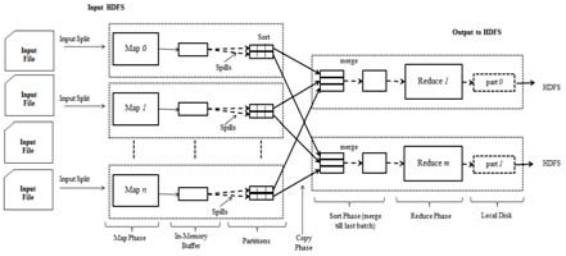

Figure 3. The Hadoop MapReduce Architecture [19]-[20]

As illustrated in Fig. 3, Hadoop breaks the input data into multiple data items by new lines and runs the map function once for each data item, giving the item as the input for the function. When executed, the map function outputs one or more key-value pairs. Hadoop collects all the key-value pairs generated from the map function, sorts them by the key, and groups together the values with the same key. For each distinct key, Hadoop runs the reduce function once while passing the key and list of values for that key as input. The reduce function may output one or more key-value pairs, and Hadoop writes them to a file as the final result.

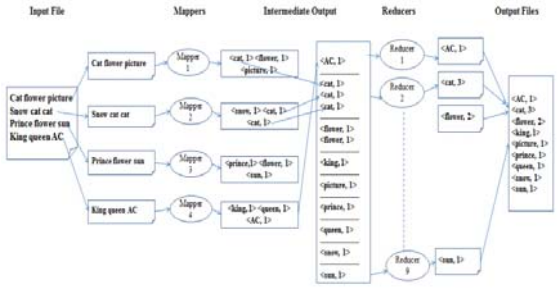

Figure 4. The WordCount Example in MapReduce Programming

As illustrated in Fig. 4, the WordCount example [20] uses MapReduce to count the number of word

occurrences within a set of input documents. Mainly steps as following; 1.) Partition input file into *M* partitions, 2.) Create *M* Map tasks, read *M* partitions in parallel and create intermediate pairs, 3.) Store them into local storage file into *M* partitions, 4.) Wait for all Map workers to finish, and partition intermediate pairs into *R* regions and 5.) Write the output of the reduced workers to file(s) as the output such as <*cat, 3*> means count word "cat" in the document is equal to 3 times.

1.3 Repository Pattern

The repository pattern [21]-[23] is a data access pattern that promotes a more loosely coupled approach to data access. Instead of having a controller or the business model that contains the data access logic, a separate class or set of classes called a repository takes responsibility of persisting the application's business model. The repository pattern nicely complements the key design principle of the MVC pattern separation of concerns. By using this pattern, that isolate the data access layer from the rest of the application and take advantage of the benefits of using POCO classes.

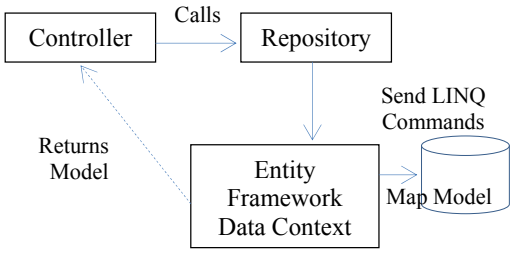

Figure 5. Interactions from controller with a repository

As illustrated in Fig. 5, the controller uses the injected repository to retrieve a list of auctions. Using this strategy, it is more convenient to test the controller that contacts with the repository.

III. THE MVC DATA MODEL ON HADOOP MAPREDUCE

This section describes major steps employed in the MVC data model so as to clearly identify the steps for processing the data and apply in the business services.

A. The Architecture of the Proposed Method

The system architecture comprises five main parts; Data Sources, Map Phase, Reduce Phase, Data Model and User's Interface. Further details are shown in Fig. 6.

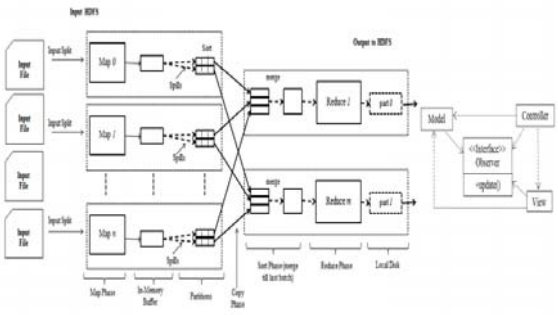

Figure 6. The Architecture Overview

As illustrated in Fig. 6, the system consists of five steps; 1.) Data Sources that is generated from 5 categories of data sources such as the registrar system, web mail, web blogs, news and the English Wikipedia that are included in the data file system at local system, 2.) Map Phase divides the input into ranges by the InputFormat and creates a map task for each range in the input. The JobTracker distributes those tasks to the worker nodes. The output of each map task is partitioned into a group of key-value pairs for each reduce by the default is 64 MB, 3.) Reduce Phase, collects the various results and combines them to answer the larger problem the master node was trying to solve. Each reduce pulls the relevant partition from the machines where the maps executed, then writes its output back into HDFS. Thus, the reduce is able to collect the data from all of the maps for the keys it is responsible for and combine them to solve the problem. 4.) Data Model creates an entity data model (EDM) [24]-[25] for collected data as a class file and 5.) User's Interface, for generating the application that runs on every browsers and devices such as mobile device in every platforms.

IV. EXPERIMENTAL EVALUATION

In this section, the evaluation approach and reports of the evaluation results are discussed in detail.

A. DataSet and Experimental Settings

This paper collected data from 5 mainly datasets; 1.) The registrar's information systems and student affairs information, 2.) Web mail, 3.) The web blogs, 4.) news and 5.) The English Wikipedia snapshot from January 2012 that the file size as 350 GB., 1 TB., 520 GB., 250 GB., and 12 TB. in orderly. The experiment was tested on HP - ProLiant DL580 G7 (4U). Our machine specification was as follows: the Intel Xeon Processor E7-4807, RAM 64 GB., and HDD 2x HP 300GB 6G SAS 10K 2.5in DP ENT HDD as the namenode (master machine). However, the commodity hardware is 40 machines as the Intel Xeon CPU, W3508 @ 2.40 ghz with 4.00 GB for RAM and 64-bit operating system as datanode (slave machines).

B. Experimental Results

For the evaluation process, this paper presented a MVC data model on Hadoop MapReduce with for delivering the business services depend on each requirement of users.

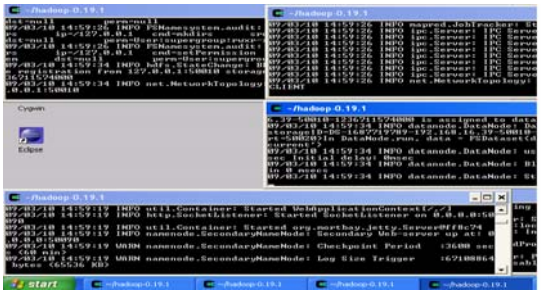

Figure 7. Start all components of Hadoop

As illustrated in Fig. 7, the configuration files used by the Hadoop cluster all the files in the Hadoop installation. To start a Hadoop cluster, we need to start both HDFS and Map/Reduce cluster simultaneously. Typically, we choose one machine in the cluster in order to act as the NameNode, as well as one machine to act as the JobTracker, exclusively. The rest of the machines act as both a DataNode and TaskTracker and they are referred to as slaves.

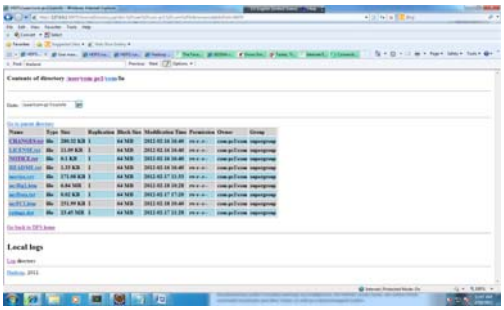

Figure 8. The contents of data files in directory of Hadoop

In Fig. 8, the contents of many data files in the data file system (that were stored in the directory) are illustrated: */user/com-pc1/com/in*. The system pulls the data from 5 categories into the batch paradigm.

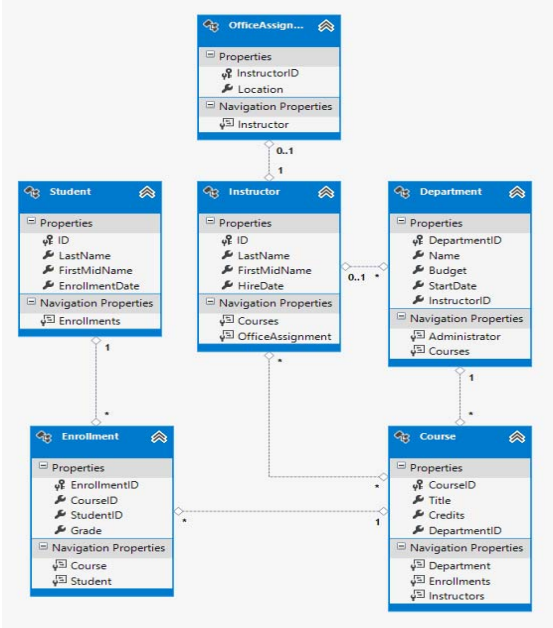

Figure 9. Data Entity Framework runs on Top of Hadoop

As illustrated in Fig. 9, from the collection of dataset that store in the data file, can generate the data model as data entity. The data entity retrieve the related data and link the data together, example the registrar system information that consist of 5 entities as office assignment, student, instructor, department, enrollment and course. In the case, developer can generate the class of data entities depends on requirement that data model based on class.

Figure 10. A screenshot of the student list display

As illustrated in Fig. 10, the user interface can be generated with html5 language and support in every browsers and devices platform. The student data display the items of name, enrollment date and number of course that linked the related data together and can modify and delete data.

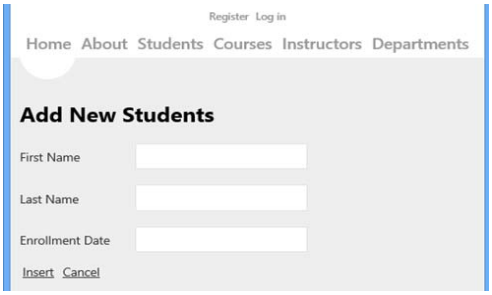

Figure 11. A screenshot of the modifying a new student

As illustrated in Fig. 11, the system support for inserting a new student in the system. The student entity consists of many properties such as first name, last name and enrollment date.

V. CONCLUSIONS AND FUTURE WORK

The main objective of this paper is to extract, transform and store the data collected from the data file system or dfs on Hadoop. The Hadoop stores the dataset as the batch oriented processing that has 5 categories file types. The key insight driving in this study is to use the MVC (Model-View-Controller) pattern for utilizing and applying the collected data depend on each user's requirement. Based on the experiment, the MVC patterns to support the separation of the components of programs (and re-use of the business logic across applications) were shown. The developers of User interface (UI) can focus exclusively on the UI screens without bogged down with business logic and easy on the business logic implementations, modifications, updations without concerning the look and feel and it has nothing to with business logic. Furthermore, it supports a parallel that MVC can build the classes, while the UI developers can involve in designing UI screens simultaneously, resulting the interdependency issues and time conservation.

REFERENCES

[1] Jeffrey Palermo, Jimmy Bogard, Eric Hexter, Matthew Hinze and Jeremy Shinner, *ASP.NET MVC 4 in Action*, Manning Publications Co, United States of America, 2012.

[2] Jon Galloway, Phil Haack, Brad Wilson and Scott Allen, *Professional ASP.NET MVC 4*, John Wiley & Sons, United States of America, 2012.

[3] "Model-View-Controller", *http://msdn.microsoft.com/en-us/library/ff649643.aspx,* May. 1, 2013.

[4] Yalezo S. and Thinyane M., "Architecting and Constructing an SOA Bridge for an MVC Platform", in *Proc. 4th Int. IEEE Conf. on Word Congress on Software Engineering (WCSE 2013)*, pp. 45-49, 2013.

[5] Mcheick H. and Yan Qi, "Dependency of Components in MVC distributed architecture", in *Proc. 24th Int. IEEE Conf. on Electrical and Computer Engineering (CCECE 2011)*, pp. 691-694, 2011.

[6] Tao Peng, Lianying Sun and Bao Hong, "Design and Implementation of ATM simulation system based on MVC pattern", in *Proc. Int. IEEE Conf. Educational and Information Technology (ICEIT 2010)*, pp. V1-328-V1-331, 2010.

[7] Ren Yongchang, Liu Zhongjing, Xing Tao and Chen Xiaoji, "Research on Structure and Functionality of Network Teaching Integrated Platform Based on MVC", in *Proc. Int. IEEE Conf. on Information Management, Innovation Management and Industrial Engineering (ICIII 2011)*, pp. 177-180, 2011.

[8] Guanhua Wang, "Application of Lightweight MVC-like structure in PHP", in *Proc. Int. IEEE Conf. on Business Management and Electronic Information (BMEI 2011)*, pp. 74-77, 2011.

[9] Kupp N. and Makris Y., "Applying the Model-View-Controller Paradigm to Adaptive Test", *IEEE Trans. on Design & Test of Computers*. Vol. 29. No. 1, pp. 28-35, 2012.

[10] "HDFS Federation", *http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/Federation.html ,* Feb. 1, 2012.

[11] Jason Venner, *Pro Hadoop*, Springer, pp. 177-205, 2009.

[12] Yunhong Gu. Robert Grossman, "Toward Efficient and Simplified Distributed Data Intensive Computing", *IEEE Trans. on Parallel and Distributed Systems,* Vol. 22, No. 6, pp. 974-984, 2011.

[13] Jun Yan, Benyu Zhang, Ning Liu, Shuicheng Yan, Qiansheng Cheng, Weiguo Fan, Qiang Yang,Wensi Xi, and Zheng Chen, "Effective and Efficient Dimensionality Reduction for Large-Scale and Streaming Data Preprocessing", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 18, No. 3, pp. 320-333, 2006.

[14] Lv L., Wang Y., Yan M. and Cheng L., "Design of Cloud data warehouse and its application in smart grid", in *Proc. Int. IEEE Conf. on Automatic Control and Artificial Intelligence (ACAI 2012)*, pp. 849-852, 2012.

[15] "Model-View-Controller", *http://msdn.microsoft.com/en-us/library/ff649643.aspx,* May. 1, 2013.

[16] "Learn About ASP.NET MVC", *http://www.asp.net/mvc,* May. 1, 2013.

[17] Nitin Sawant and Himanshu Shah, Big Data Application Architecture Q&A - A Problem-Solution Approach, Apress, 2013.

[18] Vignesh Prajapati, *Big Data Analytics with R and Hadoop*, PACKT Publishing, 2013.

[19] Wichian Premchaiswadi and Walisa Romsaiyud, "Optimizing and Tuning MapReduce Jobs to Improve the Large-Scale Data Analysis Process", *International Journal of Intelligent Systems*, Vol. 28, Issue 2, pp. 185-200, 2013.

[20] Alex Holmes, *Hadoop in Practice*, Manning Publications Co, United States of America, 2012.

[21] Brambilla M. and Origgi A., "MVC-Webflow: An AJAX Tool for Online Modeling of MVC-2 Web Applications", in *Proc. 8th Int. IEEE Conf. on Web Engineering (ICWE 2008)*, pp. 344-349, 2008.

[22] Minsu Choi, Jinsang Kim, Won-Kyung Cho and Jinwook Burm, "Area-efficient fast scheduling schemes for MVC prediction architecture", in *Proc. Int. IEEE Conf. on Circuits and Systems (ISCAS 2011)*, pp. 575-578, 2011.

[23] Liquan Shen, Zhi Liu, Ping An, Ran Ma and Zhaoyang Zhang, "Low-Complexity Mode Decision for MVC", *IEEE Trans. on Circuits and Systems for Video Technology*. Vol. 21. No. 6, pp. 837-843, 2011.

[24] Hyun Jung La and Soo Dong Kim, "Balanced MVC Architecture for Developing Service-Based Mobile Applications", in *Proc. 7th Int. IEEE Conf. on e-Business Engineering (ICEBE 2010)*, pp. 292-299, 2010.

[25] Yonglei Tao, "Component-vs. Application-level MVC architecture",in *Proc. 32nd Int. IEEE Conf. on Frontiers in Education)*, pp. T2G-7-T2G-10, 2002.