ISSN: 2053-3578 I.F. 12.34

MAPREDUCE SYSTEM MODEL BASED ON CLOUD COMPUTING INFRASTRUCTURE

Sh.Y.Isroilov

Samarkand branch of the Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, PhD

Abstract. *This paper proposes a fault-tolerant and scalable MapReduce system model based on modern cloud computing infrastructures. By leveraging cloud services such as table storage and message queues, the model eliminates single points of failure and improves data handling. The system enhances reliability, simplifies distributed programming, and efficiently processes large datasets using cloud-native parallel computing capabilities.*

Key words. *Cloud Computing, Cloud Services, Parallel Processing, MapReduce, Cloud Infrastructure.*

Introduction. The cloud is an elastic runtime environment that includes multiple stakeholders, providing a service that is scalable at multiple levels for a given level of service quality. More specifically, the cloud is a platform or infrastructure that enables the execution of code (services, applications, etc.) in a managed and elastic manner, where “managed” means that reliability is automatically ensured according to predefined quality parameters, and “elastic” means that resources are used according to actual current requirements, while all definitions are unconditionally respected. Elasticity includes scaling resources and data up and down, as well as load balancing data transfer with Google Docs. Because of this combinatory capability, these types are often also called “components”.

Cloud Services Infrastructure

Infrastructure as a Service (IaaS), also known as resource clouds, provide resources (managed and scalable) to the user as services. In other words, they essentially provide advanced virtualization capabilities. Accordingly, various resources can be provided via a service interface: data clouds and data warehouses provide reliable access to potentially dynamically changing data, adapting the use of resources to access requirements and/or quality definitions. Examples are Amazon S3, SQL Azure Compute Clouds, which provide access to computing resources, i.e. processors. Until now, such low-level resources could not be used on their own, so they are usually provided as part of a “virtualized environment” (not to be confused with PaaS below), i.e. hypervisors. Cloud providers therefore usually offer the ability

to provide computing resources, usually virtualized, on which cloud services and applications can run (i.e. direct access to resources, as opposed to PaaS, which offers a full suite of software for developing and building applications). IaaS (Infrastructure as a Service) offers additional capabilities beyond a simple computing service. Examples: Amazon EC2, Zimory, Elastichosts.

Platform as a Service (PaaS): Provides computing resources through a platform on which applications and services can be developed and deployed. PaaS typically uses specific APIs to control the behavior of the hosting system server, which iterates according to user requests (e.g., access speed). Since each provider provides its own API according to its key capabilities, applications developed for one cloud provider cannot be ported to another cloud host, but attempts are made to extend common programming models with cloud capabilities (e.g., MS Azure). Examples: Force.com, Google App Engine, Windows Azure.

Modern cloud computing service providers are expanding the composition and functionality of services that are provided together with the main product — computing power. In addition to the main computing resource rental service — Elastic Compute Cloud (EC2), the Amazon Web Service (AWS) service provider provides file data storage services, a block device rental service, a scalable non-relational database (DB), a relational DB, and other services totaling more than 20. These services, along with server rental services, represent the infrastructure of cloud services. Each of the service providers has its own cloud services, which differ significantly in composition and functions.

Despite the abundance and diversity of cloud services, they have some common features. The most popular service providers, such as Microsoft Azure and Amazon Web Service, have a number of services that are completely similar in function. These services form the infrastructure of modern cloud computing and are already an integral part of it. Table 1 contains a list and brief description of the infrastructure of cloud services Microsoft Azure and Amazon Web Service.

Table 1 - Amazon and Microsoft Cloud Services

	AWS	**Azure**
Servers	Elastic Compute Cloud (EC2)	Azure Compute
File storage service	Simple Storage Service (S3)	Binary Large Object (BLOB) Service
Structured data	SimpleDB	Table Service

ISSN: 2053-3578 I.F. 12.34

Messaging	Simple Queue Service (SQS)	Queue Service
Block devices	Elastic Block Store (EBS)	Azure Drive

Cloud services are aimed at simplifying the development of distributed systems. The use of cloud services in the development of systems seems to be a promising direction for the development of distributed computing. Today, there are ways to solve problems based on the MapReduce method on the cloud computing infrastructure. This opportunity is provided by the Amazon Elastic MapReduce service, which is part of the AWS services. But all existing implementations today use only cloud computing nodes that host existing MapReduce runtimes, such as Apache Hadoop. Existing approaches do not use additional cloud services. The implementation of the runtime environment for the MapReduce method using cloud services seems promising. This will increase the fault tolerance of systems and simplify their development.

The Google MapReduce programming model [1] is designed to process large amounts of data using massively parallel processing. The programming model is based on simple basic principles:

1. sequential traversal of input data;
2. calculation of key/value pairs from each block of input data;
3. grouping all intermediate values by key;
4. sequential traversal by group keys;
5. reduction of data dimensionality in each of the groups.

The programming model is simple, and at the same time parallelism is effectively supported. The programmer can abstract from the issues of parallel and distributed programming, since the MapReduce program execution environment itself takes care of the distribution of tasks across the cluster's computing machines, network performance, fault tolerance, etc. The basic principles of the programming model impose some restrictions on the tasks being solved, but it is suitable for solving many problems that cannot be solved on a single computing device due to the large size of the data (terabytes, petabytes): counting word occurrences in a large text, distributed grep, distributed sorting, etc.

Despite the simplicity of the MapReduce software model, its implementation is a complex technical task [2]. The main difficulty lies in ensuring fault tolerance of the system when operating in a distributed environment consisting of hundreds and possibly thousands of computing machines (servers). In addition, fault-tolerant mechanisms for grouping and

ISSN: 2053-3578 I.F. 12.34

transferring data between servers must be provided. Existing MapReduce implementations, such as Apache Hadoop, implement the specified capabilities and are complex, comprehensive software projects containing hundreds of thousands of lines of program code. This paper considers the issue of implementing the MapReduce method based on existing cloud computing technologies. Existing cloud computing services today offer not only computing resources (servers) on demand, but also additional cloud services that facilitate the development of distributed programs in the cloud. Cloud services today form the cloud computing infrastructure. The paper proposes a model of the MapReduce system on the infrastructure of modern cloud computing.

Analysis of MapReduce Method

Building an execution environment for the MapReduce method requires a detailed analysis of the method itself, for which it is convenient to use not a program model, but a data model that describes the input and output data of the Map and Reduce procedures .

MapReduce transforms a set of key-value pairs from one set (K^1 - set of keys, V^1 - set of values) into a set of key-value pairs from another, resulting set (K^2 - keys , V^2 - values). Moreover, the set of the second pairs is significantly smaller than the set of the first pairs. Often, the output of MapReduce is a single pair $\langle K^2, V^2 \rangle$.

the Map procedure is to directly transform a pair of first sets into a set of pairs of second sets $\langle K^1, V^1 \rangle \to \{\langle K^2, V^2 \rangle\}$. A typical example of such a mapping is the transformation of pairs <file name, file contents> into a set of pairs <word, word frequency in file> in the distributed word frequency counting problem or in the index construction problem.

The runtime must be able to run multiple computing nodes (servers) in parallel and sequentially run the Map procedure with the input data $\langle k_i^1, v_i^1 \rangle$.

MapReduce Runtime

Map procedure :

Input : $\{\langle k_i^1, v_i^1 \rangle\}$ - a set of pairs of initial keys and values, $k_i^1 \in K^1$ and . One pair is passed to one $v_i^1 \in V^1$ Map handler .

Output : $\{\langle k_i^2, v_i^2 \rangle\}$ - a set of pairs $k_i^2 \in K^2$ and $v_i^2 \in V^2$. One Map handler can produce a set of pairs as a result.

Reduce procedure :

ISSN: 2053-3578 I.F. 12.34

Input : $\left\{\left\langle k_i^2, \{v_j^2\}\right\rangle\right\}$ - a set of pairs of keys and a set of values from the resulting set, $k_i^2 \in K^2$ and $v_i^2 \in V^2$, where the keys in the pairs are not repeated, i.e. $\forall i, j: k_i^2 \neq k_j^2$.

Output : $\left\{\left\langle k_i^2, v_i^2\right\rangle\right\}$ - a set of pairs of keys and values from the result set, $k_i^2 \in K^2$ and $v_i^2 \in V^2$.

Figure 2.2 illustrates the sets processed by the Map (left) and Reduce (right) procedures. Here the keys are distinguished by colors.

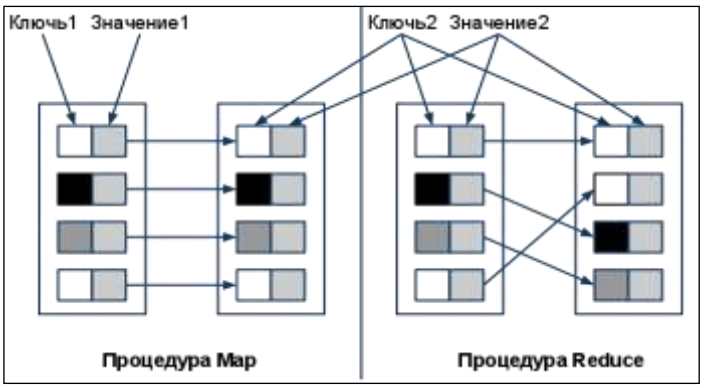

Figure 1 - Map and Reduce procedures

From the point of view of the implementation of the MapReduce runtime environment, the transition from the Map procedure to Reduce is important . We will call this transition the PreReduce procedure . Figure 1 - represents PreReduce as a transition from the left part of the image to the right part. In accordance with the semantics of the MapReduce method, the PreReduce procedure must group all intermediate values by keys from the set K^2 (item 3 in the MapReduce principles listed above). The algorithms for grouping keys depend on the way the data is presented and their location after executing the Map procedure . Thus, the PreReduce procedure is used not only when moving from Map to Reduce but also when saving the results of the Map procedure . The procedure for saving the results of Map is called EmitIntermediate , and it accepts a pair [1, 3] as input $\left\langle k_i^2, v_i^2\right\rangle$.

Some implementations of MapReduce use not the elementary Map-Reduce programming model , but a more generalized Split-Map-Shuffle-Sort-Reduce [42]. In it, grouping is performed in the form of two steps Shuffle and Sort , which can be changed by the user, unlike the traditional MapReduce method [1, 4].

The main difficulty of implementing the MapReduce runtime environment on cloud services infrastructure is the implementation of the grouping mechanism, i.e. the PreReduce

ISSN: 2053-3578 I.F. 12.34

step . The system model illustrated in Figure Map job and a Reduce job .

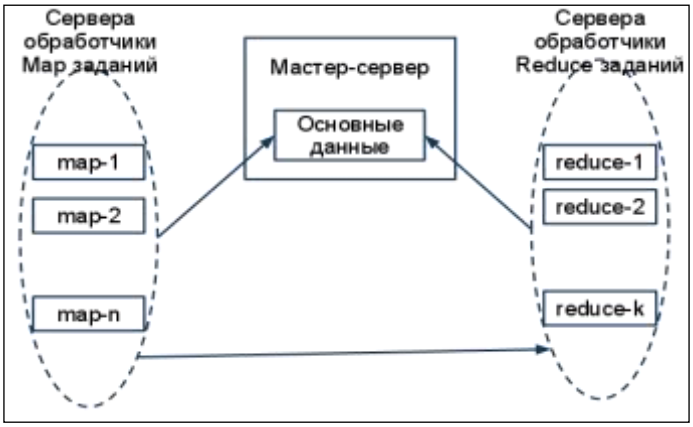

Figure 2 - Model of a system with one master server

MapReduce System Model on Cloud Services Infrastructure

Processing large amounts of data, typical for tasks using the MapReduce method , leads to the following problems in the PreReduce step :

1. List grouping problem: $\left\{\left\langle k_i^2, v_i^2\right\rangle\right\} \to \left\{\left\langle k_j^2, \left\{v_j^2\right\}\right\rangle\right\}$, where K in the resulting set are unique;

2. The problem of distributing tasks among $\left\langle k_i^2, \left\{v_j^2\right\}\right\rangle$ Reduce task processor servers .

The above problems are solved by using a system design model for the MapReduce method based on scalable and fault-tolerant cloud services (**Ошибка! Источник ссылки не найден.**).

The model uses a cloud table service and a queue service to store master data (Master Data), as well as a set (pool) of servers processing Map and Reduce tasks. In practice, the model can be implemented in Amazon clouds Web Services based on SimpleDB , Simple services Queue Service (SQS) and Elastic Cloud Computing . Or same V Azure clouds : Table Service, Queue Service, Azure Compute.

The model $M=\left\langle T, Q_{in}, T_u\right\rangle$ consists of cloud service infrastructure elements such as tables T and T_u and queues Q_{in}.

- The table T is intended for storing pairs of intermediate values $\left\langle k_i^2, v_i^2\right\rangle$ - the results of executing the Map procedure ;

- The queue Q_{in} is used to store keys that have not yet been processed by the k_i^2 Reduce procedure , and the keys in the queue must be unique, i.e. $\forall i, j: k_i^2 \neq k_j^2$.

ISSN: 2053-3578 I.F. 12.34

- Table T_u - also for storing unique keys k_i^2 and the status of their processing by the Reduce procedure : "not processed", "processed".

Table fields T: "key" k_i^2, "value" v_j^2. The keys k_i^2 are not unique, i.e. the condition $\forall i, j: k_i^2 \neq k_j^2$ is not met, since the table contains the output data of the Map procedure . Data from the table is retrieved by the "key" field, this field must be indexed, if we use the terms of relational databases.

Based on the model, M we formulate formal requirements for the PreReduce procedure at each step of the MapReduce method .

Split (before executing the Map procedure):

- table y T and T_u empty;

- the queue Q_{in} is empty;

Shuffle (after executing the Map procedure):

- the table T contains all the results of executing Map procedures in the form of pairs k_i^2, v_j^2;

- the queue Q_{in} contains all unique keys k_i^2 obtained as a result of executing the Map procedure ;

- The table T_u contains all unique keys k_i^2 with the status "not processed".

Sort (after executing the Reduce procedure):

- the table T does not change and contains all the results of executing Map procedures in the form of pairs k_i^2, v_j^2;

- the queue Q_{in} is empty;

- The table T_u contains all unique keys k_i^2 with the status "processed".

In the proposed model, all functions for storing the main data have moved from the master server to scalable and fault-tolerant cloud services. Cloud services, having scalability and fault-tolerance properties, are able to increase the fault-tolerance of the system as a whole. For this, the single master server must be replaced by a set of servers-processors of control tasks, in addition to the sets of servers of Map -tasks and Reduce -tasks.

Model Analysis

Figure shows a traditional system implementing the MapReduce method . Fault tolerance in the traditional implementation is provided only at the level of the processor servers

ISSN: 2053-3578 I.F. 12.34

[2,4]. Fault tolerance of the master server is not provided at the level of the system model.

Figure shows a system implementing the MapReduce method based on the proposed model.

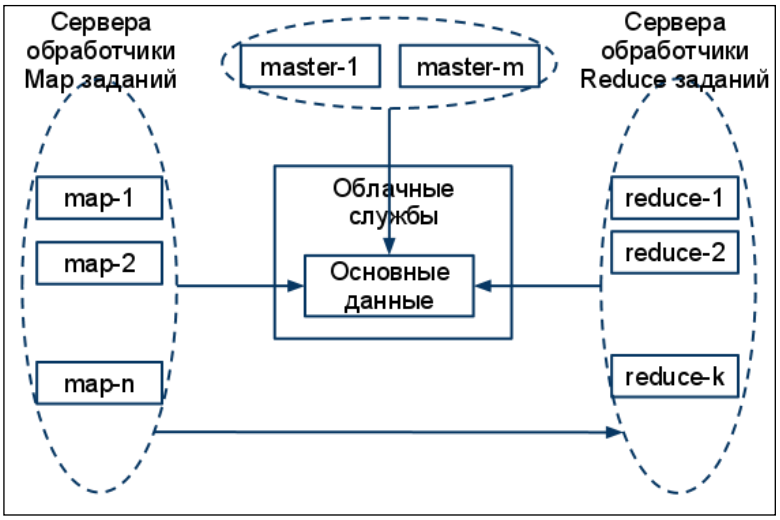

Figure 3 - proposed system model

In the proposed model, the master data is located on the cloud service side, not on the master server side, and the single master server is replaced by multiple control task processor servers. Thus, the proposed model does not contain elements that are presented in a single copy. Such elements are replaced by multiple processor servers, which becomes possible with the use of cloud services.

The paper shows that modern computing resource providers such as Amazon Web Services and Microsoft Azure , have a similar composition of cloud services providing computing resources. Services form the architecture of modern computing clouds. To solve computing problems in the "cloud", it is necessary to map the problem being solved to the infrastructure of cloud services.

This subsection presents a system model that allows the MapReduce method to be implemented on a modern cloud computing infrastructure. The model is easy to implement due to the use of cloud services, and it is shown that the model provides a higher level of fault tolerance and scalability than modern approaches implementing the MapReduce method .

References

1. Cano, L., Carello, G., Ardagna, D. A framework for joint resource allocation of MapReduce and web service applications in a shared cloud cluster (2018) Journal of Parallel and Distributed Computing, 120, pp. 127-147.

DOI: 10.1016/j.jpdc.2018.05.010

2. R. Lammel. Google's MapReduce programming model — Revisited // Science of Computer Programming. Volume 68, Issue 3, 2017. 208-237

3. Jens Dittrich, Jorge-Arnulfo Quian´e-Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty, and Jorg Schad. Hadoop++: Making a Yellow Elephant Run Like a Cheetah // Proceedings of the Very Large Database Endowment, Vol 3(1), 2010. Pp. 518–529.

4. R. Chen, H. Chen, B. Zang. Tiled-MapReduce: optimizing resource usages of data-parallel applications on multicore with tiling // Proceedings of the Parallel Computing Technologies, 2015, pp.523-534.