Real Time Big Data Analysis by Using Apache Kudu and NoSQL Redis in Web Applications

Assoc. Prof. Dr. Pavel Petrov
University of Economics - Varna, Varna, Bulgaria
petrov@ue-varna.bg

Prof. Dr. Georgi Dimitrov
University of Library Studies and Information Technologies, Sofia, Bulgaria
g.dimitrov@unibit.bg

Assoc. Prof. Dr. Oleksii Bychkov
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine
bos.knu@gmail.com

Abstract

In the recent years data processing in Big Data applications are moving from a batch processing model to a live streaming processing model. The live streaming processing model allows to process data in a real time or near real time manner. In many cases, there are need of combining data from various sources - both new generated data and data previously gathered and archived. In each case a different storage mechanism is appropriate to achieve best flexibility and to avoid the so-called bottleneck when processing data. In this article we review some of the possibilities of using Apache Kudu and NoSQL Redis systems which made them suitable to be used together for fast processing of streaming data.

Keywords: Big Data, Hadoop, Kudu, Redis, NoSQL, Apache, PHP, web applications.

JEL Code: C88, O32 DOI: https://doi.org/10.36997/IJUSV-ESS/2020.9.1.26

Introduction

Apache Hadoop's Hadoop Distributed File System (HDFS) is suitable for efficient storage of large data. The main feature of storing data in HDFS is that once saved, the data cannot be changed, and only read and delete operations are possible. Also, the architectural features of HDFS do not allow effective real-time big data analysis. For real-time analysis (Bulut&Erol, 2018; Petrivskyi et al., 2020), it is appropriate to use the Apache HBase system, which uses HDFS, but the time to crawl the recorded data volumes is large. Apache Kudu can be used to overcome these problems (Fig. 1).

Kudu was established in 2015 by Cloudera as an internal project to create a new columnar database management system of the NoSQL type and supporting SQL for operational analysis of rapidly changing big data. In 2016, the source code was distributed under an open license by the Apache Software Foundation.

The main purpose of Apache Kudu is to work with fast-changing data. There are options for both quick add operations and for updating and deleting rows in real time. Relatively fast crawls of columns for real-time analysis of one layer for data storage are performed (Fig. 2).

1. Creating a data model with the analytical tool Apache Kudu

Kudu could be used as so called "cluster coordinator", who keeps track of "alive" servers in a cluster and redistributes data after server failures (Lipcon et al., 2015; Shook, 2013). It is also possible to be combined Redis with other Big Data technologies, like ELK and not only Hadoop (Clouder, 2018).

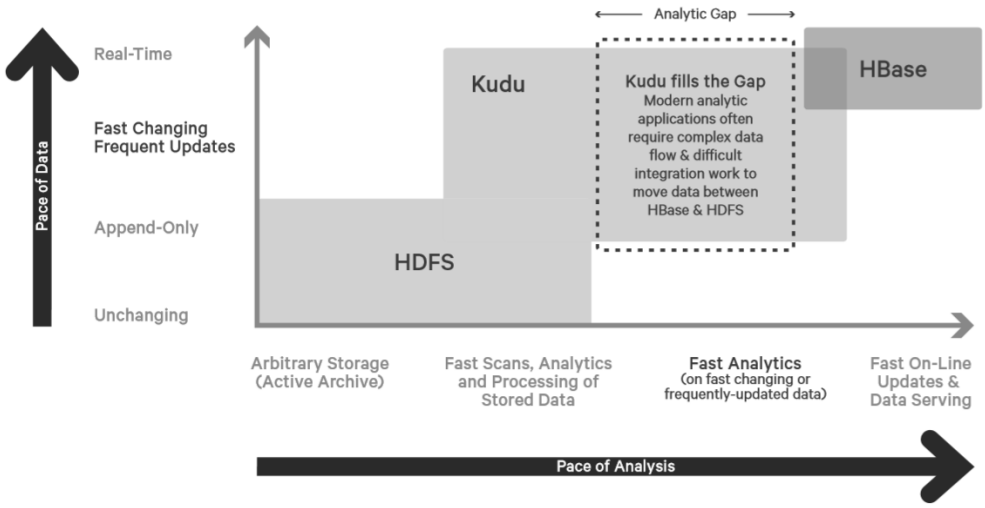

Figure 1. Apache Kudu's place in relation to HDFS and HBase in big data analysis via Hadoop.

Source: Apache Kudu Datasheet (2017). https://www.cloudera.com/content/dam/www/ marketing/resources/ datasheets/cloudera-kudu-datasheet.pdf.landing.html.

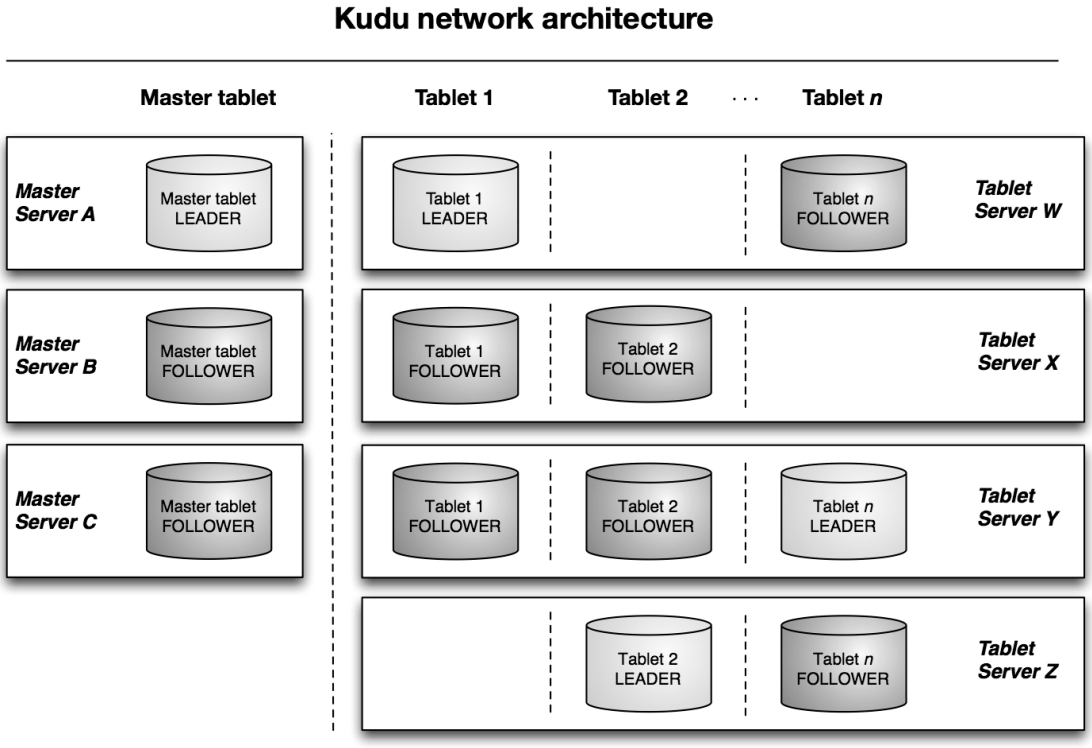

Figure 2. Architecture of the Apache Kudu.

Source: Introducing Apache Kudu. Architectural Overview. https://kudu.apache.org/docs/

Another interesting use case scenario is selection of datasets for real time dashboards (Michele et al., 2019; Petrova et al., 2019; Vasilescu et al., 2019).

Big data is composed of three main dimensions: Volume, Variety, and Velocity. The dimension Velocity refers to the speed of data processing. There are cases where real time or near

real time speed of processing are required. With the wide spread of big data technology this high-speed data processing become more popular, but there are limitations on bandwidth and latency for real time or near real time processing. That is why there are some gaps between an analytics system and a real end-to-end system (Wang et al., 2017; Marcu et al., 2017; Petrov&Valov, 2019).

The data model when working with the analytical tool Apache Kudu is realized through tables. Tables in Kudu are like tables in traditional DBMSs, and their design is important for high performance. It is necessary to approach each case individually, as there is no common design approach that is suitable (Dimitrov et al, 2016; Malkawi et al., 2020) for each table. Important points in designing the structure of the tables are setting the appropriate data type of the columns, selecting the primary key and allocating the columns in different tables.

Tables in Kudu are divided into parts called tablets, which are distributed among many tablet servers. An entire row of a table is always located on one tablet. The way the rows are distributed on the tablets is set when creating the table.

To choose a suitable partitioning strategy, it is necessary to know the data model and the expected operations that will be performed with the table - mostly read or mostly write. In treatments which mainly involve recording, it is appropriate to distribute the recordings between the tablets so as not to load a single tablet. For a predominant reading, it is appropriate that all data be located on the same tablet.

The two types of data splitting by tablets are: splitting ranges and splitting by hash value.

When dividing by range, the rows are distributed on the tablets based on a special common key for multiple rows (Fig. 3). Ranges should not overlap and can be added and removed dynamically.

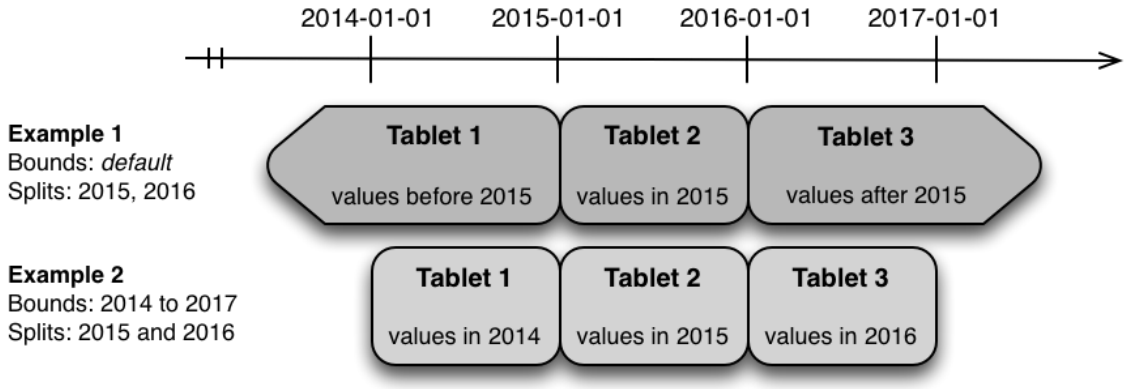

Figure 3. Examples of dividing rows by ranges in different tablets.

Source: Apache Kudu Schema Design (2019). https://kudu.apache.org/docs/schema_design.html

When dividing by hash value, the rows are divided into tablets based on the hash value by which the rows are placed in one of several groups (Fig. 4). This splitting strategy is appropriate when the order of the records is irrelevant. In this case, the recording operations are randomly distributed between the tablets, which reduces the likelihood of obtaining "bottlenecks" in data processing.

It is also possible to apply a multi-level separation strategy that combines the benefits of the two types of separation, reducing their disadvantages (Fig. 5).

The main scenarios for using Apache Kudu in big data systems are the following:

- Processing of a constant input data stream in real time. These are situations where new data arrives frequently and constantly. They must be available in near real time for reading, crawling, and changing. Apache Kudu could quickly insert and update with efficient column crawling.

- Study of time series. In time series, data groups are organized according to the time in which they occurred. It is suitable for studying the effectiveness of various indicators over time or for predicting future behaviour based on past data. Apache Kudu is suitable for working with time series.

- Creating forecast models. In machine learning using large data sets, the model may need to be updated and changed frequently. This can happen, for example, while training is taking place because of a change in the simulated situation. A researcher may want to change one or more factors in the model (Panayotova et al., 2016; Pashev et al., 2019) to see what happens over time. Updating a large set of data stored in files in HDFS requires a lot of resources, as each file must be completely overwritten. In Apache Kudu, data updates are performed in near real time, and the researcher can adjust the parameter, restart the query, and refresh, for example, a chart in seconds or minutes, rather than hours or days. Batch or incremental algorithms can also be executed on data in near real time.

- Combining data in Apache Kudu from existing ("legacy") systems. Businesses generate data from multiple sources and store it in a variety of systems and formats (Balabanova et al., 2016; Georgiev et al., 2018; Iliev et al., 2010; Kostadinova et al., 2018). It is possible by using other subsystems in Hadoop, such as Impala (Quinto, 2018), to use data from different sources and in different formats without the need to modify existing systems.

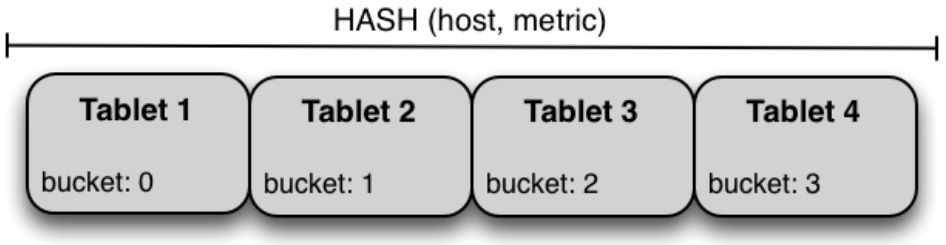

Figure 4. Example of dividing rows by hash values into different tablets.

Source: Apache Kudu Schema Design (2019). https://kudu.apache.org/docs/schema_design.html

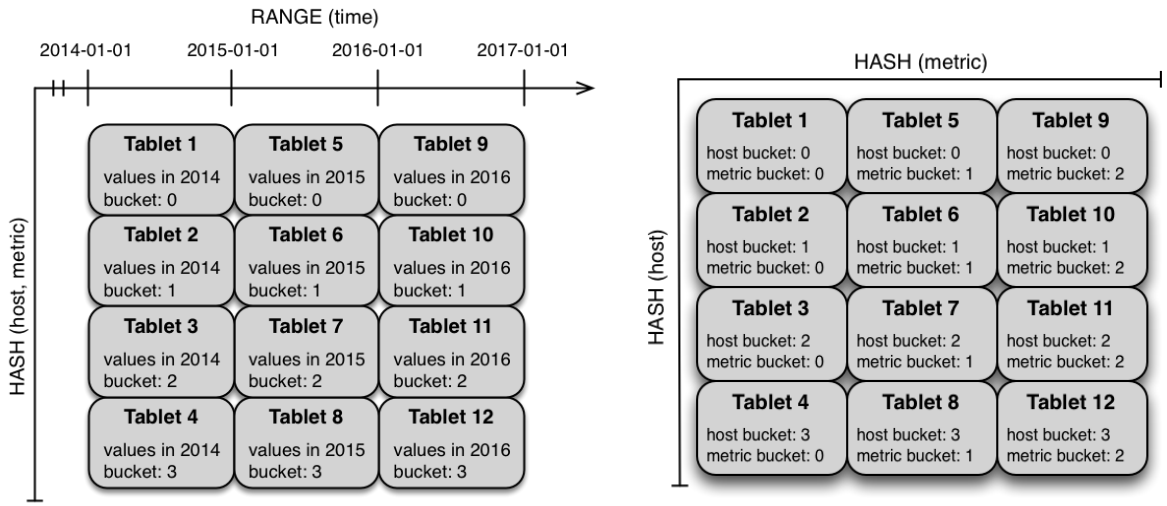

Figure 5. Example of dividing into several levels - by ranges and by hash values and example of dividing into several levels - by hash values and again by hash values.

Source: Apache Kudu Schema Design (2019). https://kudu.apache.org/docs/schema_design.html

2. Basic administrative operations with Redis

NoSQL systems are suitable for building heavily loaded web applications. They provide very high speeds for searching, retrieving, and writing data, sometimes many times higher than that offered by relational database systems. In recent years, they have also been used to store and process large amounts of data ("Big Data") in the range of tens of petabytes to several exabytes.

Data in most NoSQL systems is stored in the so-called. hash tables and the basic operations that can be performed are relatively simple: SET - setting a key value; GET - return value by key; DELETE - delete a key value. On some systems (usually storing data only in RAM), you can also set the lifetime of the key, after which the key and its corresponding value are automatically deleted.

Among the most popular NoSQL systems is Redis (REmote DIctionary Server, http://redis.io/), which is a free software server system that stores data in RAM. According to a study by the Austrian consulting firm solidIT, which maintains the site http://db-engines.com, in the period 2012-2020, the most popular open source key-value NoSQL system is Redis.

Not only single (scalar) values (numbers and strings) can be stored as values, but also different types of data, such as hashes, lists, sets, sorted sets, array of bits, etc.

To work with Redis, it is necessary to download and install a suitable installation package for the respective operating system. For the operating system, it is recommended to use the UNIX / GNU / Linux version, but there are also various precompiled versions for Windows that can be used in application development. Under Windows, it is not necessary to perform a classic installation, as there are variants of the system such as "portable apps" and the procedure is simpler: for example, a 64-bit version can be downloaded from the Microsoft Archive page at github.com (https://github.com/microsoftarchive/redis). After downloading, for example, the file Redis-x64-3.2.100.zip (about 5 MB), it can be unzipped in a custom directory and thus the installation process ends. After unpacking, the main files in the directory are the following:

- redis.windows.conf - an example text configuration file for setting server settings, for example, which IP address and port number to "listen" (accept requests) from the server; the maximum amount of memory to occupy the application; log file settings; work directories; replication, etc .;
- redis-server.exe - the application itself - Redis server - takes up about 1 MB;
- redis-cli.exe - client program for connection to the server;
- redis-benchmark.exe - program for testing the speed of operations.

After starting the server (redis-server.exe), the home screen displays service information. The server can be accessed through the client program (redis-cli.exe) and commands can be entered (see Fig. 6).

We will briefly pay attention at the most used commands, which can initially be submitted for execution through the client program (redis-cli.exe), and later will be used in PHP programs (Petrov, 2008). To begin with, the connection to the server can be checked using the PING command, where the server must respond with PONG, if everything is normal:

127.0.0.1:6379> PING

PONG

Other useful administrative commands are:

INFO - displays information about the settings and statistics for the server.

SELECT DB_NUMBER - sets any active database. The default is the one with number 0. There are usually 16 databases initially.

DBSIZE - returns the number of keys in the current database.

CONFIG GET DATABASES - returns the number of databases.

FLUSHALL - deletes all keys from all databases.

FLUSHDB - deletes all keys from the active database.

MONITOR - displays in real time all requests to the server. It is good to use a client in a

separate window.

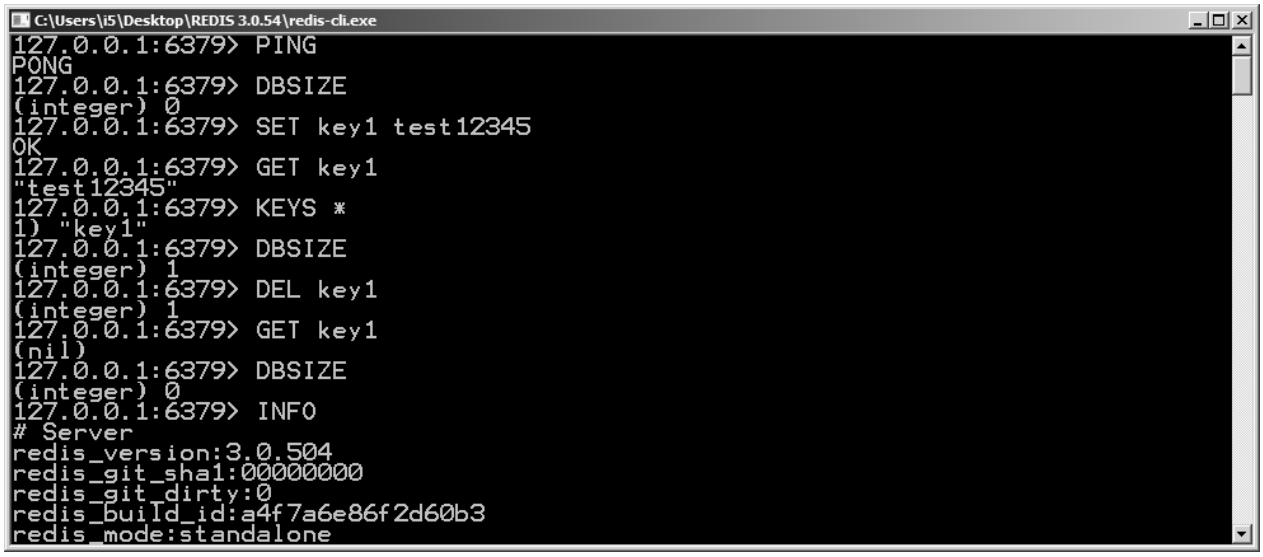

Figure 6. Work screen of redis-cli.exe.

The data can be set to a lifetime in seconds with the EX (EXpire) parameter. The following example sets the value of the x key, which will be deleted after 1 second. With a longer lifetime, the TTL (Time To Live) command can be used to check how much time is left before the key is deleted. The SETNX (SET if Not eXists) command sets a new key value, but only if it did not previously exist. As we have already said, in addition to single values, data structures can also be stored in Redis. When working with these data types, both commands like those given above and some new commands specific to the data type are used. In front of the command name there is a letter that indicates what type of data it refers to: H - for hash (Hash), L - for list (List), S - for set (Set), Z - for sorted set (sorted set) and etc.

What is specific about the hash (associative array) as a value in Redis is that both the key and the value are strings. To distinguish the keys that are in the value from the master key to which they are associated, they are called fields. Many commands are used to work with single pairs of value fields: HSET, HSETNX, HGET, HDEL, HEXISTS, HINCRBY, HINCRBYFLOAT, etc.

Lists, as a data structure in Redis, are an ordered set of elements (generally strings), each of which has a serial number in the list - an index. List operations are the same as dynamic array operations (in some programming languages, the Vector and ArrayList classes). There are different options for inserting items into a list: LSET, LINSERT, RPUSH / LPUSH, RPUSHX / LPUSHX, LLEN, LINDEX, LRANGE, RPOP / LPOP, LREM, LTRIM and others.

Sets are like lists, except that they contain unique elements. In addition, sets do not have an index and order. They implement the concept of set in mathematics in a programmatic way. The main operations are: SADD, SREM, SPOP, SRANDMEMBER, SISMEMBER, SMEMBERS, SCARD, SMOVE, SUNION $A \cup B = \{x \mid x \in A \vee x \in B\}$, SUNIONSTORE, SINTER $A \cap B = \{x \mid x \in A \wedge x \in B\}$, SINTERSTORE, SDIFF $A - B = A \cap \overline{B} = \{x \mid x \in A \wedge x \notin B\}$, SDIFFSTORE and others.

Sorted sets are like sets in that they consist of unique elements. They are similar to lists, as the elements are arranged - each element is assigned a real number (score), which is used to arrange the elements. This number is often called weight or priority depending on the context. The main and specific operations with sorted sets related to the handling of elements and their weights are: ZADD, ZREM, ZCARD, ZSCORE, ZRANK / ZREVRANK, ZINCRBY, ZCOUNT, ZLEXCOUNT, ZRANGE / ZREVRANGE, ZRANGEBYLEX / ZREXRANYRANBY ZREMRANGEBYLEX, ZREMRANGEBYSCORE, ZUNIONSTORE, ZINTERSTORE, etc.

3. Using Redis in PHP programs

To work in PHP with Redis, in addition to a working server, it is necessary to install the so-called PHP client. There are different client options (for a full list of Redis clients see: Clients, https://redis.io/clients), of which two options are recommended: PhpRedis (A PHP extension for Redis, https://github.com/phpredis/phpredis, http://windows.php.net/downloads/pecl/releases/redis/) and Predis (Windows and feature-complete Redis client for PHP and HHVM, https://github.com/nrk/predis). Using PhpRedis involves installing additional libraries and making several changes to configuration files. For this reason, it is easier to use the Predis client, which is fully implemented in PHP and only needs to copy the directory in which its files are located.

The installation procedure for the Predis client is as follows:

1. Download the latest version of Predis from https://github.com/nrk/predis/releases (for example predis-1.1.1.zip).

2. From the archive file (for example predis-1.1.1.zip) the "src" directory is copied to the directory where the PHP program that will work with Redis is.

3. The directory "src" is renamed with a more meaningful name - "Predis".

4. In the PHP program with the operator "require" the file 'Predis/Autoloader.php' is loaded and the static method register () is called from the Autoloader class. Objects of the Predis \ Client class can then be created, and methods for working with the Redis server can be called.

The following program shows the basics of using Redis:

```
<? php
require 'Predis/Autoloader.php'; // load file
Predis/Autoloader::register(); // the static method is executed
$r = new Predis\Client (); // create a new object
$r->set('key1', 'test123'); // key1 is set to test123
$v = $r->get ('key1'); // returns the value of the key key1
print $v; // the value of $v is output to standard output
?>
```

When the program is distributed to other computers, it is necessary to include the subdirectory where the Redis client is.

Below is the program redis.php, which is a complex example of working with Redis. Method names can be written in lowercase or uppercase and are the same as Redis command names. In the example, we intentionally use capital letters to make it easier to see the methods associated with Redis commands.

```
<?php
require 'Predis/Autoloader.php';
Predis\Autoloader::register();
$r = new Predis\Client();
try {
        $r->connect();
} catch (Exception $e) {
        print 'No connection to Redis server!';
        exit;
}
print '<pre>';
print_r($r->PING()); //PONG
print_r($r->SELECT(3)); //OK
print_r($r->SET('key1', 'value1')); //OK
print_r($r->GET('key1')); //value1
print_r($r->EXISTS('key1')); //1
print_r($r->DEL('key1')); //1
```

```
print_r($r->GET('key1')); //null
print_r($r->FLUSHDB()); //OK
print_r($r->DBSIZE()); //0
print_r($r->QUIT());
?>
```

Conclusion

The combined use of Apache Kudo and Redis opens interesting possibilities in terms of optimizing real-time operations. Both systems have advantages that make them suitable for use in different situations. It is feasible the input data received in real time to be stored and processed in Kudu. A middleware application could fetch the results from Kudu and store the data in REDIS system. From the programmer's point of view, it will be easier that the end user web application (written for example on PHP - Petrov et al., 2019; Petrov et al., 2020) to work directly with REDIS instead with Kudu. Thus, the rich analytical possibilities that Hadoop provides can be relatively easily used by a web application with which end users work directly.

References

1. Apache Kudu Datasheet (2017). [Online] Available from: https://www.cloudera.com/content/dam/www/marketing/resources/datasheets/cloudera-kudu-datasheet.pdf.landing.html [Accessed 10/10/2020]
2. Apache Kudu Schema Design (2019). [Online] Available from: https://kudu.apache.org/docs/schema_design.html [Accessed 10/10/2020].
3. Balabanova, I., Georgiev, G., et al. (2016). Classification of Teletraffic Service Devices by K-NN, ANFIS and ANN Classificators. *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp.1-5. doi:10.1109/BlackSeaCom.2016.7901585
4. Bulut F., & Erol M.H. (2018). A Real-Time Dynamic Route Control Approach on Google Maps using Integer Programming Methods. *International Journal of Next-Generation Computing*, 9(3), pp.189-202.
5. Clouder, A. (2018). Combining Redis with Hadoop and ELK for Big Data. [Online] Available from: https://www.alibabacloud.com/blog/combining-redis-with-hadoop-and-elk-for-big-data_582482 [Accessed 10/10/2020]
6. Dimitrov, G., Panayotova, G., Garvanov, I., et al. (2016). Performance analysis of the method for social search of information in university information systems. In *3rd International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, Lodz, Poland, IEEE, pp.149-153. doi:10.1109/ICAIPR.2016.7585228
7. Georgiev G., Balabanova I., et al. (2018). Identification of Sine, Squire, Triangle and Saw-tooth Waveforms with Uniform White and Inverse F Noises by Adaptive Neuro - Fuzzy Interface System. *Journal of Engineering Science and Technology Review*. 11(3), pp.128–132. doi:10.25103/jestr.113.17
8. Iliev, P., Salov, V., & Petrov, P. (2010). *Virtualni sistemi*. Varna: Nauka i ikonomika,
9. Introducing Apache Kudu. Architectural Overview. [Online] Available from: https://kudu.apache.org/docs/ [Accessed 10/10/2020]
10. Kostadinova, I., Toshev, R., et al. (2018). Temporal Analysis of the Pedagogical Adoptions use and Application of the Augmented and Virtual Reality Technologies in Technical Subject Areas. In *11th Annual International Conference of Education, Research and Innovation*, ICERI2018 Proceedings, Seville, Spain: IATED, pp.4387-4393.
11. Lipcon, T., Alves, D., Burkert, D., et al. (2015). Kudu: Storage for fast analytics on fast data. Cloudera, inc, 28. [Online] Available from: https://kudu.apache.org/kudu.pdf [Accessed 10/10/2020]

12. Malkawi, R., Saifan, A. A., et al. (2020) Data Mining Tools Evaluation Based on their Quality Attributes. *International Journal of Advanced Science and Technology*, 29(3). pp.13867-13890.
13. Marcu, O. C., Costan, A., Antoniu, G., et al. (2017). Towards a unified storage and ingestion architecture for stream processing. In *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, pp.2402-2407.
14. Michele, P., Fallucchi, F., & De Luca, E. W. (2019). Create Dashboards and Data Story with the Data & Analytics Frameworks. In *Research Conference on Metadata and Semantics Research*, Springer, pp.272-283.
15. Panayotova, G., Dimitrov, G., et al. (2016). Modeling and data processing of information systems. In *3rd International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, Lodz, Poland, IEEE, pp.154-158. doi:10.1109/ICAIPR.2016.7585229
16. Pashev, G., Rusenova, L., Totkov, G., & Gaftandzhieva, S. (2019). Business Process Modelling & Execution Application in Work Education Domain. *TEM Journal, 8*(3), pp.992-997. doi:10.18421/TEM83-42
17. Petrivskyi, V., Dimitrov, G., Shevchenko, V. et al. (2020). Information Technology for Big Data Sensor Networks Stability Estimation. *Information and Security, 47*(1), pp.141-154. doi:10.11610/isij.4710.
18. Petrov, P. (2008). *Sarvarno programirane*. Varna: Nauka i ikonomika.
19. Petrov, P., & Valov, N. (2019). Digitalization of Banking Services and Methodology for Building and Functioning of Fintech Companies. *Izvestia Journal of the Union of Scientists - Varna. Economic Sciences Series, 8*(1), pp.110-117. doi:10.36997/IJUSV-ESS/2019.8.1.110
20. Petrov, P., Buevich, A., Dimitrov, G., et al. (2019). A Comparative Study on Web Security Technologies Used in Bulgarian and Serbian Banks. In *19 International Multidisciplinary Scientific Geoconference SGEM 2019: Conference Proceedings*, 19(2.1), pp.3-10.
21. Petrov, P., Dimitrov, P., et al. (2020). Using the Universal Two Factor Authentication Method in Web Applications by Software Emulated Device. In *20 International Multidisciplinary Scientific Geoconference SGEM 2020: Conference Proceedings*, 20(2.1), pp.403-410. doi:10.5593/sgem2020/2.1/s07.052
22. Petrova, S., Stefanov, S., Ivanov, S., Sergeev, A., & Getova, I., (2019). Information systems used in Bulgarian university libraries as online public access catalogs. *International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM*, 19(2.1), pp.353-360. doi:10.5593/sgem2019/2.1/S07.046
23. Quinto, B. (2018). High Performance Data Analysis with Impala and Kudu. In *Next-Generation Big Data*. Apress, Berkeley, CA. pp.101-111
24. Shook, A. (2013). Making Hadoop MapReduce Work with a Redis Cluster. [Online] Available from: https://tanzu.vmware.com/content/blog/making-hadoop-mapreduce-work-with-a-redis-cluster [Accessed 10/10/2020]
25. Vasilescu, C., Suciu, G., & Pasat, A. (2019). A New Method to Help the Human Resources Staff to Find the Right Candidates, Based on Deep Learning. In *The International Scientific Conference eLearning and Software for Education*, "Carol I" National Defence University, v.3, pp.240-246. doi:10.12753/2066-026X-19-170
26. Wang, K., Bian, B., Cao, P., & Riess, M. (2017). Experiences and Lessons in Practice Using TPCx-BB Benchmarks. In *Technology Conference on Performance Evaluation and Benchmarking*, Springer, pp.93-102.