Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

A software architecture for monitoring big data storage platforms

Büyük veri saklama ortamlarının izlenmesine yönelik yazılım mimarisi

Mehmet Sıddık AKTAŞ

[1]Department of Computer Engineering, Faculty of Electrical and Electronics Engineering, Yildiz Technical Univ., Istanbul, Turkey.
aktas@yildiz.edu.tr

Received/Geliş Tarihi: 12.03.2016, Accepted/Kabul Tarihi: 07.03.2017
* Corresponding author/Yazışılan Yazar

doi: 10.5505/pajes.2017.45722
Research Article/Araştırma Makalesi

Abstract

NoSQL-based big data storage platforms provide similar fundamental big data management functionalities in addition to various other functionalities that differ for each platform. For example, document-oriented NoSQL big data storage platforms are commonly used for organizing and managing documents, while graph-based databases are designed for data whose relations are represented by graphs. In turn, different NoSQL-based platforms are often used together, as each provides distinct capabilities. Within this study based on our literature review, it is seen that a hybrid platform, which could perform real-time monitoring tasks on top of different big data NoSQL platforms, is lacking. In order to address this issue, this paper proposes a novel system architecture. The proposed system architecture runs as a piece of add-on software one layer above the NoSQL platforms and provides monitoring tasks on these platforms. Within the research, a prototype of the recommended system architecture is made and the testing results are provided in detail. The prototype of the proposed architecture focused on monitoring the system's distributed structure and data structure, memory usage, and disk usage. In order to prove the practical usage of the proposed system architecture, various performance experiments were applied to the prototype application. In this paper, we report on the promising results of the performance experiment.

Keywords: NoSQL, Big data, Hybrid platform, Monitoring platform, Hybrid architectural structure, Management of big data

Öz

Büyük Veri NoSQL saklama platformları, aynı temel fonksiyonel özellikleri sağlamakla birlikte, birbirlerinden ayrılan geniş özellik setleri de sunabilmektedir. Örneğin, doküman-tabanlı NoSQL saklama ortamları dokümanların organizasyonu için kullanılırken, çizge veri tabanları, veriler arasındaki ilişkilerin çizgeler olarak tanımlanabildiği veriler için kullanabilmektedir. Bunun sonucu olarak, farklı platformlar, sağladıkları ayrıştırıcı işlevselliklerinden dolayı birlikte kullanılabilmektedir. Ancak, farklı büyük veri saklama platformları üzerinde gerçek zamanlı olarak izleme işlemlerini yapabilecek, tek bir programlama ara yüzü üzerinden kullanılabilen, bir platformun eksikliğinin olduğu görülmektedir. Bu eksikliği gidermek amacıyla, bir sistem mimarisi önerilmektedir. Önerilen sistem birbirinden farklı NoSQL platformları üzerinde üzerinde eklenti olarak çalışmakta ve izleme işlemlerinin yapılabilmesine olanak sağlamaktadır. Çalışma kapsamında önerilen sistem mimarisinin prototipi gerçekleştirilmiştir ve elde edilen deneyim detaylarıyla sunulmaktadır. Prototip uygulama, sistemin dağıtık mimari yapısını ve veri yapısını, bellek kullanımını ve sabit disk kullanımını takip etmek odaklı olarak geliştirilmiştir. Önerdiğimiz sistem mimarisinin doğruluğu ve pratik kullanımını ortaya koymak amacıyla prototip uygulama üzerinde performans testleri uygulanmış ve olumlu olarak elde edilen sonuçlar paylaşılmıştır.

Anahtar kelimeler: NoSQL, Büyük veri, Hibrit platform, İzleme platformu, Hibrit mimari yapısı, Büyük veri yönetimi

1 Introduction

In the early 2000s, NoSQL platforms were introduced to solve performance issues arising from the use of classical relational databases as a means of dealing with the increasing size of data. These platforms, where the data are stored in various forms such as text documents or graphs, offer data analysis capabilities with high performance on data sizes exceeding petabytes.

Today, we observe that different NoSQL platforms appear to be in line with the growth of businesses that deal with big data. NoSQL platforms are able to store data in various formats, such as key-value tables, document-oriented formats, and graph-oriented formats. Key-value tables based NoSQL platforms store data in a schema-less model, with simple key-value pair to store, access, and manage data. Document-oriented NoSQL platforms store the data in a document format (e.g., JSON document format). On these platforms, the data are held in structures called "collections." in contrast to the table structures of conventional relational databases. Contrary to the rule of generating the tables of relational databases according to a specific data schema, no specific data model is sought out for JSON documents within the "collection." Each document is independent and may differ from others in terms of its data structure. Another type of NoSQL platform is one where the data are stored according to a graph data structure.

Unlike other NoSQL platforms, graph-oriented platforms store the relationships among data. These platforms are designed according to graph theory. NoSQL platforms provide similar fundamental functionalities in addition to various other functionalities that differ on each platform. Consequently, different NoSQL structures may be used together because of the distinct functionalities they provide. For example, an e-trading site would use a document-oriented NoSQL platform to store JSON objects created as a result of clickstreams, while collecting data from itsonly users on social media and storing them on a NoSQL platform with a graph data structure. The distributed structure of NoSQL platforms requires real-time monitoring. The monitoring these platforms is dependent on monitoring tools specifically developed for these environments. NoSQL platforms may have some common monitoring functionalities as well as specialized monitoring functionalities. This shows us that a common programmable interface would allow one to monitor multiple NoSQL platforms. In our literature review, we identified an important deficiency in monitoring these platforms-the lack of a common monitoring tool in the case of different NoSQL platforms being used together. Each of the aforementioned NoSQL platforms has its own monitoring tools. In turn, monitoring a NoSQL platform is only possible with its respective monitoring tool. Consequently, no monitoring tool of a certain platform may be used to monitor another NoSQL platform.

To eliminate this deficiency, a monitoring platform architecture will be proposed in this paper. This architecture, which operates one level higher than NoSQL platforms as a piece of add-on software, is capable monitoring in real time common properties of NoSQL platforms in a hybrid structure. The proposed architecture was designed based on client/server architecture. A prototype was developed to show the usability of the proposed architecture. The prototype focused on monitoring the system's distributed structure and data structure, memory usage, and disk usage. The developed prototype platform is an application that enables real-time monitoring of different NoSQL databases. It enables monitoring in various reporting formats (e.g., tabular presentation and visual presentation) by retrieving data from NoSQL databases on the system. In order to prove the accuracy and practical use of the proposed system architecture, performance tests were executed on the prototype application. The results showed that such an architecture creates a negligible processing overhead. The rest of the article follows with a literature review and the definition of the research problem. Then, the proposed system architecture and its prototype will be discussed in detail. Next, the performance assessment of the prototype, followed by the presentation of results and future studies, is introduced.

2 Literature review

The term "NoSQL" stands for "Not Only SQL."' An open source, light, relational database without a standard SQL interface, developed by Carlo Strozzi, was first used in 1998 [1]. In the early 2010s, when the difficulties of working with conventional databases (due to the increased size of data) became obvious, NoSQL databases gained great popularity. NoSQL database systems are distinguished by their higher scalability and unstructured data schema in comparison to relational database systems [2],[3]. While relational databases meet all the requirements of ACID (Atomicity, Consistency, Isolation, Durability), NoSQL databases do not meet them completely. Instead, they prioritize the requirements of sustainability (in case problems occur in any cluster element of the distributed system) and the requirements of preserving data integrity. NoSQL platforms provide BASE (Basically Available, Soft-state, and Eventually Consistent) principles in place of ACID principles [4]. As mentioned above, unlike tables in classical relational databases, NoSQL platforms do not hold data based on a specific data model. On the contrary—data are stored in key-value pairs in data structures such as documents or graphs. In this article, the most popular document-oriented (MongoDB [5], CouchBase [6]) and graph-oriented NoSQL (Neo4J [7]) platforms were examined.

Today's NoSQL platforms have programming interfaces that enable self-monitoring. While these platforms involve platform-specific, stand-alone, real-time monitoring tools (e.g., MongoVUE), tools are designed to work only with their corresponding NoSQL platform. There are various monitoring tools for key-value based, document-based, and graph-oriented NoSQL platforms. The key-value oriented NoSQL platform Cassandra [8] uses a monitoring tool called DataStax OpsCenter [9]. OpsCenter reports on metrics such as the active and passive number of nodes, the number of read/write operations in clusters, the cluster delay, and the disk usage. The document-oriented platform MongoDB includes various monitoring tools. MongoVUE [10] is one example of these tools. A variety of statistical data about NoSQL storage nodes (shards), databases, document sets (collections), hard disk nodes, and memory usage are accessible on MongoVUE. The graph-oriented NoSQL platform Neo4J uses the JConsole monitoring tool [11]. This monitoring platform also presents data about the number of nodes, the number of data per node, and the disk usage.

In our study, we observed that different NoSQL technologies have similar monitoring functionalities. However, there exists no hybrid monitoring platform that would allow one to monitor different NoSQL platforms over a single programming interface. This missing aspect of NoSQL monitoring tools was the triggering factor for us to initiate this research.

3 Definition of the problem

Today, it is obvious that almost all NoSQL platforms have specific programming interfaces to monitor their capabilities, and there exist dedicated monitoring tools that can only interact with these programming interfaces. Currently, monitoring various NoSQL platforms via one monitoring tool is not possible. This drives the need for a hybrid monitoring platform for cases where different NoSQL databases are being used together and need to be monitored. To develop a tool capable of monitoring different NoSQL platforms, the common functionalities of these platforms should be utilized. How the software architecture of a hybrid monitoring tool, working one level above the available NoSQL platforms, should work, along with how to verify it to meet such a requirement, comprises the research problem of this paper.

4 Proposed software architecture

The proposed Hybrid NoSQL Monitoring Platform (HNMP) is depicted in Figure 1. As depicted in this figure, the hybrid monitoring platform runs one level above the existing NoSQL platforms. The HNMP has no dependency on technologies at the lower level. Information such as which NoSQL databases are involved and in which directory the relevant libraries (e.g., .jar files) and relevant configuration files are located in the file system is stored in a configuration file (Hybrid-Mapping.xml).

In the proposed software architecture, Facade Design Pattern In the proposed software architecture, the facade design pattern is utilized. The facade pattern provides a unified interface to a set of interfaces in a subsystem. Here, the subsystem is a cluster of various NoSQL systems. It defines a higher-level interface that makes the subsystem easier to use and wraps a complicated subsystem up in a simpler interface [12].

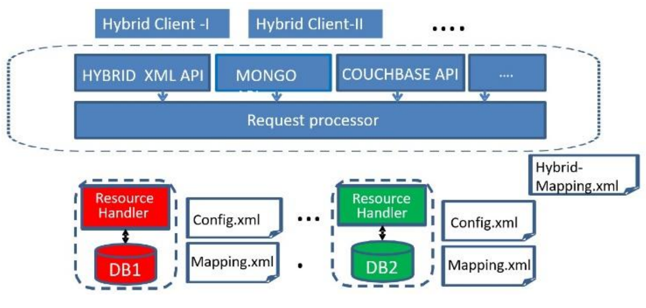

Figure 1: hybrid NoSQL monitoring software arcitecture.

In the proposed architecture, a resource handler module is used for each subsystem. The hybrid platform communicates with each resource handler module using two different configuration files: Mapping.xml and Config.xml. The first file maps the monitoring functions provided by the hybrid NoSQL monitoring tool, with monitoring functions enabled by the NoSQL technology at the lower level. The second file includes the IP addresses of the underlying NoSQL instances, which are accessible via distributed servers within the NoSQL database clusters.

The resource handler module includes the required libraries to access the database of the platform to be monitored. During the development of our proposed system's prototype, this module was realized as a Java library (.jar file format).

To identify the core monitoring functions of the HNMP, we identified the common monitoring functionalities of existing NoSQL databases. These monitoring functions are listed in Table 1. The XML API of the proposed system provides monitoring capabilities for these functions over different NoSQL platforms.

The Hybrid NoSQL Monitoring Platform uses the located resource handler library of the NoSQL databases on the next highest level and performs monitoring functions by accessing the relevant database. The facade design pattern provides a common interface for the lower-level systems' programming interfaces. Thus, the deployment of lower-level systems over a higher-level interface is enabled independently of the technologies deployed by the lower-level systems. The complexity is reduced by wrapping the complicated subsystem at the lower level. In the proposed architecture, the end-user interacts with the HNMP, which, in turn, communicates with the rest of the lower-level subsystems across the facade object.

5 Hybrid NoSQL monitoring functions

As a design requirement for the system to monitor various NoSQL platforms, an examination of these platforms was required to identify common monitoring functionalities. In this study, common methods to monitor NoSQL platforms were classified under three main categories: methods to monitor general information related to the system's distributed structure and data structure; methods to monitor memory usage; and methods to monitor disk usage.

As mentioned earlier, the HNMP uses the Mapping.xml file to map these hybrid monitoring functions to their corresponding implementations in the resource handler modules specific to each NoSQL platform. Table 1 shows examples of the hybrid monitoring methods and their corresponding monitoring functions in the corresponding NoSQL systems. Detailed information on these monitoring functions is provided below.

1. Methods to monitor general information related to the system's distributed structure and data structure are the methods commonly developed for NoSQL platforms to inquire about the nodes involved in these platforms, the number of active/passive nodes, and the data structures of platforms such as the database list,
2. Methods to monitor memory usage are the methods commonly developed for NoSQL platforms to inquire about the memory usage data of nodes,
3. Methods to monitor disk usage are the methods commonly developed for NoSQL platforms to inquire about the disk usage data of nodes.

6 Prototype implementation

A prototype software was developed to show the usability of the Hybrid NoSQL Monitoring Platform architecture. The MongoDB and CouchBase NoSQL platforms were deployed as subsystems in the prototype, which was created with the Java programming language.

Figure 2 illustrates the usage scenario of the prototype system. The developed prototype implements the Hybrid NoSQL Monitoring Platform functions with a restful-based web service that can be accessed via user applications. The Hybrid Platform Web Service provides a programming interface that enables one to monitor various NoSQL platforms on a single platform. This interface can be used through a web-based application that we have developed under the scope of this prototype. This client application was implemented with PrimeFaces technology and demonstrates how to interact with the hybrid platform.

As shown in Figure 2, the system accesses each NoSQL database through APIs provided by the NoSQL database. This API is packaged within the resource handler of the corresponding NoSQL database. The resource handler module is used to access data about the platform's database.

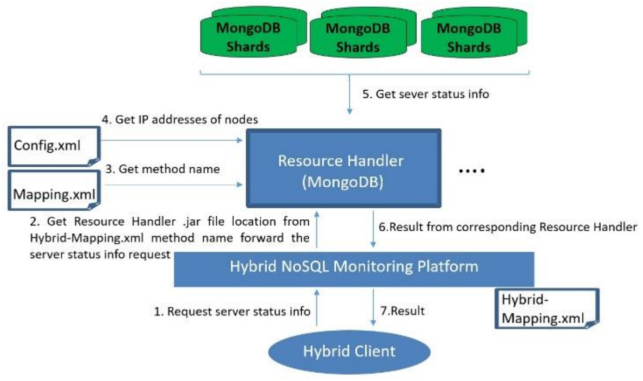
MongoDB Shards
MongoDB Shards
MongoDB Shards
5. Get sever status info
4. Get IP addresses of nodes
Config.xml
3. Get method name
Mapping.xml
Resource Handler (MongoDB)
....
2. Get Resource Handler .jar file location from Hybrid-Mapping.xml method name forward the server status info request
6.Result from corresponding Resource Handler
Hybrid NoSQL Monitoring Platform
1. Request server status info
7.Result
Hybrid-Mapping.xml
Hybrid Client

Figure 2: The usage scenario for hybrid NoSQL monitoring tool.

Tablo 1: An example of mapping hybrid monitoring functions and NoSQL monitoring functions.

	MongoDB monitoring functions	MongoDB monitoring functions	MongoDB monitoring functions
System's monitoring functions	getServerStatus()	getServerStats()	serverStatus()
Memory Usage monitoring functions	getMem()	getRamHdd()	MemoryStatus()
Disk Usage monitoring functions	getDBStats()	getNodeStats()	DBStatus()

The system retrieves the resource handler data of the NoSQL platform to be monitored from the Hybrid-config.xml file, shown in Appendix 1. As mentioned previously, the resource handlers are packed as ".jar" files. Each resource handler has a standard programming interface that is externally accessible and contains code related to platform-specific monitoring methods.

For each resource handler to be added to the system, two configuration files should be created. Mapping.xml is the configuration file where hybrid monitoring methods and lower-level systems' monitoring methods are mapped. An example of the Mapping.xml file is shown in Appendix 2. The other file to be used by the system (i.e., Config.xml) contains the IP addresses of the nodes on the NoSQL platform. An example of the Config.xml file is provided in Appendix 3. The system acquires the IP addresses of the nodes to be monitored by reading the Config.xml file, and it monitors the NoSQL databases on these nodes.

7 Evaluation

Performance tests were conducted to test system availability. The initial results of the Hybrid NoSQL Monitoring platform were first discussed in [13]. The technical details of the computers used in these tests are as follows. The processor used was Intel Core i7 3630QM, equipped with 2. 4 GHz. The memory of the server was 8GM RAM, and the size of the hard drive was 750 GB. We used Java version 1. 8. 0_25 in our implementation. Throughout the tests, nano.Time() was used as the Java timing function [14]. Seven different tests were performed to measure the system performance. In other words, the execution is repeated seven times under different message-loads. In each test, the average delay time in sending a monitoring request and obtaining the results was measured. These tests were repeated for Hybrid API, MongoDB, and CouchBase. The basic purpose was to show that the performance load from the hybrid structure on the system was negligible (in terms of seconds). The results are depicted in Figure 3.

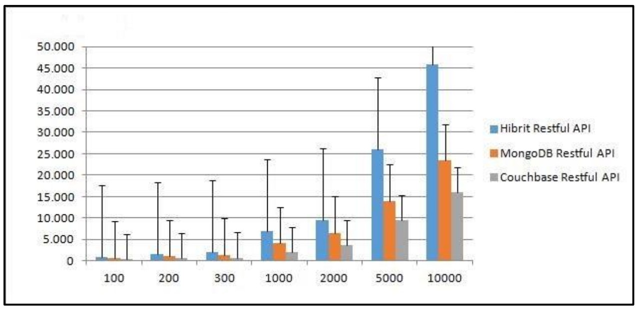

Figure 3: Hybrid restful API performance evaluation results. X-axis stands for iteration number. Y-axis stands for time in milliseconds.

The first three tests were performed for low-scale request volumes. Over one hundred observations, the results indicate that a MongoDB monitoring request for server statustakes 677 ms, while the same type of CouchDB monitoring request takes 302 ms. On the other hand, Hybrid Monitoring reqest for server status takes only 986 ms. These results show that our proposed hybrid platform generates a negligible load on NoSQL platforms. The rest of the test results indicate that if the number of sequential requests on the system increases, the delay time increases exponentially. The main reason for this is that as the simultaneous connection size increases, the system starts using more threads from the thread pool. In turn, this creates an additional need for time for the operating system to perform thread scheduling runs and prioritize the threads.

8 Conclusions and future work

Within the scope of this study, a system architecture was proposed for a hybrid platform that performs real-time monitoring on various big data platform systems. The proposed software architecture runs as a piece of add-on software on NoSQL platforms and provides monitoring tasks without any necessary additional development on these platforms. In our study, we created a prototype of our proposed system architecture. In order to prove the practical usage of the proposed system architecture, performance tests were executed on the prototype application. The test results indicate that the load generated by the Hybrid NoSQL Monitoring Platform is negligible. Testing this system on various types of NoSQL environments will be among our future projects.

9 Acknowledgments

We would like to thank Oguz Aydin and Yusuf Onder for their contribution to this study. Financial support was provided by the TUBITAK project numbered 114E781 and titled "Development of Methods for the Identification of Disinformation and Copyright Violations in Social Media Software with Provenance."

10 References

[1] Han J. "Survey on NoSQL database". *6th International Conference on Pervasive Computing and Applications*, Port Elizabeth, South Africa, 26-28 October 2011.
[2] Leavitt N. "Will NoSQL databases live up to their promise?". *Computer Magazine,* 43(2), 12-14, 2010.
[3] Muthukkaruppan K. "Storage ınfrastructure behind facebook messages". *Proceedings of International Workshop on High Performance Transaction Systems,* New york, NY, 2011.
[4] Pritchett D. "BASE: An acid alternative". *File Systems and Storage*, 6(3), Pages 48-55, 2008.
[5] Wei-ping Z, Ming-xin L, Huan C. "Using mongoDB to implement textbook management system instead of MySQL". *3rd IEEE International Conference on Communication Software and Networks*, Xi'an, China, 2011.
[6] Yishan L. "A performance comparison of SQL and NoSQL databases". *IEEE Pacific Rim Conference*, Victoria, BC, Canada, 2013.
[7] Angles R. "A Comparison of current graph database models". *IEEE 28th International Conference on Data Engineering Workshops*, Arlington, VA, USA, 2012.
[8] Tudorica B, Bucur C. "A comparison between several NoSQL databases with comments and notes". *10th Roedunet International Conference*, Iasi, Romania, 2011.
[9] Mishra V. *Beginning Apache Cassandra Development*, 1st Edition, New York, NY, Apress, 2014.
[10] Griffin M. A Preliminary Exploration of Database Performance for Use With 'big data' projects in the aviation industry. Master Thesis, School of Science LYIT, Gortlee, Letterkenny, Co. Donegal, Ireland 2014.

[11] Robin L, Gibbs C, Coady Y. "MADAPT: managed aspects for dynamic adaptation based on profiling techniques". *Proceedings of the 3rd workshop on Adaptive and reflective middleware*, New York, NY, 2004.
[12] Prechelt L, Unger-Lamprecht B, Philippsen M, Tich W. "Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance". *IEEE Transactions on Software*, 28(6), 595-606, 2002.
[13] Aydın O, Onder Y, Aktas M. "NoSQL büyük veri yapılarının izlenmesine yönelik hibrit sistem platformu". *Basarim-15 Conference*, Ankara, Turkey, 2015.
[14] Lee J K, Lee J Y. "Android programming techniques for improving performance". *3rd International Conference on Awareness Science and Technology*, Dalian, China, 2011.

Appendix A

Example Hybrid-Mapping. xml

```
<resources>

<resource>
<db>MongoDB</db>
<path>. . \MongoDB\target\</path> </resource>
<resource>
<db>Couchbase</db>
<path>. . \Couchbase\target\</path> </resource>

</resources>
```

Appendix B

Example Mapping. xml

```
<methods>
<method>
<hybrid>serverStatus</hybrid>
<original>getServerStats</original>
</method>
<method>
<hybrid>DBStatus</hybrid>
<original>getNodeStats</original>
</method>
<method>
<hybrid>MemoryStatus</hybrid>
<original>getRamHdd</original>
</method>

</methods>
```

Appendix C

Example Config. xml

```
<shards>

<shard>
<host>127.0.0.1</host><port>27020</port> </shard>

<shard>
<host>127. 0. 0. 1</host> <port>27025</port>
</shard>

</shards>
```