Stanisław DENIZIAK, Grzegorz ŁUKAWSKI, Mariusz BEDLA, Adam KRECHOWICZ
DEPARTMENT OF INFORMATION SYSTEMS, KIELCE UNIVERSITY OF TECHNOLOGY, POLAND
7 Tysiaclecia PP Ave., 25-314 Kielce, Poland

A Scalable Distributed 2-layered Data Store (SD2DS) for Internet of Things (IoT) systems

Abstract

The problem of effective storing and processing of data is a very important aspect of every IoT application. To fulfill these requirements, NoSQL scalable datastores are frequently used. Scalable Distributed 2-layer Data Structures (SD2DS) are examples of such systems. SD2DS is a general purpose structure that can be used as a distributed datastore, easy to adapt to many different needs. In the paper, an enhanced architecture of IoT systems (denoted as IoT*) supplemented with SD2DS is proposed and evaluated. The combination brings many advantages, such as the possibility to create one unified structure consisting of many servers accessible by many clients, usage of different media for storage (such as RAM, hard disk, databases and cloud) and a single access method for both data and data sources.

Keywords: IoT, SD2DS, datastore.

1. Introduction

The problem of effective storing and processing of data is a very important aspect of every IoT application. Moreover, the data may be very diverse and variable in size (e.g. from a single floating point value of thermometer readings to multiple MiB video clips). Also, the data may be stored using various media (e.g. RAM, hard disks, SSD, cloud) and thus must be effectively managed for efficient access.

There are many different methods of storing data which may be used in such a case. Classic RDBMSs have already reached their limits in terms of scalability, especially in the case of distributed systems. As an alternative, NoSQL scalable datastores may be used [5]. Their lack of some features of the relational database, such as strong consistency and transactions, is compensated by great efficiency and scalability.

Many modern applications have complex and differentiated persistence requirements which can be hardly fulfilled by one method of storing data. One solution cannot be optimal for many, different needs. The polyglot persistence [5] is an answer for this problem. In this approach, many different methods of storing data are used. Each of them is tailored to specific requirements related with different type of data or user needs. However, the use of a few databases, datastores, file systems, clouds, etc. in one application can be problematic. A Scalable Distributed Two-layer Data Structure (SD2DS) [1] can be used to integrate many methods of data storing. The SD2DS guaranties efficient and scalable storage for varied data. The paper presents an idea of usage of the SD2DS for IoT systems.

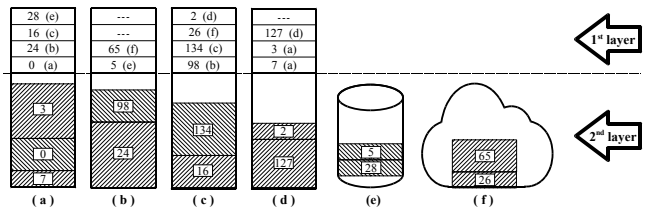

Fig. 1. A sample SD2DS with four buckets in RAM, one disk storage and one cloud storage

The rest of the paper is organized as follows. In the next section, the motivation is presented. In Section 3, the SD2DS structures are described. In Section 4, the SD2DS for IoT systems is presented. Experimental results are described in Section 5. The paper ends with conclusions.

2. Motivation

The classical relational database model is not keeping pace with modern applications and programming models. RDBMS stores data in a structured manner, with relationships between columns and tuples, what is no longer the case for many applications. Moreover, the ACID requirements [5] may be sacrificed for the sake of throughput and scalability.

Such a situation may take place in the case of many IoT applications, where a large number of small chunks of data are to be stored and processed by many clients. There is also no extra needs for preserving many relations between those chunks. This is the case where NoSQL datastores take their part. There are a few general types of NoSQL datastores using different models of data [5]: key–value (e.g. memcached [6]), columns (e.g. Cassandra [7]), documents (e. g. MongoDB [8]) and graphs (e.g. Neo4j [9]). Among them, key–value stores are the most useful in the case of IoT, as we assume that the data has no one, specific structure and comes with various numbers and sizes. Thus, the data is treated as a set of bytes supplemented with a unique key.

Due to the fact that the data may be stored in many different media (RAM, HDD, cloud), the key-value datastore must be flexible enough and take form of a cache memory/index able to integrate many different methods of storing data. The SD2DS can fulfill this requirements due to its adaptability and versatility. Especially, in the case of IoT systems there is strong demand on storing specific metadata like the type of a sensor, the time of read acquisition, the mean error etc. Moreover, due to the nature of IoT systems, the amount of clients that simultaneously access the data might be large.

3. Scalable Distributed Two-Layer Data Structures

Scalable Distributed 2-layered Data Structures (SD2DS) can be categorized as key-value NoSQL stores. Data is stored as the so-called components identified by keys. A component is built of two parts: a header and a body, stored in the first and the second layer, respectively. The header contains the so-called locator (e. g. memory address, IP address, URL) pointing a place where the associated body is stored. The header may also include metadata related with fault tolerance, load balancing and so on. Components are stored in buckets placed in nodes of a multicomputer. The architecture of the SD2DS guaranties that data are stored in an efficient and scalable way. During expansion of the structures new servers are added. This results not only in a larger capacity (data scalability) but it also increases the total performance (throughput scalability).

SD2DS are general purpose structures that can be adapted to many different needs. This also applies to buckets organization. Until now two approaches to this matter have been proposed. In the first one, the data from both the layers are stored together. The buckets are internally divided into two parts containing the headers and the bodies. In the second approach, two types of buckets are introduced, one for each layer. From the bucket point of view, the division of data is done externally. To improve the performance, in both approaches the buckets are stored in the main memory.

The first layer of the SD2DS, called the file, contains component headers. For management of the first layer, there should be used a scalable and efficient algorithm, such as LH* [2], B+ trees [10], etc. Our previous experiments showed that the first one was especially useful in this case. It has many advantages like simplicity, efficiency and predictability. The main drawback of the algorithm is related to a significant amount of the data transferred during expansion (the so-called splits). However, when it is used in the first layer of SD2DS it loses its importance because the headers are small and the splits take a relatively short time.

There are many different ways of organizing the second layer, called the storage. Every bucket can be stored in other form and on other medium, like database, datastore, etc. This gives an opportunity to adjust it to specific needs. Many factors can be taken into account. In the context of IoT systems, such factors may include a variety of sizes and types of data. For example, very small pieces of data (e. g. 32 bit numbers) can be grouped into larger portions and very large pieces of data (e. g. video films) can be divided. Moreover, keys also may have more complex structure, tailored to specific needs, organized in a form of records or objects containing all necessary information to find quickly the requested data.

Figure 1 shows a sample SD2DS with four buckets in the first layer. Locators are denoted as letters in parentheses. There are four buckets in the second layer which are stored in RAM (a to d), one disk/database storage (e) and cloud storage (f). Thanks to the two-layer design, all the data are uniformly accessible through the first layer. Moreover, if we assume that the component bodies stay in the same place once inserted (depending on a particular implementation), a client, after retrieving the locator, may access the data multiple times bypassing the first layer.

4. SD2DS for IoT

In a typical case, an IoT server (denoted as IoT below) is used to store data coming from many different sources (e. g. sensors), connected through gateways. Then, the data are accessed and processed by clients. A modified IoT system architecture, supplemented with SD2DS servers (buckets) for storing the data is presents in Figure 2.

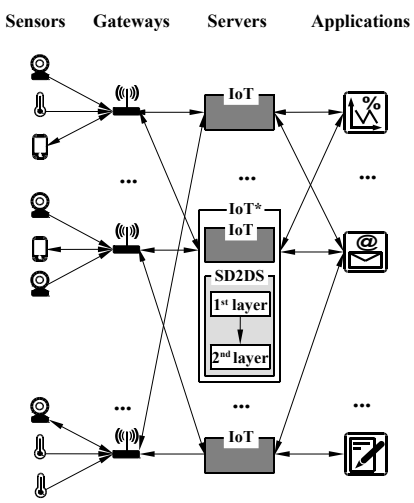

Fig. 2. The proposed SD2DS for IoT architecture

In the enhanced architecture, a server (denoted as IoT*) may contain a part of the SD2DS datastore, playing the role of an index of the data (the first layer) and the data storage (the second layer) using different media. The modification brings more advantages, such as the possibility to serve many clients simultaneously and indexing all the data using the first layer of the SD2DS.

The IoT* server may also play a role of a relay, allowing for direct access to a given sensor/data source. For example, if a client requests the most recent image from a camera, the IoT* server may take the image on demand and immediately return it to the client, simultaneously storing the image as a component in the SD2DS for further access. If it is impossible to take an image currently, the server may return the image stored most recently instead. This way, the sensor may be effectively shared between many clients (and servers).

5. Experimental results

For evaluation of the proposed SD2DS IoT architecture, an experiment using a real multicomputer SD2DS implementation was performed. From 1 to 16 servers were used, each server stored one bucket in its main memory. The SD2DS consisted of the first layer managed with SDDS LH* and the second layer using a simple expansion algorithm. The clients were allowed to perform one of the two simple operations on the datastore: add and get.

Figure 3 shows the results of the experiment. The vertical axis shows the average time of an add/get operation performed on the SD2DS, using different number of servers (buckets) simultaneously (horizontal axis). Three different component sizes were tested, thus the average processing time for larger components became longer.

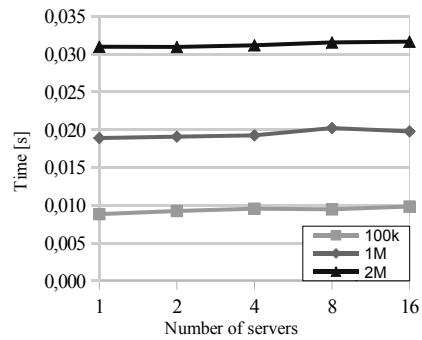

Fig. 3. The average time of adding records with different sizes

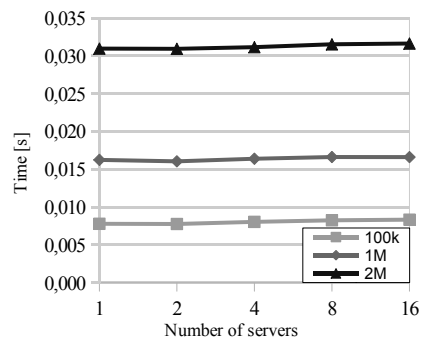

Fig. 4. The average time of getting records with different sizes

The results show that the SD2DS is capable to scale into many servers if needed, while keeping almost constant throughput from a single client point of view. Moreover, as more servers are used, the more clients may access the structure simultaneously without the risk of becoming a bottleneck.

6. Conclusions

The paper contains an outline of the Scalable Distributed Two-layer Data Structures (SD2DS) architecture and a proposal of using the SD2DS for the need of IoT systems. We propose an extended architecture where the SD2DS supplements servers of

a classic IoT system. The combination brings many advantages, such as the efficient scalability, the heterogeneous storage (RAM memory, hard drives, databases, cloud etc.), the possibility to create a unified structure connecting many servers and accessed by many clients simultaneously.

The SD2DS is a general, very flexible idea which may be adapted to specific needs. The presented results of our experiments (Figures 3 and 4) show that the structure may easily scale to many servers while keeping the high throughput (the limit of maximum 16 servers results from our hardware limitations). The same results comes from our previous experiments [1, 4].

The extended IoT architecture assumes that all the servers are connected using the SD2DS into a single unified structure (Figure 2). The servers (denoted as IoT*) are supplemented with the buckets for indexing and storing the data. Moreover, we propose the possibility to access a given sensor/camera on demand, returning the most recent reading to a client. This way the data sources are handled the same way as the data itself (using the first layer of SD2DS), creating many new interesting possibilities, which we are going to study in the future.

The SD2DS is the most efficient in the case of relatively large component sizes, starting from hundreds of kilobytes [1, 4]. Thus, in the case of many smaller portions of data (e. g. single numbers representing thermometer readings) they should be aggregated and stored as a single component. Another option is to store the data directly in the first layer of the SD2DS, without the usage of the locator. Both options will be studied in our future research.

The research used equipment funded by the European Union in the Innovative Economy Programme, MOLAB - Kielce University of Technology.

7. References

[1] Sapiecha K., Łukawski G.: Scalable Distributed Two–Layer Data Structures (SD2DS). IGI Global, IJDST, 4(2), pp. 15–30, 2013.
[2] Litwin W., Neimat M.A., Schneider D.A.: LH* – A Scalable, Distributed Data Structure. In ACM TODS, vol. 21, No. 4, pp. 480–525, 1996.
[3] Di Pasquale A., Nardelli E.: Scalable Distributed Data Structures: A Survey. In Proceedings of WDAS, vol. 9, pp. 87–111, 2000.
[4] Krechowicz, A., Deniziak, S., Łukawski, G., Bedla, M.: Preserving Data Consistency in Scalable Distributed Two Layer Data Structures. Beyond Databases, Architectures and Structures, CCIS, 2015, p. 126–135.
[5] Sadalage, P. J., Fowler, M.: NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Addison-Wesley, Upper Saddle River, NJ, 2013.
[6] memcached - a distributed memory object caching system, http://memcached.org {accessed 4th September 2015}
[7] Cassandra: The Apache Cassandra Project, http://cassandra.apache.org {accessed 4th September 2015}.
[8] MongoDB http://mongodb.org {accessed 4th September 2015}.
[9] Neo4j, the World's Leading Graph Database, http://neo4j.com {accessed 4th September 2015}.
[10] Litwin W., RP*: A Family of Order Preserving Scalable Distributed Data Structures, VLDB, 1994, pp. 342–353.

Received: 21.04.2015 Paper reviewed Accepted: 02.06.2015

Stanisław DENIZIAK, DSc, eng.

He graduated of Faculty of Electronics of the Warsaw University of Technology, defended his doctoral thesis in 1994, and habilitation in 2006. He is the head of the Division of Computer Science of the Kielce University of Technology. His research interests include design of embedded systems, Internet of things, logic synthesis for FPGAs. He is a member of IEEE and IEEE Computer Society.

e-mail: s.deniziak@tu.kielce.pl

Grzegorz ŁUKAWSKI, PhD

Grzegorz Łukawski is an Assistant Professor in the Department of Information Systems at Kielce University of Technology. He received a PhD in computer science from Gdansk University of Technology. His research interests include distributed systems, load balancing and fault tolerance.

e-mail: g.lukawski@tu.kielce.pl

Mariusz BEDLA, PhD

Mariusz Bedla is an Assistant Professor in the Department of Information Systems at Kielce University of Technology. He received a PhD in computer science from Gdansk University of Technology. His research interests include object-oriented technologies, parallel and distributed systems, databases and datastores.

e-mail: m.bedla@tu.kielce.pl

Adam KRECHOWICZ, MSc

Adam Krechowicz is a Teaching Assistant in Department of Information Systems, Kielce University of Technology, Poland. His research interest include Transaction processing, Service Oriented Architectures and Mobile Robots.

e-mail: a.krechowicz@tu.kielce.pl