TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON)
31 Oct - 3 Nov 2023. Chiang Mai, Thailand

Benchmarking Database for A Case Charging of Data Sponsor in Telecom

Van Chuong Do*, Van Duong Nguyen*, Cong Dan Pham*, Ngoc Tien Nguyen*
Ngoc Hieu Pham**, Duc Dung Nguyen**
*OCS Research Center, Viettel High Technology, Viettel Group, Hanoi, Vietnam
**Global Technical Center, Viettel Network Corporation, Viettel Group, Hanoi, Vietnam
(chuongdv2, duongnv21, danpc, tiennn18, hieupn, dungnd18)@viettel.com.vn

***Abstract*—Database plays an important role in software architecture and greatly influences system performance. In this paper, we evaluate how two databases, Cassandra and Aerospike, adapt the data sponsor concurrency problem in the telecom sector. To obtain the evaluation result, we provide a method by building the test cases in different scenarios.**

***Keywords*—Database, Benchmarking, Cassandra, Aerospike, 5G, Data Sponsor, Flow Charging**

I. Introduction

As we mentioned, databases have an important role in a software system. Especially in the telecom sector, when the number of customers increases, the explosion of new services in the 4G 5G era. These challenges set the requirements for large data storage and transaction processing performance of the database.

This paper discusses the concurrency problem of data sponsors in telecom and considers how databases adapt to this problem. A data sponsor is a service that the customer service provider (CSP) provides data offers to businesses or individuals with many subscribers to use a sharing account. When many devices share a sponsor, the transaction per second (TPS) requirement for concurrent device access needs a database with higher performance. Data sponsor and pricing models are studied in previous research, for example, [1], [2], [3], or in IoT domain as [4]. Sponsored data is a new pricing model that allows content providers (CPs) to subsidize some of this cost. While much smart data pricing (SDP) research has introduced various ways to charge end-users for data access [5], [6], [7] sponsored data instead introduces a new party to data pricing: content providers (CPs). This model is applied in business by many content providers (CPs) company as Facebook [8], Internet Service Providers (ISPs) such as AT&T and Verizon [9], [10], Syntonic Wireless and DataMi, U.S Federal Communications [11]. Data sponsors in the 4.0 era is a favorite service of large enterprises because of its flexible application capabilities in a variety of purposes, bringing a lot of value in connecting with customers. Industry groups should apply Data Sponsor flexibly and reasonably to maximize their use and bring efficiency to their brand promotion or customer care campaigns. Database technologies help telecom service providers to provide this service.

Database plays an important role in software architecture and greatly influences the system's performance. This is especially important for the online charging system (OCS), with most of the time spent querying and updating the database. There are two main types of databases: SQL and NoSQL.

SQL has a long history and supports vertical scaling. ACID (Atomicity, Consistency, Isolation, Durability) transaction NoSQL supports horizontal scaling. The horizontal scale support helps NoSQL have the advantage of dealing with data in industries rising over time. In addition, removing properties ACID helps the transactions' latency become much smaller. The transference from SQL to NoSQL helped us succeed with our OCS version 3.0. We consider benchmark performance for two NoSQL Databases used in our OCS for the Data Sponsor problem: Cassandra and Aerospike. Casandra this is a popular database and free, and Aerospike is an in-memory database that is faster and licensed software.

- Cassandra: It is a distributed database and supports wide column mode. It provides zero downtime ability, non-single fail of point, linear scalability [12]. It also supports the language CQL, likely SQL, and is friendly with users. In addition, it also helps online transactions. Cassandra supports well for Big Data problem [13], [14], [15].
- Aerospike: It is a distributed database supporting multi-model. It supports an in-memory database engine then it has high performance. Aerospike is suitable for applications requiring low latency. Moreover, Aerospike supports scalability with zero downtime. It also supports the language AQL as SQL.

Our contribution is that we build the test cases to give a detailed performance evaluation for two NoSQL Cassandra and Aerospike. We also evaluate how these databases adapt to the concurrency problem of mobile data sponsor. How the criteria and settings were considered with two different databases for evaluation? Based on the characteristics of the data sponsor problem: High levels of concurrent occur on small numbers of records. Customer information is usually stored on NoSQL databases to meet the performance and speed of transaction processing. Therefore, we evaluate two typical NoSQL databases in terms of high performance and popularity: Aerospike and Cassandra. The main consideration is the performance and latency of the two databases when the frequency of high transactions occurs on the same record.

The rest of the paper is as follows. Section II details the system architecture and mentions the transactions which are read and written on the database. Section III describes how to implement Datas chema and Simulator. Section IV delineates our testing methodology and the simulation result. Finally, Section V summarizes the results and discusses the direction of future research.

II. Architecture

A. Software architecture

The architecture of our OCS for the data sponsor problem is shown in Figure 1.

Middleware Systems: The middleware system helps to provide association and division between different network elements and services in telecommunications. Middleware includes MSC, SMSC, and IMS.

- MSC: A Mobile Switching Center (MSC), sometimes referred to as a Mobile Switching Server (MSS), is a component of 2G and 3G cellular networks that approves or rejects voice communications from one device to another and helps manage roaming.
- SMSC: A Short Message Service Center (SMSC) is a network element in the mobile telephone network. Its purpose is to store, forward, convert and deliver Short Message Service (SMS) messages
- IMS (IP Multimedia Subsystem): The IP Multimedia Subsystem or IP Multimedia Core Network Subsystem (IMS) is a standardized architectural framework for delivering IP multimedia services

Gateway: Integrates with Middleware systems through protocols such as SMPP, DCC, and Restful to convert into internal protocols and route to applications in the Online Charging System.

Applications: The set of applications plays a crucial role in charging for various Telco services such as Voice, Data, SMS, and PCRF, as well as NonTelco services like IoT and Mobile Money. These applications are responsible for processing charging requests and interacting with the Gateway to receive incoming requests. After performing the charging process, the applications also communicate with the ABM (Account Balance Management) system to retrieve and update data profiles based on the charging activities.

ABM: Account Balance Manager (ABM) is a module playing the role of a proxy for querying and updating data on a database (DB). DB is a database software playing the role of storage data operations. For non-supporting transaction NoSQL database, ABM also plays simulator transactions suitable for operational purposes. The use of ABM helps separate logic application and logic storage data and helps to simplify in development of operations. To simulate transactions along with ACID properties, ABM performs:

- ABM performs data querying before updating and saves the data state for rollback purposes. The rollback mechanism helps ABM achieve the atomicity property of the transaction.
- ABM ensures data consistency by validating the state of the data before updating it into the database. For example, the account balance must be greater than 0.
- ABM employs the Multiversion Concurrency Control (MVCC) technique [16] to ensure isolation during concurrent data processing. MVCC is a concurrency control technique that allows transactions to read and write data simultaneously without causing data conflicts. It ensures that each transaction views data in a specific version, avoiding inconsistent data reads caused by other transactions performing simultaneous updates. The lightweight transaction feature in Cassandra and the check and set mechanism in Aerospike are utilized to support the implementation of MVCC in ABM.
- Durability is ensured by NoSQL databases.

NoSQL Database is a database software playing the role of storage data operations.

B. Transaction

ABM is a simulator transaction suitable for processes combined with available database support. Read-Write transactions perform Charging and billing operations. Aerospike supports read-write transactions by UDF or Record Read-Modify-Write (compare and set) [17]. Cassandra also supports read-write transactions by lightweight transaction [18]. Charging flow is performed based on comparing and setting features of the database as follows:

- Step 1: The application performs query client data from the database.
- Step 2: The application performs charging previous using the data session and reserves a quota for the next session based on business policy. (When subscribers access the internet and use data service in some pre-defined duration time, we call a session).
- Step 3: The application updates the client's data on the database through ABM.
- Step 4: ABM receives the profile update message of the client, including the updated profile and actions acted on the profile.
- Step 5: ABM builds request compare and set, updates client's profile, and sends request update to the database.
- Step 6: If ABM receives a successful response from the database, it sends the successful response to the Application. Suppose it gets failed response because of the failures of compare and set, then ABM proceeds to retry. If ABM receives the failed response because of other errors, it returns failure to the application.

In the case of failure of request update because of compare and set fail, data have changed, and request updates have been built based on old data. The flow retry is following:

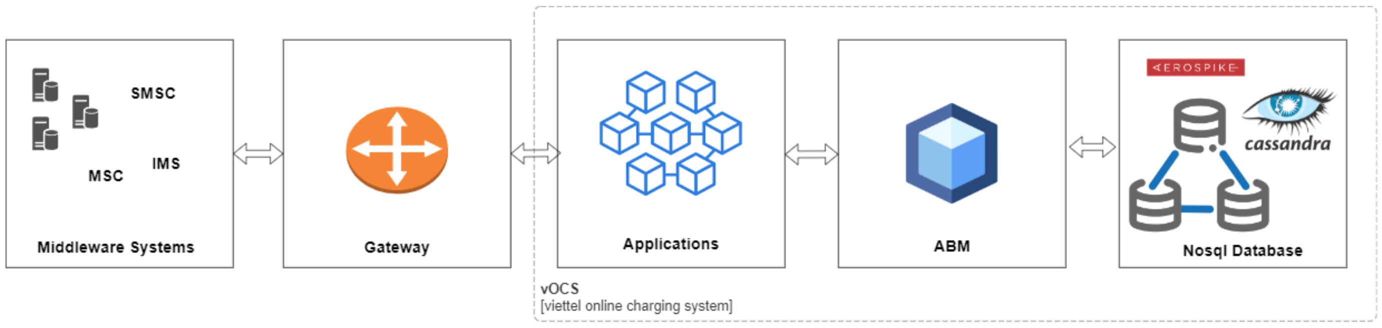


Figure 1. Software architecture

- Step 1: Verify if the condition retry is satisfied or not. If the condition retry is invalid, it returns failure to the Application. If the condition retry is valid, perform step 2.
- Step 2: ABM sends a query request to the database to get the latest profile of the client.
- Step 3: If ABM fails to receive the client's profile, it returns failure to the Application. If it received the client's profile successfully, perform the next step.
- Step 4: Update profile based on a received request from the Application. Build request compare and set and sends new request update profile to database.
- Step 5: If the database returns a successful response, it returns success to the Application. If it returns a failure response, perform step 1.

III. Implementation

A. Dataschema

The information of the client is stored in a profile. When testing performance for the data sponsor problem, we use a simple profile as in Table 1.

TABLE 1. Profile

Name of attributes	Data type	Description
subId	String	client's id
msisdn	String	client's number
email	String	email address
address	String	place of residence
firstName	String	Name
birthDay	long	birthday
sex	byte	gender
generation	long	logic clock for update profile
balanceMap	map<long, Balance>	list of client's account

TABLE 2. Balance

Attributes	Data type	Description
balanceId	long	Account Id
balType	long	Account type (prepaid, postpaid, promotion)
gross	long	current money
consume	long	spent money
reserve	long	reserve money from client

B. Simulator

We use the java programming language to develop simulators application and ABM. Application and ABM interact with each other by TCP/IP protocol. We use lib micro chassis developed based on the netty framework to communicate between the application and ABM. The transmitted messages between ABM and the application are in JSON format. We also use the library DataStax java driver version 4.14.1 to integrate with Cassandra [19]. We also use Aerospike java client version 4.4.20 to integrate with Aerospike [20].

IV. Experimentation

A. Methodology

1) Data preparation: For the data sponsor problem, in the actual context, maybe there are a hundred thousand to a million devices charging on the same account. Therefore, we mainly test performance when updating data for the 1 record on the database. We consider only one record for one data sponsor because we consider the concurrency problem, not storage. We evaluate the performance and TPS for query updates simultaneously for all devices in the same data sponsor. The data we use is a profile including entire attributes and has a size of 150 bytes. The profile is configured with replication factor = 3 on both databases, Cassandra and Aerospike. The consistency level for Cassandra is SERIAL. The consistency level for aerospike is ALL.

2) Software and Hardware for Benchmark: We use Cassandra server version 4.3.0 and Aerospike version 5.0. Cassandra and Aerospike were installed on the Red Hat Enterprise Linux system version 7.8. Table 3 lists the configurations of deployed servers in this study. .

TABLE 3. Hardware for Benchmark

	Database servers
CPU	Intel (R) Platinum 8260 @ 2.40GHZ (96cpus)
RAM	256GB 2933 MT/s
Network	BCM57414 NetXtreme-E 10Gb/25Gb RDMA
Disk	Dell Express Flash NVME P4610 1.6TB SFF

We tune the configuration following the guide optimization of Cassandra and Aerospike.

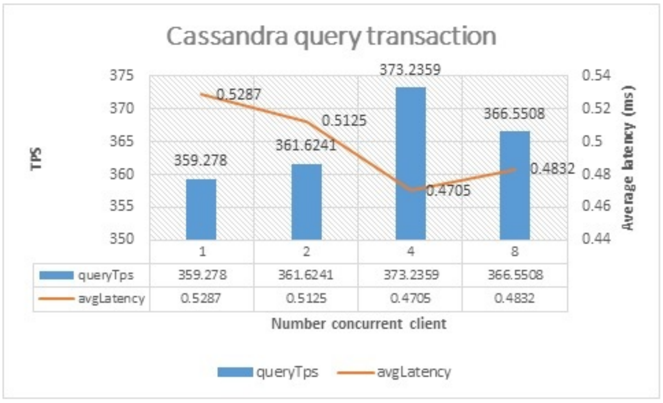
3) Test scenario: We use different test scenarios to discover the TPS maximum of Cassandra and Aerospike.

- Scenario 1: 1 application performs charging flow.
- Scenario 2: 2 applications perform simultaneously charging flow.
- Scenario 3: 4 applications perform simultaneously charging flow.
- Scenario 4: 8 applications perform simultaneously charging flow.

4) Get results: For every 1s, we log out the TPS, latency, and error results. To avoid bias in a performance evaluation, we take the average of 100 consecutive measurements.

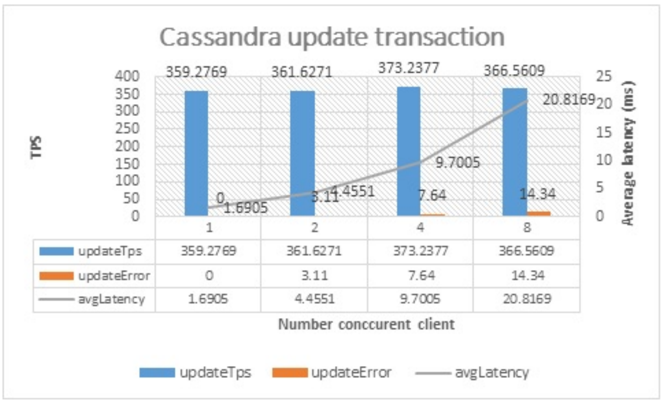
In the next Section, we present the test result and discuss how it meets for concurrency problem. In the figures, the numbers $1, 2, 4, 8$ refer to the number of applications.

B. *Test result*



	1	2	4	8
queryTps	359.278	361.6241	373.2359	366.5508
avgLatency	0.5287	0.5125	0.4705	0.4832

Figure 2. Cassandra query

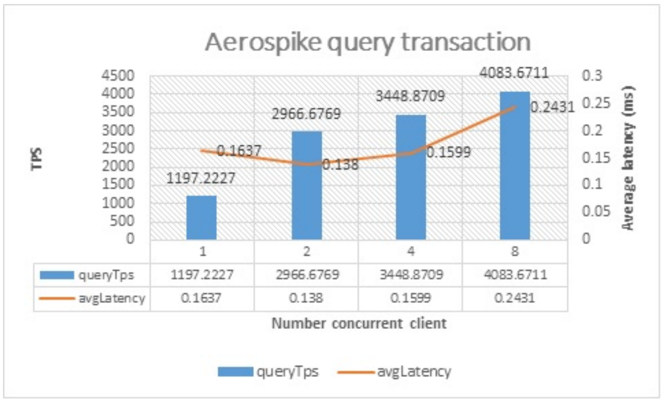


	1	2	4	8
updateTps	359.2769	361.6271	373.2377	366.5609
updateError	0	3.11	7.64	14.34
avgLatency	1.6905	4.4551	9.7005	20.8169

Figure 3. Cassandra update

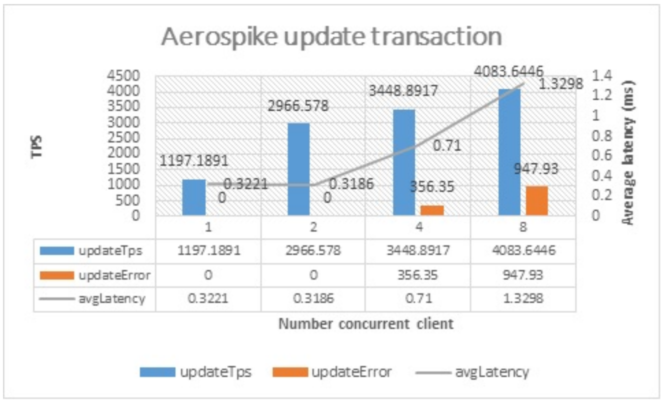
1) Cassandra: According to Figures 2 and 3, TPS seems not to change when the number of concurrent client increase. The difference between the highest TPS and the lowest TPS is 14 TPS. Similarly, query latency also does not change significantly when the number of concurrent clients increases. The difference between the highest and lowest latency is only 0.05ms. However, update latency increases linearly when the number of concurrent clients increases. When this number increases, the error rate update also increases linearly. The amount of error update is also corresponding to the interruption of service and affects the customer experience. Furthermore, Cassandra archives 359 TPS and query latency 0.5287ms, and update latency 1.6905ms is the best-used case.

Due to the nature of read-modify-write transactions, the query TPS is equal to the update TPS. A query command will be executed only after a successful update. When the update TPS reaches its maximum (as the level of concurrency increases, leading to conflicting updates and affecting latency), however, the query TPS has not reached the maximum of the entire database. Therefore, the latency of queries is not affected (or affected only minimally).



	1	2	4	8
queryTps	1197.2227	2966.6769	3448.8709	4083.6711
avgLatency	0.1637	0.138	0.1599	0.2431

Figure 4. Aerospike query



	1	2	4	8
updateTps	1197.1891	2966.578	3448.8917	4083.6446
updateError	0	0	356.35	947.93
avgLatency	0.3221	0.3186	0.71	1.3298

Figure 5. Aerospike update

2) Aerospike: Similar to the case of Cassandra, the TPS of query and update are the same. TPS increases when the number of concurrent clients increases. TPS at the number of concurrent clients 8 is 4 times more than TPS at the number of concurrent clients 1. When the number of concurrent clients increases from 1 to 4, query latency does not change significantly. However, when this number increases to 8, query latency increases significantly 0.08ms. For update transaction, latency and the amount of error update increases quickly when the number of concurrent client increases from 2 to 8. In the best-used case, Aerospike attains TPS 2966 with a query latency of 0.138ms and an update latency of 0.3186ms.

TABLE 4. DATABASE PERFORMANCE

	TPS	Query latency (ms)	update latency (ms)
Cassandra	359.278	0.5287	1.6905
Aerospike	2966.6769	0.138	0.3186

3) Evaluation result of Cassandra and Aerospike: The test is conducted to discover the maximum TPS achievable when multiple applications are concurrently updating the same data in the database. Simultaneously, the test also evaluates the impact of the number of concurrent updates on the system's latency, the success rate of messages (with retries), and the corresponding error transactions per second.

The Multiversion Concurrency Control technique, allows multiple concurrent clients to read data from one record. However, at any given time, only one update command can be successfully executed. When a record is modified, and its version is increased, update requests based on the previous version will fail and require retries by querying the record with the latest version and building update requests based on this data. After three unsuccessful retries, the request is considered an update error.

Therefore, as the number of concurrent clients increases, if the number of requests per second has not exceeded the maximum TPS, the latency will increase (due to the need to retry updates caused by conflicts), until too many retries increase the number of error transactions. When the requests per second exceed the maximum TPS, even retrying updates cannot improve the situation, leading to a limitation where the success of TPS cannot increase, and the latency worsens.

To overcome this limitation, systems that require higher-frequency concurrent data updates need to implement alternative measures such as:

- Dividing data into smaller data records to reduce the impact on the same data.
- Using solutions to merge multiple small transactions into a larger transaction for execution (near-real-time charging or offline charging).

For Cassandra and Aerospike, the number of update errors increases linearly as the level of concurrency rises. When update errors occur, it indicates a service failure. Therefore, we will only consider the TPS and latency of Aerospike and Cassandra in the scenario where there are no update errors. As in Table 4, Aerospike has TPS 11 times greater than Cassandra. The query latency of Aerospike is faster, approximately 0.4ms, than Cassandra. In addition, the update latency of Aerospike is 1.3ms faster than Cassandra. The significantly slower Cassandra than Aerospike is due to database architecture. Two main points that help Aerospike dominate are the following:

- Because Aerospike is an in-memory database (IMDB), data is processed in RAM. Meanwhile, Cassandra store data on DISK. The difference in the write-read speed of RAM and DISK leads Aerospike to attain better latency in write-read data.
- Cassandra bears a high cost of performance when using the Paxos consensus algorithm to install lightweight transactions.

4) Suitable level for data sponsor: Every client makes a load in sharing models of 1 sponsor - 1000 devices, 1 sponsor - 10,000 members, 1 sponsor - 100,000 members, 1 sponsor - 1000,000 members, with average of every 5 minutes, a member requests again to use the resource of the sponsor. TPS concurrent update on a sponsor is shown in Table 5.

TABLE 5. DATASPONSOR TPS

No.	Case	Concurrent TPS per Sponsor data
1	1 sponsor 1000 members	3.3 tps
2	1 sponsor 10,000 members	33 tps
3	1 sponsor 100,000 members	333 tps
4	1 sponsor 1000,000 members	3333 tps

We can see that Cassandra meets enough of the case of 1 sponsor and 100,000 members. Aerospike can satisfy the problem of 1 sponsor and nearly 1000,000 members. However, in actual context implementation, when the number of members gets millions, Cassandra also adapts to this problem with a simple technique. That is to open more sponsors and divide members into different sponsors.

V. CONCLUSION

The evolution of the 4G 5G era, with the explosion of many millions of customers, requires a new technology database. In real-time charging for 5G service, to satisfy customer experience and operate smooth services, we use the latest modern database as Aerospike. In the paper, we evaluate how two NoSQL, Cassandra and Aerospike adapt the concurrency problem for data sponsors.

Some limitations to mention in the scope of the review:

- The simulation behavior because the transaction into the system may be different from the actual customer behavior.
- The review is narrow in scope, assuming that every customer transaction on the same data sponsor is updated to the data sponsor's common data, but in reality, there are many methods of organizing and processing data. other data (for example offline charging, aggregating charges by day, hour, and minute cycle) to reach this problem of data sponsor.

In the future, we will continue to discover business models in the real world, new services, and new operations and study how the database can adapt. Specially, we will study how to apply data sponsor service in IoT service, and other digital services in 5G, and find out the performance of database technologies for these services.

References

[1] C. Joe-Wong, S. Sen, and S. Ha, "Sponsoring mobile data: Analyzing the impact on internet stakeholders," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1179–1192, 2018.

[2] Z. Wang, L. Gao, T. Wang, and J. Luo, "Monetizing edge service in mobile internet ecosystem," *IEEE Transactions on Mobile Computing*, vol. 21, no. 5, pp. 1751–1765, 2020.

[3] Y. Zhao, Q. Tan, X. Xu, H. Su, D. Wang, and K. Xu, "Congestion-aware modeling and analysis of sponsored data plan from end user perspective," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. IEEE, 2022, pp. 1–11.

[4] Y. Zhu, W. Bao, D. Wang, and J. Liu, "A stackelberg queuing model and analysis for the emerging connection-based pricing in iot markets," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2022, pp. 417–425.

[5] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "Smart data pricing (sdp): Economic solutions to network congestion," *Recent advances in networking*, vol. 1, pp. 221–274, 2013.

[6] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "Incentivizing time-shifting of data: a survey of time-dependent pricing for internet access," *IEEE Communications Magazine*, vol. 50, no. 11, pp. 91–99, 2012.

[7] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "Smart data pricing: Using economics to manage network congestion," *Communications of the ACM*, vol. 58, no. 12, pp. 86–93, 2015.

[8] J. Malcolm, C. McSherry, and K. Walsh, "Zero rating: What it is and why you should care," *blog, Deeplinks, Electronic Frontier Foundation*, vol. 4, 2016.

[9] AT&T. (2017) At&t sponsored data. [Online]. Available: http://www.att.com/att/sponsoreddata/en/index.html#fbid=fNjHshoHkg_

[10] Verizon. (2017) Freebee data. [Online]. Available: http://freebee.verizonwireless.com/business/freebeedata

[11] J. Kastrenakes. (2017) Republicans are ready to take down the fcc. [Online]. Available: http://www.theverge.com/2017/2/10/14577404/republicans-fcc-reform-modernization-removeconsumer-protections

[12] [Online]. Available: https://cassandra.apache.org

[13] A. Chebotko, A. Kashlev, and S. Lu, "A big data modeling methodology for apache cassandra," in *2015 IEEE International Congress on Big Data*. IEEE, 2015, pp. 238–245.

[14] K. Anusha, N. Rajesh, M. Kavitha, and N. Ravinder, "Comparative study of mongodb vs cassandra in big data analytics," in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2021, pp. 1831–1835.

[15] N. Bodiroga, M. Antić, P. Zečević, and M. Bjelica, "Evaluation of fleet management data collection backend using cassandra database," in *2021 Zooming Innovation in Consumer Technologies Conference (ZINC)*. IEEE, 2021, pp. 177–181.

[16] P. Bernstein and N. Goodman, "Concurrency control in distributed database systems," *ACM Computing Surveys*, vol. 13, pp. 185–, 06 1981.

[17] Aerospike. Developers: Understanding aerospike transactions. [Online]. Available: https://aerospike.com/blog/developers-understanding-aerospike-transactions/

[18] Cassandra. Cassandra documentation. [Online]. Available: https://cassandra.apache.org/doc/latest/cassandra/architecture/guarantees.html

[19] dataStax. datastax java driver. [Online]. Available: https://docs.datastax.com/en/developer/java-driver/4.14/

[20] Aerospike. Java client library release notes. [Online]. Available: https://download.aerospike.com/download/client/java/notes.html

Van Chuong Do received his B.Sc. degree in Information Technology in 2020 from Hanoi University of Science and Technology, Vietnam. He is currently a software engineer in Viettel High Technology, Viettel Group. He has many years of experience with distributed databases and high-performance computing, especially for telecommunications. His research interests include distributed computing and cloud computing.

Van Duong Nguyen received his B.Sc. degree in Mathematics and Informatics Engineering from Hanoi University of Science and Technology, Vietnam in 2015. He is currently manage a database team of the Research Center for OCS (Online Charging Platform) of Viettel High-Tech Industry Corporation (VHT), Viettel Group. He has many years of experience in research and development of 5G core systems such as online charging system. His research interests include telecommunications core networks and database technologies.

Cong Dan Pham received his PhD. Degree in Probability and Statistics from Aix-Marseille University, France in Jun 2014. He has academic experience in universities like Aix-Marseille University, Hanoi University of Education, and industry experience companies like Viettel Group. He has published various journal and conference papers in IEEE and ScienceDirect. His research interests include probability and statistics, data science and machine learning.

Ngoc Tien Nguyen received his B.Sc. degree in Information Technology in 2020 from University of Engineering and Technology - Vietnam National University, Hanoi. He is currently a software engineer in Viettel High Technology, Viettel Group. He has industry experience with databases and high-performance computing, especially for telecommunications.

Ngoc Hieu Pham received his B.Sc. degree in Electronics and Telecommunications from Ha Noi University of Technology, Vietnam in 2009. He is currently managing the OCS team of the Global Technical Center of Viettel Network Corporation (VTNET), Viettel Group. He has many years of experience in designing, implementing and maintaining OCS systems. His research interests include telecommunications core networks and Cloud Computing.

Duc Dung Nguyen received his B.Sc. degree in Mathematics and Informatics in 2010 from Hanoi University of Science - Vietnam National University, Hanoi. He is currently an IT engineer in Viettel Network, Viettel Group. He has industry experience with databases and high-performance computing, especially for telecommunications.