Wireless Pers Commun (2017) 94:149–164
DOI 10.1007/s11277-015-3161-0

Cloud System for User-Customized Mobile Interface Sharing in Wireless Environment

Sanghyun Park[1] · Hoyong Ryu[2] · Kuinam J. Kim[3] · Jinsul Kim[1]

Published online: 26 February 2016

Abstract In this paper, we use a network-based cloud environment, the software keyboard is built into the smart mobile devices will be customized to provide the user interface. Customized user interface using a network-based interface for the cloud as the optimized interface with a variety of individual application environments, user will have a desired interface for their usage. If the user requests the Node.js and JQuery-based interface to provide an interface, it will provide an interface quicker. The server processes the data quickly and builds server data. There has an algorithm which was developed to return the existing MongoDB. We will use M/M/1, M/M/C Queueing Model for using performance analysis and testing existing MongoDB Server which applied the algorithm to provide and manage user interface effectively. In addition, We propose SLAN to share data between among users, and test this method.

Keywords Software keyboard · MongoDB · Mobile interface · Cloud service · User-customized mobile interface · Cloud system

✉ Kuinam J. Kim
harap123@hanmail.net

✉ Jinsul Kim
jsworld@jnu.ac.kr

Sanghyun Park
Sanghyun079@gmail.com

Hoyong Ryu
hyryu@etri.re.kr

[1] School of Electronics and Computer Engineering, Chonnam National University, 77 Yongbong-ro, Buk-gu, Gwangju 500-757, Republic of Korea

[2] Smart Network Research Department, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

[3] Department of Convergence Security, Kyonggi University, Suwon, Korea

1 Introduction

In general, we use a mouse interface with a hardware keyboard and the X, Y coordinates of the QWERTY format to control the PC. This is a wired or wireless mouse, anyone can use quickly and easily. However, the hardware keyboard does not vary in type, if you want to have a hardware keyboard of your own, you need to customize. Recently the interface in smart devices and Tablet PC has developed increasingly which changed from hardware to software interface to control. However, the software which we provided in mobile devices has a limitation because hardware and software doesn't compatible. As a cloud services offering a variety of smart mobile environment, users will be provided environment which they want to use. Currently we mainly use web storage IaaS (Infrastructure as a Service) [1] store data using a cloud [2] service Network anytime, and share data across a variety of devices. In addition, we can see the data in a variety of device configurations using the viewer program which is provided with the dedicated program. Smart mobile device and the application is also being developed with high specification, and can change the internal system to suit individual usage. The first, Android-based smart mobile devices, you can change the phone number keypad, texting layout, the layout of the main keyboard keypad to suit your usage. Users can download the application anytime. And we can change the soft-keyboard easily. However, the interface just a background image if we use physical interface, it will have some limitation which make the changed isn't easy. In general, the layout is in the application service, but the type is limited. Also it can't easily share interface with the other side of the interface which is disadvantage. In this paper, we have a wide variety of environments to provide a customized interface efficiently. The interface is based on the user's environment to use with a variety of applications that fit the customized interface. We propose that an interface is provided by the cloud environment [3], and user create their own layout platform. Further, in order to provide efficient and quick interface MongoDB [4] was used but it has disadvantages. So, this paper proposes a new algorithm is applied into MongoDB [5] server using Queueing Model [6]. Performance analysis and testing is proved through experiments.

2 Related Works

The interfaces with variety of smart mobile devices are provided by the environment. "BrailleType: Unleashing Mobile Phones Touch Screen over Braille" [7] thesis has developed in the United Kingdom. As shown in Fig. 1, the visually impaired can easily send text messages. This application is mainly used in smart mobile devices and is based

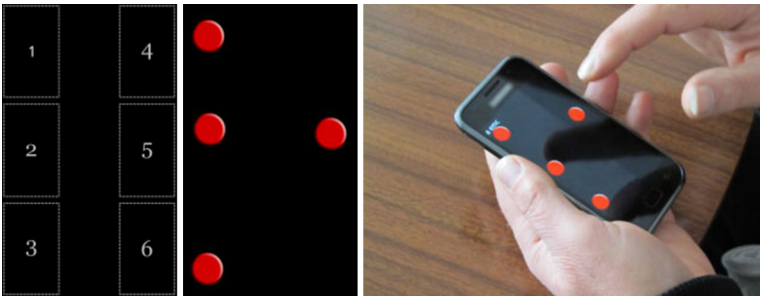


Fig. 1 Braille touch interface for visual handicap

on the Braille system is configured differently; it can speed up the typing about six times than the traditional way. The interface is what you see in Fig. 1, there are six parts, such as the button area, use your finger when you touch the display area of the air button, voice output alphabet. This application is made for low-vision or blind users, but it is effective to control other software.

Figure 2 show an "iGrasp Grasp-based Adaptive Keyboard for Mobile Devices" [8] by recognizing the user's hand position in the screen, the application will change the location of the software keyboard. The application aware of the location of the user's hand with the Arduino and Capacitive Touch Sensor, the shape and position of the keyboard is changed according to the user's hand position. The keyboard is split in half, grab the keyboard with both hands, grasp with one hand the keyboard is changed into one. In addition, by using a sensor, and recognizing the user's hand position, the position of the keyboard will change according to the position. While this application has the advantage that it can be used quickly because they can change the position according to the user's hand position, but the user cannot change their desired interface layout.

Software interface and related papers are user-developed software interface, typing quickly by using a software interface for the efficiently purpose. Typically, the software interface to change the layout of an application is the existence with a variety of applications; also user is able to change the skin layout which was provided by their own.

Figure 3 is an application in which user change the software to suit their own smart mobile device interface layout. The background of the interface as shown in the illustration, the user can easily change, but other can't produce desired interface directly or can't share their interface widely. Users can provide some information to developers to change the interface. The software interface will be assigned to developer; by their developed we can use this kind of software widely. Usually the software interface for the common people and persons with disabilities, children, elderly people, such as the classification of people smart mobile devices cannot use the interface provided by smart mobile devices. We will provide the hardware and software interface to the cloud environment. Users can use the platform proposed in this paper, users can easily create the layout of the software interface, and users can share the layout of the interface between the user interface layouts.

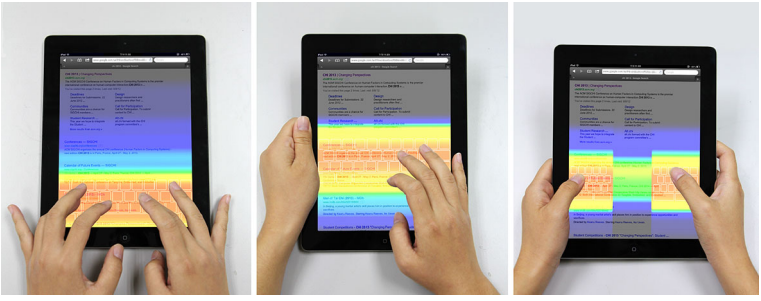


Fig. 2 iGrasp keyboard

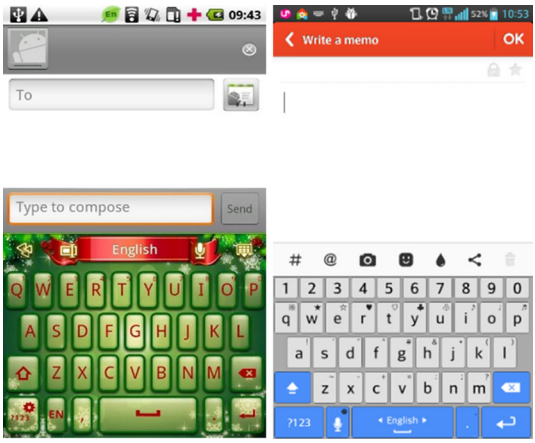


Fig. 3 Software keyboard layout

3 User Customizable Interface for Server Environments

3.1 Customized Interface for Sharing Server System

In this paper, the system shows in Fig. 4 was constructed to provide a customized interface. The cloud system [9, 10] environment including four types of servers, each server has the data management function of MongoDB based. Interface server stores information on the software interface layout, a user or administrator can store or create software. Interface layout can be given to Private, Public authority according to their authorized production. If the user sets the Private the interface layout, thus you cannot access and manufacture by using only their layouts. If Public state is set in the interface layout, anyone can use the interface, because it is public layout. In addition, the user can directly create an interface layout; the other user can create a new interface based on this layout. If the setting is enabled in the properties of the Public interface layout, the layout can create new based on the layout. User use interface creation tools when you publish an interface, the interface is stored on a server therefore it can be used by other users. Interface provides the interface

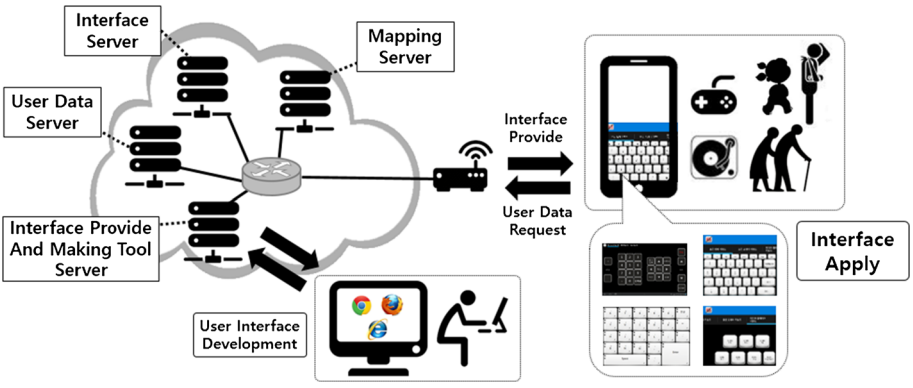


Fig. 4 Cloud system architecture for providing an interface

creation tool server analyzes the user information when a user requests the interface, serves to provide an interface that they need. Interface provides the interface creation tool server may direct a user to connect, if the user is making the request or server provides an interface tools, the interface can store. Interfaces and interface authoring tool server users receive data between cloud servers and can access. User data server has a function for identifying the user, the user's name, ID, Password, E-Mail, etc. The information that is configured to identify the user. If the user requests the interface, the server analyzes the user's information to provide a custom interface for the user, and provides the interface also. Mapping server is a function for matching an interface layout data and the user data. Generally, the data between the server join MongoDB allows to overcome this problem because it is composed of the non-relational data. Mapping server does not have a complete data, and has only information for each user choose which interface for their usage. It has only a data bit interface ID and the user ID information with the mapping server.

3.2 Platform Based on Cloud of Customized Interface

In order to provide a custom interface to the user it must be able to easily create all the interfaces. In this paper, we provide a web-based system platform in the cloud. Platform production systems are cloud PaaS (Platform as a Service) similar to the type you need to install a separate SDK without using a Web browser, anytime, anywhere, and you can use the authoring interface platform. Interface creation tools are based on the development of HTML5 [11, 12]. Figure 5 shows the interface structure of the platform for production systems. Node.js [13] is an event-based asynchronous input and output capabilities of the network in a manner that has the data. In addition, Node.js is compatible with Android-based mobile devices because it uses JavaScript. Node.js provides continuous updates and fast speed because it uses Google JavaScript [14]. Therefore, the Web browser and the operation based on Node.js [15, 16], the user data in the data management server through the Node.js, interface data, and stores the mapping data reads. Web site available to the actual user is in charge of JQuery, and is provided to the user HTML5 form. In addition, JQuery is able to provide Web information quickly in the mobile environment. The user, as shown in Fig. 5 may be provided with an interface creation tool using a smart mobile device or PC. If a user is using the JQuery-based platform, when they store after making the interface, the data stores in a data management server will use the Node.js. When a user requests a data interface, and they produce interface directly, JQuery retrieve data

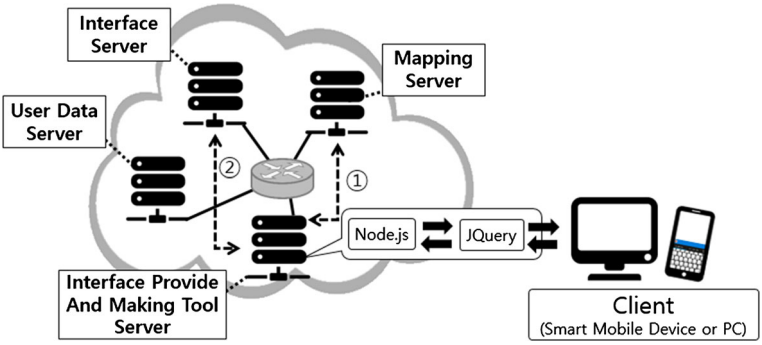


Fig. 5 Platform structure for making the system interface provided

requested by the user, is transmitted and passes the instruction to the instruction from the data management server through the Node.js [17, 18]. The imported data is a file which has been created and will be transmitted to the smart mobile devices.

Figure 6 is a user interface platform which provided with a web browser. The user is using a mouse or a finger to set the interface which is the key, the size, the key position, key values. Interface has a function of the two. First, press the first function is main user interface that uses function keys to type in a letter or document, you can create a document by input the correspond key values. The second function is a function key as a video player, a music player, and provides a key interface for a variety of applications such as games. If you want to next a movie, you can use the designated function keys, such as volume up or volume down. The user can use the interface is made by their conventional; the interface may be made by modifying their desired. The interface provided by the existing smart mobile devices is limited, but the layout of the system in this paper is produced in a variety of layouts can be used by users to fit a smart mobile device environment, cloud environment at any time via the interface layout.

3.3 Providing User Customized Interface in Mobile Environments

Figure 7 shows the process when a user requests the interface. If the user enters an ID and Password using the keyboard interface applications <UserLogin ID 'User_ID', PW 'Password'> a (Fig. 7①) request command will be transmitted in formats. Request command <Request ID 'User_ID', PW 'Password'> and sends this to the Node.js web server via JQuery (Fig. 7②, ③), Node.js using User Data Server to check whether the user is correct or not. Interface provides the user confirmation when the server from the User Data Server, informs the user's identity to Mapping Server (Fig. 7④). Mapping Server will check the interface information by the user based on the username and request to select User Data Server Interface Server (Fig. 7⑤). And Mapping Server sends the search interface layout ID to provide server interface (Fig. 7④). Interface providing server requests the interface layout information to the interface server, using the interface ID (Fig. 7⑥). Node.js information obtained from the interface, using the server, creating a data file in JSON form (Fig. 7⑦). JQuery sent to the user with a JSON file to send crafted JSON file path http//XXX.XXX.XXX.XXX/UserID/xx.JSON 'format (Fig. 7⑧). Usually when you transfer the data to the server data is sent into a string format. But the network between the server and the requesting user have amount of the requested data if you have

Fig. 6 The production platform interface

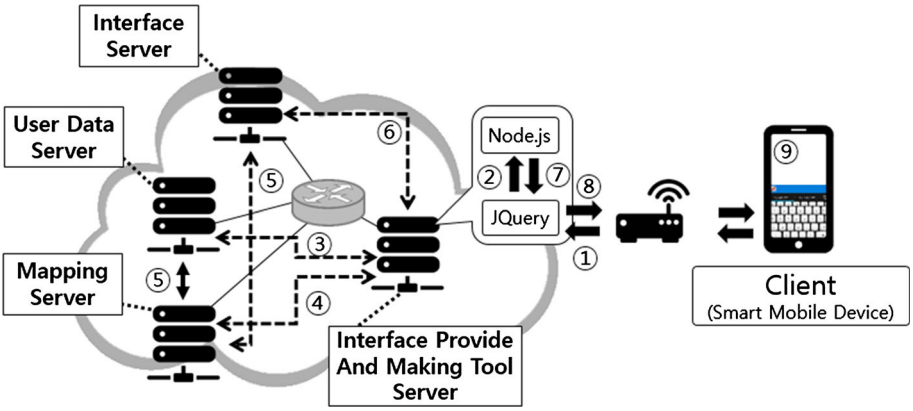


Fig. 7 The method provides a user interface

bad requested data, the corruption might occur. Thus, the user types a string in JSON format rather than the file because it can send and receive the information with no errors using the Http protocol even if we have poor network environment.

In Fig. 8 on two parts of the left side of the figure, user enters username and password before in login process. After the username and password are exist in database then a window that appears allows user to define a user-defined interface. In Fig. 8 on the two parts of the right side of the picture is a screen with the requested interface can be set up for several purposes. If you select the top part of the list interface will be available immediately. Applications allow you to quickly change your preferred interface supports the slide mode where user can easily change the interface. The third illustration in Fig. 8 shows a view to create a document by using a custom interface. The fourth picture in Fig. 8 uses the movie, play, stop, next, movie playback, using volume control. Figure 8 were tested using the Samsung Galaxy S4, the application must be supported by a multi-language display function of the dual system. Many people use the smart mobile devices as a variety of shapes inside the device with different resolution screen. Equation (1) calculates the size of

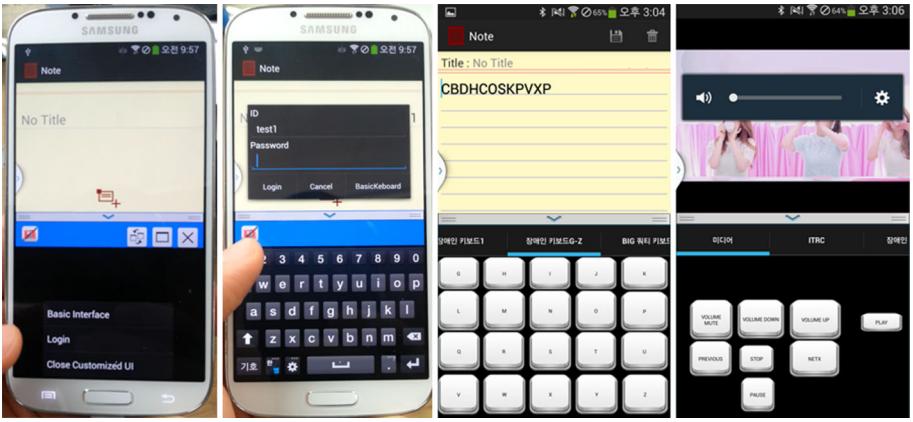


Fig. 8 Provides a custom interface depending on the application environment

the keyboard in the web authoring tools K_{width} is used for calculating a variety application of devices, D_{width} is horizontally mobile device based on the position of Fig. 8, the resolution value of the user mobile device. S_{width} for horizontal resolution of a Web authoring tool, R_{width} means the horizontal size of the key value that applies to your mobile device.

$$\frac{K_{witdh} \times D_{witdh}}{S_{witdh}} = R_{witdh} \tag{1}$$

$$\frac{K_{height} \times D_{height}}{S_{height}} = R_{height} \tag{2}$$

As shown above, the expression (2) seeks the direction of the vertical. For example, Web authoring tools are called the horizontal size of the keyboard and make the size 50 when the Web authoring tools resolution is 1280 × 720. If that resolution of the display language used by the user smart mobile devices is 1920 × 1080, it is calculated by 50 × 1920/1280 with 75 applicable sizes.

3.4 Interface Servers Interlocking Systems for Sharing Information

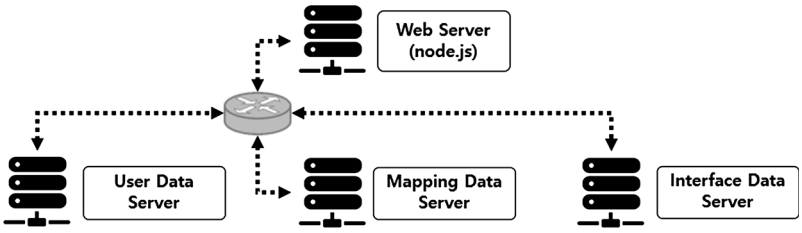
MongoDB stores BSON [19, 20] data in the file. Users can use the Web server when they save the data, depending on the type of data that is stored in the repository in MongoDB. BSON place of storage is stored, when a user makes a request to access data over the web to each of the sub-file in the BSON MongoDB server [21], it processes the data. The data is

Table 1 JSON data retrieval and 'A102' user generated content

```
<UserInterface UserID="A102'>
  <InterfaceID="#0010">
    <Key label="c" Value="0x21" Width="24" High="24" X="0" Y="0" · · · · />
      · · · ·
      · · · ·
    </InterfaceID>
  <InterfaceID="#0012">
    <Key label="d" Value="0x22" Width="24" High="24" X="25" Y="25" · · · ·
/>
      · · · ·
      · · · ·
      · · · ·
</UserInterface>
```

stored in XML format in a similar way, when the user makes a request, the contents of Table 1 the keyboard interface will be archived in JSON file. When user request data via user devices to retrieve the JSON file, it will send it to the user. BSON file is full of data because the Web server and the connection are managed by the web server. BSON files are linked with Web server, because Web server manages the entire data. The Join of each Web servers prevents data redundancy and MongoDB mapping using the management server. Figure 9 is a structure for processing the data of each MongoDB using Node.js. MongoDB internally in Node.js to join each function because the process does not connect the data of each file by controlling the data of BSON MongoDB. When a user login using the mobile device or a PC, it will connecte to MongoDB Node.js UserData Server checks the correct user. If the user is true, Node.js will use the user ID, and read the data from the Mapping Server. Node.js using ID of the user ID of the interface associated with the physical interface and then it accesses to the information of the interface Server for returning the interface layout information. For example, if the user requests A102 request associated with the Interface ID #A102 which uses the information in the Mapping Server 0010, #0012, #0023 and reads the Node.js after that it reads interface ID using the information to the physical layout of the interface server and transmits to the user with a JSON file. Transfer method of strings using a non-http. This is 3G, a variety of network environments, such as WIFI, LET can transfer data securely.

Table 2 provides the user interface an algorithm to use for users with mobile devices if the ID and Password to request data is sent to the server. The user ID and Password of *u*. User Check function to determine whether the user is wrong and correct and then Mapping Data Search function will search for a user by using the interface. *i* using the Interface ID to retrieve the ID of the interface, using the Interface Data Search function retrieves information of the interface. *l* is the interface data, and generates a JSON file using the Data file Create function. *j* denotes the interface based on JSON file requested by the user, and allows the user to get the JSON files using the Data Sending function. If we use the existing MongoDB to share the interface to the user that they can use it to generate interface quickly. We use a conjunction with three MongoDB data servers. Assuming that if the algorithm applies three MongoDB the performance analysis will be better.



User ID	User P.W
A102	*****
B101	*****
•••	•••
F100	*****
Z103	*****

User ID	Interface ID
A102	#0010
	#0012
	#0023
•••	•••
B101	#0010
	#0023

Interface ID	Layout Info
#0010	•••
#0012	•••
•••	•••
#0021	•••
#0023	•••

Fig. 9 Interface data storage structure

Table 2 Interface algorithm providing

```
WHEN u Request the Keyboard Interface Data THEN
  //mongoose.connect('mongodb://localhost/UserData');
  IF User Check (u ) == true THEN
    //mongoose.connect('mongodb://localhost/MappingData');
    i = Mapping Data Search(u )
      WHILE i != NULL THEN
        //mongoose.connect('mongodb://localhost/InterfaceData");
        j = Interface Data Search( i )
      END WHILE
      j = Data File Create ( i )
      IF j != NULL THEN
        Data Sending ( i )
      END IF
    END IF
  END WHEN
```

Performance analysis of queuing model (Queuing Model) [22] are used to design the data communications and network theory to the behavior of the attribute system allows us to do with fast analysis, design, evaluation. The incoming data is being processed in the first place in the queue (First in First Out) storage of data in the form. First, for performance analysis uses an M/M/1 and M/M/C Queueing Model. The first M represents the probability density for the arrival time, the second M refers to the probability density, C is the last one, or the number of the server for the time of service. Queueing Model used in this paper follows above description.

$$\rho = \frac{\lambda}{\mu} < 1 \tag{3}$$

$$L = \sum_{i=1}^{\infty} np(i) = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda} \tag{4}$$

$$\text{Little's formula } (L = \lambda\omega) \quad W = \frac{L}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu - \lambda} \tag{5}$$

$$Lq = \sum_{i=1}^{\infty} (i-1)P(i) = \frac{\rho^2}{1-\rho} = \frac{\lambda^2}{\mu(\mu - \lambda)} \tag{6}$$

$$\text{Little's formula } (\mathrm{L} = \lambda\omega) \quad W_q = \frac{L_q}{\lambda} = \frac{\rho^2}{\lambda(1-\rho)} = \frac{\rho}{\mu - \lambda} \tag{7}$$

$$W = \frac{L}{\lambda} \tag{8}$$

$$W = \frac{1}{\mu} + \left(\frac{r^0}{c!(c\mu)(1-\rho)^2} \right) p(i) \tag{9}$$

λ is the data, ρ is the arrival rate, μ is the service rate, the length of the average queue L_q, W_q the average wait time, P (i) refers to a condition when the probability of i (number of users). L is the average number waiting in the system, W the average time spent in the system (service hours), r means the average packet inside the router. Queueing Standard Model using the Eqs. 1–7, Eq. (8), derive a 9 M/M/1 and M/M/C Queueing Model performs performance analysis. Figure 10 shows the design of a structure in accordance with the server structure MongoDB Queueing Model.

In Fig. 10, λ is input data, and μ refers to a service rate. Figure 10a MongoDB servers apply M/M/1 Queueing Model refers to when they were one. Using this model with the server is a server, and 1 day compared to the 3 days. If you apply the algorithm in the Server 3 as compared to when they did not apply to the structure of the algorithm is not applied M/M/1 Queueing Model as shown in Fig. 10a. When user don't applying the algorithm, each of the Queue Server and Server 3 are each MongoDB [23] because it does not share data with each other in non-relational. While using the format M/M/1, each of the servers look like M/M/3 non-relational type because it is the internal server data is often overlap. Figure 10b is made of three servers, since the algorithm is composed of a relational application to share data among each other. Therefore, the data is shared, rather than M/M/3, by applying a performance analysis of Queueing Model. Depending on the configuration of each Server Queueing Model in the same conditions is applied, this paper proves the efficiency of the proposed algorithm, resulting in performance analysis and test server configuration and shows results.

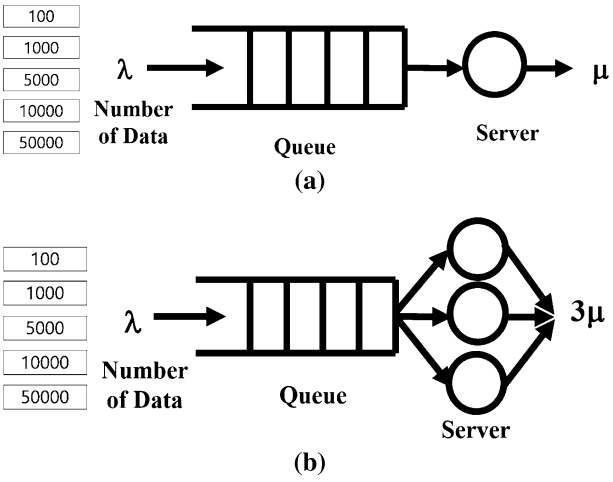


Fig. 10 **a** M/M/1 Queueing model, when Server has one storage, **b** M/M/c Queueing model, when Server has 3 storage spaces which are applied with algorithm

4 Test Results and Performance Analysis

Figure 11 shows the actual data, 100, 1000, 5000, 10,000, 50,000, using test data. M/M/1 is one of the data servers when using MongoDB, and even allow for redundant data storage space as can be seen because the rate increases the width of the one constant remains. M/M/3, is the case of using the three MongoDB data server, and if you don't applied the algorithm in one of the data servers because of duplicate data, it takes a lot more time than that. And Fig. 3 data server may increase due to the data redundancy in the data, there are three time because the search server in sequence takes the most. The last three data server application of the algorithm generates a reduced amount of data if there is a duplicate data because the data is shared with each other. For example, the whole input data is 100, if the duplicated data is 30 days, the total number of data with real data server is 70 carbon atoms. Thus, a typical server can handle more data faster; memory can be efficiently used. We tested the actual MongoDB data rate based on the results of the performance analysis. Figure 12 shows the actual data retrieval speed test of MongoDB server performance analysis and as a result was able to get a similar result.

Figure 12 shows the actual data, 100, 1000, 5000, 10,000, 50,000, using test data. When you have a data server for MongoDB, is constantly increasing the processing speed. However, the algorithm is not applied to the three MongoDB data servers, it duplicate data input, an increase in the number of data than the data entered. In addition, when users request data, the server must search in order to search for duplicate data, and thereby can be time-consuming. Algorithm for three MongoDB data servers because the duplication of data in the relational case can be reduced, due to the number of total data. Therefore, when you enter data in the other servers in the same environment and able to quickly process data differently, proves the experiments. In addition, the server is not wasted because there is memory to process the redundant data, the data input can quickly process the data even if more.

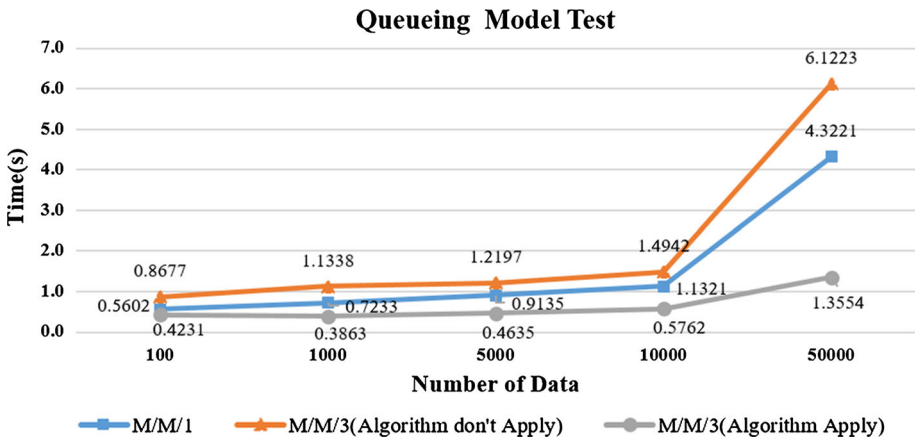


Fig. 11 Performance test with Queueing model applied

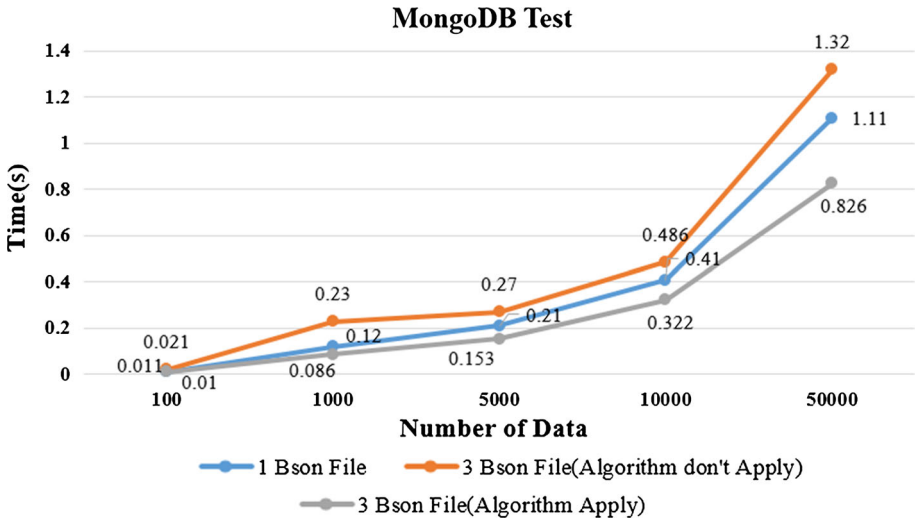


Fig. 12 MongoDB reaction speed test by number of Bson file

5 Test Results and Performance Analysis

In this paper, the existing hardware interfaces and software interfaces provided by the cloud environment as a personalized interface. We allow users to be provided with the desired interface with the MongoDB data server, if they don't have the interface, they can make their own desired interface. Also user can create the user interface and share. Sharing interface by using a web browser and user can easily be provided to other with a smart device. We were used MongoDB data server to share the interface to the manufacturer or user generated efficiently, and to solve the problem of duplicate existing MongoDB. In this paper, we use the three data servers MongoDB which data was managed, and applying an algorithm to create a non-relational data in a relational server. In addition, we confirmed that user can provide rapid analysis of data on the performance of the test. By maintaining the data in a relational MongoDB that it is able to provide user interface with efficiently, and variety of interfaces through the cloud where user can access anywhere. Future research plans to study is sharing quickly data between users on the mobile cloud, not cloud servers.

Acknowledgments This research was partially supported by the IT R&D program of MSIP (Ministry of Science, ICT and Future Planning) / NIPA (National IT Industry Promotion Agency) [12221-14-1001, Next Generation Network Computing Platform Testbed] and also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2013R1A1A2013740). Furthermore, this work was also supported by 'The Cross-Ministry Giga KOREA Project' grant from the Ministry of Science, ICT and Future Planning, Korea.

References

1. Seth, J. K. (2013). Chandra S (2013) A novel design to increase trust in cloud IaaS model. *International Journal of Computer Science Issues (IJCSI), 10*(4), 329–336.

2. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., et al. (2010). A view of cloud computing. *Communications of the ACM, 53*(4), 50–58.
3. Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., & Epema, D. (2010). *A performance analysis of EC2 cloud computing services for scientific computing, Cloud Computing* (pp. 115–131). Heidelberg: Springer.
4. Sharma, Yashvardhan, Verma, Saurabh, & Kumar, Sumit. (2013). A context-based performance enhancement algorithm for columnar storage in MapReduce with Hive. *International Journal of Cloud Applications and Computing (IJCAC), 3*(4), 38–50.
5. Dede, E., Govindaraju, M., Gunter, D., Canon, RS & Ramakrishnan, L. (2013). Performance evaluation of a mongodb and hadoop platform for scientific data analysis. In *Proceedings of the 4th ACM workshop on Scientific cloud computing*, (pp.13–20). ACM, .
6. Liu, Y., & Whitt, W. (2012). A many-server fluid limit for the Gt/GI/st + GI queueing model experiencing periods of overloading. *Operations Research Letters, 40*(5), 307–312.
7. Oliveira, J., Guerreiro, T., Nicolau, H., Jorge, J., & Goncalves, D. (2011). BrailleType: Unleashing braille over touch screen mobile phones. In *Human-Computer Interaction? INTERACT* (vol. 6946, pp. 100–107). Berlin Heidelberg: Springer.
8. Cheng, L-P., Liang, H-S., Wu C-Y., & Chen, M. Y. (2013). iGrasp: Grasp-based adaptive keyboard for mobile devices. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 3037–3046), ACM.
9. Wettinger, J., Andrikopoulos, V., Strauch S., & Leymann, F. (2013). Enabling dynamic deployment of cloud applications using a modular and extensible PaaS environment. In *Cloud computing (CLOUD), 2013 IEEE sixth international conference on. IEEE* (pp. 478–485).
10. Colombo-Mendoza, L. O., Alor-Hernandez, G., Rodriguez-gonzalez, A. & Valencia-garcia, R. (2014). MobiCloUP!: A PaaS for cloud services-based mobile applications. *Automated Software Engineering, 21*(3), 391–437.
11. Anthes, Gary. (2012). HTML5 leads a web revolution. *Communications of the ACM, 55*(7), 16–17.
12. Akhawe, D., et al. (2013). Data-confined html5 applications. In *Computer security? ESORICS 2013* (vol. 8134, pp. 736–754). Heidelberg: Springer.
13. Zhao, S. M., Xia, X. L., & Le, J. J. (2013). A real-time web application solution based on Node.js and WebSocket. *Advanced Materials Research, 816*, 1111–1115.
14. Zhu, Y. (2012). Introducing google chart tools and google maps API in data visualization courses. *IEEE Computer Graphics and Applications, 6*, 6–9.
15. Ivan, C., & Popa, R. (2014). Cloud based cross platform mobile applications building and integrating cloud services with mobile client applications. *Advances in Computer Science: An International Journal, 3*(2), 69–77.
16. Systa, K., Mikkonen, T. & Jarvenpaa, L. (2014). HTML5 agents: Mobile agents for the web. Web. In *Information systems and technologies* (vol. 189, pp. 53–67). Heidelberg: Springer .
17. Holzinger, A., Treitler, P. & Slany, W. (2012). Making apps useable on multiple different mobile platforms: On interoperability for business application development on smartphones. In *Multidisciplinary research and practice for information systems* (Vol. 7465, pp. 176–189). Heidelberg: Springer.
18. Workman, R., & Christ, T. (2013). Mobile phone course applications using jQuery mobile. *Society for Information Technology & Teacher Education International Conference, 2013*(1), 3814–3817.
19. Gyrodi, C., Gyrodi, R., Pecherle, G., Olah, A. (2015). A comparative study: MongoDB vs. MySQL. In *Engineering of modern electric systems (EMES), 2015 13th international conference on. IEEE* (pp. 1–6).
20. Chanda, S., & Foggon, D. (2013). Introducing non-relational databases. In *Beginning ASP. NET 4.5 Databases* (pp. 49–59). Apress .
21. Tian, X., Huang, B., & Wu, M. (2014). A transparent middleware for encrypting data in MongoDB. In *Electronics, computer and applications, 2014 IEEE Workshop on. IEEE* (pp. 906–909).
22. Yang, K.-K., Low, J. M. W., & Cayirli, T. (2014). Modeling queues with simulation versus M/M/C models. *Journal of Service Science Research, 6*(1), 173–192.
23. Boicea, A., Radulescu, F., & Agapin, L. I. (2012). MongoDB vs oracle–database comparison. In *2012 Third International Conference on Emerging Intelligent Data and Web Technologies. IEEE.*

Sanghyun Park received the B.S. Degree in Computer and Information from the University of Korea Nazarene in 2010, and the M.S. degree in School of Electronics and Computer Engineering, Chonnam National University, South Korea. He worked as an engineer in System Development Team of Media Flow Company from 2010 to 2012. He is now studying Ph.D. Degree in School of Electronics and Computer Engineering, Chonnam National University. His research interests are Interactive Media, Systems Development, Embedded systems, Digital Media and Cloud computing.

Hoyong Ryu received the B.S., M.S., and Ph.D. degrees in electronic communication engineering from the Kwangwoon University, Seoul, Korea. In 1993, 1995 and 1999 respectively. From January 1999, he is Principal Member of Researcher with eth Electronics and Telecommunications Research Institute, he is engaged in research on the project manager of Network Software Platform Research Section. His research interests include Internet and routing mobile IP and security.

Kuinam J. Kim is a professor of Information Security Department in the Kyonggi University, Korea. He received his Ph.D. and MS in Industrial Engineering from Colorado State University in 1994 and 1993. His B.S in Mathematics from the University of Kansas. He is Executive General Chair of the Institute of Creative and Advanced Technology, Science, and Engineering. His research interests include cloud computing, wireless and mobile computing, digital forensics, video surveillance, and information security. He is a senior member of IEEE.

Jinsul Kim received the B.S. Degree in computer science from University of Utah, Salt Lake City, Utah, USA, in 2001, and the M.S. and Ph.D. degrees in digital media engineering, department of information and communications from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2005 and 2008. He worked as a researcher in IPTV Infrastructure Technology Research Laboratory, Broadcasting/Telecommunications Convergence Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea from 2005 to 2008. He worked as a professor in Korea Nazarene University, Chon-an, Korea from 2009 to 2011. Currently, he is a professor in Chonnam National University, Gwangju, Korea. He has been invited reviewer for IEEE Trans. Multimedia since 2008. He was General Chair of IWICT2013/2014/2015, ICITCS2014, ICISA2015, ICMWT2015 and ICISS2015. His research interests include QoS/QoE, Measurement/Management, IPTV, Mobile IPTV, Smart TV, Multimedia Communication and Smart Space/Works.