International Conference on Informatics for Development 2011 (ICID 2011)
http://icid.uin-suka.ac.id/2011/
Printed ISSN:2088-5520
e-ISSN: 2089-7820

Evaluation of Protocol Buffers as Data Serialization Format for Microblogging Communication

Canggih Puspo Wibowo
Yogyakarta, Indonesia
canggih@jteti.gadjahmada.edu

Abstract—**Microblogging is getting popular nowadays in web 2.0. Twitter, which is an example of application used for microblogging, has many users online at the same time and also exchanges a lot of data over the internet every seconds. This paper discusses the evaluation of Protocol Buffers as an alternative of JSON to be used as data serialization format in microblogging data communication, which in this case is used on Twitter. The results show that the implemented Protocol Buffers is a better alternative for data serialization format than JSON. In terms of performance, it can be observed that Protocol Buffers has better performance than JSON. The resulting data serialization has smaller file size, thus resulting in faster data exchange compared to JSON, so protocol buffers will be a better alternative to be used for data serialization.**

Keywords-microblogging; twitter; serialization; protocol buffers

I. INTRODUCTION

Nowadays, microblogging is very popular among internet users all around the globe. Microblogging provide easiness to one of the most basic human need, which is communication. People like or need to share something with the others. As for communication, the object to be shared is information. Thus, currently millions of internet users interact with each other by microblogging, using application such as Twitter. Hence, every second, there are huge numbers of data exchanges on Twitter. This issue will disrupt whole data exchanges on the Internet.

In Twitter application, a format of data serialization is used to communicate data from server to client and vice versa. Before the data is sent, either from client to server or vice versa, it must be modified to a certain format according to serialization technique being used. Types of data serialization format commonly used in the Internet are XML and JSON. Twitter itself is actually supporting the use of both formats, but JSON is more frequently used than XML, hence it will be discussed in this paper.

Furthermore, plain-text type format such as XML and JSON have relatively larger size, thus it will affect the data exchange speed. Nowadays, internet has been widely used around the globe and millions of users utilize internet at the same time, while bandwidth and internet load becomes limited, speed becomes an important factor, considering that every users might require fast data communication and reliability. Hence, data size needs to be minimized while retaining the original information.

To tackle this problem, Google has released a new data serialization format which is binary based for data communication in the internet, namely Protocol Buffers (protobuf). Protobuf has several advantages over plain-text format (which has been utilized earlier for data serialization). Some of the advantages include more rapid, efficient data exchange, and it is programmable [5]. Protobuf also has been used in Google's internal systems.

II. LITERATURE REVIEW

A. Microblogging

Microblog was derived from the term weblog, which has been developed progressively in the era of Web 2.0. Weblog itself is frequently updated website, owned by only one person, and contains data that is displayed in chronological order [1]. The difference between those two is the amount of data that can be updated by the user, in this case Microblog allows only 140 characters each update [2]. Currently, some social networking applications such as Twitter have microblog characteristics. Thus the concept of microblog is a combination of weblogs and instant messaging.

Other examples of commonly used microblog besides Twitter are Plurk, Jaiku, and Pownce. Each of them has different concepts, features, and perhaps also different type of users. Features provided by microblog in general can be divided into three terms, namely information sharing, information seeking, and friendship-wide relationship [2]. Recently, microblog development has become rapidly increased in line with the rapid development of mobile technology that allows users to access social networking sites and microblog easier. The development of mobile technology also makes people to update status more often [3].

Printed ISSN:2088-5520
e-ISSN: 2089-7820

B. *Serialization*

Serialization is a process of converting data into a stream of bits to be sent through either a wired or wireless medium or stored in a storage medium [4]. The opposite process, which is called deserialization, is the process of converting a stream of bits back to the original data. The process of serialization-deserialization is used on the whole internet.

There are two kinds of serialization, which are the serialization into a human-readable format (text) and non-human-readable format (binary) [4]. Example for serialization to text are XML and JSON format, whereas for the binary are Protocol Buffers, BSON, Thrift, and others. The main difference between both of them is that the serialization format of the text is can be understood directly by human (even in a serial format), while for the binary is not readable.

C. *Protocol Buffers*

Protocol Buffers is a binary data format used for data serialization. Protobuf was developed by Google to handle internal problems in terms of the protocol request / response on the server that experiences development version. Previously, the data format being used can only support one version, so when the protocol progresses, the entire format of the data must be adjusted [5]. This is undesirable for systems development. On the other hand, Protobuf will ignore new data fields that are not supported. In other words, at one time, protobuf format can be used both by the old system and new system. During its development, very precise protobuf is used on systems that require large amounts of data exchange within a short time and limited bandwidth [6].

Protobuf uses files that contain the definition of the data structure to be sent in the form of name-value pairs. The definition is commonly called *protobuf messages definition* or *protobuf messages scheme.* The definition is stored in .proto format. A simple example of a protobuf message definition which contains information about data structures is displayed in Fig. 1.

```
message Person {
        required string name = 1;
        required int32 id = 2;
        optional string email = 3;
        enum PhoneType {
                MOBILE = 0;
                HOME = 1;
                WORK = 2;
        }
        message PhoneNumber {
                required string number = 1;
                optional PhoneType type = 2;
        }
        repeated PhoneNumber phone = 4;
}
```

Figure 1. Sample of protobuf message definition

Then, the .proto file that contains protobuf message definition is compiled to generate a data access class according to the programming language used. Currently, the programming languages supported by the official compiler from Google are only Java, C + +, and Python [5]. Despite that, many parties make their own compilers for other programming languages. Through the data access class that has been generated by the compiler, the data can be processed easily to generate protobuf binary data.

Protobuf is generally similar with XML and JSON in terms of its use in data serialization, except that it is simpler, smaller, and faster than XML and JSON. There are many systems that use protobuf now, such as the web services system which utilizes protobuf as their new data serialization format, replacing the role of XML that has been used [7][8]. XML was considered too slow when used in large scale systems [6]. Some advantages of protobuf when compared to XML are [8]:

- Protobuf uses data access class that allows data to be accessed easily through program.
- Protobuf message definition can be changed without affecting the existing system
- It does not require meta-data, so that the data sent is smaller in terms of size.

III. TWITTER DATA COMMUNICATION

In this paper, the data structures used in Twitter is the data sent by the server to the client in response to status updates via Twitter web client. The response from server contains information about status and appropriate user data in the form of JSON. This paper observes the response data because Twitter users update their status very often compared to other activities using Twitter. The JSON data used in this paper will be compared with protobuf data.

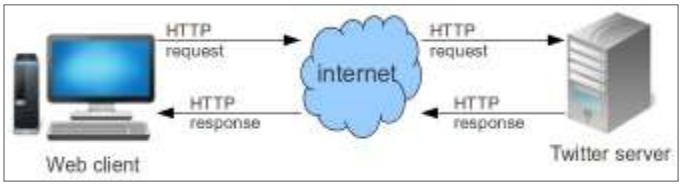

Figure 2. Twitter communication scheme

As seen in Fig. 2, web client performs an HTTP request to a server by utilizing the twitter REST API provided [9]. Request sent by client is then responded by the server by sending the response data in JSON format. Response data is broadly divided into two parts, namely status information and user information [9]. Both data are used to display the status on the web client. Only some information sent by the server is displayed to user. Some of it is stored and used for other

Printed ISSN:2088-5520
e-ISSN: 2089-7820

purposes, such as user data which is used to update status in the future.

IV. PROTOBUF MESSAGE DEFINITION USED

On the systems which use protobuf as data serialization format, protobuf message definition should be utilized to give a schematic definition of data structures that will be sent. Protobuf message definition used in this paper is adjusted based on response data sent by the server when the client sends Twitter status update. From this information, protobuf message

```
message update{
        optional string in_reply_to_status_id_str = 1;
        optional string id_str = 2;
        optional bool truncated = 3;
        optional string created_at = 4;
        optional string in_reply_to_user_id_str = 5;
        optional string contributors = 6;
        repeated usermessage user = 7;
        optional bool favorited = 8;
        . . . . .
        optional string text = 19;
}

message usermessage{
        optional bool is_translator = 1;
        optional string profile_background_image_url_https = 2;
        optional bool protected = 3;
        optional string id_str = 4;
        optional bool follow_request_sent = 5;
        . . . . .
        optional string profile_link_color = 38;
}
```

definition is built using code shown in Fig. 3.

Figure 3. Protobuf messages definition used

Protobuf message definition consists of two parts, the message *update* and message *usermessage*. Message *update* contains data about the Twitter update status which has 19 fields, whereas message *usermessage* contains 38 fields data about the user who sends the status. In the message *update,* tag number 7 is used to define a field that use *usermessage* as data type. This means basically, message *usermessage* is nested inside message *updates*. Because protobuf message definition is created based on data from Twitter response, it is difficult to know whether particular field is required or not, so all of the fields are defined with optional rule to make the implementation easier. Optional rule means the field can have zero or one value. It will be undesirable if a field uses required rule but there is data which has no value on that particular field. So, it is clear that using optional rule is recommended by protobuf developers because of the safety implementation [5]. Among all fields in messages on Fig. 3, there is only one field which use repeated rule, on tag number 7. The field implements this rule because it is the provisions of protobuf in defining a nested message.

In protobuf message definition, tag number given on each field is used to determine the sequence of field on serialization. This sequence was adapted from the sequence of data delivered in JSON file which is sent by Twitter server.

V. TESTING

The test is carried out using a PC with the following hardware: Processor Intel Atom N280 1.66 GHz, 1 GB of RAM, and Ubuntu Operating System. The programming language used is Java with OpenJDK 1.6, for both JSON and protobuf. In the test, length of time required for file serialization process between protobuf and JSON is compared. And for the second test, the comparison is done based on the resulting file size. The serialization file size will influence heavily on the internet network. The bigger file size means greater bandwidth needed for data exchange.

Some adjustments are made for the testing, both in the protobuf and JSON files creating process. From data obtained from the Twitter, both the field ID and user ID contain integer values in large size. For JSON file, Java provides big integer data type that can accommodate the large integer value. Whereas protobuf cannot accommodate big integer value as it does not have data type that contains more than 64 bits integer. To tackle this problem, any big integer values in Java is converted to bytestring data before being stored in protobuf as bytes.

In protobuf testing, Java code generation provided by Google is implemented to process the protobuf data. Serialization time on protobuf is measured by calculating the time differences between two timestamps that are put before and after the process. The time required to make serialization files is in the terms of micro or nanoseconds while the timestamp used (which is provided in Java) has the accuracy of milliseconds, therefore looping file creation is done million times and then the average time is calculated to obtain more accurate result.

Afterward, JSON serialization file is done using JSONObject class support. This class is provided in the java library. The process of making JSON file using this class is done by entering data into value pairs object then serialize it in string form.

The data included in each file is taken from the results of random twitter status updates. Currently, the data is not the original data obtained directly from Twitter, instead it is artificial data with the same content as the original.

VI. RESULTS

Result of the first test is used to calculate the time required to make protobuf and JSON serialization files. The result is shown in Fig. 4.

Printed ISSN:2088-5520
e-ISSN: 2089-7820

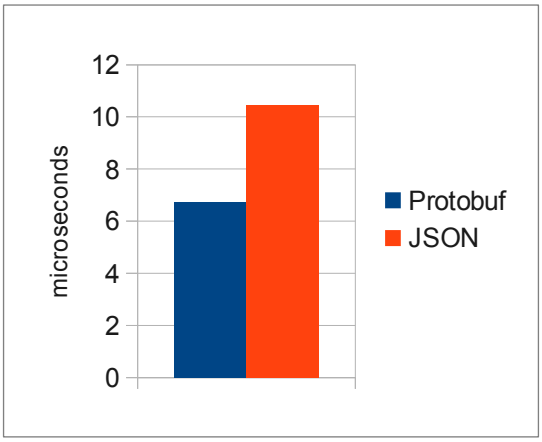

Figure 4. Required time for serialization

The required time for serialization in protobuf is smaller than JSON according to Fig. 4. Even though the time difference is not relatively huge, but it will affect significantly when dealing with large amount of data. The test result varies depending on the applied technique or tool. In this case, the test is limited to one specific tool for each data serialization format, the use of java code generation in protobuf and JSONObject class in JSON. The algorithm used on each technique is not discussed in this paper, so the test result emphasizes on those techniques, thus the result may differ if other techniques or tools were utilized. Fig.5 below displays the resulting serialization file size.

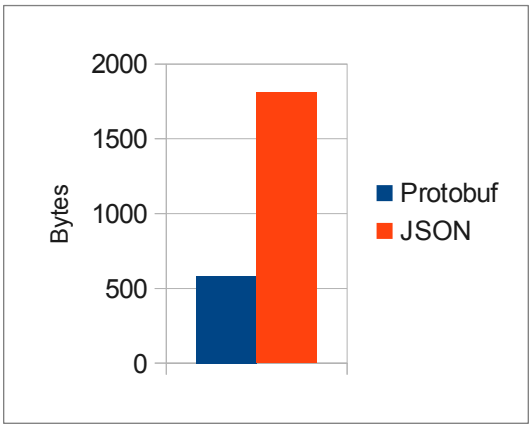

Figure 5. Result of serialization file size

From the Fig. 5, it can be seen that the resulting protobuf file size is approximately three times smaller than JSON. Serialization file size greatly affects the performance of data communication because file size will be proportional to the required time to transmit data from client to server and vice versa. In this case the data is sent from Twitter server to client. So, protobuf has better performance than JSON in terms of speed, time, and size.

VII. Conclusions and Future Works

This paper discussed the use of Protocol Buffers as a data serialization format in Twitter as an alternative for JSON. The use of binary data format such as protobuf enables the data size becomes smaller and faster serialization speed. This affects the speed of data transmission from server to client and vice versa so that communication via internet becomes more efficient. However, in this paper, only a part of Twitter data communications is tested, which is the server response on status update. Binary data formats such as protobuf is shown to have better performance in terms of serialization of data, especially in terms of serialization speed and file size. Therefore, protobuf can be used as an alternative to replace the JSON for Twitter data communications and other microblogging applications.

For future studies, more comprehensive testing is required for particular data and methods. One example is to directly convert Twitter communication data into protobuf and then observe the performance.

References

[1] M. Ebner and M. Schiefner, “Microbloging : more than fun?”, Proc. IADIS Mobile Learning Conference 2008.

[2] M. Ebner, “Introducing Live Microblogging: How Single Presentations Can Be Enhanced by the Mass”, in Journal of Research in Innovative Teaching March 2009, pp. 91-100

[3] S. Fox, K. Zickuhr, and A. Smith, “Twitter and Status Updating Fall 2009”, Pew Internet Project, October 21 2009.

[4] M. Cline, “Serialization and Unserialization”, http://www.parashift.com/c++-faq-lite/serialization.html, accessed on May 25 2011.

[5] Google, http://code.google.com/apis/protocolbuffers/docs/overview.html accessed on May 25 2011

[6] T. O'Brien, “The Common Java Cookbook”, Chapter 14: Protocol Buffers, Discursive.

[7] C.P. Wibowo, L.E. Nugroho, and B.S. Hantono, “Implementasi Protocol Buffers pada aplikasi weblog client dan server”, Proc. Conf. of Information Technology and Electrical Engineering 2011.

[8] M. Oreskovic, “Evaluation of Google Protocol Buffers as a potential data serialization technique for wide use in bioinformatics”, unpublished.

[9] Twitter, “REST API Resources”, https://dev.twitter.com/docs/api, accessed on May 26 2011.