Fast Logging in Time Series for a Computer Security Incident Response

Motoyuki OHMORI†

†Center for Information Infrastructure and Multimedia, Tottori University

鳥取大学 Tottori University

Abstract

- Time Series Database (TSDB) dedicated for a computer security incident
- Lower storage and faster search
- Performance improvement adopting logging message normalization
- Lock-free clustering support for scalability

Background

In order to avoid data breach, it is important to quickly and accurately identify and confine a suspicious host when a computer security incident happens. When an external organization alerts a suspicious host, an IP address of the host is given. In order to identify the suspicious host, ones may then search for the IP address in related logs. Searching time of logs is then important to shorten a delay to identify a host. To this end, there have been already several logging systems such as fluentd, kibana, and splunk that are based upon text messages of syslog.

These existing logging systems are, however, not so efficient. For example, we have experienced that fluentd caused high CPU usages and lost log messages. In addition, these existing logging systems are not dedicated for a computer security incident. They, therefore, need more time to search for required logging messages for an incident.

Research Goals

Research goal is implementing a fast logging database dedicated for a computer security incident that has:

1. ***Scalable logging database that requires smaller storages.***
2. ***Fast search especially for recent logging messages***.

Basic Ideas of the Fast Logging Database

1. *Binary Based Key Value Store*

– **Low performance of existing logging systems based on text message**

Existing logging systems are basically based upon text messages while logging messages of network or security equipment usually are in pre-defined text format. The fast logging database then stores binary values only in a record in a table, and text messages are indexed in another table.

2. *Time Series Database (TSDB)*

– **Each record always has a timestamp**

Since each logging message must have a timestamp, the fast logging database always stores the timestamp as a primary key. All records are basically stored in ascending order of timestamps. Some records are, however, not strictly in ascending order for lock-free operation described later.

3. *Fixed Record Length*

– **Length of all records in a table is always same**

In order to improve search performance, the fast logging database has the same record length for a table as same as recent Relational Database Management System (RDBMS).

4. *Timestamp Index*

– **A location of a record at a timestamp is indexed**

Regarding searching a record, a timestamp is usually specified for a computer security incident. In order to improve searching speed, a location of a record at a timestamp is indexed. Since the number of logging messages may depend upon daytime or night, timestamp index improves searching a record of a specified timestamp.

5. *Logging Message Normalization*

– **Logging message format are automatically normalized:**

Because a logging message is usually output using ***printf*** functions, the fast logging database indexes a message format. A Logging message is then stored as a tuple of a message format index and variable values, i.e., variable arguments of ***printf***.

6. *Lock-Free Clustering Support*

– **Lock free insertion and lookup**

The fast logging database does not strictly consider an order of a logging message. Timestamps in records are, therefore, not always in ascending order. This nature may delay to finish all search in order to make sure that the all log messages in specified timestamp in search are examined. This nature, however, makes insertion and lookup operations *lock-free*. Lock-free clustering can be then archived.

7. *Recent in Memory and Old in Disk*

– **Recently added records are in memory, and older records in disk**

When a computer security incident happens and quick response is necessary, recent logging messages are searched in most cases. Older messages are not required to be fast to be searched because its search itself is enough delayed already. The fast logging database then always keeps recent logging messages in memory as much as possible, and write the messages to a disk if possible or all memory is consumed. Even after older messages are searched once, these older messages are not in memory in order to prioritize quick response for a recent computer security incident.

Overview of the Fast Logging Database

Figure 1. depicts overview of the proposed fast logging database. The fast logging database consists of multiple database servers. All multiple database severs has the same IP address for IP anycasting. Network equipment or other servers (e.g., Web server, mail server and so on) send logging messages to the fast logging database using syslog protocol. The logging messages are then stored into one of database servers.

When one, say a CSIRT member, searches for logging messages, one sends a search request to one of database servers. The database server receiving a request forward the request to the other database servers. All database severs then sends responses back to one who sends a search request.

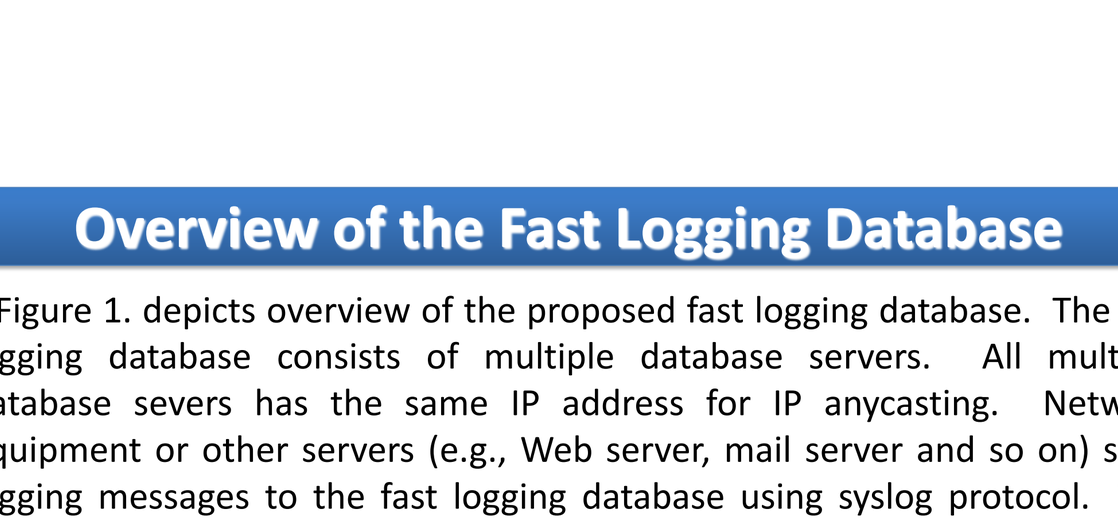

Figure 1. overview of the fast logging database