Scalable NoSQL Provenance Logging

James R. Tapsell

Information Security Group, Smart Card and IoT Security Center

The Smart Card and Internet of Things Security Centre

Objectives

The project has the following objectives:

- Recording provenance event records from NoSQL Databases in a scalable way
- Offer a way to record these provenance event records for future access
- Store the entire event, so that at a later stage full analysis can be performed

Introduction

This project works to log audit events from MongoDB databases, without using the enterprise edition, and also writing the provenance event records out over *STDOUT* as json records, allowing then to be easily stored to files, or streamed over the network.

This technique allows multiple servers to be stood up, and the provenance event records from all of them to be streamed to one centeral log recorder.

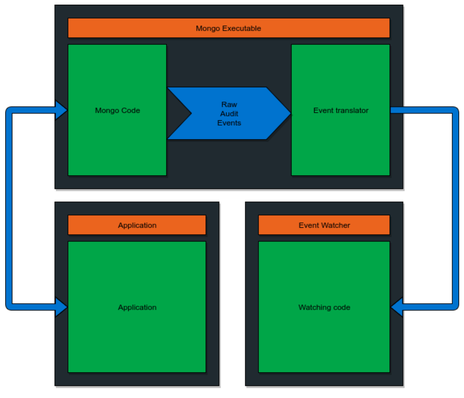

Figure 1: System diagram

Elements

The system is made of the following elements

- Application - An application causing database events to occur
- Mongo code - The MongoDB NoSQL database code
- Event translator - Converts in-memory MongoDB audit events into Provenance Event Records
- Watching code - Uses the streamed events

Important Result

MongoDB events can easily be recorded and streamed over the network to a waiting server, or recorded locally to the server, and then collected at regular intervals.

Event records

Audit events were created with the following structure

- *root*
  - *client*
    - **id** - The client ID number
    - **isSysem** - If the command occurs without a remote port
  - **eventType** - The type of the event
  - *eventData* - The raw event data

They were stored in this way to keep events similar, preventing namespace collisions and making it easier to quickly parse them.

The first revision used the BSONObj toString method provided by MongoDB, but this caused major slowdown as parsing it reduced the speed at which events could be parsed.

A mix of a custom JSONiser and the BSONObj jsonString was used to create RFC7159[1] compliant JSON, which matches the schema provided on http://www.json.org/[2]

Results

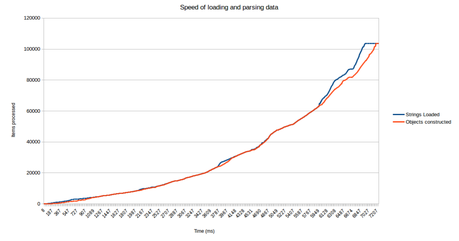

Figure 2: Speed comparison

As can be seen, once standards complieant JSON was used, the speed of parsing the events for use in a second system is almost as fast as the speed of reading, tests were ran on a laptop with the following specs:

- CPU : i7-6500U (4 cores, 1 thread per core)
- RAM : 12GB
- Disk : SSD
- OS : Ubuntu 16.04.3

Conclusion

It is quite possible to log provenance events, convert them to provenance event records and stream them out over the network, with minimal effect on the speed of the database that the logger runs on.

Example Event Record

```
{
  "client":{
    "id": 1,
    "isSystem": false
  },
  "eventType": "logCreateUser",
  "eventData": {
    "user": {
      "username":"username",
      "full": "username@admin",
      "db": "admin"
    },
    "customData": null,
    "roles":[
      "userAdminAnyDatabase@admin",
      "dbAdminAnyDatabase@admin",
      "readWriteAnyDatabase@admin"
    ]
  }
}
```

References

[1] Google Inc for IETF T. Bray, Ed. Rfc 7159. *https://tools.ietf.org/html/rfc7159*. Accessed: 2017-08-28.

[2] The json.org website. *http://www.json.org/*. Accessed: 2017-08-28.

Contact Information

- Web: www.jrtapsell.co.uk/urop-2017-bc
- Email: James.Tapsell.2015@rhul.ac.uk