Report 3: Loggy

Alexandru Matcov

September 25, 2024

1 Introduction

This seminar covers the implementation of a logger and the use of the logical time which orders random log events received from a pool of workers. Two solutions are implemented and covered in this seminar: Lamport time stamps and Vector Clocks.

2 Main problems and solutions

The following section discusses 3 cases represented in this assignment. These are later compared between each other to evaluate performance and reliability of each.

2.1 Basic Random Implementation

The initial implementation has a rather basic solution. It is very dumb: it prints the log events from each worker once immediately it is received by the logger. Thus, it is a FIFO order of the messages. The only holdback that prevents the messages to be sent to the logger immediately is the jitter time. The jitter value introduces random delay between the sending of a message and the sending of a log entry. This makes the *send* messages to be printed in the log before the *received*.

2.2 Lamport Time Stamps

Implementing the Lamport time stamps makes the printing of the log events from the workers a bit more clear and somewhat more ordered. The principle of the method is to have a simple counter for the messages that are being sent and received. Thus, every time a message is *sent* or *received* the counter increments by one. However, the implementation of just the Lamport time stamps is not enough as the log events would still be printed out of order. The important element in the implementation is a **holdback queue**. This guarantees the order of the log events in an ascending order. So, the queue keeps track that message with a smaller time stamp won't be printed after

receiving a new message with a bigger time stamp. But even this solution has a flow - only one of the nodes with the same time stamp will be printed in the log.

2.3 Vector Clocks

Vector Clocks come to solve the open problem left by the Lamport time stamps. The idea is to use a set of vectors, one for each of the nodes.This way we have the same implementation of the Lamport time stamps, but with the advantage of keeping track of the time stamps of each node individually. This way each worker has time stamps of all the others. The solution now is much better ordered and with a small holdback queue.

3 Evaluation

3.1 Basic Random Implementation

In this implementation we can see the messy order of the log events being printed. With a jitter of 0 we can keep a relatively clean queue, but with the increase of it the logs start to become heavily unordered. With a jitter > 0 the *received* messages will always arrive after the *sending* messages. In Figure 1 represents a relatively clean print with jitter set to 0. Figure 3 represents the situation of messy log prints of the events.

Figure 1: Basic Implementation jitter=0

Figure 2: Basic Implementation jitter set

Figure 3: Comparison of Basic Implementations

3.2 Lamport Time Stamps

It is clearly seen that with the implementation of the logical time through the Lamport time stamps the ordering improves much better. And this is the case also of introducing the jitter - order doesn't change, as the earlier events will be printed always before the later, with the exception of events that have the same time stamp. This is depicted in Figure 4.

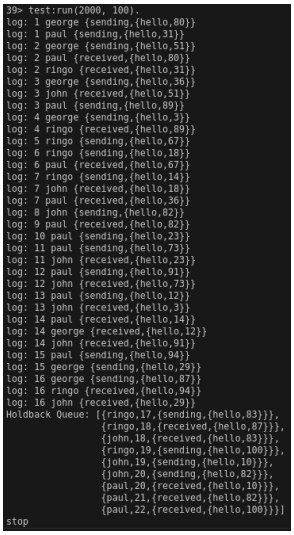

Figure 4: Lamport time stamp

3.3 Vector Clocks

The Vector clocks depict in figure 5 the solution that couldn't be achieved by the simple Lamport time stamps implementation. Now it is observed that the holdback queue is much smaller, in fact constant to always 4 entries.

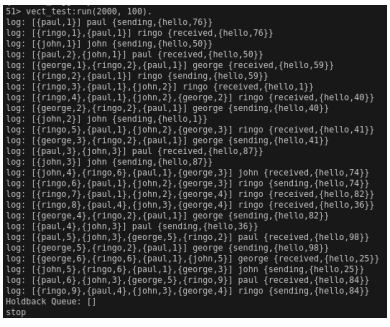

Figure 5: Vector Clocks

4 Conclusions

In conclusion, this assignment was great understand the use of logical time and why the implementation of physical time is impossible in a good working environment of distributed systems. However, the usage of vectors in big distributed systems setups with a huge number of processes can be costly in processing.