COMP4801 Final Year Project

Interim Report

Building Fast Blockchain Applications on an In-datacenter Blockchain Platform

Focus: Network Ordering on Consensus Protocol

Supervisor: Dr. Cui Heming

Mentor: Ji Qi

Member: Pang Chun Ho Nelson, BEng(CompSc), 3035568403

Date of Submission: 23rd January 2022
Notice

This project underwent an administrative group split in the first semester. It changed from a group project to an individual project. Therefore, content related to Kauri presented in Project Plan is removed from this project.

Abstract

Large scale distributed systems such as Nasdaq stock exchange and Amazon Web Services (AWS) are seeking the next generation implementation with better performance and security. Blockchain becomes a popular research topic to find the future enterprise solution for these distributed systems. To meet the demanding performance requirements, blockchain system needs to utilize different techniques and the state-of-the-art consensus protocols. This project investigates a leading consensus protocol, HotStuff, and a high-performance trading blockchain system, BIDL, which is developed by Dr. Cui’s research team. The project revamps the existing implementation of HotStuff by integrating a UDP/IP multicast function to achieve a greater throughput and less latency. The deliverables of the project are a new implementation of HotStuff and its evaluation. As of this stage, the software development is almost completed. Evaluation is expected to conduct in the second semester.
Contents

1. Project Background.................................................................................................................. 4
   1.1 Project Scope.................................................................................................................... 5
   1.2 Project Objective.............................................................................................................. 5
2. Methodology .......................................................................................................................... 5
   2.1 Perquisite Technical Knowledge – libuv library ............................................................ 6
   2.2 HotStuff TCP & UDP Communication Design ............................................................... 6
   2.3 HotStuff UDP Multicast Socket Programming Design ................................................... 7
   2.4 Codebase Modification .................................................................................................. 8
   2.5 Evaluation Design ......................................................................................................... 9
3. Current Progress .................................................................................................................... 10
4. Development Challenge ........................................................................................................ 10
4. Next Deliverable .................................................................................................................... 10
5. Project Schedule ................................................................................................................... 11
6. Conclusion ............................................................................................................................. 11
7. References ............................................................................................................................ 12

List of Figures

Figure 1. Workflow of BIDL ...................................................................................................... 4
Figure 2. HotStuff TCP Communication Diagram ................................................................. 6
Figure 3. I/O Buffer Diagram ................................................................................................ 7

List of Tables

Table 1. HotStuff Network Type ............................................................................................... 6
Table 2. Experiment Setup for HotStuff Performance Evaluation ........................................... 9
Table 3. Project Schedule ......................................................................................................... 11
1. Project Background

The following introduces the basic structure of BIDL and explain why its network ordering feature can produce a better performance.

![Figure 1. Workflow of BIDL](image)

Figure 1 illustrates the five phases of committing transactions to the blockchain in BIDL. Phase 1 is the transaction submission from the clients. Phase 2 is the UDP/IP transaction multicast from the Sequencer/Leader to the consensus nodes and normal nodes. Phase 3 is ordering transaction hashes through consensus protocol run by consensus nodes. Phase 4 is execution of transactions through normal nodes. Phase 5 is committing transactions if and only if the results from Phase 2 and Phase 3 are equal [1].

In Phase 2, Sequencer/Leader is an important component of BIDL to boost throughput and minimize latency due to two reasons. First, it uses UDP/IP multicast, instead of TCP/IP, to transmit the transactions. Assume that there are N consensus nodes, 1 UDP/IP multicast can replace N TCP/IP sends, hence Phase 2 can be more efficient by utilizing UDP/IP multicast. Second, it can order the transactions at the network level by stumping a sequence number on each data packet such that the application-level consensus protocols only need to verify the order established by the Sequencer/Leader (Network Ordering), leading to a more efficient and simpler communication design [2].
For Phase 3, consensus protocol is a modular component in BIDL that means BIDL provides an abstraction on consensus protocol and can run different consensus protocols without intervening other components. The project will focus on HotStuff consensus protocol. HotStuff an advanced version of Practical Byzantine Fault Tolerance (PBFT) consensus protocol with a communication complexity of \(O(n)\) while PBFT has \(O(n^2)\). Hence, it is valuable to integrate HotStuff into BIDL and compare its performance with other sophisticated consensus protocols.

1.1 Project Scope

The project scope is limited to Phase 1, Phase 2, and Phase 3 only, including their implementation and performance evaluation.

1.2 Project Objective

In the existing implementation, HotStuff does not support the Sequencer’s UDP/IP multicast function. Thereby, the current evaluation of BIDL-HotStuff does not reveal the performance enhancement introduced by the Sequencer. This project aims to correct the problem by enabling HotStuff to use UDP/IP multicast.

2. Methodology

The existing communication channel in HotStuff is TCP/IP with an asynchronized and event-driven I/O design. This session discusses the methodology to design and develop UDP/IP multicast function in HotStuff.
2.1 Perquisite Technical Knowledge – libuv library

In addition to the rudimentary TCP and UDP socket programming, HotStuff codebase utilizes libuv which is a C++ event-driven network library. libuv supports similar functions of Linux poll(2). A “libuv poll” is a listener of a file descriptor, it invokes a callback function when the file descriptor encounters specified events. Hence, “libuv poll” is useful for being a timer or a socket listener in HotStuff.

2.2 HotStuff TCP & UDP Communication Design

![HotStuff TCP Communication Diagram](image)

Figure 2. HotStuff TCP Communication Diagram

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Port Number</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Network</td>
<td>TCP 20xxx</td>
<td>submit transactions, collect f+1 votes</td>
</tr>
<tr>
<td>Peer Network</td>
<td>TCP 10xxx</td>
<td>transmit vote, transmit heartbeat message</td>
</tr>
<tr>
<td></td>
<td>UDP 30xxx</td>
<td>multicast proposal</td>
</tr>
</tbody>
</table>

Table 1. HotStuff Network Type
Figure 2 and Table 1 show two types of networks in HotStuff. In Figure 2, lines in blue represent the client network. The client connects to all consensus nodes as it needs to send transactions and then collect at least \("f+1\) votes to ensure the transactions are verified by the consensus protocol. \(\text{“}f\text{”}\) represents the number of faulty nodes in the byzantine fault tolerance (BFT) consensus protocol. Lines in black represent the peer network, peer connects to each other for sending vote and heartbeat message. In Table 1, UDP connection type is a new part of the codebase. At the level of socket programming, multicast group IP address, UDP socket file descriptor and dedicated I/O buffers need to be configured.

### 2.3 HotStuff UDP Multicast Socket Programming Design

![Figure 3. I/O Buffer Diagram](image)

Figure 3 shows how send and receive buffers serve the communication between Leader and peer nodes. From the perspective of Leader, it creates a Conn object whenever it is connected to a peer. A Conn object represents an established TCP connection which is linked to a send buffer and a receive buffer. Send buffer works in one direction only, for sending message from Leader to peer. Vice versa for receive buffer.
HotStuff application adopts a Master-Slave architecture for multithreading. Every Conn object is handled by one worker thread only. While a worker can serve multiple Conn objects. I/O buffer in HotStuff is an event-driven multi-producer single-consumer lock-free queue (Event-driven MPSCQ).

The Logic of Event-driven MPSCQ
1. when a MPSCQ is created, it is first linked to a file descriptor, i.e. a TCP socket
2. then it registers a callback function, i.e. invoking a TCP send() to peer
3. “libuv poll” is listening to a “WRITEABLE” event on the file descriptor
4. Event Loop is started by the main thread
5. a thread pushes a message to MPSCQ, producing a “WRITEABLE” event
6. callback function is invoked, the message is dequeued and sent via TCP send()

- Multi-producer : multiple threads can enqueue, i.e. all worker threads
- Single-consumer : only one thread can dequeue, i.e. a dedicated worker thread
- To ensure thread safe and lock-free : C++ <atomic.h> Compare and Swap (CAS)
- For receive buffer, the logic is similar with a “READABLE” event.

2.4 Codebase Modification

BIDL-HotStuff is using source code from libhotstuff with Docker deployment. A new pair of send and receive buffer are necessary for UDP multicast. They follow the same data structure and procedures as TCP ones. Only the functions related to broadcasting and receiving proposal are required to change. I/O functions for vote and heartbeat message remain intact.

By considering unreliable UDP transmission, Reliable Data Transfer (RDT) 3.0 Protocol will be developed to resist duplicated and corrupted data packets. RDT 3.0 Stop and Wait design can guarantee the reliable delivery of proposal message.
2.5 Evaluation Design

<table>
<thead>
<tr>
<th>Independent Variable</th>
<th>Comparison Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With Sequencer (UDP)</td>
</tr>
<tr>
<td></td>
<td>Without Sequencer (TCP)</td>
</tr>
<tr>
<td>Throughput &amp; Latency</td>
<td>Throughput &amp; Latency</td>
</tr>
<tr>
<td>Number of Consensus Node</td>
<td></td>
</tr>
<tr>
<td>Transmission Loss Rate</td>
<td></td>
</tr>
<tr>
<td>Data Packet Payload Size</td>
<td></td>
</tr>
<tr>
<td>Consensus Block Batch Size</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Experiment Setup for HotStuff Performance Evaluation

Table 2 provides a template for one experiment test case. Experiments should only tune one independent variable at a time in order to examine the performance change specifically. Comparison metrics being throughput and latency to represent the efficiency of HotStuff. Consensus nodes will be deployed with Docker container on the HKU CS servers to simulate the real application.

Goal of Experiments
1. To prove that consensus protocol with sequencer (UDP multicast) can achieve higher throughput at similar or less latency
2. To identify the optimal throughput and latency in representative combinations of parameters
3. Current Progress

Presently, codebase modification is near to completion, meaning that around 60% of the project target is fulfilled. The remaining parts of the software development is RDT 3.0 and debugging. Code change and release management are conducted through GitHub. A technical document is prepared to explain the workflow of the existing and new implementation, so as to facilitate codebase understanding for future BIDL-HotStuff contributors.

4. Development Challenge

The design for UDP multicast is theoretically simple. The real challenge lies in the asynchronized multithreading implementation where there are a lot of conditions need to be considered. Due to nature of the event-driven design, the codebase involves heavy knowledge of libuv library, multithreading architecture, and Linux notification system. It creates a steep learning curve for the developers. In addition, no technical document is provided to illustrate workflows of the codebase, hence generating more time cost for the new contributors. In view of the situation, a technical document is prepared amid the development process.

4. Next Deliverable

After the software development, HotStuff needs to deploy with Docker container. Due to the change in network connection, reconfiguration of the Docker container is required. The next and final deliverable is an evaluation report through conducting a set of experiments. The report shows the peak performance of BIDL-HotStuff in distinct combinations of parameters. The evaluation is expected to take 1 month.
5. Project Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 Oct, 2021</td>
<td>Project Plan &amp; Project Website</td>
</tr>
<tr>
<td>23 Jan, 2022</td>
<td>First Deliverable</td>
</tr>
<tr>
<td>(Current Stage)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interim Report: HotStuff UDP Implementation</td>
</tr>
<tr>
<td></td>
<td>Technical Document for Codebase</td>
</tr>
<tr>
<td>25 Jan, 2022</td>
<td>First Presentation</td>
</tr>
<tr>
<td>18 Apr, 2022</td>
<td>Second Deliverable</td>
</tr>
<tr>
<td></td>
<td>Final Report: Evaluation of HotStuff</td>
</tr>
<tr>
<td>19 - 22 Apr, 2022</td>
<td>Final Presentation</td>
</tr>
<tr>
<td>4 May, 2022</td>
<td>Exhibition of Project</td>
</tr>
</tbody>
</table>

Table 3. Project Schedule

6. Conclusion

In sum, the project goals are enabling HotStuff to employ BIDL’s sequencer multicast capability and evaluate the results. The goals are meaningful to the current BIDL-HotStuff development as it has not fully utilized the performance improvement given by the sequencer. Despite the extensive workload on technical document and complex implementation, software development is in good progress. The final stage involves experiment test case design and adjustment. The next target is preparation for Docker deployment. It is expected that the project can be finished before the mid of second semester.
7. References
