

# Blockchain implements in Christie

155753A 氏名 赤堀貴一 指導教員：河野 真治

## Abstract

Data corruption and inconsistency in computers causes severe problem. Therefore, we detect corruption and inconsistency by Blockchain. The Blockchain is a distributed system. It is possible to compare between hash value and data which is correct. Even if there are incorrect operation and tampering, data are possible to recover with Blockchain.

We are developing Christie and GearsOS. Christie is a distributed framework and GearsOS is a Operating System. GearsOS Filesystem will refer to Christie. if implementing block chain in Christie and implementing it in GearsOS, makes it possible to detect data corruption and inconsistency in the GearsOS file system. In addition, it is possible to configure decentralized distributed network between Gears OS. if it does not configure, data is protected. So, our purpose can be achieved.

In this study, We implement Blockchain in Christie and run it in distributed environment on PC cluster.

## 1 研究目的

ブロックチェーンとは分散型台帳技術とも呼ばれ、複数のトランザクションをまとめたブロックを Hash でつなげたものを、システムに参加しているすべてのノードが保持できる技術である。

当研究室では分散フレームワークとして Christie を開発している。これは GearsOS にファイルシステムに組み込む予定がある。そのため、Christie にブロックチェーンを実装し、GearsOS に組み込むことにより、GearsOS のファイルシステムにおいてデータの不整合を検知できる。また、GearsOS 同士による分散ファイルシステムを構成することができ、非中央的にデータの分散ができるようになる。もし分散システムを構成しない場合でもデータの整合性保持は行え、上記の目的は達成できる。

本研究では、Christie にブロックチェーンを実装し、実際に学科の PC クラスタ上の分散環境で動かす。

## 2 ブロックチェーン

ブロックチェーンを実装することは次のようなメリットが有る。

- データの追跡、検証が容易である。
- 中央管理者が存在しない。単一障害点がない。

データの追跡、検証が容易であるのは、ブロックの構造によるものである。ブロックの構造は簡易化すれば次のようなものである。

- 前のブロックの暗号化ハッシュ。
- タイムスタンプ。
- トランザクションリスト。

ブロックは図 1 のように hash でつながっている。一つのブロックが変更されれば、その後に連なるブロックも整合性が保たれないため、これによってデータの整合性保持が行える。

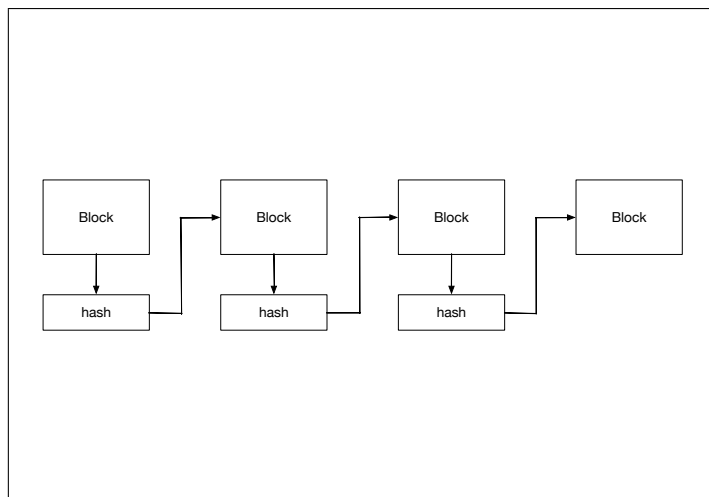


図 1: hash chain

トランザクション、ブロックともにノード間で伝搬され、ノードごとに検証される。そして検証を終え、不正なトランザクション、ブロックであれば破棄する。検証に通った場合は、トランザクションは Transaction Pool に Transaction を貯めておき、ブロックはブロックチェーンに取り組み、検証したノードからトランザクション、ブロックがブロードキャストされる。ノード間は P2P で通信が行われている。

同時に異なるノードで複数のブロックができることを、fork という。これによってブロックチェーンの分岐が起こる。ブロックチェーンの分岐を収束させるにはコンセンサスアルゴリズムを使用する。

### 3 コンセンサスアルゴリズム

コンセンサスアルゴリズムとは、一意の値を分散環境上で決めるためのアルゴリズムである。

今回は分散アルゴリズムとして Paxos を実装した。

Paxos は 3 つの役割のノードがある。

**proposer** 値を提案するノード。

**acceptor** 値を決めるノード。

**learner** acceptor から値を集計し、過半数以上の acceptor が持っている値を決める。

Paxos のアルゴリズムに入る前に、定義された用語を説明する。以下にその用語の定義を示す。

**提案** 提案は、異なる提案ごとにユニークな提案番号と値からなる。提案番号とは、異なる提案を見分けるための識別子であり、単調増加する。値は一意に決まっていほしいデータである。

**値 (提案) が accept される** acceptor によって値 (提案) が決まること。

**値 (提案) が選択 (chosen) される** 過半数以上の acceptor によって、値 (提案) が accept された場合、それを値 (提案) が選択されたと言う。

Paxos のアルゴリズムは 2 フェーズある。

1 つ目のフェーズ、prepare-promise は次のような手順で動作する。

1. proposer は提案番号  $n$  を設定した提案を過半数以上の acceptor に送る。これを prepare リクエストという。
2. acceptor は prepare リクエストが来たら次の動作をする。
  - (a) もし、以前に送られた prepare リクエストの提案番号より、今送られてきた prepare リクエストの提案番号のほうが大きければ、それ以下の提案番号の提案を拒否するという約束を返す。この状態を Promise したという。
  - (b) もし、値がすでに accept されていれば、accept された提案を返す。

2 つ目のフェーズ、accept-accepted は次のような手順で動作する。

1. proposer は過半数の acceptor から返信が来たならば、次の提案を acceptor に送る。これを accept リクエストという。
  - (a) もし、約束のみが返ってきているならば、任意の値  $v$  を prepare リクエストで送った提案に設定する。

(b) もし、accept された提案が返ってきたら、その中で最大の提案番号を持つ提案の値  $v'$  を prepare リクエストで送った提案の値として設定する。

2. acceptor は accept リクエストが来た場合、Promise した提案よりも accept リクエストで提案された提案番号が低ければ、その提案を拒否する。それ以外の場合は accept する。

このアルゴリズムによって、各 acceptor ごとに値が一意に決まる。値を集計、選択するのは Learner の役割である。Learner が値を集計する方法には 2 つの方法がある。

1. Acceptor によって値が accept された時に、各 Learner に送信される。ただし、Message 通信量が、Acceptor の数  $\times$  Learner の数になる。
2. 1 つの Learner が各 Learner に選択された値を送信する。1 の方法に比べて Message 通信量が少なくなる (Acceptor の数 + Learner の数になる) 代わりに、その Learner が故障した場合は各 Learner が Message を受け取れない。

2 つの方法はメッセージ通信量と耐障害性のトレードオフになっていることがわかる。

Paxos でコンセンサスを取ることは、Proof of Work と比較して次のようなメリットがある。

- CPU のリソースを消費しない
- Transaction の確定に時間がかからない。

### 4 Christie

Christie は当研究室で開発している分散フレームワークである。Christie は Java で書かれているが、当研究室で開発している GearsOS に組み込まれる予定がある。そのため、GearsOS を構成する言語 Continuation based C と似た概念がある。Christie に存在する概念として次のようなものがある。

- CodeGear(以下 CG)
- DataGear(以下 DG)
- CodeGearManager(以下 CGM)
- DataGearManager(以下 DGM)

CG はクラス、スレッドに相当し、DG は変数データに相当する。CGM はノードであり、DGM、CG、DG を管理する。DGM は DG を管理するものであり、put という操作により変数データ、つまり DG を格納できる。

DGM には Local と Remote と 2 つの種類があり、Local であれば、Local の CGM が管理している DGM に対し、DG

を格納していく。Remote であれば接続した Remote 先の CGM の DGM に DG を格納できる。DG を取り出す際にはアノテーションを付けることで、データの取り出し方も指定できる。Take, Peek という操作があり、Take は読み込んだ DG が消えるが、Peek は DG を消さずにそのまま残す。また、RemoteTake, RemotePeek というものもあり、リモート先を指定することにより、RemoteDGM からデータを取ることができる。

CG は CGM によって実行されるが、実行するには CG に必要な DG が全て揃う必要がある。もし DG が全て揃わない場合、CGM はずっと listen し、データが揃うまで実行を待つ。

## 5 Christieでのブロックチェーンの実装

Christie でブロックチェーンのブロック、トランザクション、Paxos を実装した。

その際の、Christie の便利な点を述べる。

- ブロック、トランザクションを送るのが簡単。Christie は DataGear という単位でデータを保持する。そのため、ブロックやトランザクションは DataGear に包めばいい。
- TopologyManager でのテストが便利。dot ファイルが有れば、TopologyManager が任意の形で Topology を作れる。そのため、ノードの配置については理想の環境を作れるため、理想のテスト環境を作ることができる。
- ソースコードの機能ごとにファイルが実装できるため、見通しが良い。Christie は CbC の goto と同じように関数が終わると setup によって別の関数に移動する。そのため自然に機能ごとにファイルを作るため、見通しが良くなる。

不便な点を以下に述べる。

- デバッグが難しい。DG での key のスペルミスなどが起こると、CodeGear が実行されず、wait される問題が出る。
- TakeFrom, PeekFrom の使い方が難しい。TakeFrom, PeekFrom は引数で DGM name を指定する。しかし、DGM の名前を静的に与えるよりも、動的に与えたい場合が多かった。
- Take の待ち合わせで CG が実行されない。2つの CG で同じ変数を Take しようとする時、setup された時点で変数がロックされる。このとき、片方の CG は DG がすべて揃っているのに、すべての変数が揃っていないもう片方の CG に同名の変数がロックされ、実行されない場合がある。

## 6 実験, 評価

ブロックチェーンにおいて、分散環境上でテストしなければいけないのはコンセンサスアルゴリズムである。そのため、Paxos を実装し、琉球大学の PC クラスタ上で動かした。

今回は単純化のために、整数でコンセンサスを取る。ノードは proposer が 2 つ、acceptor が 3 つ、learner が 1 つという構成で実験する。

Paxos の実験は 3 回行った。Paxos には log4j2 を実装しており、log を取るようにしている。そのため、その log から値が一意に決まるまでを調べた。1 つの実験の結果を図 2 に示す。

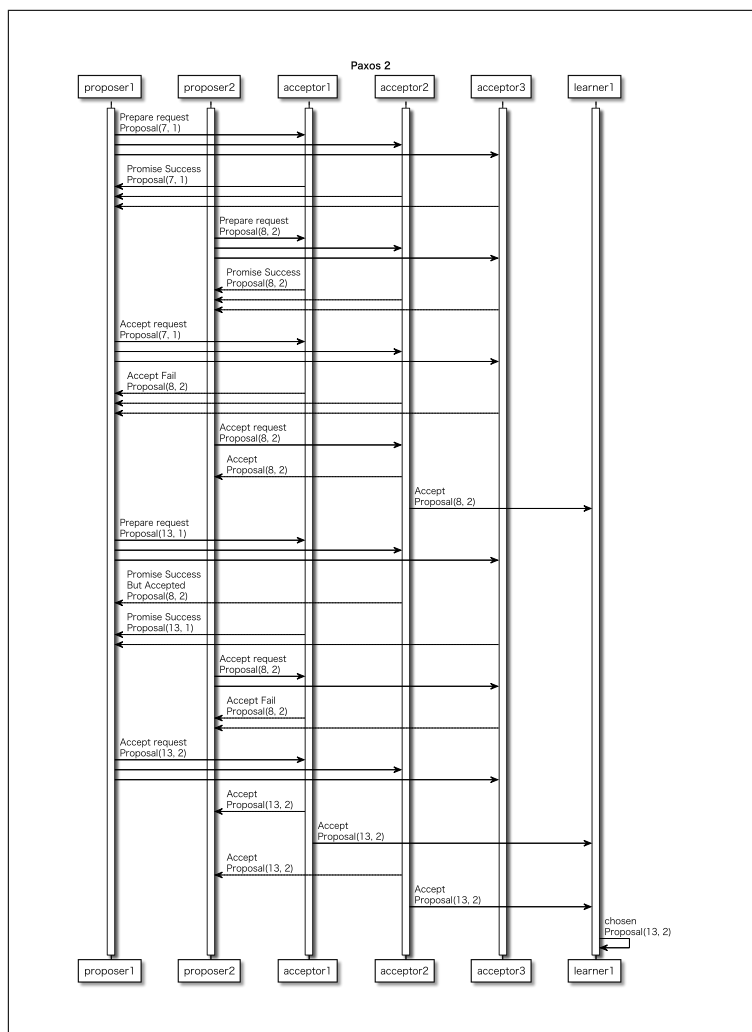


図 2: 実験の結果 1

実際に Learner の値が一意に決まっていることがわかる。

## 7 まとめ

本研究ではブロックチェーンと、分散環境上で値を一意に決めるコンセンサスアルゴリズムを述べ、Christie という分散フレームワークで実際に Paxos を作り、PC クラスタ上で動かした。その結果、コンセンサスの結果として一意のデータが取り出せることがわかった。

まだPCクラスタ上ではブロックチェーンを動かすことができない。しかし、Block, Transaction, Hash 生成, 署名のためのクラスはいずれも作られている。Transaction においてはまだファイルのデータを入れる機能は実装していないが、これらを組み合わせれば簡易的なブロックチェーンが作れる。そのため、あとは Paxos とこれらのクラスをどのようにつなげるかが問題となる。また、Proof of Work を行うクラスも作られている。そのため、ブロックチェーンが実装できた場合、このクラスを用いて Paxos と速度比較も行える。

今後の課題としては、Paxos によるブロックチェーンを作り、分散クラスタ上でファイルのやり取りができるかを見る。その際にどの程度スケラビリティがあるのか、Proof of Work と比較し、どの程度速度が違うのかを見ていきたい。

## 参考文献

- [1] 河野真治. 分散フレームワーク christie と分散木構造データベース jungle. IPSJ SIG Technical Report, May 2018.
- [2] Kaito TOKKMORI and Shinji KONO. Implementing continuation based language in llvm and clang. *LOLA 2015*, July 2015.
- [3] 照屋のぞみ. 分散フレームワーク christie の設計, March 2018.
- [4] Leslie Lamport. Paxos made simple, 01 Nov 2001.
- [5] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*, Accessed: 2019/2/15.
- [6] *Ethereum Homestead Documentation*, Accessed: 2019/2/17.