Christieによるブロックチェーンの実装

155753A 氏名: 赤堀 貴一 指導教員: 河野 真治

1 研究目的

コンピュータにおいてデータの破損や不整合は深刻な異常を引き起こす原因となる. そのため, 破損, 不整合を検知するために, 近年注目されたブロックチェーン技術を用いる. ブロックチェーンは分散システムとして注目されており, データの破損や不整合をハッシュ値によって比較できる. そして, 誤操作や改ざんがあった場合でも, ブロックチェーンを用いることで簡単にデータの追跡が行える.

当研究室では分散フレームワークとして Christie を開発しており, これは GearsOS にファイルシステムとして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破損, 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検証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.

よって, Christie にブロックチェーンを実装し, 分散環境でのデータの整合性保持, 追跡を行う.

2 ブロックチェーン

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参照できる技術である. 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.
- nonce
- トランザクションリスト.

ブロックは前のブロックと暗号化ハッシュでつながっている. 前のブロックのハッシュは, これらのパラメータをつなげてハッシュ化しているため, 現在のブロックのハッシュが作られる際も, その前のブロックのハッシュに依存して作られる. そのため, もしブロックを改ざんしたいとしたら, そのブロックにつながるすべてのブロックを改ざんしなければならない.

その仕組みだけならば複数のブロックのハッシュを同時に改ざんすることで, データが改ざんされてしまう可能性がある. しかし, ブロックに付け加える場合にある作業を行わせ, それによってある条件に収まる Hash を作らせることで, 改ざんの可能性を少なくしている. 例えば, ビットコインだと Proof of Work という計算問題を解かせ, Hash を生成する. これは単純にはソースコード 1 のような問題を解くのと同義である.

ソースコード 1: "Proof of Work を単純にしたコード"

```
while(1){
    randomSeed(前のHash + nonce)
    // 0 < rand() < 10000
    このブロックのHash = rand() \% 10000
    if (このブロックのHash < 100){
        break
    }
    nonce ++
}
```

実際には $0 < rand() < 10000$ はもっと大きな値であり, またこれは複数ノードでの分散環境下で計算される. もし, 条件に合うブロックのハッシュが生成できたならば, 他のノードによってそのハッシュが実際に生成されるかどうかを調べる. これを検証するのは, ハッシュが条件に収まっているか否かを判定するだけで良いので容易である. しかし, 新しい条件に収まる Hash は簡単には求まらない. しかも, 1 つを改ざんすればそれに連なるブロックすべての Hash が変わるため, これら全部を書き換える計算量は膨大なものになる. この仕組みにより, 改ざんを起きにくくしている.

トランザクションの中身はデータ, 前のトランザクションと後ろのトランザクションのハッシュ, 暗号鍵でのトランザクションの署名となっている. 署名により, 誰のトランザクションかが簡単にわかる.

もし至るところでブロックが作られた場合, ブロックに分岐が発生する場合がある. これをフォークという. これは一種の競合状態である. 分岐したブロック同士にある一定の差がついた場合, 長い方を正しいものとし, 競合を解決する.

通信は p2p で行われ, トランザクションやブロックが来た

場合, ノードはそれらが正当なものかをルールに従って検証する. そしてトランザクション, ブロックが承認された場合, 他のノードにそのトランザクション, もしくはブロックを送り, 承認されなかった場合は破棄する. これによって, 承認されたものだけがネットワーク上に伝搬されていく.

このような複数の仕組みによって, 中央管理者を必要とせずにデータの整合性保持が行われる..

3 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.

4 やったこと

中間予稿までにやったこととして, コンセンサスアルゴリズム Paxos の論文を読み, Christie に TopologyManager という機能を実装した.

Paxos を読んだ理由は, コンセンサスアルゴリズムの調査である. 実際, Paxos もビットコインで使用される候補に上がったコンセンサスアルゴリズムである. 分散システムはどのようなコンセンサスアルゴリズムを用いているかで性能が変わる. 例えばビットコインのコンセンサスアルゴリズム Proof of Work は, 計算量を多くして改ざんを起こりにくくしているが無駄が多く, 10 分以内で解かれないように動的に条件を変更している. これは先ほどの, 同時にブロックを変更するのを防ぐため, つまり信頼性を上げるためであるが, 速度面で大きな課題となる. 分散ファイルシステムを構成するにはスケーラビリティが課題であり, ノードの数が多くなればなるほど通信時間がかかる. そのため, コンセンサスリズムとして有名な Paxos の論文を読んだ.

Christie に TopologyManager を実装した理由は, Christie のコードに慣れるため, そして TopologyManager 上に分散システムを実装するのが容易になるからである. TopologyManager とは, ノードに Topology を構成させ, ノードごとにどこのノードにつながればいいかを指定する機能である. Christie では静的, 動的なトポロジー管理ができる. 静的では dot ファイルというものにノードごとの関係を記述する. 動的ではノードの木構造を作る.

また, ブロックチェーンについては実際にブロックを実装し, 簡易的ではあるが Proof of Work を動かして理解を深めた. 分散環境の実装はこれから行う.

5 これからやること

ブロックチェーンのトランザクション部分と分散環境を実装する. そして, 実際に分散環境下においてブロックチェーンを動かし, データの整合性保持, 追跡が行えるかを確認していく. コンセンサスアルゴリズムも調査していき, ファイルシステムに組み込めるコンセンサスアルゴリズムを探していきたい.

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.