ブロックチェーンと IPFS を組み合わせたデータ連携システムの実装と評価

石田達郎[1] 大橋盛徳[1] 近田昌義[1] 藤村滋[1] 中平篤[1]

概要：デジタルトランスフォーメーションの推進に伴い業界や企業を跨いだデータ連携やファイル流通への期待が高まっている。今回の報告では、権限をブロックチェーンで、ファイルを分散ストレージ IPFS で管理し、制御プロセスで両者を連携させるシステムを実装した。以前の提案から性能観点、セキュリティ観点での機能追加を行い、そのファイル登録の性能を評価した

Implementation and Evaluation of Data Sharing System Combining Blockchain and IPFS

TATSURO ISHIDA[1] SHIGENORI OHASHI[1] MASAYOSHI CHIKADA[1]
SHIGERU FUJIMURA[1] ATSUSHI NAKADAIRA[1]

1. 背景

近年、デジタルトランスフォーメーションの推進に伴い業界や企業を跨いだデータ連携やファイル流通への期待が高まっている。経済産業省では、企業、政府、自治体でのデジタルトランスフォーメーションを推進しており[1]、これまで以上にデータの連携やファイル流通が促進されていくと期待される。しかし、同じ業界での競合企業間や業界を跨ぐ場合では、システム同士の結合が難しい実態がある。システム統合を行う場合にはその管理のためのコンソーシアムが必要と考えられるが形成は容易ではない。これらは、デジタルトランスフォーメーションを勧めるための大きな障壁になる。

そこで、参加者間でデータ管理の透明性の確保を前提とするブロックチェーン（以下 BC）を利用することで社間を跨いだデジタルトランスフォーメーションの課題を解決したい。BC は、Bitcoin[2]で用いられているトラストレスなデータ管理を行う技術である。以下に特徴を挙げる

(1)権限分散したデータ管理システムであり、特定の管理者を持たない。

(2)複数の端末で同じデータを管理し、互いに検証しながら同一の台帳として保持する。参加している複数の端末で互いに内容を検証しあっているため、内容の改ざんが困難である。

(3) BC 上にコントラクトの形でプログラムを組み込むことで、スマートコントラクトとして自動的にプログラムを実行できる。

上記のうち、特に(1)の性質によって前述の懸念点は払しょくできる。他の業界同士であっても、プロトコルさえ決まっていれば同一の基盤で情報をやりとりすることが容易であるため、業界を跨いで同じシステムを用いることが可能である。

同様に、競合他社が利用しているシステムであっても、(1)(2)によって競合の組織同士であっても同一のシステムを利用することが可能である。なぜなら、権限分散システムで管理者が競合の組織ではないうえ、内容が改ざんしづらいのであり、競合の組織からの内容改ざんなどを心配する必要がないためである。

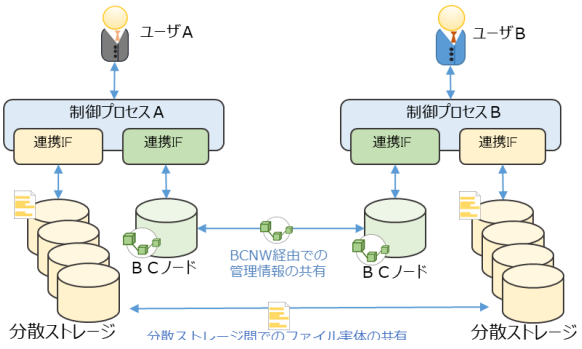

図 1 システム概要図

しかし、BC をファイル流通用のシステムとして使う場合、複数端末で同じデータを持ち合い検証するため、ファイルサイズが大きくなるほどコピーが増える。そのため、ノード間での通信量が増加する点、また、ストレージ容量の観点で非効率になってしまう。

そこで、BC 外部にストレージを用意し、データストレージとして使い BC と組み合わせる方法も検討できるが、この場合ストレージサービスに権限が集中してしまう。

上記の課題を解決するために、BC と分散ストレージを

1 NTT サービスエボリューション研究所

組み合わせる技術が検討されてきた[3][4][5]。BC と分散ストレージを組み合わせることで以下の利点がある。

・BC に契約の情報を載せ、ファイルの実体を IPFS ノードに載せることで、すべての端末に同じデータを載せる必要がなく、システム全体のファイルサイズの点で効率的になる。

・分散ストレージでは、ファイルが断片化されているため、ファイルの一部が流出しても内容を読み取られにくい。

これまでの議論をまとめる。電子ファイルを BC と分散ストレージを組み合わせたシステムで流通させる。特定の管理者がおらず、内容の改ざんが困難であるため、競合の組織であってもシステムに参加できる。BC と分散ストレージを用いて、特定の場所でファイルを管理しないことで、セキュアかつ効率よくファイル管理ができる。

2. 実装

我々も、BC と分散ストレージを連携させる手段を検討してきた[5]。

基本検討では、BC と外部のストレージを連携させるための制御プロセスを用意し、制御プロセスを信頼できるプロセスとしたモデルであった。

基本方式検討の後、今回は実サービスとして利用するための実装を行った。

基本方式検討時との差分は以下である。

(1) 基礎検討時には、BC プロダクトとして Ethereum を用いたが、今回の実装では、BC プロダクトとして Hyperledger Fabric [6]（以下 HLF）を用いた。

(2) 分散ストレージとして InterPlanetary File System [7]（以下 IPFS）を用いているが、今回は IPFS に冗長性を持たせる IPFS クラスタ[8]を実装した。

(3) 複数の組織で本システムを使う際に、特定の組織にファイルを渡したくないといった要求があると考え、特定のノードにファイル共有させないようにできるファイル共有の制限を設けた。

(4) ファイル削除機能として、IPFS ノードからファイルを物理削除できるような要求を出すようにした。

(5) IPFS には、特定のファイルを明示的にキャッシュしておく Pinning という機能がある。性能面からこの機能を無効化した。

(6) 制御プロセスにユーザの本人性検証のための電子署名を追加し、ブロックチェーンで署名検証をすることでユーザ認証を行うこととした。

HLF は、分散台帳基盤であり、参加者を絞った許可制の BC を志向している。コンセンサスアルゴリズムが入れ替え可能なプラガブルな設計で、今回、コンセンサスアルゴリズムとして Raft を利用した。バージョンは 1.4 系を利用した。IPFS は、ファイルを断片化して複数のストレージで管理するシステムである。利用者は、ファイルを一意に示す URL を用いて、そのファイルの物理実態がどのストレージに保存されているのか、またそのファイルがどのように分散して保存されているかも意識する必要がない。

HLF と IPFS は共通のインターフェイスを持たないので、これらを連携させるための制御プロセスを用意した（図１）。ファイル登録をする場合を例にとると、制御プロセスは、ユーザからの要求に従って BC にファイルの管理情報を登録し、IPFS にファイル実体を登録する制御を行う。詳しい動きは後述する。

・システムの基本的な機能

ファイルの登録、ファイルを参照する、文書の更新をする、ファイルの論理削除を行う、ファイルの物理削除を行うといった機能を実装した。また、文書が更新された履歴を参照することができる。基本的に、ユーザの操作は BC に記録される。

・システム動作シーケンス

実装したシステムの基本的な動きを以下に記す（図 2）。

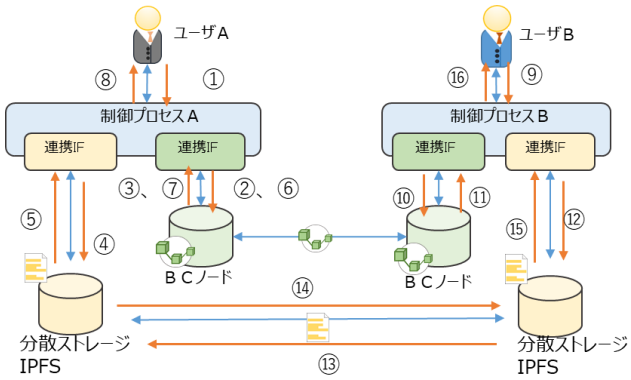

図 2 リクエストの処理順

この図の中では、IPFS クラスタは省略されている。まずユーザ A がファイルを登録するシーンを考える。①ユーザ A が、コンテンツ A を制御プロセス A に登録要求する、②制御プロセスは BC ノードに同じファイルが既に登録されているかどうかを調べる。③ファイルがない場合、ファイル登録可能であることを制御プロセスに返却する、④ファイル実体を IPFS に配置する、⑤IPFS はコンテンツ実体の場所を一意に示す URL を制御プロセスへ返却する。⑥制御プロセス A はファイル A を一意に示す文書 ID と、ファイル実体を示す URL と、権限情報を BC に登録する、⑦ファイル登録したことを制御プロセスに返却し、⑧でユーザ A にファイル登録を返却する。

次に、ユーザ A が登録したファイルを、ユーザ B が参照するシーンを考える。

⑨ユーザ B は、文書 ID をキーに参照要求を自身の制御プロセス B に対して出す、⑩制御プロセス B は、BC に対してユーザ B の参照権限を確認する、⑪確認し、権限が存在すれば BC ノードは制御プロセス B に対して権限があることを返却する、⑫制御プロセス B は、IPFS にファイルを要求する。目的のファイルが、ユーザ B が管理する IPFS ノードに目的のファイルがあればすぐ⑮に従って返却する、⑬目的のファイルがなければ、制御プロセスに対してファイルを要求してよいかを確認したうえで、他の IPFS ノードにファイル共有の要求をする、⑭ユーザ B の IPFS ノードにファイルが返却される、⑮IPFS ノードは制御プロセスに対してファイルを返却、⑯制御プロセスはユーザ B にファイルを返却する。

制御プロセスへ IPFS からのファイルを実際に受け渡しする前に、BC で権限の有無を確認することで、権限のない制御プロセスへのファイルの移動がないようにしている。

3. 実験

ファイルの登録要求を、一定時間の間に処理できる速度を調査した。

測定環境は、AWS を用いて構築した。（表 1）

表 1 測定環境

環境	AmazonWebService(AWS) 東京リージョン
インスタンス	m5d.xlarge
ブロックチェーン	Hyperledger Fabric(v1.4.2) node x 2 orderer x 1 CA x 1
分散ストレージ	InterPlanetary File System(IPFS) IPFS Cluster x 3 IPFS Node x 3

二つの組織を想定し、表 1 のインスタンス群を標準構成として組織 A、組織 B の二つのセットを用意した。

コンテンツ登録を単数または複数同時に要求し、それぞれの時間を計測した。手法は、クライアント相当のインスタンスから要求を出し、その要求がクライアント内で発生したときの時刻から、システムにファイル登録されクライアントに応答が戻ってくるまでの時間を調査した。

これは図 2 では、①から⑧までに相当する。登録するファイルは、数バイトのテキストファイルで、要求ごとに異なるファイルを登録することとした。ただし、ファイルサイズには、ばらつきがないようにした。

ファイル名	0000000001.txt
ファイル内	0000000001

上記のようなファイルを、要求ごとにインクリメントすることで要求ごとに異なるファイルを用いた。

1 要求を複数回行い、1 要求当たりの平均値の時間変化を調べた。クライアントからノードのリクエストとして出すよう、ShellScript で記述した。

4. 結果

1 要求を 10 回計測し、その平均を求めた（表 2）。それぞれの計測は十分な時間を空けて、前の処理の影響がないように実施した。1 要求の値が 0.58 秒でトランザクションを処理していることから、本システムでは約 1.7 トランザクション/秒でファイル登録を処理できると期待される。同時に複数要求を出すとその値は低くなっている。

トランザクション/秒は表 2 に記載した。

表 2 コンテンツ登録の速度

ContentAdd	1req
average	0.58
stdev.	0.03
	(units:seconds)

表 3 登録速度を、トランザクション/秒で表示

1req	
1.7	
(units:transaction / seconds)	

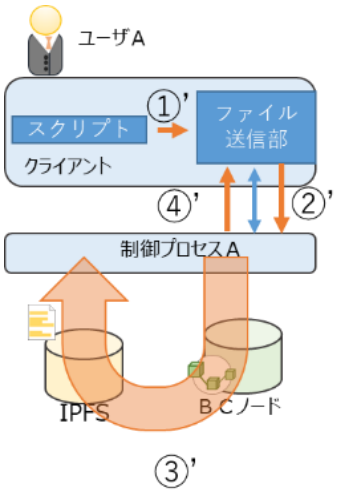

図 3 処理イメージ

次に、ファイル登録処理のボトルネックを評価することとした。表 2 のうち、特にクライアント内部での処理にかかる影響を減らして調査するために、図 2 の①から⑧まであるファイル登録の処理のうち、①の要求を送信するタイミングから、応答を受信したタイミングまでを計測するように変更した。

これを、図 2 の一部を詳細化したものが図 3 になる。結果を表 4 に示した。トランザクション/秒は表 5 にまとめた。

表 1 の結果は①'から④'の間であるが、表 4 の結果は、②'から④'までの間である。これによりクライアント内の処理にかかる時間の影響を減らすことができる。

表 2 と表 4 を見比べると 0.58 秒から 0.44 秒であり、24%程度の時間短縮が見られる。

表 4 コンテンツ登録の速度(2)

ContentAdd	1req
average	0.44
stdev.	0.03
	(units:second)

表 5 登録速度を、トランザクション/秒で表示(2)

	1req
	2.25
(units:transaction / second)	

5. 考察

今回は複数の組織を想定しているが、AWS の同一リージョン内で複数組織を想定した環境を構築した。複数組織が、国を跨いでの実行を確かめるために、複数リージョンにまたがる形で組織 A と組織 B を構築し、ネットワークの影響を評価することも必要だろう。2.実装で記述したうち、今回の実装において性能に影響を与えそうな変更点として、(2)IPFS クラスタ、(5)Pinning、(6)ユーザ認証があげられる。IPFS クラスタについては、今回 10byte 程度のテキストファイルであったため、ファイルサイズを大きくした場合や、十分なファイルを登録した際に、システムの速度に大きな変化がないことを確かめる必要があるだろう。ファイルサイズが大きくなると、IPFS および IPFS クラスタはファイルをチャンク化して登録するため、その処理の有無で性能が変化することが考えらえる。また、多くのファイルを登録した場合は、ファイルの参照についても、同様の評価が必要になると思われる。ファイル参照は、図 2 における⑨から⑯の過程であり、この速度調査は今後の調査対象としたい。

(5)については、その有無によって速度の変化が起きると考えられるものの評価できておらず、今後の課題となる。(6)のユーザ認証については、ユーザからの要求に対して真にユーザからの要求であるかを確認する目的で、BC 内で署名検証を行っている。図 2 のうち、②の BC への要求の際に、検証を行っている。BC での検証は、ブロック生成の時間に従うため、通常のサーバでの署名検証よりも時間がかかると考えられる。

今回は、1 要求を出したのみであり、並列的に 10、100 という要求を出した際の負荷を計測する必要がある。

6. まとめ

今回の報告では、権限を BC で、ファイルを IPFS で管理し、制御プロセスで両者を連携させるシステムを実装した。以前の提案から性能観点、セキュリティ観点での機能追加を行い、そのファイル登録の性能を評価した

参考文献

[1] 経産省 METI DX
https://www.meti.go.jp/policy/digital_transformation/index.html

[2] Bitcoin: A Peer-to-Peer Electronic Cash System https://bitcoin.org/bitcoin.pdf

[3] M.Steichen *et al.*, “Blockchain-Based, Decentralized Access Control for IPFS", IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData),30 July-3 Aug. 2018

[4] S.Wang *et al.*, "A Blockchain-Based Framework for Data Sharing With Fine-Grained Access Control in Decentralized Storage Systems", IEEE Access,Volume:6, 38437-38450pp., 29 June 2018

[5] S.Ohashi *et al.*, "Token-Based Sharing Control for IPFS", IEEE International Conference on Blockchain, 14-17 July 2019

[6] https://www.hyperledger.org/projects/fabric

[7] https://ipfs.io/