デジタルプラクティス　Vol.10 No.3(July 2019)

特集号招待論文

ブロックチェーン基盤ソフトウェア性能検証 ―Hyperledger Fabric，Quorum，Ethereum の横串比較―

尾根田 倫太郎[1]　秋田 佳記[1]　何 岩彬[1]　竹林 陽[1]　小倉 拓人[1]　鈴木 貴之[2]

[1]三菱UFJインフォメーションテクノロジー（株）　[2]（株）三菱UFJ銀行　デジタル企画部

近年，ブロックチェーン技術に注目が集まっており，EthereumやHyperledger Fabricなどを始めとするさまざまな種類のブロックチェーン基盤ソフトウェア☆1がリリースされている．
一方で，それらのソフトウェアがどのような業務特性に適しているかについての情報や，ブロックチェーン技術の課題としてよく挙げられる1秒あたりの取引性能（スループット等）等々の情報について，各ソフトウェアを調査，検証し，比較した論文は少なく，社会に情報が不足していると言える．
そこで我々は，さまざま存在するブロックチェーン基盤ソフトウェアについて，どのような業務特性に適しているか，どう使い分ければよいか社内で標準化することを最終目的に，まずは各ソフトウェアの内部アーキテクチャの調査と性能の検証を行った．本稿はその成果を論文として公開するものである．

1. ブロックチェーン基盤ソフトウェアの性能検証先行例

ブロックチェーン基盤ソフトウェアの性能は，さまざまな企業や大学などにより検証され，情報が公開されている[1][2][3]．一方で，1つのブロックチェーン基盤ソフトウェアについて検証した情報は多いものの，同一の環境で同一の処理内容のスマートコントラクト（ブロックチェーン基盤ソフトウェア上で動作するプログラム），同じユースケースで複数のブロックチェーン基盤ソフトウェアを性能比較した情報は少ない．加えて，ブロックチェーン基盤ソフトウェアの開発元自身が性能を検証した結果の公開である場合が多く，詳しい検証条件の記載がない文献も多い．そこで本稿では，可能な限り条件を揃えた上で複数のブロックチェーン基盤ソフトウェアについて性能検証を行い，比較を行う．

2. 検証対象のブロックチェーン基盤ソフトウェア

将来的に複雑な処理を行うことを見据え，スマートコントラクト機能を持ち合わせているブロックチェーン基盤ソフトウェアに絞って検証対象を選定した．

筆者らの所属する組織は，金融機関向けシステムの開発・保守運用を行う会社であるため，エンタープライズ用途を前提に設計されているHyperledger FabricとQuorumを検証対象として選定した．

Hyperledger Fabricと Quorumは，複数の組織が集まってブロックチェーンネットワークを構成するブロックチェーン基盤ソフトウェアである．このようなブロックチェーン基盤ソフトウェアをコンソーシアム型ブロックチェーンと呼ぶ．

一方で，誰でもネットワークに参加可能なブロックチェーンをパブリック型ブロックチェーンと呼ぶ．このパブリック型ブロックチェーンとの性能比較も併せて行うため，Ethereumも検証対象とした． パブリック型ブロックチェーンとコンソーシアム型ブロックチェーンとの比較を**表1**に示す．

表1　パブリック型とコンソーシアム型の比較

ネットワーク形態	パブリック型	コンソーシアム型
特徴	取引後にノード間のデータ同期を行う	取引と同時にノード間のデータ同期を行う
管理主体	管理者無し	管理者有り
基盤ソフトウェア例	Bitcoin，Ethereum	Hyperledger Fabric，　Quorum など
ノード保持者	誰でも保持可能	許可制
	不特定，悪意のある参加者を含む可能性がある	ノード保持者の身元が判明しており，信頼できる．
コンセンサス方式	Proof of Work 等	Byzantine Fault Tolerance 系のアルゴリズム，Raft など
	消費電力が多い	高速，低消費電力
データの記録完了保証	無し	有り

パブリック型ブロックチェーンの例として，最もよく知られているブロックチェーンであるBitcoinを挙げているが，本稿ではスマートコントラクト機能を持ち合わせたブロックチェーン基盤ソフトウェアのみを検証対象としているため，スマートコントラクト機能がないBitcoinは検証対象外とする． 各ブロックチェーン基盤ソフトウェアのバージョンは，本稿執筆時点に公開されており検証が完了している**表2**のバージョンとする．

表2　検証対象ソフトウェア，バージョン

Hyperledger Fabric 1.4
Quorum 2.1
Ethereum 1.8

3. 性能検証環境，検証条件

本章ではハードウェア環境（3.1節），および検証対象であるブロックチェーン基盤ソフトウェアの検証の前提（3.2～3.5節）について述べる．

3.1 ブロックチェーン基盤ソフトウェア共通事項

3.1.1 IaaSスペック

Amazon Web Service上で，**表3**のインスタンスを作成し検証を行った．アプリケーションもブロックチェーンノードも表3のインスタンスを使用している．

表3 検証に使用したIaaSのスペック 脚注[☆2]

インスタンス名	m5.large
CPU コア型番	Intel Xeon Platinum 8175M CPU @ 2.50GHz
vCPU 数	2 個
ECU ☆2	8
メモリ容量	8GB
ストレージ種別	EBS
ストレージ容量	30GB
ネットワーク帯域	10Gbps
リージョン	すべて ap-northeast-1 アジアパシフィック (東京)リージョン内で検証
ブロックチェーンノード数	台帳を持つノードを 4 台，同一 Availability Zone（同一サブネット）内に配置．

3.1.2 検証で利用したユースケース

本稿の目的は，共通のユースケースを用いることで，複数のブロックチェーン基盤ソフトウェアの性能を比較・評価することである．その処理性能を計測する際に動作させるスマートコントラクトとして，貿易金融取引や契約内容を電子化したものなど，複雑な処理内容のスマートコントラクトを使用することも可能ではある．一方で，スマートコントラクトの処理内容が複雑であると，業務処理内容の複雑度が処理性能に及ぼす影響が大きくなり，純粋なブロックチェーン基盤ソフトウェアの性能を抽出できない可能性がある．よって，処理内容がシンプルである口座Aから口座Bに1コイン資金移動を行う，送金処理をユースケースとすることで，ブロックチェーン基盤ソフトウェアの基本性能を明らかにすることとした．

3.1.3 計測項目

性能検証では，ブロックチェーン基盤ソフトウェアに対して，アプリケーションサーバ（以下APサーバ）上に立ち上げた複数のプロセス（以下検証プロセス）から合計10万件となるようにトランザクションを並列に送付し，処理性能の指標として，取引開始から取引の記録完了までに要した時間，およびスループットを算出した．

計測した各項目の数値を基に，以下の観点で比較を行った．

① 時間経過に応じて処理性能がどのように変化するか
② 検証プロセスの多重度の変化に対して処理性能がどのように向上するか

ここで多重度とは，検証時に同時並列的にトランザクションを送付する検証プロセス数の合計である．異なる多重度で複数回検証を実施し，その処理性能の変化を比較する．1台のAPサーバ上に並列起動できるプロセス数は各ブロックチェーン基盤ソフトウェアの提供するSDKの仕様や開発言語によっても異なるため，多重度を増やす過程でAPサーバ側のCPU使用率などのリソース使用状況に応じて適宜APサーバの台数をスケールアウトし，APサーバ側が性能検証のボトルネックとなることを避ける．

3.2 Hyperledger Fabricにおける検証前提

3.2.1 Hyperledger Fabricとは

Hyperledger Fabric[4]とは，The Linux Foundation傘下のHyperledger（オープンソースコミュニティ）が開発しているオープンソースのブロックチェーン基盤ソフトウェアである．企業の本番システムで使用されることを前提に設計されている．

Hyperledger Fabricは情報を処理・保持するノード（Peer）とトランザクションを集めてブロックを生成するノード（Orderer）で構成される．Peerには，スマートコントラクトを実行してトランザクションを検証するEndorserと，ブロックを検証・記録するCommitterの2種類が存在し，EndorserはCommitterの機能も備える．

3.2.2 検証ネットワーク構成

Hyperledger Fabric検証時のネットワーク構成を**図1**に示す．

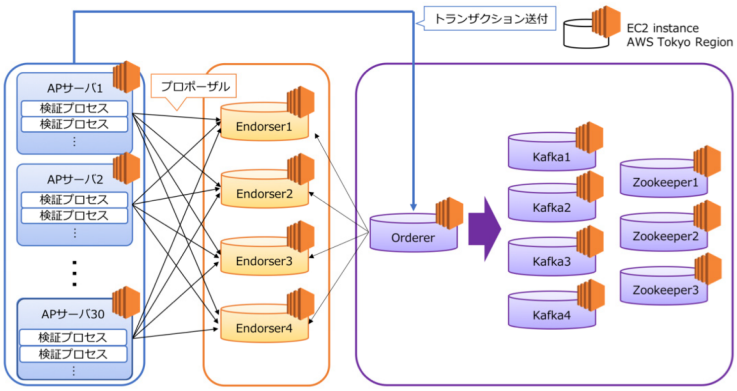

図1 Hyperledger Fabric 検証ネットワーク構成

ブロックチェーンネットワークは改ざん耐性を維持するために，2f+1台（fは障害を許すノードの台数）以上のEndorserから合意を得ることを必須とする場合が多い．企業がブロックチェーン基盤ソフトウェアを使用する場合はシステムに冗長性を持たせたいため，障害を許すノード台数fを1台とした．よって，ネットワーク全体のノード台数は，合意形成が必要なノード3台と，冗長化ノード1台の計4台構成とした．取引を前進させるための条件（Endorsement Policy）についても4台のうち3台以上の合意を必要とする設定を行った．

組織を4つ定義し，1つの組織に1台のノードを所属させた．

Ordering Serviceは，シングルノード構成で取引IDを採番するSoloモードと，冗長構成で取引IDを採番するKafka/Zookeeperモードの2種類が選択可能である．本実験では，冗長構成が必要となる本番環境により近い環境で検証を行いたかったため，Kafka/Zookeeperモードを利用し冗長構成とした．

Kafkaは最低4台，Zookeeperは最低3台用意することを各プロダクトが推奨しているため，推奨されている最小構成とした．

APサーバは30台用意し，1台のAPサーバの中で複数のNode.jsのインスタンスを検証プロセスとして起動する．

加えて，過去の実験よりHyperledger Fabricは取引処理の並列度を上げなければ高い性能を発揮できないことから，Endorserにおける検証処理の並列度（validatorpoolsize）をデフォルト値の1パラレルから400パラレルに，Ordererが1ブロックに含めるトランザクション数（batchsize.max_message_count）をデフォルト値の1個から400個に設定値の変更を行った．

3.2.3　検証方法

アドレスAからアドレスBに送金するスマートコントラクト（Go言語）を開発し，そのスマートコントラクトを検証プロセスから呼び出す形で検証を行った．各検証プロセスからシーケンシャルに各ノード（Endorser）へ，取引依頼のループ送信を行う．つまり，検証プロセスの多重度の分だけ取引が同時並列送信されている．

Hyperledger Fabricは，処理の並列度を上げなければスループットを稼ぐことができないため，検証プロセスの多重度を，70多重～900多重まで変化させそれぞれのスループットを測定した．

1件あたりの取引に着目した，シーケンス図および処理時間計測範囲を**図2**に示す．

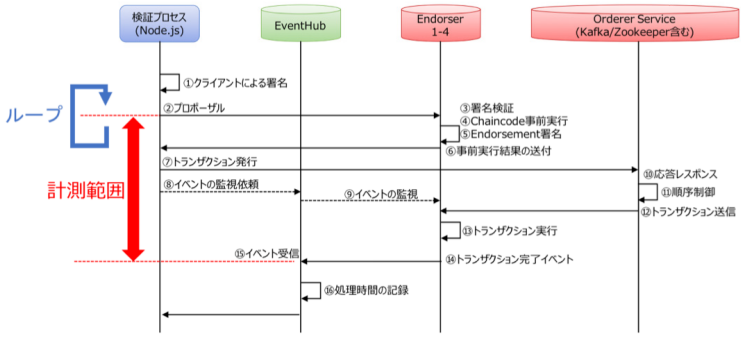

図2　Hyperledger Fabric処理シーケンス

検証プロセスがノードへ取引を依頼した時刻（図2内の②プロポーザル）から，取引記録完了の通知がなされるまで（図2内の⑮イベント受信）の時間を，取引1件あたりの処理時間として記録した．Hyperledger FabricのSDKではトランザクションのコミットをアプリケーション側に

通知するEvent Hubと呼ばれる機能が提供されており，検証プロセスはこれを用いてイベントの受信を行う．

3.3　Quorumにおける検証前提

3.3.1　Quorumとは

Quorum[5]は，J.P．Morganが開発し，現在ではオープンソースとして公開されているブロックチェーン基盤ソフトウェアである．スマートコントラクトにEthereumと互換性があり，データの秘匿化機能を重視しているという特徴を持つ．

3.3.2　検証ネットワーク構成

検証ネットワーク構築には，Quorum環境を簡単に構築できるQuorum Maker[6]を利用し，4台のQuorumノードを構築した．各ノードはコンテナ上で動作しており，相互にRPCプロトコル（**図3**中の黒線）で通信する．Quorumに対してトランザクションを送付するアプリケーションには，Web3.js（バージョン0.20.6）を利用した．このバージョンのWeb3.jsでは，Quorumからの応答の発生をウェイト無しの無限ループ（ビジーウェイト）で監視する実装となっている．このため，単位時間あたりのトランザクション数を増やすために1台のAPサーバ上で複数の検証プロセスを起動したとしても，各検証プロセスがビジーウェイトによってAPサーバのCPUリソースを取り合うため，トランザクションの投入スループットがスケールせず，単位時間あたりに処理できるトランザクション数は変化しないという制約があった．

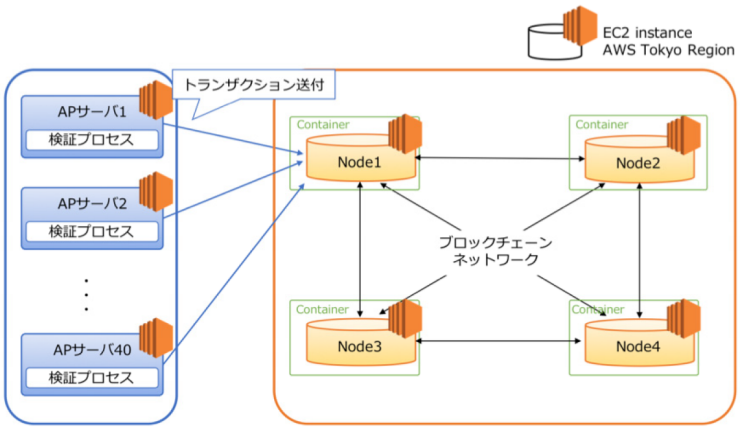

図3　Quorum検証ネットワーク構成

この制約のため，1台の仮想マシンあたり1つの検証プロセスを動作させることとした．多重度を1つ上げるために1台のAPサーバが必要となってしまうため，多重度の限界は40台までとし，1～40多重まで検証を行った．

ノード間の合意形成アルゴリズムは，Istanbul-BFTとRaftから選択可能である．Istanbul-BFTは，ネットワークに参加しているノードの半数以上が賛成している場合のみ取引を成立させる仕組み（以下ビザンチン耐性）が備わっている代わりに，多数のメッセージのやり取りや演算に時間がかかるアルゴリズムである．一方Raftは，ネットワーク参加ノードの中から1台選ばれたリーダノードがデータを生成し，そのデータをネットワークに参加している他ノードへ送信し

て同期する仕組みであり，ビザンチン耐性を備えていない．Hyperledger Fabricもビザンチン耐性を備えていないこと，今回環境構築に利用したQuorum Makerは検証時点ではRaftしか利用できないことから，Quorumも同様にビザンチン耐性を備えていないRaftを選択した．

3.3.3　検証方法

アドレスAからアドレスBに送金するスマートコントラクト（Solidity）を開発し，そのスマートコントラクトを検証プロセスから呼び出す形で検証を行った．各検証プロセスからは，Raftのリーダノードに対してトランザクションを送信した．

トランザクション送信時における各ノードの処理フローと処理時間の計測範囲を**図4**に示す．処理時間としては検証プロセスから取引を依頼した時刻から，検証プロセスが応答（図中の⑧レスポンス）を受け付けた時刻までを取引1件あたりの処理時間として測定を行った．

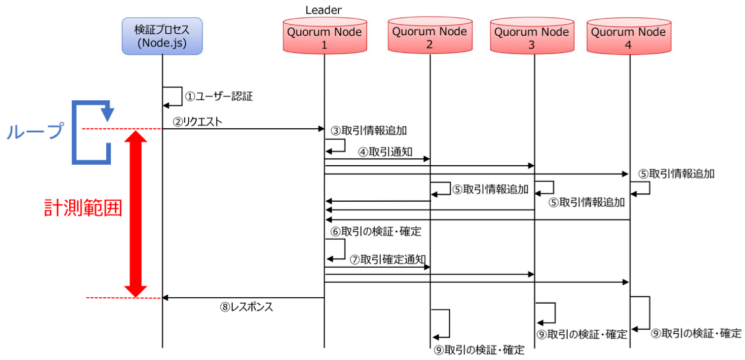

図4　Quorum処理シーケンス

3.4　Ethereumにおける検証前提

3.4.1　Ethereumとは

Ethereum[7]は，Vitalik Buterinが当初開発を行い，今はオープンソースで開発が進められているブロックチェーン基盤ソフトウェアである．参加ノードを絞らない，誰でも参加可能なパブリックなブロックチェーンを指向している点が，本稿の他のブロックチェーン基盤ソフトウェアと比した場合の特徴である．

3.4.2　検証ネットワーク構成

4台のEthereumノードを構築し，相互にRPCプロトコル（**図5**の黒線）で接続してプライベートなブロックチェーンネットワークを構成した．パブリックなメインネットやテストネットには接続をしていない．

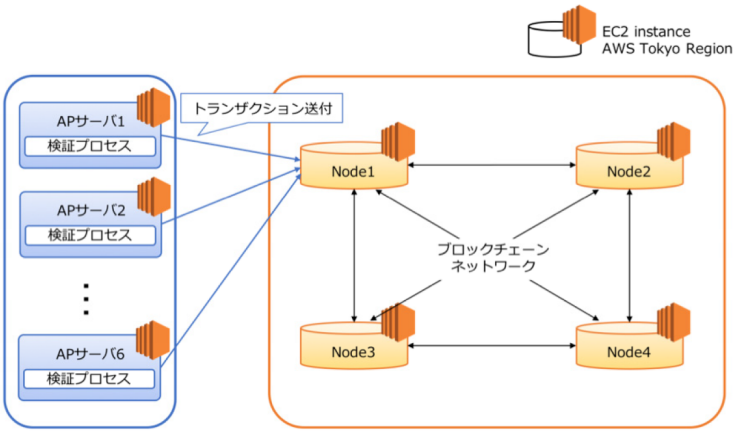

図5　Ethereum検証ネットワーク構成

3.4.3　検証方法

Quorumと同様にアドレスAからアドレスBに送金するスマートコントラクト（Solidity）を開発し，そのスマートコントラクトを検証プロセスから呼び出す形で検証を行った．

Ethereumには取引が完了した際にノード側から検証プロセスへ通知するような仕組みがない．そのため，検証プロセスから取引を依頼した時刻から，ブロックに取引が記録された時刻までを取引1件あたりの処理時間として測定を行った．

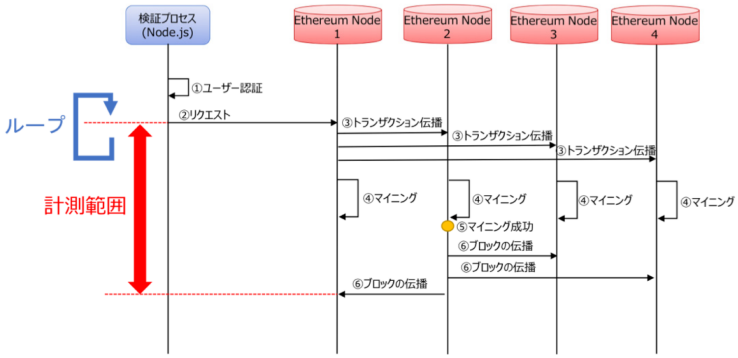

図6　Ethereum処理シーケンス

3.5　各プラットフォームの検証前提比較

3.4節までに述べた各プラットフォームにおける検証の前提条件を**表4**にまとめた．トランザクションを送付するアプリケーションはいずれもNode.jsで実装されているが，Quorum，Ethereumの場合は1台のAPサーバ内で複数の検証プロセスを起動できないことから，多重度を増やすためにはその分のAPサーバが必要となる点がHyperledger Fabricと異なる．

表4　各プラットフォームの検証前提　脚注☆3

	Hyperledger Fabric	Quorum	Ethereum
バージョン	1.4	2.1	1.8
スマートコントラクト開発言語	Go	Solidity	Solidity
アプリケーション開発言語	Node.js	Node.js	Node.js
コンセンサスアルゴリズム	Endorser Orderer モデル ☆3	Raft	Proof of Work
ノード数	4 台 (Endorser) + Ordering Service	4 台	4 台
IaaS	AWS EC2 m5.large	AWS EC2 m5.large	AWS EC2 m5.large
AP サーバ数	30 台	40 台	15 台
計測した多重度	70 ~ 900 多重	1 ~ 40 多重	1 ~ 15 多重
処理時間の計測範囲	リクエスト送付から応答受付まで	リクエスト送付から応答受付まで	リクエスト送付からブロックに取引が記録されるまで

その他，Hyperledger FabricにおけるOrdering Serviceのように各プラットフォーム特有の機能的な差異を除き，可能な限り同条件下で性能検証を行った．

4. 性能検証結果

性能検証として，各ブロックチェーン基盤ソフトウェアに対して複数のプロセスから合計10万件のトランザクションを送付し，各トランザクションの処理時間を計測した．以下，各基盤における処理結果を示し，時間経過に伴う処理性能の変化と多重度の変化に伴う処理性能の変化について考察する．

4.1 Hyperledger Fabric

4.1.1 時間経過に伴う処理性能の変化

図7より，Hyperledger Fabricでは，時間経過に対して安定したスループットを維持し続けていることが分かる．加えて，APサーバのプロセス多重度を300多重，600多重，900多重と増加させた場合，スループットは大きく向上しない一方でグラフ上下のばらつき幅が大きくなるということが分かる．

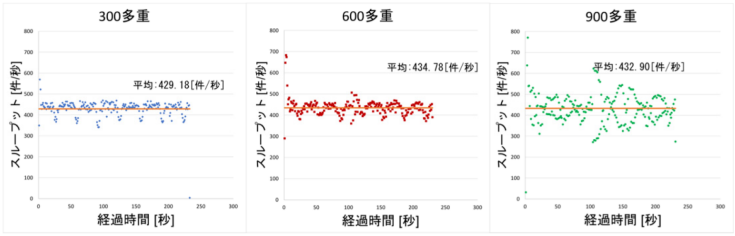

図7　経過時間に対するスループット変化（黄色は平均値）

4.1.2　多重度の変化に伴う処理性能の変化

図8を見ると，多重度を上げるごとに取引1件あたりの平均処理時間については短くなっているものの，多重度が増加するに従って処理時間の分散が大きくなっている．また，各多重度の結果における99.5パーセンタイル値は，300多重で1043ミリ秒，600多重度で1145ミリ秒，900多重で1392ミリ秒であった．つまり，Hyperledger Fabricは，プロセス多重度を上げすぎるとスループットは増加しないにもかかわらず，処理に時間のかかる取引が発生する割合が増えていくという特性を持っていると言える．

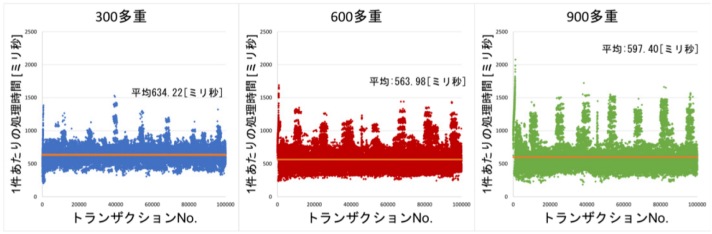

図8　検証プロセス多重度を上げた場合の取引1件あたりの処理時間変化（黄線は平均値）

また，**図9**より，本実験のIaaSスペックの場合，300多重度以降は多重度を上げた場合でもスループットがそれほど増加しないため，300多重度付近が1取引あたりの処理時間を維持したままスループットの最大値を実現できる多重度のピーク値であると考えられる．

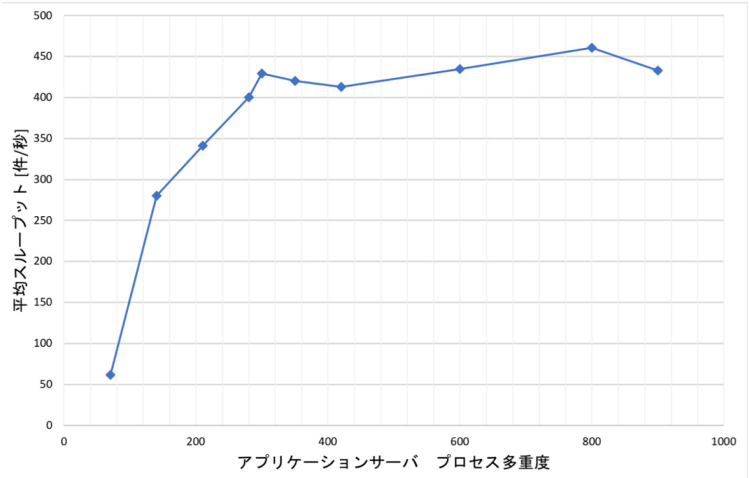

図9　検証プロセス多重度を増加させた場合のスループット増加

どのノードがボトルネックになっているのか明らかにするために，300多重，600多重，900多重の場合におけるEndorserの平均ハードウェアリソース使用率を**表5**，**図10**に，Ordererの平均リソース使用率を**表6**，**図11**にそれぞれ示す．表5，図10より，どの多重度においてもEndorserのCPU使用率のみが高い値で推移しており，Endorserがボトルネックになることが分かった．600多重，900多重では平均のCPU使用率が300多重と比較して下がっているが，図10に示すようにグラフの振れ幅が大きくなっている．

表5　各多重度におけるEndorserのハードウェアリソース平均使用率

	300 多重	600 多重	900 多重
平均 CPU 使用率[%]	94.55	83.88	84.91
平均メモリ使用率[%]	59.38	62.98	63.04

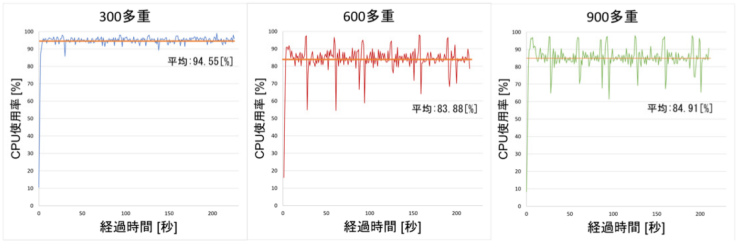

図10　経過時間に対するEndorserのCPU使用率（黄線は平均値）

表6　各多重度におけるOrdererのハードウェアリソース平均使用率

	300 多重	600 多重	900 多重
平均 CPU 使用率[%]	29.24	29.78	30.66
平均メモリ使用率[%]	38.46	38.98	42.23

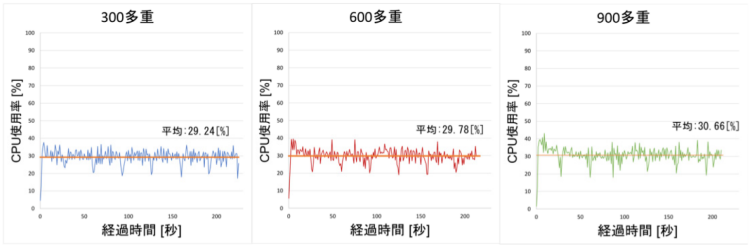

図11　経過時間に対するOrdererのCPU使用率（黄線は平均値）

メモリ使用量についてはどのノードもまだ余裕がある状態であった．また，表6，図11に示す通り，OrdererのCPU使用率は多重度によらず30%以下という結果になった．

4.2　Quorum

4.2.1　時間経過に伴う処理性能の変化

図12より，多重度を上げるにつれスループットが増加し，経過時間に対して安定したスループットを継続していることが分かる．また，スループットの平均値と最大値の差の振れ幅も小さく安定した性能を継続して発揮できていることが分かる．

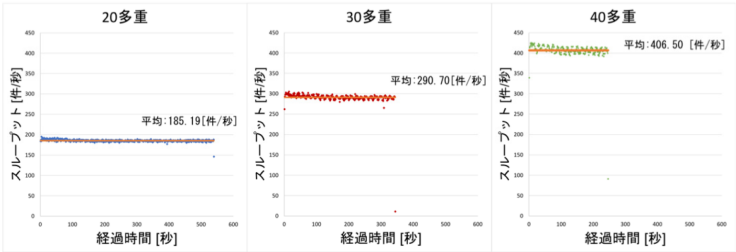

図12　経過時間に対するスループット（黄線は平均値）

4.2.2　多重度の変化に伴う処理性能の変化

図13より，20多重～40多重の間であれば，1件あたりの処理時間が安定しており，多重度の増加に伴って処理時間が増加するような傾向もないことが分かる．

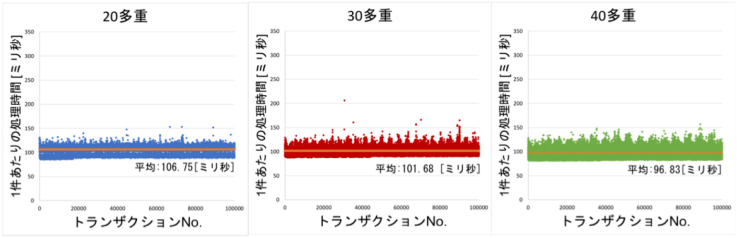

図13　取引の記録完了までの時間（黄線は平均値）

図14より，多重度を上げるほどスループットが線形にスケールすることが分かった．40台以上の多重度の計測は，大量の仮想マシンの作成や制御が必要となることから，スループット限界の把握までは断念した．

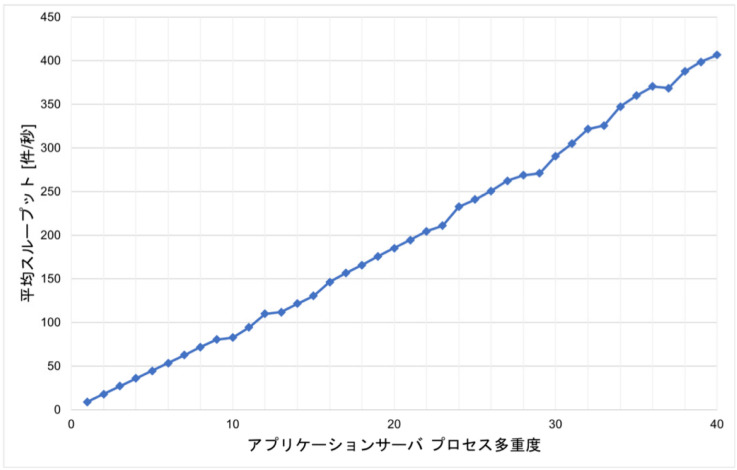

図14　検証プロセス多重度を増加させた場合のスループット増加

表7にリーダノードのCPU使用率，メモリ使用率を示す．いずれも40多重の段階でも50%にも満たない値となっており，リソースに余裕があるという点を示している．

表7　各多重度におけるリーダノードのハードウェアリソース平均使用率

	20 多重	30 多重	40 多重
CPU 使用率平均[%]	14.69	28.81	45.57
メモリ使用率平均[%]	13.81	14.03	15.12

さらに加えて特筆すべき点として，多重度を増加させた場合でも，1件あたりの処理時間のばらつきが大きくならないという点がHyperledger Fabricとは異なる．

4.3　Ethereum

4.3.1　時間経過に伴う処理性能の変化

他のブロックチェーン基盤ソフトウェアの図と比較すると，**図15**からはスループットが低く，最大値と平均値の差も大きいことが分かる．

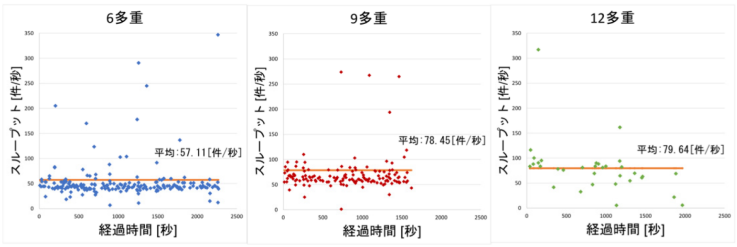

図15　経過時間に対するスループット（黄線は平均値）

4.3.2　多重度の変化に伴う処理性能の変化

図16からは，取引記録完了までの時間が長く，上下幅も大きいことが分かる．Ethereumでは取引が後から失敗になる可能性がある．取引がほぼ確実に受理されるためには，取引後に6ブロック生成される（マイニング間隔15秒×6ブロック＝理論値90秒）まで待つ必要があるとされている．取引が後から失敗となる可能性を極力排除したいエンタープライズ向けユースケースでは，これを守る必要があるために取引完了までの時間が長期化している．

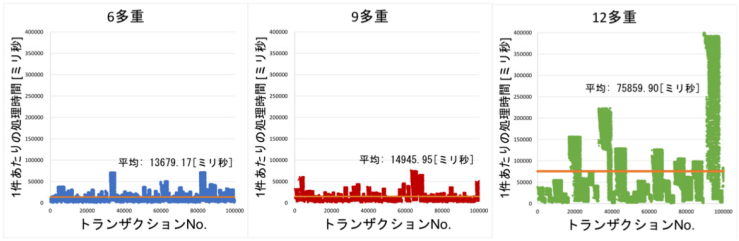

図16　取引の記録完了までの時間（黄線は平均値）

トランザクションの処理時間の上下幅は多重度を増加させるごとに大きくなり，**図17**に示すとおり多重度が10多重から15多重までの間でスループットが向上しにくくなった．10多重以下の多重度ではある程度スループットが線形に増加しているものの，それ以上の多重度ではスループットが上下するなど，不安定な傾向にある．

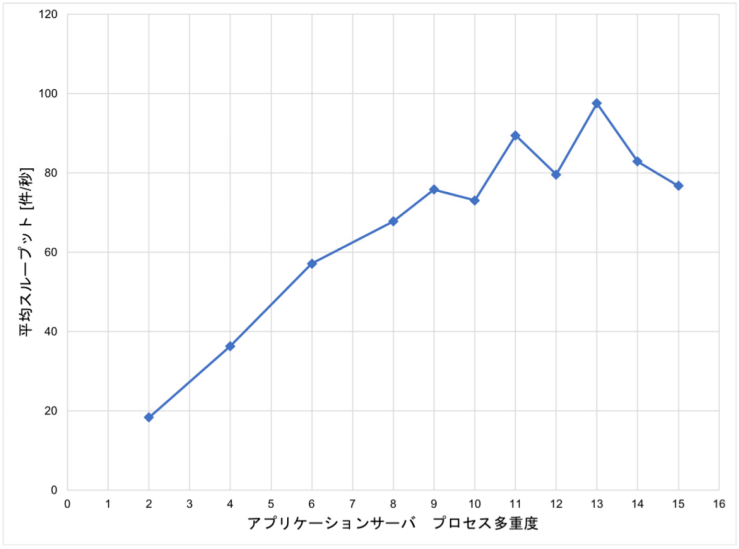

図17　検証プロセス多重度を増加させた場合のスループット増加

スループットや取引時間の上下幅が大きい原因は，ノード台数が4台と少ないためマイニング間隔が15秒に収束しておらず，マイニング間隔の振れ幅が大きいためだと予想される．このことからも，そもそも何万台というノードを接続しあうパブリック型を想定して設計されているため，エンタープライズユースには現状不向きであることが分かる．

検証プロセスからのリクエストを受信したノードのリソース使用率について**表8**に示す．Ethereumではどの多重度においてもCPU使用率が100%近くに達している．これは，Ethereumではマイニングが行われ，その計算にリソースが割かれているためである．

表8　各多重度におけるノードのハードウェアリソース平均使用率

	6 多重	9 多重	12 多重
CPU 使用率平均[%]	99.80	99.81	99.84
メモリ使用率平均[%]	58.69	66.58	61.65

スループットや取引完了までの時間の課題を解消する案は検討が進められているため今後のバージョンアップに期待したい．

5. まとめ

各ブロックチェーン基盤ソフトウェアについて，取引が記録完了するまでの時間や，スループットなどの情報を**表9**にまとめた．ここでは複数の多重度について検証を行った中で，最もスループットが高くなったものを選択した．

表9　取引完了までの平均処理時間および平均スループット

	Hyperledger Fabric	Quorum	Ethereum
多重度プロセス多重度	800	40	13
平均スループット [件/秒]	471.70	406.50	97.53
1件あたりの平均処理時間 [ミリ秒]	643.94	96.84	146,294

横串で検証・比較することで，Hyperledger Fabircは比較的スループットが高いものの高負荷時に取引完了までの時間が増加する傾向があること，Quorumは取引完了までの時間が短くスループットにもまだスケールする余地があることなど，ブロックチェーン基盤ソフトウェアによって特性が大きく異なることが分かった．

スループットについて，Hyperledger Fabircが最も高性能のように見えるが，Quorumについては性能限界までまだ計測できていないことから，今回計測した平均スループットではHyperledger Fabricの方が性能が高いと言えるものの，Quorumの限界性能がHyperledger Fabricよりも低いとは言えない．Ethereumについては，ネットワークに接続可能なノード台数に重きを置いているため，トレードオフとしてスループットが低いことは当然と言える．

ブロックチェーン基盤ソフトウェアを本番システムで使用する場合においては，このような実測値を元にして業務特性や非機能要件にマッチしているか判断する必要がある．

加えて，検証したブロックチェーン基盤ソフトウェアにおいては，検証プロセスの多重度を上げなければ高スループットを実現できなかった．よって，高スループットを実現するためには，取引を多数のAPサーバに均等に割り振り同時並行処理数を可能な限り増やすことが重要である点も，設計考慮点として特記しておきたい．

最後に，本稿ではブロックチェーン基盤ソフトウェアの基本性能を把握することに重点を置いたため，全ノードを同一のAvailability Zone（1つのサブネット内）に配置したが，ブロックチェーンネットワークを本番運用する際は通常，災害対策のために広域にノードを分散配置させることが想定される．場合によっては複数クラウドにまたがるネットワークを構成する可能性がある．遠隔地にノードを配置した場合や複数クラウド利用時の検証については別の機会としたい．

謝辞　本稿をレビュー，修正いただいた日本IBM 吉濱様，日本IBM 上條様，大阪大学 鬼塚様に感謝いたします．皆様からのフィードバックによって，本稿の品質を大きく向上させることができました．

参考文献

1）Thakkar, P., Nathan, N. S. and Viswanathan, B. : Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform, https://arxiv.org/abs/1805.11390 (参照 2019-04-04)
2）佐藤竜也，長沼　健，根本　潤，福地開帆，山田仁志夫：ブロックチェーン基盤Hyperledger Fabricの性能評価と課題整理，電子情報通信学会インターネットアーキテクチャ研究会（2017）, https://ci.nii.ac.jp/naid/40021160493
3）戸澤晶彦，河内谷清久仁，堀井　洋：Hyperledger/Fabricの性能解析，情報処理学会論文誌プログラミング, Vol.10, No.3, pp.20-20 (2017), https://ci.nii.ac.jp/naid/170000148648

4）Hyperledger Fabric : https://www.hyperledger.org/projects/fabric (参照2019-04-04)
5）Quorum : https://www.jpmorgan.com/country/US/EN/Quorum (参照2019-04-04)
6）Quorum Maker : https://github.com/synechron-finlabs/quorum-maker (参照2019-04-04)
7）Ethereum : https://www.ethereum.org/ (参照2019-04-04)

脚注

☆1　Hyperledger Fabircはビザンチン障害耐性を持たないため，厳密にはブロックチェーンに該当しないが，本稿ではそれら分散台帳基盤ソフトウェアも含め，総称してブロックチェーン基盤ソフトウェアと表現する．

☆2　EC2 Compute Unitの略で，CPUの性能を単純比較しやすくするためのCPU処理能力値．1ECUは1.0-1.2 GHz 2007 Opteron または 2007 Xeon プロセッサのCPU能力と同等．

☆3　バージョン1.0以降のHyperledger Fabricでは，Endorsementフェーズ，Orderingフェーズ，Validationフェーズの3フェーズからなる独自の仕組みで合意形成を行う．

尾根田 倫太郎（非会員）rintaro_oneda@mufg.jp
2011年 三菱UFJインフォメーションテクノロジー（株）に入社．銀行内のクラウド基盤の開発業務を経て，2016年より研究開発部門にてブロックチェーンのR&Dに着手．

秋田 佳記（非会員）yoshiki_akita@mufg.jp
2014年 三菱UFJインフォメーションテクノロジー（株）に入社．2016年よりブロックチェーン技術のR&Dに従事．Ethereum，Quorum，Hyperledger Fabricの性能検証を担当．

何 岩彬（非会員）ganhin_ka@mufg.jp
2014年 三菱UFJインフォメーションテクノロジー（株）に入社．2017年よりブロックチェーン技術のR&Dに従事．Hyperledger Fabricの性能検証を担当．

竹林 陽（非会員）hinata_takebayashi@mufg.jp
2016年 三菱UFJインフォメーションテクノロジー（株）に入社．Ethereum，Quorumの性能検証を担当．

小倉 拓人（非会員）takuto_ogura@mufg.jp
2018年 三菱UFJインフォメーションテクノロジー（株）に入社．Hyperledger Fabricの性能検証を担当．

鈴木 貴之（非会員）takayuki_15_suzuki@mufg.jp
2018年より（株）三菱UFJ銀行へ出向．Quorumの性能検証を担当．

採録決定：2019年4月26日
編集担当：鬼塚　真（大阪大学）