ETHTERAKOYA SC-WG

of Nodes (Network size) - the number of validating nodes participating in consensus -

問い

性能評価基準の策定において、コンセンサスに関与するノード数は幾つに設定すべきか？
あるいは各基盤における設定基準はどうあるべきか？

基礎調査

Hyperledger Fabric 2.0での検証結果を通じた論文***"Performance Analysis of Hyperledger Fabric 2.0 Blockchain Platform ",(2020)*** より、ノード(peer)のスケーリングと、ブロックサイズ(bs)、秒間辺りに発生するトランザクション(TPSin)のスケーリングとで以下の傾向が見受けられた。

単一組織内

- 発生トランザクション:10tpsでは、ブロックサイズの容量に関係なく、ノードが増加してもレイテンシー(トランザクション処理時間)への影響は軽微。(基本的に速い: 100ms以下)
- 発生トランザクション:100tpsでは、ブロックサイズの容量に関係なく、ノードの増加に応じて、レイテンシーへの影響が大きい。
  - ✓ 最小ノード構成:2と最大ノード構成:16とでは、約3,000msのレイテンシーの差異が発生。
- 発生トランザクション:1,000tpsでは、ブロックサイズの容量・ノード増加に関係なくレイテンシーは概ね悪化する。(2,500-3,500ms)

複数組織間

- 発生トランザクション:1,000tpsでは、複数組織間によるノード構成において、ブロックサイズの容量・ノード増加に関係なくレイテンシーは概ね悪化する。

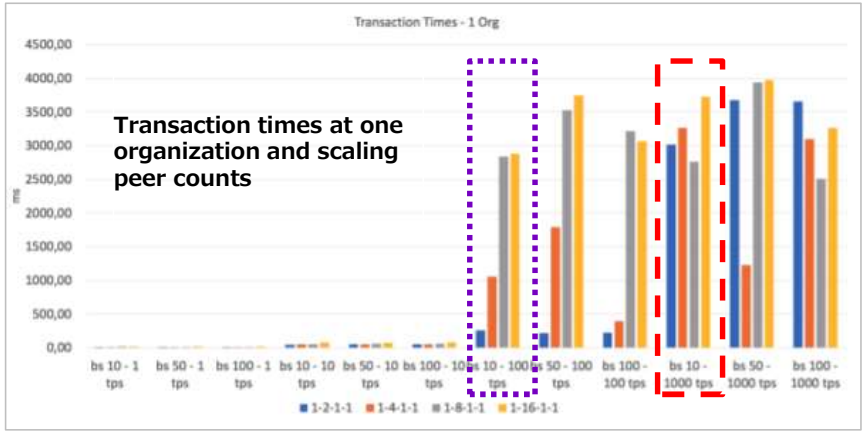
共通事象

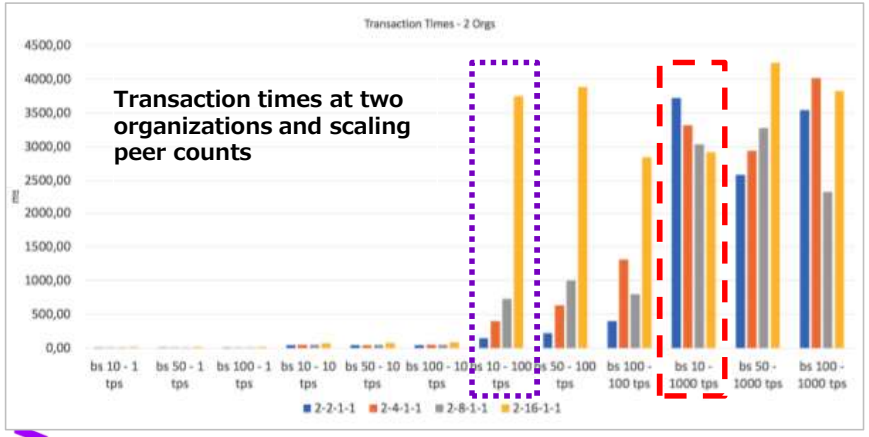
- 発生トランザクション:100tps以上の場合、ノード数:16では、発生トランザクション・ブロックサイズ増加に限らず、レイテンシーの推移に偏りは見受けられない。

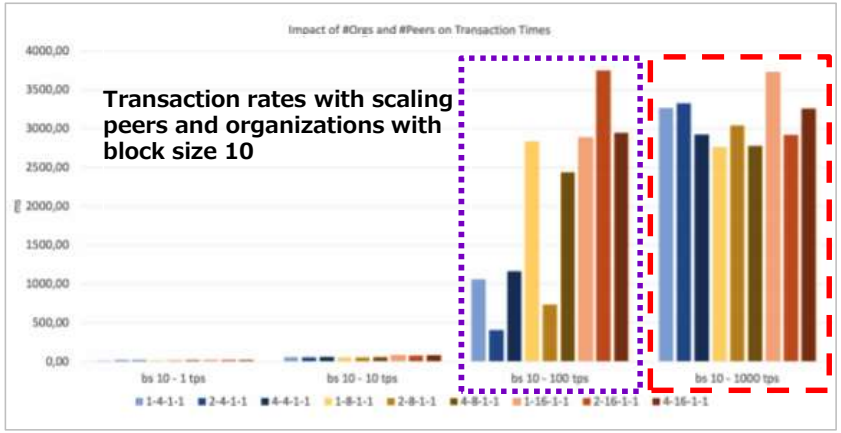
ETHTERAKOYA SC-WG

of Nodes (Network size) - the number of validating nodes participating in consensus -

資料







凡例

- bs10-100tps
- bs10-1000tps

- 発生トランザクション:100tpsでは、ノードの増加に応じてレイテンシー悪化の傾向が見受けられる。
- 発生トランザクション:1000tpsでは、ノード数に応じたレイテンシーの差異は軽微。
- ノード数:16の場合、発生トランザクション:100tps以上では一定したレイテンシーの状態を示す。

ETHTERAKOYA SC-WG

of Nodes (Network size) - the number of validating nodes participating in consensus -

基礎調査

プライベートで組んだ複数ブロックチェーン基盤での検証結果を通じた論文***"Performance Evaluation of Permissioned Blockchain Platforms",(2020)*** より、Hyperledger Fabric以外の各基盤(Ethereum(PoW,PoA)、Corda、Quorumでのノード(peer)のスケーリングに応じたパフォーマンスの傾向は以下の通り。

Ethereum

- PoWでは、基本的にレイテンシー、スループット共にパフォーマンスは悪いが、ノードのスケーリングに左右されない為、ノード数を増やし続けても一定したパフォーマンスを出すことが可能。(特にスループットはノード数:12以上から安定している。)
- PoAは、コンセンサスに関与するノードを中央集権的に管理することが出来る為、ノード数が少ない間はパフォーマンスはPoWに比べてよいものの、ノード数の増加がある一定数(ノード:12)に達すると、以降はスループットが低下する。

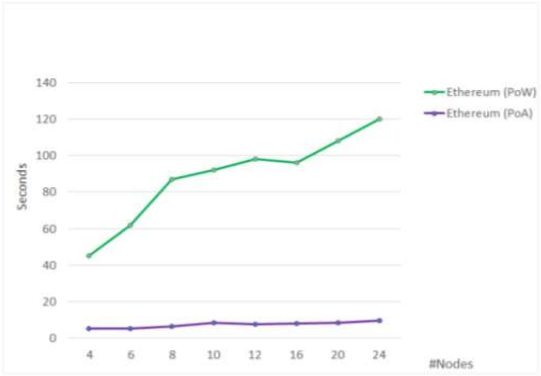


Fig. 3. Latency comparison between PoW-Ethereum and PoA-Ethereum

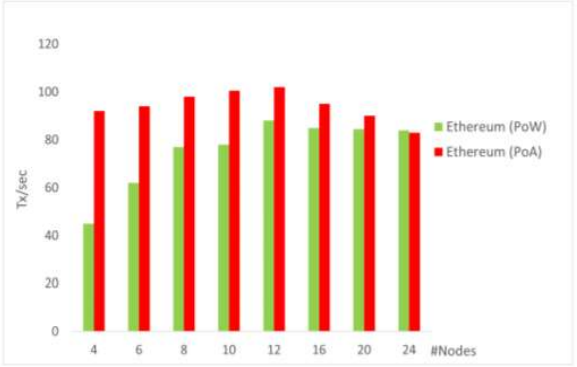


Fig. 4. Throughput comparison between PoW-Ethereum and PoA-Ethereum

ETHTERAKOYA SC-WG

of Nodes (Network size) - the number of validating nodes participating in consensus -

基礎調査

Corda

- Cordaではv4.3-4.5の複数のバージョンで検証。
- バージョンアップに応じてパフォーマンス自体は向上しているものの、何れのバージョンでも、ノード数の増加に応じて直線的にスループット悪化の傾向が顕著に見受けられる。(Fig.5)
- レイテンシーに関しては、v4.5ではノード増加によってあまり影響を受けなくなる傾向が示されている。ただし、ノード数:10を超えた場合での測定結果はない為、更にノード数を増加させた場合のパフォーマンスについては判別できない。

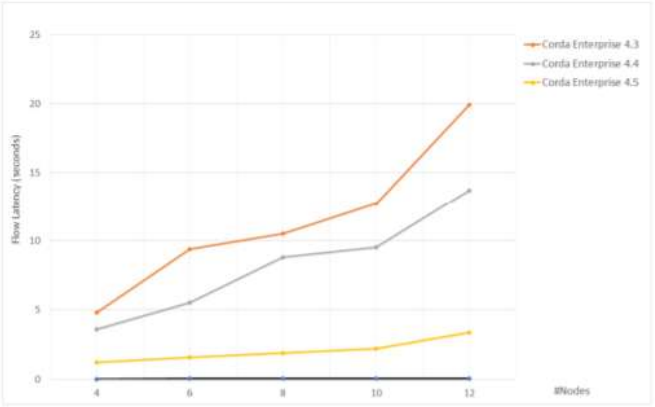


Fig. 6. Latency Comparison among Corda Enterprise 4.3,4.4 and 4.5

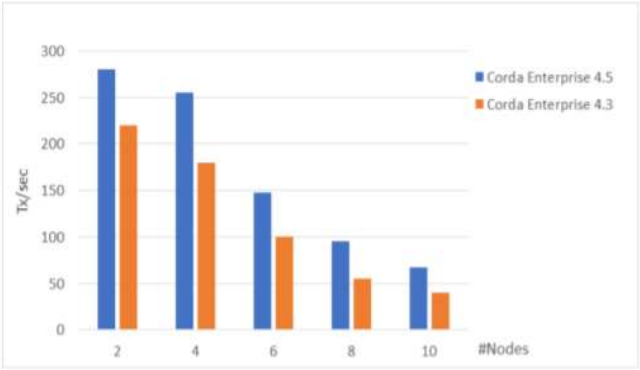


Fig. 5. Throughput Comparison between Corda Enterprise 4.3 and 4.5

ETHTERAKOYA SC-WG

of Nodes (Network size) - the number of validating nodes participating in consensus -

基礎調査

Quorum

- ノードの増加に応じて、レイテンシー、スループットともに悪化する傾向が見受けられる。特にノード数:4を超えてからのスループットの悪化は顕著。
- ノード数:10では、レイテンシーに例外的な事象が観測されているが、論文によると、実験環境の制約によってノード数:10以上では処理が出来なくなる為、測定自体が出来ない状態であるとの見解を示す。

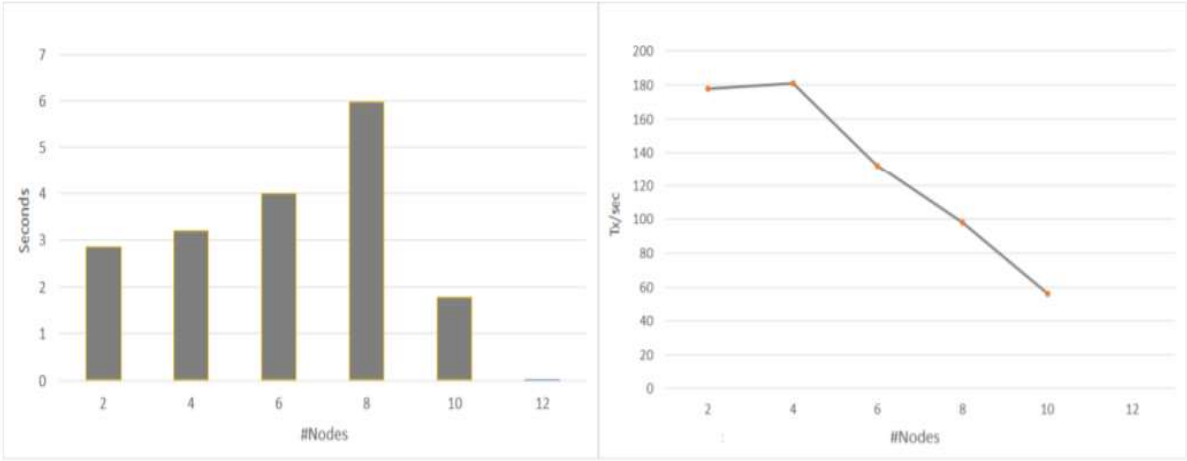


Fig. 7. Latency and Throughput measurement of Quorum (Nodes)

ETHTERAKOYA SC-WG

of Nodes (Network size) - the number of validating nodes participating in consensus -

問いに対する仮説

- 企業間でのブロックチェーンを活用したビジネス展開を鑑みた時、単一組織(単一企業)内で運用することは、通常想定しづらく、複数組織間による複数ノードによるコンソーシアム運営が主流になってくると想定される。
- 上記前提とし、性能評価基準策定においてあり得るべきノード構成を検討する場合、コンセンサスに直接参加するノード・発生トランザクションの増減によってスループット、レイテンシーに影響を及ぼす可能性を極力排除する必要がある。
- 幾つかの論文を通じた基礎調査から窺えることは、パーミッションド・ブロックチェーンは、コンセンサスに関与するノードの増加に応じてスループット、レイテンシー共に悪化する傾向を示している。ただし、Hyperledger Fabric 2.0での検証結果が示唆するように、ノードが一定数に達した段階で安定した傾向を示すことも窺える。
- 一方、パーミッションレス・ブロックチェーンはコンセンサスの方式にもよるが、ノードの増加は、レイテンシーには影響を及ぼし続けるものの、スループットに与える影響は軽微であることが窺える。
- 以上より、性能評価策定の基準となるコンセンサスに関与するノード構成は、各基盤ごとに異なってくるものの、ノード増加の閾値が超えた段階で一定したパフォーマンスに落ち着く状態への収束点を最適なノード数として設定すべき。
    - 例：Hyperledger Fabric 2.0であれば、コンセンサスに関与するノード数(Endorsment Peer):16。Ethereum(PoW)であれば、ノード数:24。

ETHTERAKOYA SC-WG

Typical processing

問い

性能評価を実測する際の最適なプロセスモデルは何か？

考え方の前提

考え方の前提

- ブロックチェーン基盤は同一基盤であってもバージョン相違によって、性能に大きな差異が生じる。
- 性能評価の基準は、性能評価するために実行されるスマートコントラクトにも依存する為、コードの複雑度合・ステップ数の長さ等によって、以前のバージョンや異なる基盤など、他の性能評価との比較ができなくなってしまう。
- スマートコントラクトを通じた性能評価を実施する際、あり得るべきTypical processing(最適なプロセスモデル)を定義する際には、基盤・バージョン相違による影響を極小化することが必要になる。
- 一方、性能評価は最終的には実用レベルでのユースケースに即して実施されることを考慮した場合、各ケースに応じたモデルを示すことも検討される。

ブロックチェーンを用いたシステムに対する性能評価実施時のプロセスモデルの選定観点

- ブロックチェーンの性能評価を実施する際、プロセスモデルとしては、以下のポイントでの検証が一般的。
  1. ブロックチェーン上にトランザクションを発生する際に実施するトランザクションへの署名
  2. スマートコントラクト等を実行し、データ(ステート情報、等)がブロックチェーン上に取り込まれるまでの処理
  3. ブロックチェーン上に取り込まれたデータ(ステート情報、等)をスマートコントラクト等を通じて呼び出す処理
- No.1はブロックチェーンネットワークへの負荷に直接かかわるものではない為、性能評価におけるプロセスモデルとしては、No.2-3のタイミングとする。

Typical processing

サンプル

汎用モデル

- 以下のように、ランダムな値を保持するアセットを生成し、ステートから値を呼び出すスマートコントラクトのようなシンプルな形式であれば、基盤・バージョン差異による影響を極小化し得る。

Listing 1: Java Interface for the test smart contract

```
class DefaultContract implements ContractInterface {
    @Transaction()
    public boolean defaultAssetExists(String _id);

    @Transaction()
    public boolean createDefaultAsset(String value);
}
```

出典:
Performance Analysis of Hyperledger Fabric 2.0 Blockchain Platform(2020)

ETHTERAKOYA SC-WG

Typical processing

サンプル

Token Transfer

- Token transferを性能評価におけるプロセスモデルとして設定する場合、EthereumのERC20を汎用モデルとし、基盤に応じてカスタマイズ。
- 考え方の背景として、トークンを金銭授受(アセット移転)に読み替えたとすると、金銭授受行為自体は、当事者間においては過去の移転履歴はあまり気にせず、実際に送り手から受け手に確実に送金(もしくは手渡し)による移転が完了すればよい。

- **メソッド: ERC20**
  - ✓ **残高照会**
    - ➢ function balanceOf(address _owner) public view returns (uint256 balance)
  - ✓ **トークン移転**
    - ➢ function transfer(address _to, uint256 _value) public returns (bool success)

Typical processing

サンプル

サプライチェーンの来歴

- サプライチェーンにおいては、原材料や製品のライフサイクル全体を来歴情報としてトラッキングできる必要がある。一例としては、製品が最初に転送されたときに、製品情報をトランザクションに追加し、その後の各トランザクションでは、前回のトランザクションのハッシュを参照することで、製品の出所を追跡するハッシュチェーンを形成。

- **メソッド: 来歴参照コントラクト**
  - ✓ **来歴登録**
    - ➢ サンプルでは、予め登録された製品(もしくは原材料)の前回トランザクションを参照し、ブロックチェーン上で承認済みであれば、今回の移転情報(別コントラクトで承認された製品移転時のトランザクション含む)および前回トランザクションを来歴更新コントラクトに追加している。
  - ✓ **来歴照会**
    - ➢ サンプルコード(疑似コード)にはないが、実際にはユーザが来歴参照する為の照会用メソッドが必要になる。プロダクトコード等をキーとして登録された来歴情報を呼び出す。

来歴登録コントラクトの疑似コード(pseudo-code)

```
   Input: The massage sender's address (msg.sender),
          receiver's address (receiver), current
          timestamp (now), current tr (currentTr),
          previous tr (previousTr), authorization list
          (AL), tr count (trCount)
1  AL is the set of all authorized users' Ethereum address in
   this contract;
2  if msg.sender ∈ AL then
3      if previousTr has valid in the blockchain then
4          register currentTr, msg.sender,
           receiver, previousTr, and now to the
           blockchain;
5          productCount++;
6      else
7          Revert contract state and show an error.
8      end
9  else
10     Revert contract state and show an error.
11 end
```

出典:
Smart Contract-Based Product Traceability System in the Supply Chain Scenario(2020)

ETHTERAKOYA SC-WG

Typical processing

Appendix

ブロックチェーンを用いた現状の主な処理例

- サプライチェーンの分野ではスマートコントラクトを使用したトレーサビリティのシステムを活用している。また、トレーサビリティにブロックチェーンを活用することでトークンを発行することができ、生産者や消費者の間に新たなコミュニケーションを創出できる。
- ブロックチェーンを活用した電子契約書ではスマートコントラクトを導入し、改ざん耐性、監査証跡、業務の自動化を実現している。また、電子署名された契約書のハッシュ値をブロックチェーンで管理することもできる。例：法律に則った契約書の作成と契約の実行を行うOpenLawは契約書とユーザーIDのハッシュ値をEthereum上に書き込まれることで信頼性・透明性が担保される。
- 不動産という実物資産をデジタルの権利をブロックチェーン上でトークン化することで権利の譲渡をブロックチェーン上で完結させることができる。
- 銀行業界では、既存の証券をセキュリティー・トークン化し、管理を容易にしている。またトークン化されていることで、譲渡の際に、ブロックチェーン上の記録と、受益証券発行信託の原簿を書き換えるだけで権利が移転できるようになる。
- 先物やオプション取引でスマートコントラクトを導入し、口座管理やマッチング業務を自動実行することができる。また、レポート作成の際に、約定情報などを容易に引き継げるインタフェースを導入することで、デリバティブ商品のライフサイクル全体をスマートコントラクトで一元管理することが可能になる。
- デジタル通貨は、データ自体が現金と同等の価値を持ち、ファイナリティがある。価値を移動させることで日本円と連動するデジタル通貨(XST-JPY)が利用できる店舗で使用できる。