Crossing makes the Future.
まじわる力で みらいを創る

ETHTerakoya

Hyperledger Caliperで行う Besuの性能測定 (Try＆Error)

NTTテクノクロス株式会社
デジタルトランスフォーメーション事業部
第一事業ユニット
兼松 和広

NTTテクノクロス

1. 会社紹介

NTTグループ企業の一つ。

2016年よりFINTECH調査研究の1テーマとしてブロックチェーンに着手。
その後、NTT研究所やグループ各社、および一般企業よりBC関連の開発や実証実験などに多数参加している。

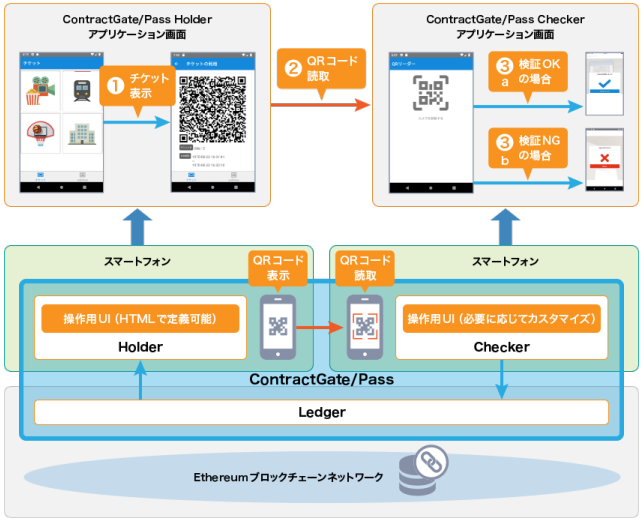
また、自社製品として、スマートフォン上のアプリケーションとそのバックエンドシステムのサービスや
BC上のデータを見やすくするモニタリングツールなどを提供。

Holder
ブロックチェーン上のパス利用アプリケーション

パスの保有者であることを証明するために、ブロックチェーン上で管理されているチケットなどの価値情報をQRコード化して表示するためのスマートフォン上のアプリケーション

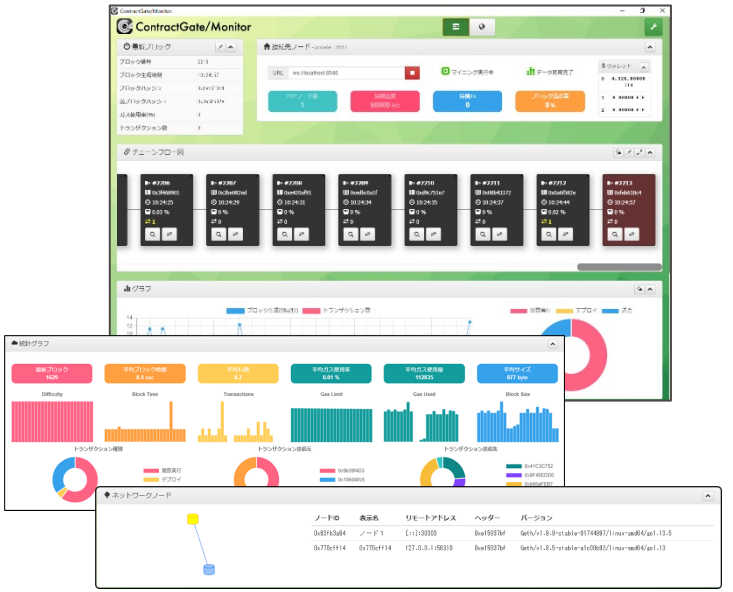
Checker
ブロックチェーン上のパス照合アプリケーション

Holderで表示されたQRコードを、ブロックチェーン上のデータにアクセスして照合するためのスマートフォン上のアプリケーション



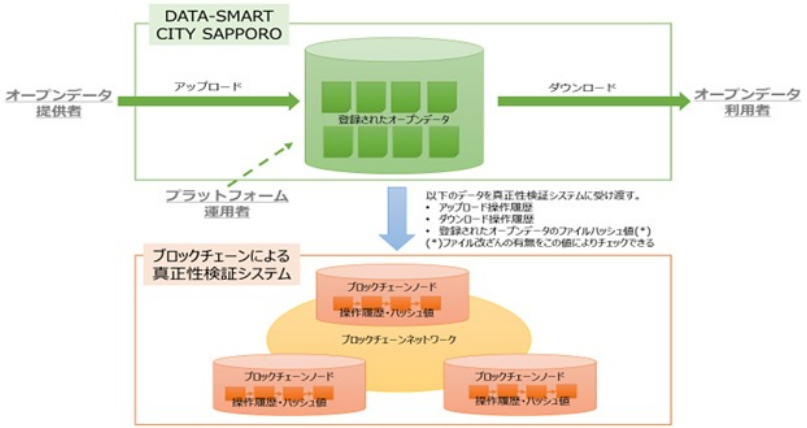
https://www.ntt-tx.co.jp/products/contractgate/pass.html

Monitor
ブロックチェーン上のデータ可視化ツール



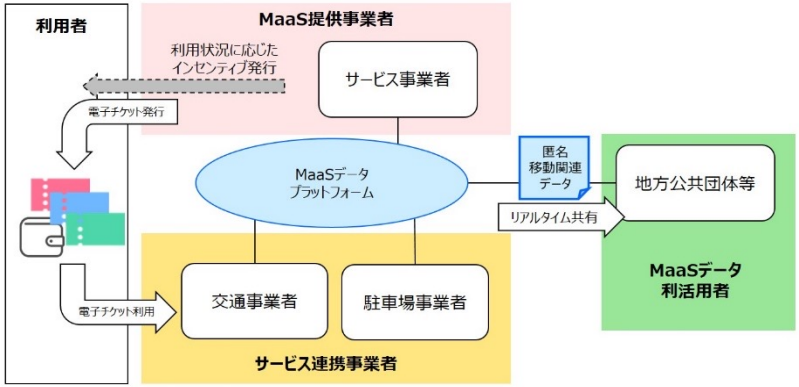
https://www.ntt-tx.co.jp/products/contractgate/monitor.html

2. 過去の実証実験



2018年1月から2018年3月に一般財団法人さっぽろ産業振興財団と札幌市が運営するプラットフォーム上におけるオープンデータに対してブロックチェーン技術を導入した実証実験を実施。

https://www.ntt-tx.co.jp/whatsnew/2018/181018.html



沖縄の交通系ICカードを活用し、あらゆる公共交通機関を一つにまとめて提供する沖縄版MaaSの実現に向け、ブロックチェーン技術により移動関連データを蓄積・共有・活用可能な「MaaSデータプラットフォーム」を開発し、那覇市/豊見城市で実施した実証実験にて、その有用性を確認。

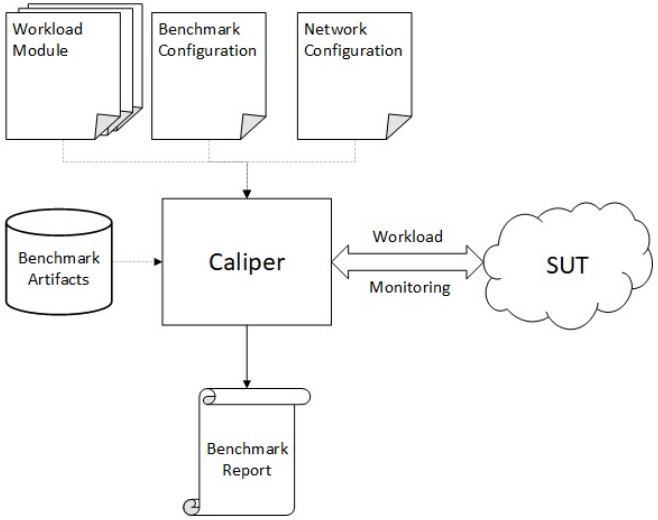
https://www.ntt-tx.co.jp/whatsnew/2021/210412.html

3. Hyperledger Caliper とは

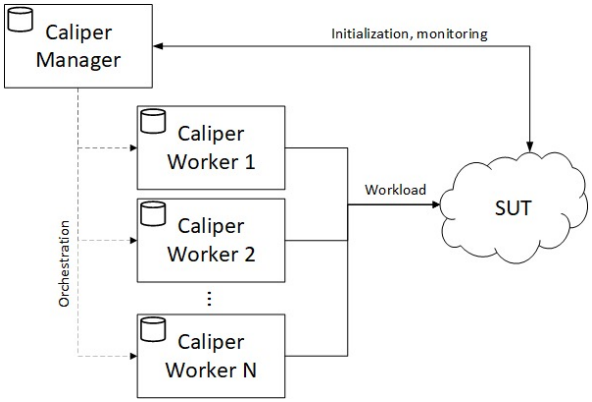
- Hyperledger コミュニティが提供しているブロックチェーンの性能測定ツール。
- ライセンスは Apache License Version 2.0を採用している。
- 事前定義されたユースケースを使用し、
  さまざまなブロックチェーンソリューションをテストし、結果を取得できる。
- 現在、以下のプロダクトに対応している。
  - Ethereum
  - Hyperledger Besu
  - Hyperledger Fabric (v1.x, v2.x)
  - FISCO BCOS
- パフォーマンスの指標としては、以下のような値が出力可能。
  - 成功率
  - トランザクションスループット
  - トランザクション実行待ち時間（最小、最大、平均）

3. Hyperledger Caliper とは

Caliper 全体概要図



複数ワーカーによる負荷発生



https://hyperledger.github.io/caliper/v0.4.2/architecture/

- テスト対象システム（SUT）に対してワークロードを生成し、その応答を継続的に監視
- 観測されたSUT応答に基づいてレポートを生成
- 複数のワーカープロセスによる負荷実施などにも対応している。

- Caliperは以下の構成部品を持つ。
  - ベンチマーク設定ファイル：ベンチマークの実行方法、SUTの監視設定。
  - ネットワーク構成ファイル：SUT側に対してのアクセス設定。
  - ワークロードモジュール：実行する負荷スクリプト(Node.jsモジュール)。
  - ベンチマークアーティファクト：ベンチマークを実行するために必要な部品。スマートコントラクトなど。

4. Hyperledger Caliper による測定

- 基本的には「ブロックチェーンプロダクトのインフラ部分」を測定するツール。
- 実証実験の要件定義フェーズにて
  BCプロダクト(Ethereum, HLF, Cordaなど)の選定が行われることがあるが、
  性能面というよりは機能面で選定が行われることが多い。
  - 顧客側としてはそこまで専門的な性能測定を求めてはいないため、
    プロダクト間の性能測定なども重点が置かれないことが多い。
- 自社独自の取り組みとしてHyplerledger Besuの機能調査を行うことになったため、
  併せて簡単なBesuの性能特性を測定してみることとなった。
- **ここでは、そのTry&Errorの流れについて、順を追って話していきたいと思います。**

4. Hyperledger Caliper による測定

(1) プロダクト別の差分測定

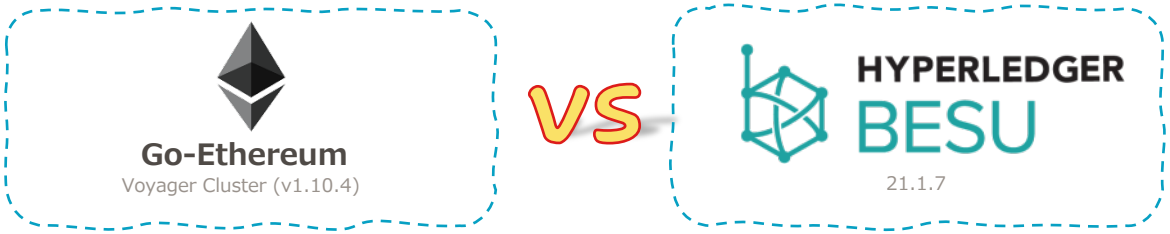
- Go-EthereumもBesuもSolidity言語によるスマートコントラクト実装となるため、共通のソースを用いて試験を行う。

```
pragma solidity >=0.4.22 <0.6.0;

contract simple {
    mapping(string => int) private accounts;

    function open(string memory acc_id, int amount) public {
        accounts[acc_id] = amount;
    }

    function query(string memory acc_id) public view returns (int amount) {
        amount = accounts[acc_id];
    }

    function transfer(string memory acc_from, string memory acc_to, int amount) public {
        accounts[acc_from] -= amount;
        accounts[acc_to] += amount;
    }
}
```

- ERC20をさらにシンプルにしたソースで口座番号と残高を記録するのみ。
  - open：連想配列型に対し、1か所の値を更新する。
  - query：連想配列型から値を1か所参照する。
  - transfer：連想配列型の2か所の値を更新する。

4. Hyperledger Caliper による測定

(1) プロダクト別の差分測定



◆ まずは小規模な環境を作った際に、性能に極端な劣化が発生していないかを確認する。

マシン環境

CPU: Core i7 7700U(3.6GHz)
vCPU: 2
Memory: 8GB
Storage: 40GB

ブロックチェーン設定

合意形成アルゴリズム： PoA (clique)
ブロック生成間隔: 5秒
ブロックガス上限：21,733,540
ノード数：1台

◆ Caliperから負荷発行を実施してみる。

負荷ツール設定

発行パターン：固定レート
目標TPS：
open：50TPS
query: 100TPS
transfter: 50TPS

```
$ npx caliper launch manager ¥
--caliper-bind-sut besu:latest ¥
--caliper-benchconfig benchmarks/scenario/simple/config.yaml ¥
--caliper-networkconfig networks/besu/1node-clique/networkconfig.json ¥
--caliper-workspace .
```

4. Hyperledger Caliper による測定

(1) プロダクト別の差分測定

Hyperledger Besu

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	1000	0	63.3	29.41	2.44	14.75	22.3
query	1000	0	100.2	0.03	0.00	0.00	100.2
transfer	1000	0	58.1	19.06	0.39	9.41	28.6

Go-Ethereum

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	1000	0	63.0	26.32	2.15	14.36	25.0
query	1000	0	100.1	0.02	0.00	0.00	100.1
transfer	1000	0	57.7	20.20	2.10	11.97	28.5

- このレベルの負荷では測定条件を合わせるとGo-EthereumもBesuもそれほど差はない。
- 当然ではあるが、トランザクション発行のない参照と更新での差は大きく出ている。(特にLatency、応答時間)
- ブロック生成タイミングなどにより、(従来のシステムよりも)測定結果に揺らぎがあるため、
  きちんとやるのであれば、複数回試行した上で集計が必要。

- 面白いのは、値を1か所しか更新しないopenが2か所の更新が走るTransferよりも「わずかに遅い」結果が出ている点。

  - 個人的見解では、すでにストレージスロットが確保されているmapping上の値2個を修正する方が、
    新たにストレージスロットを確保する処理よりも軽いためではないかと想定している。
  - このように直感的な「処理の重さ」と違う特性を示すことがあるため、注意が必要。

4. Hyperledger Caliper による測定

(2) 合意形成の差分測定

◆ 合意形成の設定のみ変更して性能の差分を取得する。

HYPERLEDGER BESU

ブロックチェーン設定

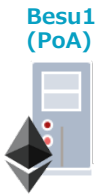
合意形成アルゴリズム： PoA (clique)
ブロック生成間隔: 5秒
ブロックガス上限：21,733,540
ノード数：1台

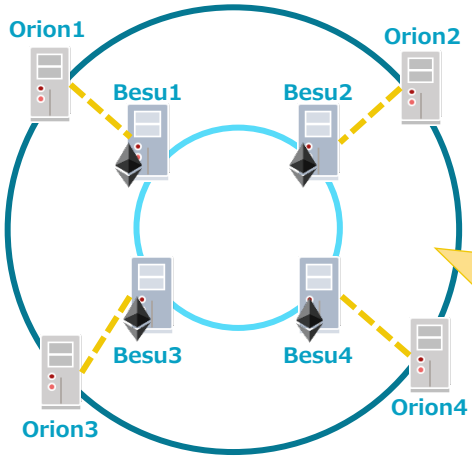
VS

HYPERLEDGER BESU

ブロックチェーン設定

合意形成アルゴリズム： IBFT
ブロック生成間隔: 5秒
ブロックガス上限：0x1fffffffffffff
ノード数：4台 (besu4+orion4)





一見、マシン(プロセス)の台数が異なるため、IBFT有利に見えるが、IBFTに於いて台数の増加は性能面では優位に働かない。各合意形成の最低限のノード台数ということで4台(セット)とした。

4. Hyperledger Caliper による測定

(2) 合意形成の差分測定

Hyperledger Besu (PoA)

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	1000	0	63.3	29.41	2.44	14.75	22.3
query	1000	0	100.2	0.03	0.00	0.00	100.2
transfer	1000	0	58.1	19.06	0.39	9.41	28.6

Hyperledger Besu (IBFT)

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	1000	0	190.5	15.53	7.20	11.75	74.9
query	1000	0	100.2	0.07	0.00	0.00	100.1
transfer	1000	0	91.9	14.13	4.56	9.24	54.2

- **IBFTの方が合意形成のオーバーヘッド分「重くなる」ことを想定していたが、結果が異なった。**

- 全体としてはIBFTの方がTPSが高いが、「Min Latency」はPoA＜IBFTとなっており想定どおり。
- 一方でPoAは「Max Latency」が高いTxが存在している。

- ガス上限の設定が異なっている事が原因で想定しない結果となってしまっている。
  - PoAはブロックガス上限21,733,540に対し、IBFTは377,777,777,777,777,777。
  - PoAでは負荷に対し、「ガス上限で詰まっている(ためMax Latencyが高い)」のに対し、IBFTではAvg Latencyとの差が少ない。

- ブロックチェーンプロダクトやシステムの評価では、
  ガス上限(ブロックサイズ)が系全体の処理性能に影響を与えるため、測定の際はきちんと値を合わせていく必要がある。

4. Hyperledger Caliper による測定

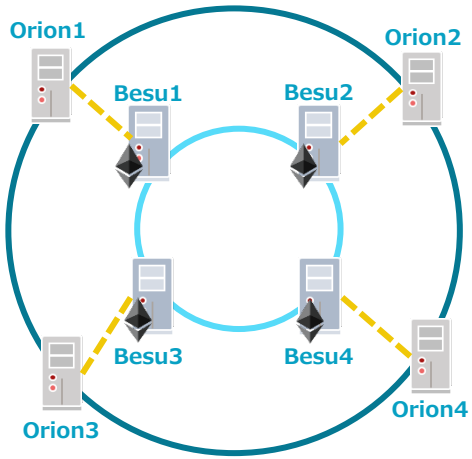
(3) ブロック生成間隔の差分測定

- 今度はブロック生成間隔のみ変更して性能の差分を取得する。

HYPERLEDGER BESU

ブロックチェーン設定

合意形成アルゴリズム： IBFT
ブロック生成間隔: **2秒**
ブロックガス上限：0x1fffffffffffff
ノード数：4台 (besu4+orion4)

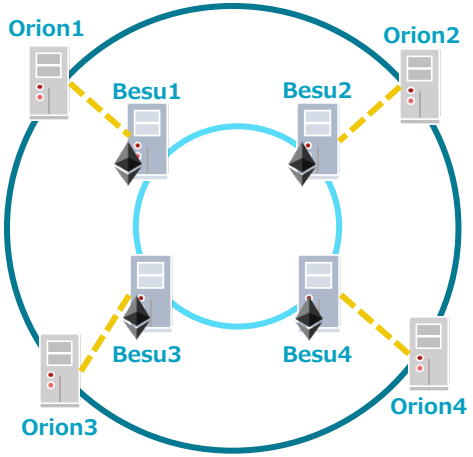


VS

HYPERLEDGER BESU

ブロックチェーン設定

合意形成アルゴリズム： IBFT
ブロック生成間隔: **5秒**
ブロックガス上限：0x1fffffffffffff
ノード数：4台 (besu4+orion4)



4. Hyperledger Caliper による測定

(3) ブロック生成間隔の差分測定

Hyperledger Besu (IBFT 5秒)

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	1000	0	190.5	15.53	7.20	11.75	74.9
query	1000	0	100.2	0.07	0.00	0.00	100.1
transfer	1000	0	91.9	14.13	4.56	9.24	54.2

Hyperledger Besu (IBFT 2秒)

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	1000	0	309.2	17.27	6.45	12.18	78.4
query	1000	0	100.2	0.15	0.00	0.00	100.2
transfer	1000	0	104.5	13.53	4.84	9.19	54.6

- **2秒の方が5秒よりも「かなり高速となる」ことを想定していたが、結果が異なった。**
- ブロック生成間隔というのは厳密には異なり、ブロック生成待ち時間の設定であるため、
  「トランザクションが限界まで詰まっている」負荷状況においては性能に与える影響が少ない。
  - ブロックサイズが十分である環境においては、
    IBFTに参加する(投票する)ノードの数の多さや規定の投票率に到達するまでの各ノードの応答時間が重要。
  - この検証ではネットワーク的に「とても近い」状況で測定を行っているため、
    4台ノード構成やブロック待ち時間などの影響が判断しにくい。
- ブロックチェーンプロダクトやシステムの評価では、
  系全体のネットワーク的分散状況が処理性能に影響を与えるため、
  測定の際は想定環境の検討を行い、可能な限り本番(想定)に近い構成でのテストを行う必要がある。

4. Hyperledger Caliper による測定

(4) ノード配置の差分測定

- 今度AWS上の複数のAZにノードを分散して性能の差分を取得する。

HYPERLEDGER BESU

ブロックチェーン設定

合意形成アルゴリズム： IBFT
ブロック生成間隔: 5秒
ブロックガス上限：0x1fffffffffffff
ノード数：4台 (besu4+orion4)

マシン環境

インスタンス: m5.large
vCPU: 2
Memory: 8GB
Storage: gp1

AWS

VS

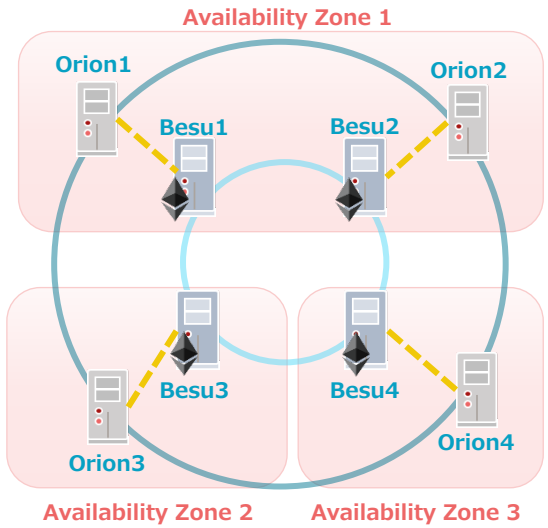
HYPERLEDGER BESU

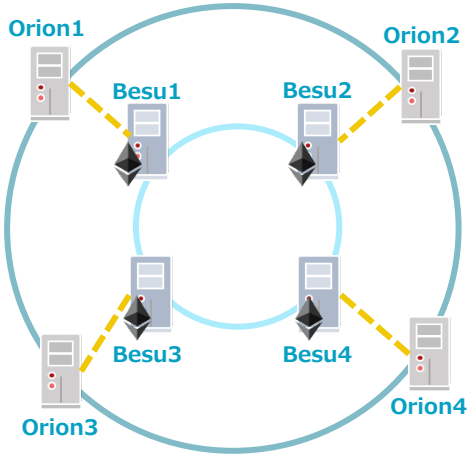
ブロックチェーン設定

合意形成アルゴリズム： IBFT
ブロック生成間隔: 10秒
ブロックガス上限：0x1fffffffffffff
ノード数：4台 (besu4+orion4)

マシン環境

CPU: Core i7 7700U(3.6GHz)
vCPU: 2
Memory: 8GB
Storage: 40GB





4. Hyperledger Caliper による測定

(4) ノード配置の差分測定

Hyperledger Besu (ローカル)

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	1000	0	309.2	17.27	6.45	12.18	78.4
query	1000	0	100.2	0.15	0.00	0.00	100.2
transfer	1000	0	104.5	13.53	4.84	9.19	54.6

Hyperledger Besu (AWS)

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
open	10000	0	111.9	14.99	3.69	9.00	101.5
query	10000	0	100.0	0.01	0.00	0.00	100.0
transfer	10000	0	110.1	14.82	2.97	8.99	101.5

- **AWSの方が「合意形成にかかるオーバーヘッド分遅くなる」ことを想定していたが、結果が異なった。**
- ローカルとクラウド環境の比較のため、インフラ部分での差分が大きすぎた。
  - ボトルネックとして想定していたAZ間通信はそもそもそれほどレイテンシが遅くなかった。
    (DCやAZ、時期によっても異なるが、2.5～5.0ms程度)
  - AZを分けたことにより、当然EC2インスタンスも分かれることとなり、結果的には性能が向上「してしまった」
- クラウド環境は各プラットフォームが注力して日々改善を行っているため、
  ローカル環境や過去の測定データと(公平に)比較することは難しい。(逆もまた然り)
  あくまで**「As Is」の値として理解し、実際の環境に合わせた環境(クラウド、オンプレ)で測定することが重要。**

5. まとめ

◆ Hyperledger Caliper

- ◆ 繰り返し行う負荷の測定が楽になるため、
  環境を変えつつ測定していくTry & Error が実行しやすい。
- ◆ 今後、多重実行や多様なトランザクション処理への対応など、
  機能が拡張されれば、より複雑な負荷条件が可能になるため、今後の機能拡充が望まれる。

◆ Hyperledger Besu

- ◆ 「プライベートネットワークのEthereumクライアント」として考えれば、
  Go-Ethereumより性能面で劣っているわけではない。
- ◆ 細かい設定値など、開発者視点で「気が利いている」ものが多い。
  (難易度固定、コントラクトコードサイズ設定など)
- ◆ その他、パーミッションドネットワークやプライバシーグループなど、
  Go-Ethereumでは実現されていなかった機能が拡充されており、
  EEA向け製品として利用しやすい。
- ◆ 本来、性能面での目玉機能はパブリック/プライベートTxの分割になるため、
  そこを踏まえた性能測定が実施できなかったのは心残りである。

- 今回検証に利用したスクリプトのサンプルは、
  参加メンバの手によりGithub上にも公開されております。

  https://github.com/hkiridera/caliper-benchmarks

- また、同様のテーマでのblog記事も公式HP上に公開しておりますのでご参照ください。

  NTTテクノクロス ホームページ コラム 情報畑でつかまえて「Hyperledger CaliperでBESUの性能測定」
  https://www.ntt-tx.co.jp/column/hyperledger_caliperbesu/210910/

ご清聴、ありがとうございました。