デジタルコンテンツ領域におけるブロックチェーンインフラ（Gethの研究結果に関する共有）

SingulaNet株式会社　町 浩二

スピーカー：町 浩二

2017年：メディア × ブロックチェーンの研究開発

2019年：OEMサービス 開始

論点整理

デジタルコンテンツ領域において必要なブロックチェーンの処理性能とは

一般的なアプリケーション向けに構築される分散ネットワークであれば、秒間2,000件程度で処理性能は充足する。
（数社から数十社のコンソーシアムネットワークをイメージ）

NFTの処理に必要な処理性能

前提条件	
アクティブユーザー数	**1,000万人**
アクティブ課金ユーザー数	100万人
同時利用ユーザー数	10万人
同時にMint/Transferを行うユーザー数	1万人

トランザクション処理能力が、

- 500件/秒であれば、処理待ち時間は　20秒
- **2,000件/秒であれば、処理待ち時間は　5秒**

デジタルコンテンツ領域において必要なブロックチェーンの処理性能とは

超大規模なネットワークの場合は、秒間10,000件程度の処理が必要になる可能性がある。
（パブリックネットワークをイメージ）

NFTの処理に必要な処理性能

	前提条件
アクティブユーザー数	**1億人**
アクティブ課金ユーザー数	1,000万人
同時利用ユーザー数	100万人
同時にMint/Transferを行うユーザー数	10万人

トランザクション処理能力が、

- 500件/秒であれば、処理待ち時間は　200秒
- 2,000件/秒であれば、処理待ち時間は 50秒
- **10,000件/秒であれば、処理待ち時間は 10秒**

論点

目的特化型コンソーシアムネットワークの目標値：2,000件/秒

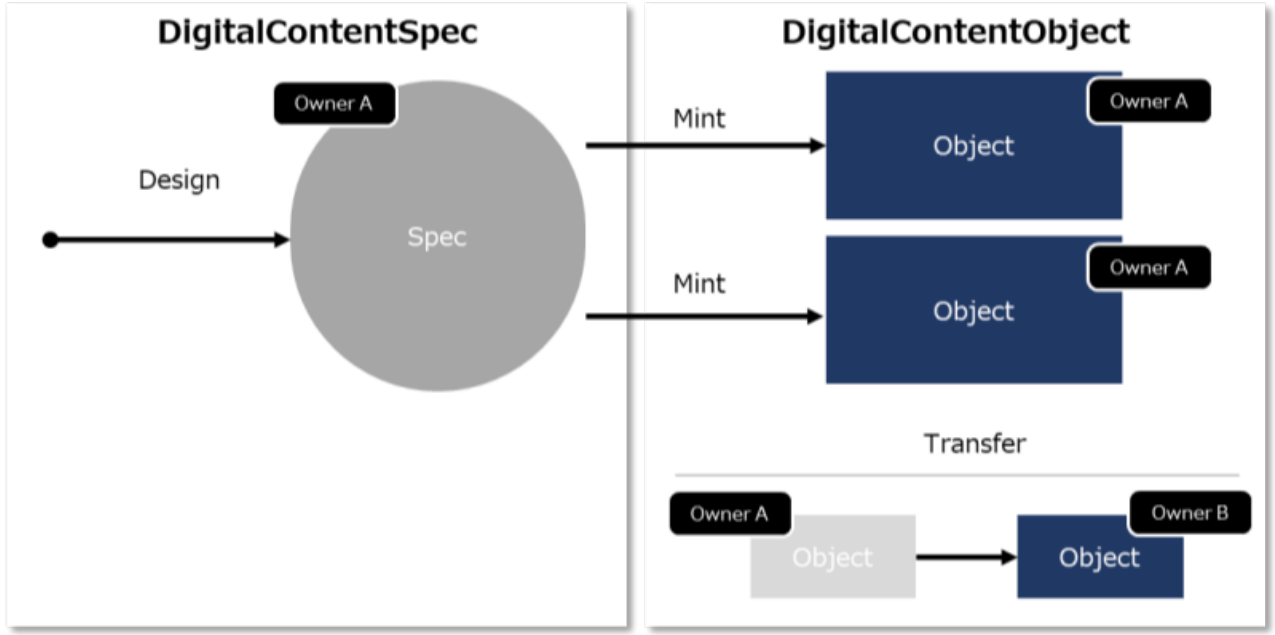
パブリックネットワークの目標値：10,000件/秒

日本発のコンテンツ業界向けブロックチェーン

著作権流通などのコンテンツ課題の解決にフォーカスしたブロックチェーンの研究開発を目的としたコンソーシアムにて新たな取り組みを始めています。Ethereumを利用した研究開発を行っており、近日中に大きな発表があるかも。

Content-NFT

著作権流通に対応した次世代NFTのスマートコントラクトのリファレンス実装を公開しています。



研究：原因追及

本日シェアする研究結果：2020年の取り組み

White Paper
- **Function-Level Bottleneck Analysis of Private Proof-of-Authority Ethereum Blockchain**

Members
- KENTAROH TOYODA (Member, IEEE)
- KOJI MACHI
- YUTAKA OHTAKE
- ALLAN N. ZHANG(Associate Member, IEEE)

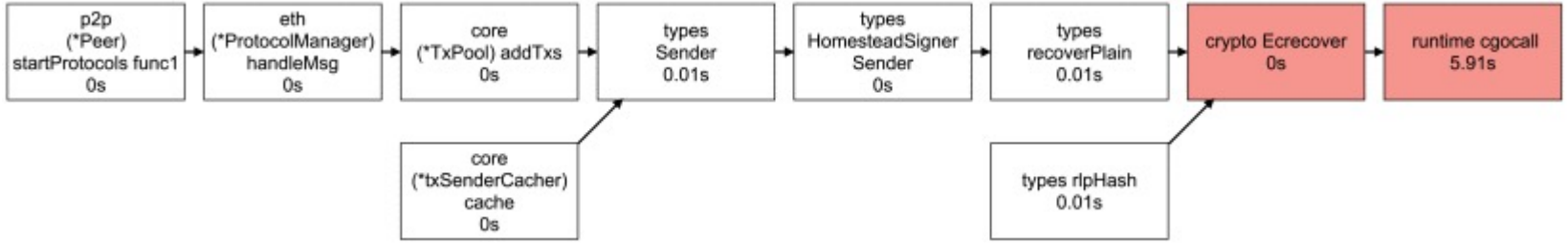
課題認識
プライベートイーサリアムブロックチェーンベースのシステムは、多くの産業部門で要求されている。
しかし、これらのシステムのスループットパフォーマンスは十分ではない。
多くの研究者がプライベートブロックチェーンのパフォーマンスを分析したが、これらの研究では根本的な原因を分析していない。

- 本研究では、プライベートイーサリアムブロックチェーンの**詳細な機能レベルのボトルネック分析**を実行した。
- イーサリアムクライアントアプリケーションのGethはgo言語を使用して開発されているため、gol言語のリソースプロファイリングツールであるpprofとカスタムgolang関数を利用して、関数にかかる時間を測定した
- パラメーターを簡単に構成してテストを実行するために、Dockerコンテナーを使用してプライベートEthereumブロックチェーンの構築プロセスを自動化するシェルスクリプトを作成した。
- 一連の実験を実施し、①**トランザクションがイーサリアムノードに到着するたびに呼び出されるボトルネック関数を特定**した。
- さらに、②**マルチスレッドが十分に活用されていない**こともわかった。

①ボトルネック関数の特定

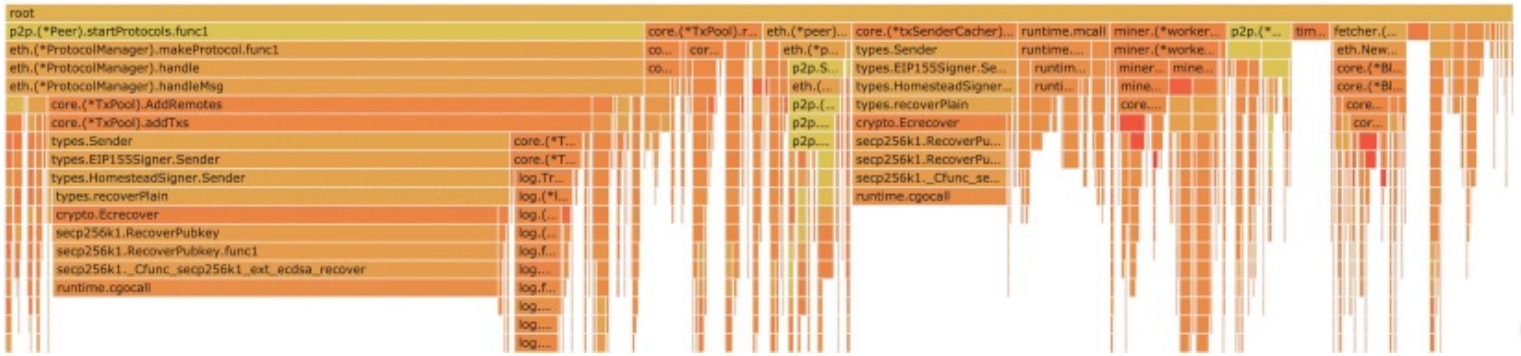
本研究で特定したPoAアルゴリズムのボトルネック処理
（実際はPoAアルゴリズムの問題ではなく、Geth全体に共通した問題だった）

cgocall … 暗号化/複合化関連の処理で、Go言語の一般関数（C言語を呼びだす処理）を使用している
go言語からC言語を呼び出すための処理は、Go言語間のIF処理またはC言語間のIF処理よりも遥かに重たい



スタックオーバーフローの*cgocall*に関する公開*QA*では・・・
“As you've discovered, there is fairly high overhead in calling C/C++ code via CGo. So in general, you are best off trying to minimize the number of CGo calls you make. “
「ご存知のとおり、*CGo*を介して*C / C ++*コードを呼び出すとかなり高いオーバーヘッドが発生します。したがって、一般的には、*CGo*呼び出しの数を最小限に抑えることをお勧めします。」

出典：*https://stackoverflow.com/questions/28272285/why-cgos-performance-is-so-slow-is-there-something-wrong-with-my-testing-code*



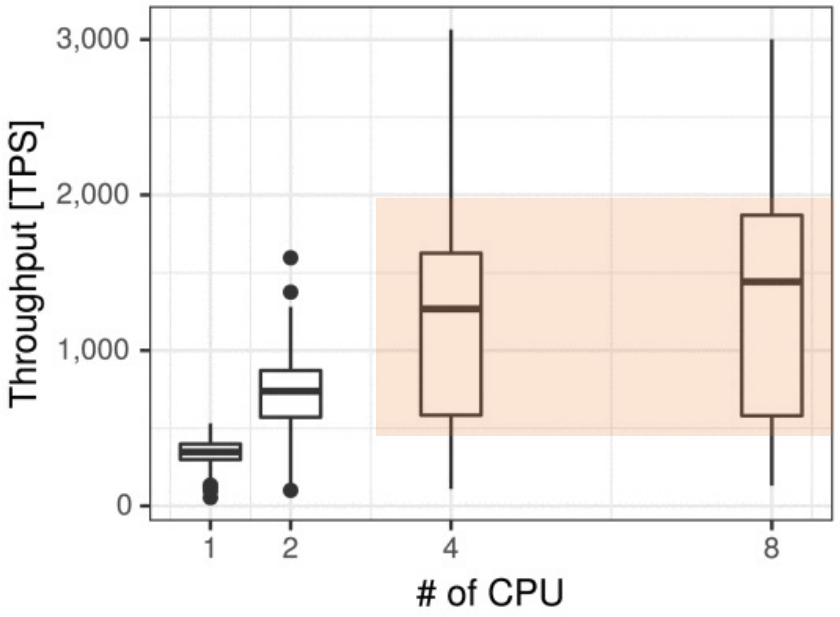
全てをGo言語で書く？

全てC言語で書く？

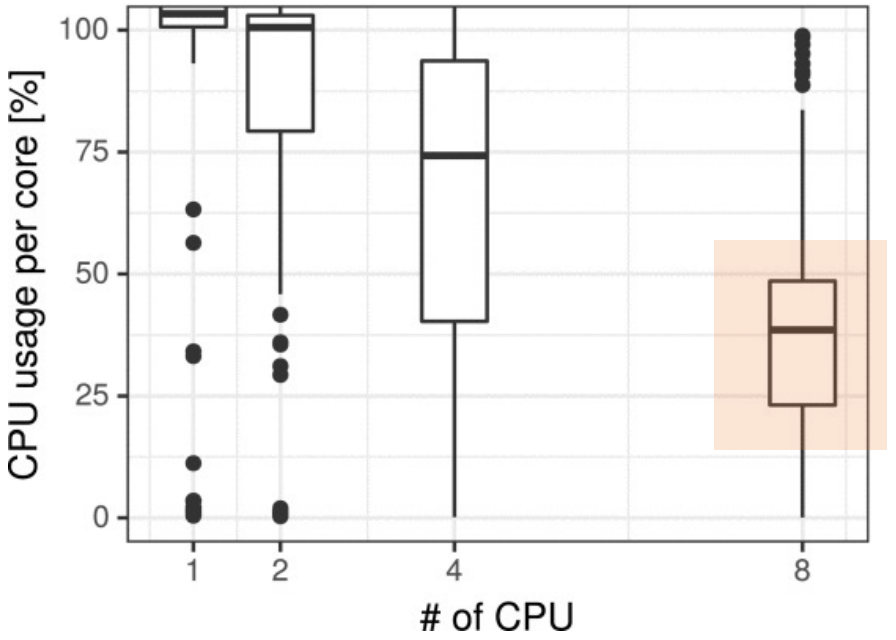
②マルチスレッドが十分に活用できていない

マルチスレッド処理は4CPUの並行以上では頭打ち

Gethの基本処理に何等かの待機時間が発生しているなどの理由でCPU分散処理を有効に活用できていない



4CPUと８CPUで処理件数に大きな差はない



８CPUではCPU稼働率が著しく低下している

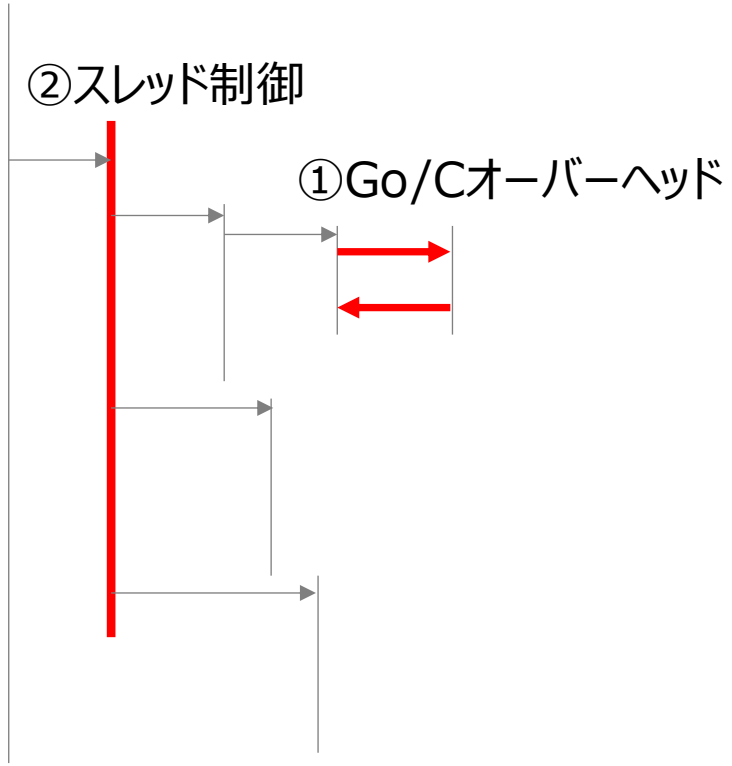
Goのメイン処理における
スレッド制御を改善？

結論

- **Ethereum Gethのパフォーマンスは、スレッド制御機構とGo/C言語間のインターフェース処理がボトルネックとなっている**
- **秒間2,000件を超える処理性能の向上には、汎用的な処理部分における改修が必要（コンセンサスアルゴリズムの話ではない）**

デジタルコンテンツ領域のパブリックネットワーク（秒間10,000件）の実現に向けた処理性能のボトルネック

CPUのコア数を増やした場合に限界までスレッドを並行起動できるようにする



暗号化/複合化などの処理で使用している言語間のI/F処理のオーバーヘッドを軽減する

御清聴ありがとうございました