令和３年１２月２０日判決言渡

令和３年（行ケ）第１００６０号　審決取消請求事件

口頭弁論終結日　令和３年１１月１１日

判　　決

原　　告　　株式会社三菱ＵＦＪ銀行

同訴訟代理人弁護士　　高　　橋　　雄　一　郎

同　　阿　　部　　実　佑　季

同訴訟代理人弁理士　　林　　佳　　輔

同　　荒　　尾　　達　　也

被　　告　　特　　許　　庁　　長　　官

同指定代理人　　須　　田　　勝　　巳

同　　田　　中　　秀　　人

同　　梶　　尾　　誠　　哉

同　　金　　子　　秀　　彦

主　　文

１　原告の請求を棄却する。

２　訴訟費用は原告の負担とする。

事 実 及 び 理 由

第１　請求

特許庁が不服２０２０－１２５４３号事件について令和３年３月１９日にした審決を取り消す。

第２　事案の概要

本件は，特許拒絶査定の不服審判請求を不成立とした審決の取消訴訟である。

1　特許庁における手続の経緯等（当事者間に争いがない。）

⑴　原告は，令和元年７月９日，名称を「システムおよび処理方法」とする発明について，特許出願（特願２０１９－１２７８９４号。以下「本件出願」という。）をしたが，令和２年６月２２日付けの拒絶査定を受けた。

⑵　原告は，令和２年９月８日，同日付け手続補正書を提出するとともに（以下，この手続補正書による補正を「本件補正」という。），拒絶査定不服審判を請求した。

特許庁は，上記請求を不服２０２０－１２５４３号事件として審理を行い，令和３年３月１９日，本件補正を却下した上で，「本件審判の請求は，成り立たない。」との審決（以下「本件審決」という。）をし，その謄本は，同年４月６日，原告に送達された

⑶　原告は，令和３年４月２８日，本件審決の取消しを求める本件訴訟を提起した。

2　特許請求の範囲の記載

本件補正前の特許請求の範囲の記載は，請求項１ないし１０からなり，その請求項の記載は下記⑴のとおりであり（以下，請求項１に係る発明を「本願発明」という。），本件補正後の特許請求の範囲の記載は，請求項１ないし８からなり（特許請求の範囲は全文変更），その請求項の記載は下記⑵のとおりである（以下，本件補正後の請求項１に係る発明を「本件補正発明」という。下線部は，本件補正による補正箇所である。）。

⑴　本件補正前の特許請求の範囲の記載

ア　請求項１（本願発明）

台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスを生成する生成部と，

トランザクションの指示を受け付け，前記複数のプロセスのいずれかに当該トランザクションのリクエスト送信を割り当てる割当部と，を備える

システム。

イ　請求項２

前記複数のプロセスは，第１プロセスおよび第２プロセスを含み，

前記第１プロセスが前記リクエストを送信するノードは，前記第２プロセスが前記リクエストを送信するノードと同一である，請求項１に記載のシステム。

ウ　請求項３

前記割当部は，ラウンドロビン方式により前記トランザクションのリクエスト送信を割り当てる，請求項１または請求項２に記載のシステム。

エ　請求項４

前記複数のノードで形成されるネットワークにおける所定のトランザクションのリクエストに対する平均スループットとプロセス多重度との関係において，プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，

前記複数のプロセスの数は，前記第２値よりも小さい数で設定される，請求項１から請求項３のいずれかに記載のシステム。

オ　請求項５

前記複数のノードで形成されるネットワークは，コンソーシアム型である，請求項１から請求項４のいずれかに記載のシステム。

カ　請求項６

コンピュータが実行する処理方法であって，

トランザクションの指示を受け付け，

台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスのいずれかに，前記指示に基づくトランザクションのリクエスト送信を割り当てる，処理方法。

キ　請求項７

前記複数のプロセスは，第１プロセスおよび第２プロセスを含み，

前記第１プロセスが前記リクエストを送信するノードは，前記第２プロセスが前記リクエストを送信するノードと同一である，請求項６に記載の処理方法。

ク　請求項８

前記トランザクションのリクエスト送信を割り当てるときには，ラウンドロビン方式を用いる，請求項６または請求項７に記載の処理方法。

ケ　請求項９

前記複数のノードで形成されるネットワークにおける所定のトランザクションのリクエストに対する平均スループットとプロセス多重度との関係において，プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，

前記複数のプロセスの数は，前記第２値よりも小さい数で設定される，請求項６から請求項８のいずれかに記載の処理方法。

コ　請求項１０

前記複数のノードで形成されるネットワークは，コンソーシアム型である，請求項６から請求項９のいずれかに記載の処理方法。

⑵　本件補正後の特許請求の範囲の記載

ア　請求項１（本件補正発明）

管理主体が存在しないパブリック型ネットワークにおいて台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスであって，設定されるプロセス多重度に応じた複数のプロセスを生成する生成部と，

トランザクションの指示を受け付け，前記複数のプロセスのいずれかに当

該トランザクションのリクエスト送信を割り当てる割当部と，を備えるシステム。

イ　請求項２

前記複数のプロセスは，第１プロセスおよび第２プロセスを含み，

前記第１プロセスが前記リクエストを送信するノードは，前記第２プロセスが前記リクエストを送信するノードと同一である，請求項１に記載のシステム。

ウ　請求項３

前記割当部は，ラウンドロビン方式により前記トランザクションのリクエスト送信を割り当てる，請求項１または請求項２に記載のシステム。

エ　請求項４

前記複数のノードで形成されるネットワークにおける所定のトランザクションのリクエストに対する平均スループットと前記プロセス多重度との関係において，当該プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，

前記複数のプロセスの数は，前記第２値よりも小さい数で設定される，請求項１から請求項３のいずれかに記載のシステム。

オ　請求項５

コンピュータが実行する処理方法であって，

トランザクションの指示を受け付け，

管理主体が存在しないパブリック型ネットワークにおいて台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスであって，設定されるプロセス多重度に応じた複数のプロセスのいずれかに，前記指示に基づくトランザクショ

ンのリクエスト送信を割り当てる，処理方法。

カ　請求項６

前記複数のプロセスは，第１プロセスおよび第２プロセスを含み，

前記第１プロセスが前記リクエストを送信するノードは，前記第２プロセスが前記リクエストを送信するノードと同一である，請求項５に記載の処理方法。

キ　請求項７

前記トランザクションのリクエスト送信を割り当てるときには，ラウンドロビン方式を用いる，請求項５または請求項６に記載の処理方法。

ク　請求項８

前記複数のノードで形成されるネットワークにおける所定のトランザクションのリクエストに対する平均スループットと前記プロセス多重度との関係において，当該プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，

前記複数のプロセスの数は，前記第２値よりも小さい数で設定される，請求項５から請求項７のいずれかに記載の処理方法。

３　本件審決の理由の要旨

本件審決は，本件補正発明は甲第１号証「佐藤竜也ほか，ブロックチェーン基盤Hyperledger Fabricの性能評価と課題整理，電子情報通信学会技術研究報告，一般社団法人電子情報通信学会，２０１７年２月２４日，第１１６巻，第４９１号，ｐ．１６７－１７４」（以下「引用文献１」という。）に記載の発明（以下「引用発明」という。）と甲第２号証「特開２０１９－１４１３５号公報」（平成３１年１月３１日出願公開。以下「引用文献２」という。）及び甲第３号証「特開２０１０－２７８６３９号公報」（以下「引用文献３」という。）に記

載の周知技術に基づいて当業者が容易に発明することができたものであるから，本件補正は独立特許要件（特許法１７条の２第６項で準用する同法１２６条７項）を満たさないので却下すべきものであり（同法１５９条１項の規定において読み替えて準用する同法５３条１項），本願発明も引用発明と引用文献２及び３に記載の周知技術に基づいて当業者が容易に発明できたものであるから，本件出願は拒絶すべきものと判断した（以下，本件出願に係る願書に添付した明細書を，図面を含めて「本願明細書」という。別紙１参照）。その判断の要旨は以下のとおりである。

⑴　本件補正発明について

ア　引用発明の認定

ブロックチェーン（Blockchain, BC）基盤のＯＳＳプロジェクトであるHyperledgerの基盤実装の一つであるFabricについて性能を中心とした評価を行うものであって，

Fabricは，コンソーシアムあるいはプライベート型での利用を想定したＢＣ基盤であり，

マルチホスト上にまたがった環境上にFabricによるＢＣネットワークを構築し，その上で性能測定を行うものであって，

クライアントからＲＥＳＴ経由でＢＣネットワークにアクセスし，チェーンコードを実行して負荷をかけることで性能測定を行うものであり，

負荷をかける際には，複数のクライアントからの同時アクセスを模擬するため，マルチスレッドでトランザクションを並列実行するものであって，

クライアントによる負荷生成ツールプログラムの処理の流れは，スレッド毎に実行トランザクション（invoke）を指定した回数繰り返す（並列実行）ことを含むものであり，

スループット測定結果は，並列スレッド数を増やしていくとスループットも増加する傾向にあるが，ある程度並列度を増やすとスループットは頭

打ちとなり，スループットが頭打ちになった後も，それ以上に並列度を増やしていくと，挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得る，

ＢＣ基盤Hyperledger Fabricの性能評価。

イ　本件補正発明と引用発明との一致点

ネットワークにおいて台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数の処理単位であって，複数の処理単位を生成する生成部と，

前記複数の処理単位のいずれかに当該トランザクションのリクエスト送信を割り当てる割当部と，を備えるシステム。

ウ　本件補正発明と引用発明との相違点

（ア）　相違点１

本件補正発明は,「管理主体が存在しないパブリック型」ネットワークであるのに対して，引用発明の，ブロックチェーン基盤実装の一つである「Fabric」は,「コンソーシアムあるいはプライベート型での利用を想定したＢＣ基盤」である点。

（イ）　相違点２

本件補正発明は，処理単位が「プロセス」であって，生成部が複数の「プロセス」を生成し，割当部が前記複数の「プロセス」のいずれかにトランザクションのリクエスト送信を割り当てるものであるのに対して，引用発明は，処理単位が「スレッド」である点。

（ウ）　相違点３

本件補正発明は，生成部が「設定されるプロセス多重度に応じた」複数のプロセスを生成するものであるのに対して，引用発明は，そのような特定がなされていない点。

（エ）　相違点４

本件補正発明は，割当部が「トランザクションの指示を受け付け」，複数のプロセスのいずれかにトランザクションのリクエスト送信を割り当てるのに対して，引用発明は，そのような特定がなされていない点。

エ　相違点の容易想到性

（ア）　相違点１

引用発明は，コンソーシアムあるいはプライベート型での利用を想定しているとはいえ，引用発明を管理主体が存在しないパブリック型のブロックチェーンには適用できないとする技術的根拠は何ら認められないところ，引用発明を管理主体が存在しないパブリック型のネットワークに適用することには何ら技術的創意は見出せず，当業者であれば適宜実施し得る事項にすぎない。

（イ）　相違点２

引用発明において，マルチプロセスを採用して処理単位をプロセスとすることは，引用文献２及び３に示される周知技術に基づいて当業者が適宜なし得るものであり，さらに，甲第４号証「特開２０１５－８８１７６号公報」（以下「甲４文献」という。）に示されるように，プログラムを実行する単位である複数のプロセスを生成し，クライアント端末から受け取ったリクエストを，前記複数のプロセスのうちの何れかのプロセスに割り当てることが，複数のプロセスを用いたプログラム実行における極めて一般的な処理であることも踏まえれば，引用発明にマルチプロセスを採用した際に，生成部が複数の「プロセス」を生成し，割当部が前記複数の「プロセス」のいずれかにトランザクションのリクエスト送信を割り当てるとすることも，当業者であれば当然になし得るものと認められる。

（ウ）　相違点３

引用発明では，「スループットが頭打ちになった後も，それ以上に並

列度を増やしていくと，挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得る」としており，これはフロント側でリクエスト送信を割り当てるスレッドの数を制限することを示唆するものであって，スレッドの多重度を制限することを示唆するものである。

上記(イ)のとおり，引用発明において，マルチスレッドに換えて「マルチプロセス」を採用することは，当業者であれば適宜なし得る事項にすぎないと認められるところ,かかるマルチプロセスを採用した場合に,プロセス多重度を制限するため「プロセス多重度に応じた」複数のプロセスを生成することは，リクエスト送信を割り当てるスレッドの数を制限するという上記示唆に基づいて，当業者であれば容易に想到し得るものである。

(エ)　相違点４

上記(イ)のとおり，引用発明において，マルチプロセスを採用することは当業者が適宜なし得る事項と認められるところ，引用発明の「ＢＣ基盤　Hyperledger　Fabric」を実際のトランザクション処理に用いる場合,「トランザクションの指示を受け付け」て，複数のプロセスのいずれかにトランザクションのリクエスト送信を割り当てることは，当業者であれば当然になし得るものである。

⑵　本願発明について

ア　本願発明と引用発明との一致点

台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数の処理単位を生成する生成部と，

前記複数の処理単位のいずれかに当該トランザクションのリクエスト送信を割り当てる割当部と，を備えるシステム。

イ　本願発明と引用発明との相違点

（ア） 相違点ａ（相違点２と同じ）

本願発明は，処理単位が「プロセス」であって，生成部が複数の「プロセス」を生成し，割当部が前記複数の「プロセス」のいずれかにトランザクションのリクエスト送信を割り当てるものであるのに対して，引用発明は，処理単位が「スレッド」である点。

（イ） 相違点ｂ（相違点４と同じ）

本願発明は，割当部が「トランザクションの指示を受け付け」，複数のプロセスのいずれかにトランザクションのリクエスト送信を割り当てるのに対して，引用発明は，そのような特定がなされていない点。

ウ 相違点の容易想到性

前記⑴エ(イ)及び(エ)のとおり，相違点ａ及び相違点ｂは，いずれも格別のものではない。

４ 取消事由

本件補正発明の進歩性に関する判断（相違点３の容易想到性の有無）の誤り

第３ 当事者の主張

１ 原告

⑴ 相違点３の容易想到性の有無について

ア 引用文献１の記載事項に関して

（ア） 本件審決は,「引用発明では,『スループットが頭打ちになった後も，それ以上に並列度を増やしていくと，挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得る』としており，これはフロント側でリクエスト送信を割り当てるスレッドの数を制限することを示唆するものであって，スレッドの多重度を制限することを示唆するものである。」（１９頁１６ないし２１行目）とするが，誤りであり，その結果，相違点３に係る容易想到性の判断にも誤りがある。

引用文献１の「ＢＣサービス：Ｐ２Ｐプロトコル，分散台帳，コンセンサスマネージャといった要素によって構成される。Ｐ２Ｐプロトコルにより，Ｐ２Ｐでの双方向ストリーミング，フロー制御，リクエストの多重化といった機能を提供する。既存ネットワークと連携して動作する。」（１６８頁右欄３６行ないし１６９頁左欄４行目）との記載によると，引用文献１では，「フロー制御」と「リクエストの多重化」とは異なる機能としており，この記載から想定される構成は別紙６「参考図１」のとおりである。同図に示されるように，上記「リクエストの多重化」は，１つのプロセスに流入するリクエストを複数のスレッドで多重化することによってＢＣネットワークに出力するものであり，その１つのプロセスに流入するリクエストの流量（フロー）を制御するのが上記「フロー制御」である。したがって，引用発明において，「挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策」をとるのは，「フロー制御」（流量制御）であり，「リクエストの多重化」ではない。

ここで，挙動の不安定を回避するための「フロー制御」は，技術常識上，「流量制御」と同じであるから（甲２１ないし２５），「リクエストの流量制御」は，技術常識上，①キュー長の調整又は②リクエスト送信頻度の低下のいずれかである（甲１９）。

したがって，引用文献１に「挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得る」旨の記載があるからといって，引用文献１に「スレッドの数を制限することを示唆する」記載があるとはいえず，「スレッドの多重度を制限することを示唆する」記載があるともいえない。

（イ） 被告は，引用文献１における「リクエストの流量制御」とは，単位時間当たりの「リクエスト総数」，すなわち，「スレッド当たりのリクエ

スト数×スレッド数」を制御するものであって,「流量制御」を行うために，スレッド数とリクエスト数の双方が制御の対象となっている旨主張するが，引用文献１には,「リクエストの流量制御」が単位時間当たりのリクエスト総数の制御であるとの記載はないし，仮に,「リクエスト流量制御」をすることが単位時間当たりの「リクエスト総数」を制御することであったとしても，それが「スレッド当たりのリクエスト数×スレッド数」を制御することに等しいということを引用文献１の開示事項から読み取ることはできない。

また，引用文献１においては,「負荷が大きすぎること」，すなわち「単位時間当たりのリクエスト数が大きすぎること」しか課題として認識できておらず,「並列度が高すぎること」を課題として認識しているのではないから，上記課題を認識するための手段としてスレッドの数を増加させてみた測定結果にすぎない記載をもって,「スレッド数（並列度）の制御」を,「リクエストの流量制御」における課題解決手段として用いることができると読み取ることはできない。

イ　スレッドとプロセスの置換に関して

（ア）　被告は，生成部においてプロセス多重度を制限することは本件補正発明の発明特定事項に基づくものではない旨主張するが，本願明細書には,「閾値Ｔｈｎよりも小さいプロセス多重度をプロセス１１０の数としてプロセス生成装置１１に設定する」ことにより,「ブロックチェーンネットワーク５における演算処理能力にボトルネックを発生させずに，リクエストの送信側においてボトルネックを発生させることでき」,「プロセス生成装置１１において不要なプロセス１１０を生成しないようにして，リクエスト制御システム１のリソースを効率的に用いることができる」（以上，本願明細書【００４６】，図１）との記載があるから,「設定されるプロセス多重度に応じた複数のプロセスを生成する生成部」とい

う発明特定事項によれば,「リクエストの多重化を実現するプロセス数を必要な数に制御すること」や「不要なプロセスを生成せずにリソースを効率的に用いること」といった具体的な効果まで発明特定事項において特定されなくとも，本件補正発明がプロセス多重度を制限する構成として特定されたものといえるのは明らかである。

（イ） 本件審決は,「マルチプロセスを採用した場合に，プロセスの多重度を制限するため『プロセスの多重度に応じた』複数のプロセスを生成することは，リクエスト送信を割り当てるスレッドの数を制限するという上記示唆に基づいて，当業者であれば容易に想到し得るものである。」（１９頁２４ないし２８行目）とするが，誤りである。

仮に，引用文献１に「スレッドの数を制限することを示唆するのであって，スレッドの多重度を制限することを示唆する」旨の記載があるとしても，マルチプロセスではない引用発明からは，１つのプロセスにおいてスレッドの多重度を制限することが示唆されるだけであって，プロセス多重度を制限することまで示唆されることはない。

また，プロセスは１個のメモリ空間が割り当てられた実行プログラムであるのに対して，スレッドはプロセス内に所在してＣＰＵコアに対する命令を実行する単位をいい，この両者はハードウェア資源の利用態様が相違するため，これらを相互に置換することはできない。すなわち，別紙６「参考図１」に示すように，引用発明は１つのプロセスにおいてマルチスレッドを実現するものであるから，プロセスがスレッドの多重度の制限をするが，一方，別紙７「参考図２」に示すように，本件補正発明におけるマルチプロセスの多重度は，各プロセスの生成を制御する生成部が制限する。このように，引用発明におけるスレッドとプロセスとの関係は，本件補正発明におけるプロセスと生成部との関係に対応する。参考図１のような引用発明におけるスレッドとプロセスとの関係を，

参考図２のような本件補正発明におけるプロセスと生成部との関係に置換することが容易想到であるという根拠は存在しない。

ウ　顕著な効果に関して

本願明細書の記載（【００４６】，図１）によると，本件補正発明は，プロセス生成装置において不要なプロセスを生成しないようにして，リクエスト制御システムのリソースを効率的に用いることができるとの顕著な効果を奏する。

一方，引用発明においては，別紙６「参考図１」に示すように，リクエストの流量制御がリクエストの多重化の前段階のフロー制御で行われ，スレッドの数は変わることがなく，不要なスレッドが残ったままであるから，リソースを効率的に用いることができない。

エ　まとめ

以上のとおり，相違点３を容易想到と判断した本件審決の判断には誤りがある。

⑵　小括

前記⑴からすれば，本件補正が独立特許要件を充足しないとした本件審決の判断には誤りがあり，本件補正を却下して本願発明が進歩性を欠如するとして本件出願を拒絶すべきものと判断した本件審決の判断にも，誤りがある。

２　被告

⑴　相違点３の容易想到性の有無について

ア　引用文献１の記載事項に関して

（ア）　引用文献１には，スループットの計算方法を，「全スレッドによる合計リクエスト数／（全レスポンスが返ってきた時刻－リクエスト処理を開始した時刻）」とし，１スレッドあたりのリクエスト数を１０００あるいは２００に固定して（１７１頁左欄２４行ないし右欄７行目，１７１頁表３），スレッド数を変えながら測定した結果から，「ある程度並列度

を増やすとスループットは頭打ちになった.」,「スループットが頭打ちになった後も，それ以上に並列度を増やしていくと，内部的にエラーが発生する等，安定稼働が困難となる場合が見受けられた.」との評価が行われたことが開示されている（１７１頁右欄２７ないし３９行目，１７２頁図３及び１７２頁図４）。

そうすると，引用文献１における，「高負荷を与えた場合には，挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得る。」（１７１頁右欄２７ないし３９行目）との記載における「リクエストの流量制御」とは，単位時間当たりの「リクエスト総数」，すなわち，「スレッド当たりのリクエスト数×スレッド数」を制御するものであって，「流量制御」を行うために，スレッド数とリクエスト数の双方が制御の対象となっていると解されるから，スレッド数を制限することも示唆されているというべきである。

さらに，引用文献１には「２．３　本研究の課題　ＢＣの実適用に向けては，ＢＣ基盤およびその実装における現時点での技術課題の明確化が必要である．特に，金融業務への適用に向けては，一般的にトランザクションのスループットが性能面の最重要指標である.」（１６９頁左欄２９ないし３３行目），「図３にセキュリティ機能ＯＦＦ時の，図４にセキュリティ機能ＯＮ時のinvoke／queryスループット測定結果を示す．図に示す通り，セキュリティ機能ＯＦＦ／ＯＮ時ともに，invokeとqueryの両方で，並列スレッド数を増やしていくとスループットも増加する傾向にあった．ただし，ある程度並列度を増やすとスループットは頭打ちとなった．また，スループットが頭打ちになった後も，それ以上に並列度を増やしていくと，内部的にエラーが発生する等，安定稼働が困難となる場合が見受けられた．つまり，高負荷を与えた場合には，挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行

う等の対策が必要となり得る.」（１７１頁右欄２７ないし３９行目）との記載があるところ，これらの記載から，引用文献１には，トランザクションのスループットが性能面の最重要指標であり,並列スレッド数(並列度）を増やしていくとスループットが増加する傾向にあるが，ある程度並列度を増やしていくとスループットは頭打ちになり，さらに並列度を増やしていくと，高負荷が与えられて挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得ることが開示されているといえる。

そうすると，引用文献１には，並列度を増加させると高負荷となるので，この高負荷を抑制するために流量制御を行う必要があるところ，この流量制御を行うための一つの手段として並列度を制限することが示唆されている。そして，上記流量制御を行うためには,「並列度」を増やしていって「挙動が安定しなくなる」手前の「並列度」，つまり,「挙動が不安定にならない最大の並列度」(適切な並列度）に制限する必要があることが理解できる。

したがって，引用文献１には，並列スレッド数を制限することを示唆する記載があるといえる。

（イ） 原告が引用する引用文献１の「ＢＣサービス：Ｐ２Ｐプロトコル，分散台帳，コンセンサスマネージャといった要素によって構成される．Ｐ２Ｐプロトコルにより，Ｐ２Ｐでの双方向ストリーミング，フロー制御，リクエストの多重化といった機能を提供する．既存ネットワークと連携して動作する.」との記載における「フロー制御」や「リクエストの多重化」は,「ＢＣサービス」の要素である「Ｐ２Ｐプロトコル」による機能として紹介されているのであるから，ここでいう「フロー制御」及び「リクエストの多重化」は，ブロックチェーンネットワークを構成するノード同士の通信に関して説明したものであり，参考図１でいえば，

最下部の「ＢＣネットワーク」の内部で行われる通信におけるものを指すにすぎない。

イ　スレッドとプロセスの置換に関して

（ア）　引用発明のマルチスレッドと本件補正発明の複数のプロセスは，ともに，トランザクションのリクエストを送信する複数の処理単位である点で共通しており，引用文献２及び３に記載される周知技術によれば，並列処理を実行するマルチスレッドとマルチプロセスは，相互に置き換え可能なものである。

そして，引用発明のマルチスレッドは本件補正発明の複数のプロセスに対応付けられるから，マルチスレッドを生成する生成部が，複数のプロセスを生成する生成部に対応付けられることは自明である。

したがって，本件審決が，引用発明において，プロセス多重度に応じた複数のプロセスを生成することを容易想到であると判断した点に誤りはない。

（イ）　原告は，本件補正発明では，生成部においてリクエストの多重化を実現するプロセスの数を必要な数に制御することによって，不要なプロセスを生成せずにリソースを効率的に用いることができると主張するが，相違点３に係る本件補正発明の発明特定事項は「設定されるプロセス多重度に応じた複数のプロセスを生成する生成部」であって，単に「プロセス多重度」が「設定される」だけであって，「閾値Ｔｈｎよりも小さいプロセス多重度」が「設定される」ことは特定されていない。原告の主張は，本件補正発明の発明特定事項に基づくものではなく，失当である。

ウ　顕著な効果に関する主張について

前記イ(イ)のとおり，原告の主張は，本件補正発明の発明特定事項に基づくものではなく，失当である。

また，本件補正発明のようにプロセス多重度を設定するものではない引

用発明との対比によって，プロセス多重度に基づく本件補正発明が顕著な効果を奏すると主張することも失当である。

⑵　小括

相違点３を容易に想到できるとした本件審決の判断に誤りはないから，本件補正を却下すべきであるとした本件審決の判断にも誤りはなく，本願発明を審理の対象として進歩性を欠如するとして本件出願を拒絶すべきものと判断した本件審決の判断にも，誤りはない。

第４　当裁判所の判断

１　認定事実

⑴　本願発明について

本願明細書には，別紙１「本願明細書の記載事項（抜粋）」のとおりの記載があり，この記載によると，本件出願に係る発明について，次のような開示があると認められる。

ア　技術分野

本件出願に係る発明は，トランザクションをブロックチェーンネットワークにリクエストするための技術に関する（【０００１】）。

イ　背景技術

ブロックチェーンを用いた分散型台帳技術が，ビットコイン等の暗号資産（仮想通貨）を管理する方法として用いられているが，近年では，これらの技術は，このような暗号資産だけではなく，様々な資産を管理したり移動したりする方法として用いることも検討されている（【０００２】）。

ウ　発明が解決しようとする課題

分散型台帳技術では，Ｐ２Ｐ（Peer to Peer）によって接続された複数のノードによってネットワークが形成され，そのネットワークにおける複数のノードによって台帳が分散して記録されるところ，分散型台帳技術においては，ほとんどの場合，ブロックチェーンが台帳に記録され，台帳に

直接記録されない場合でも何らかの形態で用いられる（【０００４】）。

ブロックチェーンネットワークを構成する各ノードによれば，互いが保持するデータの正当性を高めるために，所定のアルゴリズムによって合意形成が行われるが，これによって，データの改ざんが難しくなり，取引の信頼性が保たれるものの，一般的には合意形成の処理には多くの時間を要すると考えられているため，大量のトランザクションが発生するような処理に適用することは難しいと考えられている（【０００５】）。

本件出願に係る発明の目的の一つは，分散型台帳技術を用いた場合でも，多くのトランザクションを効率よく処理することにある（【０００６】）。

エ　課題を解決するための手段

本件出願に係る発明の一実施形態によれば，台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスを生成する生成部と，トランザクションの指示を受け付け，前記複数のプロセスのいずれかに当該トランザクションのリクエスト送信を割り当てる割当部と，を備えるシステムが提供される（【０００７】）。

前記複数のノードで形成されるネットワークにおける所定のトランザクションのリクエストに対する平均スループットとプロセス多重度との関係において，プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，前記複数のプロセスの数は，前記第２値よりも小さい数で設定されてもよい（【００１０】）。

前記複数のノードで形成されるネットワークは，コンソーシアム型であってもよい（【００１１】）。

本件出願に係る発明の一実施形態によれば，トランザクションの指示を

受け付け，台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスのいずれかに，前記指示に基づくトランザクションのリクエスト送信を割り当てる，処理方法が提供される（【００１２】）。

オ　発明の効果

本件出願に係る発明の一実施形態によれば，分散型台帳技術を用いた場合でも，多くのトランザクションを効率よく処理することができる（【００１７】）。

カ　発明を実施するための形態

（ア）　ブロックチェーンネットワーク５における演算処理能力にボトルネックが存在する状況は，極めて多くの処理が集中した場合であって，実際には，その前の段階，すなわちトランザクションのリクエストを送信する側においてボトルネックが存在する場合が多いとの知見に基づき，リクエスト制御システム１では，ブロックチェーンネットワーク５における演算処理能力にボトルネックが存在する段階までリクエストを送信するためのプロセスを多重化することで，処理量を向上させることができる（【００２２】）。

（イ）　実施形態のブロックチェーンネットワークは，管理主体が存在するコンソーシアム型（例えば，Quorum, Hyperledger Fabric等）を想定しているが，管理主体が存在しないパブリック型（例えば，Ethereum等）であっても適用可能である。（【００２６】）。

（ウ）　プロセス生成装置１１Ａは，設定装置３９から指定された数のプロセス１１０を起動するとともに，各プロセス１１０においてトランザクションのリクエストを送信してから，ブロックチェーンネットワーク５においてトランザクションが分散型台帳に記録されたことを検出するまでの時間（以下「スループット」という。）を測定する。この測定したス

ループットを設定装置３９に送信する（【００４０】）。

設定装置３９は，プロセス生成装置１１Ａ及び指示生成装置３８を制御して，プロセス多重度（プロセス１１０の数：m個）を変更しつつ，１プロセス当たりの平均スループットを測定する。これによって，設定装置３９は，プロセス多重度と平均スループットとの関係を取得し，この関係を用いて，利用するブロックチェーンネットワーク５における最適なプロセス１１０の数を算出するが，この数は，上述したｎ個に相当するものとして，リクエスト制御システム１におけるプロセス生成装置１１に設定される（【００４２】）。

図６は，本発明の一実施形態におけるプロセス多重度と平均スループットとの関係を説明する図であるところ，プロセス多重度が低い場合，プロセス多重度の増加に伴い平均スループットは増加していくが，プロセス多重度が大きくなると，プロセス多重度が増加しても，平均スループットは増加しなくなっていくように，プロセス多重度が大きい値Ｎ２である場合の平均スループットの増加の割合は，プロセス多重度が小さい値Ｎ１（第１値）である場合の平均スループットの増加の割合（第１割合）よりも小さい（【００４４】）。特定のプロセス多重度における平均スループットの増加の割合は，図６に示すグラフにおいて，そのプロセス多重度における傾き（微分値）に相当するところ，この増加の割合（増加率）がプロセス多重度の増加に対して大きく減少し始めるとき（第２割合）のプロセス多重度が，図６に示す閾値Ｔｈｎ（第２値）に対応する（【００４４】）。

このように，閾値Ｔｈｎよりもプロセス多重度が小さい領域Ａ１においては，ブロックチェーンネットワーク５における演算処理能力にボトルネックがあるのではなく，リクエストを送信する側の処理にボトルネックがあり，一方，閾値Ｔｈｎよりもプロセス多重度が大きい領域（判

決注・前後の文脈から「量器」は「領域」の誤記と認める。）Ａ２においては，プロセス多重度を増加させても，平均スループットがほとんど増加しないことから，ブロックチェーンネットワーク５における演算処理能力にボトルネックがあることが分かる（【００４５】）。

設定装置３９が，閾値Ｔｈｎよりも小さいプロセス多重度をプロセス１１０の数としてプロセス生成装置１１に設定することにより，ブロックチェーンネットワーク５における演算処理能力にボトルネックを発生させずに，リクエストの送信側においてボトルネックを発生させることができ，プロセス生成装置１１において不要なプロセス１１０を生成しないようにして，リクエスト制御システム１のリソースを効率的に用いることができる（【００４６】）。

⑵　引用発明１について

引用文献１には，別紙２「引用文献１の記載事項（抜粋）」のとおりの記載があり，この記載によると，引用発明として，本件審決が認定するとおりのものを認定することができ，この点は，当事者間にも争いがない。

２　取消事由（本件補正発明の進歩性に関する判断の誤り）の有無について

⑴　相違点３の容易想到性の有無について

ア　引用文献１の記載事項に関して

引用文献１には，①課題として，「２．３　本研究の課題　ＢＣの実適用に向けては，ＢＣ基盤およびその実装における現時点での技術課題の明確化が必要である．特に，金融業務への適用に向けては，一般的にトランザクションのスループットが性能面の最重要指標である.」（１６９頁左欄２９ないし３３行目）を掲げ，②課題抽出に向けての目的として，「目的２：Fabric／ＢＣ基盤の性能ボトルネック抽出方法の検討　性能限界や傾向を把握するためには，そのボトルネック箇所を特定することが重要である.」（１６９頁右欄９ないし３８行目）とし，③測定方法として，マルチ

スレッドでトランザクションを並列実行して負荷をかけるものとし，その方法として，スレッドごとにトランザクションを指定した回数繰り返し，全スレッドのトランザクションが完了するまでの時間を測定することとし（１７１頁左欄３ないし２３行目），④トランザクションのスループットの計算方法を，「全スレッドによる合計リクエスト件数／（全レスポンスが返ってきた時刻－リクエスト処理を開始した時刻）」とし，１スレッドあたりのリクエスト数を１０００あるいは２００に固定し，並列スレッド数を変える等の条件で実験したところ（１７１頁左欄２４行ないし右欄７行目，１７１頁表３），⑤結果として，「並列スレッド数を増やしていくとスループットも増加する傾向にあった．ただし，ある程度並列度を増やすとスループットは頭打ちとなった．また，スループットが頭打ちになった後も，それ以上に並列度を増やしていくと，内部的にエラーが発生する等，安定稼働が困難となる場合が見受けられた．つまり，高負荷を与えた場合には，挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得る．」（１７１頁右欄２７ないし３９行目，１７２頁図３，１７２頁図４）との知見が得られたとの記載がある。

これらの記載によると，引用文献１の実験においては，スレッド当たりのリクエスト数をセキュリティ機能のＯＦＦ又はＯＮの相違に従って固定し，並列スレッド数を変化させてスループット（１秒当たりのリクエスト処理量）を測定しているのであり，「全スレッドによる合計リクエスト件数」は並列スレッド数にのみ左右されるから，引用文献１は，専ら並列スレッド数とスループットとの関係を測定したものであり，その測定結果として，並列スレッド数の増加に対するスループットは，ある程度までは増加し，一定程度で頭打ちとなり，その後は挙動不安定になるというものが得られたとするものである。そうすると，引用文献１は，並列スレッド数を増加させていけばスループットは増加するが，ある程度以降は挙動が安

定しなくなるので，その場合には並列スレッド数の増加による効果がなくなり，「リクエストの流量制限」で対応しなければならないと理解すべきものであるから，その記載内容は，スレッド数の増加による効果には一定の最大限度があることを含意するものというべきである。

以上のとおりであるから　原告の前記第３の１⑴アの主張は採用することができない。なお，原告は，引用文献１においては，「負荷が大きすぎること」，すなわち「単位時間当たりのリクエスト数が大きすぎること」を認識するための手段としてスレッドの数を増加させてみた測定結果が記載されているのにすぎず，このような記載をもって，「スレッド数（並列度）の制御」を，「リクエストの流量制御」における課題解決手段として読み取ることはできないない旨主張するが，前述のとおり，引用文献１の該当部分の記載は，単に課題認識手段としての測定結果を表示したものとはいえず，スレッドの数を増加させた場合の結果に応じて，課題解決に向けた対応策の示唆等にも及ぶものであるから，原告の前記主張は前提を欠くものというべきである。

したがって，引用文献１には，引用発明がスレッド数を制御すること，少なくとも，スレッドの多重度を設定し，これより，設定されるスレッド多重度に応じた複数のスレッドを生成するものであるとの記載があると認められる。

イ　スレッドとプロセスの置換について

（ア）　相違点３は，「本件補正発明は，生成部が『設定されるプロセス多重度に応じた』複数のプロセスを生成するものであるのに対して，引用発明は，そのような特定がなされていない点」というものである。

（イ）　引用文献２の記載事項は別紙３「引用文献２の記載事項（抜粋）」のとおりであり，引用文献３の記載事項は別紙４「引用文献３の記載事項（抜粋）」のとおりであり，甲４文献の記載事項は別紙５「甲４文献の記

載事項（抜粋）」のとおりであり，これらの記載事項からすると，並列処理を実現するに当たり，マルチプロセス及びマルチスレッドはどちらも周知の技術であり，どちらを用いて並列処理を実現するかは，当業者が技術的要件等に基づき適宜設計的に決定し得た事項であることが認められる。

（ウ） ここで，本件補正発明についてみると，本件補正発明は「トランザクションのリクエストを送信する複数のプロセスであって，設定されるプロセス多重度に応じた複数のプロセスを生成する生成部」を備えるとのみ特定され，「プロセス多重度」はプロセスの数である（本願明細書【００３１】，【００３８】）。そして，「プロセス多重度」は単に「設定される」と特定されているだけであり，また，設定される「プロセス多重度」と生成されるプロセスとがどのような関係において対応するのかは何ら特定されていない。これに対し，本件補正後の請求項４に係る発明は，「当該プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，前記複数のプロセスの数は，前記第２値よりも小さい数で設定される」と，同請求項８に係る発明は，「当該プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，前記複数のプロセスの数は，前記第２値よりも小さい数で設定される」とされており，これら発明との対比からして，本件補正発明は，これらプロセス数を所定の数に制限する特定がされていないものと理解できる。したがって，本件補正発明は，プロセス数が制御されるものであればこれらを全て含むものと認められる。

（エ） 前記アのとおり，引用発明の構成は，スレッドの多重度を設定し，

設定されるスレッドの多重度に応じた複数のスレッドを生成するものであるところ，前記(イ)のとおり，マルチスレッドとマルチプロセスのいずれの並列処理を実現するかは，当業者が技術的要件等に基づき適宜設計的に決定し得た事項であることからすれば，引用発明のスレッドの構成を適宜プロセスに代え，相違点３に係る，生成部が「設定されるプロセスの多重度に応じた複数のプロセスを生成する」ものに置換することは当業者にとって極めて容易なことであり，これにより引用発明は，前記(ウ)のとおり，本件補正発明に至ることとなる。

ウ　効果について

発明の効果が予測できない顕著なものであるかについては，当該発明の特許要件判断の基準日当時，当該発明の構成が奏するものとして当業者が予測することのできなかったものか否か，当該構成から当業者が予測することのできた範囲の効果を超える顕著なものであるか否かという観点から検討する必要があるところ，前記イ(ウ)のとおりの構成とみるべき本件補正発明の効果は，その構成から当然に当業者が予想し得る範囲内のものにすぎない。

エ　原告の主張について

前記第３の１⑴記載の原告の主張については，前記認定の中で適宜判断済みであるが，特に補足すべき点については以下のとおりである。

(ア)　原告は，前記第３の１⑴イ(ア)のとおり，本願明細書の記載を併せ考慮すれば,「設定されるプロセス多重度に応じた複数のプロセスを生成する生成部」という本件補正発明の発明特定事項によって，プロセス多重度に応じてプロセス数を制限するとの構成が特定されている旨主張する。

しかしながら,前記イ(ウ)のとおり,特許請求の範囲の記載自体から,プロセス多重度に応じてプロセス数を制限するとの構成は読み取れない

し，原告が主張する本願明細書の記載及びプロセス多重度に応じてプロセス数を制限するとの発明特定事項は，本件補正発明とは異なる別の請求項に係るものであるというべきである。

したがって，原告の上記主張を採用することはできない。

(イ) 原告は，前記第３の１⑴イ(イ)のとおり，①マルチプロセスではない引用発明からはスレッド数を制限することが示唆されるだけであって，プロセス数を制限することまで示唆されることはない，②プロセスは１個のメモリ空間が割り当てられた実行プログラムであるのに対して，スレッドはプロセス内に所在してＣＰＵコアに対する命令を実行する単位をいい，この両者はハードウェア資源の利用態様が相違するため，これらを相互に置換することはできない旨主張する。

上記①の主張についてみると，確かに，並列スレッド数増加によるスループット増加に頭打ちの効果があり，更なる増加はむしろ挙動を安定させなくなることから，スレッド数を制限することが示唆されたとしても，マルチスレッドの構成をマルチプロセスの構成に置換する際に，プロセス数を制限することまでもが直ちに動機付けられるとはいい難い。なぜならば，トランザクションのリクエストを送信する際に，マルチプロセスにもマルチスレッドと同様の頭打ち効果や挙動不安定があるとの知見を前提としていなければ，スレッド数を制限するマルチスレッドの構成を，プロセスを制限するマルチプロセスの構成に置換する動機付けはないからである。この点，「かかるマルチプロセスを採用した場合に，プロセスの多重度を制限するため『プロセスの多重度に応じた』複数のプロセスを生成することは，リクエスト送信を割り当てるスレッドの数を制限するという上記示唆に基づいて，当業者であれば容易に想到し得る」とした本件審決の説示は当を得たものとはいい難い。

しかしながら，前記(ア)のとおり，プロセス数を制限することは本件

補正発明の発明特定事項には含まれず,「プロセス多重度」に対応するプロセス数が設定されるものであればよいものと認められるから，引用発明のマルチスレッドの構成をマルチプロセスの構成に置換すれば本件補正発明に至るのであり，本件審決の判断は結論においては正当であり，原告の上記主張は，本件の結論を左右するものとはいえない。

次に，上記②の主張についてみると，マルチスレッドとマルチプロセスとがそれぞれハードウェア資源の利用態様が相違するとしても，マルチスレッド及びマルチプロセスが並列処理を行うための手法として周知であることから，格別な困難でもない限り，マルチスレッドとして構成されたものをマルチプロセスとして構成されたものに転用することは，当業者が適宜なし得る事項である。この転用の際，スレッドからプロセスへ置換する場合に両立しない部分があれば，当業者は技術常識に従い所要の変更を加えることができるのであって，本件補正発明について，それが困難であるとは認められない。

したがって，原告の上記主張を採用することはできない。

（ウ） そのほか原告がるる主張するところも前記アの認定判断を左右しない。

⑵ 小括

前記⑴の認定判断によると，相違点３は容易想到であるとした本件審決の判断は結論において相当であり，そうすると，本件補正発明は引用発明及び周知技術に基づいて当業者が容易に発明をすることができたものであるから，本件補正発明は独立特許要件を欠くものであり，本件補正は特許法１７条の２第６項で準用する同法１２６条７項の規定に違反するので，同法１５９条１項において読み替えて準用する同法５３条１項の規定により却下すべきものであって，本件補正を却下した本件審決の判断に誤りがあるとはいえず，また，本願発明についても，当業者が容易に発明をすることができたものと

いうことになるから，本件出願を拒絶すべきとした本件審決の判断にも誤りはない。

3　結論

よって，取消事由は理由がないから，原告の請求を棄却することとして，主文のとおり判決する。

知的財産高等裁判所第４部

裁判長裁判官　菅　野　雅　之

裁判官　本　吉　弘　行

裁判官　中　村　恭

（別紙１）

本願明細書の記載事項（抜粋）

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は，トランザクションをブロックチェーンネットワークにリクエストするための技術に関する。

【背景技術】

【０００２】

ブロックチェーンを用いた分散型台帳技術が，ビットコイン等の暗号資産（仮想通貨）を管理する方法として用いられている。近年では，これらの技術は，このような暗号資産だけではなく，様々な資産を管理したり移動したりする方法として用いることも検討されている・・・。

【発明の概要】

【発明が解決しようとする課題】

【０００４】

分散型台帳技術では，Ｐ２Ｐ（Peer to Peer）によって接続された複数のノードによってネットワークが形成され，そのネットワークにおける複数のノードによって台帳が分散して記録される。分散型台帳技術においては，ほとんどの場合，ブロックチェーンが台帳に記録され，台帳に直接記録されない場合でも何らかの形態で用いられる。以下の説明では，台帳を分散して記録する複数のノードによって形成されるネットワークであって，ブロックチェーンを扱うネットワークを，ブロックチェーンネットワークという。なお，本明細書でいうブロックチェーンネットワークは，必ずしもビザンチン耐性を備えた構成であることを必須としないが，ビザンチン耐性を備えた構成であってもよい。

【０００５】

ブロックチェーンネットワークを構成する各ノードによれば，互いが保持するデータの正当性を高めるために，所定のアルゴリズムによって合意形成が行われる。これによって，データの改ざんが難しくなり，取引の信頼性が保たれる。一方，一般的には合意形成の処理には多くの時間を要すると考えられているため，大量のトランザクションが発生するような処理に適用することは難しいと考えられている。

【０００６】

本発明の目的の一つは，分散型台帳技術を用いた場合でも，多くのトランザクションを効率よく処理することにある。

【課題を解決するための手段】

【０００７】

本発明の一実施形態によれば，台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスを生成する生成部と，トランザクションの指示を受け付け，前記複数のプロセスのいずれかに当該トランザクションのリクエスト送信を割り当てる割当部と，を備えるシステムが提供される。

【０００８】

前記複数のプロセスは，第１プロセスおよび第２プロセスを含み，前記第１プロセスが前記リクエストを送信するノードは，前記第２プロセスが前記リクエストを送信するノードと同一であってもよい。

【０００９】

前記割当部は，ラウンドロビン方式により前記トランザクションのリクエスト送信を割り当ててもよい。

【００１０】

前記複数のノードで形成されるネットワークにおける所定のトランザクションのリクエストに対する平均スループットとプロセス多重度との関係において，プロセ

ス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，前記複数のプロセスの数は，前記第２値よりも小さい数で設定されてもよい。

【００１２】

本発明の一実施形態によれば，トランザクションの指示を受け付け，台帳を分散して記録する複数のノードの少なくとも１つに対し，トランザクションのリクエストを送信する複数のプロセスのいずれかに，前記指示に基づくトランザクションのリクエスト送信を割り当てる，処理方法が提供される。

【００１３】

前記複数のプロセスは，第１プロセスおよび第２プロセスを含み，前記第１プロセスが前記リクエストを送信するノードは，前記第２プロセスが前記リクエストを送信するノードと同一であってもよい。

【００１４】

前記トランザクションのリクエスト送信を割り当てるときには，ラウンドロビン方式を用いてもよい。

【００１５】

前記複数のノードで形成されるネットワークにおける所定のトランザクションのリクエストに対する平均スループットとプロセス多重度との関係において，プロセス多重度の増加に対する平均スループットの増加の割合が，プロセス多重度が第１値のときに第１割合であり，当該第１値より大きい第２値以上であるときに前記第１割合より小さい第２割合であり，前記複数のプロセスの数は，前記第２値よりも小さい数で設定されてもよい。

【００１６】

前記複数のノードで形成されるネットワークは，コンソーシアム型であってもよい。

【発明の効果】

【００１７】

本発明の一実施形態によれば，分散型台帳技術を用いた場合でも，多くのトランザクションを効率よく処理することができる。

【図面の簡単な説明】

【００１８】

【図１】本発明の一実施形態におけるリクエスト制御システムの構成を示すブロック図である。

【図２】本発明の一実施形態におけるリクエスト制御システムが実行する割当処理を示すフローチャートである。

【図３】本発明の一実施形態におけるプロセスが実行する送信処理を示すフローチャートである。

【図４】本発明の一実施形態におけるリクエスト制御システムが事前設定をする場合の構成を示すブロック図である。

【図５】本発明の一実施形態におけるリクエスト制御システムが実行する設定処理を示すフローチャートである。

【図６】本発明の一実施形態におけるプロセス多重度と平均スループットとの関係を説明する図である。

【発明を実施するための形態】

【００１９】

以下，本発明の一実施形態について，図面を参照しながら詳細に説明する。以下に示す実施形態は本発明の実施形態の一例であって，本発明はこの実施形態に限定して解釈されるものではない。すなわち，以下に説明する複数の実施形態に公知の技術を適用して変形をして，様々な態様で実施をすることが可能である。

【００２０】

＜実施形態＞

［１．システムの概要］

図１は，本発明の一実施形態におけるリクエスト制御システムの構成を示すブロック図である。リクエスト制御システム１は，ユーザ端末８０からネットワークＮＷを介して電文を受信し，その電文に対応したトランザクションのリクエストをブロックチェーンネットワーク５に送信する。このリクエストは，ブロックチェーンネットワーク５における分散型台帳にそのトランザクションを記録させるための指示である。

【００２１】

一般的には，ブロックチェーンネットワーク５において分散型台帳への記録は，多くの時間を要する。したがって処理のボトルネックはブロックチェーンネットワーク５における演算処理能力に存在すると考えられていた。ビザンチン耐性を備える構成である場合には，より多くの時間を要するため，このような傾向が顕著に現れる。一方，ビザンチン耐性を備えていない構成であっても，ビザンチン耐性を備える構成よりも処理時間を要しないものの，分散型台帳への記録という観点では，同様な傾向を有している。

【００２２】

発明者は，様々な検証により，ブロックチェーンネットワーク５における演算処理能力にボトルネックが存在する状況は，極めて多くの処理が集中した場合であって，実際には，その前の段階，すなわちトランザクションのリクエストを送信する側においてボトルネックが存在する場合が多いことの知見を得た。この知見に基づき，本発明の一実施形態におけるリクエスト制御システム１では，ブロックチェーンネットワーク５における演算処理能力にボトルネックが存在する段階までリクエストを送信するためのプロセスを多重化することで，処理量を向上させることができることがわかった。このようにして処理量を向上させるための構成および処理方法について，順に説明する。

【００２３】

［２．ユーザ端末］

ユーザ端末８０は，一般ユーザが利用するスマートフォン，パーソナルコンピュータなどの端末である。この例では，ユーザ端末８０において金融機関が提供するアプリケーションプログラムが起動されると，ユーザの指示によって自身の口座から他の口座への振込処理等の手続が可能になっている。ユーザによって手続の指示がされると，ユーザ端末８０は，その手続に応じたトランザクションの指示内容を含む電文を送信する。

【００２４】

［３．ブロックチェーンネットワーク］

ブロックチェーンネットワーク５は，Ｐ２Ｐ接続ＰＰによって複数のノードが接続されている。図１においては，４つのノード５１０－１，５１０－２，５１０－３，５１０－４によってブロックチェーンネットワーク５が構成されているが，ノードの数は４つより少なくても多くてもよい。以下の説明において，それぞれのノードを区別せずに説明する場合には，単にノード５１０という。

【００２５】

ノード５１０－１は，コンピュータ５１－１と台帳５３－１とを含む。台帳５３－１は，各ノード５１０に分散される台帳のデータが記録される記録媒体である。コンピュータ５１－１では，分散型台帳技術を実現するためのプログラムがＣＰＵによって実行される。コンピュータ５１－１は，台帳５３－１へのデータの記録処理および読み出し処理を実行し，また，他のノード５１０－２，５１０－３，５１０－４において台帳のデータを分散して記録するための処理を実行する。この処理には,分散したデータの正当性を高めるための合意形成処理も含まれる。ここでは，ノード５１０－１を例として，その構成を説明したが，他のノード５１０－２，５１０－３，５１０－４においても，それぞれの構成は同様である。すなわち，いずれかのノード５１０に対してトランザクションのリクエストがされると，そのトランザクションの内容が各ノード５１０における台帳にも記録される。

【００２６】

ブロックチェーンネットワーク５においては，上述のように台帳を分散して各ノード５１０で記録する方式が用いられているが,その過程における具体的な処理は,仕様（基盤ソフトウェア：上述したＣＰＵによって実行されるプログラムに対応）によって様々であって，公知の処理が適用されればよい。したがって，ブロックチェーンネットワーク５における各ノード５１０の処理の詳細は説明を省略する。この例では，ブロックチェーンネットワークは，管理主体が存在するコンソーシアム型（例えば，Quorum, Hyperledger Fabric等）を想定しているが，管理主体が存在しないパブリック型（例えば，Ｅｔｈｅｒｅｕｍ等）であっても適用可能である。

【００２７】

［４．リクエスト制御システム］

リクエスト制御システム１は，プロセス生成装置１１（生成部），負荷分散装置１５（割当部）および通信装置１８を含む。これらの装置は，いずれもＣＰＵによってプログラムを実行することによって，以下に説明する機能をそれぞれ実現している。これらの装置のうち，一部または全部は一体の装置として実現されてもよい。

【００２８】

［４－１．プロセス生成装置］

プロセス生成装置１１は，複数（この例では，予め設定されたｎ個）のプロセス１１０－１，１１０－２，・・・，１１０－ｎを生成する。以下の説明において，それぞれのプロセスを区別せずに説明する場合には，単にプロセス１１０という。各プロセス１１０は，負荷分散装置１５からトランザクションのリクエスト送信を割り当てられると，そのトランザクションについてのリクエストをノード５１０－１に対して送信する。この例では，いずれのプロセス１１０においても，リクエストの送信先は同一のノード（この例ではノード５１０－１）として決められている。なお，上述したように，ブロックチェーンネットワーク５において使用される仕様によって,リクエストの送信先として,複数のノード５１０が指定されてもよいし，

プロセス１１０によって異なるノード５１０が指定されてもよいし，リクエスト毎に異なるノード５１０が指定されてもよい。

【００２９】

プロセス１１０は，リクエストの送信の結果，ブロックチェーンネットワーク５においてトランザクションが分散型台帳に記録されたことを検出すると，キューに入っている次に処理すべきトランザクションのリクエストを送信する。キューは，各プロセス１１０に割り当てられている。プロセス１１０は，分散型台帳に記録されたことを,ブロックチェーンネットワーク５からの通知により検出してもよいし，ブロックチェーンネットワーク５にアクセスすることによって検出してもよい。

【００３０】

［４－２．通信装置］

通信装置１８は，ネットワークＮＷを介してユーザ端末８０から電文を受信し，この電文に含まれるトランザクションの指示内容を負荷分散装置１５へ送信する。

【００３１】

［４－３．負荷分散装置］

負荷分散装置１５は，通信装置１８からトランザクションの指示内容を受信すると，複数のプロセス１１０のいずれかに，そのトランザクションのリクエスト送信を割り当てる。負荷分散装置１５は，割り当てたプロセス１１０のキューに，トランザクションの指示内容を登録する。上述したように，キューに登録されたトランザクションは，そのキューに対応したプロセス１１０からブロックチェーンネットワーク５に対して，リクエストとして送信される。この例では，ｎ個のプロセス１１０が生成されているため，プロセス多重度はｎ個となる。ｎ個の具体的な値は，事前の設定処理によって予め設定される。設定処理については，後述する。

【００３２】

［４－４．システムの動作］

続いて，リクエスト制御システム１の動作について説明する。リクエスト制御シ

ステム１の動作としては，主として，複数のプロセス１１０のいずれかにトランザクションを割り当てるための割当処理，および割り当てられたトランザクションのリクエストを送信するための送信処理を含む。

【００３３】

［４－４－１．割当処理］

図２は，本発明の一実施形態におけるリクエスト制御システムが実行する割当処理を示すフローチャートである。システムの管理者等によりリクエスト制御システム１における処理を開始する指示が入力されると，以下に説明する割当処理が実行される。プロセス生成装置１１は，予め設定された数のプロセス１１０を起動（生成）する（ステップＳ１０１）。通信装置１８による電文の受信を待機する（ステップＳ１０３；Ｎｏ）。通信装置１８により電文が受信される（ステップＳ１０３；Ｙｅｓ）と，負荷分散装置１５は，電文に含まれるトランザクションの指示内容を受け付ける（ステップＳ１０５）。負荷分散装置１５は，複数のプロセス１１０から割り当ての対象となるプロセス１１０を特定し（ステップＳ１０７），そのプロセス１１０にトランザクションのリクエストをする指示をする（ステップＳ１０９）。この指示は，例えば，トランザクションの内容等を含む指示信号によって実現される。ここで，負荷分散装置１５は，例えば，ラウンドロビン方式により，複数のプロセス１１０から割り当ての対象となるプロセス１１０を特定する。なお，この方式はラウンドロビン方式に限らず，別のアルゴリズムを用いた方式であってもよい。

【００３４】

システムの管理者等によりリクエスト制御システム１の処理を終了する指示がなければ（ステップＳ１１１；Ｎｏ），再びステップＳ１０３に戻って処理を続ける。一方，この処理を終了する指示があれば（ステップＳ１１１；Ｙｅｓ），プロセス生成装置１１がプロセス１１０を終了させ（ステップＳ１１３），割当処理が終了される。

【００３５】

［４－４－２．送信処理］

図３は，本発明の一実施形態におけるプロセスが実行する送信処理を示すフローチャートである。プロセス生成装置１１によりプロセス１１０が起動されると，それぞれのプロセス１１０において，図３に示す送信処理が実行される。まず，プロセス１１０は，プロセス生成装置１１によるプロセス終了の指示，または負荷分散装置１５からトランザクションのリクエストを送信する指示を待機する（ステップＳ２０１；Ｎｏ，ステップＳ２０３；Ｎｏ）。プロセス終了の指示があった場合（ステップＳ２０１；Ｙｅｓ）には，送信処理が終了される。

【００３６】

一方，リクエストを送信する指示を受信した場合（ステップＳ２０３；Ｙｅｓ），プロセス１１０は，送信するリクエストに電子署名の付加等の認証処理を行い（ステップＳ２１１），認証処理が施されたリクエストをブロックチェーンネットワーク５に送信する（ステップＳ２１３）。その後，プロセス１１０は，リクエストの送信の結果，ブロックチェーンネットワーク５においてトランザクションが分散型台帳に記録されたことを検出することによって,処理の完了を確認するまで待機し(ステップＳ２１５；Ｎｏ），処理の完了を確認すると（ステップＳ２１５；Ｙｅｓ），ステップＳ２０１に戻って処理を続ける。

【００３７】

以上のように，リクエスト制御システム１が動作することによって，プロセス１１０の数に対応したリクエスト処理の多重化が実現される。後述のようにプロセス１１０の数が設定されているため，ブロックチェーンネットワーク５における演算処理能力がボトルネックになる前にリクエストの送信が制限され，効率的な分散型台帳の運用が可能となる。仮に，不安定要因により，ブロックチェーンネットワーク５における演算処理能力がボトルネックになる状態に移行したとしても，大きな影響を生じないようにすることもできる。

【００３８】

［５．設定処理］　続いて，プロセス生成装置１１において生成されるプロセス１１０の数（プロセス多重度）を設定する処理について説明する。

【００３９】

図４は，本発明の一実施形態におけるリクエスト制御システムが事前設定をする場合の構成を示すブロック図である。設定処理を行う場合におけるリクエスト制御システム１Ａとしては，リクエスト制御システム１に対して，プロセス生成装置１１Ａ，指示生成装置３８および設定装置３９を含み，通信装置１８を含まない構成が例示される。ここでは，プロセス生成装置１１Ａ，指示生成装置３８および設定装置３９について説明し，他の構成については，その説明を省略する。

【００４０】

プロセス生成装置１１Ａは，設定装置３９から指定された数のプロセス１１０を起動するとともに，各プロセス１１０においてトランザクションのリクエストを送信してから，ブロックチェーンネットワーク５においてトランザクションが分散型台帳に記録されたことを検出するまでの時間（以下，スループットという）を測定する。この測定したスループットを設定装置３９に送信する。

【００４１】

指示生成装置３８は，設定装置３９からの制御に基づいて，所定のトランザクションの指示を生成して，指示の内容を負荷分散装置１５に送信する。この指示の内容は，通信装置１８を介して受信する電文に基づくトランザクションの指示内容の代わりとなるものである。

【００４２】

設定装置３９は，プロセス生成装置１１Ａおよび指示生成装置３８を制御して，プロセス多重度（プロセス１１０の数：m個）を変更しつつ，１プロセス当たりの平均スループットを測定する。これによって，設定装置３９は，プロセス多重度と平均スループットとの関係を取得する。設定装置３９は，この関係を用いて，利用するブロックチェーンネットワーク５における最適なプロセス１１０の数を算出す

る。この数は，上述したｎ個に相当するものとして，リクエスト制御システム１におけるプロセス生成装置１１に設定される。

【００４３】

図５は，本発明の一実施形態におけるリクエスト制御システムが実行する設定処理を示すフローチャートである。最初にブロックチェーンネットワーク５に接続する場合，ブロックチェーンネットワーク５における設定の変更（ソフトウェアのバージョンアップ等）があった場合等において，管理者等の指示により設定処理が実行される。まず，リクエスト制御システム１Ａは，設定装置３９の制御により，プロセス多重度を変化させながら（例えば徐々に増加させながら），平均スループットを測定する（ステップＳ５０１）。

【００４４】

図６は，本発明の一実施形態におけるプロセス多重度と平均スループットとの関係を説明する図である。プロセス多重度および平均スループットの絶対値については様々であるものの，多くのブロックチェーンネットワーク５において，このような結果が得られることは，発明者による検証から得られた知見である。すなわち，プロセス多重度が低い場合，プロセス多重度の増加に伴い平均スループットは増加していく。一方，プロセス多重度が大きくなると，プロセス多重度が増加しても，平均スループットは増加しなくなっていく。すなわち，プロセス多重度が大きい値Ｎ２である場合の平均スループットの増加の割合は，プロセス多重度が小さい値Ｎ１（第１値）である場合の平均スループットの増加の割合（第１割合）よりも小さい。特定のプロセス多重度における平均スループットの増加の割合は，図６に示すグラフにおいて，そのプロセス多重度における傾き（微分値）に相当する。この増加の割合を，以下，増加率という。プロセス多重度の増加に対して，増加率が大きく減少し始めるとき（第２割合）のプロセス多重度が，図６に示す閾値Ｔｈｎ（第２値）に対応する。

【００４５】

このように，閾値Ｔｈｎよりもプロセス多重度が小さい領域Ａ１においては，ブロックチェーンネットワーク５における演算処理能力にボトルネックがあるのではなく，リクエストを送信する側の処理にボトルネックがある。一方，閾値Ｔｈｎよりもプロセス多重度が大きい量器Ａ２においては,プロセス多重度を増加させても，平均スループットがほとんど増加しないことから，ブロックチェーンネットワーク５における演算処理能力にボトルネックがあることがわかる。

【００４６】

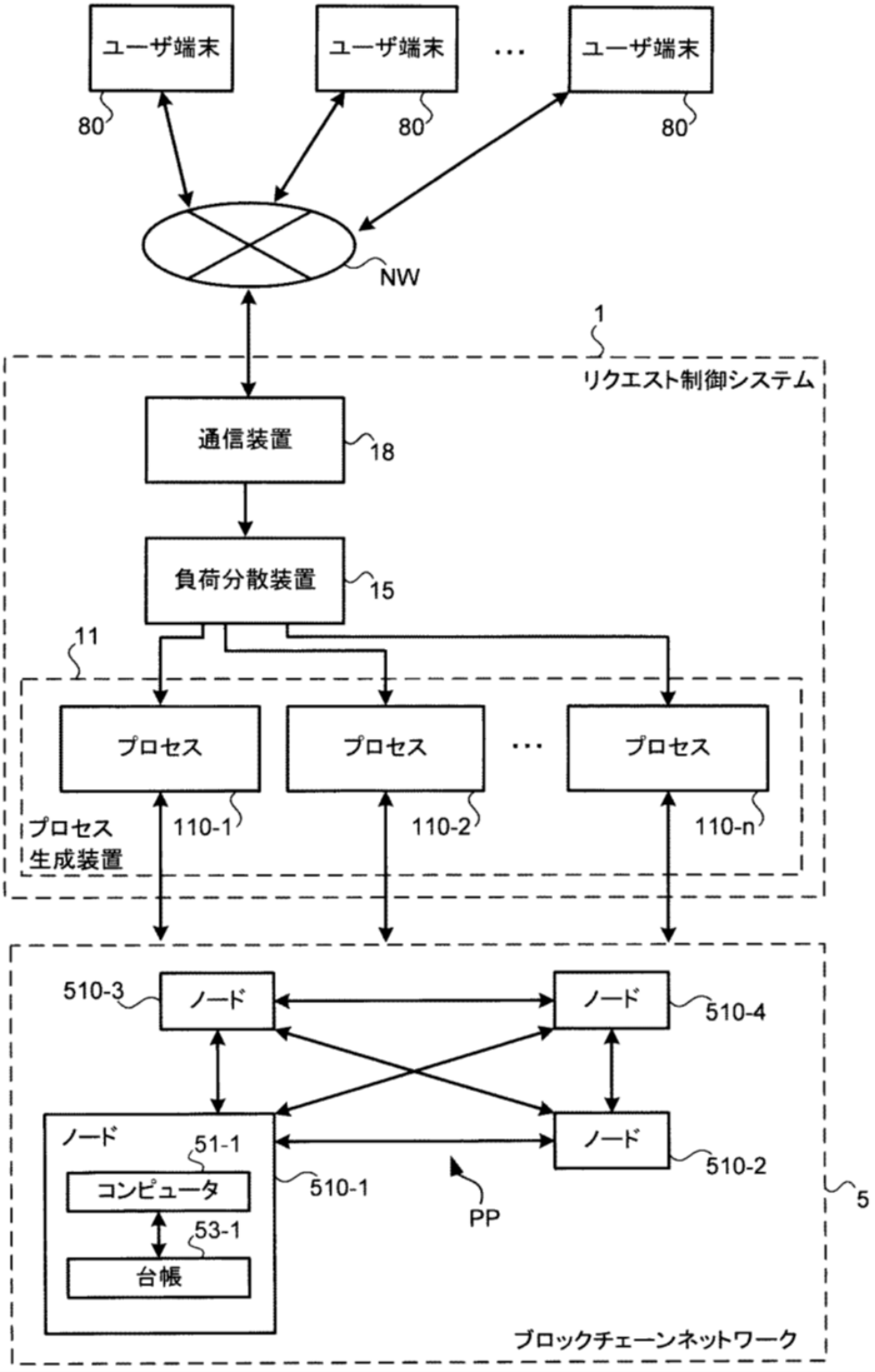
図５に戻って説明をつづける。リクエスト制御システム１Ａ（設定装置３９）は，プロセス多重度に対する平均スループットを測定した後に，各プロセス多重度における平均スループットの増加率を算出する（ステップＳ５０３）。この増加率は，図６に示すグラフにおいて，各プロセス多重度における傾きに相当する。そして，さらに，増加率の変化に基づく所定の演算式によって，閾値Ｔｈｎが特定される（ステップＳ５０５）。設定装置３９は，この閾値Ｔｈｎよりも小さいプロセス多重度をプロセス１１０の数としてプロセス生成装置１１に設定する。このようにプロセス１１０の数を設定することにより，ブロックチェーンネットワーク５における演算処理能力にボトルネックを発生させずに，リクエストの送信側においてボトルネックを発生させることできる。したがって，プロセス生成装置１１において不要なプロセス１１０を生成しないようにして，リクエスト制御システム１のリソースを効率的に用いることができる。

【符号の説明】

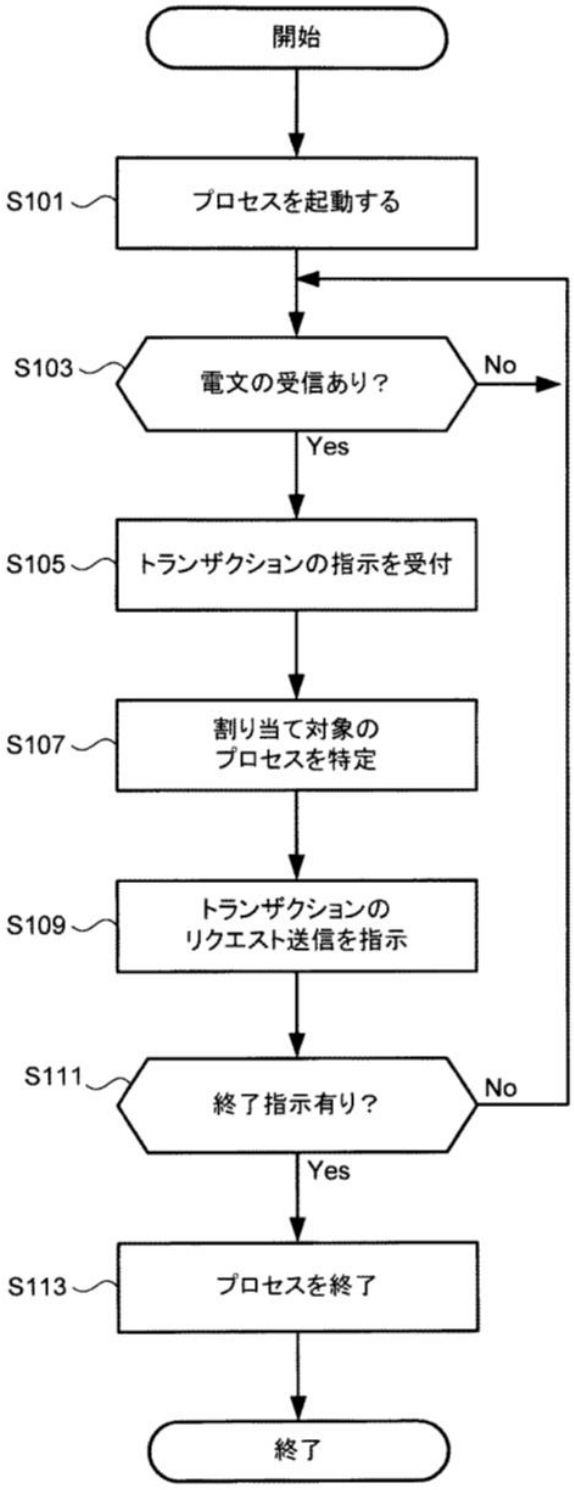
【００４７】

１，１Ａ…リクエスト制御システム，５…ブロックチェーンネットワーク，１１，１１Ａ…プロセス生成装置，１５…負荷分散装置，１８…通信装置，３８…指示生成装置，３９…設定装置，５１…コンピュータ，５３…台帳，８０…ユーザ端末，１１０…プロセス，５１０…ノード

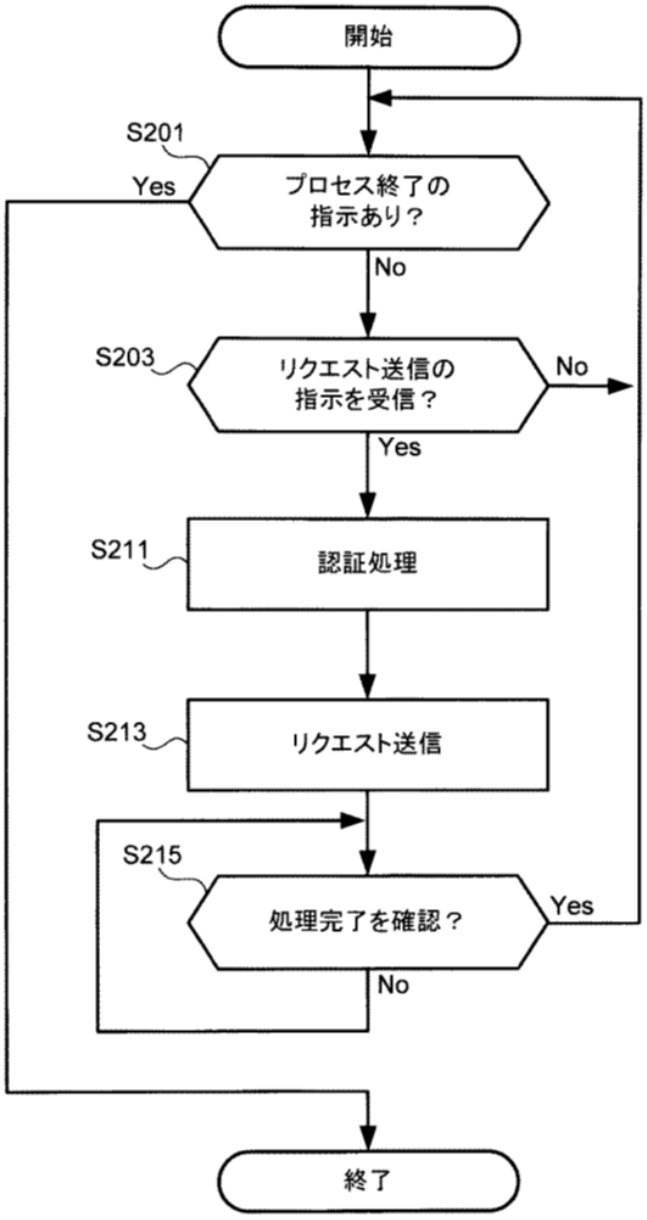
【図１】


ユーザ端末
ユーザ端末
ユーザ端末
80
80
80
NW
1
リクエスト制御システム
通信装置
18
負荷分散装置
15
11
プロセス
プロセス
プロセス
プロセス
生成装置
110-1
110-2
110-n
510-3
ノード
ノード
510-4
ノード
51-1
コンピュータ
53-1
台帳
510-1
ノード
510-2
PP
5
ブロックチェーンネットワーク

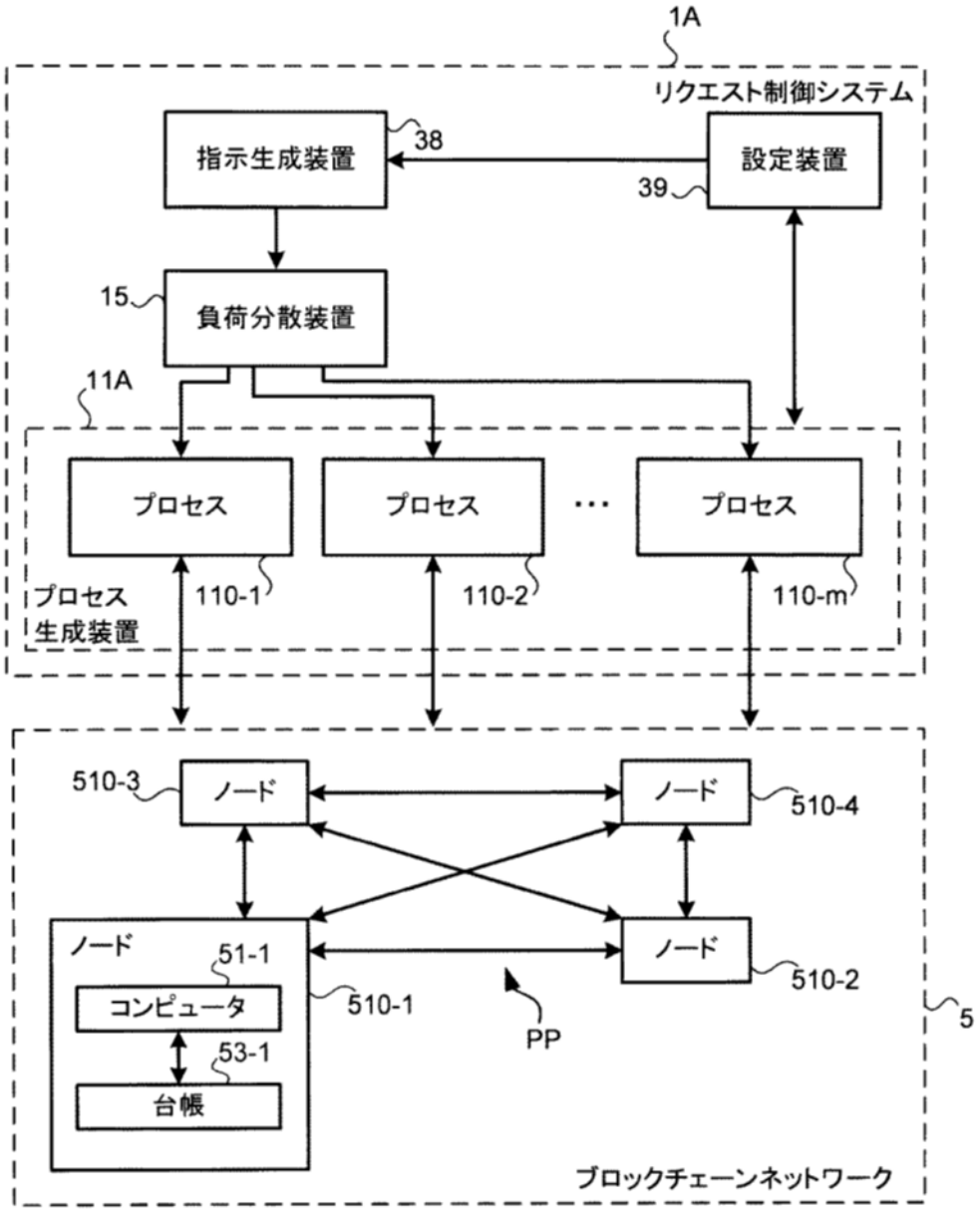
【図２】


開始
S101
プロセスを起動する
S103
電文の受信あり？
No
Yes
S105
トランザクションの指示を受付
S107
割り当て対象の
プロセスを特定
S109
トランザクションの
リクエスト送信を指示
S111
終了指示有り？
No
Yes
S113
プロセスを終了
終了

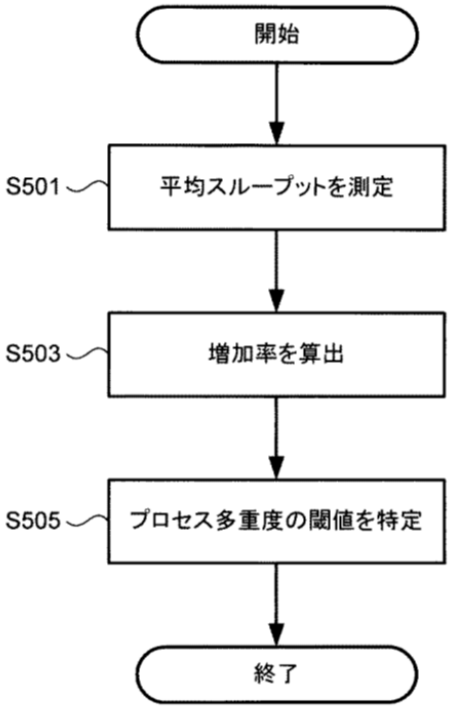
【図３】


開始
S201
Yes
プロセス終了の
指示あり？
No
S203
リクエスト送信の
指示を受信？
No
Yes
S211
認証処理
S213
リクエスト送信
S215
処理完了を確認？
Yes
No
終了

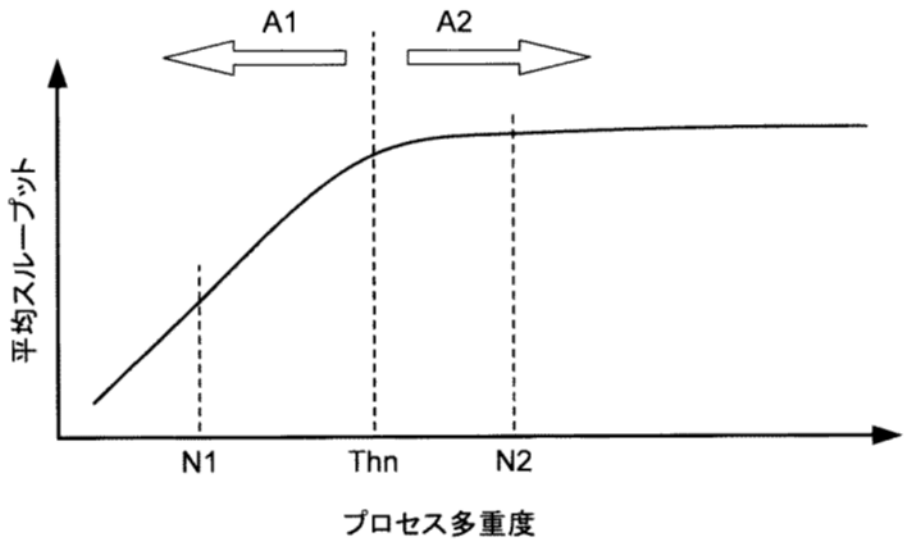
【図４】


1A
リクエスト制御システム
指示生成装置
38
設定装置
39
15
負荷分散装置
11A
プロセス
プロセス
プロセス
プロセス
生成装置
110-1
110-2
110-m
510-3
ノード
ノード
510-4
ノード
51-1
コンピュータ
510-1
ノード
510-2
53-1
台帳
PP
5
ブロックチェーンネットワーク

【図５】


開始
S501
平均スループットを測定
S503
増加率を算出
S505
プロセス多重度の閾値を特定
終了

【図６】


A1
A2
平均スループット
N1
Thn
N2
プロセス多重度

（別紙２）

引用文献１の記載事項（抜粋）

［１６７頁左欄１行ないし右欄５行目］

１.はじめに

ブロックチェーン（Blockchain, BC）技術は，破壊的イノベーションとして金融やInternet of Things（IoT）等の非常に幅広い分野への応用が期待され，注目を集めている．例えば，金融分野では，従来は第三者機関を経由して実施されてきた取引を，ＢＣ技術を用いて利用者間（P2P）の直接取引で代替することで，取引コストの削減ができると期待されている．文献[1]では，様々な分野にＢＣ技術を応用することで，将来的に日本国内の６７兆円もの市場が影響を受ける可能性があるという試算結果が示されている．

ＢＣ技術に大きな期待が集まっている一方で，現状ではセキュリティ面やシステム性能面をはじめ，エンタープライズでの実適用には課題が多いといわれている．そのため，ＢＣの実適用に向けては，ＢＣ基盤やそのオープンソースソフトウェア（Open Source Software, OSS）実装における現時点での技術課題の明確化が必要である．

そこで本研究では，ＢＣ基盤のＯＳＳプロジェクトであるHyperledger[8]の基盤実装の一つであるFabric[11]について性能を中心とした評価を行い，ＢＣ基盤およびFabricの現時点での実力を明らかにするとともに，技術的な課題を抽出にすることを目的とする．

［１６８頁右欄５ないし２６行目］

２．２　ＢＣ基盤 Hyperledger Fabric

Hyperledgerプロジェクトは，Linux Foundationが設立したエンタープライズで利用可能なＯＳＳのＢＣ基盤の開発を目的としたプロジェクトである[8]．同プ

ロジェクトは，２０１６年２月に設立され，金融機関をはじめとしたユーザ企業やＩＴベンダー等，計１００社以上が参画している．弊社も，同プロジェクトの設立時からプレミアメンバーとして参画し，コミュニティ活動に参加している．

Hyperledgerプロジェクトでは，ＢＣのユースケース，基盤の機能要件およびアーキテクチャがホワイトペーパーにまとめられている［９］．また，その実現に向けて，複数のベンダーから基盤実装が提案されている．ＩＢＭ社とＤＡＨ（Digital Asset Holdings）社の共同提案であるFabric[11]はその基盤実装の一つである．Fabricは２０１６年４月に公開され，２０１７年１月現在で開発者プレビュー版ｖ０．６までリリースされている．Fabricは前述のホワイトペーパーに対応したアーキテクチャを実装している．

Fabricは，様々な分野でのユースケースに対応可能とするために汎用性の高いＢＣ基盤機能を提供する．また，現在は，コンソーシアムあるいはプライベート型での利用を想定したＢＣ基盤となっている．Fabricの主な機能的特徴として，具体的には以下が挙げられる．

［１６８頁右欄３６行ないし１６９頁左欄４行目］

図１は，公式ドキュメントに示されるFabricのアーキテクチャである．公式ドキュメントの記載内容に従って，Fabricの主要な構成要素を説明する。

・**メンバーシップサービス**：ＢＣネットワークへの参加者，スマートコントラクト，合意形成を行う検証ノード等，ネットワーク上の全オブジェクトのＩＤを管理する．

・**ＢＣサービス**：Ｐ２Ｐプロトコル，分散台帳，コンセンサスマネージャといった要素によって構成される．Ｐ２Ｐプロトコルにより，Ｐ２Ｐでの双方向ストリーミング，フロー制御，リクエストの多重化といった機能を提供する。既存ネットワークと連携して動作する。分散台帳により，ＢＣと，台帳の（最新）状態を管理する。コンセンサスマネージャにより，プラグイン可能な合意形成アルゴリズ

ム用のインタフェースを提供する。

・**チェーンコードサービス**：スマートコントラクトを実行する軽量でセキュアな実行環境を提供する。

［１６９頁左欄２９ないし３３行目］

２．３　本研究の課題

ＢＣの実適用に向けては，ＢＣ基盤およびその実装における現時点での技術課題の明確化が必要である．特に，金融業務への適用に向けては，一般的にトランザクションのスループットが性能面の最重要指標である．

［１６９頁右欄９ないし３８行目］

３．　Hyperledger　Fabricの評価

３．１　評価の目的

FabricおよびＢＣ基盤の課題抽出に向けて，以下を主な目的として評価を行う．

目的１：Fabricの現実装における実力（主に性能）の把握

Fabricは公開されて間もないため，その品質（特に非機能面）が未知数であった．そのため，実機上に環境を構築して動作検証を行い，さらには性能を測定することで，Fabricの現実装における実力を把握する必要がある．性能評価においては，より本格的なＢＣネットワークを構築することが望ましい．

目的２：Fabric／ＢＣ基盤の性能ボトルネック抽出方法の検討

性能限界や傾向を把握するためには，そのボトルネック箇所を特定することが重要である．しかし，Fabricでは，そのような調査や分析を行う手段が整備されていない．そのため，性能ボトルネックの抽出方法の検討が必要である．Fabric自体のバージョンアップ時等には，再度性能評価を行うことが予想されるため，ボトルネック抽出は繰り返し実行できるようにシステム化することが望ましい．

３．２　評価方法

３．２．１　概要

先に示した評価の目的１と２を達成するために，以下の評価方針を採用した．

目的１の達成に向けた方針

性能評価に適した本格的な評価環境として，マルチホスト上にまたがった環境上にFabricによるＢＣネットワークを構築し，その上で性能測定を行う．

目的２の達成に向けた方針

Fabric（およびＢＣ基盤）の性能ボトルネックを容易に抽出可能とするためのモニタリング環境も合わせて整備する．

［１７１頁左欄３ないし２３行目］

３．２．３　測定方法

クライアントからＲＥＳＴ経由でＢＣネットワークにアクセスし，チェーンコードを実行して負荷をかけることで性能測定を行った．チェーンコードには，Fabricの公式プロジェクトに付属のサンプルチェーンコード「ｍａｐ」を利用する．本サンプルチェーンコードは簡易キーバリューストアとして動作する．なお，負荷をかける際には，複数のクライアントからの同時アクセスを模擬するため，マルチスレッドでトランザクションを並列実行した．

クライアントによる負荷生成ツールプログラムの処理の流れは以下のとおりである．

１．　ユーザログイン（セキュリティ機能ＯＮ時のみ）
２．　ｍａｐチェーンコードをＢＣ上にデプロイ
３．　スレッド毎に実行トランザクション（invoke）を指定した回数繰り返す（並列実行）
４．　全スレッドの実行トランザクションが完了するまで（レスポンスがかえってくるまで）待つ
５．　スレッド毎に参照トランザクション（query）を指定した回数繰り返す（並

列実行）

６． 全スレッドの参照トランザクションが完了するまで（レスポンスがかえってくるまで）待つ

［１７１頁左欄２４行ないし右欄７行目］

ここで，今回の測定におけるトランザクションのスループット計算方法は以下のとおりである．

スループット(tx/s)

$$= \frac{\text{全スレッドによる合計リクエスト件数}}{\text{全レスポンスが返ってきた時刻－リクエスト処理を開始した時刻}}$$

３．２．４　実験パラメータ

今回の測定で利用した実験パラメータ表３のとおりである．これらは性能に与える影響が特に大きいと想定したパラメータである．セキュリティ機能ＯＦＦとＯＮ時のそれぞれの場合で測定した。ＯＮ時には，メンバーシップサービスを利用して，ネットワークへの参加ノードの認証やトランザクションの秘匿化が行われる．一方，ＯＦＦ時にはメンバーシップサービスは利用されず，認証や秘匿化は行われない．

［１７１頁右欄２７ないし３９行目］

３．３　結果と考察

図３にセキュリティ機能ＯＦＦ時の，図４にセキュリティ機能ＯＮ時のinvoke／queryスループット測定結果を示す．図に示す通り，セキュリティ機能ＯＦＦ／ＯＮ時ともに，invokeとqueryの両方で，並列スレッド数を増やしていくとスループットも増加する傾向にあった．ただし，ある程度並列度を増やすとスループットは頭打ちとなった．

また，スループットが頭打ちになった後も，それ以上に並列度を増やしていくと，

内部的にエラーが発生する等,安定稼働が困難となる場合が見受けられた.つまり,高負荷を与えた場合には，挙動が安定しなくなる場合があるため，フロント側でリクエストの流量制御を行う等の対策が必要となり得る.

［１７０頁図２］

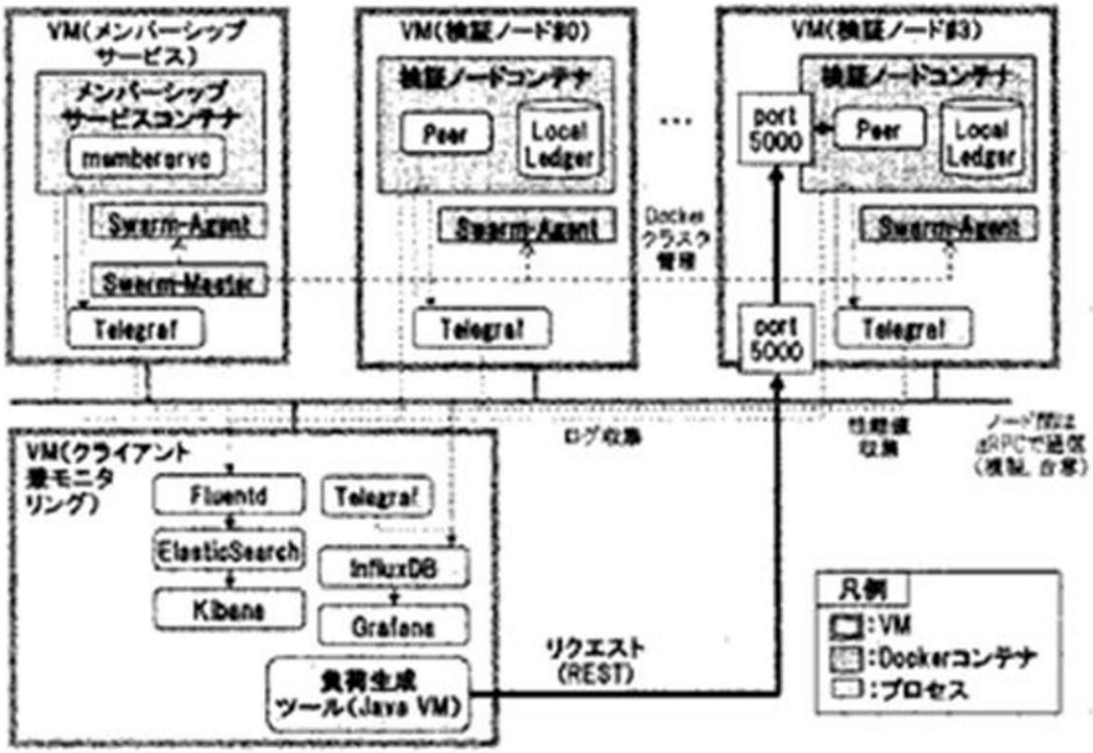


図２　評価環境のシステム構成図

Figure 2　System architecture of evaluation environment.

［１７１頁表３］

表３　実験パラメータ

Table3 Parameters of experiments

パラメータ	設定値のパターン
セキュリティ機能	ＯＦＦ，ＯＮ
合意形成アルゴリズム	Batch PBFT（バッチサイズ＝２）
クライアントのスレッド数	１，２，４，６，８，１０，１２，･･･
スレッドあたりの リクエスト数	１０００（セキュリティ機能ＯＦＦ） ２００（セキュリティ機能ＯＮ）

［１７２頁図３］及び［１７２頁図４］

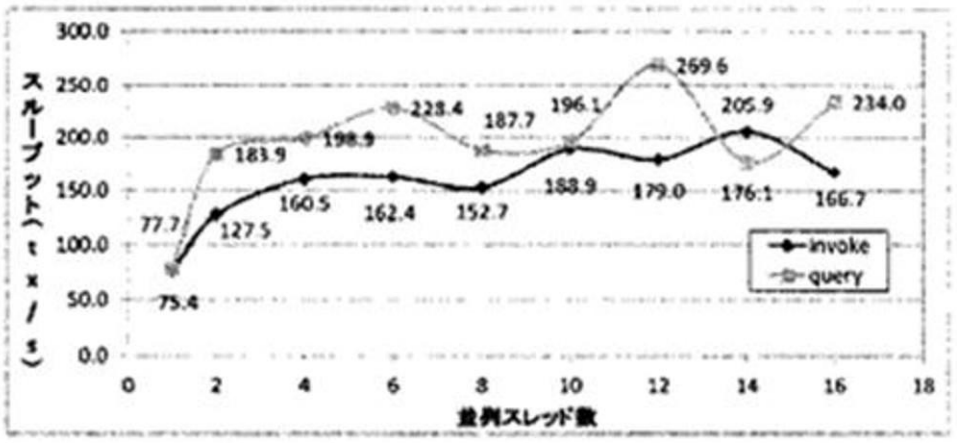


図３　セキュリティ機能 OFF 時のスループット

Figure 3　Throughput of transactions (Security: OFF).

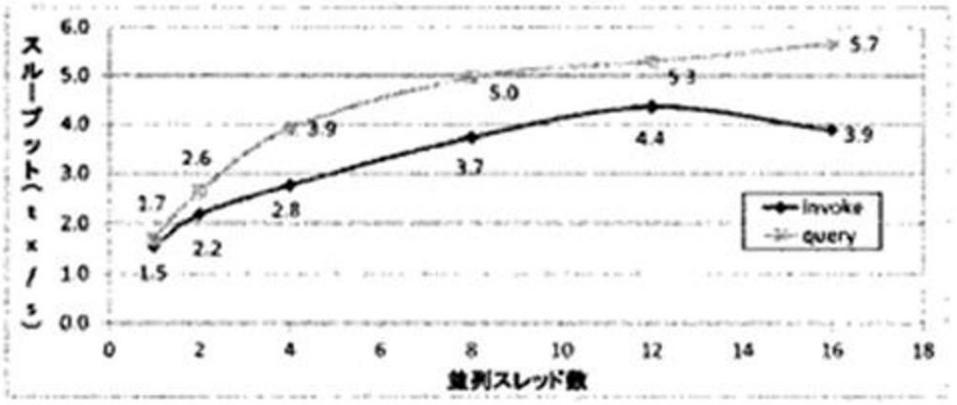


図４　セキュリティ機能 ON 時のスループット

Figure 4　Throughput of transactions (Security: ON).

（別紙３）

引用文献２の記載事項（抜粋）

【０００８】

プロセッサー１１は，画像形成装置１０の動作に必要な演算及び制御などの処理を行うコンピューターの中枢部分に相当する。プロセッサー１１は，ＲＯＭ１２又は補助記憶デバイス１４などに記憶されたシステムソフトウェア，アプリケーションソフトウェア又はファームウェアなどのプログラムに基づいて，画像形成装置１０の各種の機能を実現するべく各部を制御する。プロセッサー１１は，例えば，ＣＰＵ（central processing unit），ＭＰＵ（micro processing unit），ＳｏＣ（system on a chip），ＤＳＰ（digital signal processor）又はＧＰＵ（graphics processing unit）などである。あるいは，プロセッサー１１は，これらの組み合わせである。プロセッサー１１は，好ましくは，マルチコアＣＰＵ，又はＧＰＵとＣＰＵとを備えるプロセッサーである。複数のコアを備えるプロセッサーは，マルチスレッド又はマルチプロセスなどの並行処理を並列処理することで高速に処理することが可能なためである。なお，並行処理とは，複数のスレッド又はプロセスなどを，時分割などによって切り替えながら同時的に処理すること，あるいは並列処理することなどである。また，並列処理とは，複数のスレッド又はプロセスなどを，複数のコアで同時に処理することなどである。

【００６７】

プロセッサー１１は，上記の実施形態においてマルチスレッドで行っている処理をマルチプロセスで行っても良い。

（別紙４）

引用文献３の記載事項（抜粋）

【００４５】

並列処理の手法としては，マルチスレッドやマルチプロセスを用い，又はプログラム内での繰返し処理によって行うことができる。図９は，経路多重度３の場合の疎通確認をマルチスレッドで行う一例を示している。このスレッドは，経路多重度数だけ起動され，第１の多重経路のスレッドと第２の多重経路のスレッドと第３の多重経路のスレッドとで,それぞれ異なる疎通経路について同時に疎通確認を行う。

（別紙５）

甲４文献の記載事項（抜粋）

【０００２】

情報処理装置（コンピュータ）は，例えば，コンピュータプログラム（略してプログラムとも記す）を実行する際に，プログラムを実行する単位である複数のプロセスを生成する。さらに，このような場合には，情報処理装置は，プロセス内に，処理を実行する単位である複数のスレッドを生成する。

【００１８】

＜第１実施形態＞

図１は，本発明に係る第１実施形態の情報処理装置の構成を簡略化して表すブロック図である。第１実施形態の情報処理装置１０１は，例えばＣＰＵ（Central Processing Unit）１０２を備えたコンピュータである。ＣＰＵ１０２は，記憶装置（図示せず）に格納されているコンピュータプログラム（プログラム）を読み出し当該プログラムを実行することによって，情報処理装置１０１の全体的な動作を制御する。

【００１９】

この第１実施形態では，情報処理装置１０１（ＣＰＵ１０２）は，プログラムを実行する単位である複数のプロセスを生成する。プロセスは，ＣＰＵ１０２の機能部の一つであり，当該プロセスの動作（処理）を管理する機能を備えている。例えば，プロセス（ＣＰＵ１０２）は，当該プロセスが受けたリクエストに応じた処理を実行する単位であるスレッドを生成（設定）する。プロセスは，通常，複数の処理を実行することから，複数のスレッドを生成可能となっている。当該プロセスが持つことができるスレッドの上限数は予め設定されている。

【００６４】

図３は，第４実施形態の情報処理装置の構成を簡略化して表すブロック図である。

この第４実施形態の情報処理装置は，サーバ装置（コンピュータ）１０であり，情報通信網（ネットワーク）７０を介して複数のクライアント端末３０に接続されている。また，サーバ装置１０はデータベース６０に接続されている。

【００６５】

クライアント端末３０は，利用者が情報を入力するためのキーボード等の入力手段と，各種の情報を表示するためのディスプレイ等の出力手段とを備える。ここで，クライアント端末３０としては，例えば，パーソナルコンピュータ（パソコン），タブレット型端末またはＰＤＡ（Personal Digital Assistant）端末が考えられるが，これらに限定されない。

【００６６】

サーバ装置１０は通信部４０を備えており，当該通信部４０によって，サーバ装置１０は，クライアント端末３０とデータの送受信を行う。

【００６７】

サーバ装置１０は，さらに，例えばＣＰＵを有し，当該ＣＰＵにより実現される機能部として，プロセス１１と，障害対策部１００とを備えている。さらに，サーバ装置１０は，記憶媒体であるメモリ５０を備えている。

【００６８】

プロセス１１は，コンピュータプログラム（プログラム）の実行単位であり，プログラムを実行する際に生成される。この生成されるプロセス１１には，メモリ５０内に，専用の記憶領域が割り当てられる。なお，サーバ装置１０には，通常，複数のプロセス１１が生成されるが，ここでは，図示の簡略化のために，一つのプロセス１１のみ表すこととする。

【００６９】

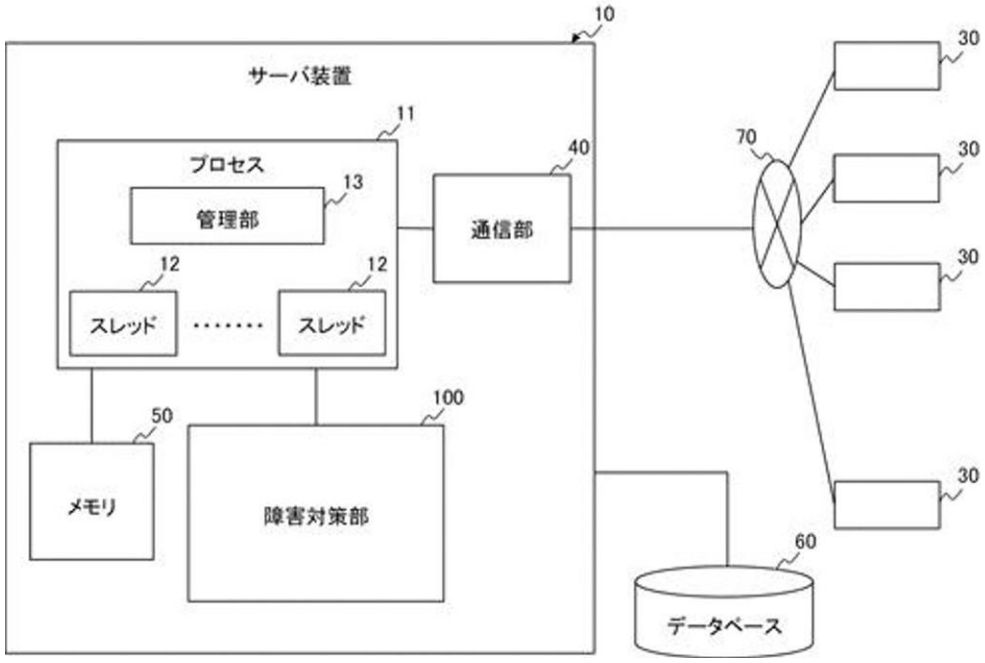
プロセス１１は，管理部１３を備えている。この管理部１３は，プロセス１１の動作を管理する機能を備えている。例えば，管理部１３は，プロセス起動時に，予め初期値として定められた複数の待機状態のスレッド１２を生成する。また，管理

部１３は，各スレッド１２に，各スレッド１２を識別するスレッド識別情報を付与する。さらに，管理部１３は，プロセス１１に対して割り当てられたメモリ５０内の記憶領域から，それら生成した各スレッド１２に，予め定められた容量を持つ記憶領域を割り当てる。

【００７２】

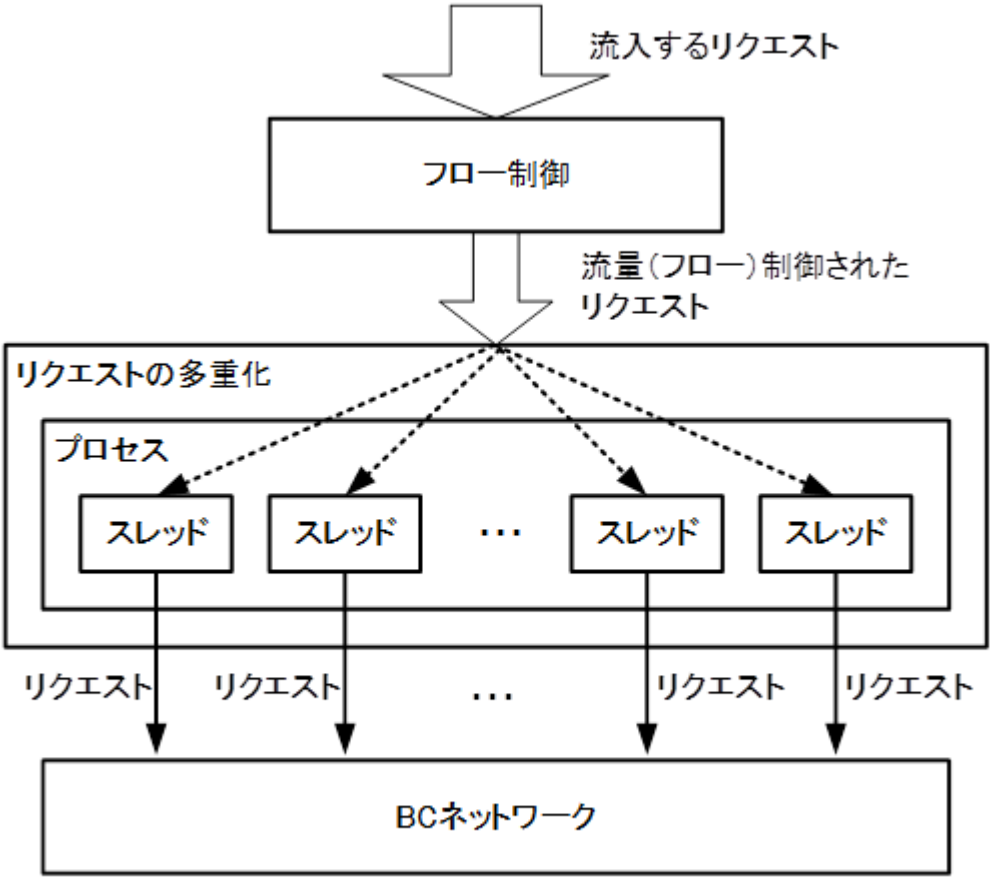
通信部４０は，クライアント端末３０から受け取ったリクエストを，複数のプロセス１１のうちの何れのプロセスに渡すかを判断する機能を備えている。リクエストとは，例えば，データベース６０内のデータを検索する要求や，データを更新する要求である。

【図３】


10
サーバ装置
11
プロセス
13
管理部
12
スレッド
12
スレッド
40
通信部
70
30
30
30
30
50
メモリ
100
障害対策部
60
データベース

（別紙６）

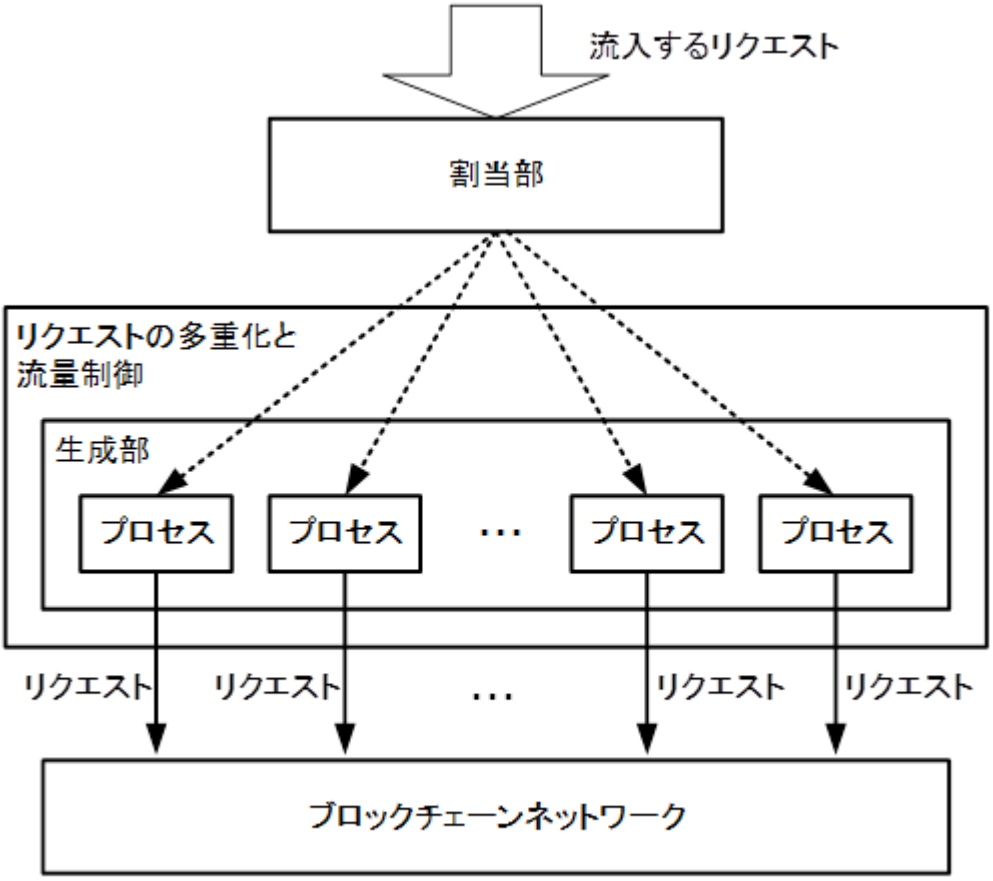
参考図１



参考図1

（別紙７）

参考図２



参考図2