mijin プロジェクト

実証実験報告書

2016/08/28

ver1.1

アララ株式会社

目次

更新履歴

日付	バージョン	更新内容
2016-08-28	1.00	初版作成
2016-09-28	1.10	結論の更新

1 実証実験の背景・目的

(1) 2008 年、サトシナカモトという人物が、ブロックチェインと呼ばれることとなった、非集権、分散型の仮想通貨システムであるビットコインの論文を投稿、2009 年にはビットコインのソフトウェアをネットに公開し、運用が開始された。

(2) ブロックチェインは、以下の特徴を持つ。

(ア) 任意の複数のコンピュータ上で同じデータを同期する分散型

(イ) 取引データブロックをチェーンでつなぐことにより、データ改ざんが非常に困難

(ウ) インターネット上に接続されたPC、サーバ問わず何台でもノードとして参加できる仕組みをパブリック型と呼ぶ。管理者を置き、非公開型として企業などが運用する仕組みをプライベート型と呼ぶ。

(3) mijin は、2014 年 1 月にブロックチェイン 2.0 プロジェクトとして開発が始まり、オープンソースとして公開されている、パブリック型ブロックチェーンソフトウェアである NEM をベースに、プライベート型として開発されたソフトウェアで、テックビューロ株式会社が開発したものである。

(4) 多対多で通貨を送金、受金するシステムの有用性、コストメリットは他社実証実験等で証明されているが、当社のポイントプラスシステムのような、多対 1 で大量トランザクションが発生するようなシステムへの有用性は不明である。

(5) そこで、当社の主力サービスである、ポイントプラスに近いシステム構成を想定し、実証実験を行い、性能、コスト、システムの特徴を把握することで、当社サービス、製品開発の方向性を明確にするための一助とすることを目的とする。

(6) また、多対多の取引についても、通貨以外でのさまざまな利用が提案されているが、実際に何ができて何ができないのかが不明瞭なため、本実証実験を通じて、それらのノウハウも得ることを目的の一つとする。

参考

(1) サトシナカモト:https://ja.wikipedia.org/wiki/%E4%B8%AD%E6%9C%AC%E5%93%B2%E5%8F%B2

(2) ビットコイン:
https://ja.wikipedia.org/wiki/%E3%83%93%E3%83%83%E3%83%88%E3%82%B3%E3%82%A4%E3%83%B3

(3) ブロックチェイン:
https://ja.wikipedia.org/wiki/%E3%83%96%E3%83%AD%E3%83%83%E3%82%AF%E3%83%81%E3%82%A7%E3%83%BC%E3%83%B3

(4) テックビューロ社 mijin:
http://mijin.io/ja/about-mijin

2 結論

課題はいくつかあるものの、不整合性が発生することなく、分間 3,000 以上の取引が可能であり、データの改ざんが困難かつデータ消失を防止する仕組みに実用性、容易性が高いことが証明された。

多対1で大量トランザクションが発生する当社 point+plus のようなシステムにも十分適用が可能であるとの結論に達した。

しかしながら、mijin で実現できる機能以外は周辺システムで吸収する必要があるため、それらの信頼性、可用性を担保することが重要となる。

3 実証実験概要

3.1 期間

実証実験システムの設計・開発・環境の準備
2016 年 6 月 21 日～8 月 12 日
実証実験実施
2016 年 8 月 15 日～8 月 26 日

3.2 対象システム

独自のアララコインを用意し、アララ社内で発生する各種支払いを、アララコインで社員が行えるシステムを想定する（以降アララコインシステムと称する）。

主に、Happy 食堂の支払い、オフィスファミマの支払いを想定している。

※将来的にアララコインをパブリックで利用できる仮想通貨として利用する案もあったが、本実証実験では考慮していない。

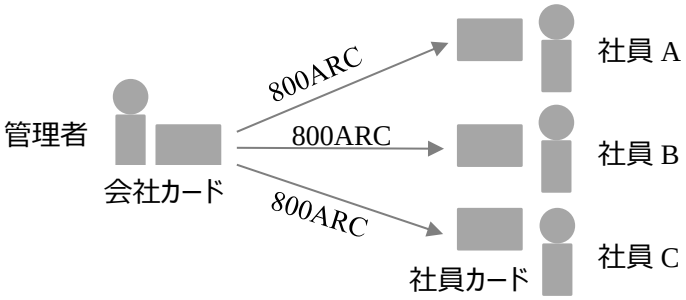
3.3 アララコインシステム

3.3.1 ユースケース

アララコインシステムの主なユースケースは以下の通り。

以下は、小規模システムのイメージ。会社カードは1枚で、会社カード1枚と、全社員間との間で送金、受金を行うことで支払い処理とする。

社員カードへのチャージ（例）

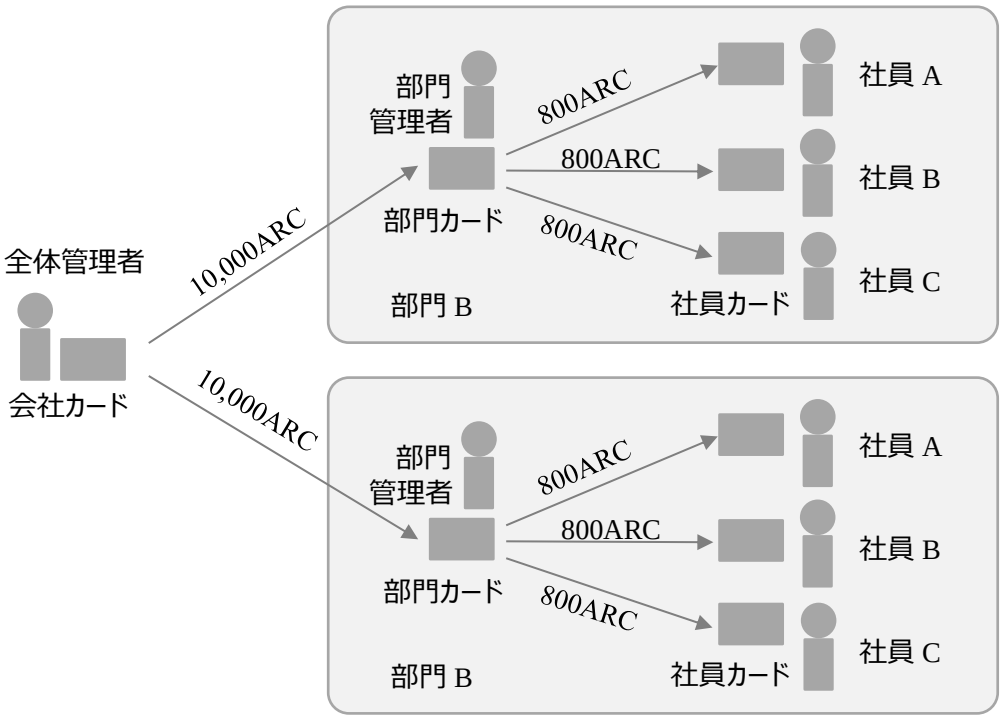


社員利用（例）

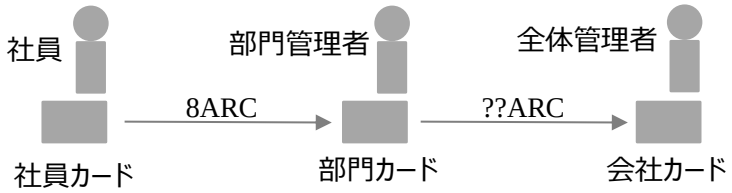
社員は、利用前に Web またはスマホから支払いを行う。

一方、会社規模が大きい場合、部門毎のコントロールができるようにすることと、会社カードへの取引が集中しないようにするため、以下の構成とする。

社員カードへのチャージ（例）



社員利用（例）



社員は、利用前に Web またはスマホから支払いを行う。
部門・会社間は、毎回行わず、必要によりまとめて行う。

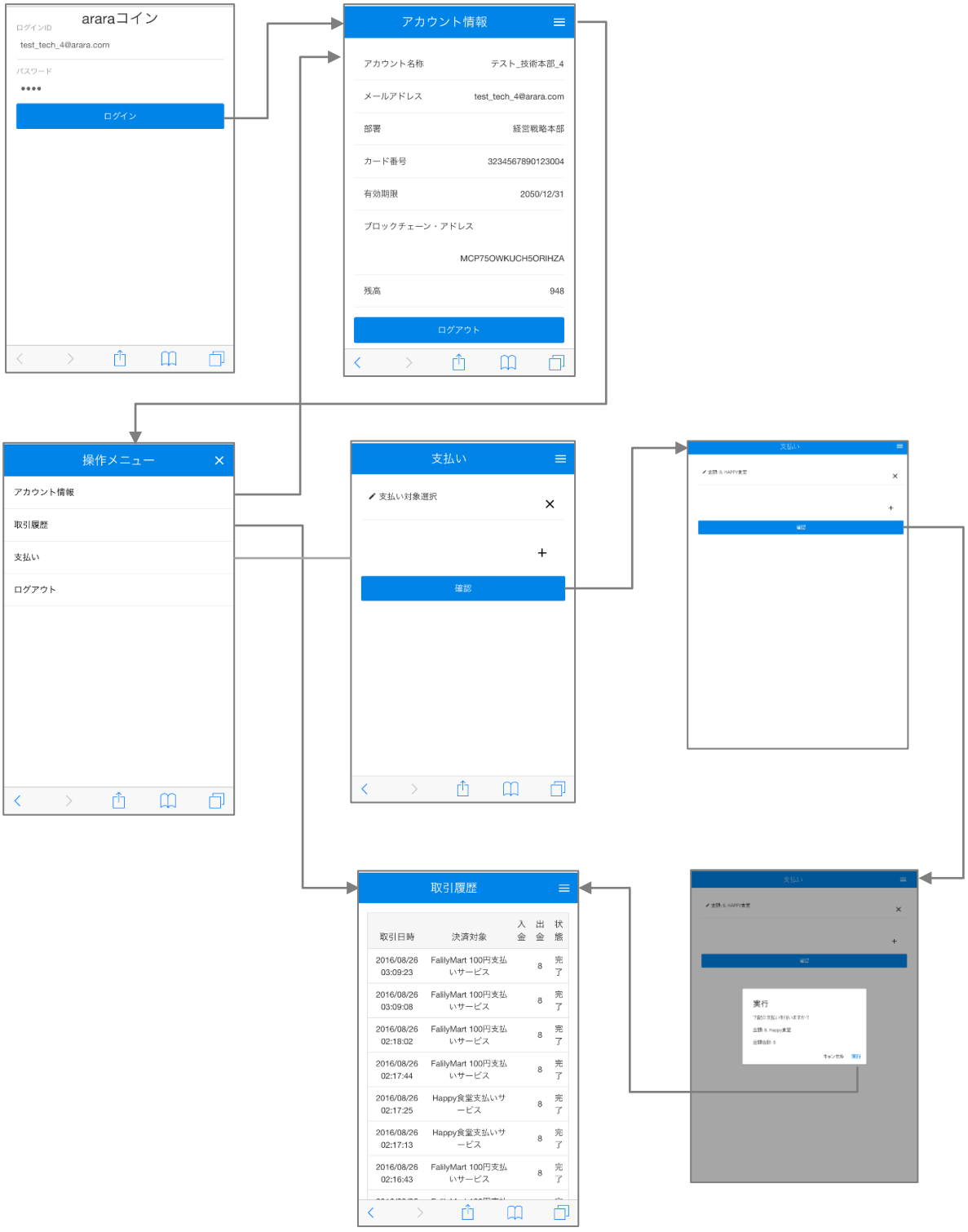
3.3.2 管理系機能一覧

#	機能	内容	実証実験実装
A1	多組織対応	事業所や部署毎に独立した処理、管理が行える	あり
A2	カード発行	独自の番号体系でカードを発行できる。	なし
A3	有効期限	カード毎に有効期限が設定できる。	なし
A4	有効期限延長	利用した場合などに有効期限延長が行える。	なし
A5	ステータス変更	特定カードに対し有効／無効化、有効期限の変更ができる。	なし
A6	チャージ機能	特定カードに対し、組織から送金、受金ができる。	あり
A7	一括チャージ機能	複数の特定カードに対し、組織から送金ができる。	あり
A8	集計機能	期間を指定し、個別カード、個別組織、上位組織毎にモザイク（コイン）の送金・着金の合計が表示できる。	なし
A9	総量調整	コイン（モザイク）総量が不足しそうな時に、追加できる。	なし

3.3.3 ユーザ系機能一覧

#	機能	内容	実証実験実装
U1	スマホ対応	スマホアプリ（iPhone）で利用できる。	あり
U2	スマホ対応	スマホアプリ（Android）で利用できる	あり
U3	支払	100 円、200 円などよく使う支払は、簡単に行え、かつ金額指定での支払いも可能。	あり
U4	集計	月毎のチャージ、支払額が参照できる。	なし
U5	履歴	月毎のチャージ、支払履歴が参照できる。	あり
U6	社員間送金・受金	他社員へアララコインを送金、受金できる。	なし

3.3.4 画面遷移図



4 実証実験詳細

4.1 システム構成

以下に、実証実験で利用したシステム構成図を示す。

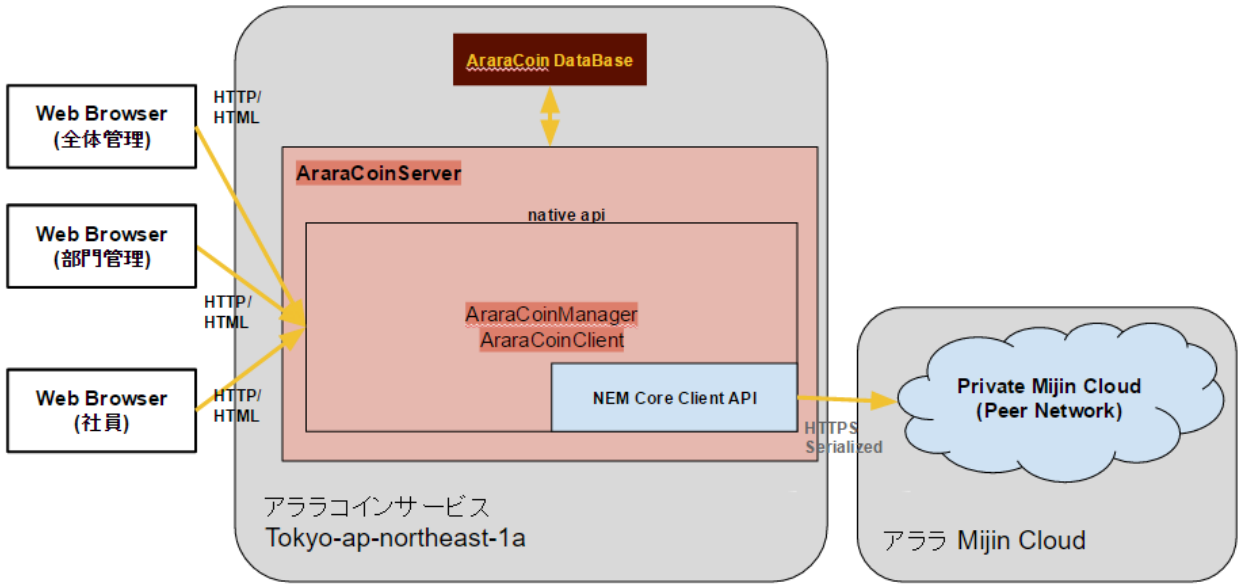


図 4-1 システム構成図全体

表 4-1 アララコインサーバスペック

ロケーション	amazon aws Tokyo-ap-northeast-1a	
インスタンスタイプ	t2.medium	
CPU	2 コア	
メモリ	4G	
ストレージタイプ	SSD	gp2
データベース	PostgreSQL	ver9.5

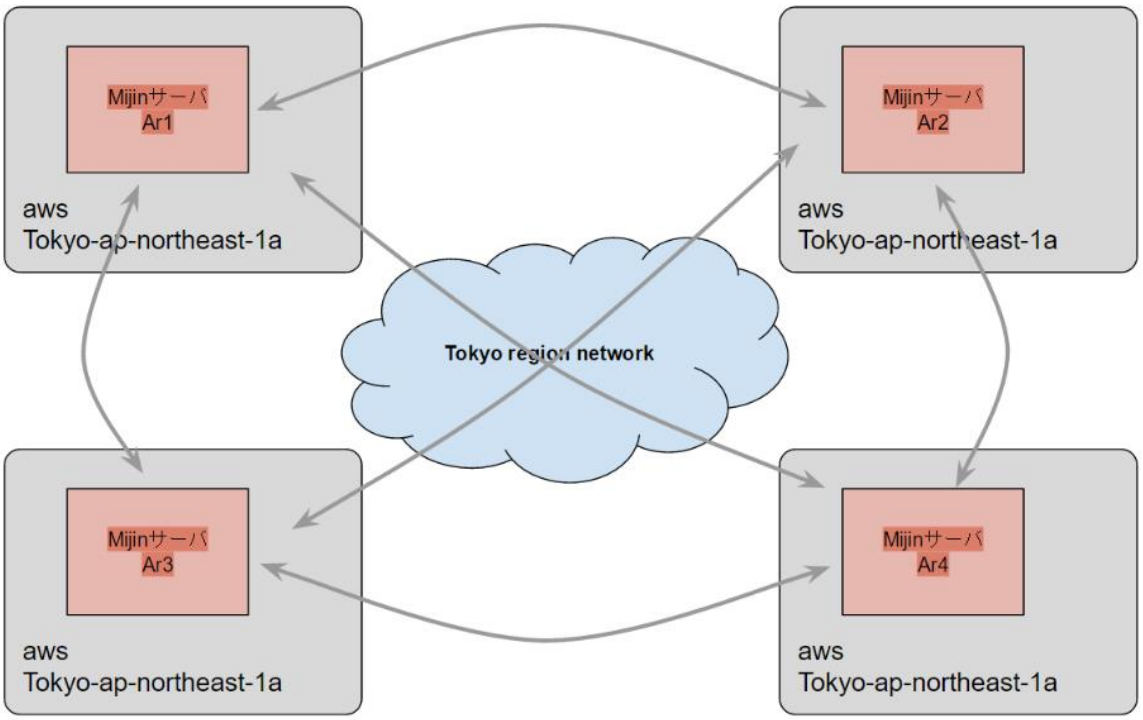


図 4-2 アララ Mijin Cloud 構成図１（レイテンシー低：RTT 平均約 0.5msec）

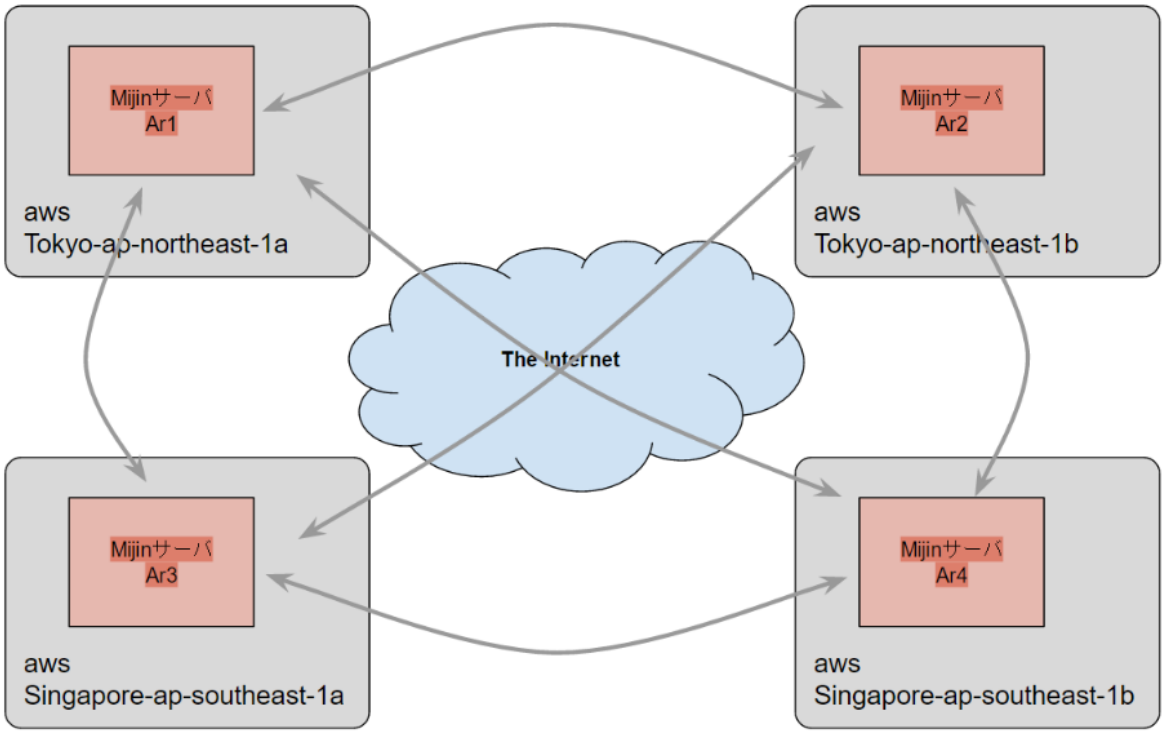


図 4-3 アララ Mijin Cloud 構成図 2（レイテンシー高：RTT 平均約 75msec）

表 4-2 アララ Mijin Cloud サーバスペック

ロケーション	amazon aws Tokyo-ap-northeast-1a Tokyo-ap-northeast-1b Singapre-ap-southeast-1a Singapre-ap-southeast-1b	
インスタンスタイプ	m3.large	
CPU	2 コア	
メモリ	7G	※JavaVM に 7Gbyte を割り当てた場合、プロセスダウンが頻発したため、4Gbyte に設定
ストレージタイプ	SSD	gp2

表 4-3 大量データ取引用クライアントＰＣスペック

ロケーション	アララ本社	
CPU	Intel Corei3 2 コア 4 スレッド 3.1GHz	
メモリ	12G	
ストレージタイプ	HDD(SATA) 7200rpm	

4.2 実証実験のポイント

今回の実証実験は、以下をポイントとして実施した。

1. 整合性
   複数台の mijin サーバへ取引を行った場合に矛盾が生じないか。サーバ間の同期が一時期取れなくなっても矛盾が生じないか。
2. 性能
   １カードに対する処理性能や、多部門での大量処理を行った場合に、実用的な処理能力があるか。
   クライアント→アララコインサービス、アララコインサービス→アララ MijinCloud 間で、ボトルネックが発生しないか。

4.3 事前準備

検証に際し、以下の準備を行った。

ユーザ（カード）数	100,000 名
部門件数	1024 部門
蓄積取引量	約 560 万取引

4.4 実証実験方法

実証実験は、テスト用クライアントにテスト用ツール（今回別途開発したもの）をインストールし、パラメータを変更することで取引を行う。パラメータは以下の通り。

表 4-4 テストパラメータ項目一覧

ユーザ（カード）数	全テストで固定（100,000 名）
部門数	全テストで固定（1,024 部門）
クライアント待ち時間	テストクライアントが次の取引リクエストを投げるまでの待ち時間
クライアントスレッド数	テストクライアントが同時に取引を実行する並列度
サーバ接続モード	複数の mijin サーバにどのように接続するかを指定する。 ・部門・サービス固定モード：部門・サービスが同じ場合は常に 1 台のサーバに接続するモード ・ラウンドロビンモード：順番にサーバに接続するモード
ネットワークレイテンシ	サーバ間ネットワークレイテンシ

テストは、5 分程度上記パラメータで取引を行い、以下のデータを取得する。

サーバ負荷	大よそのロードアベレージ
処理時間	１分毎に処理した取引数
アララコインサーバ処理時間	アララコインサーバでの処理時間。リクエストを受けてから、mijin の取引要求応答が返ってくるまでの時間（未承認状態）
mijin 処理時間	mijin に取引要求を出してから、応答が返ってくるまでの時間（未承認状態）
データ容量	データサイズ（件数に対してどの程度サイズが大きくなるか） ※特定テストケースにて実施

4.5 実証実験内容

4.5.1 サマリ

テスト#	テスト内容
データ整合性確認 1	10 万カードから、1024 部門へ支払を行う。部門に対し、接続する mijin サーバは固定とし、不整合が発生しづらい状況を確認する。カードはランダムに選択する。
データ整合性確認 2	10 万カードから、1 部門へ支払を行う。カードはランダムに選択するが取引を連続で 4 回行い、4 台の mijin サーバへ振り分け、不整合が発生しやすい状況を確認する。
データ整合性確認 3	データ整合性確認 2 の状況下で、数台のサーバをリブートし、その後サーバ間のデータが同期されることを確認する。
性能確認 1	10 万カードから、1 部門へ大量リクエストを行う。カードはランダムに選択し、4 台の mijin サーバへラウンドロビンで振り分ける。
性能確認 2	10 万カードから、1024 部門へ大量リクエストを行う。カード、部門はランダムに選択し、ラウンドロビンで振り分ける。

4.5.2 整合性確認１（１部門・サービスに対し１サーバへの取引）

4.5.2.1 内容

mijin サーバは、4台すべてに同じデータを持っている。すなわち、1台が処理したデータを他の3台に都度同期しているということである。よって、1カードに対し、複数の mijin サーバへ短時間に複数取引を行うと、整合性が取れなくなる懸念があるため、まずは1カードに対し、1サーバへ取引が行われる構成で、整合性が取れることを確認した。

4.5.2.2 システム構成

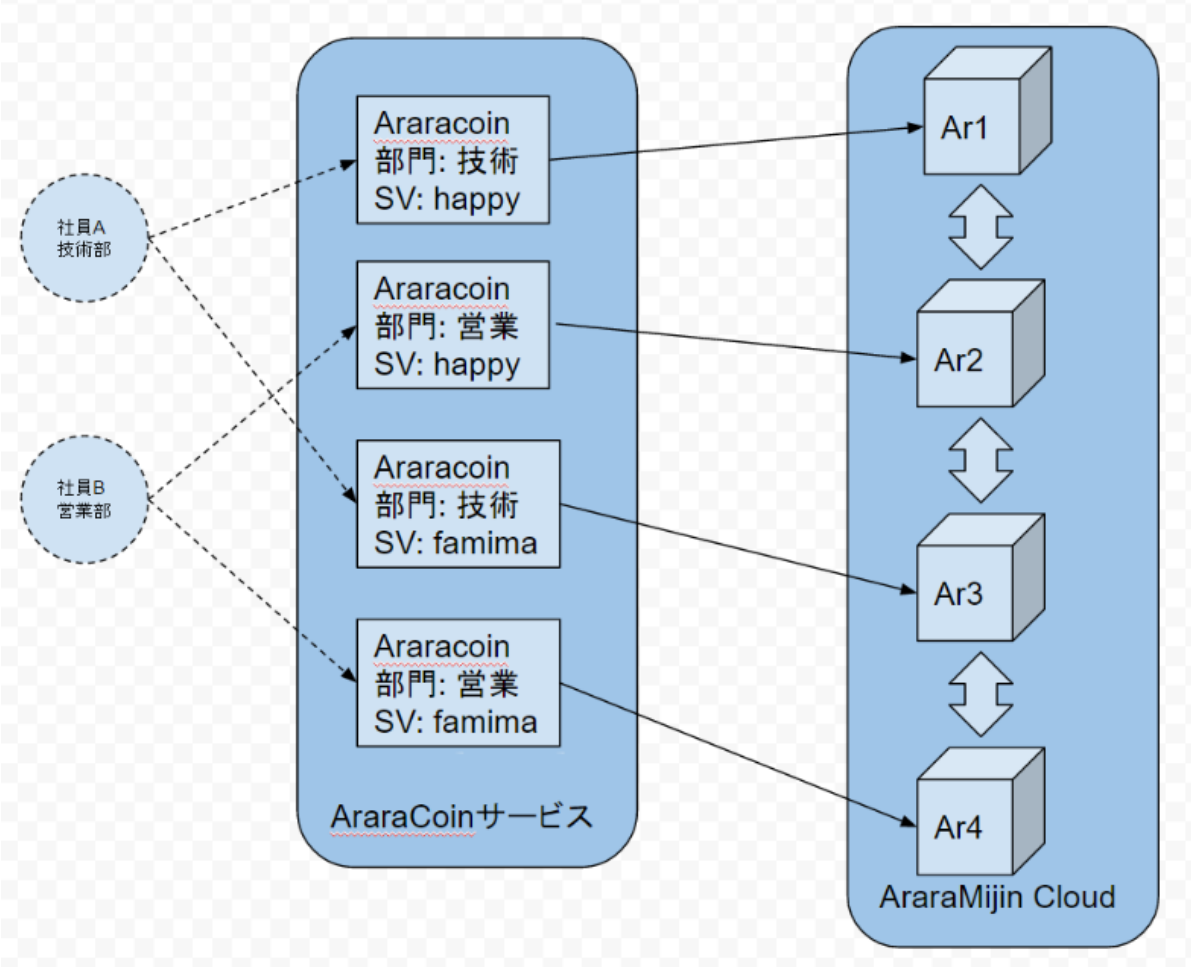


図 4-4 １部門･サービスに対し、１サーバへのリクエスト

4.5.2.3 取引シーケンス

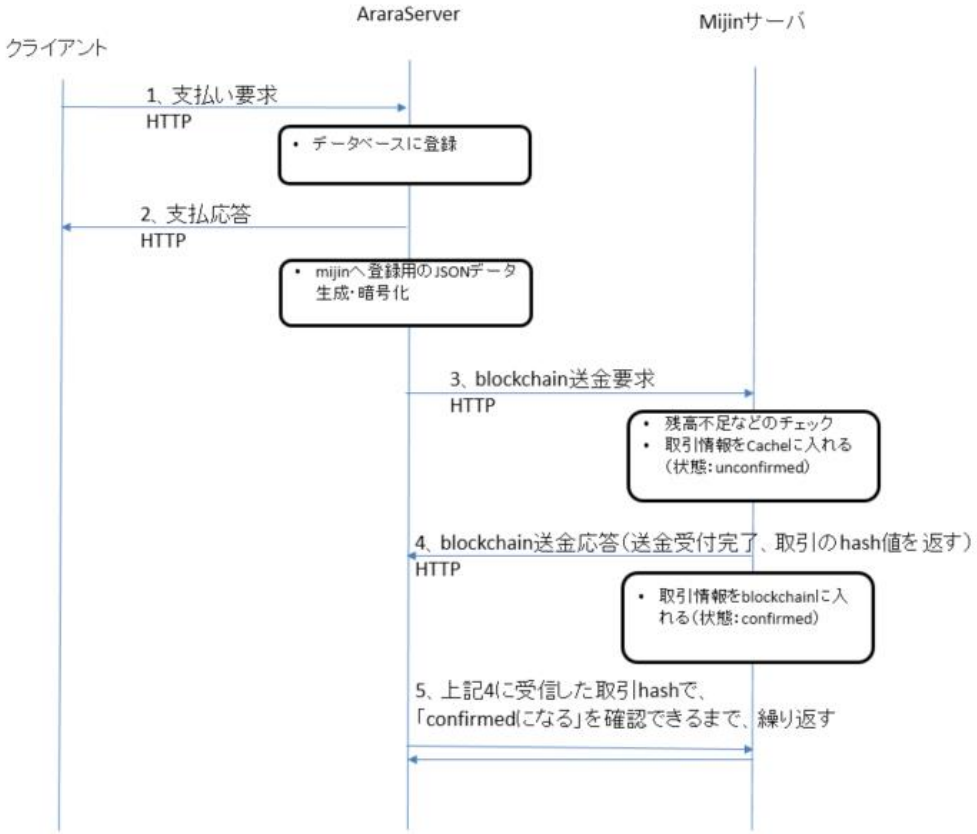


図 4-5 取引シーケンス

4.5.2.4 検証パラメータ

ユーザ（カード）数	100,000 名	
部門数	1,024 部門	
クライアント待ち時間	0msec	
クライアントスレッド数	1	整合性をチェックするため 1 とする
サーバ接続モード	固定	部門・サービス毎に接続サーバは固定

4.5.3 整合性確認 2（1 部門・サービスに対し 4 サーバへの取引）

4.5.3.1 内容

mijin サーバは、4台すべてに同じデータを持っている。すなわち、1台が処理したデータを他の3台に都度同期しているということである。よって、1カードに対し、複数の mijin サーバへ短時間に複数取引を行うと、整合性が取れなくなる懸念があるため、1カードに対し、4サーバへ取引を連続して行う構成で、整合性が取れることを確認した。

4.5.3.2 システム構成

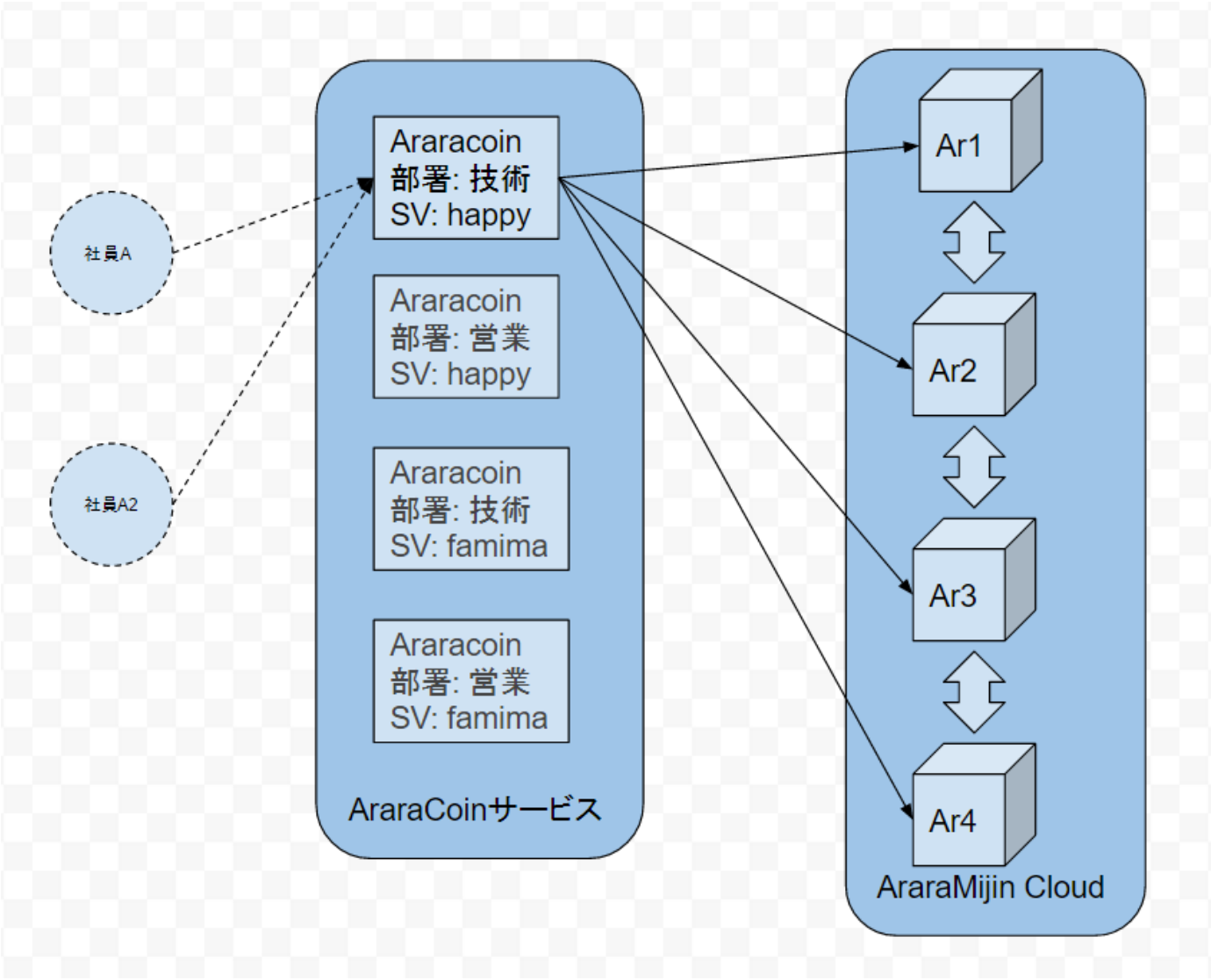


図 4-6 1 部門･サービスに対し、4 サーバへのリクエスト

4.5.3.3 取引シーケンス

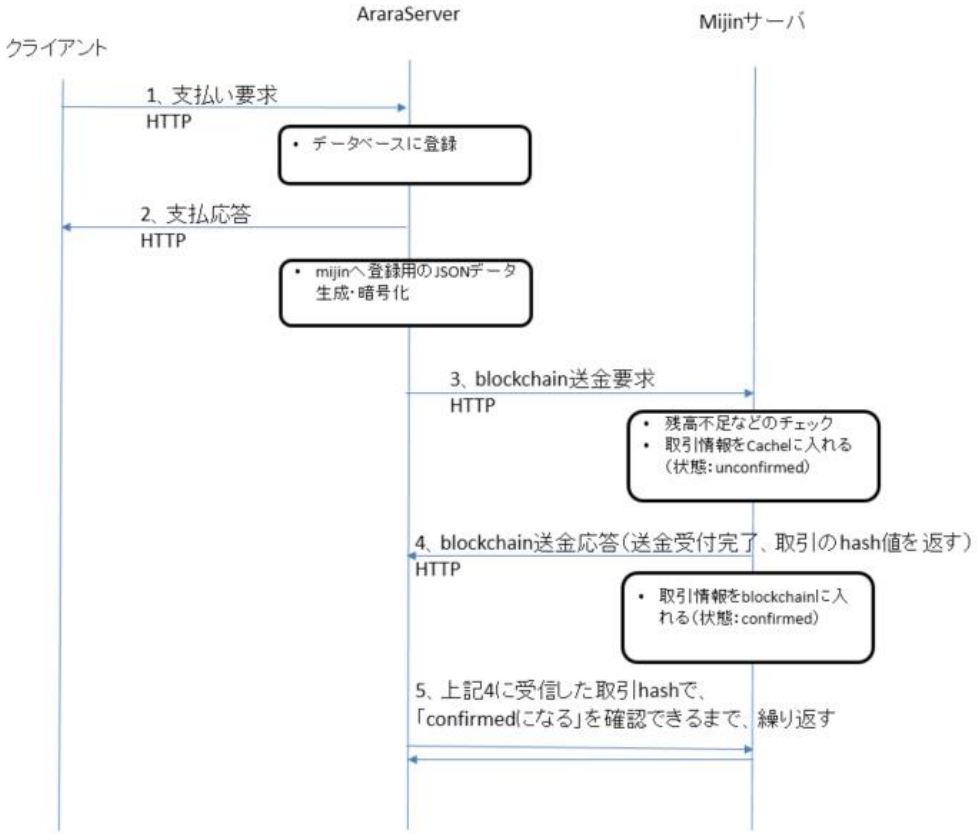


図 4-7 取引シーケンス

4.5.3.4 検証パラメータ

ユーザ（カード）数	100,000 名	
部門数	1 部門	
クライアント待ち時間	0msec	
クライアントスレッド数	1	整合性をチェックするため 1 とする
サーバ接続モード	ラウンドロビン	順番に接続していく

4.5.4 整合性確認 3（サーバのリブート）

4.5.4.1 内容

mijin サーバは、4台すべてに同じデータを持っている。負荷分散や性能を考慮すると、分散接続を積極的に行った方が望ましいと思われる。しかしながら、あるサーバがダウンやリブート、復旧した場合にサーバ間のデータ整合性が取れなくなるケースも考えられ、その場合は手動復旧させる必要が出てくるため、運用負荷が懸念される。よって、サーバがリブートを繰り返した場合でも、整合性が保たれることを確認する。

4.5.4.2 システム構成

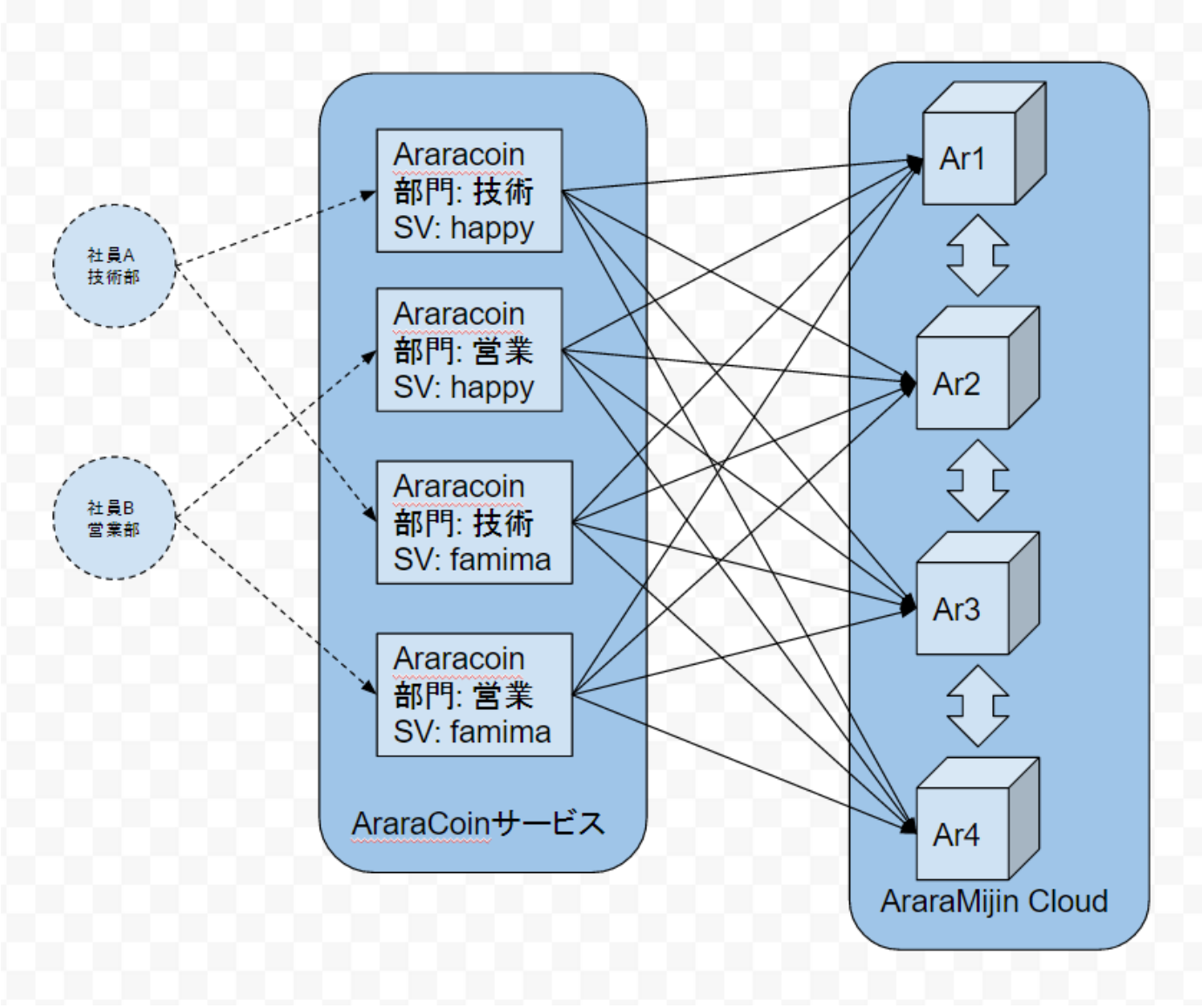


図 4-8 複数部門･サービスに対し、4 サーバへのリクエスト

4.5.4.3 取引シーケンス

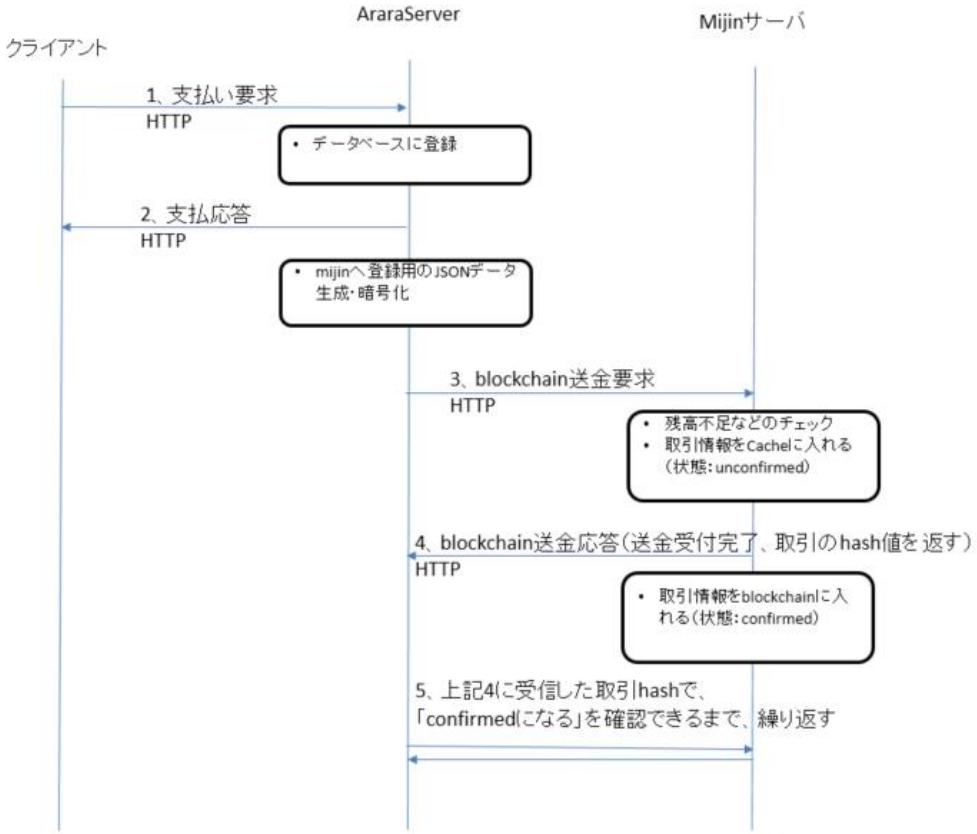


図 4-9 取引シーケンス

4.5.4.4 検証パラメータ

ユーザ（カード）数	100,000 名	
部門数	1 部門	
クライアント待ち時間	0msec	
クライアントスレッド数	1	
サーバ接続モード	ラウンドロビン	順番に接続していく

4.5.4.5 検証方法

上記取引を連続で行っている最中に、4台のうち2台を定期的にリブートする。リブート方法は、mijin が適切な終了を行わないようにするためには電源のハード Off が良いが、時間節約のため mijin プロセスの強制終了(kill -KILL)にて実現することとした。

4.5.5 性能確認１（１部門への取引）

4.5.5.1 内容

部門を考慮せず、1カードに対し集中取引を行った場合、整合性の確認などで性能に影響が出る可能性を考え、性能を確認した。

4.5.5.2 システム構成

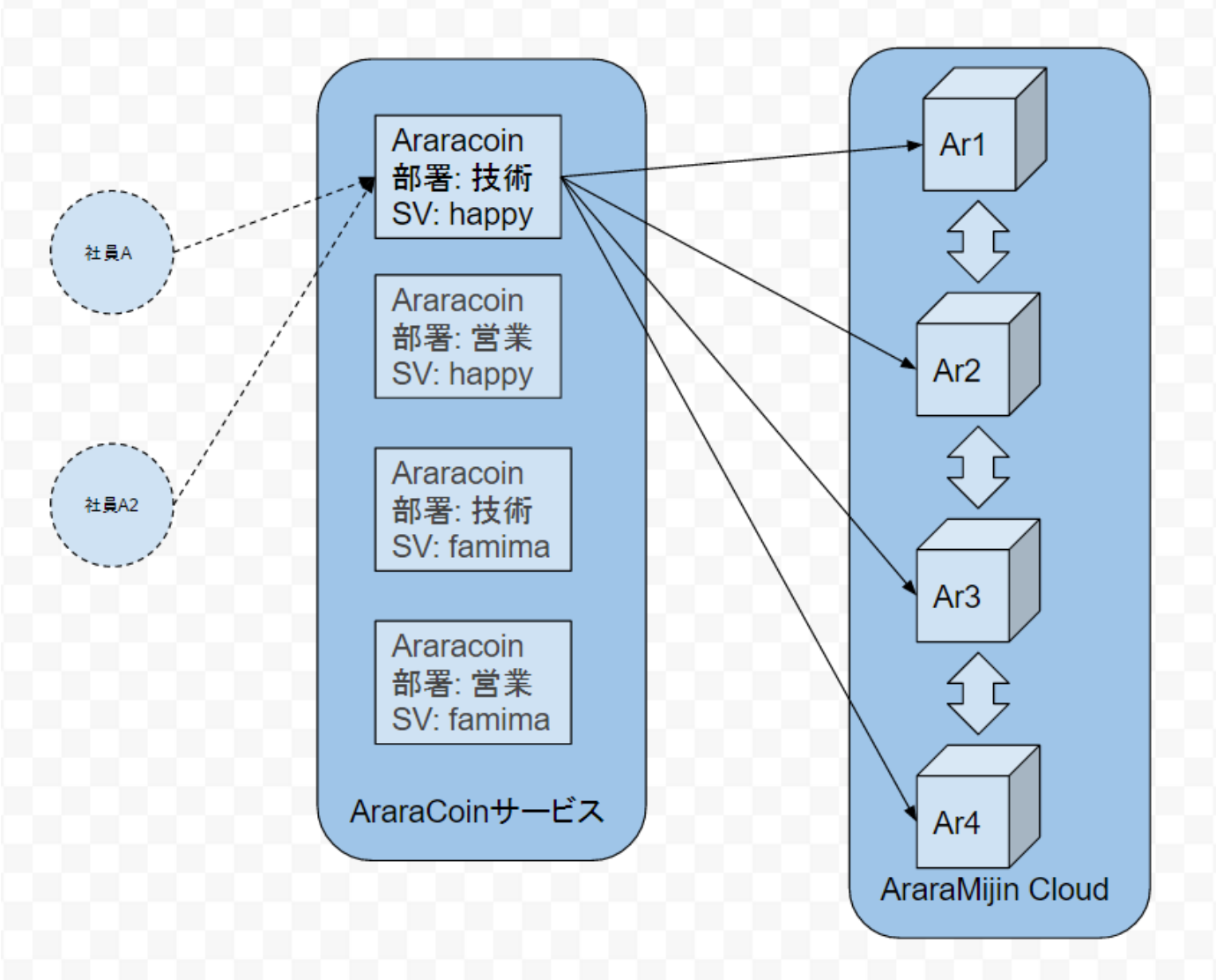


図 4-10 １部門・サービスに対し、4 サーバへのリクエスト

4.5.5.3 取引シーケンス

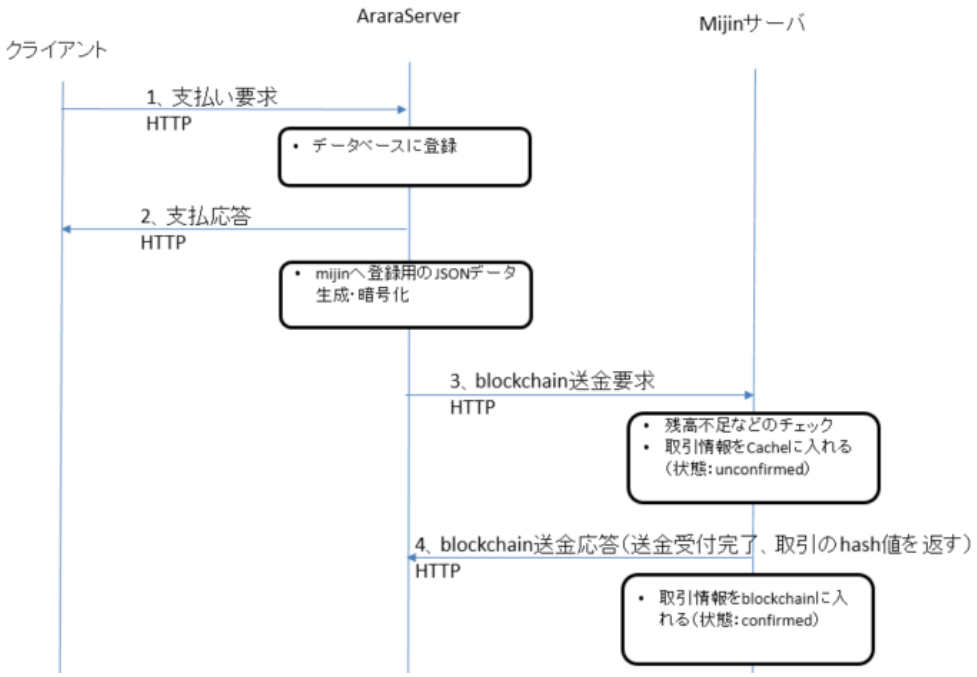


図 4-11 取引シーケンス

4.5.5.4 検証パラメータ

ユーザ（カード）数	100,000 名	
部門数	1 部門	
クライアント待ち時間	0msec	
クライアントスレッド数	10	性能を確認するため 10 とする
サーバ接続モード	ラウンドロビン	順番に接続していく

4.5.1 性能確認 2（複数部門への取引）

4.5.1.1 内容

部門を考慮せず、1カードに対し集中取引を行った場合に対し、複数部門で取引を分散させた場合の性能差異を確認した。

4.5.1.2 システム構成

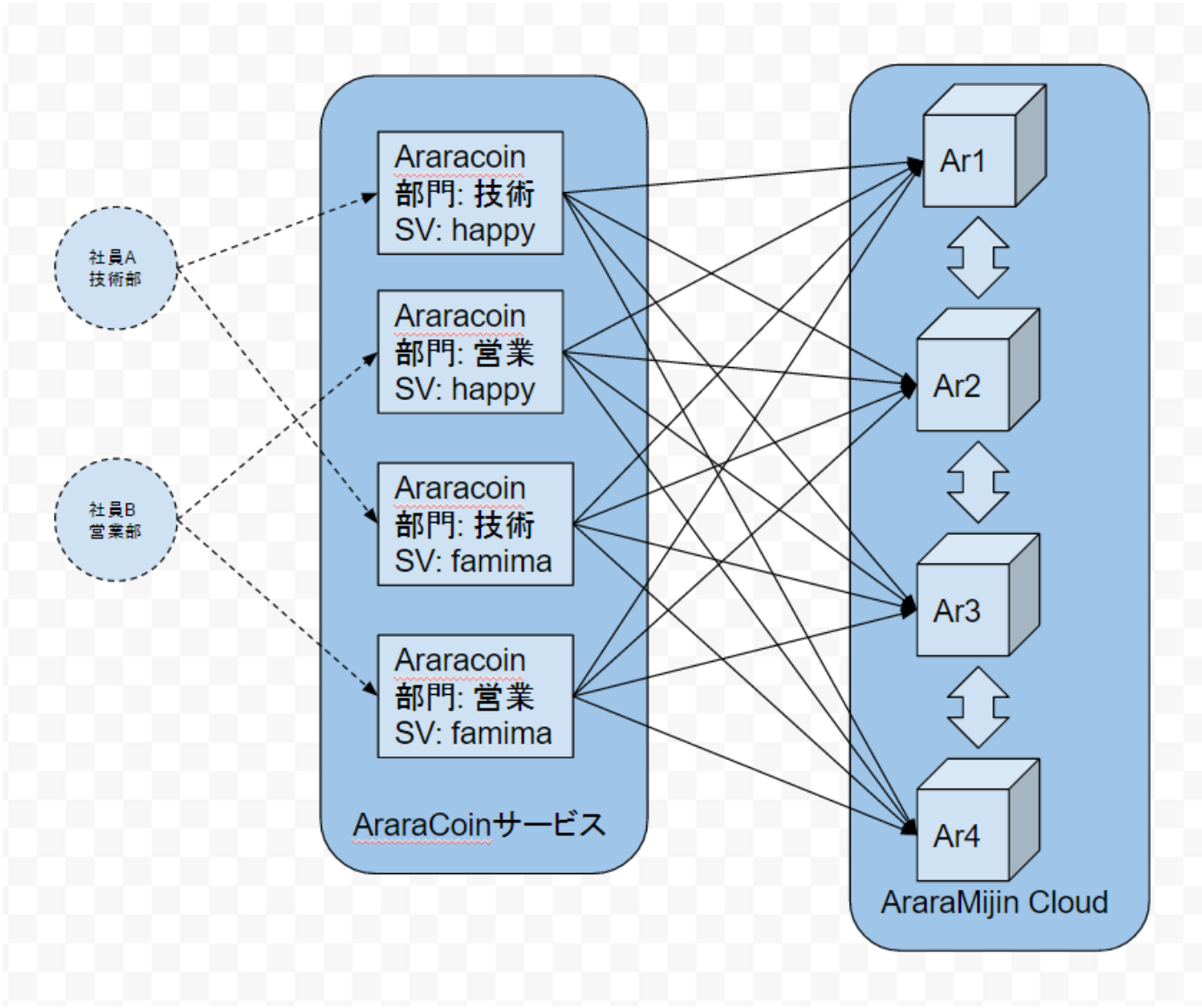


図 4-12 １部門･サービスに対し、4 サーバへのリクエスト

4.5.1.3 取引シーケンス

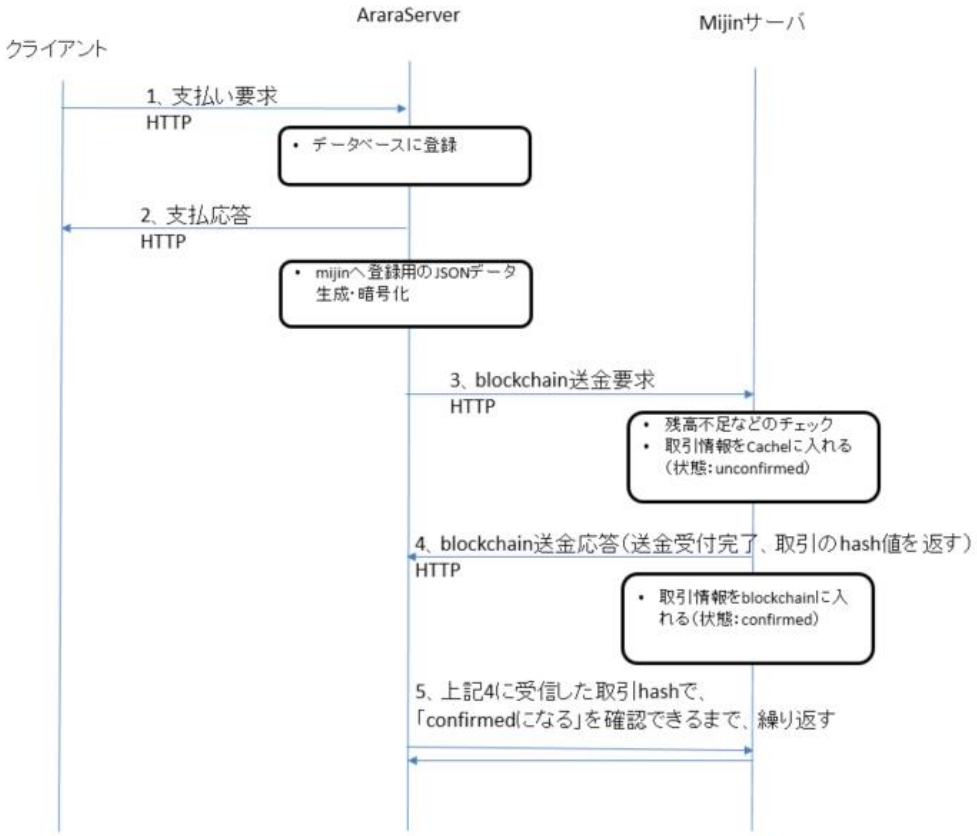


図 4-13 取引シーケンス

4.5.1.4 検証パラメータ

ユーザ（カード）数	100,000 名	
部門数	1,024 部門	
クライアント待ち時間	0msec	
クライアントスレッド数	10	性能を確認するため 10 とする
サーバ接続モード	ラウンドロビン	順番に接続していく

5 実証実験結果と考察

5.1 データ整合性

どのテストケースについても、整合性が崩れ、承認できないケースは確認できなかった。しかしながら、現在の mijin はモザイク（アララコイン）で取引を行う場合に必ず手数料(xem)が取られるため、xem の残高不足でエラーが数回発生する事象があった。これは、標準でパブリックな環境に置いた場合に、悪意のある API 攻撃対策のため手数料を課しているだけで、設定で手数料を 0 にすることが可能であるとの回答を、テックビューロ社から得ているため、特に問題ではないと判断している。

また、テスト 4.5.3 整合性確認 2（1部門・サービスに対し 4 サーバへの取引）を実施した際、取引日時、送信者、受信者、取引額が同じ場合に重複取引としてニュートラルが返される現象が多く発生した。これは、同じ時間に同じような取引が実施された際、mijin 側で重複取引を防止する機構が働いたものと見られる。取引額などを都度変更し取引を行うようなテストを行う必要があったと思われる。

耐障害性については、取引実施中に定期的にリブートを行ったが、復帰後正常に同期が完了したことをログで確認し、復帰したサーバで過去取引が確認できることを確認できたため、特に問題となることはなかった。

5.2 性能

5.2.1 サーバ負荷

アララコインサーバと mijin サーバとのネットワーク距離が短く、レイテンシーが小さい場合、どのテストケースについても、mijin サーバのロードアベレージが 2 を超えることは無かった。よって、秒間百程度の取引であれば、問題ないレベルでの処理が可能と考えられる。

一方、アララコイン側のAPサーバ側は mijin サーバに比べて多少負荷が高くなった。原因として、mijin へのリクエストデータを作成するのにハッシュの計算や署名を行うためにCPUパワーが使われたものと思われる。

また、まれにアララコインサーバ側でランタイムエラーが発生した。原因不明であるが、発生率は極めて低いことから、再試行することなどで対処可能と思われる。

5.2.2 取引承認確認処理

当初、下図 5.の承認チェックを行うことをすべてのテストで前提としていたが、対象の取引の確認リクエストを行うと、クライアント側でランタイムエラーが高確率で発生することが分かった（約 80%で発生）。

これは、対象取引を含めたそれ以前の取引データが全部で 20 件返却される仕様となっており、1取引毎にチェックを行うと、取引数×20 件のデータが取得、返却されるため、この辺りのロジックで処理が遅くなりエラーが発生していると推測される。今後の課題と思われる。

※以上からすべてのテストはこのチェックを行わない状態で行った。

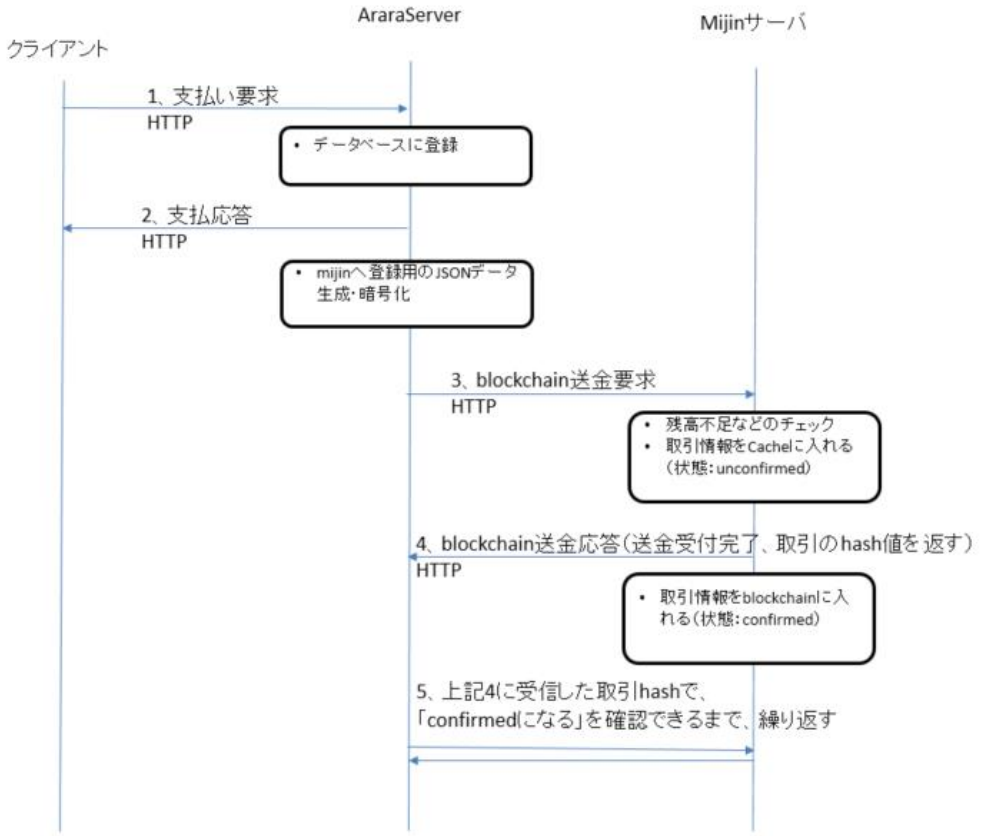


図 5-1 取引シーケンス

5.2.3 取引性能

どの処理も性能的にはそれほど変わらなかった。多カード対1部門の構成で分間約 3,000 取引、多カード対多部門の構成で分間約 3,800 取引となった。テストデータのばらつきを考えると、この差が mijin の特性によるものなのかを断定することはできないと考える。

公称値は秒間数万との報告もあり、今回の結果は低い結果となったが、原因としては、どちらのテストもサーバ負荷が高くないことから、クライアント側の処理性能不足、もしくはアララコインサーバ側のスレッド数等の最適化ができていなかったことなどが考えられる。よって、さらに多くの取引を行うことができる可能性はあるが、本テストでは分間 3,800 取引、時間 22 万取引は行えることが分かった。

表 5 取引性能（図 4-2 アララ Mijin Cloud 構成図１（レイテンシー低：RTT 平均約 0.5msec））

	多カード対1部門	多カード対多部門
取引数（分間）	約 3,000 取引	約 3,800 取引
アララAPI時間（シーケンス 2 直後～4）	約 600msec	約 700msec
mijin 応答時間（シーケンス 3～4）	約 10msec	約 10msec
mijin リクエスト遅延	なし	なし
承認時間	測定不能（数秒内には完結しているように見えた）	

レイテンシーが高い環境でのテストは奇妙な結果となった。mijin 応答時間は約5倍になっており、レイテンシーが 150 倍程度になっていることを鑑みると特に違和感はない。一方、アララAPI時間が約 310 秒かかっており、非

常に時間がかかっている。しかしながら、サーバの負荷は全くないため、どこかにボトルネックが存在すると推測、原因調査を行った。

結論としては、アララコインサーバで利用した NEM 開発メンバが公開しているクライアントプログラムサンプルと、NEM Core Client API において、mijin サーバに必要以上の負荷を掛けない仕組みが組み込まれており、クライアント側にリクエストキューとして蓄積してしまっていたことが原因であった（SleepFuture の利用）。

レイテンシーが小さい環境では、それでも分間 3,000 のリクエストを遅延なく処理できていたが、レイテンシーが大きい環境では、分間 3,000 のリクエストを処理しきれず、クライアント側にリクエストキューが蓄積され、処理が平均 310 秒という結果になってしまったものと思われる。

この問題を改善するためにプログラムの修正を行い、再度実施してみたところ、レイテンシーが大きい環境でもレイテンシーが小さい環境と遜色ない結果を得ることができた。

表 6 取引性能（図 4-3 アララ Mijin Cloud 構成図 2（レイテンシー高：RTT 平均約 75msec））
プログラム改修なし

	多カード対1部門	多カード対多部門
取引数（分間）	約 3,300 取引	約 3,400 取引
アララAPI時間（シーケンス 2 直後～4）	約 310sec（増加傾向）	約 310sec（増加傾向）
mijin 応答時間（シーケンス 3～4）	約 50msec	約 45msec
mijin リクエスト遅延	あり	あり
承認時間	測定不能（数分内には完結しているように見えた）	

表 7 取引性能（図 4-3 アララ Mijin Cloud 構成図 2（レイテンシー高：RTT 平均約 75msec））
プログラム改修あり

	多カード対1部門	多カード対多部門
取引数（分間）	計測なし	約 3,100 取引
アララAPI時間（シーケンス 2 直後～4）	計測なし	約 114msec
mijin 応答時間（シーケンス 3～4）	計測無し	約 108msec
mijin リクエスト遅延	なし	なし
承認時間	計測無し	測定不能（数分内には完結しているように見えた）

6 まとめ

（1）使い方により整合性が崩れるようなことはなく、時間 20 万件程度の取引には耐えられることが分かった。
（2）改ざんが困難、かつ、データ同期の仕組みにより、データの完全性が保証されることが分かった。
（3）分散システムにより、高い可用性を有していることが分かった。
（4）電子マネーシステムへの実用可能性は、整合性、性能の面から十分にあると思われる。
（5）上記（1）～（4）を踏まえると、大幅なコストダウンの可能性も見込まれる。

－以上－