分散型メタバースサーバにおける ログイン時のレスポンス性能に関する評価

松原　麻佑†　　小口　正人†

† お茶の水女子大学　〒 112–8610 東京都文京区大塚 2-1-1
E-mail: †mayu@ogl.is.ocha.ac.jp, ††oguchi@computer.org

あらまし　将来的な普及が期待されている「メタバース」は，電子データとして構築された仮想３次元空間に，インターネットを通じて接続し利用する新しいサービスである．このメタバースサービスの高性能化のために，サーバのレスポンスに関する評価を行う．具体的には，メタバース実行時のサーバ側の実測評価に焦点を当て，ログイン時のサーバの動作解析等を通し，ボトルネックに関する考察を行う．

キーワード　メタバース，Second Life，分散型サーバ，OpenSim，SSD，Ganglia

An Evaluation of Response while User Login in Distributed Metaverse Server

Mayu MATSUBARA† and Masato OGUCHI†

† Ochanomizu University
2-1-1 Otsuka, Bunkyo, Tokyo, 112–8610 Japan
E-mail: †mayu@ogl.is.ocha.ac.jp, ††oguchi@computer.org

1. はじめに

近年，一般家庭におけるブロードバンドネットワークの普及や PC の性能向上によって，ユーザ主導のインターネット文化が形成されている．それにより，ユーザ同士のネットワークを通じたコミュニケーションや情報提供手法の発展が強く求められてきている．本研究で注目したメタバースというサービスは，インターネット上に構築された現実世界に似た仮想３次元空間を指す．その空間に接続することで，よりリアルに，より容易にコミュニケーションを実現出来る場所を提供するものである．

ブログや SNS が浸透し，ユーザが情報サービスに対して受け身ではなく，発信者であり創作者になっている潮流から，メタバースサービスは大いに注目されている．世界最大規模のメタバースサービスとしては，米 Linden Reserch 社が提供している Second Life が有名であるが [1]，日本でも，(株) サイバーエージェントがアメーバピグというサービスを提供し始めており，多くの利用者を獲得している [2]．またこの他，IMVU [3]，Moshi Monsters [4]，Meet-Me [5] などのメタバースが知られている．一方で，クライアント端末，特にそのグラフィクスに要求されるスペックがある程度高いこと(注1)，レスポンスが遅いことから，期待されている程は普及していないという現実が挙げられる．そのため本研究では，クライアントを含めたメタバースシステムにおけるレスポンス短縮を目的とし，そのボトルネックの解析を行う．

2. メタバース

2.1 メタバースの概要

メタバースとは，インターネット上に作られた仮想空間サービスの総称である．サービスの利用者は，この空間内でアバタという自分自身の分身を操作し，仮想空間の中での創作活動や他の利用者とのコミュニケーションが可能である．例として，メタバース内で流通する独自通貨を利用し，アイテム（アバタの衣服や乗り物，家など）の購入や販売をする事が可能なサービスや，企業内でのコミュニケーションに適したサービスなどがある．メタバースに類似したサービスとしてオンラインゲームが挙げられるが，メタバースは，サービス運営元の用意した

(注1)：Second Life クライアントビューアの推奨動作環境は CPU が 1.5GHz 以上，メモリが 1GB 以上，グラフィックカードが Nvidia は 9600 以上，ATI は 4850 以上などとなっている

目標（敵を倒すなど）はなく，利用者が主体となって，目的にあった活動をする．

2.2 Second Life

Second Life は世界最大規模のメタバースサービスで，その名の通り,「第２の人生」を楽しむ場所の提供として位置づけられている．その特徴としては，視覚的に美しいグラフィクスを提供している，スムーズなインタフェースでモノ作りが可能である，作成したオブジェクトの売買や，Second Life 内で有効な通貨であるリンデンドルを US ドルに換金することが可能であるという点が挙げられる．ただし現実世界と大きく異なる点として，この空間内では物理的な制約を一切受けない．アバタは自由に空を飛ぶことができ，一瞬にして世界中の土地にアクセス出来る．Second Life を運営している Linden 社は，この Second Life のサーバソフトを現段階では非公開としているが，一方でクライアントビューアのコードを公開し始めた．

2.3 OpenSim

OpenSim はメタバースサーバのオープンソフトであり，いわば仮想世界を構築するためのプラットホームである [6]．Second Life のサーバとは別物であるが，公開されている Second Life のクライアントビューアを参考にして作られたサーバであるため，Second Life のクライアントビューアからこの OpenSim サーバに接続可能であり，Second Life サーバとの互換性が保証されている．またこの OpenSim はオープンソフトであるため，自分が所有しているサーバ機で構築が可能であり，カスタマイズも容易に出来る．

更に OpenSim は，スタンドアロンモードとグリッドモードの 2 つの動作モードで構築可能である．スタンドアロンモードは OpenSim サーバを 1 つのデスクトップアプリケーションとして起動でき，容易に仮想世界サービスを立ち上げることが出来るものである．OpenSim は Second Life と共通のデータフォーマットを使用しており，現在多くのデザイナーたちが島を制作する際，OpenSim のスタンドアロンモードで試作品のプレビューを行っている．スタンドアロンモードはその特性を生かし，3D プレゼンテーションツールとしての利用や，病院，遊園地の案内などに使用出来る．しかしスタンドアロンモードは設定が簡単ではあるが拡張性がないため，複数の OpenSim サーバの相互接続や大規模な仮想世界サービスの構築には向いていない．

一方グリッドモードは，Second Life のような広大な仮想世界サービスを構築するための動作モードである．グリッドモードでは，離れた場所にある OpenSim サーバ同士を相互接続させ，複数の島からなる巨大仮想空間をシームレスに構築出来る．これにより島がそれぞれ別の OpenSim サーバにあっても，隣の島に移動したり離れた島にテレポートをしたりすることが出来る．アーキテクチャとしては，複数のサーバ機が協調してサービスを提供する分散型メタバースサーバとなる．

2.4 既存研究

文献 [7] では Second Life のクライアントのビューアの性能評価が行われており，これによると Second Life の実行中，クライアントの負荷は相当高くなっている．また，文献 [7] では，サーバ側の解析としてはシュミレーションや数値解析のみが行われている．本研究では，メタバース実行時のサーバとクライアントにおける実測結果を基にサーバの動作解析を行い，メタバースシステムの評価を行う．

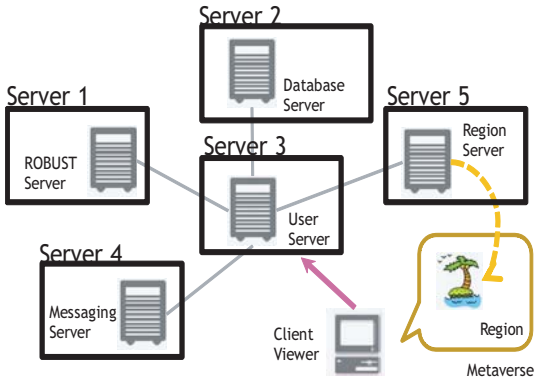


図 1　分散型メタバースサーバ

3. 研究内容

本稿では，メタバースが将来 Web のように普及することを期待しているため，メタバースサーバを分散型で構築し，評価を行っている．分散型サーバは，広域な範囲で通信されることが一般的である．そこで本稿では，メタバースサーバを個々のサーバモジュールで分散させるだけでなく，広域分散環境の下での評価を行う．その後，個々のサーバモジュールのログイン時の働きとその連携がどのように行われているか調べ，レスポンスの遅い理由として考え得る可能性について，検証を行う．

4. 実験環境

分散型サーバは，データのやりとりがリンクを構成しており，分散したサーバ間で情報が有機的に結びつけられたシステムである．Web サーバはこの分散型サーバの典型例と言える．

OpenSim などメタバースにおける分散型サーバは，サーバ同士のリンクが張られている．OpenSim サーバのどこかにクライアントはアクセスをしており，クライアントが欲しいデータは，あちらこちらに散らばった OpenSim サーバのどこかに置かれている．

OpenSim0.6.8 を用いてメタバースサーバをグリッドモードで構築すると，User サーバ，ROBUST サーバ，Messaging サーバ，Region サーバ，Database サーバの 5 つのサーバモジュールに分かれ，役割を分散させることが出来る．これらのサーバモジュールは，同一のマシン上で実行することも，異なるマシン上で実行することも可能である．User サーバは，ユーザ（アバタ）の管理サーバ，ROBUST サーバは，オブジェクトとアバタの持ち物とグリッドの位置などを管理するサーバ，Messaging サーバはメッセージ処理を行うサーバ，Region サーバは，リージョン（土地，一般には 256m x 256m の領域）を管理するサーバである．

本研究での分散型メタバースサーバの実験環境を，図 1 に示す．各サーバに，各サーバモジュール（User サーバ，ROBUST サーバ，Messaging サーバ，Region サーバ，Database サーバ）

表 1　サーバとクライアントのスペック

	サーバ	クライアント
OS	Fedora Core12	Windows XP
CPU	Intel Xeon 3.6GHz、2.4GHz	Intel Pentium 4 CPU 3.00GHz
Memory	4GB、512MB	2.5GB
OpenSim	0.6.8	—
Graphic	—	Intel 82945G Express Chipset Family

を構築した．また実験で使用した各サーバとクライアントのスペックを表 1 に示す．クライアントは一般的な性能のものを使用している．

5. 広域分散環境における分散型メタバースサーバの評価

分散型サーバは，広域な範囲で通信されることが一般的である．そこで，本稿では，メタバースサーバを個々のサーバモジュールで分散させるだけでなく，広域分散環境の下でのレスポンス測定を行った．図 2 に示すように，クライアントまたはサーバ機 1 台のみを dummynet を挟んで構築し，片道遅延時間を 0ms から 100ms まで変化させた．

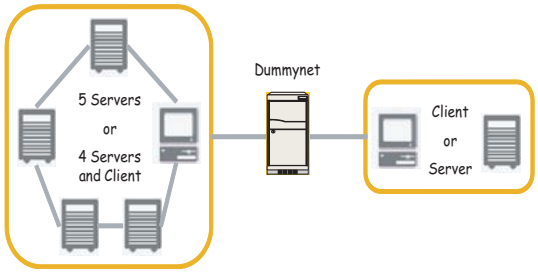


図 2　Dummynet を用いた広域分散環境

5.1 分散型サーバにおける広域分散環境時のテレポート時間の評価

メタバースサービスが普及し，現在の Web のような広がりを見せた場合，巨大なメタバース空間が構築されることになる．その際に，物理的に分散したサーバ間で，ある島から他の島へテレポートする機会が増えると予測し，分散型メタバースサーバにおけるテレポート評価を行う．図 2 に示すように，テレポート先のリージョンサーバに dummynet を挟んで構築した．片道遅延時間を 0ms から 100ms まで変化させ，テレポート時間を測定した．今回の測定では，3 つのシナリオを用意した．teleport1 は，メタバース空間内で隣の位置にある島へのテレポート，teleport2 は，メタバース空間内で 3 つ先の島へのテレポート，teleport3 は，メタバース空間内で 10 先の島へのテレポートである．

結果を図 3 に示す．teleport1 と teleport2，3 を比較すると，テレポート先によりテレポート時間に差がでることが明らかになった．しかしその差は僅かであり，OpenSim システムにおいて，広域分散環境によるテレポートへの影響は少ないと言える．

5.2 分散型サーバにおける広域分散環境時のログイン時間の評価

次に，誰もが OpenSim システムを使用する際に必ず行う，ログインの実行における時間の測定を行った．本研究ではログイン時間を，ユーザがログインボタンを押してから，実際に OpneSim システムが使用出来るようになるまでの時間と定義

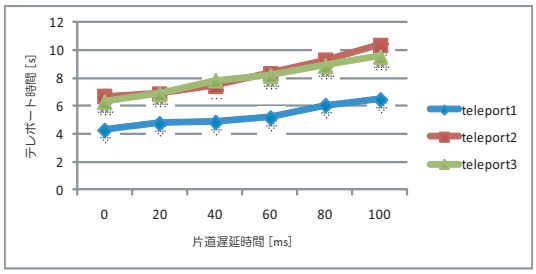


図 3　広域分散環境における分散型メタバースサーバ構築時のテレポート時間

する．図 4 から，ユーザサーバにおいては，0ms と 100ms で差が出ているが，その差は僅かであり，OpenSim システムにおいて，広域分散環境によるログイン時間への影響は少ないと言える．一方で，元々のログイン時間は，最速の場合でも約 54 秒かかっている．ログインまでの時間としては，Web を基準に考えると明らかに長すぎる．

そのため本研究では，メタバースサーバの元々のログインレスポンス短縮を目的とし，OpenSim システムにおけるログイン時のボトルネックの解析を行う．

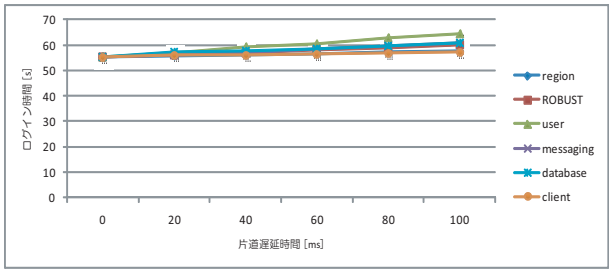


図 4　広域分散環境における分散型メタバースサーバ構築時のログイン時間

6. ログ解析によるサーバモジュールの働きと連携の検証

分散型 OpenSim サーバは複数台のサーバマシンが協調動作を行っているため，サーバのログイン処理においても，振舞が複雑である．つまり，様々なボトルネックが存在し得ると考えられる．そこで，個々のサーバモジュールのログイン時の働きとその連携を検証する．

前述した通り，OpenSim0.6.8 を用いてメタバースサーバをグリッドモードで構築すると，User サーバ，ROBUST サーバ，Messaging サーバ，Region サーバ，Database サーバの 5 つのサーバモジュールに分かれる．ユーザのログイン処理中，この 5 つのサーバモジュールには図 5 のようなログが出力される．これらの表示されたログの解析を行った．具体的には，全てのサーバモジュールから得られたログを時間でソートし，[9] と [10] を参考にしつつ，処理の順序を追った．

ログイン時の処理手順を示す．OpenSim サーバは，ログイン時に大きく 2 つの処理を行っている．認証処理とアバタの初期化である．まず，認証処理について，その手順を図 6 と共に示す．

server	time	process
user1	11:34:43	[LOGIN BEGIN]: XMLRPC Received login request message from user 'TEST' 'AVATAR'
user2	11:34:43	[LOGIN]: XMLRPC Client is Second Life Developer 2.0.0.203055, start location is last
user3	11:34:43	[LOGIN]: Telling TEST_SIM2 @ 1000,1000 (http://192.168.10.248:9000/) to prepare for client connection
user4	11:34:43	[LOGIN]: Found appearance for TEST AVATAR
user5	11:34:43	[USER AUTH]: Verifying session fb7df435-e970-c380-13c6-0a8d1d8ec1b3 for cf341ad0-f306-42e4-90a4-58a2
user6	11:34:43	[UserManager]: CheckAuthSession TRUE for user cf341ad0-f306-42e4-90a4-58a227608ecd
robust1	11:34:43	[INVENTORY SERVICE]: Getting inventory skeleton for cf341ad0-f306-42e4-90a4-58a227608ecd
user7	11:34:43	[MSGCONNECTOR]: Sending login notice to registered message servers
user8	11:34:43	[USER SERVER FRIENDS MODULE]: BEGIN XmlRpcResponseXmlRPCGetUserFriendList from 192.168.10.250:3
user9	11:34:43	[USER SERVER FRIENDS MODULE]: END XmlRpcResponseXmlRPCGetUserFriendList from 192.168.10.250:359
user10	11:34:43	[LOGIN]: Notified : http://192.168.10.250:8006 about user login
user11	11:34:43	[LOGIN END]: XMLRPC Authentication of user TEST AVATAR successful. Sending response to client.
region1	11:34:44	[CLIENT]: Told by user service to prepare for a connection from TEST AVATAR cf341ad0-f306-42e4-90a4-58a
region2	11:34:44	[CONNECTION BEGIN]: Region TEST_SIM2 told of incoming root agent TEST AVATAR cf341ad0-f306-42e4-90
region3	11:34:44	[OGS1 USER SERVICES]: Verifying user session for cf341ad0-f306-42e4-90a4-58a227608ecd
region4	11:34:44	[CONNECTION BEGIN]: User authentication returned True
region5	11:34:44	[CONNECTION BEGIN]: Region TEST_SIM2 authenticated and authorized incoming root agent TEST AVATAR
region6	11:34:44	[CAPS]: Registered seed capability /CAPS/7da79ce7-9b41-474c-8230-b01ce5b8cfce0000/ for cf341ad0-f306-
region7	11:34:44	[OBJECTADD]: /CAPS/OA/9e13e479-3591-4da8-bb3b-241a1a9aaafc/
region8	11:34:44	[EVENTQUEUE]: Adding new queue for agent cf341ad0-f306-42e4-90a4-58a227608ecd in region TEST_SIM2

図 5　ログイン処理中のサーバモジュールに出力されるログの一部

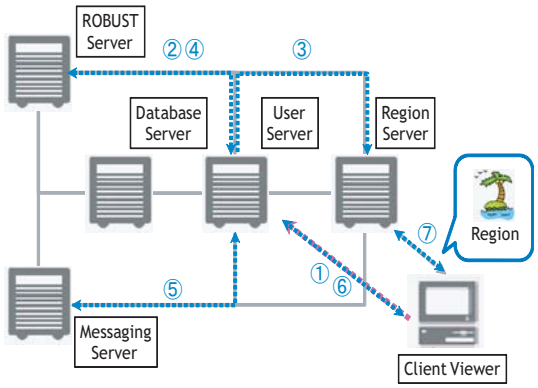


図 6　ログイン時のサーバにおける認証処理手順

（1）　クライアントから User サーバへ，ユーザ名，パスワード，アクセスする Region 情報を送信する．

（2）　User サーバから ROBUST サーバへ，アクセスする Region の位置情報を問い合わせ，ROBUST サーバは Region サーバの IP アドレス，座標などの情報を返す．

（3）　User サーバから IP アドレスを元に，アクセスする Region サーバへ，ランダムなキー値とユーザ ID を送信する．

（4）　User サーバから ROBUST サーバへ，ユーザ ID を用いて持ち物情報を問い合わせ，ROBUST サーバは持ち物情報のデータを返す．

（5）　User サーバから Messaging サーバへ，ユーザのログインを伝える．

（6）　User サーバからクライアントへ，Region サーバの情報や持ち物情報，手順 3 で Region サーバに送信したキー値を送信する．

（7）　クライアントから Region サーバへ，キー値を送信し，合致するとコネクションが確立する．

以上でクライアントが認証され，認証処理は終了となる．この間には 22 個の処理が行われ，その時間は，約 7 秒かかっている．

次に，アバタの初期化について，その手順を図 7 と共に示す．

（1）　Region サーバから User サーバへ，ユーザ ID を用いてアバタのデータ要求し，アバタの外形データと服装データの ID を受け取る．

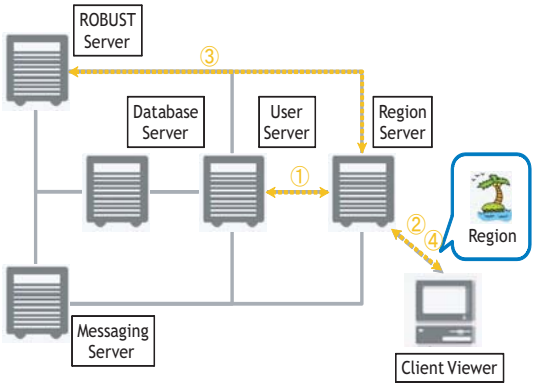


図 7　ログイン時のサーバにおけるアバタ初期化手順

（2）　Region サーバからクライアントへ，受け取ったデータを送信し，クライアントは服装データの ID を Region サーバに返す．

（3）　Region サーバから ROBUST サーバへ，服装データの ID で服装データを要求し，ROBUST サーバは服装データを送信する．

（4）　Region サーバからクライアントへ，服装データを送信し，クライアントはそのデータをアバタに貼り付ける．

以上でアバタの初期化終了となり，クライアントは OpenSim にログイン完了となる．この間には 22 個の処理が行われ，約 37 秒の時間がかかっている．

また，ログイン中に行われた 44 個の処理のうち，ほぼ全ての処理時間が 1 秒未満だったのに対し，Region サーバ上の 2 つの処理のみ，長い時間がかかっていることが分かった．それは，Region 上に新たなキューを加えること（約 6 秒），User サーバ上のユーザの容姿をアップデートすること（約 33 秒）である．つまり，これらの処理性能を上げることにより，ログイン時間の大幅な短縮が可能となる．

これらの処理が遅い理由として考え得る可能性は，ストレージアクセスが遅い，CPU 負荷が高い，メモリの利用が飽和，ネットワークの帯域幅が不足，ノード間のデータ通信とデータ処理の累積時間が長い等である．次章からその検証を行う．

7.　SSD を使用した高速データベースアクセス実験

ログインレスポンスが遅い理由として挙げられるうちの 1 つである，ストレージアクセスが遅い可能性について，以下に述べる実験で確認を行った．

7.1　SSD

SSD(Solid-State Drive) とは，記憶媒体としてフラッシュメモリを用いるドライブ装置であり，HDD と同じ接続インタフェースを備え，ハードディスクの代替として利用出来るものである．本研究では，インテル社の X25-M Mainstream SATA Solid-State Drive ，容量は 80GB タイプの SSD を使用した．まず，この SSD の性能評価を図 8 で示す．OpenSim システムのデータベースは MySQL を使用している．そのため，この SSD の性能測定にはデータベースベンチマーク TPC-C の

MySQL 向けの簡易実装である，tpcc-mysql というベンチマークを使用した [11].

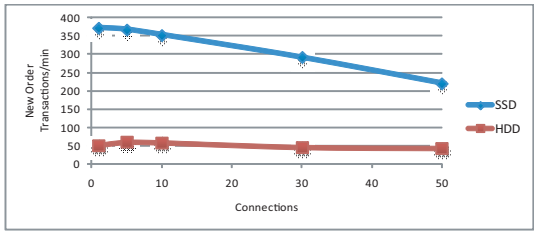


図 8　SSD の性能評価

横軸はデータベースに対する同時接続数，縦軸は 1 分間あたりに処理した new-order トランザクション数である．また，データベースのスケールファクターになる warehouse は 500 に設定した（I/O バウンドの負荷をかけたい場合，400〜1000 が推奨されている）．図 8 から，HDD と SSD において，4〜7 倍の性能差があることがわかる．つまり，この SSD を使用することは，ストレージアクセスの高速化を目的とする本実験において，有効である．

7.2　SSD を使用したログイン時間の測定実験

実験で使用している分散型メタバースサーバにおいて，Database サーバの HDD を SSD に替えた．つまり，SSD の性能による HDD との比較実験を行うため，実験環境を図 9 のように変更した．この環境の下，再度ログイン時間の測定を行った．

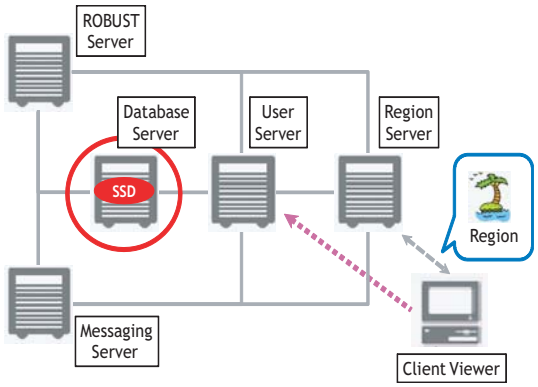


図 9　SSD を用いた実験環境

7.3　測定結果のまとめと考察

Database サーバの HDD を SSD に換えても，ログイン時間の短縮には繋がらなかった．また，同様の実験を，オブジェクトやユーザの持ち物情報を管理している ROBUST サーバにおいても行ったが，ログイン時間に全く変化はなかった．つまり，現在の OpenSim システムによるログイン時間は，サーバ側のストレージアクセス時間がボトルネックになっているものではないと考えられる．

8.　CPU，メモリ，ネットワーク使用率の測定

次に，CPU 負荷が高い，メモリの利用が飽和している，ネットワークの帯域幅が不足しているといった可能性に対し，以下に述べる実験で確認を行った．

8.1　Ganglia

Ganglia はクラスタ管理モニタリングツールである [8]．対象ノード（全メタバースサーバ）に Ganglia をインストールすることで，自動的にデータを収集し，視覚的に見やすいグラフを自動的に作成してくれる．Ganglia を使用し，全メタバースサーバのユーザログイン時における振舞を測定した．

8.2　測定結果

結果を図 10，図 11，図 12 に示す．ユーザは，23：35 に本実験で使用している OpenSim システムにログインしている．

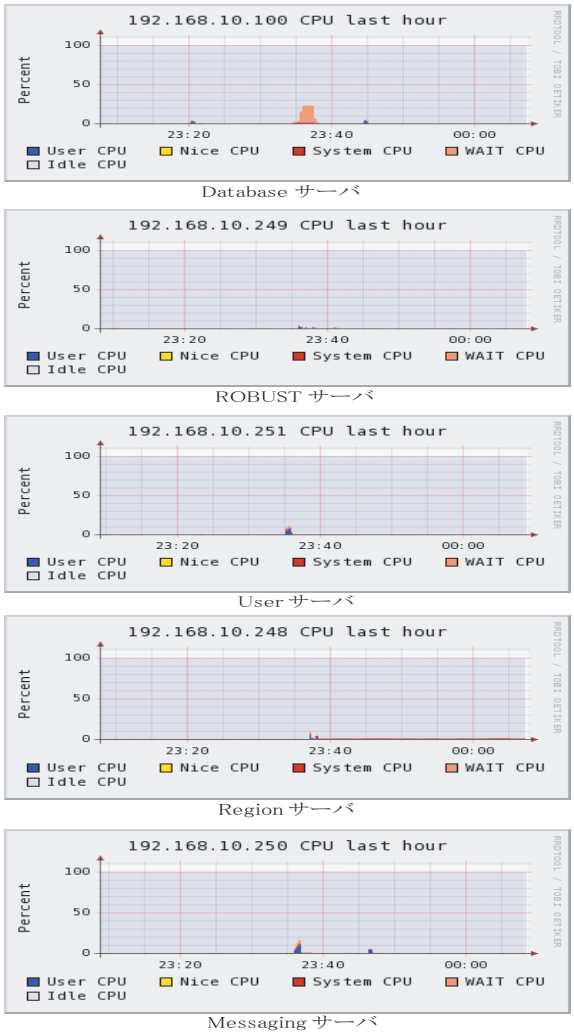
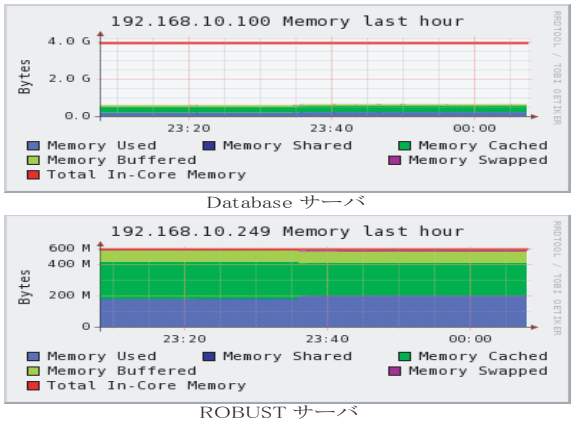


図 10　各サーバモジュールにおけるユーザログイン時の CPU 使用率



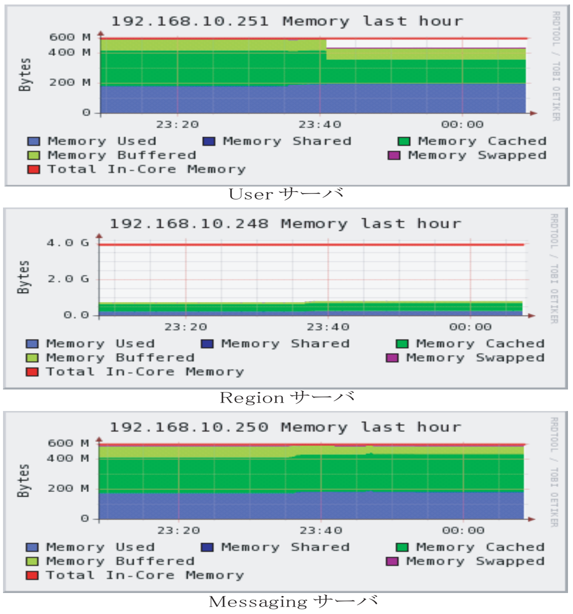


図 11　各サーバモジュールにおけるユーザログイン時のメモリ使用率

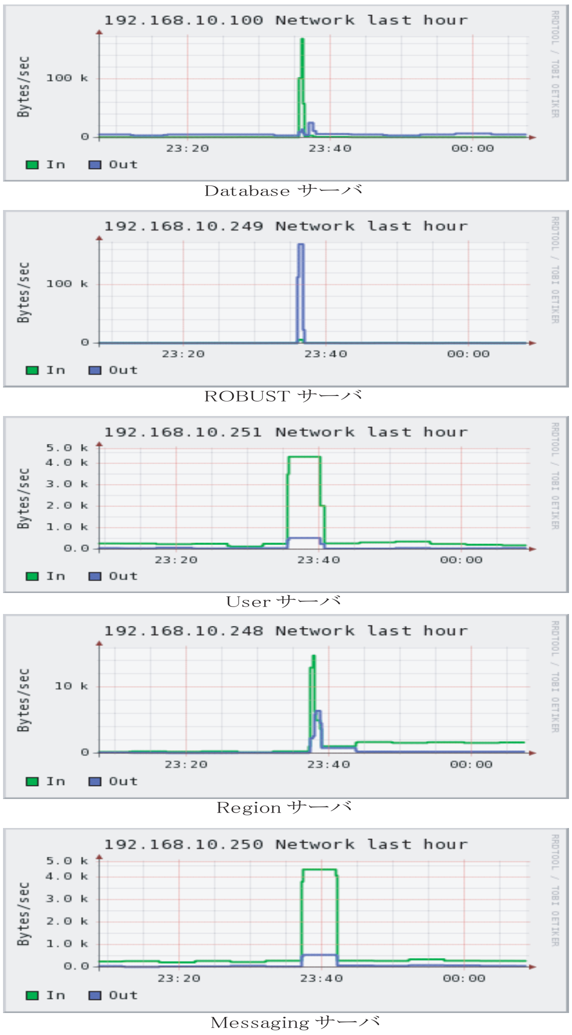


図 12　各サーバモジュールにおけるユーザログイン時のネットワーク使用率

CPU においては，どのサーバも使用率が低いが，メモリにおいては，ROBUST サーバ，Messaging サーバ，User サーバが飽和している状態であり，ネットワークにおいては，Database サーバと Region サーバの通信量が多かった．しかし，ネットワークの帯域幅は飽和している状態ではないため，可能性として，メモリ利用における飽和が，ボトルネックかもしれないと考えられる．

8.3　メモリ増設によるログイン時間の測定実験

メモリ利用における飽和が，ボトルネックである可能性について，メモリ量を増設し，再度ログイン時間の測定を行うことで確認した．つまり，メモリ使用率が高い，ROBUST サーバ，Messaging サーバ，そして User サーバにおいて，メモリ量を 512MB から 1GB に増やし，ログイン時間の測定を行った．

結果は，ROBUST サーバ，Messaging サーバ，そして User サーバにおいて，メモリ量を増加させても，ログイン時間の短縮には繋がらなかった．つまり，現在の OpenSim システムによるログイン時間は，サーバ側のメモリ量がボトルネックになっているわけではないと考えられる．

9.　パケットキャプチャによる通信解析

最後に，ノード間のデータ通信とデータ処理の累積時間が長いといった可能性に対し，実験により確認した．

6 章で述べたように，サーバ処理時間が長いものは，Region 上に新たなキューを加えること（約 6 秒），User サーバ上のユーザの容姿をアップデートすること（約 33 秒）である．これらの処理は，ログイン処理の後半部分に集中している．そのため，通信パケットにおいて，後半部分に重点を置き，解析した．

図 13 のように，tcpdump コマンドを用いて，全メタバースサーバのユーザログイン時における通信パケットをキャプチャした．その後，全サーバの後半のパケットを取り出し，時刻でソートした．更に，IP アドレスを基にどのサーバからどのサーバへパケットを飛ばしているかを確認した．

解析を進めると，処理時間が長いログインの後半のパケットは，全て Region サーバとクライアント間のパケットであることが判明した．つまり，両者のやりとりが，ネットワーク遅延等により長い時間がかかっている可能性がある．

21:29:04	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:04	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:04	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:04	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:04	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:04	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	c	re	IP	192.168.10.11.socks	192.168.10.248.cslistener	UDP, length 106
21:29:05	re	c	IP	192.168.10.248.cslistener	192.168.10.11.socks	UDP, length 51

図 13　tcpdump によるパケットキャプチャ

そこで次章では，以前の実験環境よりも高性能なサーバとクライアントマシンを使用し，OpenSim システムを構築した後，ユーザのログイン時間の測定を行った．

10.　高スペックな Region サーバとクライアント端末を用いたログイン時間測定実験

初めに表 2 のように，Region サーバのスペックのみを上げ，

ログイン時間を測定した．しかし，結果はログイン時間短縮には繋がらなかった．

そこで，次はクライアント端末の性能のみを上げ，ログイン時間を測定した．使用したクライアント端末は表 3 に示す．

表 2　Region サーバのスペック変更

	変更前	変更後
OS	Fedora Core12	Fedora Core12
CPU	Intel Xeon 3.6GHz	Intel Quad-Core Xeon 2.6GHz
memory	4GB	8GB

表 3　クライアントのスペック変更

	変更前	変更後
OS	Windows XP	Windows 7
CPU	Intel Pentium 4 CPU 3.00GHz	Intel Core2 Quad CPU 2.66GHz
memory	2.5GB	8GB
Graphic	Intel 82945G Express Chipset Family	NVIDIA Quadro NVS 295

その結果，図 14 のように，ログイン時間が平均約 31 秒に変化した．また，再現性も保たれていることがわかる．

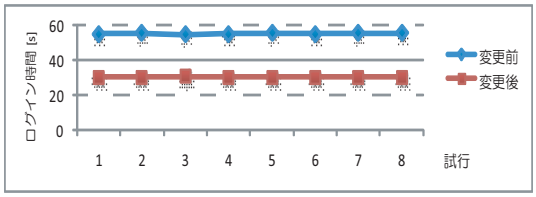


図 14　クライアント端末変更時のログイン時間

つまり本実験において，クライアント端末のスペックを上げた場合に，ログインレスポンスの短縮に繋がることが明らかになった．本実験で使用したクライアント端末は，64bitCPU と 64bit 対応 OS を搭載しているため，一度に処理出来るデータ量が多く，更にメタバースでは重要なグラフィックの処理が速くなったことが要因だと考えられる．この詳細を調べるため，次章の実験を行った．

11.　各サーバモジュールにおけるパケット解析

本章では，クライアント端末のスペックを変更した前後で，各サーバモジュールにおける振舞の変化を検証する．具体的には，tcpdump コマンドを用いてパケットをキャプチャし，各サーバモジュールのパケット数とパケット通信時間を比較する．

まず，クライアント端末を変更する前後における，パケット数とパケット通信時間の比較結果を図 15 と図 16 に示す．

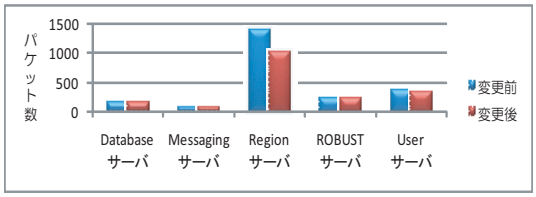


図 15　各サーバモジュールにおけるパケット数

Region サーバにおける処理パケット数とパケット通信時間が，他と比べて格段に大きいことがわかる．また，クライアント端末の変更後，Region サーバにおけるパケット数と通信時間が大きく減少していることがわかる．

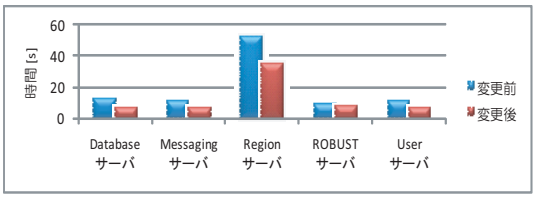


図 16　各サーバモジュールにおけるパケット通信時間

そこで，具体的にどのようなパケットが減少しているのか，解析を進めた．

結果を図 17 に示す．この結果により，UDP 通信におけるパケットの数が極端に減少していることがわかる．OpenSim システムにおいて，サーバモジュール間の通信には TPC(HTTP) が用いられており，UDP 通信は，クライアントと Region サーバの通信のみで使われている．これはメタバースの土地やアバタに関する精細なグラフィクス等の膨大なデータを両者で細かく処理しながらやりとりしている部分であり，クライアントの性能向上が，OpenSim システムにこのような形で確かに影響を与えたことが確認出来る．

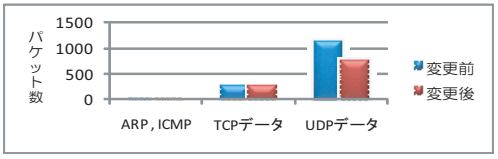


図 17　Region サーバにおけるパケット数

12.　まとめと今後の課題

本稿では，現在の典型的なスペックを有するサーバ機とクライアント端末を用いて OpenSim によるメタバース環境を構築した場合，ログインレスポンスが極めて悪くなる事を示し，この原因の解析を行った．メタバースシステムは，様々なサーバモジュールが役割を分担し，サーバモジュール間やクライアントとの間で複雑な処理が行われているため，性能を決める要因も様々なものが考えられる．本研究ではまず，ログ解析結果などを基にして，これらの処理の振舞を明らかにした．

次に各サーバモジュールが動作するサーバ機において，性能のボトルネックとなり得る要因を整理し，これらについて 1 つずつ調べて行った．その結果，サーバ機単独のストレージアクセス性能や CPU 負荷、メモリ量などによる性能低下ではない事が明らかになった．

そこでさらに，サーバモジュール間やサーバモジュールとクライアントとの間でデータ処理を伴うやり取りの部分に焦点を当てて，詳しく調べて行った．その結果，サーバモジュールの 1 つである Region サーバとクライアントとのやり取りの部分がログインレスポンス全体に極めて大きな影響を与えていることが明らかになり，このやり取りの際にデータの処理と送受信を繰り返すクライアント側における処理が重いため，クライアントのマシンスペックを向上させるとログインレスポンスが劇的に改善する事がわかった．

PC のマシン性能は時間と共に向上していくものであるが，

ユーザは常に最新のマシンを使用している訳ではない．さらに近年の傾向として，クライアント端末はスペックや重量を軽量化し，処理をサーバ側で行わせるいわゆるシンクライアントの利用が広がっており，高性能なクライアントを前提としたシステムは受け入れられにくいことが考えられる．従って，将来的にメタバース環境を広く普及させるためには，本稿で示したサーバモジュールとクライアントとのやり取りの部分において，クライアント側で行われている処理の一部をサーバ側で処理するように実装を変更して行くことが望ましいと言える．

今後の課題としては，まずクライアントの CPU や通信の負荷などを測定し，クライアントのスペックを変えた時にこれらにどの程度影響するかを定量的に測定する．そしてクライアントにおける処理の一部をサーバ側に移して性能評価を行って行く事により，クライアント側を軽量化した場合にどの程度現実的なシステム構築が可能であるか明らかになる．それらのデータを基に，広く受け入れられるメタバースのクライアントとサーバの構築法を明らかにする事が本研究の最終目標である．

文　　献

[1] Second Life: http://jp.secondlife.com/
[2] アメーバピグ: http://pigg.ameba.jp/
[3] IMVU: http://www.imvu.com/
[4] Moshi Monsters: http://www.moshimonsters.com/
[5] Meet-Me: http://www.meet-me.jp/
[6] OpenSimulator: http://opensimulator.org/
[7] Sanjeer Kumar et.al.: ”Second Life and the New Generation of Virtual Worlds”,IEEE Computer,vol.41,No.9, pp.46-53 Sep. 2008.
[8] Ganglia Monitoring System: http://www.ganglia.info/
[9] Second Life wiki: http://wiki.secondlife.com/wiki /Current_login_protocols
[10] Think IT: http://thinkit.co.jp/article/129/4?page=0,0
[11] tpcc-mysql: http://d.hatena.ne.jp/sh2/20090212