Vol.2010-DBS-151 No.11
2010/11/12

分散型メタバースサーバの ログインタイム短縮に関する一検討

松 原 麻 佑[†1]　小 口 正 人[†1]

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，サーバの負荷とレスポンスという観点から評価を行う．本研究では，オープンソースソフトウェアとして提供されており，ユーザが所持するマシンにインストールしてサービスを提供可能な分散型メタバースサーバである OpenSim システムを用いて評価を行う．すなわち，メタバース実行時のサーバ側の実測評価に焦点を当てる．

A Examination about Shorter Login Time in Distributed Metaverse Server

Mayu Matsubara[†1] and Masato Oguchi[†1]

”Metaverse”, which is expected to be spread in the future, is a new service connecting to three dimensional virtual space constructed as electronic data through the Internet. We have evaluated it from the viewpoint of load and response of the server for making it to be high performance. In this paper,OpenSim server that provides the metaverse service is constructed and evaluated. That is to say, our research work focuses on server side of Metaverse system.

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している．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から[⋆1]，期待されている程は普及していないという現実も挙げられるが，それでもなお，このメタバース業界に大手企業が続々と参入してきており，少しずつ広がりをみせている．2010 年 3 月のデータによると，Second Life は過去 30 日間で 100 万人のユーザがログインしており，Second Life 内の土地の広さや Second Life 内で使用出来る通貨の流通量は，増加傾向にある．また PC の性能向上は日進月歩の状態が続いていることから，端末のスペック不足の問題は今後解消の方向に進むと考えられる．

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

†1 お茶の水女子大学
Ochanomizu University

⋆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 6700 以上，ATI は X800 以上などとなっている

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．

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側の実測結果を基に，サーバの動作解析を行う．

3. 集中型と分散型メタバースサーバの評価環境

メタバースはクライアント・サーバ型のシステムである．

3.1 集中型メタバースサーバの実験環境

集中型サーバは，サービスを提供する側と受ける側に役割分担されており，多くのクライアントから様々なリクエストを受けて処理するサーバが特定箇所に集中しているシステムである．データは一箇所に集まり，利用者のアクセスも一箇所に集中する．

OpenSim における集中型サーバは，スタンドアロンモードとグリッドモード両方で構築出来る．本研究ではまず，ある特定の１台のマシンのみに OpenSim をインストールし，集中型メタバースサーバの構築を行った．クライアントは，この１台にアクセスする．

3.2 分散型メタバースサーバの実験環境

分散型サーバは，データ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，ユーザサーバ，ロバストサーバ，メッセージングサーバ，リージョンサーバの４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．ユーザサーバは，ユーザ（ア

バタ）の管理サーバ，ロバストサーバは，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，メッセージングサーバはメッセージ処理を行うサーバ，リージョンサーバは，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 と図 2 に示す．図 1 は，2 台のサーバマシンを使用し，最もサーバ数が少ない場合を想定している．1 台にはユーザサーバ，ロバストサーバ，メッセージングサーバ，リージョンサーバを構築し（データベース管理も含む），他方のサーバマシンでは，リージョンサーバのみを構築し，1 台目のサーバへリンクを張っている．サーバは，CPU が Intel Xeon 3.6GHz と 2.4GHz，Main Memory は 4GB と 512MB，OS は Fedora Core12（Linux2.6.31.5）で OpenSim0.6.8 を使用している．これに対し図 2 は，5 台のサーバマシンに各サーバモジュールを分散させて配置した．これは，最もサーバ数が多い場合を想定している．

一方クライアントには，CPU が Intel Pentium 4 CPU 3.00GHz，Main Memory は 2.5GB，OS は Windows XP，グラフィックは Intel 82945G Express Chipset Family の PC を使用した．

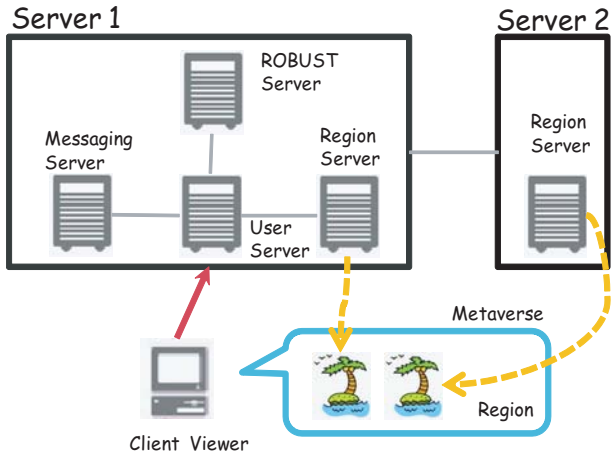

図 1 2 台分散型メタバースサーバ

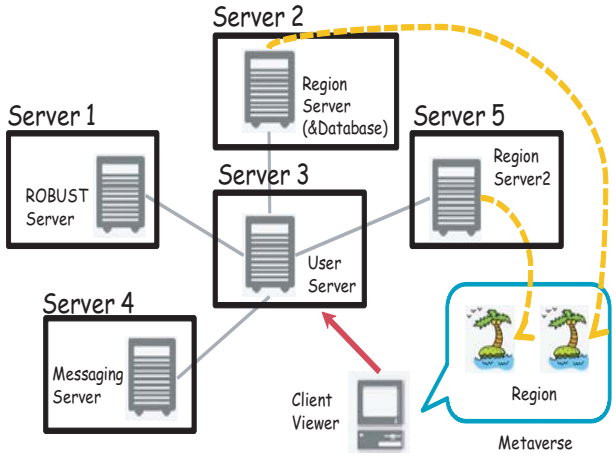

図 2 5 台分散型メタバースサーバ

4. 集中型と分散型サーバのログイン時間

4.1 実 験 概 要

本研究では OpenSim サーバを集中型と分散型で構築し，ユーザのログインにおける時間を測定する．ここでのログイン時間とは，ユーザがログインボタンを押してから，OpenSim が完全に立ち上がるまでの時間を指す．

4.2 測定結果と考察

ログイン時間の測定結果を図 3 に示す．

各サーバ構築法におけるログイン時間は，集中型サーバで構築した場合，スタンドアロンモードにおいては約 16 秒，グリッドモードにおいては約 18 秒，分散型サーバで構築した場合，2 台の分散型サーバでは約 32 秒，5 台の分散型サーバでは約 31 秒かかった．

OpenSim システムを集中型サーバで構築すると，スタンドアロンモードで動作させた場合と，グリッドモードで動作させた場合において，全体的にグリッドモードの方が時間はかかるものの，それほど大差は見られなかった．また，OpenSim システムを分散型サーバで

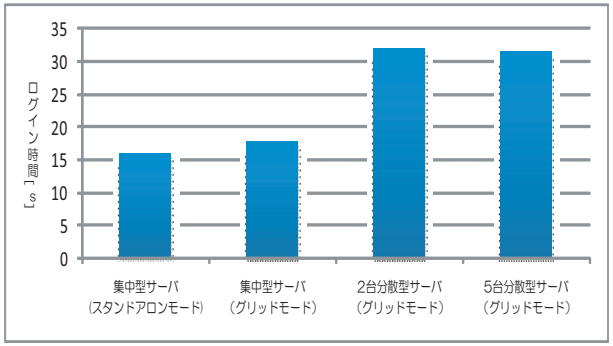

図 3 サーバ構築法ごとのログイン時間

構築すると，2 台で構築した場合と，5 台で構築した場合においても，差は見られなかった．しかし，前半 2 つ（集中型サーバ）と後半 2 つ（分散型サーバ）を比較すると，約 2 倍ほどのログイン時間の差が出るという特徴があげられる．

以上の実験結果により，集中型サーバと分散型サーバで差はあるが，いずれも場合も，トータルの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である．メタバースが今後より広く用いられていくためには，レスポンスの向上は必須であると考えられる．そのため我々は，以降においてメタバースサーバのレスポンスに関する解析を行った．

5. 分散型サーバにおけるユーザログイン時のログ解析

5.1 実験概要

本稿では，メタバースが将来的に Web のように普及すると予測し，分散型メタバースサーバにおけるユーザログイン時のログ解析を行う．つまり，グリッドモードで構築した OpenSim サーバを用いて，分散型サーバを構築し，ユーザのログインにおけるサーバのプロファイリングを行う．具体的な実験内容は，ユーザが OpenSim へログインした際，各サーバからログイン処理中に出てきたログを収集し，各サーバごとの処理時間を計算する．その後，全サーバの処理を時間でソートし，サーバ間の通信処理を色分けすることで，どの処理にどのくらいの時間がかかっているのかを解析する．

5.2 各サーバモジュールの処理手順

5 台の分散型サーバ（グリッドモード）における，サーバアクセス時の処理手順を図 4 で示す．この解析は，図 5 のように，各サーバに表示されたログを集め，それらを時間でソートした後，各ログに対し処理内容で色分けをすることで，導き出した．その際には，[8] を参考にしている．

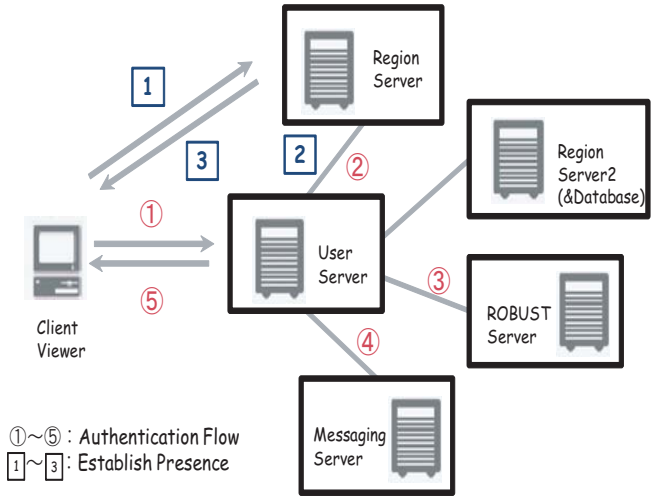

図 4 分散型サーバにおけるログイン処理手順

まず，アバタの認証処理を行う．

(1) クライアントからユーザサーバへアクセス（ユーザ名とパスワードを送信）．
(2) ユーザサーバからリージョンサーバへアクセス（ユーザ ID を渡す，コネクション要求）．
(3) ユーザサーバからロバストサーバへアクセス（容姿や持ち物情報を要求）．
(4) ユーザサーバからメッセージサーバへアクセス（ユーザのログインを伝える）．
(5) ユーザサーバからクライアントへアクセス（再度セッションの確認）．

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

server	time	process
User1	16:55:35	16:55:35 - [LOGIN BEGIN]: XMLRPC Received login request message from user 'TEST' 'AVATAR'
User2	16:55:35	16:55:35 - [LOGIN]: XMLRPC Client is Second Life Developer 2.0.0.203055, start location is last
User3	16:55:35	16:55:35 - [LOGIN]: Telling TEST_SIM2 @ 1000,1010 (http://192.168.10.248:9000/) to prepare for client connection
ROBUST1	16:55:35	16:55:35 - [INVENTORY SERVICE]: Getting inventory skeleton for 38f6d23a-a8fe-4c9a-8532-1b5331943b40
messaging1	16:55:36	16:55:36 - [LOGIN]: User TEST AVATAR logged into region 1099511628034560 as root agent, building indexes for user
messaging2	16:55:36	16:55:36 - [MESSAGE SERVICE]: Requesting friends list for 38f6d23a-a8fe-4c9a-8532-1b5331943b40 from http://192.16
User4	16:55:36	16:55:36 - [LOGIN]: Found appearance for TEST AVATAR
User5	16:55:36	16:55:36 - [USER AUTH]: Verifying session a0eaf9a0-7834-b4d5-b067-44e58472d89b for 38f6d23a-a8fe-4c9a-8532-1b5
User6	16:55:36	16:55:36 - [UserManager]: CheckAuthSession TRUE for user 38f6d23a-a8fe-4c9a-8532-1b5331943b40
User7	16:55:36	16:55:36 - [MSGCONNECTOR]: Sending login notice to registered message servers
User8	16:55:36	16:55:36 - [USER SERVER FRIENDS MODULE]: BEGIN XmlRpcResponseXmlRPCGetUserFriendList from 192.168.10.250:3
User9	16:55:36	16:55:36 - [USER SERVER FRIENDS MODULE]: END XmlRpcResponseXmlRPCGetUserFriendList from 192.168.10.250:38
User10	16:55:36	16:55:36 - [LOGIN]: Notified : http://192.168.10.250:8006 about user login
User11	16:55:36	16:55:36 - [LOGIN END]: XMLRPC Authentication of user TEST AVATAR successful. Sending response to client.
region1	16:55:36	16:55:36 - [CLIENT]: Told by user service to prepare for a connection from TEST AVATAR 38f6d23a-a8fe-4c9a-8532-1b5
region2	16:55:36	16:55:36 - [CONNECTION BEGIN]: Region TEST_SIM2 told of incoming root agent TEST AVATAR 38f6d23a-a8fe-4c9a-85
region3	16:55:36	16:55:36 - [OGS1 USER SERVICES]: Verifying user session for 38f6d23a-a8fe-4c9a-8532-1b5331943b40
region4	16:55:36	16:55:36 - [CONNECTION BEGIN]: User authentication returned True

図 5 グリッドモードにおける全サーバのログ

次に，アバタの初期化を行う．

(1) クライアントからリージョンサーバへアクセス（アバタをリージョンサーバに追加）．
(2) リージョンサーバからユーザサーバへアクセス（フレンドリストを得る）．
(3) リージョンサーバからクライアントへアクセス（アバタの容姿のアップデート）．

この間には 28 種の処理が行われ，約 23 秒の時間がかかっている．これらの測定をサーバごとに分けると，各々のサーバモジュール実行時間は図 6, 図 7 のようになる．

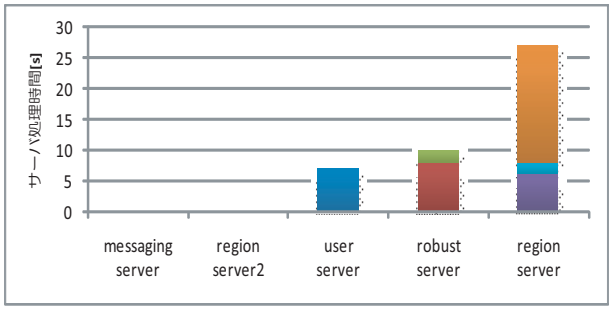

図 6 2 台の分散型メタバースサーバにおけるログイン時の各サーバモジュール処理時間

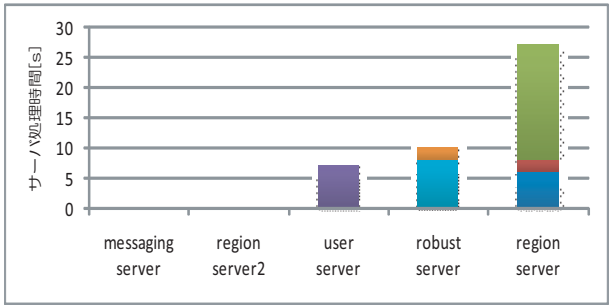

図 7 5 台の分散型メタバースサーバにおけるログイン時の各サーバモジュール処理時間

5.3 測定結果のまとめと考察

図 6，図 7 から，分散型メタバースサーバを 2 台で構築した時と 5 台で構築した時において，ログイン時間も，ログイン時のサーバ処理内容も，ほぼ同様の結果が得られた．また，分散型メタバースサーバを 2 台で構築した時と 5 台で構築した時，ユーザのログイン時に，サーバ側に出力されるログ（例：図 5）を比較したところ，ほとんど変わらなかった．

更に，図 6，図 7 にも表れている，時間がかかる具体的な処理を明らかにするため，ログの解析を進めた．ほとんどの処理が 1 秒未満で行われるのに対し，数秒かかった処理は，以下の 4 つである．リージョンサーバにおいて，リージョンにユーザの新しいキューを加えることに約 6 秒，ユーザの容姿をアップデートすることに約 19 秒かかっている．更にロバストサーバにおいては，ユーザのインベントリ情報の取得に約 8 秒かかり，ユーザサーバにおいては，認証処理をクライアントに送信することに約 7 秒かかっている．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．

6. SSD を使用した高速データベースアクセス実験

特に処理時間の長いものとして，ユーザ容姿のアップデートやインベントリ情報の取得が挙げられる．そこで，本研究では，OpenSim システムは，データベースアクセスが遅いという仮説を立て，検証していった．

6.1 SSD

SSD 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 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[9]．

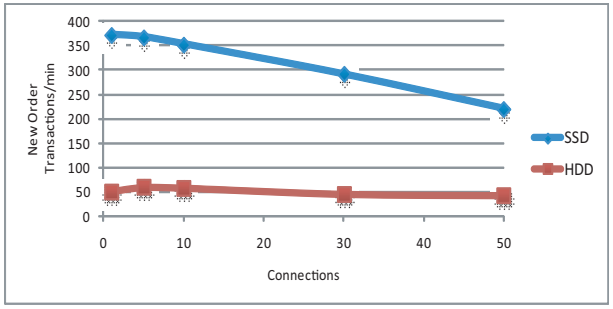

図 8 SSD の性能評価

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，データベース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．

6.2 SSD を使用したログイン時間の測定実験

本研究では，データベースアクセスが遅いと仮定し，実験で使用している分散型メタバースサーバにおいて，データベースサーバの HDD を SSD に替え，再度ログイン時間の測定を行った．

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．つまり，データベースを扱うサーバを 1 つ増やした．

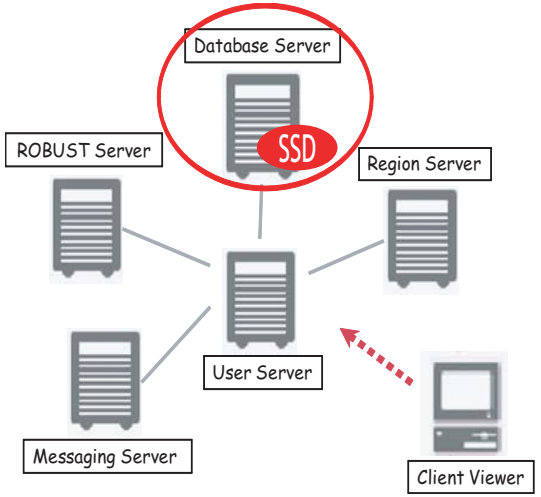

図 9 SSD を用いた実験環境

6.3 測定結果のまとめと考察

データベースサーバの 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ロバスト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，サーバ側のデータベースアクセス時間そのものが原因ではないと考えられる．

7. まとめと今後の課題

本稿では，OpenSim を使用し，集中型サーバと分散型サーバを構築し，ユーザがメタバースサービスへのログイン時間の測定とメタバースサーバのプロファイリング，SSD をサーバに用いた場合の，ログイン時間の測定を行った．

その結果から，本稿で注目している分散型サーバは，集中型サーバよりも約 2 倍のログイン時間がかかることがわかった．そのため，分散型サーバの構築法を変化させ，サーバのプロファイリングを行い，サーバ処理の改善点を検証した．そこで，サーバ側のデータベースアクセスが遅いと仮定し，SSD を使用することで，ログイン時間の変化を見た．しかし，

Vol.2010-DBS-151 No.11
2010/11/12

ログイン時間の短縮には繋がらなかったため，長いログイン時間は，データベースアクセス時間そのものが原因ではないと考えられる．

今後は，サーバ間のネットワークや CPU 使用率を見ていくことで，更にログイン処理時の OpenSim サーバの解析を進めていく．そして，ログイン時間のボトルネックとなっている箇所を解明し，改善していく．

参 考 文 献

1) SecondLife, 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) http://wiki.secondlife.com/wiki/Currentloginprotocols
9) http://d.hatena.ne.jp/sh2/20090212