# NEUTRANS BIZ - OpenRelay
# ネットワークパフォーマンスレポート

FurtherSystem Co.,Ltd.
Ryuichi Saeki

| date | version | |
|---|---|---|
| 2019-07-31 | v1.0 | created |

## 1. 目的

1. Synamon社VR ソリューション製品 NEUTRANS BIZにおいて現段階で同一入室数をどこまで増やせるかを確認する。
2. FurtherSystem社にて開発中のリアルタイムサーバ、OpenRelayが現段階で実運用においてどの程度のパフォーマンスを発揮できるかを確認する。
3. どこを改善すればさらに増やすことが可能かを確認する。

以降、本ネットワークパフォーマンス検証のことを"本検証"と記載する。

## 2. 前提

| | |
|---|---|
| **OpenRelay未実装分・・・** | OpenRelayは現在ルーム一覧機能とVoice機能が未実装、Voiceの負荷を考慮する必要があるが、<br>多人数接続時全てのログインユーザが完全同時に会話することは想定しにくい。 |
| **組み込み不具合・・・** | NEUTRANS BIZへOpenRelay組み込みの際の２件の不具合がある。共有オブジェクト同期が行えない、<br>スケールが崩れるが負荷を与えることに関しては問題ないため無視する。 |
| **同一ルームに対して実施・・・** | 同一サーバ・同一ルーム内に対してのクライアントの最大ログイン数を確認する。 |
| **リプレイクライアントの使用・・・** | クライアントの接続負荷を与える上で、全てを実クライアントで行うと莫大なリソースを要する。<br>以下のような疑似クライアントを用意することでリソース費用・開発費用を抑える。*1<br><br>・実クライアントの通信内容を記録し、ユーザIDのみを書き換えた通信内容を再生するだけのクライアント、リプレイクライアントを用意する。<br>・ABループ機能を用意する。通信負荷を与え続けたい場合、ABループ機能を有効にすると指定した範囲内の通信を繰り返し送信し続ける。<br>・ABループを機能を無効にするとログイン処理のみを行い以降は受信のみを行う受信待機モードで稼働する。 |
| **実クライアントでの接続・・・** | 実クライアントでの接続は必ず最低1クライアント分は行う。FPSの劣化状況や、描画負荷の状況の確認。<br>また、リプレイクライアントでの擬似負荷が意図しない負荷になり、現実に起こりうる負荷と乖離してしまわないようにするため、必ず実クライアントでの目視確認も行う。 |
| **Legacy Shaderの使用・・・** | 事前の同時接続数確認にてStandardShaderの負荷が高く31クライアント以上が接続できないと言うことを把握していた。適切な負荷を与えるため、本検証ではStandardShaderをLegacyShaderに切り替える。 |
| **ログイン間隔の制約・・・** | 事前の同時接続確認にて、ログイン時の負荷が高くあまり間隔を短くするとログイン数が著しく低くなってしまうことがわかっている。適切な負荷を与えるため、本検証ではログインの際に8秒のクールタイムを設け、ログイン負荷を回避する。 |

*1 リプレイクライアントは一般的に攻撃用途にも利用できてしまうため本試験のためのみの例外実装とする。
NEUTRANS BIZ側にも例外実装を行うことで稼働させるように調整済み。
OpenRelay側にも例外実装を行うことで稼働させるように調整済み。
実サービスを想定した場合にリプレイクライアントを使用した通信は利用できないようにしてある。

## 3. 各テストケースと結果一覧

| # | テストケース | 実施概要 | 確認内容 | 結果 |
|---|---|---|---|---|
| 1 | **N:N 通信負荷** | 接続するクライアントが全て送受信を行い負荷を与える。*1 | 最大ログイン数確認<br>CPU,NW負荷状況確認 | **MAX 504**<br>結果#1参照 |
| 2 | **1:N 通信負荷** | 接続するクライアントのうちごく一部が送受信を行い、その他多数は受信のみを行う。*2 | 最大ログイン数確認<br>CPU,NW負荷状況確認 | **MAX 504**<br>結果#2参照 |
| 3 | **参考 描画負荷** | Standard Shaderを使用した際の最大ログイン数の記録。 | 参考資料 | **MAX 31**<br>参考#3参照 |

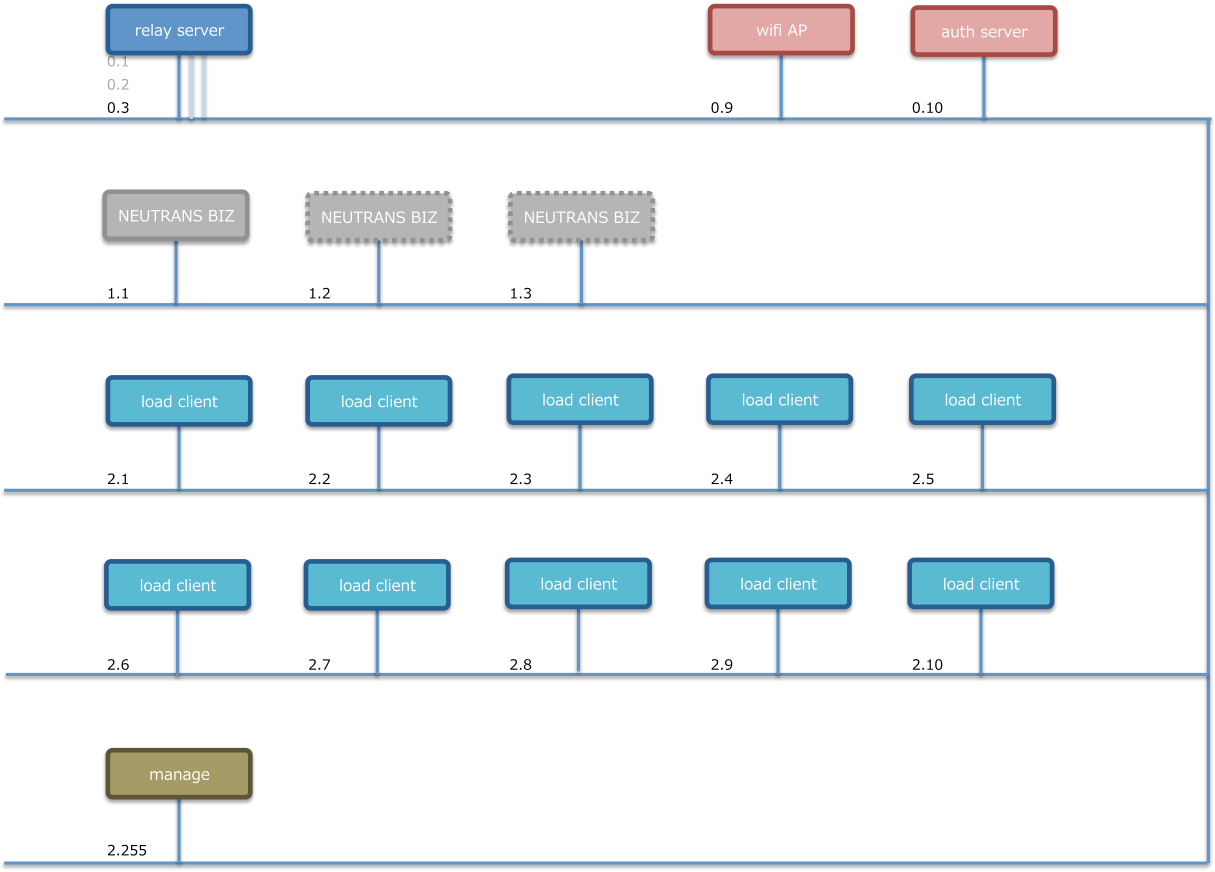
*1 1. NEUTRANS BIZにて1クライアントほどログインする。挙動確認に使用する。
2. 以降のログインはリプレイクライアントを使用し、N:N通信を再現する。
3. 八秒間隔でリプレイクライアントをリアルタイム通信ネットワークへ参加させる。
4. 各リプレイクライアントはそれぞれあらかじめ用意されている送信データを
ループで繰り返し送信し続け、N:N通信相当の擬似負荷を与える。
5.負荷情報を記録するため、200ログイン区切りで100秒程度ログインを停止するのインターバル時間を用意する。

*2 1. NEUTRANS BIZにて1ユーザほどログインする。挙動確認に使用する。
2. 以降のログインで配信リプレイクライアントと、待機リプレイクライアントを使用することでN:1通信を再現する。
3. 配信リプレイクライアントを20クライアントほど八秒間隔でログインさせる。配信負荷として利用する。
5. 各配信リプレイクライアントははそれぞれあらかじめ用意されている送信データを
ループで繰り返し送信し続け、N:1通信の1側相当の擬似負荷を与える。
4. 待機クライアントを八秒間隔でリアルタイム通信ネットワークへ参加させる。
5. 各待機リプレイクライアントはそれぞれあらかじめ用意されている送信データを
一度だけ送信し、以降はデータを送信せず、コネクション接続維持用のハートビートパケットのみを
60秒間隔で送信し、受信のみを行うリプレイクライアントとして稼働させる。N:1通信のN側相当の擬似負荷を与える。
6. 負荷情報を記録するため、200ログイン区切りで100秒程度ログインを停止するのインターバル時間を用意する。

## 4. 検証構成 - サーバ・ノード

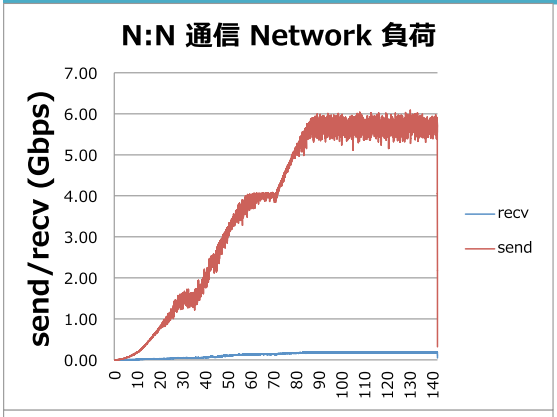
| | |
|---|---|
| **relay server**・・・ | OpenRelayを稼働させるサーバ、今回の負荷検証のパフォーマンス情報の計測対象。 |
| **wifi AP**・・・ | auth serverがUTPケーブルでのネットワーク接続ができないため、wifi経由でのネットワーク参加をできるようにするため用意。 |
| **auth server**・・・ | NEUTRANS BIZの認証処理を行う、負荷検証の初期の認証処理でのみ利用。 |
| **NEUTRANS BIZ**・・・ | NEUTRANS BIZ クライアントアプリケーション、VRHMDでの動作状況などの確認に使用する。 |
| **load client**・・・ | リプレイクライアントを大量に起動してrelay serverに負荷をかける。 |
| **manage**・・・ | 負荷検証の管理ノード、試験の開始やデータの回収、インターネット接続の例外ルーティングなどを管理する。 |

## 4. 検証構成 - ネットワーク



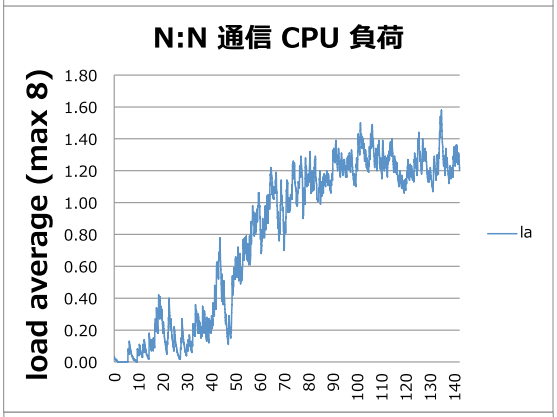

catalyst2960 network 10.0.0.0/16

## 5. 結果 - #1 N:N 通信負荷



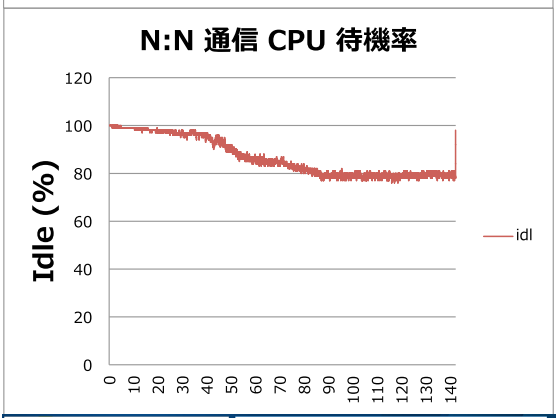

* リプレイクライアントでN:N通信の負荷をかけつつログイン数を増やし、負荷傾向を確認
* x軸は時間(分)
* 90分でログインが頭打ちとなっている
* 段差のある部分は200ログインごとの負荷観測作業時間用の100秒インターバル

**>> ログイン数が多くなるほど送信負荷が指数関数的に増大している**
**>> 送信負荷は6Gbpsまで到達している**
**>> 性能に余裕がない場合、通常100秒インターバル部分で減少が起きるがその傾向はない**



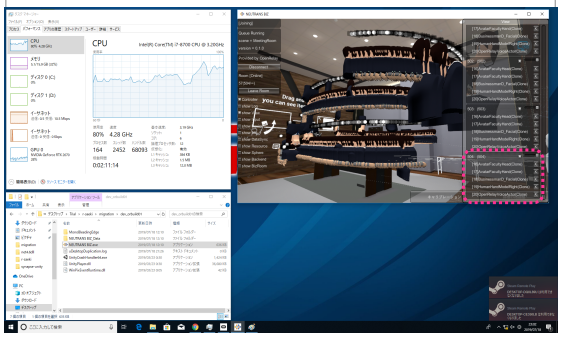

* x軸は時間(分)
* 90分でログインが頭打ちとなっている

**>> max8に対して1.2~1.6CPU使用率は15%程度で低い**
**>> ログイン頭打ち後もCPUを使用し続けていることから通信負荷は継続している**



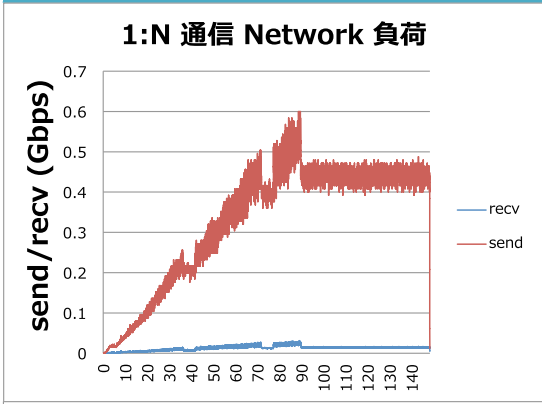

* x軸は時間(分)
* 90分でログインが頭打ちとなっている

**>> ログインが頭打ちになるまで緩やかに待機率が減少**
**>> ログインが頭打ちになると待機率80%を推移**
**>> ログイン頭打ち後もCPUを使用し続けていることから通信負荷は継続している**



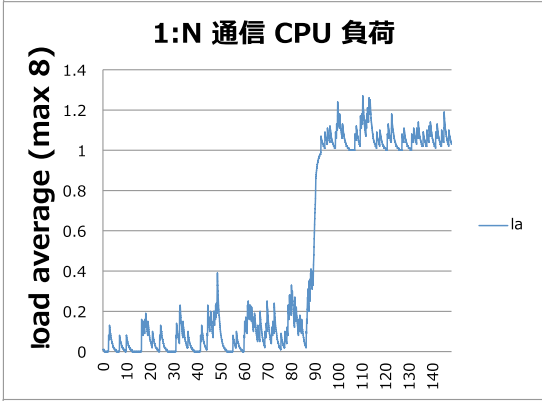
**>> ログインが504で頭打ちになっていることを確認**

## 5. 結果 - #2 1:N 通信負荷



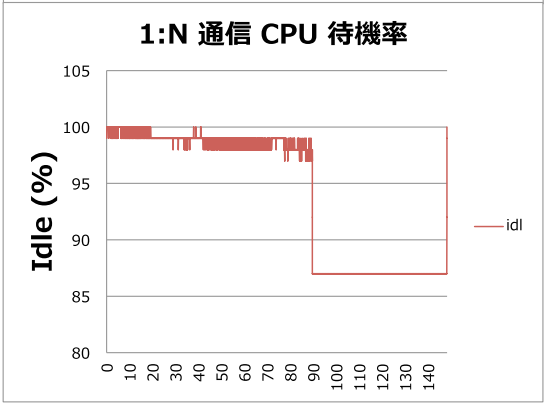

* リプレイクライアントでN:1通信の負荷をかけつつログイン数を増やし、負荷傾向を確認
* x軸は時間(分)
* 90分でログインが頭打ちとなっている
* 段差のある部分は200ログインごとの100秒のインターバル

**>> ログイン数が多くなるほど送信負荷が１次関数的に増大している**
**>> 送信負荷は0.6Gbpsまで到達してその後0.45Gbpsを推移**
**>> N:N通信のグラフと比較するとログイン負荷の起伏が大きく出ているが**
**N:N通信と比較すると全体のトラフィック量が少ない為、大きく見える**
**>> N:N通信のグラフと比較すると波形は全体的に安定しており負荷も小さい**
**>> 性能に余裕がない場合、通常100秒インターバル部分で減少が起きるがその傾向はない**



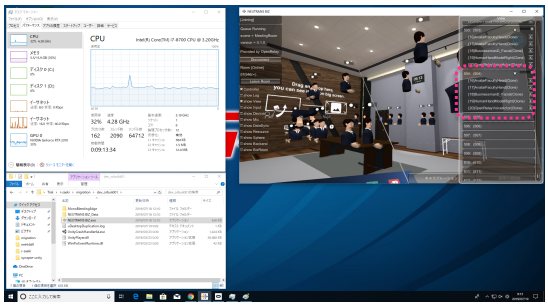

* x軸は時間(分)
* 90分でログインが頭打ちとなっている

**>> max8に対して1~1.2CPU使用率は13%程度で低い**
**>> ログイン頭打ち後もCPUを使用し続けていることから通信負荷は継続している**
**>> N:N通信のグラフと比較すると波形は安定しており負荷もやや小さい**

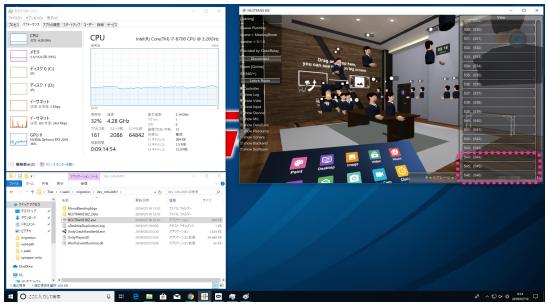



* x軸は時間(分)
* 90分でログインが頭打ちとなっている

**>> ログインが頭打ちになるまで待機率はほぼ100%**
**>> ログインが頭打ちになると待機率87%を推移**
**>> ログイン頭打ち後もCPUを使用し続けていることから通信負荷は継続している**
**>> N:N通信のグラフと比較すると波形は安定しており負荷もやや小さい**

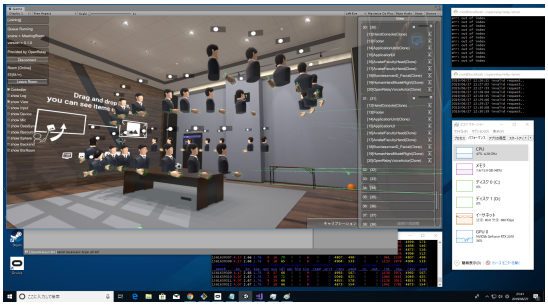


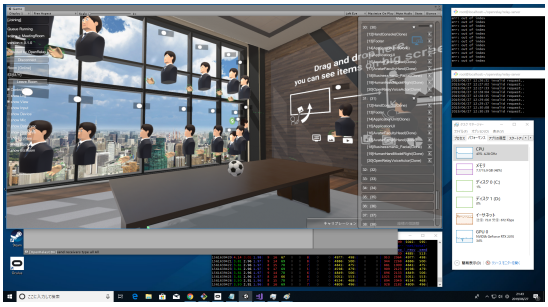
**>> 最後にログイン成功しているのは504**

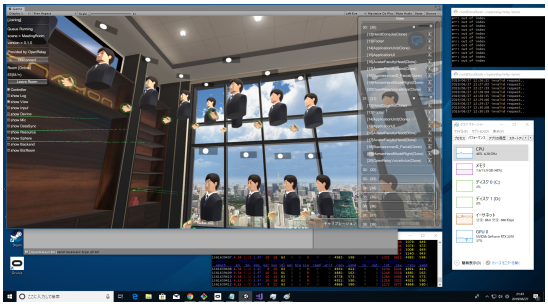


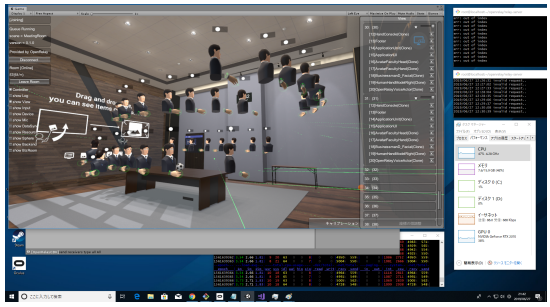
**>> 504で頭打ち後も546までログイン試行を行った形跡がある**

# 5. 参考 - #3 描画負荷









＊ 左記は負荷検証を行う前に31までしかログインできないことを事前準備で確認した際のもの
＊ Standard ShaderをLegacy Shaderに切り替えると31以上ログインできるようになった

## 6. 考察

1. N:N通信は２次関数の傾向、1:N通信は1次関数の傾向の負荷がかかる。

2. リソースにはかなり余裕がある為、504はN:Nの性能限界ではない可能性あり。また1:Nの性能限界関してはそれをさらにを大きく超える見込み。

## 7. 結論

1. アプリケーションの改修でログイン数504をより実運用可能な状態に近づけることができる。

2. 504よりさらに大きいログイン数実現にはアプリケーションレイヤ外に及ぶ広範囲での改善が必要。

*1 サーバは専用に用意し、ログイン間隔は8秒以上空けること。

* ログイン数504は性能限界ではない可能性が高いが、便宜上ここではアプリケーション実装での性能限界を504と表現する。