日本ソフトウェア科学会第 28 回大会 (2011 年度) 講演論文集

VNC を用いた授業用画面共有システムの設計と実装

谷成 雄　大城 信康　河野 真治

各クライアントを Tree 型に接続し、親が配信したデータをリレーさせることで分散 VNC アプリケーションを実装した。通常の VNC では配信者へ負荷が集中する設計となっている.。例えば、大学の講義等で VNC を用いて画面共有を行った時、クライアントの増加に比例して配信者への負荷が増えてしまう。この問題を解決する為に、Tree 構造にクライアントを接続させ、Top のクライアントから子供へデータを送ることでスケーラビリティを持たせた。その結果、クライアントの数を増やしてもサーバ側への負荷を抑えることができた。また、 VNC Refrector との性能比較も行う。

1　はじめに

普段授業を行う際、プロジェクタなどを使って授業を進めている。しかし、後ろの席から見えにくいなどの不便を感じることがよくある。授業をうけている生徒の手元にパソコンがあるならば、そこに先生のスライドを表示して授業を進めれば後ろの席に座っても手元に画面があるので見えづらいという問題は解消される。

VNC（画面共有）を使えば、スライドを生徒の手元の画面に表示することができる。しかし、多人数の生徒が先生のパソコンに同時に接続してしまうと処理性能が落ちて授業の進行に画面がついていかなくなってしまう。この問題は一つのコンピュータに多人数が繋がいるときに起こる問題である。

本研究では多人数で画面共有ができるようにクライアントをツリー構造に接続させ、上から順番にデータを流していくという方法で新しい VNC クライアント、TreeVNC の設計と実装を行った。

Design and implementation of Screen Sharing System with VNC for lecture

Yu Taninari, Nobuyasu Oshiro,Shinji Kono, 琉球大学工学部情報工学科 並列信頼研究室, Dept. of The Department of Information Engineering, University of Ryukyus Concurrency Reliance Laboratory.

2　VNC について

VNC は、Rfb プロトコルを用いて遠隔操作を行うリモートデスク Top ソフトである。VNC はサーバ側とクライアント (ビューア) 側に分かれていて、サーバを起動し、クライアントがサーバに接続を行い遠隔操作を可能にする。

2.1　Rfb プロトコル

Rfb (remote frame buffer) プロトコルは、GUI 操作をリモートアクセスで行うためのプロトコルである。画面の描画の更新は画面の差分が発生した部分を矩形毎で送り描画される。また、画面の描画データに使われるエンコードが多数用意されており、また独自のエンコードを実装することもできるシンプルなプロトコルである。

3　TreeVNC の方針

まず、多人数が参加している授業で VNC を使う場合に起こる問題は、最初で述べたように、一つのコンピュータに多人数が繋がる一極集中型では、処理性能が大幅に落ちてしまうところが問題である。この問題を解決する為に、クライアント同士を接続させ、画面描画のデータを受け取ったクライアントが次のク

ライアントにデータを流すという方法を思いついた。画面共有を行いたいクライアントが一種の VNC サーバ自体にもなる。また、クライアント同士の接続はツリー構造で行うことで管理をしやすくなると考えた。クライアント同士の接続の管理はツリーの一番上にいる PC(トップ) で行い、このトップだけが VNC サーバへ接続を行うようにする。

今回作成した TreeVNC は、上記の実装でツリー状にクライアントを接続していくように実装を行った。

4 先行事例

4.1 Vnc Reflector

Vnc Reflector は、Vnc サーバとクライアントとの間に入り、Vnc サーバとの通信を代わりに行うプログラムで開発されている。クライアントは Vnc Reflector へ接続するので、Vnc サーバとの接続は Vnc Reflector のみとなり、Vnc サーバ側の負荷を減らすことができる。

5 TreeVNC の実装

TreeVNC は tightVNC の java 版の viewer を元に作成を行った。実装の細かい内容は以下で説明する。

5.1 tightVNC viewer

tightVNC は tightVNC というプロトコルをサポートしたフリーの VNC 用ソフトである。2011 年 8 月 9 日現在と C++で作成された VNC サーバ用と Windows 版、それと Java 版のビューアが公開されている。

5.2 tree structure

今回は、ホストに対しクライアントがツリー状に繋がっていくように実装した。ツリーの構成は以下の手順で行う。

1. クライアントが接続する際、ホストに接続をしているプロキシ（今後このプロキシのことを Top と記述する）に接続する。
2. Top はクライアントにどこに接続すれば良いかを知らせる。（このときに親の番号と自分の番号それからリーダーであるかどうかを一緒に知らせる）
3. クライアントは Top から指定されたノードに接続を行う。

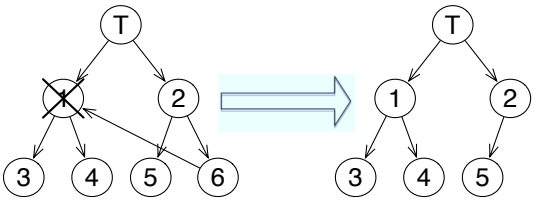

図 1 1 番の木が落ちたときの再接続の処理 (T は Top)

5.2.1 tree の再構成

今回の実装はクライアントがツリー状に繋がっているので、親ノードが落ちると子ノードも一緒に落ちてしまう。そこで、tree の再構成が必要になる。

1. 親ノードが落ちた際に、子ノードの中で一番若い番号の子ノードが Top に対して自分の親ノードが落ちたことを報告する。（親ノードの番号を知らせる）
2. Top は木の番号が一番大きいノードに対して 1 で報告を受けた親ノードの代わりになるように命令を出す。
3. 親ノードがいなくなった子ノードたちは Top に対して、2 で新しく繋がった親ノードの IP アドレスを教えてもらいそのアドレスに対して接続をおこなう。上記の構成の場合、一つのノードが落ちた場合に再接続を行うノードは 2 分木の場合 3 ノードである。

5.3 クライアントとの通信

TreeVNC は、受け取った画面の描画データをそのまま自分に繋がっている次のクライアントに送信する。描画データを受け取ったクライントはまた次のクライアントへデータをそのまま送信する。内部では、まず受け取った描画データの読み込みを先に行い Bytebuffer でコピーを行い、次のクライアントへの送信と自身のビューアへの描画を並列に行っている。

5.3.1 FramebufferUpdate

Rfb プロトコル での画面の描画の更新は、FramebufferUpdate 単位で行われる。FramebufferUpdate は更新を行う画面の矩形の数がまず送られ、次に x 座標、y 座標、横幅、縦幅、エンコードのタイプ、描画データが 1 セットとなり矩形の数だけデータが送られてくる。描画データはエンコードのタイプにしたがった方法で送られてくる。

5.3.2 描画データの先読み

FramebufferUpdate で送られくるデータ量はエンコードタイプまで読みこめば知ることができる。例えば、RAW エンコードの場合は width * height * 4 のバイト数が送られてくる。4 はピクセルのデータ量である。ヘッダーと計算で求めた数の分だけバイトを読み込むことで 1 つの FramebufferUpdate 単位でコピーを行うことができる。

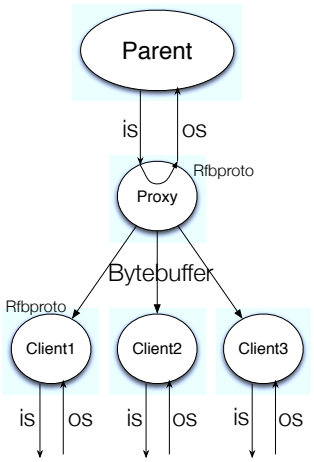

図 2 Vnc サーバ (Parent) から受け取ったデータを自身に繋がっているクライアントへ送信する。クライアントは次のクライアントへデータを送信する。

5.3.3 MulticastQueue

画面が更新された際に更新をクライアントに伝えなければならない。ノードが多数ある場合、一人一人に更新を知らせるのではなく、同時に画面の更新を知らせたい。同時に更新を知らせるために、CountDownLatch を用いて MultiCastQueue を作成した。

CountDownLatch 一回 CountDown されたときに待機しているスレッドを解放するように宣言する。更新情報が来るまで await を用いてスレッドを待機させる。更新情報が来たとき CountDown を行う。すると、スレッドが開放されるので同時に更新情報を参照することができる。

5.3.4 timeout

MultiCastQueue を使ってのデータの取得には問題が発生した。それは、接続してきたクライアントがデータを取得しない状況、例えばサスペンド状態になったときに Top のメモリの中にデータが残り続けるというものである。メモリに残り続けたデータはやがてメモリオーバーフローを引き起こしてしまうのである。

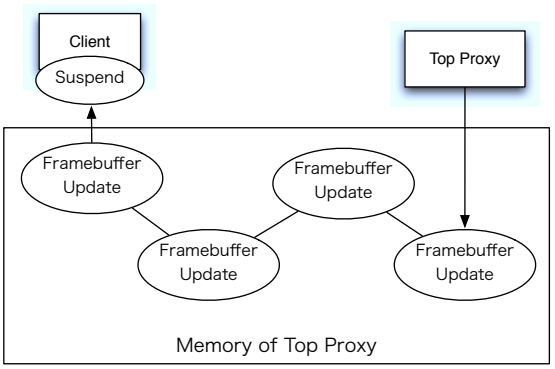

図 3 クライアントサスペンド時の Top のメモリ

そこで、ある一定の時間がたつと代わりにデータを poll してくれる TimeOut 用のスレッドを作成した。TimeOut スレッドはサスペンドしているクライアントの代わりにデータを取得する。

TimeOut スレッドがクライアントの代わりにデータを取得することで、MulticastQueue の中からデータが削除され Top のメモリを圧迫することがなくなった。

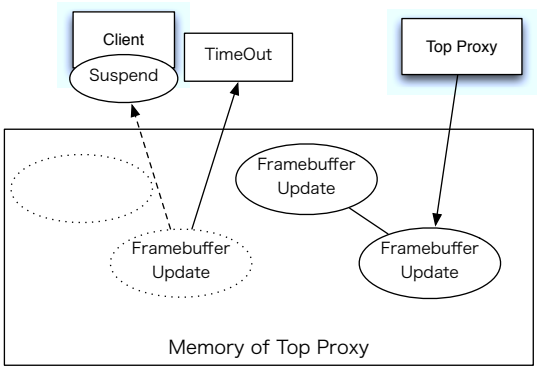

図 4　TimeOut が poll を行う

5.4　圧縮の問題

VNC で扱う Rfb プロトコルには、使えるエンコーディングのタイプの 1 つとして ZRLE(Zlib Run-Length Encoding) がある。ZRLE は Zlib で圧縮されたデータとそのデータのバイト数がヘッダーとして付けられ送られてくる。Zlib はフリーのデータ圧縮及び解凍を行うライブラリである。可逆圧縮アルゴリズムの deflater を実装している。deflater を用いてデータの圧縮が行える。

5.4.1　java.util.zip.deflater のバグ

deflater はプリセット辞書をもち、Zlib 圧縮されたデータはその辞書を用いて解凍が行われる。辞書は更新されることもあるので Zlib 圧縮されたデータを解凍する為には辞書のデータも受け取る必要がある。しかし、Java にはこの Zlib の辞書を相手へ書きだす (flush) する機能が無い。元々の Zlib の規約にはこの辞書を flush する機能があったが Java には実装されていなかった。

5.4.2　ZRLEE(ZRLE Economy)

そこで、Top が ZRLE で受け取ったデータを unzip し、データを zip し直して最後に finish() をいれることで初めからデータを読んでいなくても解凍を行えるようにした。このエンコードは ZRLEE エンコードと定義した。一度 ZRLEE エンコードに変換してしまえば、そのデータをそのまま流すだけで良い。よって変換は Top が行う一回だけですむ。ただし、deflater では前回までの通信で得た辞書をクリアしないといけないため、クライアント側では毎回 deflater は新しいものを使うことになる。また、ZRLEE はクライアント側が対応していなければならないという問題がある。

5.4.3　ZRLE と ZRLEE のデータ圧縮率の比較

RAW,ZRLE,ZRLEE のデータ量の比較を行った。図 5 は 1920 * 1080 の画面の全描画にかかるデータ量を測った結果を示した図である。ZRLEE の方がデータ量が少なくですんでいる。これは、ZRLE(Zlib) が初めに送られた辞書を用いての解凍が余り有効的に働いていない場合があるからだと思われる。つまり VNC の場合は ZRLEE の様に毎回辞書のデータを付加させて送ってもデータ量に差がでないのである。

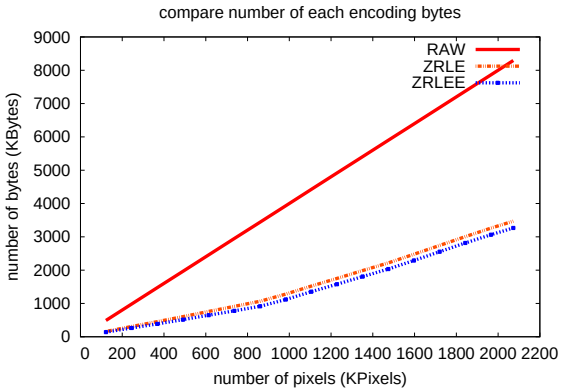

図 5　RAW,ZRLE,ZRLEE による 1 画面 (1920*1080) 描画にかかるデータ量

6　評価

TreeVNC の実装を行い、Vnc Reflector との比較を行った。テスト環境は Blade サーバ上に VM を 48 台用意して行った。Blade サーバの外に TreeVNC と Vnc Reflector を起動させた PC を置き、VM48 台にアクセスさせた。

6.1　Vnc Reflector との比較結果

一極集中型の Vnc Reflector は、スループットが 2 メガから 5 キロまで落ちた。一方 TreeVNC の方はスループットを 2 メガのままをキープすることが確

認できた。これは、1 本の通信帯へ 48 台がアクセスする Vnc Reflector と違い、クライアント同士が通信を行う為である。

6.2 TreeVNC の利点

1. クライアント同士がデータのやり取りをするので Vnc サーバへ負荷が少ない。
2. 一極集中型ではないので、多人数で使う際に画面表示のストレスが少ない。
3. Top 自身がビューアを持つこともできる。

6.3 TreeVNC の欠点

1. 独自のエンコーディング形式を使っているので、圧縮する場合専用のクライアントが必要。
2. みんなの通信速度が上がる分、スイッチへの負荷が高い。

7 まとめ

本研究では、VNC と Tree 構造を用いて画面共有システムを開発した。結果スループットを落とすことなく 48 台で接続を行うことがきた。

Zlib 圧縮に使用される java.util.zip.deflater のバグも発見したので、代わりとなるエンコード ZRLEE の実装を行った。

クライアントへのデータ転送をマルチキャストで行う等、実装には改良の余地がある。

8 参考サイト

RFB Prococol 3.8
http://srgia.com/docs/rfbprotocol3.8.html
2011:08:08

RFC 1950 ZLIB Compressed Data Format Specification version 3.3 日本語訳 - futomi's CGI Cafe
http://www.futomi.com/lecture/japanese/rfc1950.html
2011:08:08