Vol.2011-IOT-12 No.14
2011/2/28

P2P を利用した画面配信システムの性能改善に関する研究

杉田裕次郎† 白澤竜馬† 亀澤健太† 松下翔太† 東剛秀†
田中貴章† 小田謙太郎† 下園幸一† 山之上卓†

P2P を利用した教育支援システム SOLAR-CATS の画面配信システムの性能改善について述べる. SOLAR-CATS は端末同士を P2P 接続し平衡二分木状のオーバレイネットワークを構築することによりデータ配信にかかる時間を大幅に削減し, 接続端末数に関して高いスケーラビリティを実現している. この画面配信システムに新たに VNC を利用したキャプチャ, LZO や QuickLZ などの LZ 系圧縮, JNI を用いたネイティブの圧縮コード, Java New IO を用いたネットワーク処理などを適用し, 1 対 1 通信を行った際のフレームレート, CPU 使用率, 通信帯域を測定し性能を評価した. 実験結果から VNC を利用した画面キャプチャと LZ 系圧縮による画像データの圧縮は性能改善に大きな効果があることを確認した.

Performance Improvement of the Screen Broadcasting Sub-System using P2P

YUJIRO SUGITA†, RYOUMA SHIRASAWA†,
KENTA KAMEZAWA†, SHOTA MATSUSITA†,
TSUYOHIDE HIGASHI†, TAKAAKI TANAKA†,
KENTARO ODA†, KOUICHI SIMOZONO†,
and TAKASHI YAMANOUE†

This paper shows performance improvement of the Screen Broadcasting Sub-System of an educational supporting system named SOLAR-CATS. This system can minimize the transfer time of image data and maximize scalability for the number of terminals with a balanced binary tree overlay P2P network. We evaluated the performance by measuring a frame rate, CPU utilization, and band-width utilization in the case of with and without applying the improvemant techniques: screen capture mechanism with VNC, LZ compressions such as LZO and QuickLZ, and network processing with Java New IO. From our results, we find that the capture mechanism with VNC and image compressing with LZ compressions are effective for performance improvement.

† 鹿児島大学
Kagoshima University

1. はじめに

近年, 多くの教育機関に情報端末室が設置され PC 等の情報端末を利用した教育が一般化している. しかし, その利用法としては PC そのものの使い方を学習するような場面が多く情報端末を十分に活用出来ているとは言えない. 情報端末の持つ能力を発揮するための仕組みを提供することで様々な講義において情報端末を有効利用することができると考えられる. 我々は P2P 技術を利用しネットワークを通じて協調作業などが行える教育支援システム SOLAR-CATS[1) 2)]を開発した.

SOLAR-CATS は, 複数人で同時編集可能なお絵かきツール, 英作文支援ツール, スクリプトの実行, ファイル転送, チャット, SOLAR-CATS 上の操作記録・再生機能などの様々な機能を持ち, その中の一つとして 1 台の PC のデスクトップの様子をグループ内の PC のディスプレイにリアルタイムに表示するための画面配信システムを備えている. SOLAR-CATS では画面配信システムや協調作業機能を実現するために P2P を利用した平衡二分木状のオーバレイネットワークを構築する. これにより, 大規模な数の端末へ実時間でデータ配信することが可能となり, 端末数の増大に対して高いスケーラビリティを実現している. このときネットワークの構築に TCP を用いることで双方向通信を可能とし, また, データ配信に IP マルチキャストを使わないことにより LAN 内のトラフィックの増大を抑え, IP マルチキャストに対応していないルータをまたいだサブネット同士でも通信を可能としている. また, P2P を利用することで新たにハードウェアを購入したりサーバを設置したりせずにシステムを利用できるため導入が容易に行える. さらに Linux や Mac OS などの OS の違いを意識することなくシステムを動作させるために Java で開発を行いクロスプラットフォームなシステムを実現している.

本研究は SOLAR-CATS の画面配信システムの性能改善を目的とする. 新手法として VNC(Virtual Network Computing)を利用した画面キャプチャ, OpenVPN などに採用されている LZO[3)]や圧縮・伸張速度を重視して開発された QuickLZ[4)]などの LZ 系圧縮, JNI[5)]を用いた圧縮コードのネイティブ化, Java New IO[6)]を用いたネットワーク処理の高速化などを利用する. 画面配信システムを実現するための各工程にこれらの手法を適用しフレームレート, CPU 使用率, 通信帯域を測定し従来手法と比較することで性能改善への効果を検討する. また, 今後の目標として UltraVNC[7)]から提供されている Windows 用のビデオドライバである UVNC Mirror Driver を利用した新たなキャプチャ機構についても述べる.

Vol.2011-IOT-12 No.14
2011/2/28

2. 関連研究

画面配信を提供する関連製品として VNC 系ソフトウェアの一つである MultiVNC がある．クライアント・サーバ(CS)型システムである MultiVNC はクライアントである学生の PC の様子をサーバである教師 PC に一覧表示したり，サーバや任意のクライアント PC のデスクトップの様子を全体に配信することができる [8]．しかし，MultiVNC は教師(サーバ)側からのみの操作しかできない． SOLAR-CATS は双方向通信が可能なため協調作業を行うようなアプリケーションを動作させることができ，また，大規模な数の端末の接続にも対応できるスケーラビリティを備えている．

画像圧縮に関する既存技術として動画圧縮コーデックである H.264 や MPEG などを用いたストリーミング技術がある．これらは高い圧縮率と高画質を実現できるが負荷が高く遅延が大きくなると考えられるためリアルタイム性が求められる画面配信には向かないと考える．表 1 に SOLAE-CATS と関連技術との特性の比較を示す．

表 1. 関連技術と SOLAR-CATS の特性

	ストリーミング技術	MultiVNC	SOLAR-CATS
低帯域使用	○	○	○
高品質	○	○	○
低遅延	×	○	○
低負荷	×	○	○
双方向性	×	×	○
スケーラビリティ	△	△	◎

3. 画面配信システムの概要

SOLAR-CATS はデータ配信のために P2P で平衡二分木のオーバレイネットワークを構築する(図 1)．グループに参加している端末数を n とすると，すべてのノードにデータを配信するのに要する時間は，平衡二分木では $O(\log_2 n)$とノード数の対数(木の深さ)にほぼ比例した値となるのに対し，クライアント・サーバ(CS)型では O(n)と端末数に比例した値となる．n が 1000 台規模となったときの配信時間の差は 100 倍程度のオーダーとなり端末数の規模に大して非常にスケーラブルであると言える．平衡二分木型と CS 型の概略を図 1 に示す．

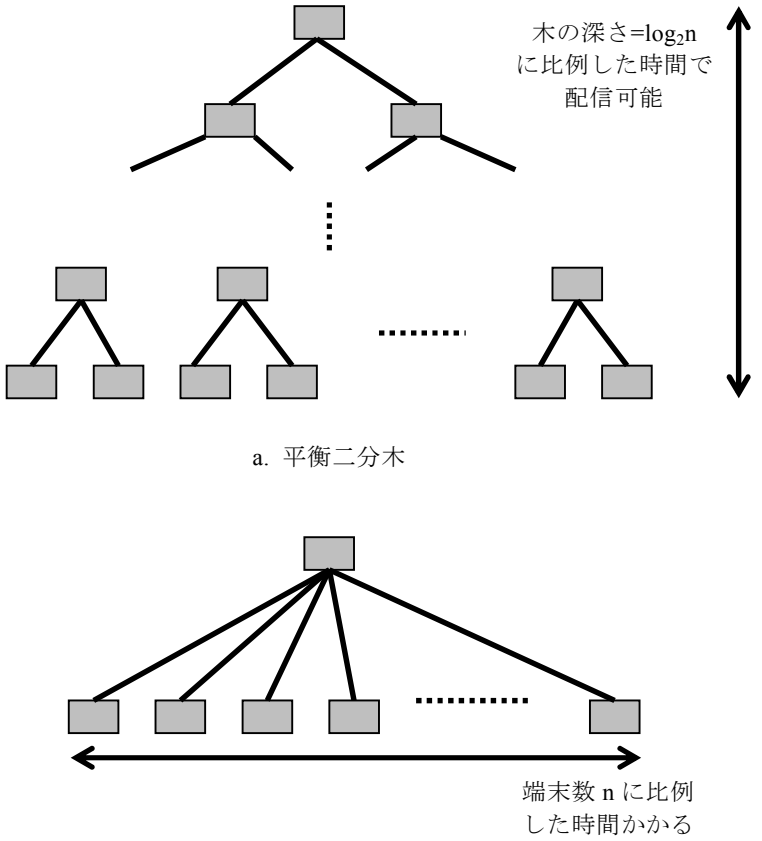

図 1. 平衡二分木と CS 型のネットワーク

図 2 は画面配信システムの概要を示している．送信端末ではまず画面キャプチャ処理を行う．次に取得した画像データを圧縮し，タイムスタンプなどのヘッダー情報を付加した後ネットワークに送り出す．画像データを受信した端末は自分に接続されているその他のノードへと転送した後，画像データを伸長しディスプレイに表示する．

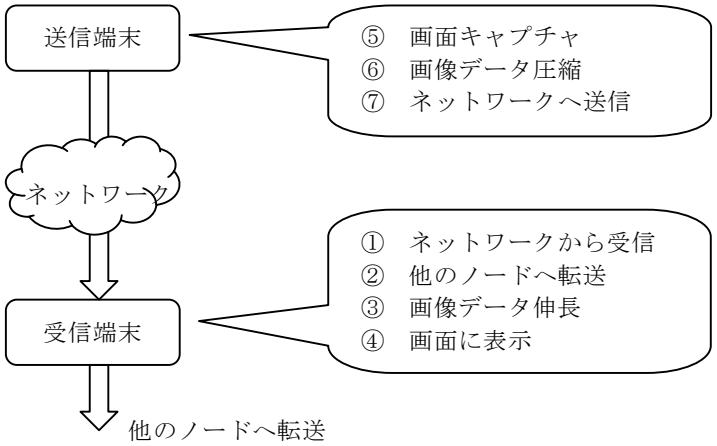

図 2. 画面配信システムの概要

4. 現在の問題点

図 2 に示されるように，画面配信システムを実現する工程として画面キャプチャ処理，画像データ圧縮・伸長処理，ネットワーク入出力処理等がり，現在のシステムにはそれぞれについて問題がある．

キャプチャ処理において以下のような問題点がある．

1. 画面に変化があった場合のみキャプチャすることで通信帯域を削減したいが，画面を監視し変化を検出するための処理が必要．
2. 変化領域の大きさに関係なく画面全体をキャプチャしてしまうため，変化領域以外の必要ない部分も処理しなければならず非効率．

また，圧縮・伸長処理のコーデックとして JPEG, SlowFine，無圧縮の 3 つが実装されているが，それぞれ以下のような問題が挙げられる．

1. JPEG は圧縮に時間がかかりフレームレートが低くなってしまう．また，単色の塗りつぶしやテキストが多いデスクトップ画像は，JPEG の特性上画質が低下しやすい．
2. SOLAR-CATS 独自のコーデックである SlowFine は，画面に変化があった初期段階では低解像度の画像を送り徐々に精細にしていくというアルゴリズムを使用している．そのため画面に連続して変化があると低解像度の画像が続いていまう．また，圧縮率も十分でないため帯域幅を圧迫している．
3. 圧縮の場合はデータサイズ大きく通信帯域を消費するため，ネットワーク入出力がボトルネックとなりフレームレートも低い．

5. 提案手法

前述した問題点を解決し画面配信システムの性能向上を図るために，各工程において以下のような手法を適用することを提案する．

1) VNC を利用した画面キャプチャ

遠隔操作用のオープンソースソフトウェアであり, Windows や MacOS などの様々なプラットフォームで提供されている VNC を画面キャプチャに利用する．VNC を用いることによりドライバレベルでのキャプチャが行えるため高速化が期待できる．また，画面に変化があるまで待機することが可能なため変化を検知する処理が必要なくなり CPU 使用率を低減することができると考えられる．今回は SOLAR-CATS に VNC クライアントの機能を実装し，ローカルホストで起動した VNC サーバに接続し TCP コネクションを通じて画像データを取得する．

2) LZ 系圧縮とフレーム間差分

画像データの圧縮に高速な圧縮アルゴリズムである LZO や QuickLZ などの LZ 系圧縮を使う．これらのアルゴリズムは圧縮速度が高速である反面圧縮率が低い．そこで，画面上の変化があった領域は多くの場合画面全体の一部分であることに着目し，キャプチャした画面と一つ前の画面とのピクセル値の排他論理和を算出した差分データを抽出する．この差分データは冗長度が高いため LZO や QuickLZ でも圧縮率を高くすることができ，速度と圧縮率を両立した圧縮が可能になると考えられる．

3) JNI を用いたネイティブの圧縮コードの利用

JNI は Java のプログラムと C 言語などで記述されネイティブで動作するプログラムとを連携させるための API である．JNI を用い Java 仮想マシン上ではなくネイティブで動作する圧縮コードで圧縮処理を行うことで高速化が期待できる．

4) Java New IO を利用したネットワーク処理

Java New IO は高速な入出力処理や非同期入出力が可能な API である．Java New IO を使うことでネットワーク入出力の高速化が期待できる. また, JNI を用いて Java のプログラムがネイティブコードと連携する場合，通常 Java コードが使うヒープ領域とネイティブコードが使うメモリ領域との間でコピーが発生する [6]. Java New IO はネイテ

ィブコードから直接アクセスできる Java ヒープ外にメモリを確保することができるためコピーの発生を抑えることができるため高速化が期待できる．

6. 実験

画面配信システムを実現するための各工程に様々な方法を適用したときのフレームレート, CPU 使用率, 通信帯域を測定し, 提案手法を用いることによる性能改善への影響を検証する．性能評価に用いるこれら３つの項目は, フレームレートをあげようとすると CPU 使用率が増大したり, CPU 使用率を抑えようとするとフレームレートが上がらなかったり通信帯域が増大したりするなどトレード・オフの関係にあるため, これら３項目のバランスで性能を評価する．

6.1 実験環境と方法

本実験ではキャプチャや圧縮処理に関する性能を評価するためにオーバレイネットワークは構築せず, 1 対 1 通信で実験を行う. 実験環境は, OS が Windows Vista, ネットワーク帯域幅が 100Mbps, 配信する画像の解像度が 1440x900 ピクセル, ビット深度が 24bits/pixel, CPU が Core2 Duo 2GHz, Java 仮想マシンが Java1.6 である．配信するデスクトップの様子を毎回同じものにするために, Windows 自動化ツールの UWSC[9]を用いた．UWSC で再生した操作を様々な方法で配信したときのフレームレート, CPU 使用率, ネットワーク使用率を測定する．各工程における検討項目を以下に挙げる．まず, キャプチャ処理では Java 標準クラスを用いる場合と VNC によるキャプチャを行う場合とを比較する．VNC は Windows 用に最適化された UltraVNC を用いる．尚, Java 標準クラスを用いたキャプチャでは画面の変化の検出処理は行わない．圧縮・伸長処理では従来の方法である JPEG, SlowFine, 無圧縮と新たに提案するフレーム間差分を併用した LZ 系圧縮とを比較する．また, Java プログラムで画像データを圧縮する場合と JNI を用いネイティブコードで圧縮処理を行う場合とを比較し, Java 仮想マシンによるオーバーヘッドを検証する．そして, ネットワーク処理においては従来の Socket クラスを用いる方法と Java New IO の SocketChannel クラスを用いた場合とを比較する．以上の検討項目を表 2 に示す．

表 2．各工程における検討項目

キャプチャ処理	Java 標準クラス, UltraVNC を利用したキャプチャ
圧縮・伸長処理	無圧縮, JPEG, SlowFine, LZ 系圧縮＋フレーム間差分, ネイティブ化した LZ 系圧縮＋フレーム間差分
ネットワーク処理	Socket クラス, Java New IO の SocketChannel クラス

6.2 実験結果と考察

実験結果を表 3 に示す．LZ 圧縮にはフレーム間差分を併用している．

表 3．実験結果

適用方法	フレームレート[fps]	CPU 使用率[%]	通信帯域[Mbps]
無圧縮	2.3	24	74
JPEG	3.5	55	5
SlowFine	10.1	62	37
LZ	9.8	58	7
LZ + JavaNewIO	10.1	64	6
LZ-Native	11.4	49	9
LZ-Native + VNC	11.6	42	10

無圧縮の場合, 解像度が 1440x900 の 1 画面当たりのデータサイズは約 30Mbits になる．そのため帯域幅が 100Mbps のネットワークでは理論上フレームレートは最大でも 3fps 程度にしかならない．実験では通信帯域を 74Mbps とほぼ使い切っており, フレームレートも 2.3fps と低かった．JPEG は通信帯域を 5Mbps と非常に低く抑えられているが CPU への負荷が大きく, フレームレートも 3.5fps と低い値にとどまった. また, 前述したようにデスクトップ画像は色が均一な領域が多いため画質の低下が著しく, とくに小さいサイズのテキストは読みづらさが目立った．独自のコーデックである SlowFine では, フレームレートが 10.1fps と上の 2 つに比べ高くなった．しかし CPU 使用率が 62[%]と高負荷であり, 通信帯域も 37Mbps と大きく消費している．

提案手法である LZ 圧縮を用いた場合, SlowFine と同程度のフレームレートと CPU 使用率でありながら通信帯域を 7 Mbps と SlowFine の 1/5 に抑えることが出来た．ネットワーク処理に Java New IO を用いた場合, フレームレートが 0.3fps しか向上せずほとんど効果が無かった．ネイティブの圧縮コードを用いた場合, フレームレートが Java の圧縮コードに比べ 2fps 程度向上した．しかし圧縮コードのネイティブ化による劇的なパフォーマンスの向上は得られなかった．SOLAE-CATS はクロスプラットフォームをポリシーとしているためネイティブコードを利用するならプラットフォームごとにネイティブコードを用意する必要があり, その労力の割に得られる効果は大きくないと言える．さらに VNC を利用したキャプチャではすべての項目において最も良い結果が得られた．また, VNC でキャプチャを行うことで画面に変化がないときの CPU 使用率を非常に低く抑えることができた．Java でキャプチャを行う場合でも画面に変化が起こった時だけキャプチャを行うなどの最適化を行えば CPU 使用率を減らすことはできる．しかし, 画面の変化を検出するためには任意のタイミングで画面をポーリングし画像全体を走査し前の状態のピクセル値と比較する処理が必要になるため, 画面に変化がなくても CPU は稼動しなければならない．ここで, Sleep を使ってポ

ーリングの回数を減らしたりマウスの動作からポーリングのタイミングを最適化したりするなどの方法が考えられるが, 前者の方法で CPU 使用率を抑えようとするとSleep 時間を長しなければならずそれだけ遅延が発生する可能性が高くなり, 後者は動画の再生などのマウスに関係なく変化するような場合に対応できない. よって Java だけで VNC ほど CPU 使用率を低く抑えるのは難しいと考えられる.

7. まとめ

画面配信システムの性能の向上を図るために, VNC を利用したキャプチャ処理, LZ 系圧縮による画像データの圧縮, JNI を用いた圧縮コードのネイティブ化, Java New IO を用いたネットワーク処理を試みた. 実験の結果から VNC による画面キャプチャは CPU 負荷の軽減やフレームレートの向上に有効であり, LZ 系圧縮を用いた画像データの圧縮は従来より高速・高圧縮でフレームレートの向上や通信帯域の削減に有効であることが分かった. 可逆圧縮である LZ 系圧縮を用いると劣化がないため画像が高品質になるとも確認できた. また, 通信帯域を大幅に削減できたことで従来よりさらにスケーラビリティが向上すると考えられる. JNI による圧縮コードのネイティブ化では期待していたほど劇的な効果は得られなかった. ネットワーク処理に Java New IO を用いる事による効果はごく僅かであった.

本実験では Windows に最適化された UltraVNC を用いた. UltraVNC を用いたキャプチャには以下のような問題が挙げられる.

1) ウィンドウをマウスで動かしたときなどに残像のように画像が乱れる.
2) ローカルホストで起動された VNC サーバに接続して TCP コネクション経由で画像を取得するため画面キャプチャのためにネットワーク処理が発生する.

これらの問題を解決するために, UltraVNC から提供されている UVNC Mirror Driver(UVNC MD)を利用することを検討している. UVNC MD は VRAM のコピーを常に保持しておく機能や画面に変化があったときにプッシュ通知する機能を持っている. また, 変化があった領域の座標, 高さ, 幅等を取得することができるため, ピクセルデータが格納されたメモリ領域から変化領域のピクセルデータのみを取り出すことができる. そのため, データ量や CPU 使用率を低減できるため高速かつ効率のよい画面キャプチャが実現できると考えられる. 今後, UVNC MD を利用した Java によるキャプチャ機能を実装することにより, UltraVNC を利用したキャプチャのような画像の乱れやネットワーク処理が発生しない画面キャプチャ機構の実現を目指す.

参考文献

1) 山之上卓, "P2P技術を利用した分散システム上の実時間操作共有システム",情報処理学会論文誌,vol.46, No.2, p.392-402,2005.
2) 山之上卓, ”構造型P2Pシステム上の放送におけるより良いデータ分割方法”, 電子情報通信学会信学技報, IA2007-18, 2007-7.
3) 杉田裕次郎, 山之上卓, 小田謙太郎, 下園幸一, ”スケーラブルな画面配信システムの性能改善”, 第63回電気関係学会九州支部連合大会, 2010年 9月 (福岡)
4) LZO -- a real-time data compression library, http://www.oberhumer.com/opensource/lzo/lzodoc.php
5) QuickLZ 1.5.x, http://www.quicklz.com/download.html
6) Java Native Interface JNI 5.0の仕様 オブジェクトの参照, http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/jni/spec/design.html#wp16785
7) New I/O API, http://java.sun.com/j2se/1.4/ja/docs/ja/guide/nio/
8) UltraVNC, http://www.uvnc.com/
9) 芝崎亮, 千葉大作, 中沢実, 服部進実, ” IT教育向けデスクトップ管理ツール「MultiVNC」の実践報告”, 情報処理学会シンポジウム論文集, 2005号8項 69-74, 2005.
10) UWSC, http://www.uwsc.info/