Application Specific Compression for Remote Visualization of Genomics Applications

Lars Ailo Bongo[1], Kai Li[2], Olga Troyanskaya[2], Tore Larsen[1], and Grant Wallace[2]

[1] Department of Computer Science, University of Tromsø, Norway
[2] Computer Science Department, Princeton University, NJ, USA
{lbongo, li, ogt, tore, gwallace}@cs.princeton.edu

Abstract. Sensitive data can be shared by only sharing visualizations. However, smooth remote interaction with the data requires higher bandwidth than practical in a WAN. We describe and present preliminary results for compression schemes exploiting redundancy in data transmissions to reduce bandwidth usage.

1 Introduction

For remote collaboration where sensitive data is analyzed, management, security, and privacy concerns can be simplified by keeping all the data at a single server and only send analyzed data in the form of visualizations to clients. However, the client users should still be able to smoothly interact with high quality visualizations. Therefore, high resolution and frequent updates are needed. But the required bandwidth is not practical for wide area networks (WANs).

Existing solutions reduce bandwidth by only sending rectangles of updated pixels. But still, interactions such as scrolling may change a large portion of the screen. However, there are many redundant transmissions, since the content of the updated often region has often been sent to the client earlier, for example when scrolling up and down. In this paper we examine if we can exploit the redundancy to reduce bandwidth. As a case study we use Genomics applications, since these visualize sensitive data using high resolution bitmaps.

2 Compression for Remote Visualization

Many visualization tools use the thin-client approach including VNC [5], Microsoft Remote Desktop and Microsoft LiveMeeting. Usually, these tools are targeted for business environments where users collaborate in task such as writing a document. This type of interaction has low bandwidth requirements and is handled well by most existing thin-client solutions. We extend VNC to handle interaction requiring higher bandwidth.

Figure 1 shows the architecture of a VNC server and VNC client. The VNC viewer (client) sends user input to the VNC Server, which applies these to application running on the server. The VNC server is also responsible for detecting

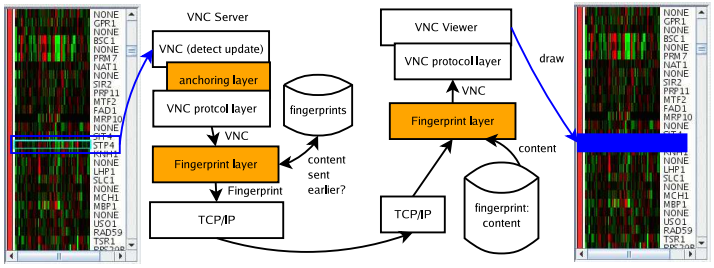

Fig. 1. A visualization from the GenVaND tool and the system architecture.

changes to screen, encode these changes using one of the VNC protocols, and for sending the encoded changes to the VNC viewer. The viewer then decodes and draws the received data.

We have extended VNC with a fingerprinting and anchoring layer. The anchoring layer is responsible for dividing the updated screen regions into subregions that allows finding repeated content. The fingerprinting layer calculates a fingerprint for each subregion and checks if a subregion with a similar fingerprint has been sent earlier. If so, only the fingerprint is sent. When the fingerprinting layer receives only a fingerprint, it uses the fingerprint as an index to a cache to get the content of the fingerprint. The content is then encoded using a VNC protocol and sent to the VNC viewer which updates the client visualization.

We use 32 bit Rabin fingerprints [4, 2, 1] as anchors to find regions of redundant data. Using fingerprints to reduce the data sent over a WAN has been used for example to improve Web performance [6], or for a network file system designed for low-bandwidth networks [3].

Key to finding repeatable content is to select anchoring points that do not change when only a part of the screen is updated. We evaluate anchoring schemes along three dimensions: (i) data compression, (ii) added overhead, and (iii) cache storage at the client. In this paper we evaluate three schemes:

1. Fixed hextile anchoring, where anchoring points are fixed to 16x16 pixel rectangles at fixed screen coordinates.
2. Content hextile anchoring, where anchoring is based on fingerprints calculated for 16x16 pixel regions of the data being displayed on the screen.
3. Content based 1D byte stream anchoring, where the 2D screen is treated as a 1D byte stream (in row order).

3 Preliminary Evaluation Results

We measured the bandwidth requirements for remote visualization of a Genomics application, and we compared the bandwidth reduction of the three compression schemes described in the previous section.

For benchmarking we use an automated GUI testing tool that allows recording and playing back user input events such as mouse movement and key types.

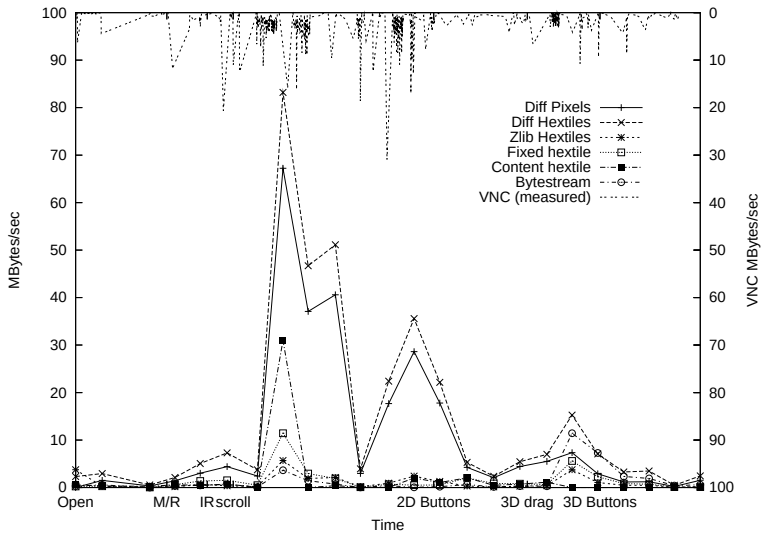

Fig. 2. Bandwidth usage for uncompressed, hextile, actual VNC (top), zlib compressed, fixed hextile anchoring, content based hextile anchoring, and 1D byte stream anchoring.

For performance experiments the time between two events is important. Also, this time should increase if the remote visualization is slow. Since the existing non commercial tools do not fulfill these two requirements we wrote our own tool. To record data we instrument our Genomics applications, all written in Java, with classes implementing the AWTEventListener interface. For playback we use the Java util.Robot class. We use pixel based synchronization to ensure that the client driving the interaction does not proceeds until its display is coherent with the servers display. Screens are synchronized at events 50 ms apart in the recorded session.

We use the xf4vnc VNC server since it supports OpenGL, which is used by one of our tools for 3D visualization. The server was run on a Dell Dimension 8400 with a 2.8 GHz Pentium 4 with 2 GB RAM. The VNC viewer on a Compaq Evo N620c laptop with a Pentium-M 1.5 GHz processor and 1 GB RAM. Both run Fedora Core 4. In addition we used an IBM eServer running FreeBSD with Dummynet to emulate WAN networks.

To get realistic traces we instrumented three widely used genomic applications: GeneVaND, Java TreeView and TIGR MeV. These were then used for real genomic analysis. We also recorded a synthetic user interaction for the GeneVAnD application, with time periods where the user does only one type of data manipulation (type, resize windows, move windows, scroll 2D model, change 2D model visualization settings, and drag 3D model). All results presented in this paper are for this session.

Figure 2 shows bandwidth requirements for the synthetic user session. Bandwidth requirements depends on the type of interaction, with scrolling the 2D

model having the highest bandwidth requirement. The bandwidth requirements are much larger than available bandwidth on todays WANs. Using *iperf* we measured bandwidth from Princeton to: *MIT*: 2.1 MBytes/sec, *GA* in San Diego: 0.4 MBytes/sec, and *Tromsø* in Norway: 0.2 MBytes/sec.

On a Gigabit Ethernet xf4vnc is not able to utilize more than 1.25 MByte/sec of bandwidth due to VNC server performance. Other servers such as UltraVNC on Windows have better performance and can provide smoother interaction.

Figure 2 shows how VNC reduces bandwidth by skipping updates, how zlib compression reduces bandwidth, and how our different history based compression schemes reduce bandwidth. One single compression scheme is not best all the time. But the total compression ratios are similar: fixed hextile (0.11), content based hextile (0.12), bytestream (0.13).

The storage requirements and number of fingerprinted regions for the anchoring schemes differ. Fixed hextiles requires 36.2 MBytes with 49837 fingerprints, content based hextiles 41.9 MBytes with 13030 fingerprints, and bytestream 34.1 MBytes with 60600 fingerprints. We are analyzing how these factors influence the performance improvement. All compression results were calculated offline using screenshots captured during playback of the synthetic trace.

4 Conclusion and Future Work

We have shown why compression is necessary for remote visualization of Genomic data over WAN, and that existing compression schemes are either inefficient or reduces quality of service. Instead we propose using content based fingerprinting to avoid sending redundant data. We describe four anchoring schemes for detecting redundant regions of updated screen pixels. Initial results shows that our compression reduces bandwidth usage. Also, we find that VNC server performance is limited by the server when network bandwidth is 10 Mbit or better.

Currently we are systematically analyzing the bandwidth reduction of other anchoring schemes, in addition to implementing a prototype which will be used to measure performance improvements.

References

[1] Broder, A. Some applications of rabin's fingerprinting method, 1993.

[2] Manber, U. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conference* (1994), pp. 1–10.

[3] Muthitacharoen, A., Chen, B., and Mazieres, D. A low-bandwidth network file system. In *ACM SOSP '01* (2001).

[4] Rabin, M. Fingerprinting by random polynomials. Tech. Rep. 15-81, Harvard University, 1981.

[5] Richardson, T., Stafford-Fraser, Q., Wood, K. R., and Hopper, A. Virtual network computing. *IEEE Internet Computing 2*, 1 (1998), 33–38.

[6] Spring, N. T., and Wetherall, D. A protocol-independent technique for eliminating redundant network traffic. In *SIGCOMM* (2000), pp. 87–95.