Proximal Workspace and VNC

Cynthia Taylor, Taurin Tan-atichat, Joe Pasquale, Amin Vahdat
University of California, San Diego
Introduction

- Workspace
- The Problem with Supporting Video
- Server Push
- Client Pull
- Virtual Network Computing
- Defining Performance

Adding a Message Accelerator

Experimental Design & Results

Conclusion
Workspace Architecture
Workspace Utilities

Display Forwarding

Sensor Input

Video Annotation
The Problem with Supporting Video

- Video is hard for Thin Client Systems
  - Frequent updates
  - Many pixel changes per update
  - All server generated
  - Becomes drastically worse over high latency
Server Push

- X-Windows is a server push system

VNC is a client-pull system.

Virtual Network Computing

- VNC is a widely-used thin client system.
- It is cross-platform and has several available open-source implementations.
- It was developed by Tristan Richardson at the Olivetti Research Lab.


How VNC Works

- It runs at the application layer and reads updates from the framebuffer.
Defining Performance

1. Client requests new update

2. Client waits

3. Server sends update

4. Client processes update
Introduction

Adding a Message Accelerator
- VNC with High Network Latency
- The Message Accelerator and VNC
- Pipelining Updates
- Message Accelerator with High Network Latency

Experimental Design & Results

Conclusion
VNC with High Network Latency

- Client sends request - 200 ms
- Server sends update - 200 ms

Update Rate = 2.5 updates/second
More Generally, Update Rate = 1/RTT
Two Approaches

- Adding a proxy, unmodified client and server
- Modify the client
The Message Accelerator and VNC

- The Message Accelerator sends requests to the server at the rate the client is processing them, and quickly receives updates from the server.
- This lets the Message Accelerator adjust for latency between the client and server.
The proxy sends requests to the client at the rate the client is processing, without waiting for a request.
Message Accelerator - High Network Latency

Client reads pipelined update from proxy - 75 ms

Update Rate = 13 updates/sec
• Introduction
• Adding a Message Accelerator
• Experimental Design & Results
• Conclusion
We use NetEm to add network delays to both client and server to simulate network latency.
Results: Message Accelerator Outperforms Unmodified System
Modify the Client (Taurin Tan-atichat)

- Goal: Have a request arrive just after the frame buffer (at server) is updated
- Have client send pre-requests
  - too many requests could overload network or server
  - too few results in suboptimal performance
Our Approach: VNC-HL

- Send a pre-request periodically
  - PRP is pre-request period

- Client: upon receiving an update and processing it (including rendering), send a request and set timer to PRP

- If timer expires, send another request (and set timer)

- If update is received, process/render, and then send request and reset timer
Pipelining of Requests
Goal

- Reach a steady state where enough frame buffer requests have been injected into the system that not many more additional requests are needed
Implementation

- Modified RealVNC for Unix
- Very simple change to request loop
VNC vs. VNC-HL
FPS Improves over Time

![Graph showing FPS Improves over Time]
Conclusions

- We can improve VNC performance by
  - having a Message Accelerator mediate the update rate over network latency
  - modifying the client to aggressively send pre-requests
- By using the Message Accelerator, we do not have to modify an existing code, avoiding issues of parallel code maintenance and source code availability
- In the VNC-HL approach, we achieved high performance by adding a very simple modification to the client