The Proximal Workspace Architecture:
A Latency-focused Approach to Supporting Context-Aware Applications

Cynthia Taylor
Thesis Advisor: Joe Pasquale
Department of Computer Science and Engineering, University of California, San Diego
cbtaylor@cs.ucsd.edu

DOCTORAL DISSERTATION EXTENDED ABSTRACT

ABSTRACT

We are developing a new enhanced cloud-based computing architecture, called the Proximal Workspace architecture to allow access and interaction between lightweight devices, and applications that represent a new generation of computation-and-data-intensive programs. We are developing a new system architecture based on supporting workspaces which provide nearby computing power to the users devices and thus mediate between them and the clouds computing resources. Specifically, a workspace provides a set of middleware utilities designed to exploit local resources, and provide specific functions such as rendering of graphics, pre-fetching of data, and combining data from different servers. More generally, the workspace is designed to run any subset of activities that cannot be run on a user's device due to computation speed or storage size, and cannot be run on a cloud server due to network latency.

KEYWORDS: Thin Clients, Context-Aware, Multimedia, Latency, Distributed Computing.

1. INTRODUCTION

In this work, our goal is to make a new generation of context-aware programs accessible to lightweight devices. Our work is inspired by three parallel trends in modern computing: the ubiquity of lightweight devices such as cellphones and PDAs, the rise of computation-and-data-intensive programs in areas of ubiquitous computing, augmented/virtual reality, machine learning, and graphics, and the rise of cloud computing. While context-aware applications offer a unique and exciting way to integrate computer programs with real life, users are often hampered by bulky equipment [1, 2]. Lightweight devices make it easy for a user to move around while using the device, without being hampered by bulky equipment. And the rise of cloud computing is making it easy to access powerful computational resources from anywhere, while at the same time creating challenging new models of how devices access both data and resources [3].

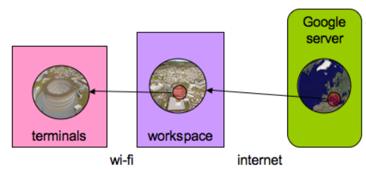

Figure 1. The workspace stores data that is too large to hold on the lightweight device.

We look at context-aware applications as presenting a new model for how data is sent between a client and server, and investigate the challenges presented by this new model. In the traditional client-server model, the client will usually request a specific piece of data from the server (e.g. a webpage, image, or multimedia file), and the server will send the client that specific item. In the new model presented by these applications, the client will send the server a location that the user is interested in, and the server will send back a large amount of information about the area surrounding that location. The user can then interactively explore this data once it is stored on the client, and the client will periodically send the user's new location to the server.

Previously, the idea of sending surrounding data has been used as a optimization for slow data transfers, i.e., pre-fetching surrounding rows in a database. In our new model, we see two things which make this new, and present new challenges. The first is the nature of the data: a typical context-aware application may send vast amounts of very detailed multimedia data to the client. The second is the nature of the user's exploration: users physically explore a three-dimensional environment in a non-linear fashion. The amount of data transferred requires significant storage capacity on the device. The combination of non-linear data retrieval by the user and the nature of the applications means that the client device must also do a significant amount of on-demand rendering of the three-dimensional map data, putting high demands on both its video card and CPU.

Moving applications to a server within the cloud in a thin-client fashion induces large delays due to network latency. To solve this problem, we propose a new system architecture whose key feature is the addition of a "workspace" as a low-latency (relative to the client) intermediary between a client and server(s). A workspace provides a set of middleware utilities designed to exploit local resources, and provide specific functions such as rendering of graphics, pre-fetching of data, and combining data from different servers. Figure 1 illustrates how the workspace fits in to this new data model.

My thesis is that a proximal workspace architecture (as describe above) will provide significantly improved user-perceived performance for thin clients running context-aware applications that are highly-interactive and computation/data intensive.

The rest of this paper is organized as follows. In Section 2, we present our workspace-based architecture. In Section 3, we discuss current and future work. In Section 4, we conclude.

2. THE PROXIMAL WORKSPACE ARCHITECTURE

A Proximal Workspace Architecture is comprised of three parts, illustrated in Figure 2: (1) *terminals*, the sensors and devices that the user actually interacts with and that provide data about the users location; (2) the *world*, consisting of the users home and/or work computers, web servers, game servers, and anything else the user interacts with through the internet; and (3) the *workspace*, a temporary computing session running on computational resources very close to the terminals, capable of extending functionality of both terminals and the world. We now describe these classes of components in more detail.

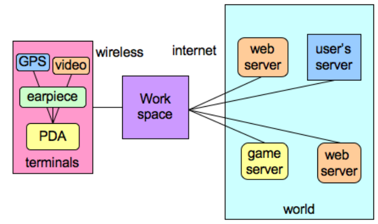

Figure 2. The system introduces a workspace between the client and servers.

Terminals consist of everything worn or carried by the user of a context-aware application.They are used for input and output. Input can be specific user actions, equivalent to mouse movements and key presses, or it can be information from sensors such as video cameras, accelerometers, GPS units, thermometers, etc. The terminals may include a hub for local communication and coordination. A set of terminals could consist of a PDA with an integrated GPS and camera, with a set of display glasses worn by the user plugged into it. Terminals form a component similar to the client in traditional thin client systems. Terminals are not required to be capable of anything more than capturing input, displaying output and communicating with the rest of the system. Terminals are expected to be in the system for long periods of time and persist over different workspace sessions.

The world is the set of servers that communicate with the rest of the system through the internet. These will be the resource servers for the application. These can include a reference website the user is visiting for the first time, or a system server with which they are in semi-constant contact. All persistent state is stored within the world. We make no assumptions about the latency between the terminals and any given server within the world. Some parts of the world may be provided by others and contain large numbers of factors we cannot control. How the world interacts with the rest of the system can have a large impact on system performance and user experience. How long a part of the world remains in the system can very from seconds to decades, and parts can remain consist over different workspace sessions.

Within the workspace there exists a set of middleware utilities that are designed to exploit local resources. These utilities are designed for specific functions, such as rendering of graphics, pre-fetching of data, and combining data from different servers. More generally, the workspace is designed to run any subset of activities that cannot be run on the client due to computation speed or storage size, and cannot be run on a server due to network latency. We aim for as much reuse between different applications as possible from these utilities. We define a *workspace server* as the machine the workspaces run on, and a *workspace session* as an individual instance of a workspace interacting with a single set of terminals on a specific workspace server.

3. CURRENT AND FUTURE WORK

We are currently working on adapting Google Earth Ancient Rome 3D to run under this system architecture, with the goal of the user being able to intuitively navigate through renderings of Ancient Rome in video glasses, without being hampered by any bulky equipment. The user interactively explores a rendering of ancient Rome, either with a keyboard and mouse or with a more sophisticated input device, and while they explore the Google Server periodically sends a very large amount of multimedia data describing the area they are exploring to the client, which then renders it on a frame-by-frame basis. This puts two burdens on the client: it must be able to store the vast amounts of data being sent, and it must be able to quickly render a complex scene.

In adapting Ancient Rome 3D to our system architecture, we must consider both where parts of the application should be distributed, and what utilities must be created to aid the distribution. The user explores Google Earth using the Space Navigator, a joystick like device that records both rotation and pressure around the x, y and z axes [5]. Since the Google Earth application cannot meet real time performance demands while running on the netbook, we move it to the workspace. Once Google Earth is running on the work space, we must create utilities to forward input from the netbook to Google Earth, and forward the display updates from the workspace server to the netbook. In order to forward the display, we use TightVNC [6]. In order to forward input from the Space Navigator, we run a program on the netbook which forwards the raw input from the Space Navigator to the workspace server, where it is then aggregated, translated into units appropriate for Google Earth, and sent to the Google Earth application via API calls.

By adapting Google Earth Ancient Rome to the Proximal Workspace architecture, we allow users to use the application on a lightweight netbook, something that it would be impossible to do running the application natively. Because of the low latency connection between the workspace server and the terminals, the performance of the application is quite good, with no lag between the display of frames.

Our work with Google Earth Ancient Rome has lead us to focus on the issues of how to best forward and transform I/O device information. As such, we are currently investigating the best way to handle I/O information with regards to the additional latency added by the network, with the goal of being as transparent to applications as possible.

4. CONCLUSIONS

We have presented a new architecture to support context-aware applications that are highly computationally intensive and that are accessed via lightweight devices, but that are too lightweight to support their actual execution. As these applications are highly interactive, it is imperative that they still execute "near" the user. Consequently, a dynamically allocated workspace that provides computational and memory resources, that is proximal to the user, and that offers a library of utilities that are pertinent to the "context-aware data model," is the novelty of our design.

In addition to building individual utilities for this architecture, our current work is to also explore how to best design the system as a whole. Moving computation from the client to the workspace adds new issues that must be solved. For example, the workspace must be able to correctly save persistent data at the end of a session with the client, which means it must have a mechanism for figuring out which data should be saved, and which server in the world to save it to. There must be a mechanism for the client to discover and be assigned to a workspace server which is capable of handling its applications. Multiple client sessions within the same workspace machine raises issues of security, privacy, and QoS scheduling. We will leverage existing work when possible when exploring these issues.

REFERENCES

[1] W. Broll, I. Lindt, I. Herbst, J. Ohlenburg, A. Braun and R. Wetzel, "Toward next-gen mobile AR games," *IEEE Computer Graphics and Applications*, pp. 40–48, 2008.

[2] A. Cheok, K. Goh, W. Liu, F. Farbiz, S. Fong, S. Teo, Y. Li and X. Yang, "Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing," *Personal and Ubiquitous Computing*, Vol. 8, No. 2 pp. 71–81, 2004.

[3] M. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," *IEEE Internet Computing*, Vol. 13, No. 5 pp. 10–13, 2009.

[4] C. Taylor and J. Pasquale, "Improving VNC Performance," Computing Science Technical Report CS2009-0943, University of California, San Diego, Computer Science and Engineering Department, May 2009, *Submitted for publication.*

[5] "3d Connexion Space Navigator," URL `http://www.3dconnexion.com/`.

[6] "Tight VNC," URL `http://www.tightvnc.com/`.