The User In Experimental Computer Systems Research

Peter Dinda Gokhan Memik Robert Dick
Bin Lin Arindam Mallik Ashish Gupta Samuel Rossoff
Department of Electrical Engineering and Computer Science
Northwestern University

ABSTRACT

Experimental computer systems research typically ignores the end-user, modeling him, if at all, in overly simple ways. We argue that this (1) results in inadequate performance evaluation of the systems, and (2) ignores opportunities. We summarize our experiences with (a) directly evaluating user satisfaction and (b) incorporating user feedback in four different areas of client/server computing, and use our experiences to motivate principles for that domain. We then generalize (a) and (b) as recommendations for incorporating the user into experimental computer systems research.

1. INTRODUCTION

Computer systems research in all of its forms has traditionally focused on the development of services and infrastructure to make it possible to more easily build and scale applications, as well as to enable new kinds of applications. The user has been kept at a considerable remove from the systems software and hardware. We think of the user as interacting with the application, not with the computer system supporting the application. Although the satisfaction that the user garners depends in large part on the decisions made by the system, the system has only a myopic view of the user through the application workload. While the use of utility functions to represent the user has been proposed for over a decade, the reality is that how well these functions operate as a model of the user, and how well they actually represent diverse user sets is largely unknown.

Over the past three years, we have investigated several computer systems problems with careful consideration, and direct analysis of the end-user. These problems are in the context of client/server computing and include resource borrowing in volunteer computing systems, scheduling of desktop replacement virtual machines in utility computing systems, power management in laptop computers, and latency in remote display systems. Based on this wide range of work, we believe that is important to advocate the following.

1. Experimental computer systems researchers should incorporate user studies into the evaluation of their systems. It is true that user studies are challenging, time consuming, often require institutional review board interaction, and generally produce small data sets. However, we have repeatedly found surprising results that would not have been apparent through typical performance evaluation. In particular, user satisfaction with the behavior of a system are extremely difficult to measure by proxy.
2. Experimental computer systems researchers should consider approaches to systems problems that *directly* incorporate feedback from the end-user. The system need not be completely invisible to the end-user, and even tiny amounts of end-user input can lead to very different system designs that produce much improved levels of measured user satisfaction.[1]

Although we believe these two points are applicable to virtually any computer system that involves human users, our experimental work, on which we elaborate in Section 2, has focused on systems problems encountered in client/server systems, such as:

- Heavyweight clients that run applications locally and have intermittent connectivity to the network to retrieve and synchronize files, heavyweight clients on which the user's personal virtual machine(s) can be downloaded, cached, and executed. This is the common use of laptop and desktop computers today.
- Heavyweight clients on which the user's personal virtual machine(s) can be downloaded, cached, and executed. This is the mode of operation suggested by the CMU/Intel Internet Suspend/Resume project [45] and the Stanford Collective project [6].
- Desktop replacement systems in which dumb thin clients interact through VNC [42], Remote Desktop [38], or

[1]Although the experience reports given in this paper focus on direct *explicit* feedback from the user, we do *not* dismiss implicit feedback. What is critical is the concept of using user feedback to customize system behavior on a per-user basis. We believe that explicit feedback is a powerful technique and that learning techniques can reduce the interaction rate. However, if implicit feedback can produce similar results, they are to be preferred. Notice also that explicit feedback techniques provide a yardstick against which implicit techniques can be compared.

similar protocols [24, 2] with multiuser operating systems or virtual machines running in centralized clusters. AJAX-based web applications are somewhat similar.

In systems such as these, as well as in "client-only" systems like PCs running Windows, there exists a tension between performance, resource use, and energy consumption that must be resolved. Resolving this tension in an optimal or at least acceptable way is the job of system-level mechanisms such as scheduling and resource management. The system tries to choose an operating point that optimizes a constrained function of these costs. All functions in interactive systems include some notion of *user satisfaction*.

Current systems rely upon the following assumptions when optimizing the configuration to meet user satisfaction requirements:

- Users are considered to be identical with respect to satisfaction. In other words, the system optimizes for a canonical user, not for specific users.
- User satisfaction can be measured implicitly. The system, in fact, optimizes for proxies of user satisfaction, such as bounded latency and jitter, minimal power consumption, or minimal price. Furthermore, the system chooses how to combine the many possible metrics of user satisfaction.

These assumptions are widely held not only within the domain of our experimental work, but across experimental computer systems in general.

In our work, we have concluded that this model of user satisfaction is simply untenable. We have found that the following set of principles are important in considering the optimization problems in client/server systems. Although many are counterintuitive, we have strong evidence that the following principles are correct.

1. *User variation.* User satisfaction with a given operating point varies dramatically between users. We have demonstrated that user satisfaction with varying degrees of contention or restriction of CPU, disk, and memory resources when using common interactive desktop applications varies considerably (Section 2.1). Similarly, we have found that user satisfaction with differing CPU frequencies varies dramatically (Section 2.2).
2. *User-specified performance.* Users can specify their personal metric for satisfaction as a function of system performance. In many cases, a user is best qualified to resolve the tension among quality metrics in a way that satisfies the user. While users vary in satisfaction, they are not ineffable. We have demonstrated how naive users can control their own CPU schedules in a desktop replacement scenario to trade off price and interactive performance (Section 2.3). Similarly, we have demonstrated an interface for letting naive users control CPU frequency to tradeoff between performance and power (Section 2.4).
3. *User-system interface.* The system software that does optimization should provide simple user interfaces to allow the user to *explicitly* indicate how well the system is trading off the desired quality metrics. Implicit and explicit expression of the user's cost function is also helpful. Sections 2.3 and 2.4 illustrate such interfaces.
4. *Learning.* Explicit user interfaces to the system must, over time, require less and less interaction. They should exploit explicit user interaction to learn user-specific preferences while simultaneously driving on-line tradeoffs among system quality metrics such as performance, power consumption, and correctness. The systems described in Sections 2.3 and 2.4 include simple learning techniques that reduce the amount of explicit user input. We also report briefly on our work evaluating the prospects for speculative remote display systems (Section 2.5), which shows how more costly learning algorithms can be fruitfully employed to predict user actions and responses.

We elaborate on these principles in Section 3, and touch on how they relate to our experimental work in the relevant sections.

Building on the experiences we report, and the principles we distill from them, are for the client/server context, we next argue that using studies to evaluate systems, and direct user input to inform them, can be generalized over experimental computer systems research, and provide advice for experimenters who want to apply these ideas (Section 4). Section 5 concludes the paper.

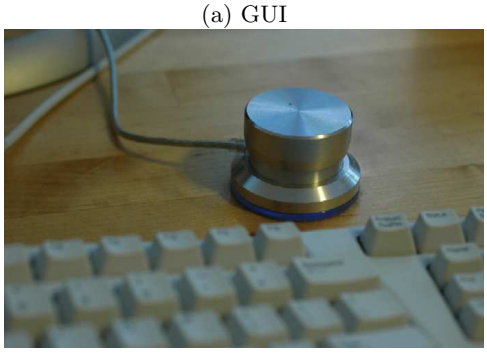
2. EXPERIENCE

We have practiced what we preach, (a) using user studies to evaluate systems, and (b) using direct user input to inform systems. In the following, we summarize the results (and cite the original work for those interested in learning more) from five distinct projects, all emerging from the domain of client/server systems. Our goal is to illustrate that our claims about the efficacy of (a) and (b) hold in diverse areas. The results also give examples of the principles we have specifically distilled for the client/server environment, which we elaborate on further in Section 3.

2.1 Measuring and understanding user comfort with resource borrowing

Many computers are highly under-utilized [39, 10, 1], a fact that many widely-used systems rely on to harvest spare resources for other purposes, a technique we refer to as *resource borrowing.* Examples in scientific computing include Condor [31, 16], Entropia [8], SETI@Home [50], Protein Folding at Home [25], DESChall [9], and the Google Toolbar [17]. Such systems are deployed on hundreds of thousands (SETI@Home) to millions (Google) of computers. The definition of "spare" in these systems is extremely conservative because the foreground user can turn the sharing systems off if they become irritating. For example, the default for both Condor and SETI@Home is to run only when the screen saver is on and there is no other significant load on the machine. The assumption is that resource borrowing

(a) GUI



(b) Discomfort button

Figure 1: User comfort with resource borrowing interface.

systems must place few restrictions on the resources provided to the interactive user when the user is active. But is this true? *How restricted can an interactive user's resources become before causing discomfort?*

To address this question, we conducted the first-ever in-depth study of user comfort with resource borrowing [19, 18]. We provided a qualitative and quantitative analysis of direct measurements of user comfort with controlled CPU, disk, and physical memory contention. In essence, the user is faced with an increasing degree of resource contention until they finally press a "discomfort button" (Figure 1) The carefully controlled study examined 33 users operating word processors, presentation software, web browsers, and games.

Figure 2 gives examples of our quantitative measurements. Our papers go into much more depth. The figures show CDFs for CPU, memory, and disk aggregated over all the tasks in our study (word processing with Microsoft Word, presentation creation with Microsoft Powerpoint, web browsing with Internet Explorer, and game-playing with Quake, a first person shooter game). The horizontal axis is the level of contention for each resource. The vertical axis is the cumulative fraction of users experiencing discomfort. As the level of borrowing increases, interactivity is increasingly likely to be affected. This is the *discomfort region*. Some users do not experience discomfort in the range of levels explored. We refer to this as the *exhausted region*. A run is a controlled buildup of contention for a given user, application, and resource that either ends in the user pressing the button at or after a high contention level is reached. There is also some probability that a user will feel discomfort even when no resource borrowing is occurring. We introduced blank runs in which no contention is applied to measure this effect. This background discomfort is the *noise floor*.

Our study addressed many aspects to user comfort with resource borrowing and their implications. However, the most important result, which can readily be seen in the data in Figure 2, is the high variation. This variation is largely accounted for by two dominant factors: the application and the user. Obviously in a real desktop environment, it is the user who is the independent factor, as it is the user who chooses the application to run. User variation within an application is also very large.

2.2 Measuring and understanding user comfort with lower clock frequencies

Having seen that there is tremendous variation in user tolerance for restrictions on CPU, disk, and memory resources, and its implications for resource borrowing systems, consider now a more prosaic context: power management in laptop computers. On the processors used in these machines, the operating system can change the clock frequency (and corresponding voltage) to trade off power consumption and performance. Similar to resource borrowing systems, most software (such as the Windows DVFS algorithm) makes the assumption that once any load is placed on the CPU (e.g., the user does anything), the frequency should be maximized. *But is this true?*

To understand the variation in user tolerance for differing frequencies (on an IBM ThinkPad T43 running Windows XP), we conducted a small ($n = 8$) randomized user study, comparing four processor frequency strategies including dynamic, static low frequency (1.06 GHz), static medium frequency (1.33 GHz), and static high frequency (1.86 GHz). The dynamic strategy is the default DVFS used in Windows XP. Our target processor has a maximum frequency of 2.13 GHz. We allowed the users to acclimatize to the full-speed performance of the machine and its applications and then had them create a presentation (Powerpoint), watch an animation (Shockwave), and play a game (FIFA Soccer). Users verbally ranked their experiences after each task/strategy pair on a scale of 1 (discomfort) to 10 (very comfortable).

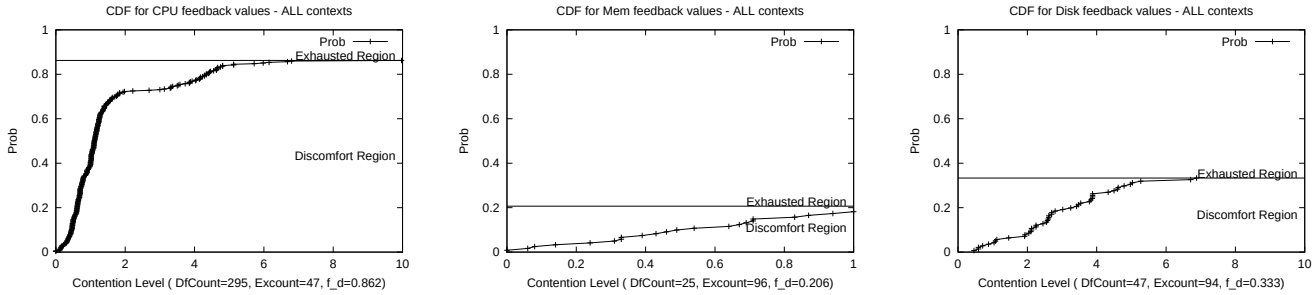
A detailed description of the study and its results are available elsewhere [36, 37, 30], but we summarize the salient points here. Figure 3 illustrates the results of the study in the form of overlapping histograms of the participants' reported comfort level for each of four strategies. Consider Figure 3(a), which shows results for the PowerPoint task. The horizontal axis displays the range of comfort levels allowed in the study and the vertical axis displays the count of the number of times that level was reported. The other graphs are similar.

As one might expect, user comfort with any given strategy (or frequency in the case of the three static strategies) is highly dependent on the application. More importantly, however, is that comfort with given strategy is strongly user-dependent. For any given strategy, there is a considerable spread in the reported comfort levels. Some users will be completely happy with a low frequency even in a demanding application like a game, while others will be displeased with anything less than the highest frequency for even the most undemanding application.

2.3 User-driven scheduling

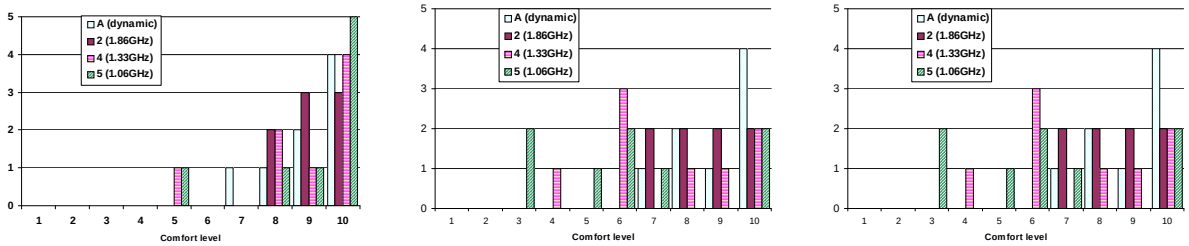
In the Virtuoso[2] project [46, 51], we seek to develop tools and techniques for distributed computing environments based

[2]http://virtuoso.cs.northwestern.edu



(a) CPU (b) Memory (c) Disk

Figure 2: User comfort with resource borrowing: CDF of discomfort with resource contention. Windows XP desktop.



(a) Presentation (b) Animation (c) Game

Figure 3: User comfort with lower clock frequencies: typical Windows laptop.

on the use of virtualization. One application of Virtuoso is a desktop replacement scenario in which a thin client interacts with a virtual machine (VM) running on a remote server. A natural question is how should the user's VM be scheduled on the server? Clearly we seek to keep the interactive user happy while minimizing his utilization of the server to leave room for more VMs.

We have demonstrated that it is feasible to make effective use of direct user interaction in the scheduling process; the user can guide the scheduling process to a solution that balances user comfort and the resource use. Because the resources needed to keep a user happy are highly dependent on the application and the user (Section 2.1), we believe user-specific adaptation is essential.

We have designed, implemented, and evaluated two schemes for incorporating direct user interaction in scheduling virtual machines (VMs) within the Virtuoso system. The first extends the "discomfort button" feedback mechanism of the user comfort study, and is discussed elsewhere [26].

Our second scheme [28, 29] uses the periodic real-time model. We have developed a user-level scheduling tool [27] that does earliest deadline first (EDF) scheduling of periodic tasks [32, 33], letting us run a VM for a given slice within a period of time. We make the period and slice directly configurable by the user through a straightforward human interface: a precision non-centering joystick. The software interface shows the user the current efficiency of their VM (% of allocated cycles actually being used) and the price (linear function of the utilization (slice/period)). Figure 4 illustrates the interface.

(a) GUI (b) Joystick

Figure 4: User-driven virtual machine scheduling.

We ran a comprehensive user study, described in detail in the cited papers, in which the 18 participants used the interface with the goal of finding a comfortable setting of lowest cost while they used a range of Windows applications. We found that almost all users felt that they were able to find a comfortable setting, as well as a comfortable setting that they believed was of lowest cost.

We have studied the statistics of the costs of the schedule that the users chose, and the amount of time they required to find an appropriate schedule. Our first principle, that there is a wide variation in user satisfaction with a given operating point, is reflected in a wide variation in the costs of the schedules that users chose. The majority of users were able to find a setting that makes them comfortable, but there wasn't just one setting. The time for a user to find a reasonable schedule is, on average, quite small (< 1 minute),

Task	Sub-task	Question	Yes	No	NA	Yes/Total	95% CT
Acclim.	Adaptation I	Do you feel you are familiar with the performance of this computer?	18	0	0	**1**	(1,1)
		Are you comfortable with these applications?	17	1	0	**0.94**	(0.84, 1.05)
	Adaptation II	Do you feel that you understand the control mechanism?	18	0	0	**1.00**	(1,1)
		Do you feel that you can use the control mechanism?	18	0	0	**1.00**	(1,1)
Word	I Comfort	Did you find that the joystick control was understandable in this task?	17	1	0	**0.94**	(0.84, 1.05)
		Were you able to find a setting that was comfortable?	18	0	0	**1.00**	(1,1)
	II Comfort+Cost	Did you find that the joystick control was understandable in this task?	17	1	0	**0.94**	(0.84, 1.05)
		Were you able to find a setting that was comfortable?	18	0	0	**1.00**	(1,1)
	III Comfort+Cost+Ext	Did you find that the joystick control was understandable in this task?	18	0	0	**1.00**	(1,1)
		Were you able to find a setting that was comfortable?	18	0	0	**1.00**	(1,1)
Powerpoint	I Comfort	Did you find that the joystick control was understandable in this task?	16	2	0	**0.89**	(0.74, 1.03)
		Were you able to find a setting that was comfortable?	18	0	0	**1.00**	(1,1)
	II Comfort+Cost	Did you find that the joystick control was understandable in this task?	17	1	0	**0.94**	(0.84, 1.05)
		Were you able to find a setting that was comfortable?	17	1	0	**0.94**	(0.84, 1.05)
	III Comfort+Cost+Ext	Did you find that the joystick control was understandable in this task?	16	1	0	**0.89**	(0.74, 1.03)
		Were you able to find a setting that was comfortable?	17	1	0	**0.94**	(0.70, 1.08)
Web	I Comfort	Did you find that the joystick control was understandable in this task?	16	2	0	**0.89**	(0.74, 1.03)
		Were you able to find a setting that was comfortable?	13	4	1	**0.72**	(0.52, 0.93)
	II Comfort+Cost	Did you find that the joystick control was understandable in this task?	17	1	0	**0.94**	(0.84, 1.05)
		Were you able to find a setting that was comfortable?	16	2	0	**0.89**	(0.74, 1.03)
	III Comfort+Cost+Ext	Did you find that the joystick control was understandable in this task?	17	1	0	**0.94**	(0.84, 1.05)
		Were you able to find a setting that was comfortable?	16	1	1	**0.89**	(0.74, 1.03)
Game	I Comfort	Did you find that the joystick control was understandable in this task?	18	0	0	**1.00**	(1, 1)
		Were you able to find a setting that was comfortable?	16	2	0	**0.89**	(0.74, 1.03)
	II Comfort+Cost	Did you find that the joystick control was understandable in this task?	17	1	0	**0.94**	(0.84, 1.05)
		Were you able to find a setting that was comfortable?	14	3	1	**0.78**	(0.59, 0.97)
	III Comfort+Cost+Ext	Did you find that the joystick control was understandable in this task?	17	1	0	**0.94**	(0.84, 1.05)
		Were you able to find a setting that was comfortable?	16	2	0	**0.89**	(0.74, 1.03)

Figure 5: Summary of user responses in study of user-driven scheduling of interactive virtual machines.

has little variation, and is likely to decline even further as a user becomes more familiar with the system. *Our system is able to use a small amount of direct human interaction to help a diverse range of users find a satisfactory schedule.* As far as we are aware, this was the first ever demonstration of the principle of using direct human interaction to inform and guide a low-level scheduling process.

2.4 User-driven frequency scaling

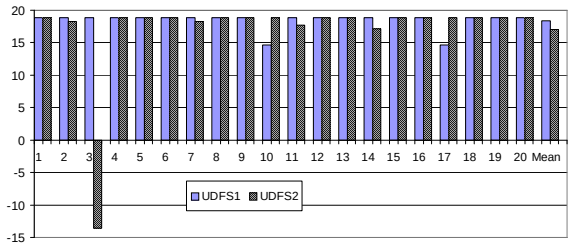
Dynamic Voltage and Frequency Scaling (DVFS) is a widely used technique for controlling power and energy use in modern processors. These processors allow the dynamic selection of clock frequency and voltage. Reducing clock frequency reduces power in linear proportion, while decreasing voltage decreases it in quadratic proportion. Further, the lowest acceptable voltage at which the processor can run depends on the clock frequency with higher frequencies requiring higher voltages. DVFS techniques typically use event-driven algorithms to set clock frequency, and then set voltage based on the chosen frequency setting.

Existing DVFS techniques ignore the user, assuming that CPU utilization or the OS events prompting it are sufficient proxies. A high CPU utilization leads to a high frequency and high voltage, regardless of the user's satisfaction or expectation of performance. To remedy this limitation, we investigated *User-Driven Frequency Scaling (UDFS)* that uses direct user feedback to drive an online control algorithm that determines the processor frequency. UDFS automatically adapts OS power management to user preferences.

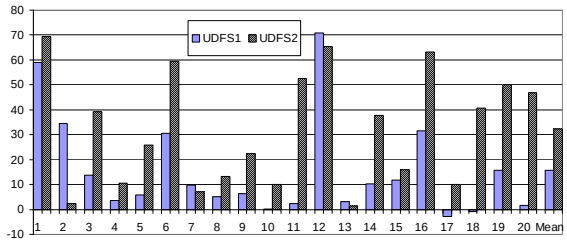
Processor frequency has strong effects on power consumption and temperature, both directly and also indirectly through the need for higher voltages at higher frequencies. The choice of frequency is directly visible to the end-user as it determines the resulting performance. There is considerable variation among users with respect to the satisfactory performance level for a given workload mix, as we have illustrated in Section 2.1, and for a given workload mix and clock frequency combination, as illustrated in Section 2.2. We exploit these variations to customize frequency control policies dynamically to individual users.

To investigate the feasibility of UDFS, we have developed two schemes to control the frequency of the CPU by considering direct user feedback. Both of these schemes try to find the ideal operating point by reducing voltage until user feedback is provided (the interface to perform the user feedback is similar to that of Section 2.1). UDFS1 is an adaptive algorithm that can be viewed as an extension/variant of the TCP congestion control algorithm [48, 53, 4, 14]. UDFS2, on the other hand, tries to find the lowest frequency at which the user feels comfortable and then stabilize there. For each frequency level possible in the processor, we assign an interval t_i, the time for the algorithm to stay at that level. If no user feedback is received during the interval, the algorithm reduces the frequency by one level. If a user is irritated, the interval for the corresponding control level is increased.

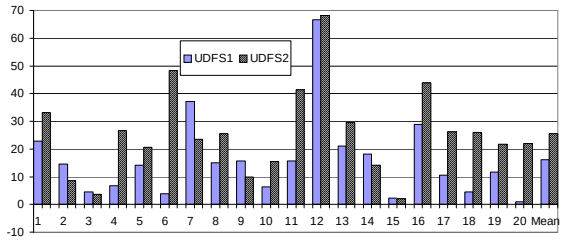
To investigate the impact of UDFS schemes, we performed a study with 20 users. The user study took around 45 minutes for each user, during which time each user operated a laptop running three different applications using the native Windows DVFS, UDFS1, and UDFS2. The laptop was connected to a National Instruments 6034E data acquisition board attached to the PCI bus of a host workstation running Linux, enabling power measurement. For the three applications studied (Shockwave, FIFA, and Powerpoint), the



(a) PowerPoint



(b) 3D Shockwave



(c) FIFA Game

Figure 6: UDFS: Percentage power improvement over Windows DVFS. The horizontal axes indicate individual users, and the mean. The vertical axes are percentage improvements.

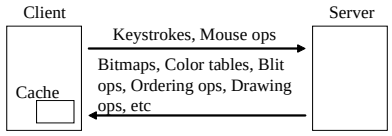
power consumption of the system can be reduced by 22.1%, averaged across all users.

In addition to this analysis, we have measured the static power consumption of the CPU by monitoring the frequency level. The static power consumption results are summarized in Figure 6, which presents both individual user results and average results for UDFS1 and UDFS2 for three different applications. The vertical axis show the percentage improvement for power over the Windows native DVFS scheme. On average, the power consumption can be reduced by 24.9% over existing DVFS scheme for all three applications using the UDFS2 algorithm.

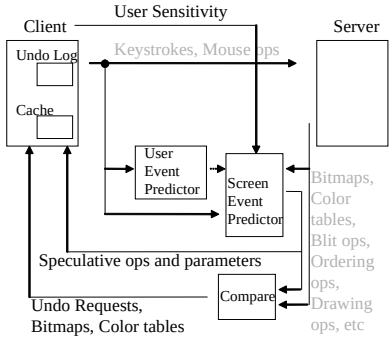
Similar to the results for user-driven scheduling, the results for user-driven frequency scaling illustrate the utility of having even a small amount of user feedback within systems software.

2.5 Prospects for speculative remote display

Remote display systems such as VNC [42], Windows RDP [38, 43], and others allow the interactive, graphical use of a remote computer or virtual machine. At its simplest, these systems can be thought of abstractly as a framebuffer on the



(a) RDP



(b) Speculative RDP

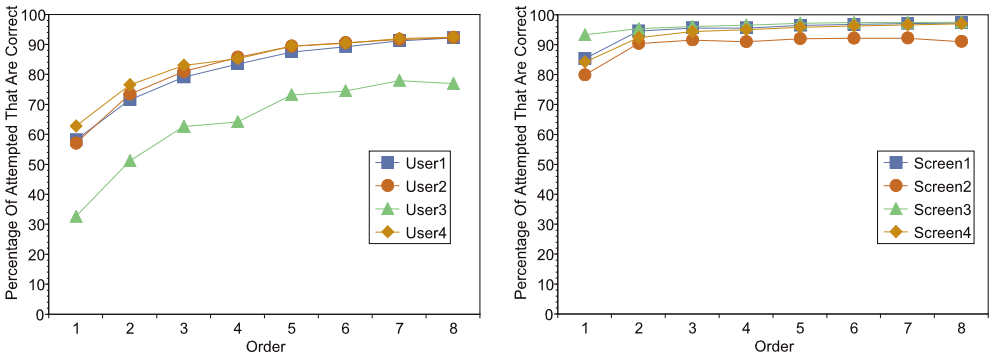
Figure 7: RDP and its proposed speculative variant.

client on which the server draws. User events (keystrokes, mouse movements, etc.) flow from client to server, and screen events (commands to update the framebuffer) flow from server to client. Unfortunately, current remote display systems, even current research systems like THINC [2], suffer when the network latency is high and/or variable [24].

We are exploring extending remote display systems in line with the client/server principles given in the introduction. In our system, the client *predicts* screen events based on the past user and screen events. If the predicted screen events refer to locally cached constructs, the client *speculatively executes* them, undo-logging them as it goes. Most user events, especially those corresponding to fine-grain interactions like typing and menu operation, have their responses computed locally, avoiding the round-trip time. The speculatively executed screen events are compared with those being returned by the server. A difference results in the undo log being used to restore the display to the point preceding the incorrectly executed event, at which point the actual event is executed. The extent of such repair operations and their frequency depends on how aggressive the predictor is, which will be set by the user. Figure 7(a) shows the current structure of an RDP client and server in simplified form, while Figure 7(b) illustrates the structure of a speculative variant of RDP.

Although we have not yet completed a speculative remote display prototype, *we have demonstrated the excellent prospects of the concept.* We instrumented the open-source rdesktop [7] client for the the RDP protocol to allow us to capture traces of user events and screen events (including drawing commands like bit blit, text drawing, line drawing, etc) that pass between client and server. We then ran a study in which four randomly selected participants in our resource borrowing study (Section 2.1) repeated their tasks. This gave us trace data for word processing, presentation creation, web browsing, and game playing.

We studied the prediction of the user and screen event streams



(a) User Events (b) Screen Events

Figure 8: Performance of simple, 1000 state, k-th order Markov model in RDP.

using simple, state-limited, k-th order Markov models, configured simply to predict the next event. The modeling was extremely simple, treating both the type of an event and its parameters collectively as a string. Also, our preliminary work only examined predicting the next event. Nonetheless, we were extremely surprised to find that we were able to predict extremely well, as shown in Figure 8. Here, we are showing the percentage of correct predictions as a function of k, given the current event has been seen at least once before. More detailed results are available elsewhere [44].

3. PRINCIPLES FOR THE CLIENT/SERVER CONTEXT

We now elaborate on the principles for optimization problems within the client/server context that we summarized in the introduction, building on the experiences reported in Section 2, and on the results and experiences of others.

User variation: There is considerable variation in user satisfaction with any given operating point. This principle is directly supported by the evidence in Sections 2.1, and 2.2. It is also born out by work studying the user annoyance with interfaces [21, 41] and latency tolerance [22, 12] within the human computer interaction community. Other results include limited user-level customization of GUIs [35, 11]. The systems community has used latency to evaluate operating systems [13], and developed initial models for interactive user workload [3].

User-specified performance: The user can and should inform the systems software of his satisfaction with the current delivered performance, which results from the current operating point. This principle follows from the first principle. By adapting to the individual user, systems software can choose an operating point that increases user satisfaction and most efficiently uses available resources. We first explained this concept in an HPDC 2004 paper [19]. In Sections 2.3 and 2.4 we illustrated two specific systems that directly incorporate user-specified performance or satisfaction. The concept has also begun to see some interest within the adaptive systems community [47], and recent power management work has sought to support per-user information to some extent. In the power management community, measured response times are typically used as a proxy for the user [34, 54] and in modern systems like Vertigo [15] these measurements can be inferred from unmodified applications and optimized in the context of a per-user profile.

Our notion of user-specified performance differs from that proposed in prior work in two fundamental ways. First, we interact directly with users at run-time to measure their satisfaction with performance. Second, we do not decouple user perceivable measurements from satisfaction. One can argue that a multi-step process exists: operating point → OS-level performance metrics (e.g., message timings) → user-level performance metrics (e.g., latency) → user satisfaction. Our principle is simpler: operating point → user satisfaction. As a consequence, it eliminates numerous sources of error from the control–feedback loop connecting user and operating point control system.

User-system interface: The interface through which the user interacts with the systems software must be simple, elegant, and understandable even for naive users. Because the user is to be involved in the online systems-level decision-making that determines performance and correctness, it is essential that the interface be simple, elegant, and usable by naive users. Note that we are asking the user to provide input to the lowest level operating system services. Sections 2.3 and 2.4 gave specific examples of such interfaces, as well as their evaluation. While the topic of user interface design and evaluation is a deep and complex one, it is important to note that any effective user interface for systems software will have to be very simple, which reduces the design space considerably, and makes evaluation easier.

Learning: The interface through which the user interacts with the systems software must learn user actions and preferences so that interactions become rarer over time. The importance of learning, the final principle, follows also from the interface requirements. Even an elegant interface that is understandable to naive users would be intrusive if the user had to interact with it frequently. We believe it is necessary to develop and apply machine learning techniques to, over time, learn the individual user's operating point → satisfaction characteristics. Sections 2.3 and 2.4 illustrated systems that use very simple learning techniques to reduce interaction rates. Section 2.5 illustrated the use of more complex and expensive techniques to predict user-visible system events. Learning techniques and interface designs interact in complex ways. We are only at the first stages at determining

just how parsimoniously we can use the user's attention.

4. GENERALIZATION AND ADVICE

Although the experiences we reported (Section 2) and the principles we drew from them (Section 3) are specific to the optimization problems in the client/server environment, we believe that it is possible to generalize from them to experimental computer systems research overall. In particular, to reiterate the second paragraph of the introduction, we advocate that researchers should (1) incorporate user studies into the evaluation of their systems, and (2) consider approaches to systems problems that leverage direct input from the user. In the following, we first elaborate on these two themes, and then offer advice to those who would like to operationalize them in their own work.

4.1 Generalizing

In our experience reports, we focused on user satisfaction in the client/server context. Of course, user satisfaction, and the idea of explicitly eliciting it as feedback from the end-user, can be broadly applied. Even for systems that operate outside the timescale of human attention, we can collect trace information, compute metrics on it, and elicit user satisfaction with those metrics.

A key finding that we are sure resonates well beyond the client/server context is that of considerable variation among users in their satisfaction with any given configuration choice. If there is any single take-away message, it is that systems researchers need to consider the *individual* user.

For those systems that can be modeled as, or include a control system element, the individual user can be thought of in at least three ways. First, the user can provide the "set point" for the system. Second, the user can provide the "error signal" of the system. The latter is essentially the approach we have taken in the reported work on power management (Section 2.4). Finally, the user can be a part of, or the whole of the control mechanism itself, determining not only when the "error" is too large, but also determining the configuration that will reduce it. It is this model that we used in the reported work in scheduling VMs (Section 2.3).

Regardless of how or even whether explicit or implicit user feedback is incorporated into systems, it is clear that as systems increasingly face users, it is vital that they be evaluated, at least in part, through user studies.

4.2 Advice

Evaluating a system via a user study is a different challenge from evaluating it using a synthetic or trace workload and straightforwardly visible metrics. Having done a number of such studies so far, we can offer the following advice:

- You should engage an expert in psychological studies or human computer interaction. Effective user studies require extremely careful controls and structure because human subjects approach an experiment at many different cognitive levels, and independent goals. At minimum, refer to the literature (for example, [40, 5, 23]). The advice of Don Norman, Benjamin Watson, and Bruce Gooch was very important to our work.

- Institutional review boards (IRBs) may have to be engaged depending on the nature of the user study. When explaining our work to colleagues, we often have heard expressed the fear that IRB involvement could become a colossal time sink. While this is possible, it is important to note a few things. First, basic IRB certification is relatively standardized and quite easy to acquire. Second, user studies in the systems context are generally classified as social science-based studies, which require considerably less paperwork than physical, interventionist studies, which is where most of the horror stories lie. Third, the paperwork required for IRB review, although tedious, tends to be quite reusable. Finally, in many cases, because a user study in the systems context is patently unlikely to lead to psychological damage, full review is unnecessary.

- User studies are invariably much smaller than the kinds of evaluations that we, as systems researchers, are familiar with. This limits the range of what can be studied, and it requires that small sample size, robust statistics [20], or full data reporting, are necessary. In our experience up to now, the effects that are measured have been quite large, which makes it possible to draw strong conclusions despite the small sample size.

- Although it is nearly impossible to engage a random subject population, there are techniques, like subsampling, that can be used to estimate selection biases.

- It is vital, especially when measuring user satisfaction, to differentiate between the actual effect and the background effect. The measured satisfaction (through surveying, level eliciting, etc) integrates satisfaction with your system with the user's general satisfaction. The equivalent of a placebo is needed to differentiate the two.

- It is vital that a user study be double-blinded to the greatest extent possible. Users can inadvertently produce overly optimistic or pessimistic behaviors if they can infer the "desired" outcome of the test.

- Ideally, we want to correlate systems-level quantities that can be easily measured with user study results in order to validate the latter. However, this is often difficult, if not impossible, due to the considerable variation in user responses. For example, as we have seen, user satisfaction with particular level of resources tends to have tremendous variation (Section 2.1). Even when we do not have a system-level quantity to measure, we can use the technique of deception [49] to convince the user that we do. For example, in the results of Section 2.3, we video-taped users in some tests, and claimed that a fictitious psychology collaborator would analyze the video tape to produce an independent assessment of user satisfaction. Comparing results in which we deceive the user into this belief, with those where we do not, helps us discount the possibility that the user is being uncooperative.

- Eliminate all user-visible extraneous information during any user study. While it is tempting for us to build interfaces that provide as much detail as the user

wants, this can dramatically skew results. For example, in the preparatory work for the study described in Section 2.1, we noticed users on one of our two identical test machines were generally more irritated by disk bandwidth borrowing. It turned out that this machine had a visible hard disk access light, while the other did not.

Our second claim is that systems researchers should consider using direct user input in their systems. We have the following advice regarding the interfaces for doing so.

- We have generally found that "out-of-band" input devices tend to work best. By out-of-band, we mean that we add input hardware or use existing input hardware that is not used for any other purpose. The interface to the system software will (ideally) be infrequently used. If we layer it on top of an existing input device, the user has to essentially perform a "cognitive context switch" to use it, unlearning the normal purpose of the interface.

- It is tempting to ask for a lot of input (in terms of frequency or dimensionality or both) from the user, but this should be avoided. Use as little input as possible, and evaluate the tradeoff between the amount of input and its utility very carefully.

- Similarly, the output portion of the interface should be as thin as possible. In some cases, it can be nonexistent as it is the performance of the system that serves as implicit output.

- It is important to understand, and account for, the fact that explicit user input can itself be a source of user dissatisfaction. When measuring the efficacy of a system based on explicit user feedback, the experimenter must have an independent gauge of this source.

- For some systems and some users, it may be possible to eliminate all user input through the use of implicit measures of the user, for example by recognizing patterns in system-level events [52]. Note that explicit feedback systems can be used as a yardstick in evaluating implicit ones. Furthermore, these are not either/or propositions. There is a spectrum that ranges from explicit feedback, to explicit feedback with learning, to implicit feedback. The latter can fall back on the former.

5. CONCLUSION

We have advocated that experimental computer systems researchers should (a) incorporate user studies into the evaluation of our systems, and (b) consider approaches to systems problems that draw on feedback or other input from the end-user. Through our experiences in applying these ideas in five different systems projects in client/server computing, we illustrated how the ideas can help us find and exploit new, and often surprising, opportunities and principles. Two particular principles we derived is that there is considerable variation in user satisfaction with any given operating point, and that this variation can be exploited through interfaces that use direct user feedback about satisfaction. We then generalized our results and offered advice to practitioners who want to apply (a) and (b) to their own systems and domains.

6. REFERENCES

[1] Anderson, T. E., Culler, D. E., and Patterson, D. A. A case for networks of workstations. *IEEE Micro* (February 1995).

[2] Barratto, R., Kim, L., and Nieh, J. Thinc: A virtual display architecture for thin-client computing. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP)* (October 2005).

[3] Bhola, S., and Ahamad, M. Workload modeling for highly interactive applications. In *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (1999), pp. 210–211. Extended version as Technical Report GIT-CC-99-2, College of Computing, Georgia Tech.

[4] Brakmo, L. S., O'Malley, S. W., and Peterson, L. L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications* (1994), pp. 24–35.

[5] Card, S., Moran, T., and Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Publishers, 1986.

[6] Chandra, R., Zeldovich, N., Sapuntzakis, C., and Lam, M. The collective: A cache-based system management architecture. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)* (May 2005).

[7] Chapman, M. rdesktop: A remote desktop protocol client for accessing windows nt terminal server. http://www.rdesktop.org.

[8] Chien, A. A., Calder, B., Elbert, S., and Bhatia, K. Entropia: architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing 63*, 5 (2003), 597–610.

[9] Curtin, M., and Dolske, J. A brute force search of DES keyspace. *;login:* (May 1998).

[10] Dinda, P. A. The statistical properties of host load. *Scientific Programming 7*, 3,4 (1999). A version of this paper is also available as CMU Technical Report CMU-CS-TR-98-175. A much earlier version appears in LCR '98 and as CMU-CS-TR-98-143.

[11] Dourish, P. Evolution in the adoption and use of collaborative technologies.

[12] Embley, D. W., and Nagy, G. Behavioral aspects of text editors. *ACM Computing Surveys 13*, 1 (January 1981), 33–70.

[13] Endo, Y., Wang, Z., Chen, J. B., and Seltzer, M. Using latency to evaluate interactive system performance. In *Proceedings of the 1996 Symposium on Operating Systems Design and Implementation* (1996).

[14] Fall, K., and Floyd, S. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *SIGCOMM Computer Communication Review 26*, 3 (1996), 5–21.

[15] Flautner, K., and Mudge, T. Vertigo: Automatic Performance-setting for Linux. *SIGOPS Oper. Syst. Rev. 36*, SI (2002), 105–116.

http://doi.acm.org/10.1145/844128.844139.

[16] Frey, J., Tannenbaum, T., Foster, I., Livny, M., and Tuecke, S. Condor-g: A computation management agent for multi-institutional grids. In *Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC 2001)* (2001), pp. 55–66.

[17] Google Corporation. Google compute. http://toolbar.google.com/dc/.

[18] Gupta, A., Lin, B., and Dinda, P. A framework and toolkit for understanding user comfort with resource borrowing. Tech. Rep. NWU-CS-04-28, Department of Computer Science, Northwestern University, February 2004.

[19] Gupta, A., Lin, B., and Dinda, P. A. Measuring and understanding user comfort with resource borrowing. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC 2004)* (June 2004).

[20] Huber, P. *Robust Statistics*. Wiley and Sons, 2003.

[21] Klein, J. T. Computer response to user frustration. Master's thesis, Massachusetts Institute of Technology, 1999.

[22] Komatsubara, A. Psychological upper and lower limits of system response time and user's preferance on skill level. In *Proceedings of the 7th International Conference on Human Computer Interaction (HCI International 97)* (August 1997), G. Salvendy, M. J. Smith, and R. J. Koubek, Eds., vol. 1, IEE, pp. 829–832.

[23] Kuniavsky, M. *Observing the User Experience: A Practioner's Guide to User Research*. Morgan Kaufmann, 2003.

[24] Lai, A., and Nieh, J. Limits of wide-area thin-client computing. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (2002).

[25] Larson, S. M., Snow, C. D., Shirts, M., and Pande, V. S. Folding@home and genome@home: Using distributed computing to tackle previously intractable problems in computational biology. In *Computational Genomics*, R. Grant, Ed. Horizon Press, 2002.

[26] Lin, B., and Dinda, P. User-driven scheduling of interactive virtual machines. In *Proceedings of the Fifth International Workshop on Grid Computing* (November 2004).

[27] Lin, B., and Dinda, P. Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. In *Proceedings of ACM/IEEE SC (Supercomputing)* (November 2005).

[28] Lin, B., and Dinda, P. Putting the user in direct control of cpu scheduling. Tech. Rep. NWU-EECS-06-07, Department of Electrical Engineering and Computer Science, Northwestern University, July 2006.

[29] Lin, B., and Dinda, P. Towards scheduling virtual machines based on direct user input. In *Proceedings of the 1st International Workshop on Virtualization Technology in Distributed Computing (VTDC)* (2006).

[30] Lin, B., Mallik, A., Dinda, P., Memik, G., and Dick, R. Power reduction through measurement and modeling of users and cpus: Summary. In *Proceedings of the 2007 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (2007). To Appear.

[31] Litzkow, M., Livny, M., and Mutka, M. W. Condor — a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems (ICDCS '88)* (June 1988), pp. 104–111.

[32] Liu, C. L., and Layland, J. W. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM 20*, 1 (January 1973), 46–61.

[33] Liu, J. *Real-time Systems*. Prentice Hall, 2000.

[34] Lorch, J. R., and Smith, A. J. Using User Interface Event Information in Dynamic Voltage Scaling Algorithms. In *Technical Report UCB/CSD-02-1190, Computer Science Division, EECS, University of California at Berkeley, August* (2002). citeseer.ist.psu.edu/lorch03using.html.

[35] MacLean, A., Carter, K., Lovstrand, L., and Moran, T. User-tailorable systems: pressing the issues with buttons. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1990), ACM Press, pp. 175–182.

[36] Mallik, A., Lin, B., Dinda, P., Memik, G., and Dick, R. Process and user driven dynamic voltage and frequency scaling. Tech. Rep. NWU-EECS-06-11, Department of Electrical Engineering and Computer Science, Northwestern University, August 2006.

[37] Mallik, A., Lin, B., Memik, G., Dinda, P., and Dick, R. User-driven frequency scaling. *IEEE Computer Architecture Letters 5*, 2 (2006).

[38] Microsoft. Remote desktop protocol (rdp) features and performance. Tech. rep., 2000.

[39] Mutka, M. W., and Livny, M. The available capacity of a privately owned workstation environment. *Performance Evaluation 12*, 4 (July 1991), 269–284.

[40] Proctor, R., and Van Zandt, T. *Human Factors in Simple and Complex Systems*. Allyn and Bacon, 1993.

[41] Reynolds, C. J. The sensing and measurement of frustration with computers. Master's thesis, Massachusetts Institute of Technology Media Laboratory, 2001. http://www.media.mit.edu/~carsonr/pdf/sm_thesis.pdf.

[42] Richardson, T., Stafford-Fraser, Q., Wood, K., and Hopper, A. Virtual network computing. *IEEE Internet Computing 2*, 1 (January/February 1998).

[43] Romano, P. Itu-t recommendation t.128 (application sharing). Tech. rep., ITU, March 1997.

[44] Rossoff, S., and Dinda, P. Prospects for speculative remote display. Tech. Rep. NWU-EECS-06-08, Department of Electrical Engineering and Computer Science, Northwestern University, August 2006.

[45] Satyanarayanan, M., Kozuch, M., Helfrich, C., and O'Hallaron, D. Towards seamless mobility on pervasive hardware. *Pervasive and Mobile Computing 1*, 2 (June 2005), 157–189.

[46] Shoykhet, A., Lange, J., and Dinda, P. Virtuoso: A system for virtual machine marketplaces. Tech. Rep. NWU-CS-04-39, Department of Computer Science, Northwestern University, July 2004.

[47] Sousa, J., Balan, R., Poladian, V., Garlan, D., and Satyanarayanan, M. Giving users the steering wheel for guiding resource-adaptive systems. Tech. Rep. CMU-CS-05-198, Department of Computer Science, Carnegie Mellon University, December 2005.

[48] Stevens, W. TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms. In *Internet RFC 2001* (1997).

[49] Stricker, L. J. The true deceiver. *Psychological Bulletin*, 68 (1967), 13–20.

[50] Sullivan, W. T., Werthimer, D., Bowyer, S., Cobb, J., Gedye, D., and Anderson, D. A new major seti project based on project serendip data and 100,000 personal computers. In *Proceedings of the Fifth International Conference on Bioastronomy* (1997), C. Cosmovici, S. Bowyer, and D. Werthimer, Eds., no. 161 in IAU Colloquim, Editrice Compositori, Bologna, Italy.

[51] Sundararaj, A., Gupta, A., and Dinda, P. Increasing application performance in virtual environments through run-time inference and adaptation. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC)* (July 2005).

[52] Theocharous, G., Mannor, S., Shah, N., Gandhi, P., Kveton, B., Siddiqi, S., and Yu, C.-H. Machine learning for adaptive power management. *Intel Technology Journal 10*, 4 (Nov. 2006).

[53] Wang, Z., and Crowcroft, J. Eliminating Periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm. In *ACM Computer Communications Review* (1992).

[54] Yan, L., Zhong, L., and Jha, N. K. User-perceived Latency based Dynamic Voltage Scaling for Interactive Applications. In *Proceedings of ACM/IEEE Design Automation Conference* (2005).