Towards Scheduling Virtual Machines Based On Direct User Input

Bin Lin Peter A. Dinda
Department of Electrical Engineering and Computer Science, Northwestern University
{binlin,pdinda}@cs.northwestern.edu

Abstract

We propose a new approach to scheduling virtual machines (VMs) on a provider CPU that is unique in that is based around the use of direct user input. *In our system, a user's VM is scheduled as a periodic real-time task. The user can instantaneously manipulate his VM's schedule using a joystick. An on-screen display illustrates the current schedule's cost and indicates when the user's desired schedule is impossible due to the schedules of other VMs or resource constraints. We report on a user study of our prototype system that reveals that even a naive user is capable of using the interface to our system to find a schedule that balances cost and the comfort of his VM. Good schedules are user- and application-dependent to a large extent, illustrating the benefits of user involvement.*

1 Introduction

Building on work that demonstrated the high variability in user tolerance for performance in interactive applications [13], we focus here on scheduling virtual machines (VMs) that support interactive desktop users. We propose, implement, and evaluate a unique new approach to CPU scheduling that is based on *direct user input*, even from users who know nothing about scheduling and don't want to learn. As far as we are aware, there is currently no scheduling approach that incorporates *explicit* input supplied *directly* by even naive users.

Our work takes place in the context of Virtuoso, a system for utility computing that is based on VMs interconnected with overlay networks [38, 34, 37, 19, 14, 39]. Each provider computer in the system can support multiple VMs, each of which can run a different operating system. A typical interactive user uses a VM loaded with an operating system such as Microsoft Windows and communicates with it using a thin client [33, 18] running a remote display system [31, 32]. Each user will see a slowed machine due to resources committed to other users.

All the VMs on a provider computer are scheduled as periodic real-time tasks. We summarize the design of our real-time scheduling system here, but our focus is on an interface to the system that is intended to allow even naive users to exploit it. The user of a VM can continuously adjust its schedule using an interface akin to a two-dimensional throttle (a joystick), up to the resource limits and the constraints of the other VMs. As he adjusts the schedule, an on-screen display shows the cost of the current schedule.

Through an extensive user study, the results of which we summarize here, we find that this interface lets all users cleanly trade off between comfort and cost, appropriately customizing their schedules without knowing anything about CPU scheduling. In other words, our interface provides an effective means for users to express their diverse CPU needs and have the system meet them.

We strongly believe that the overall approach of incorporating direct user input into resource scheduling will extend to other resources, combinations of resources, distributed systems, and autonomic computing. We conclude with a discussion of the issues that arise in this broader context.

2 Related work

Systems researchers have proposed a wide range of scheduling approaches to attempt to automatically optimize both for responsiveness and utilization. Examples include the BSD Unix scheduler [26], lottery scheduling [41], weighted fair queuing and its derivatives [3], BVT [8], SRPT scheduling [2], and periodic [22, 15] and sporadic [23] hard real-time models, as well as soft real-time models for multimedia [5, 29]. In some models, user interaction is included *implicitly* and *indirectly* in scheduling decisions. For example, the Unix scheduler provides a temporary priority boost to processes that have become unblocked. Since a typical interactive process blocks mostly waiting for user input, the boost gives it the effect of responding quickly, even in a system that is experiencing high load.

Direct user input is represented as well. For example, early work [25, 7] used on-screen buttons that encapsulated code to tailor of applications UIs. Weighted fair queuing allows users to explicitly weight each of their processes, controlling the CPU share given to each. Microsoft Windows allows a user to specify the scheduling class of a process. By raising the scheduling class of a process from "Normal" to “Above Normal”, he assures that Windows’ fixed priority scheduler will always run his process in preference to “Normal” processes that are also ready to run. As another example, Unix systems provide the “nice” mechanism to bias Unix’s dynamic priority scheduler. All of the direct user input mechanisms we are aware of, however, require that the user understand the scheduler to get good results. Indeed, with schedulers like the Windows scheduler, it is very easy for a user to live-lock the system if he doesn’t know what he’s doing. Ours is the first scheduling system to incorporate direct user input from even naive users.

Using direct user input in the scheduling process would appear to be in the purview of human-computer interaction research and psychology. However, the work in those areas has concentrated on the impact of latency on user-perceived utility of the system [17, 9], and on user frustration with different user interfaces [16, 30]. Within the systems community, related work has examined the performance of end-user operating systems using latency as opposed to throughput [10], and suggested models for interactive user workload [4]. Recent work has also demonstrated using careful user studies that many programs can be modified by naive programmers to support limited adaptation [1]. However, there are no results on using direct user input from even naive users to guide the scheduler.

Autonomic computing seeks to either automate the management of computer systems or simplify administrator-based management of them. Some work in this area has focused on direct interaction with the administrator, including capturing the effects of operator error [27], exposing debugging of configurations as a search process [42], adjusting cycle stealing so as to control impact on users [36], and using performance feedback to the administrator to help adjust policy [24]. As far as we are aware, however, no work in autonomic computing directly incorporates the end-user.

3 User diversity

Our previous paper [13] described a double-blinded, controlled intervention study of the effects of resource contention on the comfort of the users of typical interactive desktop applications on Microsoft Windows. The resources studied were CPU bandwidth, disk bandwidth, and physical memory. The applications used were identical to those described in Section 6. 38 users, including graduate students, undergraduates, and staff at Northwestern University participated. The users were recruited and selected in a similar manner to that of Section 6, and most were not knowledgeable about resource scheduling. In the study, a user was instructed to carry out a given task (e.g., replicate a drawing using Powerpoint), and to press a “discomfort button” if she felt the computer was operating uncomfortably. While the user carried out the task, we applied randomly selected resource contention profiles for the three resources.

Analyzing the contexts in which users indicated discomfort resulted in a range of qualitative and quantitative conclusions. The observation that users were generally quite intolerant to jitter led to the choice of a periodic real-time scheduling model in Virtuoso, described in the next section. Users were far more sensitive to restrictions or variations in CPU bandwidth than disk or memory.

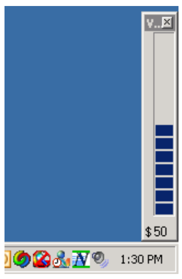
The key conclusion of the study with respect to the present paper is that there is considerable diversity in the tolerance that users have for resource contention. This diversity argues for per-user tailoring of resource usage, and leads to the question we address in this paper: How do we accomplish this per-user tailoring? We now show how to answer this question for the specific case of CPU scheduling.

4 Scheduling VMs in Virtuoso

Virtuoso is middleware for virtual machine-based utility computing that for a user very closely emulates the existing process of buying, configuring, and using an Intel-based computer, a process with which many users and certainly all system administrators are familiar. In Virtuoso, the user visits a web site, much like the web site of Dell or IBM or any other company that sells Intel-based computers. The site allows him to specify the hardware and software configuration of a computer and its performance requirements, and then order one or more of them. The user receives a reference to the virtual machine which he can then use to start, stop, reset, and clone the machine. The system presents the illusion that the virtual machine is right next to the user. The console display is sent back to the user’s machine, the CD-ROM is proxied to the user’s machine’s CD-ROM, and the VM appears to be plugged into the network side-by-side with the user’s machine. The user can then install additional software, including operating systems.

Virtuoso connects users with providers. A provider makes physical computers available on which users’ VMs can be executed. A single physical computer can host multiple VMs. A Virtuoso provider uses VMware GSX Server as its virtual machine monitor (VMM). GSX is a “type II” VMM [11], meaning that it executes as a process on an underlying operating system, Linux in our case. By scheduling the process, we schedule all the activity occurring inside the VM (all the applications running in a VM whose operating system is Microsoft Windows, for example) as a single unit.

Virtuoso uses the periodic real-time model as a unifying abstraction to describe the needs of diverse workloads and then schedule them. In the periodic real-time model, a task is run for $slice$ seconds every $period$ seconds. Using earliest deadline first (EDF) schedulability analysis [22], the sched-



(a) GUI

(b) Non-centering Joystick

Figure 1. Control interface.

uler can determine whether some set of $(period, slice)$ constraints can be met. The scheduler then uses dynamic priority preemptive scheduling with the deadlines of admitted tasks as priorities.

VSched, which is described in detail in a previous paper [20], is a user-level implementation of this approach for Linux that offers soft real-time guarantees. The software is available from virtuoso.cs.northwestern.edu. It runs as a Linux process that schedules other Linux processes. We use it here to schedule the VMs on the host computer. VSched supports $(period, slice)$ constraints ranging from the low hundreds of microseconds (if certain kernel features are available) to days. Using this range, the needs of both highly interactive VMs and long running batch VMs can be described and accommodated.

It is impossible to provide hard real-time support on most variants of the Linux kernel, and VSched does not claim to. Nonetheless, as we report in our earlier paper, its soft real-time behavior is quite good, rarely missing deadlines and only then by very small amounts. We also demonstrate that VSched effectively isolates VMs, and that, despite only controlling CPU, it serves to effectively throttle I/O as an indirect effect. Physical memory isolation is provided by the VMM we use. An important design criterion for VSched is that a VM's constraints can be changed very quickly (in about a millisecond) so that an interactive user can change his VM's performance immediately.

5 User interface

We have developed a graphical interface to allow even a naive user to easily use VSched to set an appropriate $(period, slice)$ constraint for his Windows VM. The tool indicates to the user the cost of his current schedule and allows him to directly manipulate $(period, slice)$. VSched can change the schedule of a VM in about a millisecond, allowing for very smooth control. The holy grail for such an interface is that it be invisible or non-intrusive until the user is unhappy with performance, and then can be nearly instantly manipulated to change the schedule. We have considered a range of different interfaces, but for space reasons we focus here only on the interface shown in Figure 1. The input side of our interface is a non-centering joystick. In such a joystick, the control stalk maintains its current deflection even after the user removes his hand. The horizontal and vertical deflection are mapped into increasing $period$ (left to right) and increasing utilization $(slice/period)$ (bottom to top). Note that all positions of the joystick correspond to valid schedules. Even an inexpensive joystick is sufficient. The one we used costs about $10.

The output side of the interface shows the cost of the current schedule (which can be changed in milliseconds). The specific cost function that is used is

$$cost = 100 \times \left(\frac{slice}{period} + \beta \times \frac{overhead}{slice} \right)$$

Where $overhead$ is the time to execute the scheduling core of VSched once. The purpose here is to capture the fact that as $slice$ declines, more time is spent in VSched and the kernel on behalf of the process. For typical user-selected schedules for interactive VMs, the influence of the overhead is minimal, and the cost is effectively the utilization of the user's VM.

6 User study

We conducted a user study to determine whether end-users could use our interface to find schedules for their interactive VMs that were comfortable, and to determine whether users could trade off between cost and comfort using the interface. This section summarizes our results. More detailed results can be found in a technical report [21].

6.1 Particulars

The 18 users in our study consisted primarily of graduate students and undergraduates from the engineering departments at Northwestern University, and included two participants who had no CS or ECE background. None of the users were familiar with real-time scheduling concepts. We advertised for participants via flyers and email, and vetted respondents to be sure they were at least slightly familiar with the common applications we would have them use. Each user was given $15 for participating.

The test machine was a Dell Optiplex GX270 (2 GHz P4, 512 MB, 80 GB, 17" monitor, 100 mbit Ethernet). The machine ran:

- VMware GSX Server 3.1
- VSched server running as a daemon,
- VM running Windows XP Professional, the applications (Microsoft Word 2002, Powerpoint 2002, Internet Explorer 6.0, Quake II), and our interface, and

- Logitech WingMan Attack 2 Joystick modified to be non-centering, as described earlier.

The VM was run in full-screen mode and the users were not told that they were using virtualization. The smallest slice and period possible were 1 ms, while the largest were 1 s.

6.2 Process

During the study, the user used the Windows VM for four tasks: word processing, presentation creation, web browsing, and game playing. Each task was 15 minutes long with 5 minutes for each of three sub-tasks. We asked users to answer some questions (described later) after each sub-task. We also video-taped users during the study, and the users were told that the video tape and other mechanisms would allow us to determine their degree of comfort during the study independently of the questions we were asking them. This mild use of deception, a widely used technique in psychological research [35], was employed to decrease the likelihood that study participants would lie or operate the system less aggressively than they might outside of the study.

From the user's perspective, the study looked like the following. At the beginning of each task and subtask, the joystick was recentered, corresponding to a 500 ms period with a 50% utilization. The intent was to force the user to manipulate the joystick at least once, since for all of the applications, this schedule was intolerable to us.

1. Adaptation Phase I (8 minutes): The user acclimatized himself to the performance of the Windows VM by using the applications. Questions:
    - Do you feel you are familiar with the performance of this computer? (Y/N)
    - Are you comfortable with these applications? (Y/N)
2. Adaptation Phase II (5 minutes): The user acclimatized himself to the VSched control mechanism, Figures 1(a) and (c). The user listened to MP3-encoded music using Windows Media Player and noticed how the playback behavior changed when he moved the joystick. At the beginning of this stage, the proctor told the user that moving the joystick would change the responsiveness of the computer and that, in general, moving the joystick to the upper-right would make the machine faster, while moving the joystick to the lower left would slow the machine down. However, the proctor also told them that the control was not a simple linear control, that all joystick positions are valid, and that the user should do his best to explore using the joystick control by himself. Questions:
    - Do you feel that you understand the control mechanism? (Y/N)
    - Do you feel that you can use the control mechanism? (Y/N)
3. Word processing using Microsoft Word (15 minutes): Each user typed in a non-technical document with limited formatting.
    - **Sub-task I: Comfort (5 minutes)** The user was told to try to find a joystick setting that he felt was comfortable for him. Questions:
        - Did you find that the joystick control was understandable in this application? (Y/N)
        - Were you able to find a setting that was comfortable? (Y/N)
    - **Sub-task II: Comfort and Cost (5 minutes)** The user was given a cost bar (Figure 1(a)) that showed the current cost of using the Windows VM. When the user moved the joystick, both the responsiveness of the computer and current cost change. The proctor asked the user to do their best to find a comfortable joystick setting that was of the lowest cost. Questions:
        - Did you find that the joystick control was understandable in this application? (Y/N)
        - Were you able to find a setting that was comfortable? (Y/N)
        - If yes, what's the cost?
    - **Sub-task III: Comfort and Cost With Perceived External Observation (5 minutes)** This sub-task was identical to the previous one, except that the proctor told the user that the system had mechanisms by which it could independently determine whether the user was comfortable or not and whether a comfortable setting was of the lowest possible cost. It was claimed that this analysis was achieved through measurement of the efficiency of the VM (the percentage of cycles that the user has allocated that he is actually using), analysis of their mouse, keyboard, and joystick input, and psychological analysis of the video tape. Questions:
        - Did you find that the joystick control was understandable in this application? (Y/N)
        - Were you able to find a setting that was comfortable? (Y/N)
        - If yes, what's the cost?
4. Presentation creation using Microsoft Powerpoint (15 minutes): Each user duplicated a presentation consisting of complex diagrams involving drawing and labeling from a hard copy of a sample presentation.

- The same three sub-tasks as for word processing with the same questions following each sub-task.

5. Browsing and research with Internet Explorer (15 minutes): Each user was assigned a news web site and asked to read the first paragraphs of the main news stories. Based on this, they searched for related material and saved it. This task involved multiple application windows.

    - The same three sub-tasks as for word processing with the same questions following each sub-task.

6. Playing Quake II (15 minutes)

    - The same three sub-tasks as for word processing with the same questions following each sub-task.

As the user performed the tasks, we recorded the following information:

- Periodic measurements of system and user time,
- Periodic measurements of utilization (portion of the allotted time that was spent unblocked), and
- For each joystick event, the time stamp and the new $(period, slice)$ and $cost$.

The user was unaware of the recording process. He saw only the cost of the current schedule.

6.3 Quantitative results

Figure 2 summarizes the responses of users to the questions in our study, providing 95% confidence intervals for the proportion of users who responded in the affirmative. Notice that in all cases, the responses allow us to discount the null hypothesis, that the users are responding randomly, with $> 95\%$ confidence.

The overwhelming majority of users found that they

- understood our control mechanism,
- could use it to find a comfortable position, and
- could use to find a comfortable position that they believed was of lowest cost.

Despite the disparity among the applications and the users, there was little disparity in the users' reactions. There were only two situations where a significant fraction of the users had difficulty finding a comfortable or comfortable and low-cost setting. 28% of users had difficulty finding a comfortable setting for the web browsing task (sub-task I), while 22% had difficulty finding a comfortable, low cost setting for the first person shooter task (sub-task II). In both cases, the numbers result from one of the users answering the question unintelligibly. Furthermore, that user answered "yes" to the more restrictive corresponding question (where we are attempting to deceive him into believing we can answer the question independently). For sub-tasks II and III of each task, we had the user try to find a setting that he was comfortable with and that he believed was of minimal cost. If he felt he had found a comfortable setting, we recorded its cost. Due to the space limit, we can not present the statistics for these costs here. We summarize our findings below.

- As we might expect, costs, on average increase as we look at applications with increasingly finer grain interactivity.
- There is tremendous variation in acceptable cost among the users. The standard deviation is nearly half of the average cost. The maximum cost is as much as five times the minimum cost. This should not be surprising given the wide variation among users found in a previous study of resource borrowing (Section 3).
- Nonetheless, almost all users were able to find a setting that gave them comfortable performance.

We have also evaluated the duration from starting to interact with an application in the system to when the lowest cost is first encountered. Summarizing our results, the median time is 25-60 seconds depending on the application, which includes the use of the application. The time between further interactions will decline significantly as the user will be more familiar with the application/system combination.

The upshot of the study described in this section is that we have identified a *practical* mechanism by which user input can be incorporated into the CPU scheduling process for Virtuoso. Using that input, the system can adapt the schedule of the user's VM to fit the user and the application he is currently running, the side effect of which is that the system can run more interactive users simultaneously, or allocate more time for long-running batch VMs. The user can quickly guide the system to a schedule that simultaneously optimizes both for his comfort in using an application and for low cost.

7 Issues in extension

We have demonstrated the feasibility and utility of direct user feedback as applied to CPU scheduling for interactive VMs, we now describe issues while we are working on the general adaptation problem exposed within the Virtuoso system [39, 38, 14, 40].

Virtuoso provides many adaptation and resource reservation mechanisms to improve the performance of existing, unmodified applications running in communicating VMs.

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 2. Summary of user responses in study.

These include VM migration and overlay topology configuration/forwarding rules [37], lightpath setup in optical networks [19], and local scheduling of VMs [20]. Additionally, Virtuoso can observe the network and host traffic of the VMs to probe the underlying network [14] and the application [12] for their communication topology and other resource demands [38]. Using this information to engage the adaptation and reservation mechanisms to increase an application's performance is a challenging, NP-hard optimization problem [39, 40], one that is often difficult even to pose well. Applying our concept of direct human input to it poses the following challenges.

Frequency of input: We have noted that more frequent user input leads to better performance (lower cost, lower power consumption, high application throughput, among other metrics). However, it is obvious that there must be limits to this frequency. Control algorithms that make use of direct user input must be able to work when the input is infrequent and/or aperiodic. In our view, sensible low-frequency input will take one of three forms: (1) evaluations of a current configuration, resulting in user-specific utility functions; (2) specifications of configurations; and (3) directions for search within a space of configurations. The work described here takes the later two forms.

Interface: Careful user interface design and evaluation are critical to success, especially when targeting naive users. We have generally found that having a very simple, tactile interface separate from the "main" user interface of the application or OS, is preferable because it clearly demarcates "system" control from "application" control in the user's mind, can be completely ignored when not needed, and is easier to explain. Designing an adequate process for acquiring exploiting input of form 1 is far easier than for forms 2 and 3. We next describe specific issues related to the latter forms.

Mechanism transition: In our system, changing a VM's schedule is virtually instantaneous, if the schedule is feasible on the physical host it is currently running on. If the desired schedule is not feasible, we must indicate this to the user and use a different mechanism (e.g., migrate his VM to a different host) to satisfy him. While very fast VM migration techniques now exist [6, 28], they still take much longer than changing a schedule, and have a much higher resource cost. How can we represent these time and resource costs to the user?

Categorical dimensions: A configuration can be thought of as a point within a multidimensional space. If a dimension is categorical (for example, a VM can be mapped to one of several choices, or a overlay link can be added or not), it is difficult to present it using an easily understood external interface.

Dimensionality: In the present work, we expose the schedule directly to the user. This is easy to do because its two dimensions map directly to the two dimensions of the joystick, and both dimensions are continuous. As we add resources, the number of dimensions grows and makes a simple mapping impossible. Of course, there are many examples of using low dimensional input devices to explore high dimensional spaces. A large part of the problem is how to visualize the current configuration and its neighborhood.

8 Conclusions and future work

We have described and evaluated a technique for putting even naive users in direct, explicit control of the scheduling of their interactive computing environments through the combination of a joystick and an on-screen display of cost. In so doing, we have demonstrated that with such input it is *possible* and *practical* to adapt the schedule dynamically to the user, letting him trade off between the comfort of the environment and its cost. Because the tolerance for cost and the comfort with a given schedule is highly dependent on both the applications being used and on the user himself, this technique seems very fruitful both for tailoring computing environments to users and making them cheaper for everyone.

We are currently exploring how to extend our results to scheduling other resources, combinations of resources, and in power management, focusing on the distributed adaptation problem described in the previous section.

References

[1] R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb. Simplifying cyber foraging for mobile devices. Technical Report CMU-CS-05-157, Computer Science Department, Carnegie Mellon University, August 2005.

[2] N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling: Investigating unfairness. In *Proceeds of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 279–290, 2001.

[3] J. Bennett and H. Zhang. Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM 1996*, pages 120–127, March 1996.

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

[5] H.-H. Chu and K. Narhstedt. Cpu service classes for multimedia applications. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, June 1999.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[7] P. Dourish. Evolution in the adoption and use of collaborative technologies. In *Proceedings of the ECSCW Workshop on the Evolving Use of Groupware*, September 1999.

[8] K. J. Duda and D. R. Cheriton. Borrowed-virtual-time (bvt) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. In *Proceedings of the seventeenth ACM symposium on Operating systems principles (SOSP)*, pages 261–276, 1997.

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

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

[11] R. Goldberg. Survey of virtual machine research. *IEEE Computer*, pages 34–45, June 1974.

[12] A. Gupta and P. A. Dinda. Inferring the topology and traffic load of parallel programs running in a virtual machine environment. In *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing (JSPPS 2004*, June 2004.

[13] A. Gupta, B. Lin, and P. A. Dinda. 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.

[14] A. Gupta, M. Zangrilli, A. Sundararaj, A. Huang, P. Dinda, and B. Lowekamp. Free network measurement for virtual machine distributed computing. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[15] M. Jones, D. Rosu, and M.-C. Rosu. Cpu reservations and time constraints: Efficient, predictable scheduling of independent activities. In *Proceedings of the 16th ACM Symposium on Operating System Principles (SOSP)*, 1997.

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

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

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

[19] J. Lange, A. Sundararaj, and P. Dinda. Automatic dynamic run-time optical network reservations. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 255–264, July 2005.

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

[21] B. Lin and P. A. Dinda. Putting the user in direct control of cpu scheduling. Technical Report NWU-EECS-06-07, Northwestern University, July 2006.

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

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

[24] R. Lotlika, R. Vatsavai, M. Mohania, and S. Chakravarthy. Policy scheduler advisor for performance management. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing (ICAC)*, June 2005.

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

[26] M. McKusick, K. Bostic, M. Karels, and J. Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Addison-Wesley Longman, 1996.

[27] K. Nagaraja, F. Oliveira, R. Bianchini, R. Martin, and T. Nguyen. Understanding and dealing with operator mistakes in internet services. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2004.

[28] M. Nelson, B.-H. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *Proceedings of the USENIX Annual Technical Conference*, 2005.

[29] J. Nieh and M. Lam. The design, implementation, and evaluation of SMART: A scheduler for multimedia applications. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, October 1997.

[30] C. J. Reynolds. The sensing and measurement of frustration with computers. Master's thesis, Massachusetts Institute of Technology Media Laboratory, 2001.

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

[32] P. Romano. Itu-t recommendation t.128 (application sharing). Technical report, ITU, March 1997.

[33] B. Schmidt, M. Lam, and J. Northcutt. The interactive performance of slim: A stateless thin client architecture. In *Proceedigns of the 17th ACM Symposium on Operating Systems Principles (SOSP 1999)*, pages 32–47, December 1999.

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

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

[36] J. Strickland, V. Freeh, X. Ma, and S. Vazhkudai. Governor: Autonomic throttling for aggressive idle resource scavenging. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing (ICAC)*, June 2005.

[37] A. Sundararaj and P. Dinda. Towards virtual networks for virtual machine grid computing. In *Proceedings of the 3rd USENIX Virtual Machine Research And Technology Symposium (VM 2004)*, May 2004.

[38] A. Sundararaj, A. Gupta, , and P. Dinda. 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)*, pages 47–58, July 2005.

[39] A. Sundararaj, M. Sanghi, J. Lange, and P. Dinda. An optimization problem in adaptive virtual environments. In *Proceedings of the Seventh Workshop on Mathematical Performance Modeling and Analysis (MAMA)*, June 2005.

[40] A. I. Sundararaj, M. Sanghi, J. R. Lange, and P. A. Dinda. Hardness of approximation and greedy algorithms for the adaptation problem in virtual environments. Technical Report NWU-EECS-06-06, Department of Electrical Engineering and Computer Science, Northwestern University, July 2006.

[41] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, 1994.

[42] A. Whitaker, R. Cox, and S. Gribble. Configuration debugging as search: Finding the needle in the haystack. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2004.