Increasing Interactivity by Predicting User Actions Craig Zilles <zilles@cs.uiuc.edu>

Interactive workloads are characterized by relatively long periods of idle time punctuated with bursts of activity (see Figure 1). Despite these low levels of average processor utilization, a user may not have enough computational power when it is necessary. The key problem is that computation is required in bursts (called interactive episodes, or IE's, in [F+01]), where: 1) the user performs some action, 2) some computation is performed, and 3) the results are presented back to the user. When the duration of an IE exceeds the human perception threshold—between 50 and 200 ms for most kinds of interaction [O98]—the computation latency becomes noticeable, to the detriment of the user experience (Figure 2). Flautner, et al. find that the execution time of many of the programs they studied is dominated by IE's that exceed the perception threshold.

Our insight into this problem is that sufficient computational power is available to perform the desired computation before its deadline (i.e., the perception threshold), just not before the user input is available. Thus, if the user input can be predicted, we can initiate the computation early, so that it is available in time for presentation to the user before the perception threshold (Figure 3). Presentation is deferred until the prediction is confirmed to avoid the appearance of second guessing the user.

Figure 1

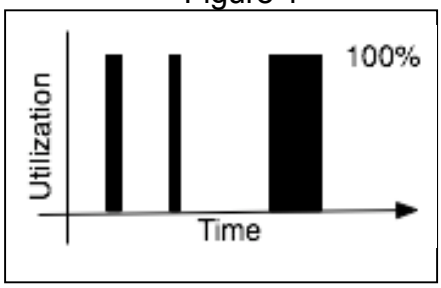


Figure 2

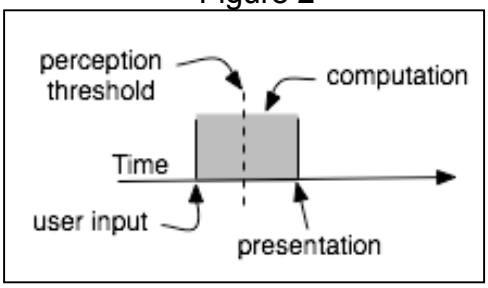
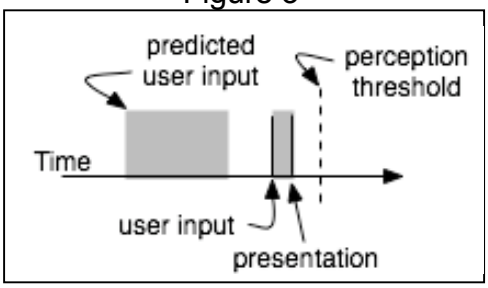


Figure 3



Predicting user actions can be surprisingly accurate, due to the repetitious nature of humans. Davison and Hirsh found that Unix command lines could be predicted with a 40 percent accuracy using a history of previous commands [DH98]. Gorniak and Poole achieved accuracies as high as 80 percent with an application-independent approach for applications with graphical user interfaces (GUI) [GP00]. We believe that this content-oriented prediction can be further improved by using kinematic cues available from trajectories in pointing devices used in modern GUIs. Furthermore, it is sufficient to predict the actions that involve a lot of processing, which is a relatively small fraction of all actions [F+01].

Executing a program with a predicted input requires speculation support to: 1) checkpoint a program's existing state, 2) sandbox the speculative execution, and 3) commit the speculation if validated by user input. While there are significant challenges involved with these technologies, they are already active areas of research for system reliability, security, and debugging (*e.g.,* [S+04, S+03, W+95]).

One obvious source of user exposed latency that may be amenable to optimization without a generalized speculation framework is that of application startup. Checkpointing is not required, because the whole process can be killed if it is found to be unnecessary. Furthermore, the ways in which programs fork each other is limited, facilitating predictability. Maybe a day will come when you click on an email attachment and it will be instantly open!

[DH98] Brian D. Davison and Haym Hirsh, Predicting Sequences of User Actions, Working Notes of the Joint Workshop on Predicting the Future: AI Approaches to Time Series Analysis, 1998.

[F+01] Krisztian Flautner, Steve Reinhardt and Trevor Mudge, Automatic Performance Setting for Dynamic Voltage Scaling, MOBICOM 2001.

[GP00] Peter Gorniak and David Poole, Predicting Future User Actions by Observing Unmodified Applications, Seventeenth National Conference on Artificial Intelligence (AAAI-2000), August 2000.

[O98[D. R. Olsen, Developing User Interfaces, Morgan Kaufmann Publishers, 1998

[S+04] Sudarshan Srinivasan, Srikanth Kandula, Christopher Andrews and Yuanyuan Zhou. Flashback: A Light-weight Rollback and Deterministic Replay Extension for Software Debugging. The proceedings of the annual Usenix technical conference (USENIX'04), June 2004.

[S+03] Michael M. Swift, Brian N. Bershad, Henry M. Levy Improving the Reliability of Commodity Operating Systems Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003.

[W+95] Y. M. Wang, Y. Huang, K.-P. Vo, P. Y. Chung, and C. Kintala, " Checkpointing and its applications," in Proc. IEEE Fault-Tolerant Computing Symposium (FTCS-25), pp. 22-31, June 1995.