Anticipatory Procrastination

Joshua Yanchar Matt Elder Bill Harris Donald Knuth Chris Hinrichs

yanchar@cs.wisc.edu elder@cs.wisc.edu wrharris@cs.wisc.edu hinrichs@cs.wisc.edu

Abstract

Fundamental theoretical results demonstrate that many, no less than 92, things are hard. Moreover, things can be boring, unpleasant, or in the pathological case, both. To redistribute such effort away from humans was the field of computer science born, shifting effort to machines and grad students.

To cut this Gordian Knot of "having to do stuff", we propose an elegant solution: "Not doing it today". We call our technique *anticipatory procrastination*. While this may seem an unintuitive approach, we feel we've solved this; it seems reasonable at a high level. We've found empirical evidence that such an approach can work. In fact, this may be the most commonly implemented solution to the general problem of getting stuff done, the well-known STUFF problem.

1. Introduction

Fundamental theoretical results demonstrate that many, no less than 92, things are hard. [3] Moreover, things can be boring [9], unpleasant [8], or in the pathological case, both [5]. To redistribute such effort away from humans was the field of computer science born, shifting effort to machines and grad students. Unfortunately, Turing proved that not all work can be shoved off onto such automata [1]. As a result, humans are left with nontrivial amounts of work, requiring advanced scheduling and work assignment heuristics. Their performance is often unacceptable.

To cut this Gordian Knot of "having to do stuff", we propose an elegant solution: "Not doing it today". We call our technique *anticipatory procrastination*. While this may seem an unintuitive approach, we feel we've solved this; it seems reasonable at a high level. We've found empirical evidence that such an approach can work. In fact, this may be the most commonly implemented solution to the general problem of getting stuff done, also known as the STUFF problem.

This is the table of contents paragraph, it will be over shortly. Section 2 describes prior academic work in the field. Section 3 illustrates our experimental methods and results, and demonstrates the scalability and parallelism of our radical approach. Section 4 concludes.

2. Related Work

We have seen this solution to STUFF throughout academic history. For an early example, consider Fermat's 1637 statement, "I have a truly marvellous proof of this proposition which this margin is too narrow to contain." This statement incorporated a landmark proof technique. The ingenuity of this technique allowed Fermat to delegate his proof to 400 years of his students while keeping his name attached to the theorem. This technique remains relevant today, most commonly encountered as exercises left to the reader.

For decades, it has been assumed that the surest way to arrive quickly at the results of a computation is through sophisticated algorithms, efficient implementation, and other such mind-rending work. Gottbrath et al., however, demonstrate that an effective technique to obtain results is to push optimization downstream. Their approach is for you to sit on your hands and wait for Intel do the work. [4]

Another clear application of Anticipatory Procrastination involves responding to undergraduate questions, or, more accurately, not responding to undergraduate questions. For example, one simulated graduate student's experience demonstrates that our algorithm improves performance. [2]

Knuth has demonstrated effective techniques for resisting low latency responses in the presence of high-performance communication channels. By batching his physical messaging in three month quanta and simply not addressing electronic messaging, Knuth ignores you more efficiently than you would otherwise be ignored. He also proves that a large subclass of STUFF has solutions that follow directly from ignoring it. [7]

A disk read head seeks slowly across its disk. Thus, an eager disk scheduler can send the disk head on an expensive seek when a small amount of waiting would have allowed it to perform an operation locally. Iyer and Druschel thus show that anticipatory procrastination can be as effective for system devices as it is for higher level task scheduling. [6]

A recent example of anticipatory procrastination in an academic context is Assignment 2 of the University of Wisconsin Advanced Operating Systems class. Results of this work are still preliminary, but are regarded within the relevant community [1] as a resounding success.

3. Experimental Methods and Results

See Figure 1.

4. Conclusion

To cut this Gordian Knot of "having to do stuff", we have proposed an elegant solution: "Not doing it today". This may have seemed an unintuitive approach, but we've found empirical evidence that such an approach can work. In fact, this may be the most commonly implemented solution to STUFF.

[1] us.

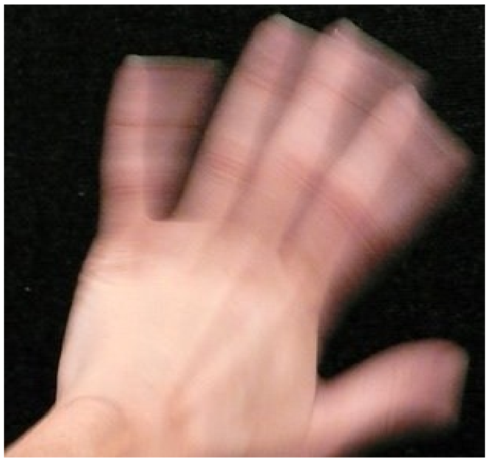

Figure 1. Summary of methods and results.

References

[1] M. Alan. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.

[2] Jorge Cham. 24-hour waiting period. `http://www.phdcomics.com/blog.php?postarchive=1&previous=1203352531`, 2008.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[4] C. Gottbrath, J. Bailin, C. Meakin, T. Thompson, and JJ Charfman. The Effects of Moore's Law and Slacking on Large Computations. *Arxiv preprint astro-ph/9912202*, 1999.

[5] Haryadi S. Gunawi, Cindy Rubio-González, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Ben Liblit. EIO: Error-handling is occasionally correct. In *Proceedings of the Sixth USENIX Conference on File and Storage Technologies (FAST '08)*, San Jose, California, February 2008. USENIX.

[6] S. Iyer and P. Druschel. Anticipatory scheduling: a disk scheduling framework to overcome deceptive idleness in synchronous I/O. *ACM SIGOPS Operating Systems Review*, 35(5):117–130, 2001.

[7] Donald Knuth. Email. `http://www-cs-faculty.stanford.edu/~knuth/email.html`, 1990.

[8] Ben Liblit, Mayur Naik, Alice X. Zheng, Alex Aiken, and Michael I. Jordan. Public deployment of cooperative bug isolation, May 24 2004.

[9] Vijayan Prabhakaran, Lakshmi N. Bairavasundaram, Nitin Agrawal, Haryadi S. Gunawi, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. IRON File Systems. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 206–220, Brighton, United Kingdom, October 2005.