Deploying Architectural Support for Software Defect Detection in Future Processors

Yuanyuan Zhou and Josep Torrellas
Department of Computer Science, University of Illinois at Urbana-Champaign
{yyzhou,torrellas}@cs.uiuc.edu

1 Motivation

Recent impressive advances in semiconductor technology are enabling the integration of ever more transistors on a single chip. This trend provides a unique opportunity to enhance processor functionality in areas other than performance, such as in easing software development or enhancing software debugging.

Exploring architectural support for software debugging is a research direction of great promise. First, it enables new tradeoffs in the performance, accuracy, and portability of software defect detection tools, possibly allowing the detection of new types of defects under new conditions. Secondly and most importantly, it opens up a new possibility: on-the-fly software defect detection in *production runs*. This last area has been under-explored due to the typically large overheads of many software solutions.

Table 1 briefly summarizes the architectural supports for software defect detection that our research group has recently proposed.

Architectural Support	Description	Overhead
ReEnact [1]	Extend the communication monitoring mechanisms in thread-level speculation to detect and characterize data races automatically on the fly.	1% to 13%
iWatcher [4]	Associate program-specified monitoring functions with memory locations. When any such location is accessed, the monitoring function is automatically triggered with low overhead without going through the OS.	4% to 80%
AccMon [3]	Detect general memory-related bugs by extracting and monitoring the set of instruction PCs that normally access a given monitored object.	0.24X to 2.88X
SafeMem [2]	Exploit existing ECC memory technology to detect memory corruption and prune false positives in memory leak detection.	1% to 29%

Table 1: Architectural supports for software defect detection recently proposed by our research group.

In general, using architectural support to detect software defects provides several key advantages over software-only dynamic approaches:

(1) Performance. Architectural support can significantly lower the overhead of dynamic monitoring if it eliminates the need for extensive code instrumentation. Such instrumentation may even interfere with compiler optimizations. Moreover, the hardware can be used to speed up certain operations. For example, AccMon uses a hardware Bloom filter to filter 80-99% of the accesses that would go to a software monitor function. As shown in Table 1, hardware support enables dynamic detection of bugs with orders-of-magnitude less overhead than commonly existing software-only tools.

(2) Accuracy. Architectural support enables ready access to execution information that is often hard to obtain with software-only instrumentation. An example is all the accesses to a monitored memory object and only those — instrumentation-based tools need to check more memory accesses due to pointer aliasing problems. This capability is leveraged by iWatcher [4]. Another example of hard to obtain information is the exact interleaving of the accesses that caused a data race in a multithreaded program running at production speeds. This capability is leverages by ReEnact [1] to detect production-run data races.

(3) Portability. Architectural support is typically language independent, cross module (capable of checking for bugs in third-party libraries), and easy to use with low-level system code such as the operating system. Moreover, it can be designed to work with binary code without recompilation.

2 Deployment Challenges and Plans

Due to the above advantages, it is highly desirable for software developers to be able to use architectural support to assist in detecting software defects. Of course, to make this happen, we need the cooperation of processor design companies such as Intel, IBM, AMD, or Sun Microsystems. To help improve the chances that one or more of these companies finds it attractive to include architectural support for software debugging in their processors, we put forward the following suggestions:

(1) The research communities working on software defect detection tools and on computer architecture can work together to identify (i) what are important or difficult-to-catch bugs (e.g. data races) that need architectural support? and (ii) what architectural extensions can be added to exisiting processors to help detect these bugs?

The architectural supports that are more likely to succeed are those that have one or several of the following features: simplicity, generality, reconfigurability and leverage existing hardware. Simple designs are those that require modest extensions to current processor hardware, such as iWatcher or SafeMem. General designs are those that can be used for multiple purposes, such as debugging and profiling, as also exemplified by iWatcher. Reconfigurable designs can easily be disabled, enabled or reconfigured for other purposes so that hardware vendors can manufacture only one type of hardware but can configure it for different users. Finally, designs that leverage existing hardware and use it for other purposes are likely to be successful. For example, SafeMem makes novel use of ECC memory for bug detection.

(2) Software companies such as Microsoft with significant leverage on processor companies can push the latter to provide the necessary architectural hooks in their processors. Software companies can also be willing to buy some special "testing" processors (processors with sophisticated bug detection support) that are priced higher than regular processors. Similarly, software companies can motivate customers to buy such processors by offering them incentives such as lower license fees or better services.

(3) Interest in the topic of architectural support for software productivity and debuggability can be broadened through workshops, conferences, and funding efforts in this area.

REFERENCES

[1] M. Prvulovic and J. Torrellas. ReEnact: Using Thread-Level Speculation Mechanisms to Debug Data Races in Multithreaded Codes. In *ISCA*, 2003.

[2] F. Qin, S. Lu, and Y. Zhou. SafeMem: Exploiting ECC-Memory for Detecting Memory Leaks and Memory Corruption During Production Runs. In *HPCA*, Feb 2005.

[3] P. Zhou, W. Liu, F. Long, S. Lu, F. Qin, Y. Zhou, S. Midkiff, and J. Torrellas. AccMon: Automatically Detecting Memory-Related Bugs via Program Counter-based Invariants. In *Micro*, Dec 2004.

[4] P. Zhou, F. Qin, W. Liu, Y. Zhou, and J. Torrellas. iWatcher: Efficient Architectural Support for Software Debugging. In *ISCA*, 2004.