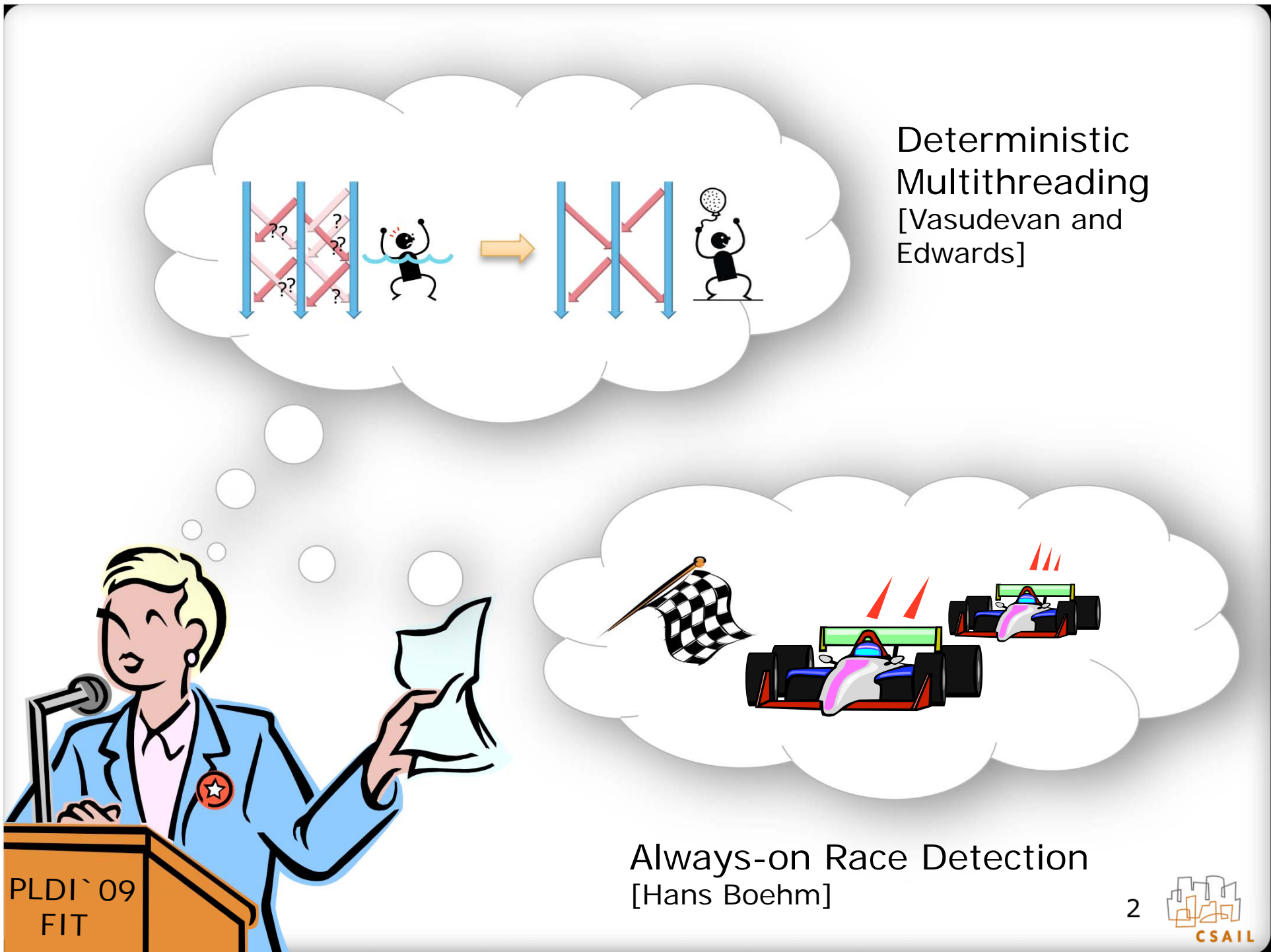


Outfoxing the Mammoths

Marek Olszewski
Saman Amarasinghe

MIT CSAIL
Commit Group

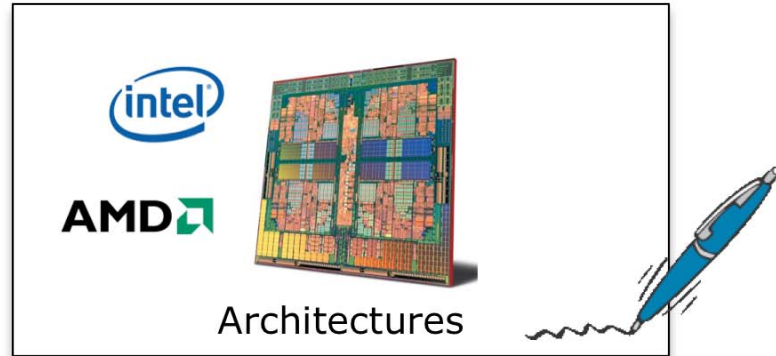
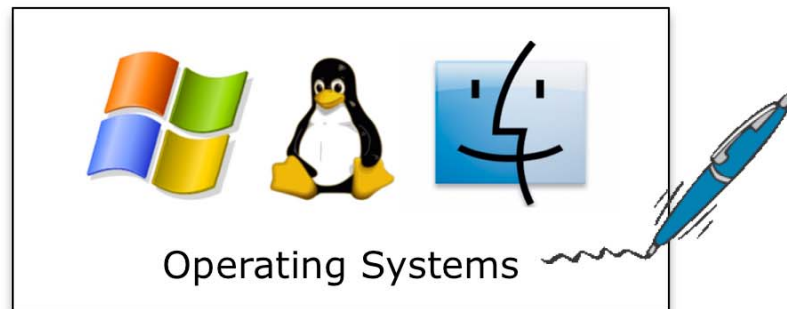
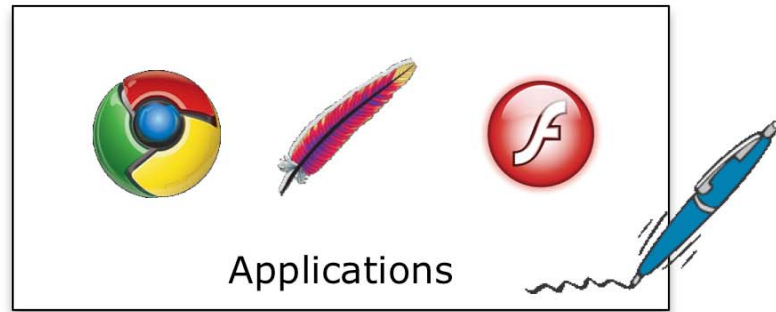




Deterministic
Multithreading
[Vasudevan and
Edwards]

Always-on Race Detection
[Hans Boehm]

Tackling these ambitious challenges:



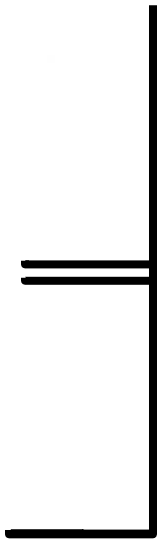


Where we
want to be

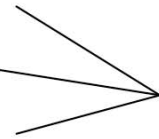


Greatest invention since the transistor:

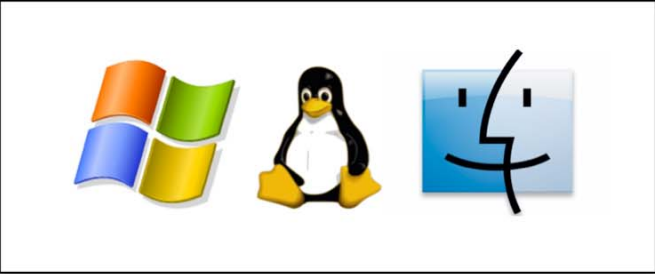




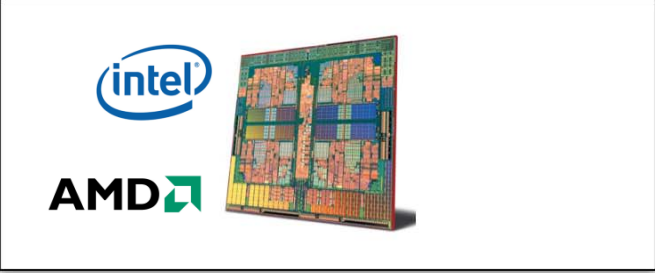
DynamoRIO
Pin
Valgrind

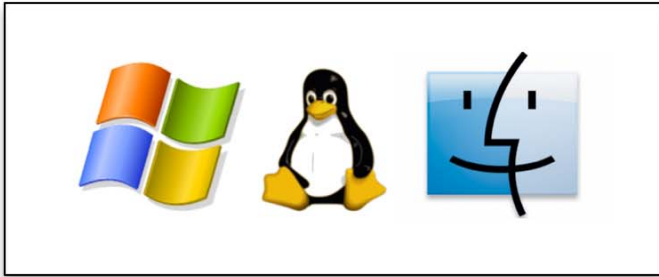


**Application-Top
Boxes**

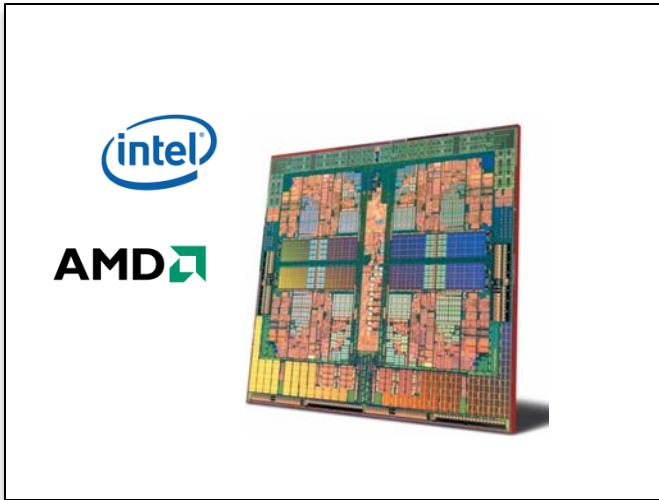


OS-Top Boxes

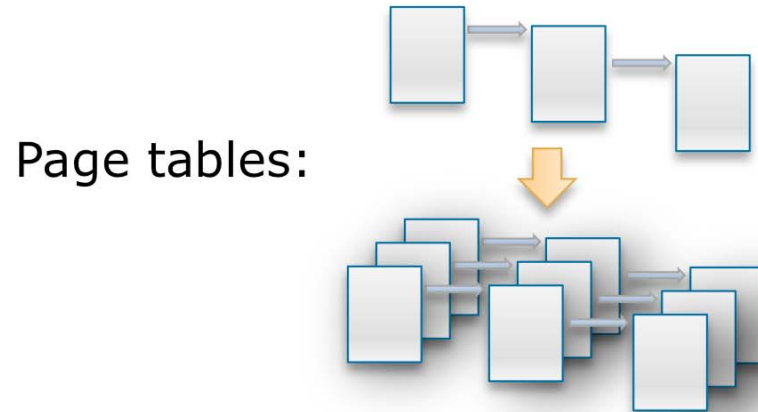




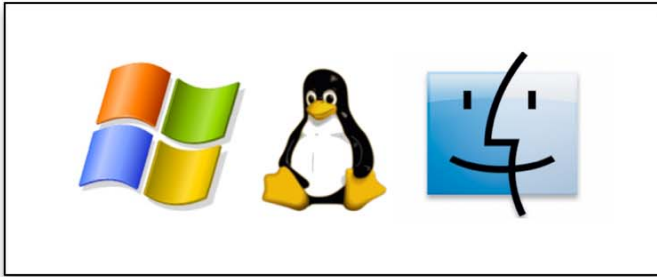
Aikido VM



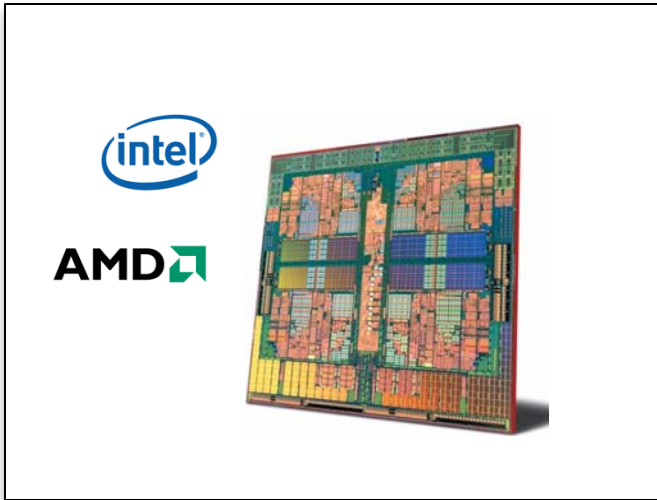
- Adds per-thread page protection support to threads running in single address space
- Duplicates shadow pages:



- Makes it possible to cheaply detect which data each thread accesses



Aikido VM



- Good for systems that instrument accesses to shared data:
 - Race Detectors
 - Deterministic Multithreading
 - STMs, TLS, etc...
- Currently conservative:
 - Instrument all instructions that ***might*** access shared memory
- Instead:
 - Dynamically detect which instructions ***definitely*** access shared ***pages***, and instrument only those

Questions?



**Aikido
VM**

<http://groups.csail.mit.edu/commit/>

Backup Slides

Originally inspired by: Overshadow by Chen *et al.* ASPLOS '08

Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems

Xiaoxin Chen Tal Garfinkel E. Christopher Lewis Prapat Subrahmanyam Carl A. Waldspurger
Dan Boneh* Jeffrey Dvoskin† Dan R.K. Ports‡
VMware, Inc. *Stanford University †Princeton University ‡MIT
{mchen,talg,lewis,pratap,carl}@vmware.com dabo@cs.stanford.edu jdvoskin@princeton.edu drkp@mit.edu

Abstract

Commodity operating systems entrusted with securing sensitive data are remarkably large and complex, and consequently, frequently prone to compromise. To address this limitation, we introduce a virtual-machine-based system called *Overshadow* that protects the privacy and integrity of application data, even in the event of a total OS compromise. *Overshadow* presents an application with a normal view of its resources, but the OS with an encrypted view. This allows the operating system to carry out the complex task of managing an application's resources, without allowing it to read or modify them. Thus, *Overshadow* offers a line of defense for application data.

Overshadow builds on *multi-shadowing*, a novel mechanism that presents different views of “physical” memory, depending on the context performing the access. This primitive offers an additional dimension of protection beyond the hierarchical protection domains implemented by traditional operating systems and processor architectures.

We present the design and implementation of *Overshadow* and show how its new protection semantics can be integrated with existing systems. Our design has been fully implemented and used to protect a wide range of unmodified legacy applications running on an unmodified Linux operating system. We evaluate the perfor-

Unfortunately, the security provided by commodity operating systems is often inadequate. Trusted OS components include not just the kernel but also device drivers and system services that run with privilege (e.g., daemons that run as root in Linux). These components generally comprise a large body of code, with broad attack surfaces that are frequently vulnerable to exploitable bugs or misconfigurations. Once such privileged code is compromised, an attacker gains complete access to sensitive data on a system. While some facets of security in these systems will continue to improve, we believe competitive pressures to provide richer functionality and retain compatibility with existing applications will keep the complexity of such systems high, and their assurance poor.

To ameliorate this problem, many have attempted to retrofit higher-assurance execution environments onto commodity systems. Previous efforts have explored executing applications handling sensitive data in separate virtual machines [10, 29, 8], using secure co-processors [7], or changing the processor architecture to introduce orthogonal protection mechanisms that protect application data from the OS [6, 13, 16, 19, 27]. Unfortunately, these generally demand major changes in the way that applications are written [7, 8, 16, 18, 28] and used [8, 10], and how OS resources are managed [10, 29]. Such radical departures pose a substantial barrier to adoption.

PetraVM Jinx

Uncover Hidden Concurrency Bugs Quickly

Complex concurrency bugs can stay hidden in your code for thousands of hours of operation, only crashing your application when certain conditions align. Jinx makes hidden bugs appear in development and test, before your customers find them.



Uses a VM to transparently checkpoint and explore different interleavings of shared memory communication to try to trigger bugs.