LOG-BASED ARCHITECTURES

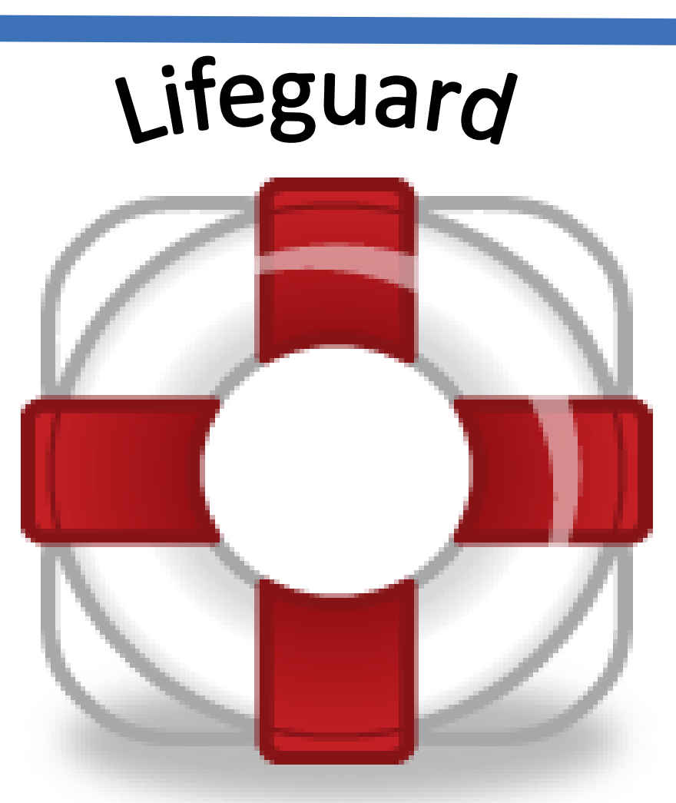
Phillip Gibbons, Michael Kozuch (Intel) Todd Mowry, Onur Mutlu (CMU)
Michelle Goodstein, Gennady Pekhimenko, Olatunji Ruwase, Vivek Seshadri (CMU)

MOTIVATION: SAFER SOFTWARE EXECUTION

Eliminating all software bugs prior to release is difficult

Lifeguards (software monitoring tools) can catch failures at runtime

Unfortunately, software-only lifeguards are too slow (10X-100X slowdown)

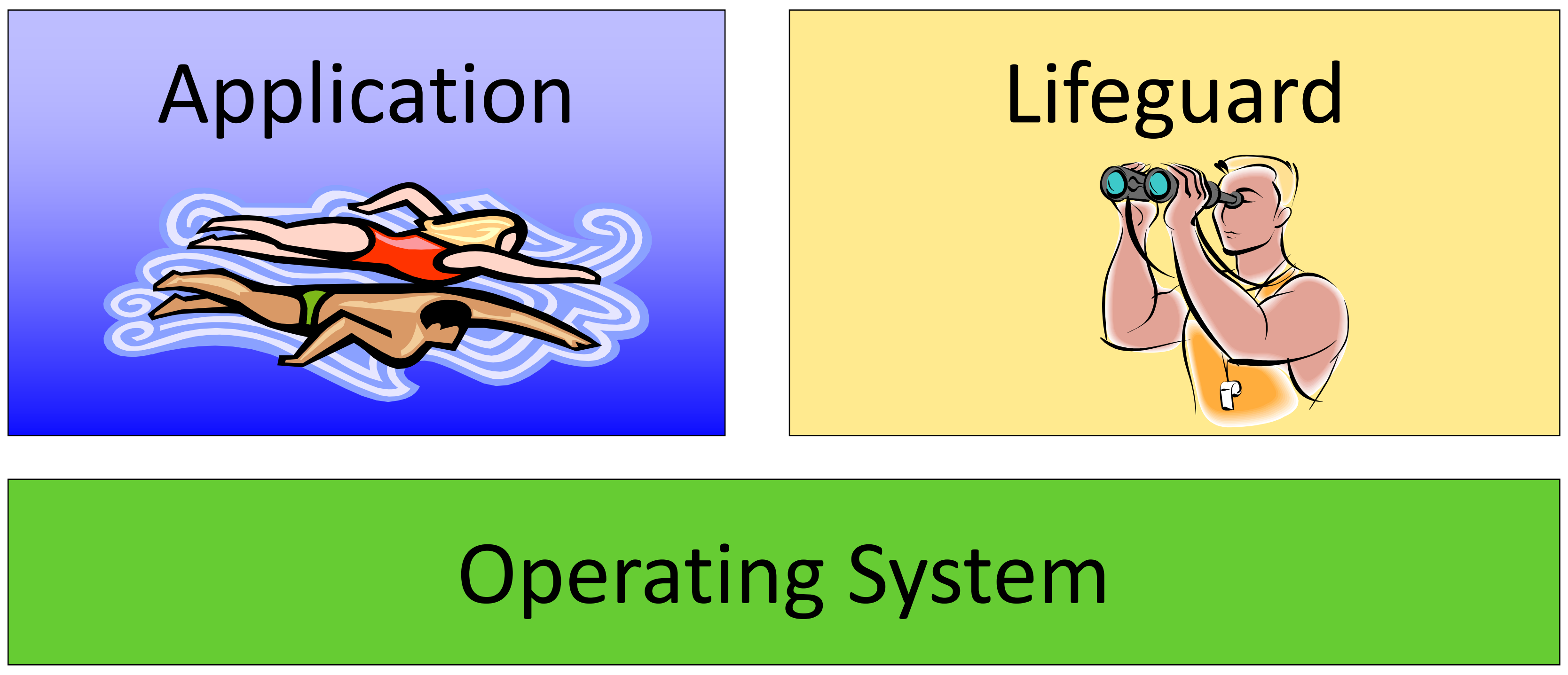


PROJECT GOAL

Design hardware and system software support enabling a broad range of powerful lifeguards without sacrificing application performance

Application:

Unmodified, but optional library annotations bridge software-hardware semantic gap

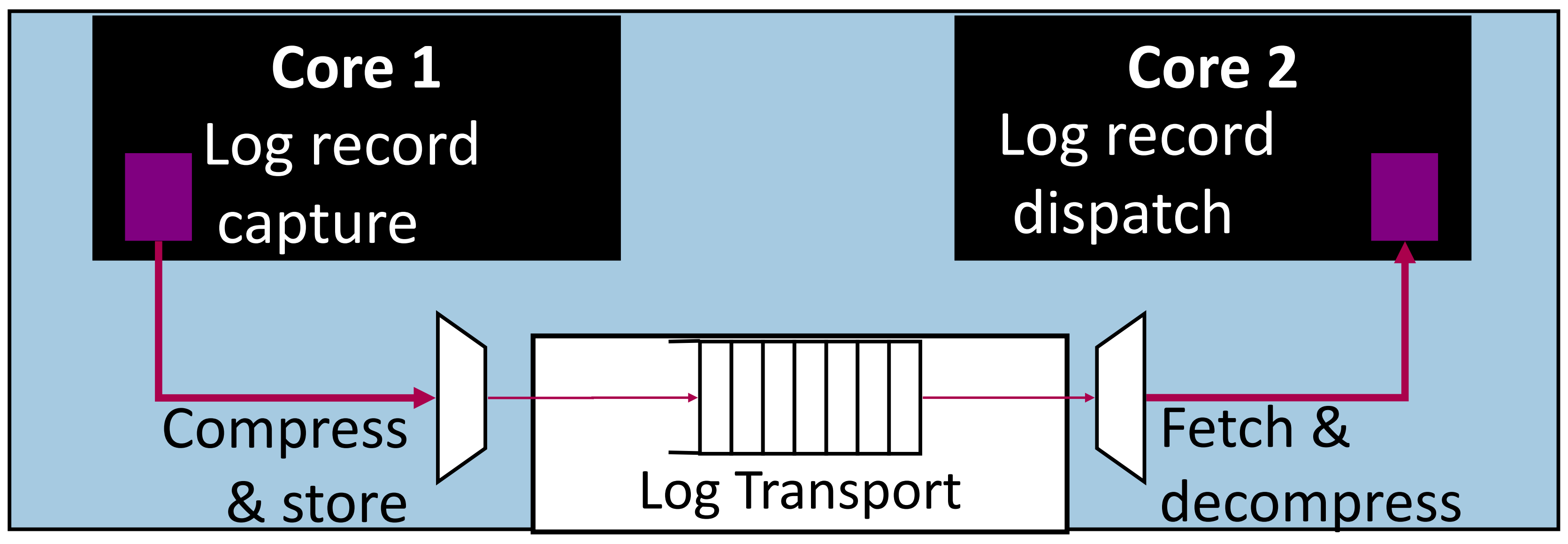


Example Lifeguards:

- Data flow tracker
- Data race detector
- Memory access checker
- Control flow verifier

Operating System:

Stop-on-system call support limits damage from software bugs



Event-driven Support:

Eliminates lifeguard fetch-and-decode loop and enables efficient filtering

KEY HARDWARE IDEAS

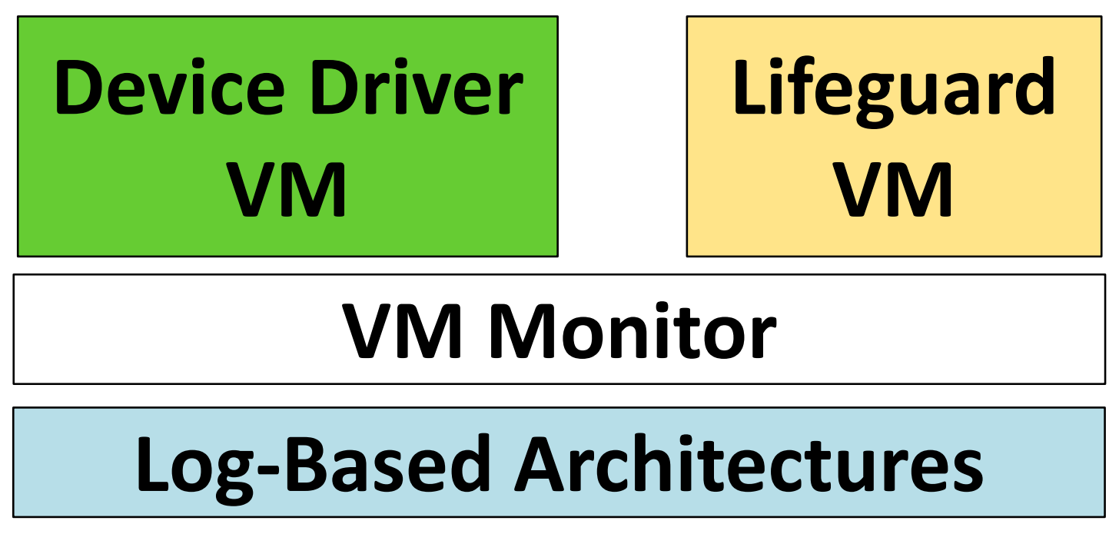
- Multicore processors provide additional execution resources to run lifeguards
- Fine-grained application events are captured in a **log** during execution
- **Hardware accelerators** speed up common lifeguard functionalities

PROGRESS TO DATE

- LBA reduces average overhead for monitoring single-core apps to 2% (vs. 20X) for AddrCheck, 40% (vs. 32X) for LockSet, and 36% (vs. 58X) for TaintCheck [ISCA 2008, Micro TopPicks 2009]
- LBA can effectively monitor parallel applications [two papers in ASPLOS 2010]
- LBA hot path optimizations reduce need for hardware accelerators [PLDI 2010]

CURRENT RESEARCH CHALLENGES

- Monitoring software updates – see poster "*Runtime Validation of Incremental Code Changes*"
- Monitoring parallel applications with improved precision on relaxed memory-consistency models
- Monitoring system software (e.g. device drivers) – see figure at right



Using Virtual Machines to Monitor Kernel-Mode Drivers

Intel Science & Technology Center for Cloud Computing