Safer Software Execution through Log-Based Architectures

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)

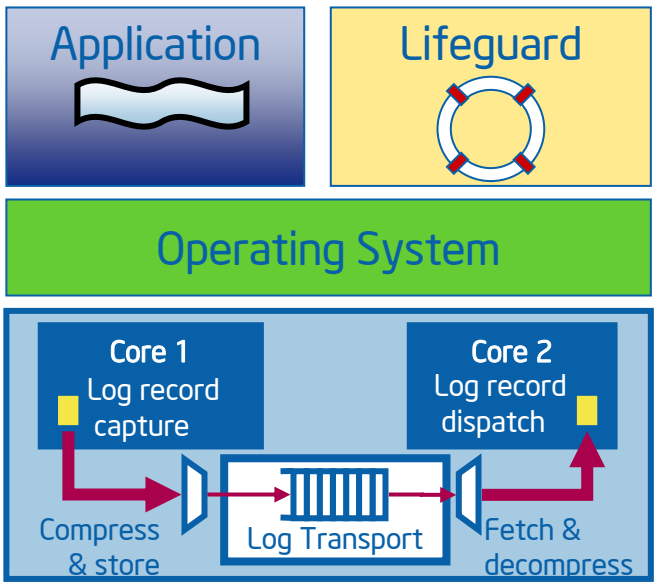
Project goal: *Design hardware support enabling a broad range of powerful lifeguards without sacrificing main program performance*

Application:

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

Operating system:

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



Example lifeguards:

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

Event-driven support:

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

Key Hardware Ideas

Multi-core processors provide additional execution resources to run lifeguards

Fine-grained application events are captured in a *log* during execution

The log can be transported via on-die cache, reducing bus contention

Log compression reduces cache space and bandwidth requirements

Research at Intel
www.intel.com/research