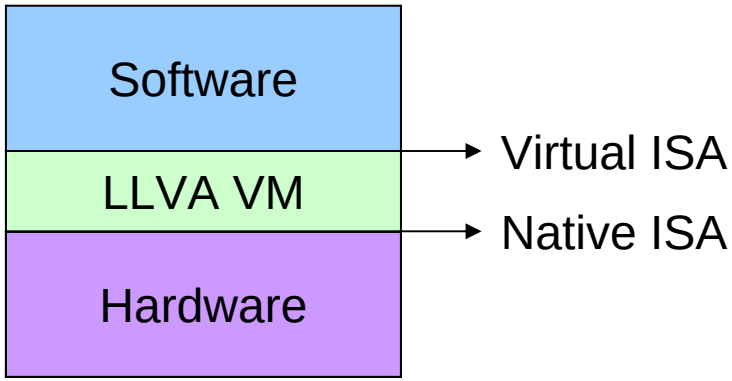
Secure Virtual Architecture

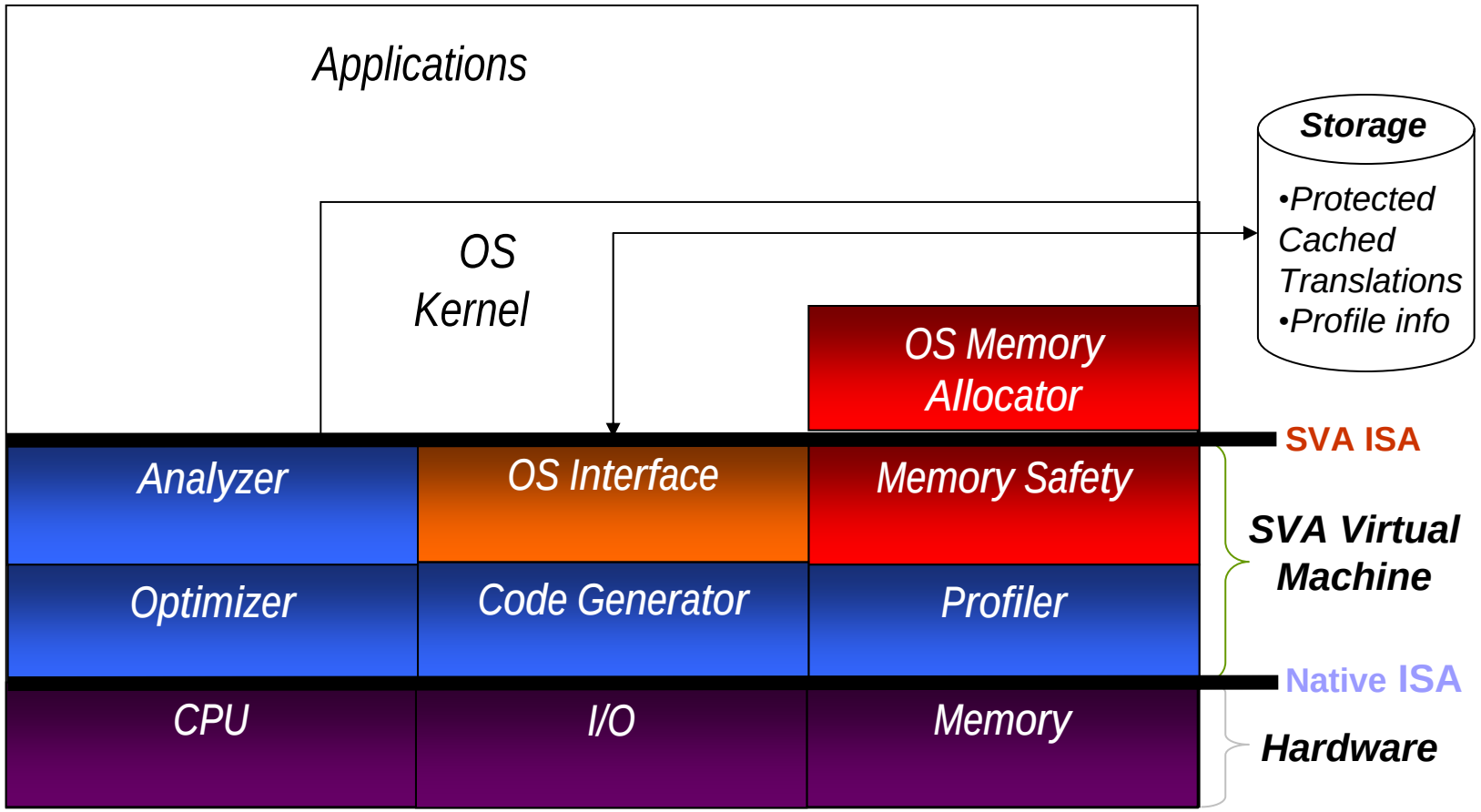
John Criswell
University of Illinois at Urbana-Champaign

Secure Virtual Machine



- What is it?
  - A *compiler-based* virtual machine running *below* an operating system
  - Enforces security policies for *all* software (including the OS kernel)
- Why build it?
  - Kernel code can be exploited
  - Allows analysis across traditional boundaries

SVA System Architecture



SVA: OS Interface

- Kernels require new functionality
  - Hardware Control
    - Performing I/O
    - Installing interrupt handlers
  - State Manipulation
    - Context switching
    - Signal handler dispatch

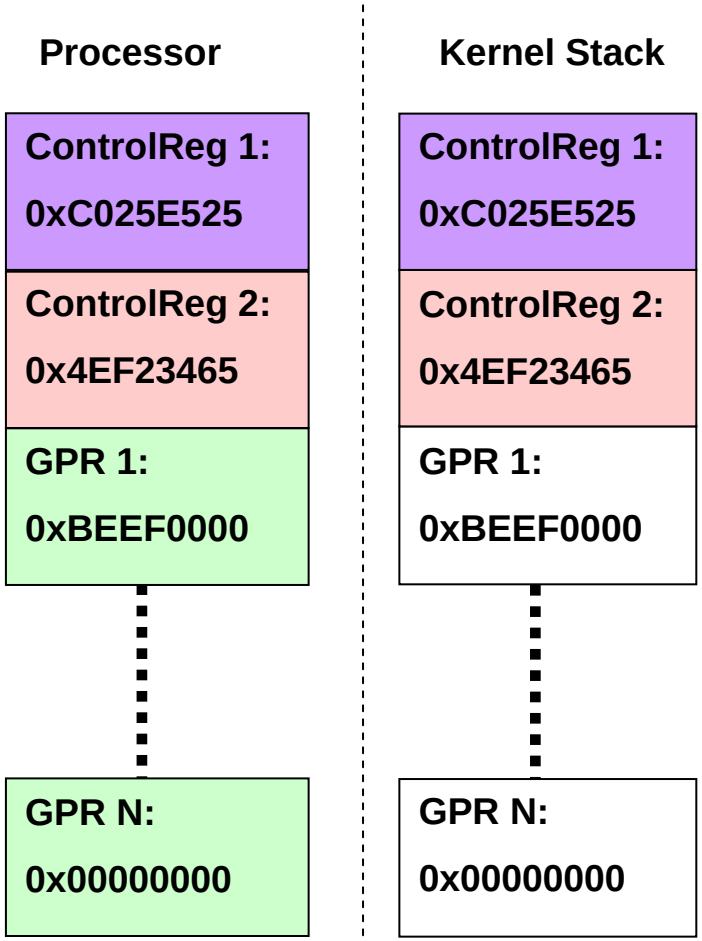
Hardware Control

- Registration functions
  - void *llva_register_syscall* (int number, int (*f)(void * icontext, …))
  - void *llva_register_interrupt* (int number, int (*f)(void * icontext))
  - void *llva_register_exception* (int number, int (*f)(void * icontext))
- I/O
  - int *llva_io_read* (ioptr_t ioaddress)
  - void *llva_io_write* (ioptr_t ioaddress, int value)
- Atomic Operations
  - int *llva_swap_and_phi* (void * address, int value)
  - int *llva_compare_and_swap* (void * address, int compare, int value)
- Memory Management
  - void *llva_load_pgtable* (void * table)
  - void * *llva_save_pgtable* ()

State Manipulation

- Allow OS to see the *existence* of native state
- OS does not understand the semantics of native state

Lazy State Saving on Interrupt



- How to take advantage of low latency interrupt facilities?
  - shadow registers (e.g. ARM)
  - register windows (e.g. SPARC)
- On interrupt, SVM saves subset of processor state on the kernel stack
- Can leave state in registers if kernel does not overwrite it
- Kernel can commit all state to memory if required
- Pointer to Interrupt Context passed to system call, interrupt, and trap handlers

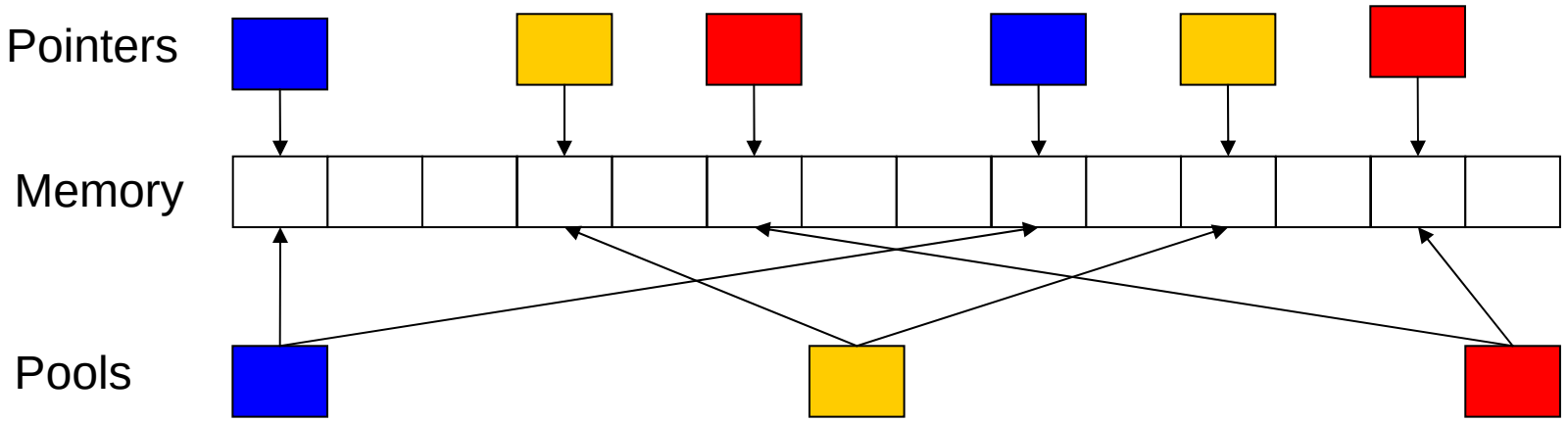
Manipulating Interrupt Context

- Interrupt Context ←→Memory
  - void *llva_icontext_save* (void * icontext,
    void * buffer
  - void *llva_icontext_load* (void * icontext,
    void * buffer)
- Commit
  - void *llva_icontext_commit* (void * icontext)
- Push function frames
  - void *llva_ipush_function* (void * icontext,
    void (*f)(…), …)

Manipulating Processor State

- Context Switching (manipulates current state)
  - void *llva_save_integer* (void * buffer)
  - void *llva_load_integer* (void * buffer)
  - void *llva_save_fp* (void * buffer, bool save_always)
  - void *llva_load_fp* (void * buffer)

SVA: Memory Safety for OS Kernels



- Use static analysis to prove safe memory accesses
- Use alias analysis (DSA) to group objects into logical pools
- Virtual machine records object allocations in pools
- Run-time checks only check objects in a single pool

Acknowledgements

- LLVA
  - Chris Lattner
  - Michael Brukman
  - Anand Shukla
  - Brian Gaeke
- SVA
  - Dinakar Dhurjati
  - Sumant Kowshik
  - Andrew Lenharth
  - Rob Bocchino
  - Brent Monroe
  - Pierre Salverda
  - David Raila
- Vikram Adve
- The LLVM Community

References (http://llvm.org/pubs)

- SVA
  - MICRO ‘03
  - VEE ‘06
  - WIOSCA ’06
- Data Structure Analysis (DSA)
  - PLDI ’07
- Automatic Pool Allocation
  - MSP ‘02
  - PLDI ’05
  - Lattner PhD Thesis
- Memory Safety
  - LCTES ‘03
  - TECS ’05
  - CASES ‘02
  - Dhurjati PhD Thesis

Questions?