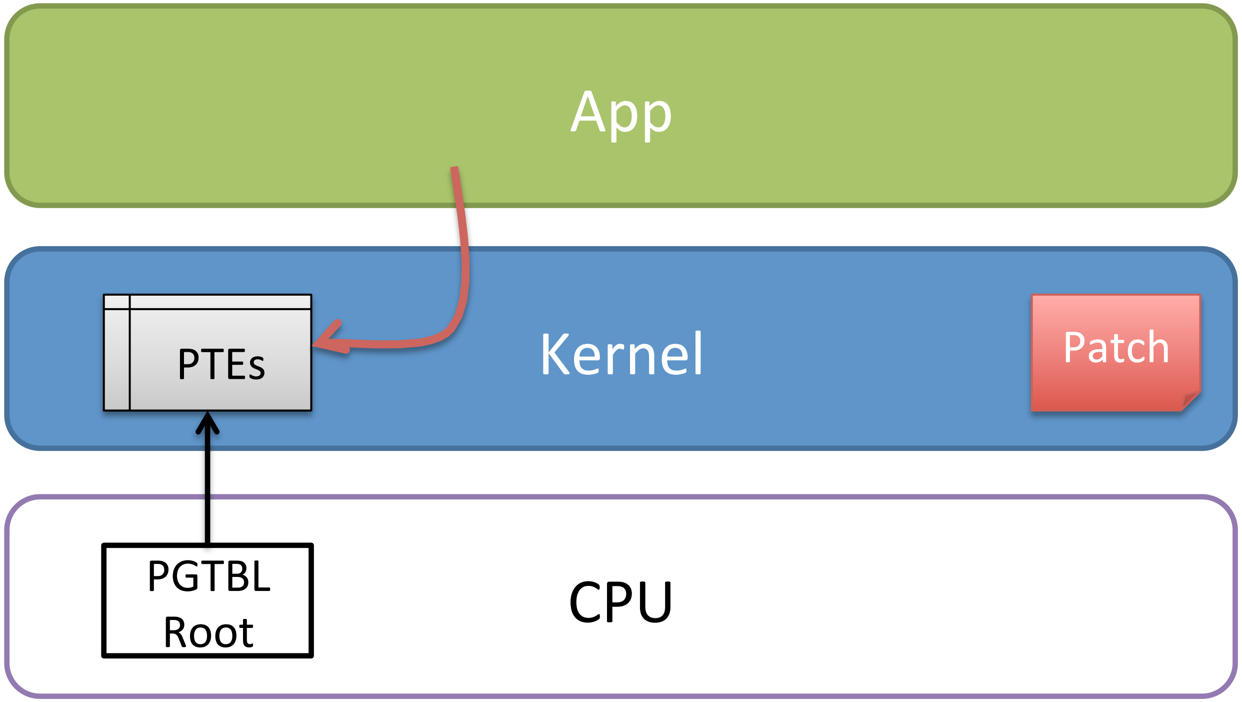
Dune: Safe User-level Access to Privileged CPU Features

Adam Belay, Andrea Bittau, Ali Mashtizadeh, David Terei, David Mazières, and Christos Kozyrakis
Stanford University

The power of privilege

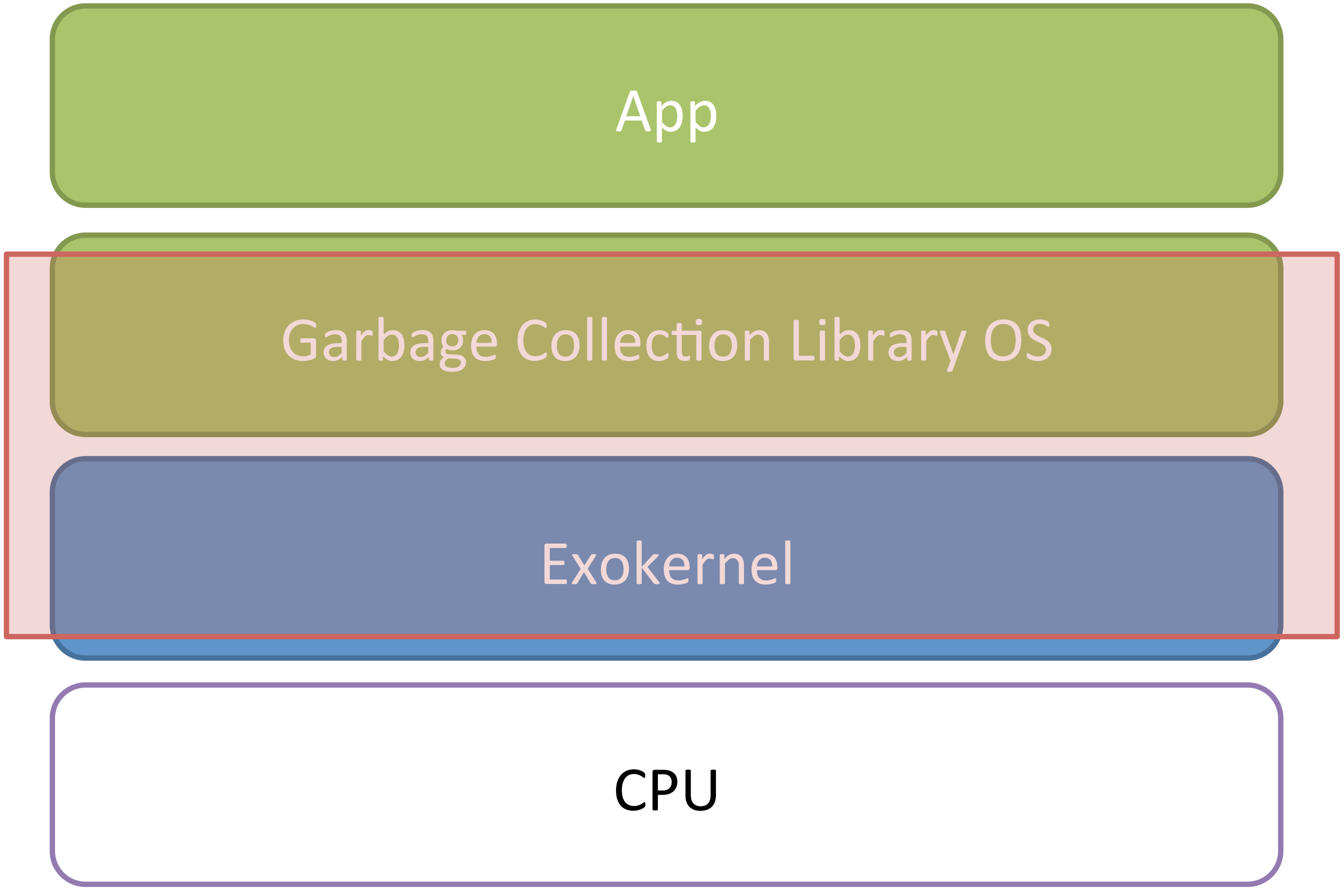
- Privileged CPU features are fundamental to kernels
- But other, compelling uses:
  - Speed up garbage collection (Azul C4)
    - Page tables provide memory access information
  - Privilege separation within a process (Palladium)
    - MMU hardware isolates compartments
  - Safe native code in web browsers (Xax)
    - System call handler intercepts system calls

Should we change the kernel?



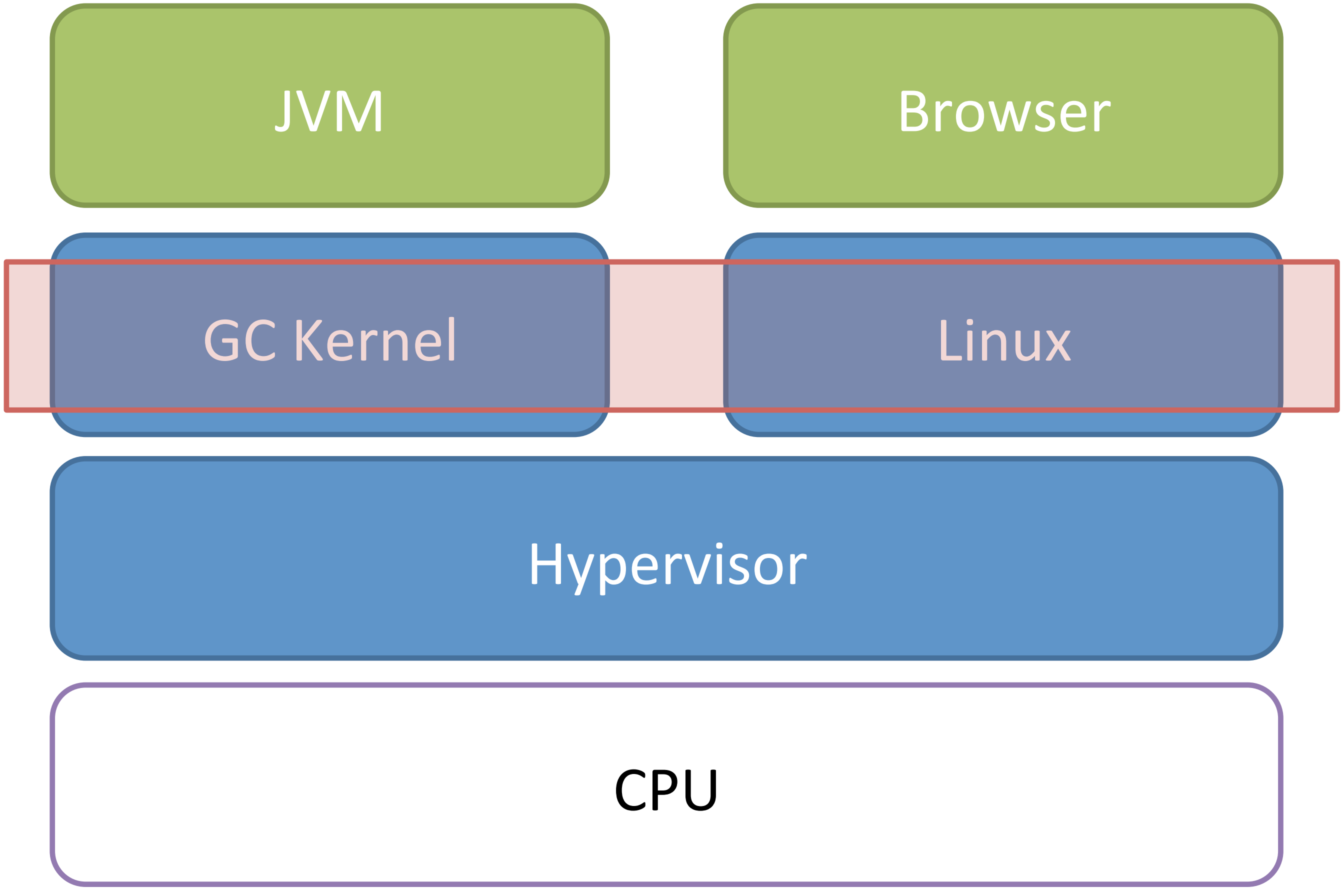
- Problem: stability concerns, challenging to distribute, composability concerns

What about an Exokernel?



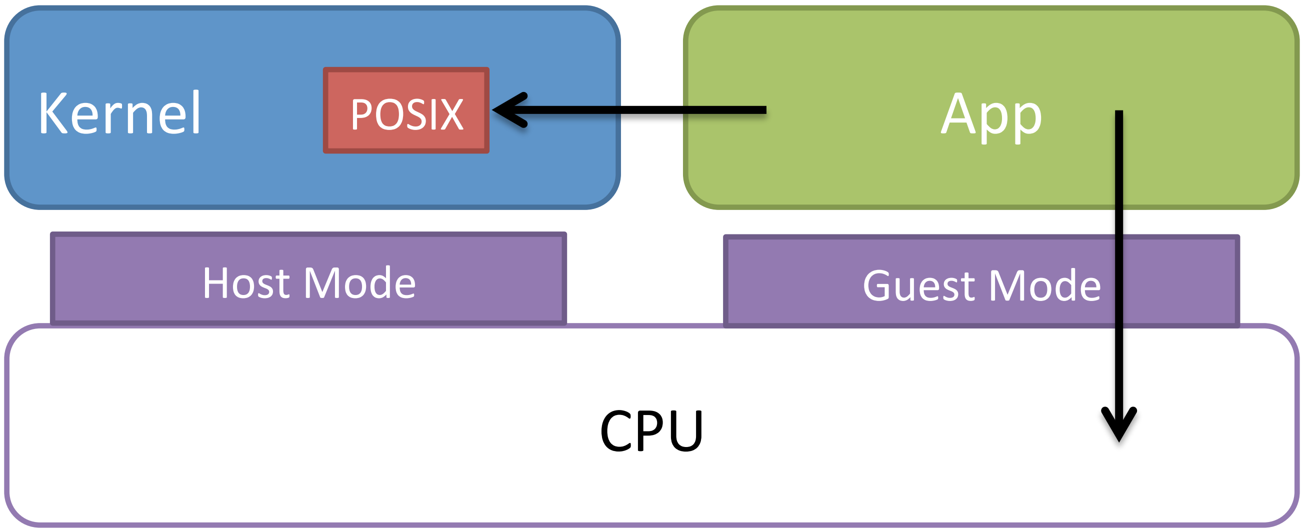
- Problem: must replace entire OS stack

What about a virtual machine?



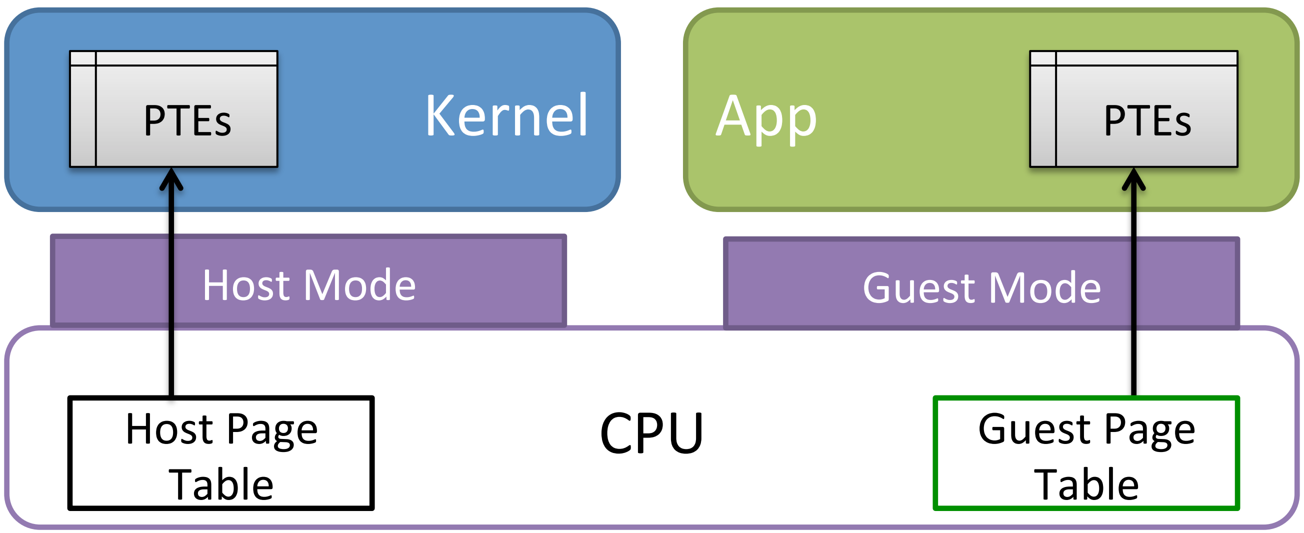
- Problem: virtual machines have strict partitioning

Dune in a Nutshell



- Provide safe user-level access to privileged CPU features
- Still a normal process in all ways (POSIX API, etc)
- Key idea: leverage existing virtualization hardware (VT-x)

Garbage collection in Dune



- Solution: control the page table directly within a process

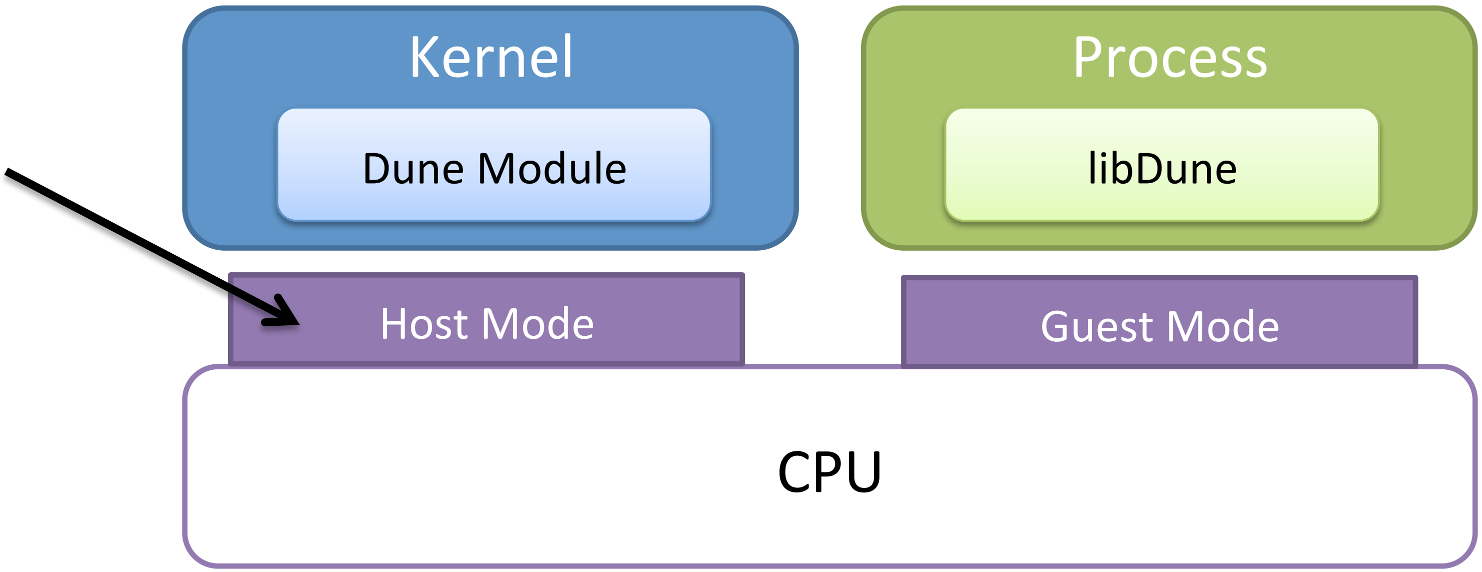
Outline

- Overview
- **Design**
- Evaluation

Available CPU features

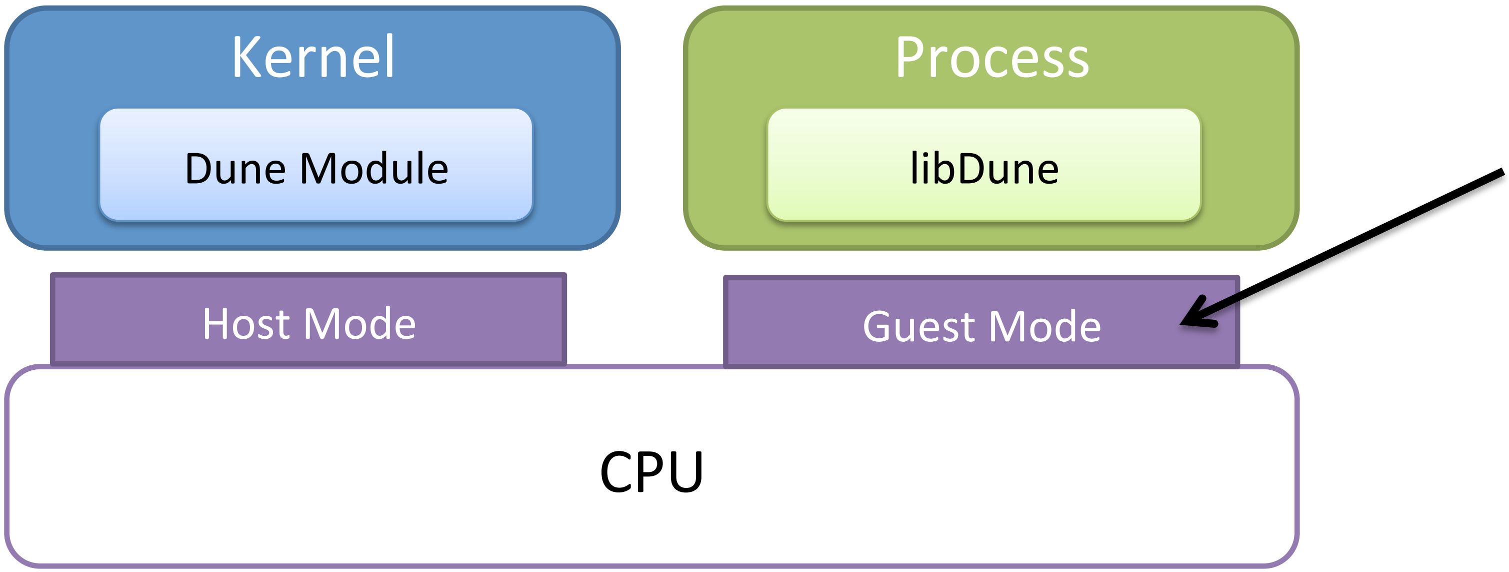
- Privilege Modes
  - SYSRET, SYSEXIT, IRET
- Virtual Memory
  - MOV CRn, INVLPG, INVPCID
- Exceptions
  - LIDT, LTR, IRET, STI, CLI
- Segmentation
  - LGDT, LLDT

Dune architecture



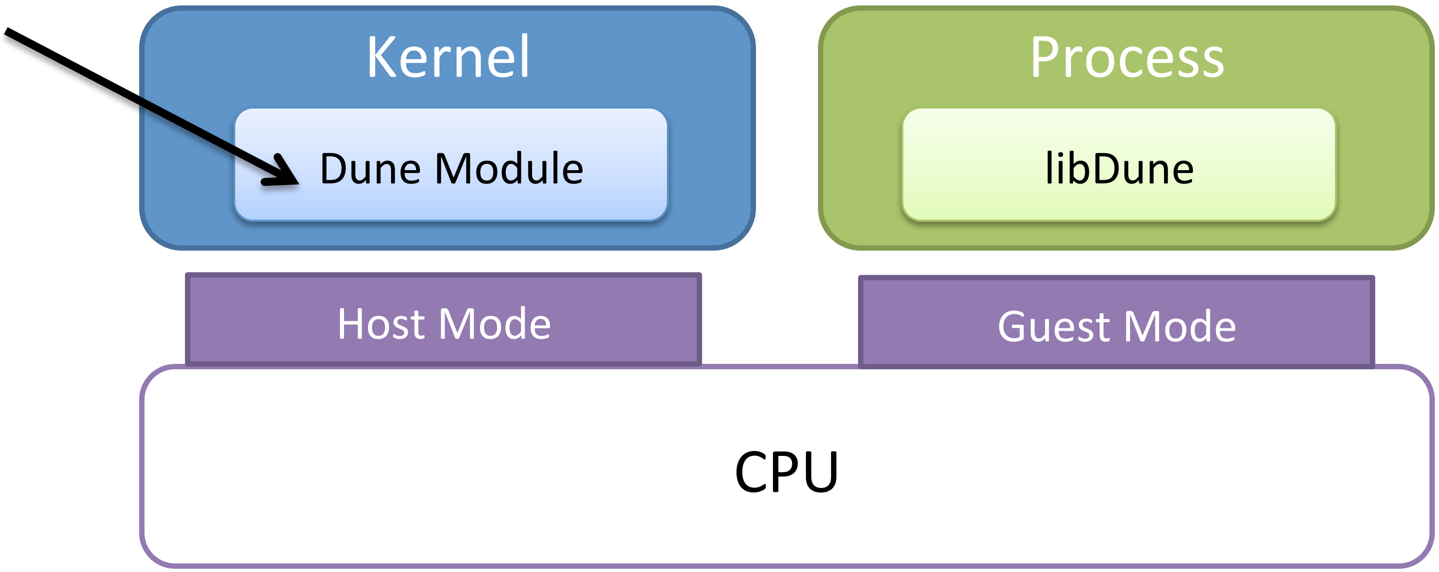
- Host mode -> VMX root mode on Intel
- Normally used for hypervisors
- In Dune, we run the kernel here
  - Reason: need access to VT-x instructions

Dune architecture



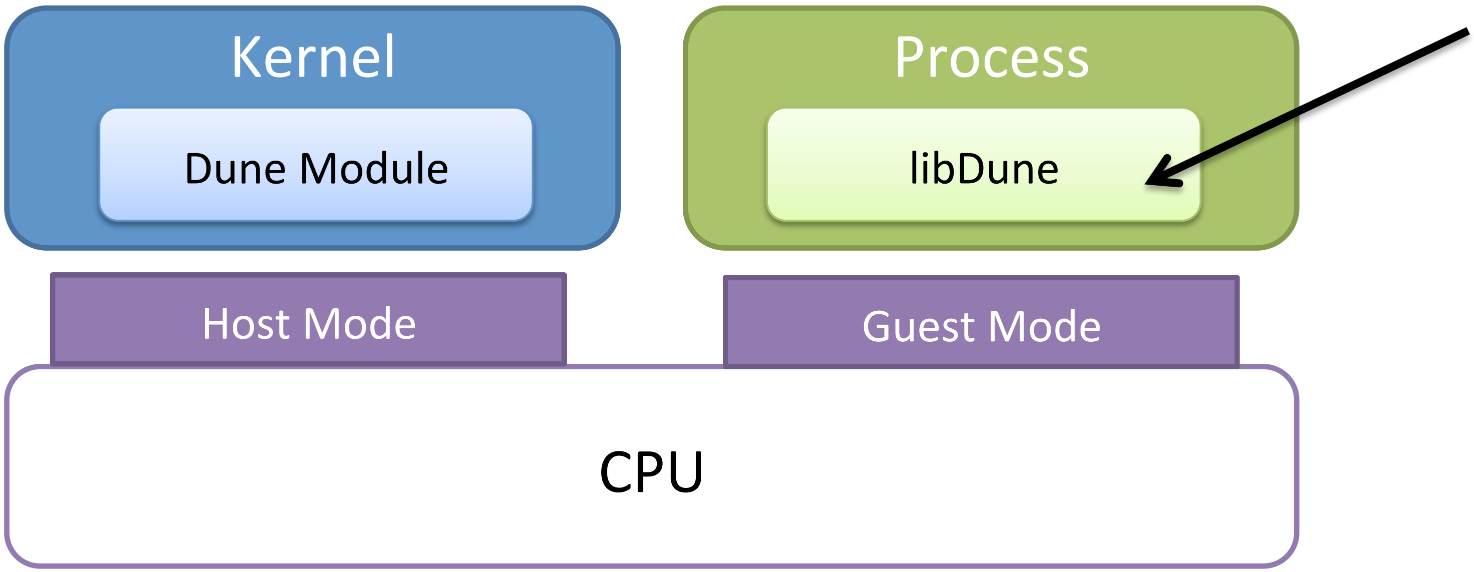
- Guest mode -> VMX non-root mode on Intel
- Normally used by the guest OS
- In Dune, we run ordinary processes here
  - Reason: need access to privileged features

Dune architecture



- Dune Module (~2500 LOC)
  - Configures and manages virtualization hardware
  - Provides integration with the rest of the kernel in order to support a process abstraction
  - Uses Intel VT-x (could easily add AMD SVM)

Dune architecture



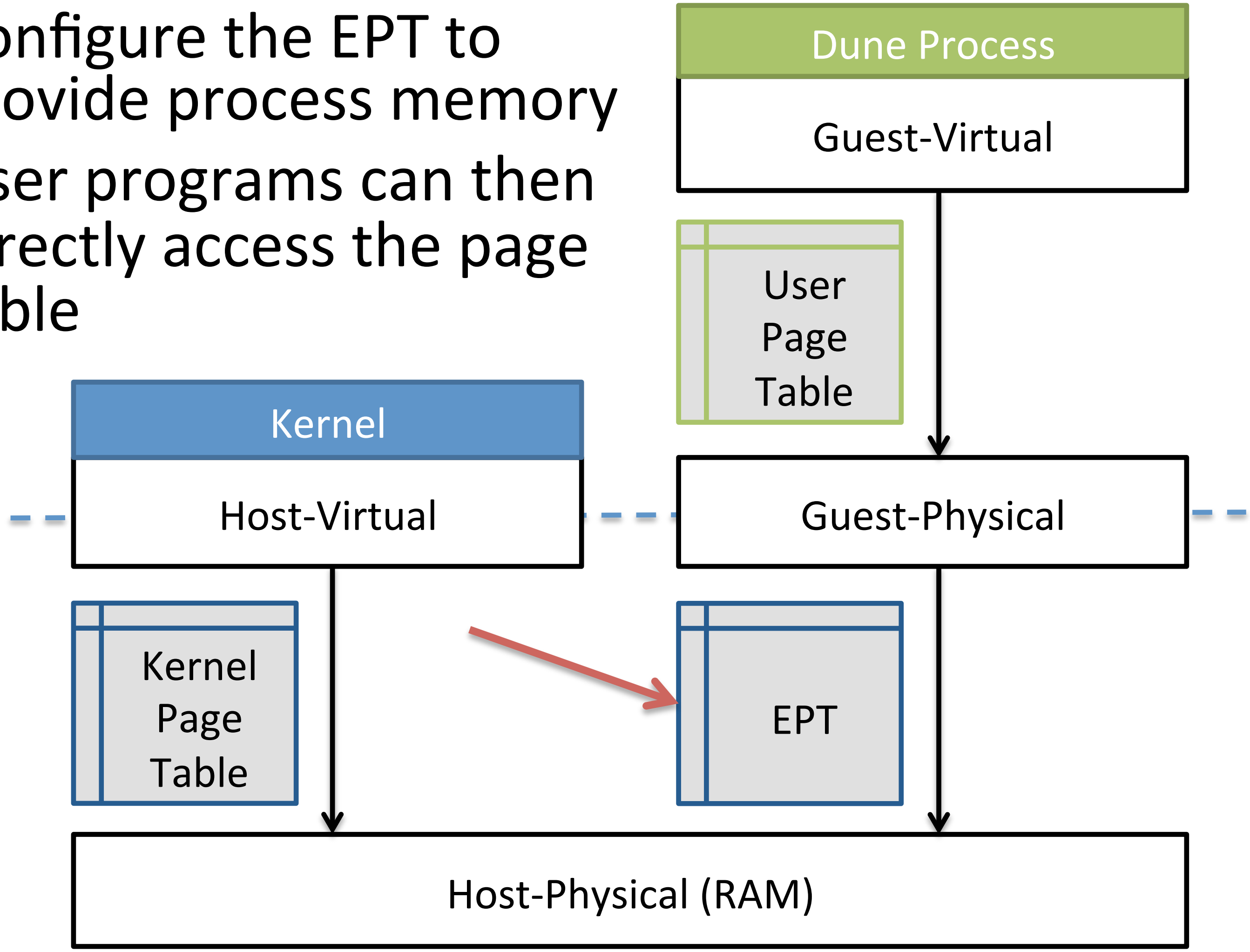
- libDune (~6,000 LOC)
  - A utility library to help applications manage privileged hardware features
  - Completely untrusted
  - Exception handling, system call handling, page allocator, page table management, ELF loader

Providing a process abstraction

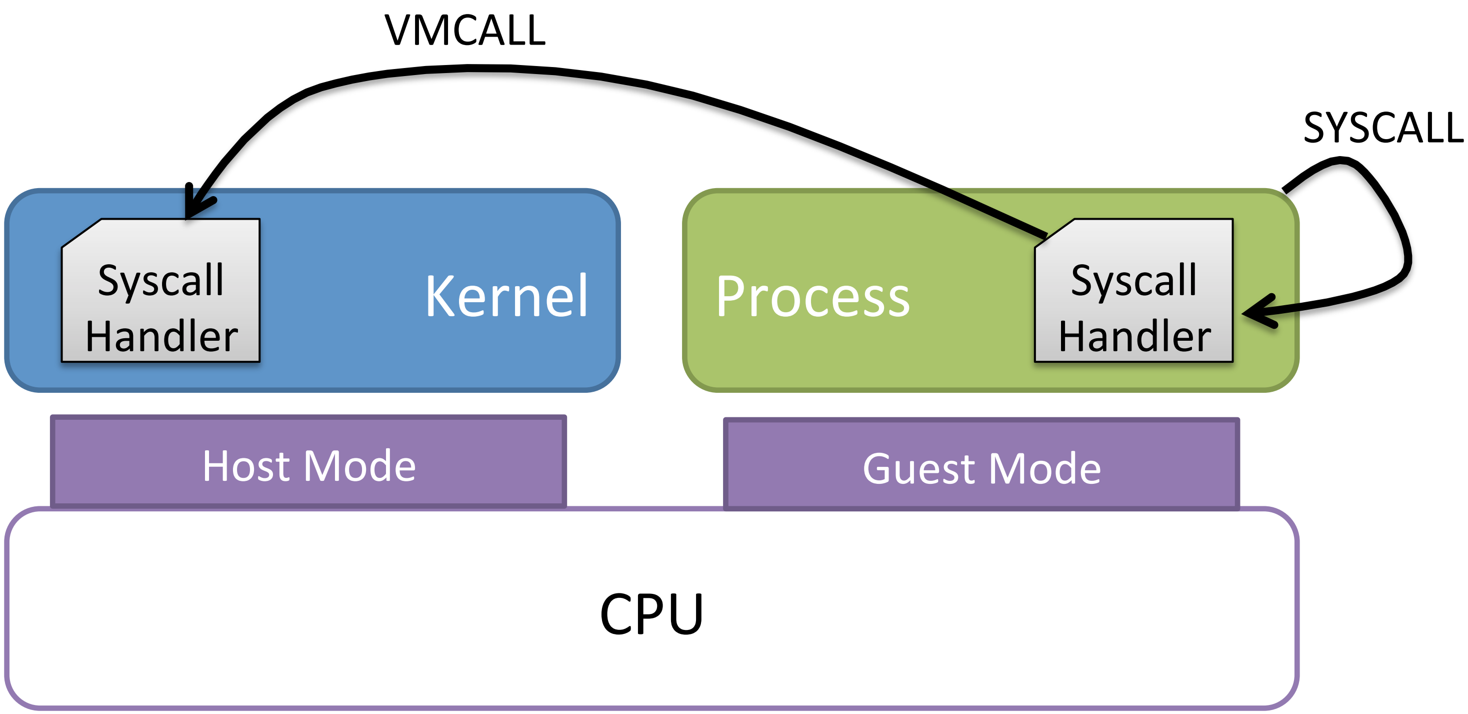
- Memory management
- System calls
- POSIX Signals

Memory management in Dune

- Configure the EPT to provide process memory
- User programs can then directly access the page table

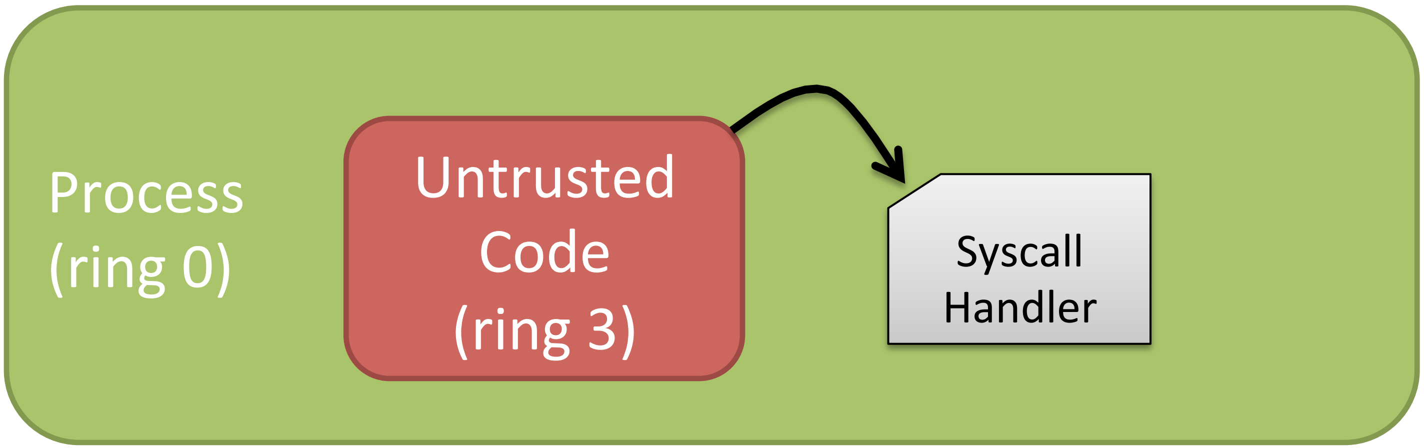


System calls in Dune



- `SYSCALL` will only trap back into the process
- Use `VMCALL` (i.e. a hypercall) to perform normal kernel system calls

But SYSCALL is still useful



- Isolate untrusted code by running it in a less privileged mode (i.e. ring 3 on x86)
- Leverage the 'supervisor' bit in the page table to protect memory

Signals in Dune

- Signals should only be delivered to ring 0
- What happens if process is in ring 3?
- Possible solution: have the Dune module manually transition the process to ring 0
  - Works but slow and somewhat complex
- Our solution: deliver signals as injected interrupts
  - Hardware automatically switches to ring 0
  - Can use CLI and STI to efficiently mask signals

Many implementation challenges

- Reducing VM exit and VM entry overhead
- Pthread and fork were tricky to integrate with the Linux kernel
- EPT does not support enough address space
- Check the paper for details

Outline

- Overview
- Design
- **Evaluation**

Evaluation

- How much overhead does Dune add?
- What potential does Dune create for optimization?
- What is Dune's performance in end-to-end use cases?

Overhead analysis

- Two sources of overhead
  - VMX transitions
  - EPT translations

(cycles)	Getpid	Page fault	Page walk
Linux	138	2,687	36
Dune	895	5,093	86

Optimization analysis

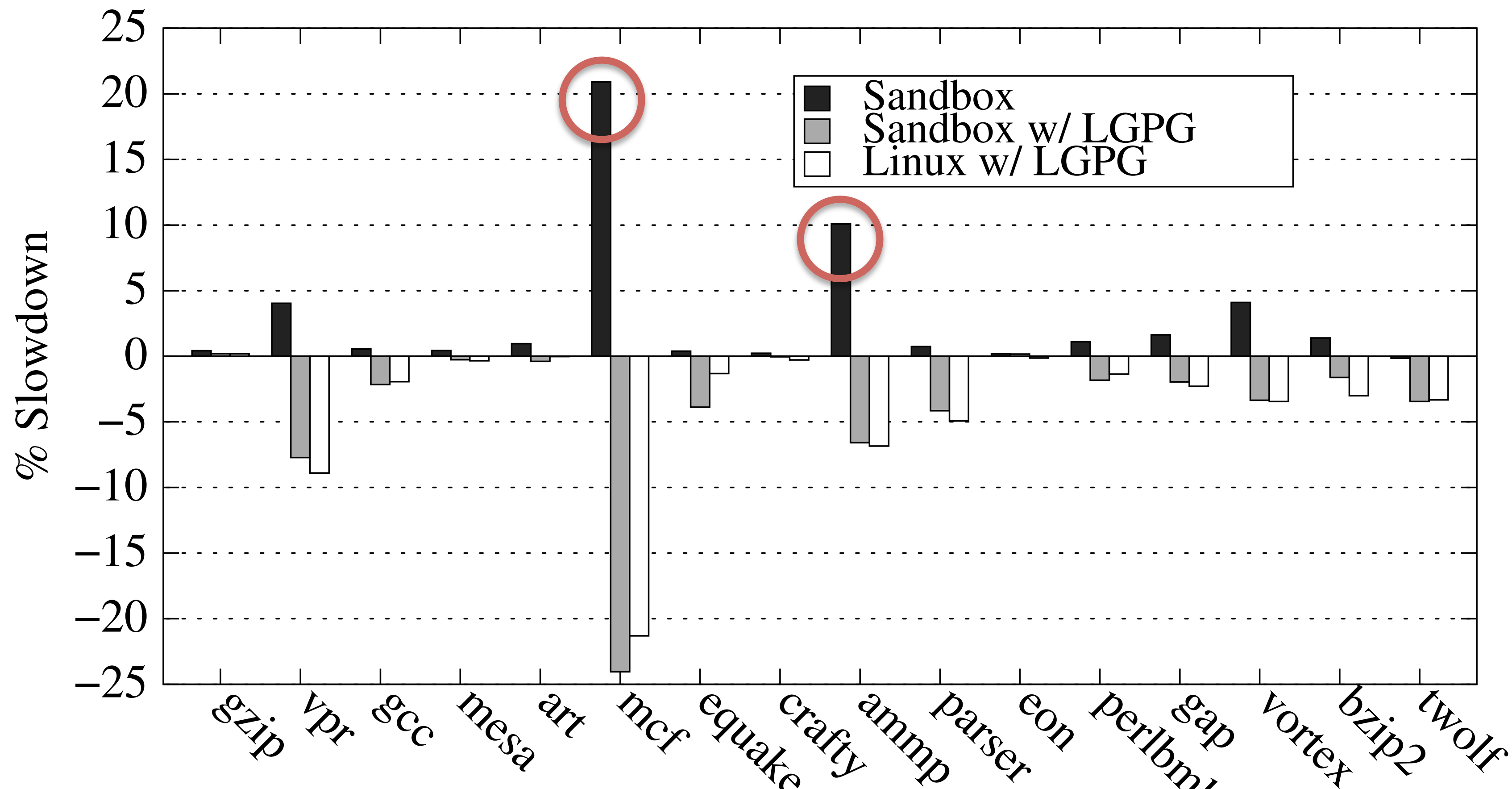
- Large opportunities for optimization
  - Faster system call interposition and traps
  - More efficient user-level virtual memory manipulation

(cycles)	ptrace (getpid)	trap	Appel 1 (TRAP, PROT1, UNPROT)	Appel 2 (PROTN, TRAP, UNPROT)
Linux	27,317	2,821	701,413	684,909
Dune	1,091	587	94,496	94,854

End-to-end case studies

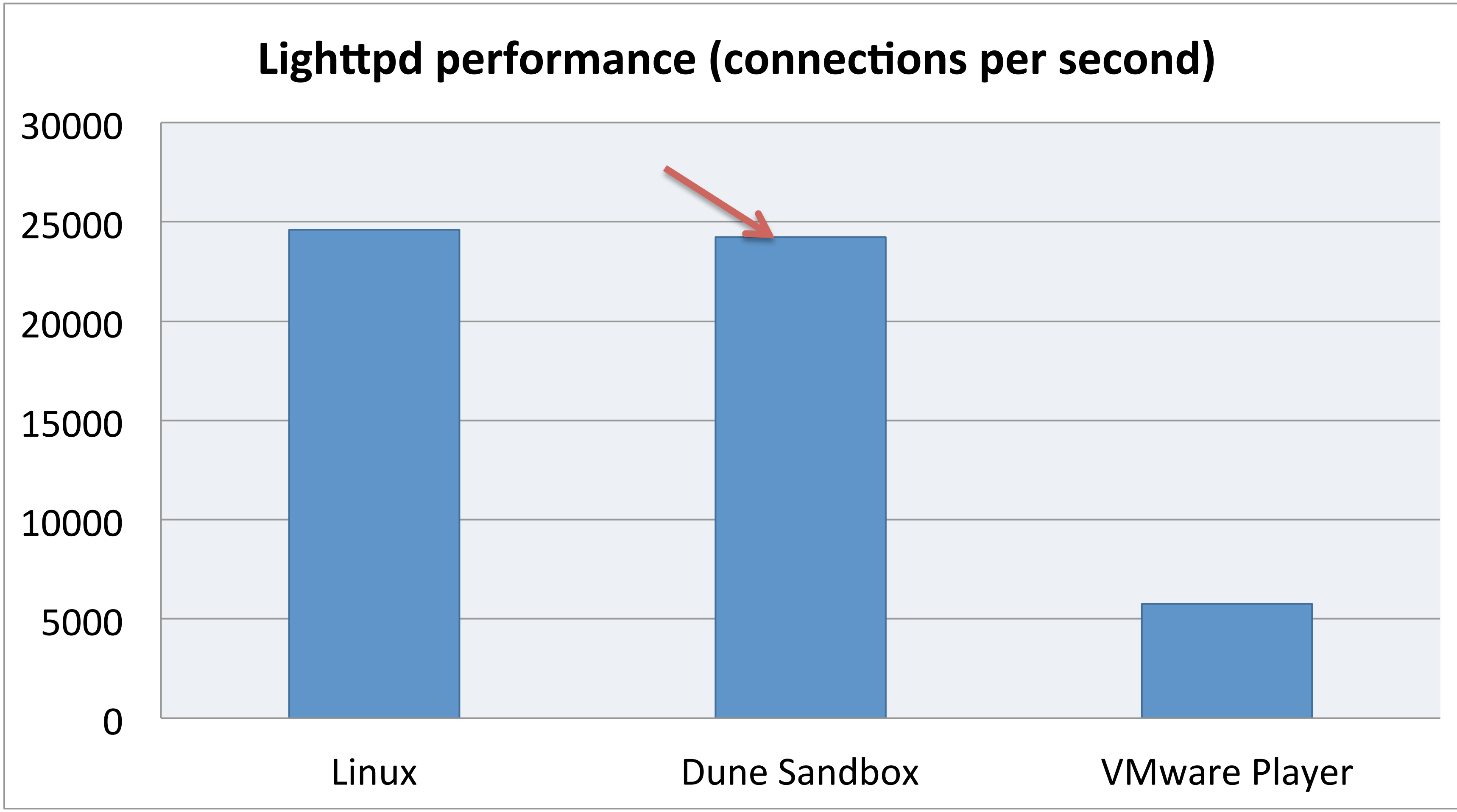
- We built and evaluated three systems
- Application sandbox (~1300 LOC)
  - Constrained the system calls performed by an untrusted binary
- Garbage collection (less than 100 LOC change)
  - Improved dirty page detection through direct access to dirty bits
- Privilege separation (~750 LOC)
  - Supported several protection domains within a single process through use of multiple page roots (with TLB tagging)

Sandbox: SPEC2000 performance



- Only notable end-to-end effect is EPT overhead
- Can be eliminated through use of large pages

Sandbox: lighttpd performance



- Slight reduction in throughput (less than 2%) due to VMCALL overhead

Performance of other use cases

- Up to 40% improvements in garbage collection performance (less than 100 LOC)
- Privilege separation system can context switch between subdomains 3x faster than Linux can switch between processes (750 LOC)

Conclusions

- Applications can benefit from access to privileged CPU features
- Virtualization hardware allows us to provide such access safely
- Dune creates new opportunities to build and improve applications without kernel changes
- Dune has modest performance overhead
- Download Dune at http://dune.scs.stanford.edu