11/13 CS240 - Dune

Announcements

For next class (Tuesday 11/18)

1. Read: Salus: Fine-Grained GPU Sharing Primitives for Deep Learning Applications
2. No reading questions

Paper

- Dune: Safe User-level Access to Privileged CPU Features
  - OSDI 2012: 10th USENIX Symposium on Operating Systems Design and Implementation
  - Two of the graduate students worked at VMware before grad school
  - The lead author, Adam Belay, was a Stanford graduate student now MIT prof

Paper Background

- Protected mode OSes limit access to privileged CPU instructions and data
- Some examples of advantages from having such access:
  - Fancy virtual memory functionality
    - Speed up garbage collection
    - Process migration
    - Distributed shared memory, and many more - Appel & Li 1991
  - Privilege separation
  - Sandboxing
- Can do these things on Linux but really slow

Existing Approaches

- Why not extend Linux to give access to privilege state?
- Why not Exokernels - OS safely exports hardware?
- Why not virtual machine monitors - Give private copy of privileged mode?

x86 privileged mode

Exceptions:
- LIDT: Load table saying where to jump on which exceptions/interrupts
- LTR: Specifies stack pointers on switch to more privileged mode
- IRET: Return from interrupt/exception (possibly lowering privilege)
- STI, CLI: enable/disable interrupts

Virtual memory:
- MOV %CRn: cr3 = page table base register (PML4 base, optional PCID),
  cr2 = page fault linear address,
  cr0, cr4 = various flags (e.g., PCID)
- INVLPG: knock single page out of TLB
- INVPCID: knock all entries matching PCID out of TLB

Privilege modes:
- SYSRET, SYSEXIT: return from system call
- IRET: return from trap/exception

Segmentation: LGDT, LLDT

VT-x: VMLAUNCH/VMRESUME/VMCALL - Run a VMCS; EPT - Extended page tables

Dune approach

- VMLAUNCH a thread in a mirror of the Linux address space
  - Open /dev/dune, do `ioctl` to register current CR3 to mirror
- System calls reissued back on the Linux side

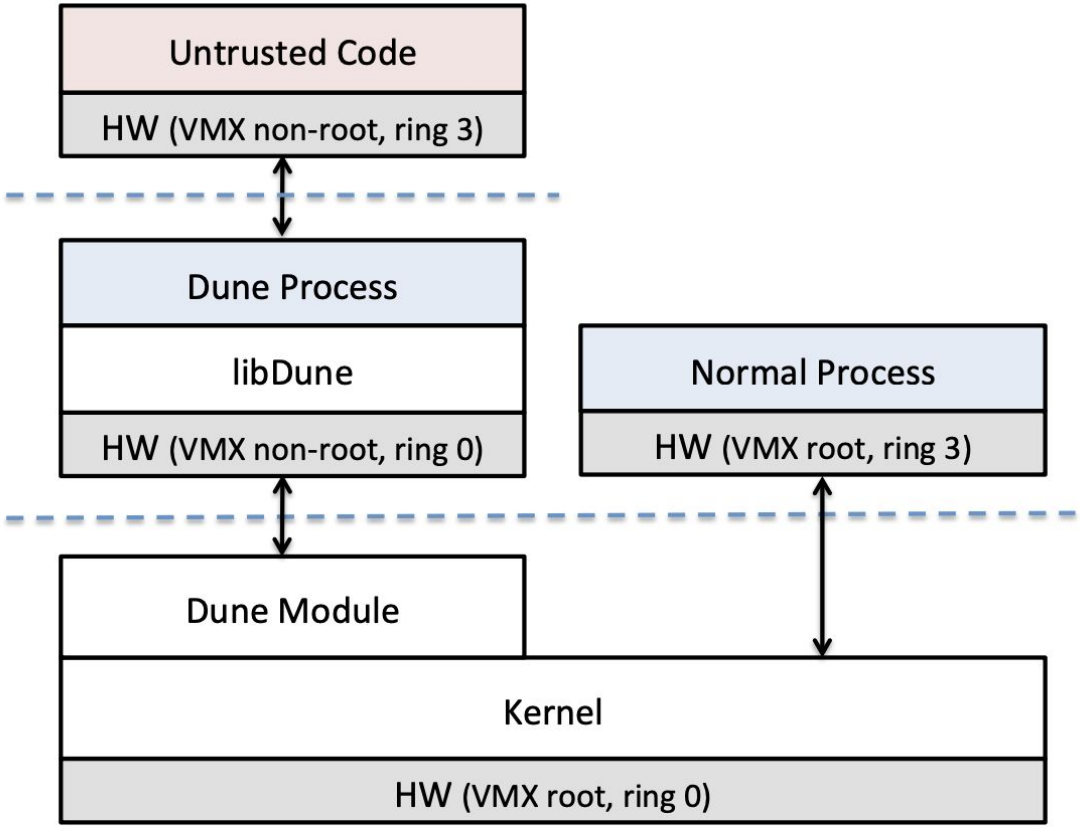


Figure 1: The Dune system architecture.

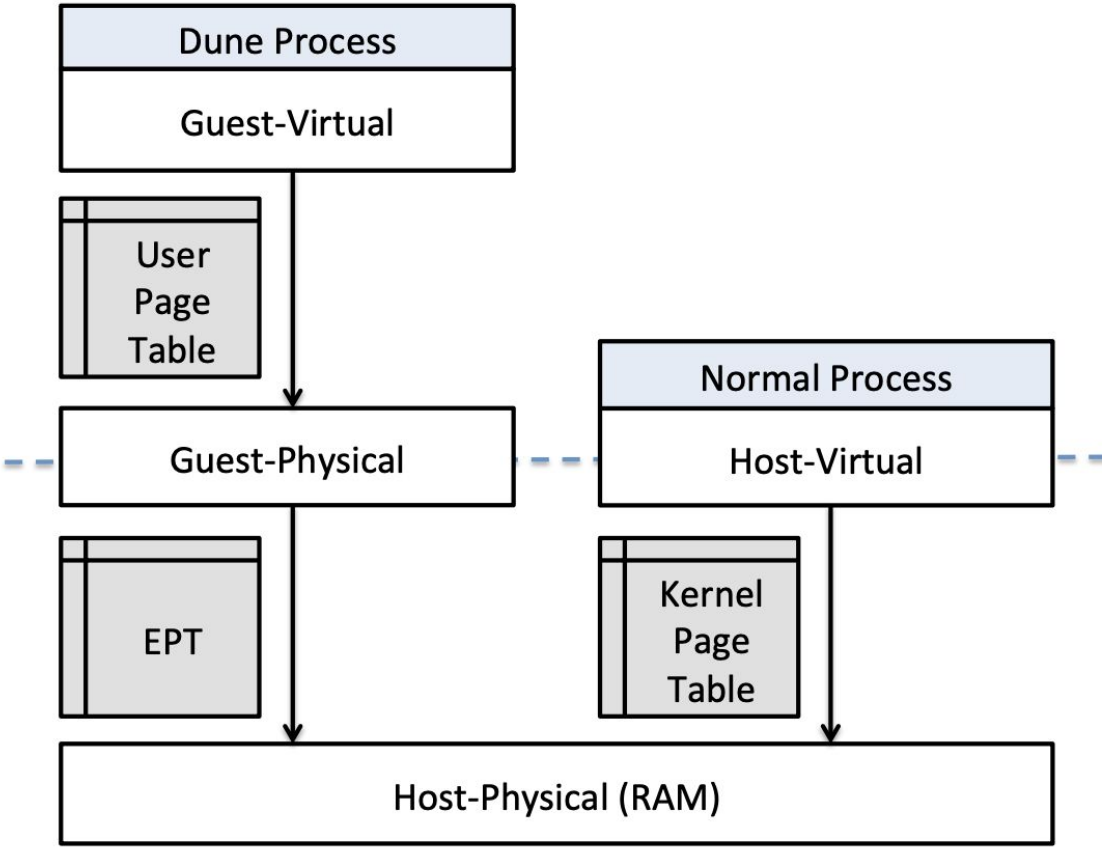


Figure 2: Virtual memory in Dune.

Dune process vs Linux process differences

- What is faster in a Dune Process?
  - Page table reads and writes
  - Trap handling
- What is slower in a Dune Process?
  - System calls (SYSCALL, VMCALL)
  - TLB misses

	getpid	page fault	page walk
Linux	138	2,687	35.8
Dune	895	5,093	86.4

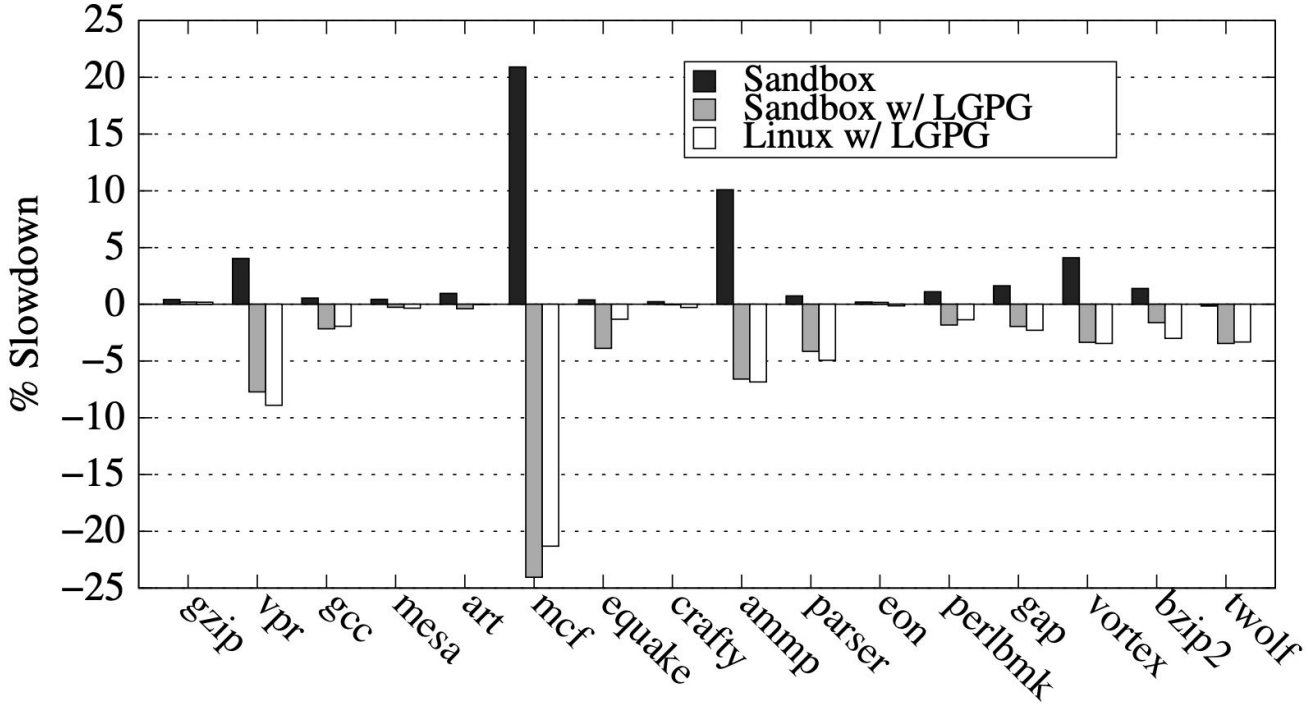
	ptrace	trap	appel1	appel2
Linux	27,317	2,821	701,413	684,909
Dune	1,091	587	94,496	94,854

- Dune has limited access to privileged registers and state, Linux none
  - CR0.PG (paging) CR0.EM (FPU)
  - Only limited page table changes allowed if system calls must work

Applications

- Sandbox
- Wedge
- Garbage collection

Sandbox slowdown on SPEC2000



Sandbox isolation compared with hosted VMM

	1 client	100 clients
Linux	2,236	24,609
Dune sandbox	2,206	24,255
VMware Player	734	5,763

Table 4: Lighttpd performance (in requests per second).

Wedge performance

- Microbenchmark: 3x to 40x faster. System: 22% faster

	create	ctx switch	http request
fork	81	0.49	454
Dune sthread	2	0.15	362

Table 5: Wedge benchmarks (times in microseconds).

	GCBench	LinkedList	HashMap	XML
Collections	542	33,971	161	10
Memory use (MB)				
Allocation	938	15,257	10,352	1,753
Heap	28	1,387	27	1,737
Execution time (ms)				
Normal	1,224	15,983	14,160	6,663
Dune	1,176	16,884	13,715	7,930
Dune TLB	933	14,234	11,124	7,474
Dune dirty	888	11,760	8,391	6,675

Table 6: Performance numbers of the GC benchmarks.

Trust in hardware protection?

Why wasn't the Dune approach picked up?