Xen/paravirt_ops upstreaming
Jeremy Fitzhardinge
XenSource
jeremy@xensource.com
Where Did We Come From?

- Initially arch/xen
- Moved to a i386 subarch
- Upstreaming stalled
- VMI posted
- Deadlock
paravirt_ops

• KS 2006: Rusty proposed paravirt_ops
• General kernel interface to hypervisors
• Standard Linux-style source-level API
  – Not an ABI
• Back-ends for each supported hypervisor
• Runtime selection
  – Single kernel runs native, Xen, VMI, etc
• Simplified initial implementation
  – i386 only
  – domU only
  – Shadow mode only
  – Uniprocessor only
  – Netfront
  – Blockfront
Xen/paravirt_ops

- Simplified initial implementation
  - i386 only
  - domU only
  - Shadow mode only
    - Writable pagetables + late pinning
  - Uniprocessor only
    - SMP
  - Netfront
  - Blockfront
Xen/paravirt_ops

- Simplified initial implementation
  - i386 only
  - domU only
  -Writable pagetables + late pinning
  - SMP
  - Netfront
  - Blockfront
  ✔ Dynticks
  ✔ preempt
More details

• Adds core Xen functionality for:
  • Hypervisor interface
  • Time
  • Reboot
  • Event channels
  • Grant tables
  • Xenbus
  • Console
  • Frontend block and net drivers
Upstreaming state

- Goal is to get it into 2.6.22
- Almost all prerequisite paravirt_ops patches are queued
- Xen patches will be queued this week
- Waiting for 2.6.21 to get out of the way...
- Need testers!
  - Look for announcement on xen-devel
Some Statistics

• 420 patches so far
  – Many dead-ends

• 45 in committed to git

• 54 queued with Andi Kleen

• 58 still outstanding
  – 28 general kernel, paravirt and vmi patches
  – 30 Xen: 90 files changed, 14219 insertions(+), 31 deletions(-)
Design Goals

• Clean interface to rest of kernel
• Use existing interfaces where possible
  – Or bend something into shape
• Need to support other hypervisors
• Need to support runtime selection
• High efficiency
• Defensible design
<asm/paravirt.h>

- struct paravirt_ops contains:
  - Patching
  - Setup, pagetable init, initial time-of-day, banner
  - Privileged instructions
  - Interrupt control
  - segments/gdt
  - tlb flushing
  - Pagetable alloc
  - pgd/pmd/pte get/set
  - Batching
• **struct paravirt_ops contains:**
  - Patching
  - Setup, pagetable init, initial time-of-day, banner
  - Privileged instructions
  - Interrupt control
  - segments/gdt
  - tlb flushing
  - Pagetable alloc
  - pgd/pmd/pte get/set
  - Batching

Could probably do with splitting out...

- cpu_ops
- pagetable_ops
Clean kernel interface

• Consistent high-level interfaces
  – Rather than a set of random hooks

void (*apic_write)(unsigned long reg, unsigned long v);
void (*apic_write_atomic)(unsigned long reg, unsigned long v);
unsigned long (*apic_read)(unsigned long reg);
void (*setup_boot_clock)(void);
void (*setup_secondary_clock)(void);

void (*startup_ipi_hook)(int phys_apicid,
                        unsigned long start_eip,
                        unsigned long start_esp);
Clean kernel interface

- Consistent high-level interfaces
  
  Rather than a set of random hooks

```c
void (*apic_write)(unsigned long reg, unsigned long v);
void (*apic_write_atomic)(unsigned long reg, unsigned long v);
unsigned long (*apic_read)(unsigned long reg);
void (*setup_boot_clock)(void);
void (*setup_secondary_clock)(void);

void (*startup_ipi_hook)(int phys_apicid,
                         unsigned long start_eip,
                         unsigned long start_esp);
```
Clean kernel interface

• Consistent high-level interfaces
  – Rather than a set of random hooks

```c
struct smp_ops {
    void (*smp_prepare_boot_cpu)(void);
    void (*smp_prepare_cpus)(unsigned max_cpus);
    int (*cpu_up)(unsigned cpu);
    void (*smp_cpus_done)(unsigned max_cpus);
    
    void (*smp_send_stop)(void);
    void (*smp_send_reschedule)(int cpu);
    int (*smp_call_function_mask)(cpumask_t mask, void *
};
```
Use existing interfaces

- Clock infrastructure dramatically simplified Xen interface
- No core changes; just a clock driver

$ ls -lL xen/unstable/linux-2.6.18-xen/arch/i386/time*.c
  -rw-rw-r-- 1 jeremy jeremy 29617 Apr  6 13:58 time-xen.c
  -rw-r--r-- 3 jeremy jeremy 9548 Sep 19  2006 time.c

$ ls -l xen/paravirt/linux/arch/i386/xen/time.c
  -rw-rw-r-- 1 jeremy jeremy 12759 Apr 14 10:02 time.c
Use existing interfaces (2)

- **Console**
  - Complete xen-console code is 860 lines
  - Xen console driver for hvc-console is 160 lines

- **Interrupts**
  - Register “xen_irq_chip”
  - Inject interrupts into top of normal interrupt handling

- **Bend existing interfaces**
  - Use paravirt-ops interfaces, even when not compiled for paravirt-ops
Support other hypervisors

• Make a single pv_op work as hard as possible
• CONFIG_HIGHPTE
  – Xen needs to map all pagetables RO
  – VMI needs to inform hypervisor of PTE mappings
  – kmap_atomic_pte serves both
    • Xen can map RO
    • VMI can inform hypervisor of mapping
Support runtime selection

• Kernel can be compiled to support multiple hypervisors

• At boot
  – Work out which hypervisor
  – Install appropriate paravirt_ops

• No config-time limitations
  – Anything unsupported must be disabled at runtime
High efficiency

- Minimal effect on native execution
- Low overhead compared to dedicated Xen/VMI/etc kernel
- Lazy updates (batching)
- Binary patching
  - Inline small instructions (sti, cli, etc)
  - Convert indirect -> direct calls for rest
  - Nop out calls to no-op functions
Binary patching

call *paravirt_ops.irq_disable
ff 15 1c 80 42 c0

native
cli; nop; nop; nop; nop; nop
fa 90 90 90 90 90

Xen
call xen_irq_disable; nop
e8 05 3d 0e 00 90

Xen, vcpu placement
movb $0x1,%fs:0xc0497221
64 c6 05 21 72 49 c0 01
Misleading Graphs!

**Microbench**

- Null call
- Null I/O
- Stat
- Open clos
- Select TCP
- Sig inst
- Sig hndl

**Fork/exec timing**

- Fork proc
- Exec proc
- Sh proc

Legend:
- Non-paravirt
- Paravirt
- Paravirt nopatch
- Paravirt-patch
Lazy updates

- Interface adds lazy_mmu and lazy_cpu modes
  - lazy_mmu_mode
    - Batches pagetable updates
    - Useful for munmap/mprotect
  - lazy_cpu_mode
    - Batches context switches (5->1)
- Maps directly to Xen multicalls
Defensible Design

- Anything I can defend in the court of lkml
Defensible Design

- Anything I can defend in the court of lkml

On Fri, Feb 16, 2007 at 02:21:07PM -0800, William Lee Irwin III wrote:

> The amount of **violence** this patch manages
> to commit is phenomenal for what little it actually
> does.
Jeremy Fitzhardinge <jeremy@goop.org> wrote:
> You're arguing that we should have a single hypervisor ABI in order to,
> among other things, reduce the test matrix, and yet the ABI is entirely
> defined by testing to see how well a given implementation runs some
> random version of Linux. And if Linux wants to use that interface in a
> different way, everyone is supposed to magically keep up.
>
> And all this is supposed to be managed by multiple disparate independent
> out-of-tree implementations?
>
> Yep, I want a pony too.
Defensible Design

- Anything I can defend in the court of lkml
Defensible Design

- Anything I can defend in the court of lkml
- Some thing easier to implement than argue
  - SMP (had to be done anyway)
  - Preempt (ugly but popular)
- Lots of opportunities for outright cleanups
  - Time
  - High-level SMP interface
  - Interrupts
Changes to Xen

- Improvements to timer interface
  - Allow 100Hz tick to be disabled
  - Error when setting timer in the past
- VCPU structure placement
  - Allows direct access to VCPU via %fs
  - Eliminates need for preempt-disable in many cases
  - Code becomes small enough for inlining
- Don't touch %gs on hypervisor entry
- All changes help, but none are necessary
Future work

- Dependent time
- suspend/resume/migrate
  - Integrate with existing suspend/resume
- Balloon
  - Implement as hotplug memory?
- x86-64
  - Waiting to see how i386-x86_64 arch merge goes
- bzImage domain builder
- dom0
Questions?