クラウドコンピューティングと要素技術

Technologies behind Cloud Computing

ソフトウェア・クラウド開発プロジェクト実践!

浅井大史
2015年4月17日
Cloud Service Models: X as a Service

• **SaaS**
  - Software as a Service
    - Google Apps, Gmail etc.

• **PaaS**
  - Platform as a Service
    - Google App Engine, heroku etc.

• **IaaS**
  - Infrastructure as a Service (a.k.a. Hardware as a Service)
    - Virtual machine, storage, network
    - Google Compute Engine, Amazon EC2 etc.

http://mail.google.com/
http://www.google.com/apps/
http://developers.google.com/appengine/
http://heroku.com/
http://cloud.google.com/products/compute-engine
http://aws.amazon.com/ec2/
COMPUTER ARCHITECTURE AND OPERATING SYSTEM
Computer Architecture

NUMA: Non-Uniform Memory Access

- Processor
- Memory controller
- Core
- Core
- Core
- Core
- Interconnect
- DRAM
- Bus
- IOH (I/O Hub)
- PCIe bus
- Direct Media Interface
- PCIe device
- ICH (I/O Controller Hub)
- Peripherals (USB, Keyboard etc.)

※ Latest Intel and AMD processors and chipsets
Instruction Set Architecture (ISA)

• Microprocessor instruction set architecture
  – CISC (Complex instruction set computing)
    • x86, x86-64 (Intel® 64, AMD 64)
  – RISC (Reduced instruction set computing)
    • SPARC, MIPS, POWER/PowerPC, ARM
Instruction Execution Cycle (abstracted)

1. Instruction fetch (IF)
   - Fetch instruction from instruction cache memory
2. Instruction decode (ID)
   - Decode the fetched instruction
   - Select registers corresponding to the instruction
3. Execute (EX)
   - Execute the instruction
4. Memory access (MA)
   - Load/Store access to memory
5. Write-back (WB)
   - Write-back the result of execution to registers
Pipeline

Instruction 1  
IF + ID + EX + MA + WB

Instruction 2  
IF + ID + EX + MA + WB

Instruction 3  
IF + ID + EX + MA + WB

...  

“Pipeline” improves throughput.
Hazards

• Structural hazards
  – due to shared hardware resource

• Data hazards
  – due to data dependencies

• Control hazards
Superscalar

| Instruction 1 | IF | ID | EX | MA | WB |
| Instruction 2 | IF | ID | EX | MA | WB |
| Instruction 3 | IF | ID | EX | MA | WB |
| Instruction 4 | IF | ID | EX | MA | WB |
| Instruction 5 | IF | ID | EX | MA | WB |

Scalar processor: No instruction-level parallelism
Vector: processor: Parallel data processing

\[\rightarrow\] Superscalar: Instruction-level parallelism
### Out-of-Order Execution/Completion

<table>
<thead>
<tr>
<th>Operation</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>r3 ← r1 + r2</td>
<td>IF</td>
</tr>
<tr>
<td>r1 ← r2 + r3</td>
<td>IF</td>
</tr>
<tr>
<td>r6 ← r4 + r5</td>
<td>IF</td>
</tr>
</tbody>
</table>

**In-order execution/completion**

```
IF	ID	EX	MA	WB
```

**Out-of-Order Execution**

```
IF	ID	EX	MA	WB
```

**Out-of-Order Completion**

```
IF	ID	EX	MA	WB
```
Other Technologies

• Cache
  – Write-Through vs. Write-Back
  – Fully Associative vs. Set Associative
  – Coherency (Multicore/Multiprocessor)

• Virtual memory
  – Page table
    • TLB (Translation Lookaside Buffer)

• Simultaneous Multithreading (Hyper-Threading Technology)
Operating System (OS)

- Resource sharing
  - CPU: Process scheduling
  - Memory: Memory management

- Privilege management
  - Not allow to execute privileged instructions to users’ process
    - System call: Dispatch a privileged instruction to kernel
      - e.g., I/O instructions, cli/sti

- Inter-process communication

- etc...
x86/x86-64 Ring Protection

- **Ring 0**: Kernel in BSD/Linux
- **Ring 1**: System call
- **Ring 2**: Userland in BSD/Linux
- **Ring 3**: etc...

- `xor rax,rbx` etc...
- `rdmsr`
- `mov cr0,rax` etc...
Multitask OS: Resource Sharing

Processor

Space

Core #3
Core #2
Core #1
Core #0

Memory

Space

Time
Scheduling Algorithms

• Systems
  – Batch
    • First-Come First-Serve
    • Shortest Job First
    • Shortest Remaining Time Next
  – Interactive
    • Round-Robin Scheduling
    • Priority Scheduling
    • Priority Scheduling w/ Multiple Queue (different quantum)
    • Shortest Process Next
    • Guaranteed Scheduling
    • Lottery Scheduling
    • Fair-Share Scheduling
  – Realtime
Technologies behind IaaS
Technologies behind IaaS

Virtual Network, Software Defined Network (details in next week)

Hypervisor / VMM, Hardware (peripheral) emulations
Technologies behind IaaS

Hypervisor / VMM
- Resource sharing
  - CPU
  - Memory
- Privilege management
- Virtual chipset / controllers
  - PIC/APIC
  - PIT
  - RTC CMOS

Virtual hardware (peripheral emulations)
- IDE/SATA HDD / CD drive
- Ethernet NIC
- Virtual display (video RAM)
- BIOS / UEFI
- etc...
Terminology

• Hypervisor / Virtual Machine Monitor (VMM)
  – A piece software/firmware that runs virtual machines

• Virtual Machine (VM)
  – A computer emulated by software
    • (usually by the assistance of hardware)

• Guest OS
  – An OS that is executed on a virtual machine
Two Types of Hypervisor

Type 1: Native (bare metal) hypervisor

- e.g.,
  - Xen
  - VMware ESX/ESXi (vSphere Hypervisor)

Type 2: Hosted hypervisor

- e.g.,
  - VirtualBox
  - VMware Workstation
  - Linux KVM (sometimes classified as type 1)

Note: Some implementations may implement virtual hardware devices combined with hypervisor.
Two Types of Virtualization

• Full virtualization
  – Support the full set of instructions (of corresponding ISA) in VMs
    • Pros: No modifications to guest OS are required.
    • Cons: Hardware-assistance is required.

• Paravirtualization
  – Support a partial set of instructions (of corresponding ISA) in VMs
    • Pros: Hardware-assistance is not required.
    • Cons: Modifications to guest OS are required.
      – Privileged instructions must be replaced with hypervisor call.
Paravirtualization: x86/x86-64 Ring Protection

- **Ring 0**: Kernel in BSD/Linux
- **Ring 1**: 
- **Ring 2**: 
- **Ring 3**: Userland in BSD/Linux

- **mov cr0, rax**
- **xor rax, rbx**
- **rdmsr**
- **etc...**
Paravirtualization: x86/x86-64 Ring Protection

- **Ring 0**: Hypervisor (Xen)
  - `mov cr0,rax`
  - `xor rax,rbx`
  - etc...

- **Ring 1**: Guest kernel (BSD/Linux)

- **Ring 2**: Guest userland (BSD/Linux)

- **Ring 3**: System call

```
rdmsr
mov cr0,rax
```
Differences between OS and Hypervisor

• Hypervisor / VMM
  – Resource sharing
    • CPU
    • Memory
  – Privilege management
  – Virtual chipset / controllers
    • PIC/APIC
    • PIT
    • RTC CMOS

• Virtual hardware (peripheral emulations)
  – IDE/SATA HDD / CD drive
  – Ethernet NIC
  – Virtual display (video RAM)
  – BIOS / UEFI
  – etc…
Differences between OS and Hypervisor

- **OS**
  - Resource sharing (e.g., scheduler)
  - Privilege management
  - POSIX
  - User management
  - Exclusive control
  - Inter-process communication

- **Hypervisor**
  - Virtual chipset/hardware
Resource Sharing

Space

Processor

Core #3
Core #2
Core #1
Core #0

Space

Memory

Time
Differences between OS and Hypervisor

• Scheduler
  – OS: Process scheduler
    • Blocked: Inter-process communication
  – Hypervisor: VM scheduler
    • Blocked: Not inter-VM; between VM and virtual devices
      → Simpler

• Memory management
  – OS: Variety of allocation sizes (several byte – ~Gbytes)
    • Usual page table entry size: 4KiB/2MiB
    • Complex allocation algorithm
      – Linux: Buddy system (for page allocation), Slab allocator (for smaller size allocation)
  – Hypervisor: ~Gbytes
    → Easier
Hardware-assisted Virtualization Technology

- Intel® VT-x, AMD-V

Generic design of hypervisor with IA Ring protection architecture **without** hardware’s virtualization assist

Generic design of hypervisor with hardware’s virtualization assist (Intel® VT-x)
Live Migration: Pre Copy

1. Clear dirty bit
2. Copy all pages
3. Start copy
4. Copy the dirty pages again

If # of dirty pages below threshold
1. pause the VM
2. copy the remaining pages
3. start the VM at destination
Live Migration: Post Copy

• Read/Write operations
  – Read
    • If the page has already copied† to destination
      – (i) read the page from destination
    • Otherwise
      – (ii) read the page from source and copy it to destination
  – Write
    • (iii) write to destination

(†Copied pages are managed by a bitmap)
“We reject: kings, presidents and voting.
We believe in: rough consensus and running code.”

by David D. Clark

RUNNING(?) CODE OF OPERATING SYSTEM AND HYPervisor
Page Table (x86-64 long mode)

Figure from Intel® 64 and IA-32 Architectures Software Developer’s Manual Vol. 3A 4.5
### Page Table (x86-64 long mode)

<table>
<thead>
<tr>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>M-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XD</th>
<th>Ignored</th>
<th>Rsbd.</th>
<th>Address of PML4 table</th>
<th>Ignored</th>
<th>PML4E: present</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ignored</td>
<td>Rsbd.</td>
<td>Address of page-directory-pointer table</td>
<td>Ignored</td>
<td>PML4E: not present</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td>PAT</td>
<td>PDPTE: 1GB page</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Address of 1GB page frame</td>
<td>PAT</td>
<td>PDPTE: page directory</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Address of page directory</td>
<td>PAT</td>
<td>PDE: page table</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Address of 2MB page frame</td>
<td>PAT</td>
<td>PDE: 2MB page</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Address of page table</td>
<td>PAT</td>
<td>PDE: page table</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Address of 4KB page frame</td>
<td>PAT</td>
<td>PDE: page table</td>
</tr>
</tbody>
</table>

**Reserved**

**Ignored**

**Note:**

1. M is an abbreviation for MAXPHYADDR.

2. Reserved fields must be 0.

---

Figure from Intel® 64 and IA-32 Architectures Software Developer’s Manual Vol. 3A 4.5
Page Table (x86-64 long mode)

/* Create 64bit page table */
pg_setup:
    movl $KERNEL_PGT,%ebx /* Low 12 bit must be zero */
    movl %ebx,%edi
    xorl %eax,%eax
    movl $(512*8*6/4),%ecx
    rep stosl /* Initialize %ecx*4 bytes from %edi */
    /* with %eax */

/* Level 4 page map */
    leal 0x1007(%ebx),%eax
    movl %eax,(%ebx)
/* Page directory pointers (PDPE) */
    leal 0x1000(%ebx),%edi
    leal 0x2007(%ebx),%eax
    movl $4,%ecx
    pg_setup.1:
        movl %eax,(%edi)
        addl $8,%edi
        addl $0x1000,%eax
        loop pg_setup.1
/* Page directories (PDE) */
    leal 0x2000(%ebx),%edi
    movl $0x183,%eax
    movl $(512*4),%ecx
    pg_setup.2:
        movl %eax,(%edi)
        addl $8,%edi
        addl $0x00200000,%eax
        loop pg_setup.2
/* Setup page table register */
    movl %ebx,%cr3
Multitask OS: Context Switch

IRETQ

Lowest address

RSP

Highest address

Direction of pop from stack

Intel® 64 and IA-32 Architectures Software Developer’s Manual Vol. 3A 5.8.5.1
Multitask OS: TSS (Task State Segment)

Figure from Intel® 64 and IA-32 Architectures Software Developer’s Manual  Vol. 3A 7.7

Point to kernel stack (Ring 0)
/* Restart a task */
_task_restart:
 /* If the next task is not assigned, immediately restart */
cmpq $0,(_next_task)
jz 1f
/* Save stack pointer */
movq (_cur_task),%rax
movq %rsp,TASK_RP(%rax)
/* Task switch (set the stack frame of the new task) */
movq (_next_task),%rax
movq %rax,(_cur_task)
movq TASK_RP(%rax),%rsp /* Copy sp0 */
movq $0,(_next_task)
/* ToDo: Load LDT/cr3 */
/* Setup sp0 in TSS */
movq (_cur_task),%rax
movq TASK_SP0(%rax),%rdx
movq (_tss),%rbp
movq %rdx,TSS_SP0(%rbp)
1:
/* Pop registers */
intr_irq_done
iretq
Hypervisor

Virtual Machine Control Structure (VMCS)
- Guest-state area: Registers etc.
- Host-state area: Host state/exit point on every VM exit.
- VM-execution control fields
- VM-exit control fields
- VM-entry control fields
- VM-exit information fields

* Current VMCS per logical processor (of physical machine)
* Active VMCS per virtual processor

1. **VMREAD/VMWRITE/VMCLEAR** to configure current VMCS
2. Setup and prepare peripherals (virtual devices)
3. **VMLAUNCH**
4. Handle the instructions corresponding to VM exits and schedule/switch VMs w/ **VMRESUME**
   - VM exits caused by
     1. instructions of VM
     2. signal by hypervisor’s VMX-preemption timer
     3. **VMCALL** from VM

Details in Intel® 64 and IA-32 Architectures Software Developer’s Manual Vol. 3C Chapter 23
How to Implement Live Migration

• Memory migration
  – Pre copy or Post copy (or hybrid)

• VM state copy
  – VM exit (at source hypervisor)
  – Just copy VMCS
  – \texttt{VMRESUME} to start VM at destination hypervisor

• Storage
  – Network storage
  – Distributed storage
  – Storage migration like memory migration

• Network
  – Copy configuration (i.e., MAC address) of virtual NICs
**Live Migration Support**

| CR3                      | Reserved² | Address of PML4 table | Ignored | PML4E: present | Ignored | M-1 | 32 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------------------|-----------|-----------------------|---------|----------------|---------|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| PML4E: not present       | Ignored   | Address of page-directory-pointer table | Ignored | Reserved | PDTE: page directory | Ignored | Rsvd. | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored |
| PDTE: 1GB page           | Ignored   | Address of 1GB page frame | Reserved | Reserved | PDTE: page table | Ignored | Rsvd. | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored |
| PDE: 2MB page            | Ignored   | Address of 2MB page frame | Reserved | Reserved | PDE: page table | Ignored | Rsvd. | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored |
| PDE: not present         | Ignored   | Address of page table | Reserved | Reserved | PDE: page table | Ignored | Rsvd. | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored |
| PDE: not present         | Ignored   | Address of 4KB page frame | Reserved | Reserved | PDE: page table | Ignored | Rsvd. | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored | Reserved | Ignored |

**NOTES:**

1. M is an abbreviation for MAXPHYADDR.

2. Reserved fields must be 0.

**Dirty bit**

*Figure from Intel® 64 and IA-32 Architectures Software Developer’s Manual Vol. 3A 4.5*
Technologies behind PaaS
Technologies behind PaaS

• LXC (Linux Containers)
  – Lightweight virtual machine
    • Run many instances

• Platforms and Frameworks
  – Platforms
    • Hadoop (for distributed computing)
    • Docker (for VM image management/deployment)
  – Frameworks
    • Ruby on Rails (for Web)
  – Load balancing
    • (These network portion will be in next week.)
Technologies behind SaaS
Technologies behind SaaS

• Rich UI over browser
  – HTML5
    • Support of new features
      – Canvas
      – Local storage / Session storage
      – Local file access w/o page transition (e.g., FileReader)
      – Drag & Drop
      – Multimedia support w/o third-party plugins
      – Offline application w/ cache
    • WebSockets / WebRTC
  – CSS3
    • New style modules e.g., gradient
  – JavaScript
    • Development of faster JavaScript engines
      – V8
      – asm.js