Lecture - III

Processes

Tevfik Koşar

Louisiana State University
January 22nd, 2008

Roadmap

• Virtual Machines
• Processes
  - Basic Concepts
  - Context Switching
  - Process Queues
  - Process Scheduling
  - Process Termination
Virtual Machines

- A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface identical to the underlying bare hardware.
- The virtual machine creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory.

Virtual Machines (Cont.)

- The resources of the physical computer are shared to create the virtual machines:
  - CPU scheduling can create the appearance that users have their own processor.
  - Spooling and a file system can provide virtual card readers and virtual line printers.
  - A normal user time-sharing terminal serves as the virtual machine operator’s console.
The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.

A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.

The virtual machine concept is difficult to implement due to the effort required to provide an exact duplicate to the underlying machine.
### VMware Architecture

<table>
<thead>
<tr>
<th>Application 1</th>
<th>Application 2</th>
<th>Application 3</th>
<th>Application 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>guest operating system (free BSD)</td>
<td>guest operating system (Windows NT)</td>
<td>guest operating system (Windows XP)</td>
<td>guest operating system (Windows XP)</td>
</tr>
<tr>
<td>virtual CPU</td>
<td>virtual CPU</td>
<td>virtual CPU</td>
<td>virtual CPU</td>
</tr>
<tr>
<td>virtual memory</td>
<td>virtual memory</td>
<td>virtual memory</td>
<td>virtual memory</td>
</tr>
<tr>
<td>virtual devices</td>
<td>virtual devices</td>
<td>virtual devices</td>
<td>virtual devices</td>
</tr>
</tbody>
</table>

- **Virtualization layer**
- **Host operating system (Linux)**
- **Hardware**
  - CPU
  - Memory
  - I/O devices

### Processes
Process Concept

- An operating system executes a variety of programs:
  - Batch system - jobs
  - Time-shared systems - user programs or tasks
- **Process - a program in execution**; process execution must progress in sequential fashion
- A process includes:
  - program counter
  - stack: temporary data

Process Control Block (PCB)

Information associated with each process
- Process state
  - (running, waiting..)
- Program counter
- CPU registers
- CPU scheduling information
  - (i.e. process priority)
- Memory-management information
  - (i.e. page & segment tables)
- Accounting information
- I/O status information
Process State

- As a process executes, it changes state
  - **new**: The process is being created
  - **ready**: The process is waiting to be assigned to a process
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **terminated**: The process has finished execution

Context Switch

- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process
- Context-switch time is overhead; the system does no useful work while switching
- Switching time is dependent on hardware support
Process Scheduling Queues

- **Job queue** - set of all jobs in the system
- **Ready queue** - set of all processes residing in main memory, ready and waiting to execute
- **Device queues** - set of processes waiting for an I/O device
- Processes migrate among the various queues
Ready Queue And Various I/O Device Queues

Representation of Process Scheduling
Schedulers

- **Long-term scheduler** (or job scheduler) - selects which processes should be brought into the ready queue
- **Short-term scheduler** (or CPU scheduler) - selects which process should be executed next and allocates CPU

Schedulers (Cont.)

- Short-term scheduler is invoked very frequently (milliseconds) ⇒ (must be fast)
- Long-term scheduler is invoked very infrequently (seconds, minutes) ⇒ (may be slow)
- The long-term scheduler controls the *degree of multiprogramming*
- Processes can be described as either:
  - **I/O-bound process** - spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** - spends more time doing computations; few very long CPU bursts

⇒ long-term schedulers need to make careful decision
Addition of Medium Term Scheduling

- In time-sharing systems: remove processes from memory “temporarily” to reduce degree of multiprogramming.
- Later, these processes are resumed ➔ Swapping

Process Creation

- Parent process create children processes, which, in turn create other processes, forming a tree of processes
- Resource sharing
  - Parent and children share all resources
  - Children share subset of parent’s resources
  - Parent and child share no resources
- Execution
  - Parent and children execute concurrently
  - Parent waits until children terminate
A tree of processes on a typical Solaris

- sched: root process for OS
- pageout: manages memory
- fsflush: manages file system
- init: root for user processes
- inetd: Networking
- dtlogin: user login screen
- ...

⇒ Unique process id’s

Process Creation (Cont.)

- Address space
  - Child duplicate of parent
  - Child has a program loaded into it
- UNIX examples
  - fork system call creates new process
  - exec system call used after a fork to replace the process’ memory space with a new program
C Program Forking Separate Process

```c
int main()
{
  Pid_t pid;
  /* fork another process */
  pid = fork();
  if (pid < 0) { /* error occurred */
    fprintf(stderr, "Fork Failed");
    exit(-1);
  }
  else if (pid == 0) { /* child process */
    execlp("/bin/ls", "ls", NULL);
  }
  else { /* parent process */
    /* parent will wait for the child to complete */
  }
}
```

Process Termination

- Process executes last statement and asks the operating system to delete it (**exit**)
  - Output data from child to parent (via **wait**)
  - Process’ resources are deallocated by operating system
- Parent may terminate execution of children processes (**abort**)
  - Child has exceeded allocated resources
  - Task assigned to child is no longer required
  - If parent is exiting
    - Some operating system do not allow child to continue if its parent terminates
      - All children terminated - **cascading termination**
Cooperating Processes

- **Independent** process cannot affect or be affected by the execution of another process
- **Cooperating** process can affect or be affected by the execution of another process
- Advantages of process cooperation
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience

Summary

- Virtual Machines
- Processes
  - Basic Concepts
  - Context Switching
  - Process Queues
  - Process Scheduling
  - Process Termination

- **Next Lecture:** Threads
- **Reading Assignment:** Chapter 3 from Silberschatz.
- **HW 1 will be out next class, due 1 week**
Acknowledgements


- “Modern Operating Systems” book and supplementary material by A. Tanenbaum

- R. Doursat and M. Yuksel from UNR