Thread Model for Work Unit

B. Ramamurthy

Modified version slides provided
by A. Tennenbaum’s Text
The Thread Model

(a) Three processes each with one thread
(b) One process with three threads
Per process vs per thread items

<table>
<thead>
<tr>
<th>Per process items</th>
<th>Per thread items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address space</td>
<td>Program counter</td>
</tr>
<tr>
<td>Global variables</td>
<td>Registers</td>
</tr>
<tr>
<td>Open files</td>
<td>Stack</td>
</tr>
<tr>
<td>Child processes</td>
<td>State</td>
</tr>
<tr>
<td>Pending alarms</td>
<td></td>
</tr>
<tr>
<td>Signals and signal handlers</td>
<td></td>
</tr>
<tr>
<td>Accounting information</td>
<td></td>
</tr>
</tbody>
</table>

- Items shared by all threads in a process
- Items private to each thread
Implementing Threads in User Space

A user-level threads package
Implementing Threads in the Kernel

A threads package managed by the kernel
Hybrid Implementations

Multiple user threads on a kernel thread

Multiplexing user-level threads onto kernel-level threads
Scheduler Activations

- Goal – mimic functionality of kernel threads
  - gain performance of user space threads
- Avoids unnecessary user/kernel transitions
- Kernel assigns virtual processors to each process
  - lets runtime system allocate threads to processors
- Problem:
  Fundamental reliance on kernel (lower layer)
  calling procedures in user space (higher layer)
Pop-Up Threads

- Creation of a new thread when message arrives
  (a) before message arrives
  (b) after message arrives
- Thread pools
Thread Scheduling (2)

Possible scheduling of kernel-level threads

- 50-msec process quantum
- threads run 5 msec/CPU burst