[537] Threads

Chapters 26
Tyler Harter
10/01/14
Review: Easy Piece 1

Virtualization

- CPU
- Memory
Review: Easy Piece 1

Virtualization

- CPU
  - Context Switch
  - Schedulers
- Memory
  - Allocation
  - Segmentation
  - Paging
Review: Easy Piece 1

Virtualization

CPU
  - Context Switch
  - Schedulers

Memory
  - Allocation
  - Segmentation
  - Paging

TLBs
  - Multilevel
  - Swapping
(1) name 3 places xv6 stores registers for context switches?
(2) how can you game the multi-level feedback queue?
(3) what field does malloc need for a free node that a kernel doesn’t need to track a free page?
(4) which segment needs different translation rules? Why?
(5) how can a TLB avoid flushing for context switches
(6) what’s the advantage of multi-level PT over segmented PTs?
(7) what H/W support is needed to do LRU swapping?
Threads
(a) not interleaved

CPU:  A  A  A  B
Disk:  A  A

(b) interleaved

CPU:  A  B  A  B  A  B
Disk:  A  A
What if there is only one process?
The future:
- same speed
- more cores

Faster programs $\Rightarrow$ concurrent execution
Goal

Write applications that fully utilize many CPUs...
Strategy 1

Build applications from many communicating processes
- like Chrome (process per tab)
- communicate via pipe() or similar

Pros/cons?
Strategy 1

Build applications from many communicating processes
- like Chrome (process per tab)
- communicate via `pipe()` or similar

Pros/cons?
- don’t need new abstractions
- cumbersome programming
- copying overheads
- expensive context switching (why expensive?)
Strategy 2

New abstraction: the thread.

Threads are just like processes, but they share the address space (e.g., using same PT).
CPU 1
running
thread 1

CPU 2
running
thread 2

RAM
PageDir A
PageDir B
...
CPU 1
running thread 1

CPU 2
running thread 2

RAM
PageDir A
PageDir B

Virt Mem (PageDir A)
CODE  HEAP  ...
Each thread may be executing different code at the same time.
CPU 1
running thread 1
PTBR
IP

CPU 2
running thread 2
PTBR
IP

RAM
PageDir A
PageDir B
...

Virt Mem
(PageDir A)
CODE
HEAP
...

...
CPU 1 running thread 1
CPU 2 running thread 2
RAM

Virt Mem (PageDir A)

CODE | HEAP | ...

PageDir A
PageDir B
threads executing different functions need different stacks
Demo: basic threads
Scheduling Problems (One CPU)

State:
0x9cd4: 100
%eax: ?
%rip = 0x195

Thread 1
%eax: ?
%rip: 0x195

Thread 2
%eax: ?
%rip: 0x195

process
control
blocks:

T1 0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
0x9cd4: 100
%eax: 100
%rip = 0x19a

Thread 1
%eax: ?
%rip: 0x195

Thread 2
%eax: ?
%rip: 0x195

T1  0x195  mov 0x9cd4, %eax
    0x19a  add $0x1, %eax
    0x19d  mov %eax, 0x9cd4
Scheduling Problems (One CPU)

**State:**
- \(0x9cd4\): 100
- \(%eax\): 101
- \(%rip = 0x19d\)

**Thread 1**
- \(%eax\): ?
- \(%rip: 0x195\)

**Thread 2**
- \(%eax\): ?
- \(%rip: 0x195\)

Process control blocks:
- \(0x195\) mov \(0x9cd4\), \(%eax\)
- \(0x19a\) add \($0x1\), \(%eax\)
- \(0x19d\) mov \(%eax\), \(0x9cd4\)
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x1a2

process control blocks:

Thread 1
%eax: ?
%rip: 0x195

Thread 2
%eax: ?
%rip: 0x195

T1

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
- 0x9cd4: 101
- %eax: 101
- %rip = 0x1a2

Context Switch
- 0x195  mov 0x9cd4, %eax
- 0x19a  add $0x1, %eax
- 0x19d  mov %eax, 0x9cd4

Thread 1
- %eax: ?
- %rip: 0x195

Thread 2
- %eax: ?
- %rip: 0x195

T1
Context Switch

State:
0x9cd4: 101
%eax: ?
%rip = 0x195

Thread 1
%eax: 101
%rip: 0x1a2

Thread 2
%eax: ?
%rip: 0x195

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:

0x9cd4: 101
%eax: 1
%rip = 0x195

T2 → 0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

Thread 1
%eax: 101
%rip: 0x1a2

Thread 2
%eax: ?
%rip: 0x195

Process control blocks:
Scheduling Problems (One CPU)

**State:**
- $0x9cd4$: 101
- %eax: 101
- %rip = 0x19a

**Thread 1**
- %eax: 101
- %rip: 0x1a2

**Thread 2**
- %eax: ?
- %rip: 0x195

---

**T2**

- 0x195 mov $0x9cd4, %eax
- 0x19a add $0x1, %eax
- 0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 102
%rip = 0x19d

Thread 1
%eax: 101
%rip: 0x1a2

Thread 2
%eax: ?
%rip: 0x195

process control blocks:

T2
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
0x9cd4: 102
%eax: 102
%rip = 0x1a2

process control blocks:

Thread 1
%eax: 101
%rip: 0x1a2

Thread 2
%eax: ?
%rip: 0x195

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T2
Scheduling Problems (One CPU)

**State:**
- 0x9cd4: 102
- %eax: 102
- %rip = 0x1a2

**Thread 1**
- %eax: 101
- %rip: 0x1a2

**Thread 2**
- %eax: ?
- %rip: 0x195

Thread 1 and Thread 2 transition:
- T2

**Thread 1 code: (0x195)**
- mov 0x9cd4, %eax
- add $0x1, %eax
- mov %eax, 0x9cd4

**GOOD!**
Another schedule
Scheduling Problems (One CPU)

**State:**
- \(0x9cd4\): 100
- \(%eax\): ?
- \(%rip\): 0x195

**Thread 1**
- \(%eax\): ?
- \(%rip\): 0x195

**Thread 2**
- \(%eax\): ?
- \(%rip\): 0x195

**Process Control Blocks:**
- \(0x195\) mov \(0x9cd4\), \(%eax\)
- \(0x19a\) add \($0x1\), \(%eax\)
- \(0x19d\) mov \(%eax\), \(0x9cd4\)
Scheduling Problems (One CPU)

State:
0x9cd4: 100
%eax: 100
%rip = 0x19a

Thread 1
%eax: ?
%rip: 0x195

Thread 2
%eax: ?
%rip: 0x195

T1
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
0x9cd4: 100
%eax: 101
%rip = 0x19d

process control blocks:

Thread 1
%eax: ?
%rip: 0x195

Thread 2
%eax: ?
%rip: 0x195

T1 0x195 mov 0x9cd4, %eax
     0x19a add $0x1, %eax
     0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
- 0x9cd4: 100
- %eax: 101
- %rip = 0x19d

Thread 1:
- %eax: ?
- %rip: 0x195

Thread 2:
- %eax: ?
- %rip: 0x195

Context Switch

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T1
Scheduling Problems (One CPU)

**State:**
- \texttt{0x9cd4}: 100
- \texttt{%eax}: ?
- \texttt{%rip} = \texttt{0x195}

**Thread 1**
- \texttt{%eax}: 101
- \texttt{%rip}: \texttt{0x19d}

**Thread 2**
- \texttt{%eax}: ?
- \texttt{%rip}: \texttt{0x195}

**Context Switch**

\begin{align*}
\text{\texttt{0x195}} & \text{ mov } \texttt{0x9cd4}, \texttt{%eax} \\
\text{\texttt{0x19a}} & \text{ add } \texttt{$0x1}, \texttt{%eax} \\
\text{\texttt{0x19d}} & \text{ mov } \texttt{%eax}, \texttt{0x9cd4}
\end{align*}
Scheduling Problems (One CPU)

**State:**
- 0x9cd4: 100
- %eax: ?
- %rip = 0x195

**Thread 1**
- %eax: 101
- %rip: 0x19d

**Thread 2**
- %eax: ?
- %rip: 0x195

**Thread 2**
- 0x195: mov 0x9cd4, %eax
- 0x19a: add $0x1, %eax
- 0x19d: mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
0x9cd4: 100
%eax: 100
%rip = 0x19a

Thread 1
%eax: 101
%rip: 0x19d

Thread 2
%eax: ?
%rip: 0x195

T2
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

process control blocks:
Scheduling Problems (One CPU)

**State:**
- `0x9cd4`: 100
- `%eax`: 101
- `%rip` = `0x19d`

**Thread 1**
- `%eax`: 101
- `%rip`: `0x19d`

**Thread 2**
- `%eax`: ?
- `%rip`: `0x195`

**Process Control Blocks:**
- `0x195` mov `0x9cd4`, `%eax`
- `0x19a` add `$0x1`, `%eax`
- `0x19d` mov `%eax`, `0x9cd4`
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x1a2

Thread 1
%eax: 101
%rip: 0x19d

Thread 2
%eax: ?
%rip: 0x195

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T2
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x1a2

Thread 1
%eax: 101
%rip: 0x19d

Thread 2
%eax: ?
%rip: 0x195

Context Switch
0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x19d

Thread 1
%eax: 101
%rip: 0x19d

Thread 2
%eax: 101
%rip: 0x1a2

Context Switch

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

T1
Scheduling Problems (One CPU)

State:
0x9cd4: 101
%eax: 101
%rip = 0x19d

Thread 1
%eax: 101
%rip: 0x19d

Thread 2
%eax: 101
%rip: 0x1a2

T1

0x195 mov 0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4
Scheduling Problems (One CPU)

**State:**
- \(0x9cd4\): 101
- \(\%eax\): 101
- \(\%rip = 0x1a2\)

**Thread 1**
- \(\%eax: 101\)
- \(\%rip: 0x1a2\)

**Thread 2**
- \(\%eax: 101\)
- \(\%rip: 0x1a2\)

**Process Control Blocks:**

- \(0x195\) \(\text{mov}\ 0x9cd4, \%eax\)
- \(0x19a\) \(\text{add}\ \$0x1, \%eax\)
- \(0x19d\) \(\text{mov}\ \%eax, 0x9cd4\)

**T1**
Scheduling Problems (One CPU)

State:
- $0x9cd4$: 101
- %eax: 101
- %rip = 0x1a2

Wrong!

0x195 mov $0x9cd4, %eax
0x19a add $0x1, %eax
0x19d mov %eax, 0x9cd4

Thread 1:
- %eax: 101
- %rip: 0x19d

Thread 2:
- %eax: 101
- %rip: 0x1a2
Timeline View

Thread 1
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

Thread 2
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

How much is added?
Thread 1
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

Thread 2
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

How much is added?
Timeline View

Thread 1

- mov 0x123, %eax
- add %0x1, %eax
- mov %eax, 0x123

Thread 2

- mov 0x123, %eax
- add %0x1, %eax
- mov %eax, 0x123

How much is added?
Timeline View

Thread 1

mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

Thread 2

mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123

How much is added?
Timeline View

Thread 1

```
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123
```

Thread 2

```
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123
```

How much is added?
Concurrency leads to non-deterministic bugs, called race conditions.

Whether bug manifests depends on CPU schedule!

Passing tests means little.

How to program: imagine scheduler is malicious.
What do we want?

Want all or none of these instructions to execute. That is, we want them to be atomic.

```asm
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123
```
What do we want?

Want all or none of these instructions to execute. That is, we want them to be atomic.

```
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123
```
What do we want?

Want all or none of these instructions to execute. That is, we want them to be atomic.

```
mov 0x123, %eax
add %0x1, %eax
mov %eax, 0x123
```

We want mutual exclusion for critical sections. That is, if I run, you can’t (and vice versa).
More Demos
Announcements

Be sure you’ve read up to Chapter 24 in OSTEP!

p2a due this Friday.

Office hours today. In office. 1pm.