CS 134:
Operating Systems
Multiprocessing

CS 134:
Operating Systems
Multiprocessing

Overview

Overview

Multiprocessing Designs

OS Implications

Programming Models

Other Issues

SIMD and MIMD

Multiple CPUs come in several flavors:

SIMD: Single Instruction, Multiple Data

- Also called vector processor
- Sample instruction: `a[i] = b[i] + c[i]` for `i` in small range (e.g., 0-3)
- Canonical example: GPUs

MIMD: Multiple Instruction, Multiple Data

I.e., 2 or more (semi-)independent CPUS

SIMD and MIMD
Multiple CPUs come in several flavors:
SIMD: Single Instruction, Multiple Data
- Also called vector processor
- Sample instruction: a[i] = b[i] + c[i] for i in small range (e.g., 0-3)
- Canonical example: GPUs
MIMD: Multiple Instruction, Multiple Data
I.e., 2 or more (semi-)independent CPUS

We won't talk further about SIMD; from an OS point of view it's just another CPU.

MIMD Approaches

MIMD can be:

- Several chips or cores, (semi-)private memories, able to access each other's memory (NUMA—Non-Uniform Memory Access)
- Several chips or cores, one memory (SMP—Symmetric Multiprocessing)
- Several boxes (possibly each SMP or NUMA) connected by network (distributed system)

MIMD Approaches
MIMD can be:
- Several chips or cores, (semi-)private memories, able to access each other's memory (NUMA—Non-Uniform Memory Access)
- Several chips or cores, one memory (SMP—Symmetric Multiprocessing)
- Several boxes (possibly each SMP or NUMA) connected by network (distributed system)

NUMA Issues

NUMA means processes access local memory faster

⇒ Allocate process memory on local CPU

⇒ Processes should have “CPU affinity”

NUMA Issues
NUMA means processes access local memory faster
⇒ Allocate process memory on local CPU
⇒ Processes should have “CPU affinity”

SMP Issues

SMPs still have caches

Introduces *cache coherency* problems:

- Processor 0 uses compare-and-swap to set a lock nonzero
- Write goes into local cache for speed
- Processor 1 reads lock from own cache, sees it's still zero...

Cure: hardware coherency guarantees

... but spinlocks now have super-high costs

- May be better to do thread switch

SMP Issues
SMPs still have caches
Introduces cache coherency problems:
Processor 0 uses compare-and-swap to set a lock nonzero
Write goes into local cache for speed
Processor 1 reads lock from own cache, sees it's still zero...
Cure: hardware coherency guarantees
... but spinlocks now have super-high costs
May be better to do thread switch

Thread switch is high cost, but may be cheaper than spinlock.

SMP Scheduling

Threads are often related

- Schedule independently or together?
- Completely independent: job completion is slowest thread
- Together: some CPUs may be wasted on waiting for events
- Always good to keep thread *x* on same CPU (because cache is filled)

SMP Scheduling
Threads are often related
- Schedule independently or together?
- Completely independent: job completion is slowest thread
- Together: some CPUs may be wasted on waiting for events
- Always good to keep thread *x* on same CPU (because cache is filled)

Distributed Systems

Many ways to communicate

Most important modern approach is. . .

Distributed Systems

Many ways to communicate

Most important modern approach is. . .

Distributed Systems

Many ways to communicate

Most important modern approach is. . . the Internet!

Distributed Systems

Many ways to communicate

Most important modern approach is. . . the Internet!

Distributed Systems

Many ways to communicate

Most important modern approach is. . . the Internet!

Communicating with skinny wires introduces new problems:

- Can't move process to other machine (or must work *hard*)
- Locking becomes *really* hard
- Programming multiprocessor systems is much harder

Distributed Systems

Many ways to communicate

Most important modern approach is. . . the Internet!

Communicating with skinny wires introduces new problems:

- Can't move process to other machine (or must work *hard*)
- Locking becomes *really* hard
- Programming multiprocessor systems is much harder

Distributed Systems

Many ways to communicate

Most important modern approach is. . . the Internet!

Communicating with skinny wires introduces new problems:

- Can't move process to other machine (or must work *hard*)
- Locking becomes *really* hard
- Programming multiprocessor systems is much harder
- . . . and what if network connection goes down?

Distributed Systems

Many ways to communicate

Most important modern approach is. . . the Internet!

Communicating with skinny wires introduces new problems:

- Can't move process to other machine (or must work *hard*)
- Locking becomes *really* hard
- Programming multiprocessor systems is much harder
- . . . and what if network connection goes down?

RPC

Programming is hard, so need abstractions that simplify things

Remote Procedure Call (RPC) makes distant system look like normal function

1. *Marshal* arguments (i.e., pack up and serialize)
2. Send procedure ID and arguments to remote system
3. Wait for response
4. Deserialize return value

Class Exercise

What are the advantages and disadvantages?

RPC

Programming is hard, so need abstractions that simplify things

Remote Procedure Call (RPC) makes distant system look like normal function

1. *Marshal* arguments (i.e., pack up and serialize)
2. Send procedure ID and arguments to remote system
3. Wait for response
4. Deserialize return value

Class Exercise

What are the advantages and disadvantages?

DSM

RPC is nice, but limits parallelism

SMPs can do cool things because memory is shared

So why not simulate shared memory across the network?

Teeny problem: hard to make it work fasta

DSM
RPC is nice, but limits parallelism
SMPs can do cool things because memory is shared
So why not simulate shared memory across the network?
Teeny problem: hard to make it work fasta

"Hard" is a gross understatement.

Load Balancing

Suppose you have servers A, B, C, and D

A and B are currently overloaded, C and D underloaded

A notices the situation and sends excess work to C and D

Simultaneously, B does the same! Now C and D are overloaded

Result can be thrashing

Common solution: have one *front-end* machine whose sole job is allocating load to others

Load Balancing
Suppose you have servers A, B, C, and D
A and B are currently overloaded, C and D underloaded
A notices the situation and sends excess work to C and D
Simultaneously, B does the same! Now C and D are overloaded
Result can be thrashing
Common solution: have one *front-end* machine whose sole job is allocating load to others

Random assignment works surprisingly well.

How Does Google Work?

Well, it's a secret...

But basically they use the front-end approach

Obvious problem: one front end can't handle millions of requests per second even if it does almost nothing

Solution: *DNS Round Robin* tricks you into picking one of many dozens of front ends (roughly at random) to talk to

How Does Google Work?

Well, it's a secret...

But basically they use the front-end approach

Obvious problem: one front end can't handle millions of requests per second even if it does almost nothing

Solution: *DNS Round Robin* tricks you into picking one of many dozens of front ends (roughly at random) to talk to

Example of Google's DNS tricks

These commands were run within 15 seconds of each other:

```
bow:2:877> host www.google.com
www.google.com has address 74.125.224.241
www.google.com has address 74.125.224.242
www.google.com has address 74.125.224.243
www.google.com has address 74.125.224.244
www.google.com has address 74.125.224.240

bow:2:878> ssh lever.cs.ucla.edu host www.google.com
www.google.com has address 74.125.239.19
www.google.com has address 74.125.239.20
www.google.com has address 74.125.239.17
www.google.com has address 74.125.239.18
www.google.com has address 74.125.239.16
```


Example of Google's DNS tricks
These commands were run within 15 seconds of each other:
bow:2:877> host www.google.com
www.google.com has address 74.125.224.241
www.google.com has address 74.125.224.242
www.google.com has address 74.125.224.243
www.google.com has address 74.125.224.244
www.google.com has address 74.125.224.240
bow:2:878> ssh lever.cs.ucla.edu host www.google.com
www.google.com has address 74.125.239.19
www.google.com has address 74.125.239.20
www.google.com has address 74.125.239.17
www.google.com has address 74.125.239.18
www.google.com has address 74.125.239.16