On the Glory of Corey
Christopher Smowton
Multicore Trouble, Outlined

- Fast cores
- Interconnect bottleneck
- So, minimise traffic
Multicore Trouble, Depicted

Core 1
Cache

Core 2
Cache

Main Memory
Line
Multicore Trouble, Depicted

Core 1

Cache

Core 2

Cache

Main Memory
Multicore Trouble, Depicted
Multicore Trouble, Depicted

Core 1

Cache

Main Memory

Core 2

Cache
Two Pitfalls

• Exchanging lines between cores when we didn't need to at all
• Exchanging more lines than we needed to
The Peril of False Sharing

- POSIX file descriptor table is process-global
- Guarded by a spinlock (on Linux)
  - And contended spinlocks bounce
- So `open`, `dup`, `close` are all expensive
- Same goes for `mmap`
Optimising for Multicore

- Make most data core-private
- Exchange as little as you can
- Like a distributed system
Fear Not, Here Comes Corey

- Address Ranges: private pieces of VAS
- Shares: private pieces of FD table
- Kernel Cores: a kthread per device
Address Ranges: Like POSIX Shared Memory, but better

- Mark some subset of your memory private
- ...and some of it public
- No locks to modify private mappings
  - And no need for expensive TLB shootdown
Two threads sharing a page

- Thread 1
- Thread 2
- Single Virtual Address Space
- Thread 1's Page
- Shared Page
- Thread 2's Page
Two processes sharing a page

Process 1

VAS

Thread 1's Page

Process 2

VAS

Shared Page

Thread 2's Page
Memory might be shared, but Page Tables are not.

- Process 1
  - Top Level PT
  - Bottom Level PT
    - Process 1's Page

- Process 2
  - Top Level PT
  - Bottom Level PT
    - Process 2's Page
Memory might be shared, but Page Tables are not.

- Process 1
  - Top Level PT
  - Bottom Level PT
    - Process 1's Page

- Process 2
  - Top Level PT
  - Bottom Level PT
    - Shared Page
    - Process 2's Page
Memory might be shared, but Page Tables are not.
Memory might be shared, but Page Tables are not.
Corey PT Sharing

- Process 1
  - Top Level PT
    - Bottom Level PT
      - Process 1's Page
  - Shared BL PT
- Process 2
  - Top Level PT
  - Bottom Level PT
    - Process 2's Page
Corey PT Sharing

Process 1

Top Level PT

Bottom Level PT

Process 1's Page

Process 2

Top Level PT

Shared BL PT

Bottom Level PT

Shared Page

Process 2's Page
Corey MM: Evaluation

![Graph showing cycles per page versus cores for different scenarios: Linux single, Linux separate, and Corey address ranges.](image-url)
MapReduce
Corey MM: Evaluation

The graph shows the time (in milliseconds) taken for different configurations with varying numbers of cores. The configurations include Linux single, Linux separate, and Corey address ranges. The time increases linearly with the number of cores for all configurations.
Shares: Similar, but for FDs

- Private and public FD tables
- Again, don't contend for the private table
- Shared shares and FD tables both serialization points
Kernel Cores

- Keep all device driver state core-local
- Ring buffer to/from kernel core
- Kernel core polls the ring and device
- User cores push requests and wait for replies
Without a Kernel Core

Core 1

DMA Buffer List

RX Queue

Misc State

Core 2
With a Kernel Core

Core 1

IPC

Core 3

IPC

Core 2

DMA Buffer List

RX Queue

Misc State
Kernel Core Eval

The graph shows the number of connections per second as a function of the number of cores. The y-axis represents connections per second, ranging from 0 to 120,000. The x-axis represents cores, ranging from 1 to 16.

There are three categories depicted in the graph:
- Polling
- Dedicated
- Linux

Each category is represented by markers: triangles, diamonds, and circles respectively. The graph illustrates how the number of connections per second increases as the number of cores increases, with different trends for each category.
Summary

- Earth shattering it is not
- But these are sensible optimisations
- And real progress
- Not a complete operating system
  - But we already have those