W4118: multikernel

Instructor: Junfeng Yang

References: Modern Operating Systems (3rd edition), Operating Systems Concepts (8th edition), previous W4118, and OS at MIT, Stanford, and UWisc

Motivation: sharing is expensive

- Difficult to parallelize single-address space kernels with shared data structures
  - Locks/atomic instructions: limit scalability
  - Must make every shared data structure scalable
    - Partition data
    - Lock-free data structures
    - …
  - Tremendous amount of engineering
- Root cause
  - Expensive to move cache lines
  - Congestion of interconnect

Multikernel: explicit sharing via messages

- Multicore chip == distributed system!
  - No shared data structures!
  - Send messages to core to access its data

- Advantages
  - Scalable
  - Good match for heterogeneous cores
  - Good match if future chips don't provide cache-coherent shared memory

Challenge: global state

- OS must manage global state
  - E.g., page table of a process
- Solution
  - Replicate global state
  - Read: read local copy ➔ low latency
  - Update: update local copy + distributed protocol to update remote copies
    - Do so asynchronously ("*split phase*")

Barrelfish overview

- Figure 5 in paper

- CPU driver
  - kernel-mode part per core
  - *Inter-Processor interrupt (IPI)*

- Monitors
  - OS abstractions
  - *User-level RPC (URPC)*

IPC through shared memory

```
#define CACHELINE (64)
struct box{
    char buf[CACHELINE-1]; // message contents
    char flag;  // high bit == 0 means sender owns it
}; __attribute__ ((aligned (CACHELINE)))

struct box m __attribute__ ((aligned (CACHELINE)));
```

```
send()
{
    // set up message
    memcpy(m.buf, …);
    m.flag |= 0x80;
}
```

```
recv()
{
    while(!(m.flag & 0x80))
        ;
    m.buf …  //process message
}
```

Case study: TLB shootdown

- When is it necessary?
- Windows & Linux: send IPIs
- Barrelfish: sends messages to involved monitors
  - 1 broadcast message ➔ N-1 invalidates, N-1 fetches
  - N unicast
  - Multicast protocol
    - Send message to each processor
    - Each process forwards to its cores
  - NUMA-aware multicast
    - Send to highest latency nodes first