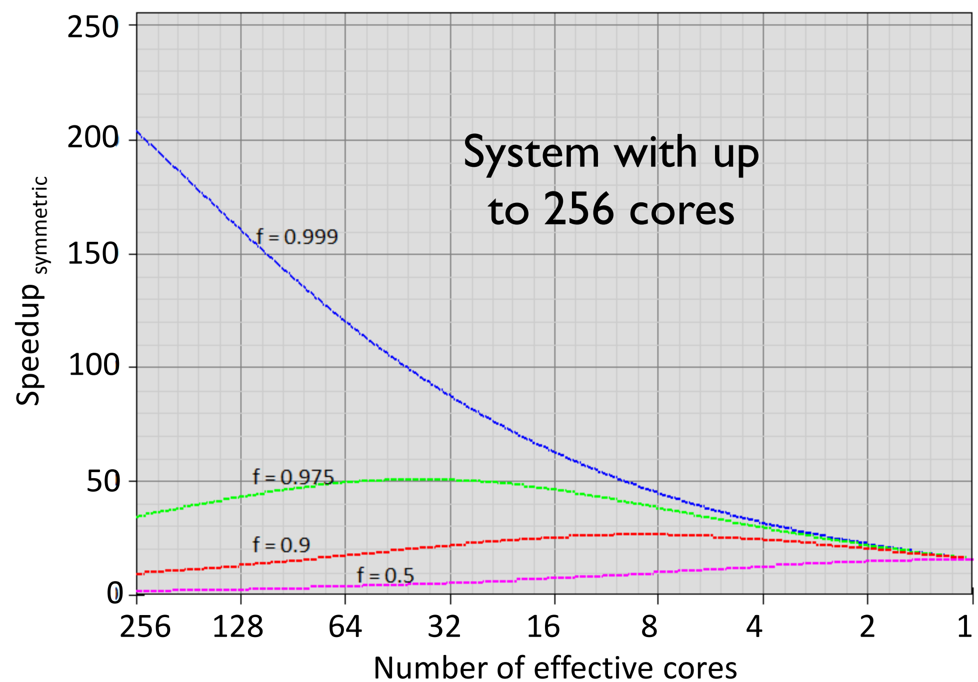
Dynamic Processors Demand Dynamic Operating Systems

Sankaralingam Panneerselvam
Michael M. Swift
Computer Sciences Department
University of Wisconsin, Madison, WI

Motivation

- Chip Multiprocessor
  - Does not support well for sequential workloads

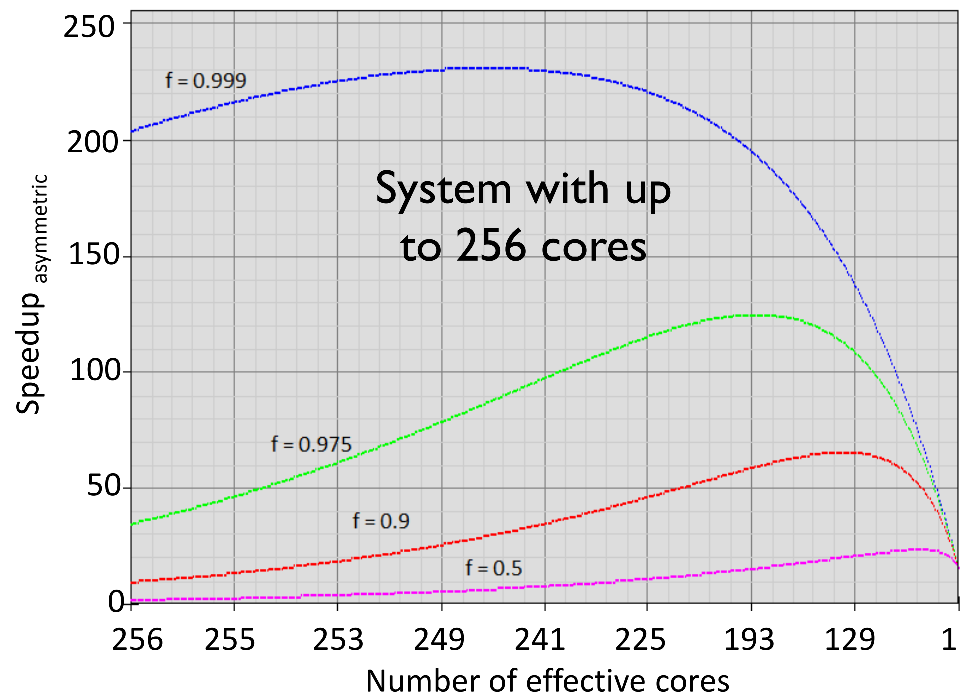
Possible Configurations



"Amdahls law in the multicore era"
[IEEE computer, July 2008]

Motivation

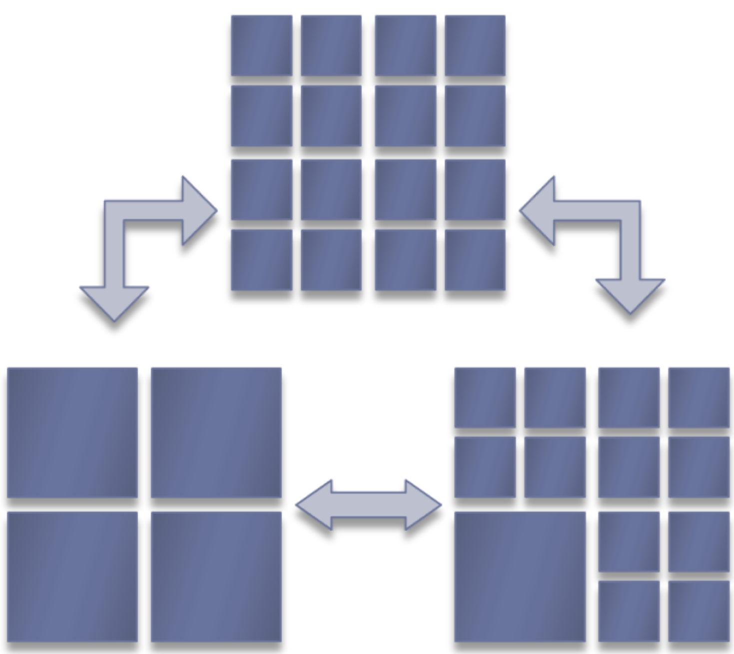
- Asymmetric Chip Multiprocessor
  - To satisfy diverse workloads

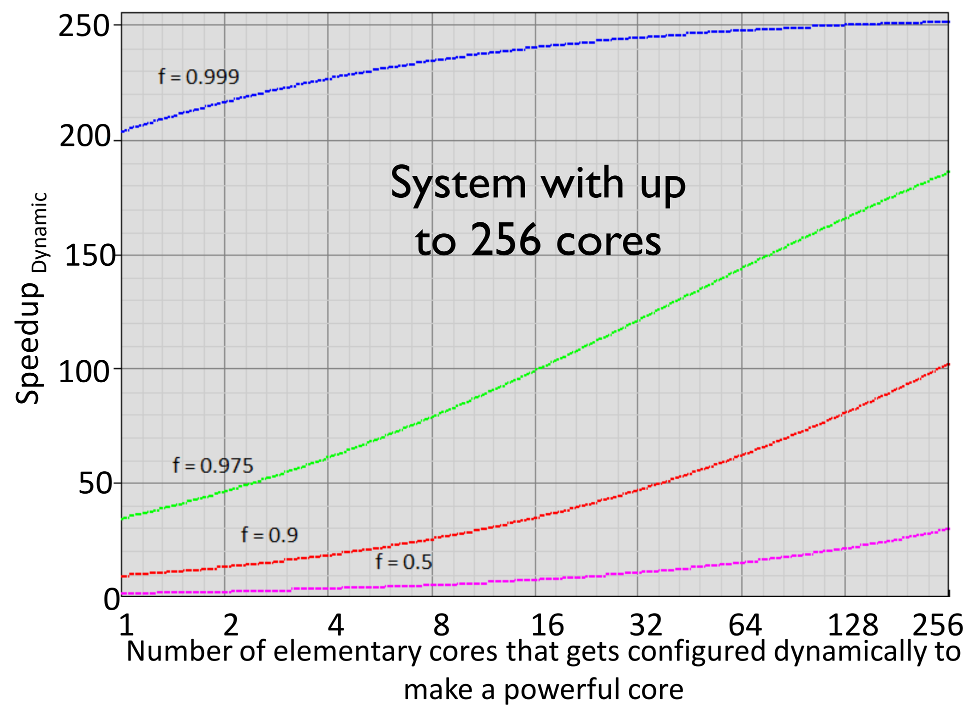


"Amdahls law in the multicore era"
[IEEE computer, July 2008]

Motivation

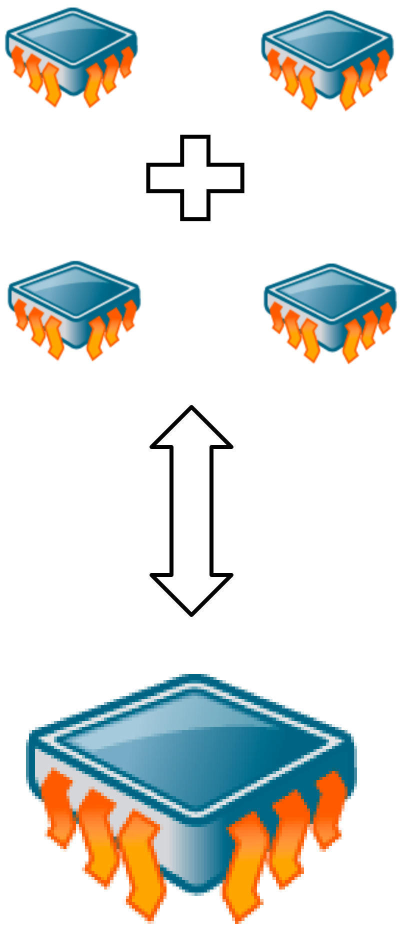
- Dynamic Multiprocessor
  - Flexible to cast to the right configuration based on the need



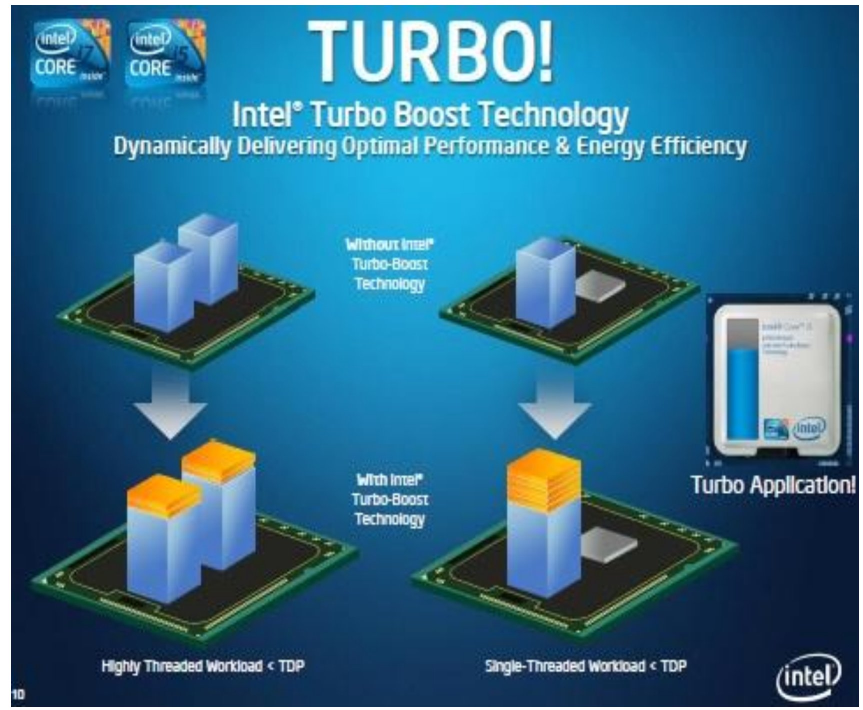


"Amdahls law in the multicore era"
[IEEE computer, July 2008]

Examples of Dynamic Multiprocessors



Core Fusion
[ISCA'07]



Intel Turbo Boost
[Nehalem]

Motivation

- Many mechanisms lead to dynamically variable processors
  - Performance
    - Merging resources: Core Fusion, Speculative Multithreading
    - Shifting power: Turbo Boost, Over-provisioned systems
  - Reliability
    - Redundant execution [ISCA'07]

Why reconfigure the OS?

- What happens if a processor goes to offline state without any notification?
  - Servicing of interrupts, IPI, Bottom halves is stopped
  - Other processors might wait for spinlock
  - RCU stall
  - Thread execution is stopped

Can the OS adapt to changing processors ?

- Common theme: the number of physical execution contexts may change dynamically and frequently
- Our work:
  - Analysis of Linux mechanisms for changing processors
  - Two new techniques for dynamically varying processors
    - Processor Proxies
    - Deferred/Parallel Hotplug

Outline

- Motivation
- Current Mechanisms
- Processor Proxies
- Deferred/Parallel hotplug

Why is changing processors hard?

- Many pieces of code know which processors are available
  - Scheduler
  - Per-CPU structures
- Distributed operations require processors to communicate
  - Communication between processors - IPI
  - Read Copy Update (RCU) mechanism

CPU dependence in Linux

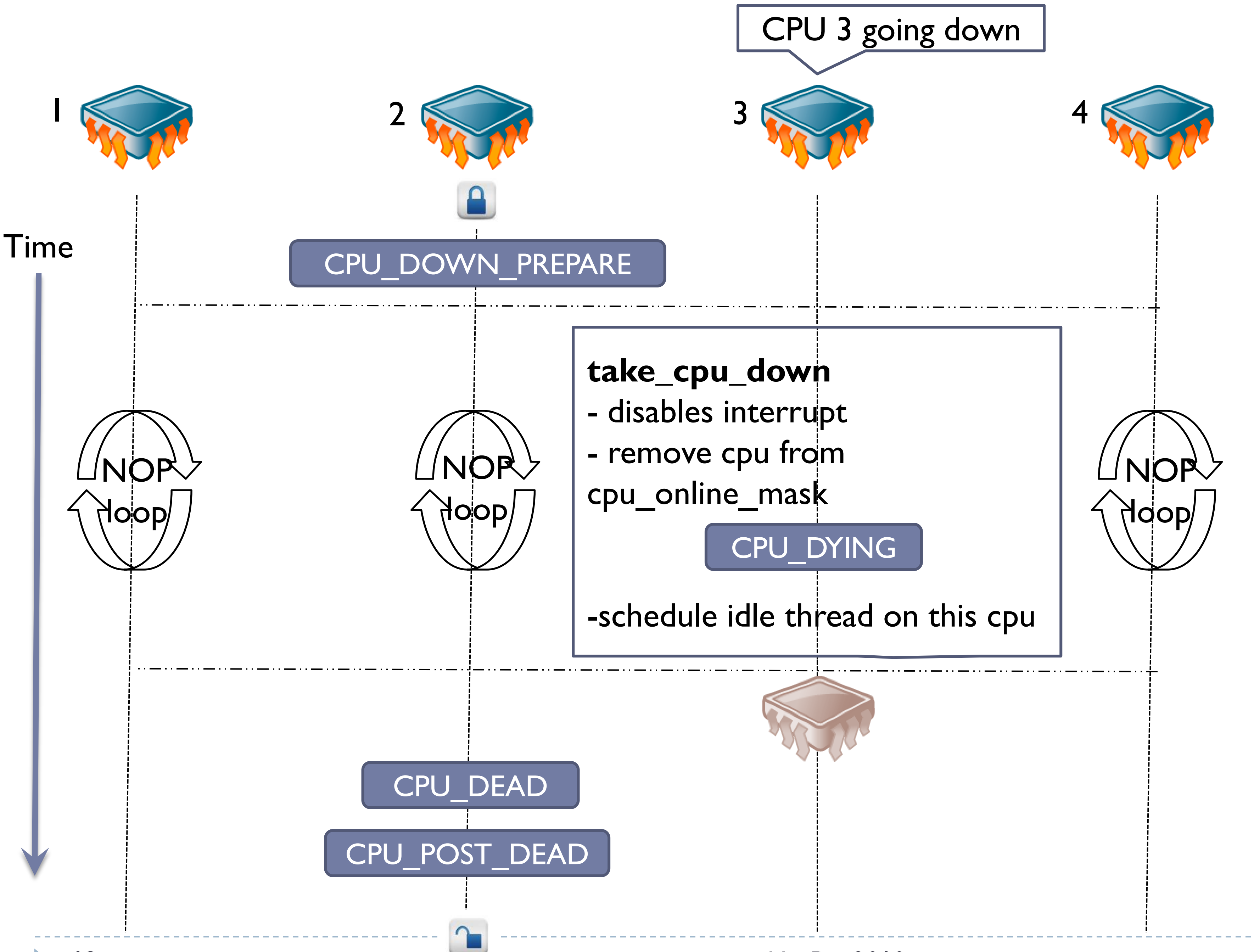
- Analysis of Linux 2.6.31-4 kernel on a 4 CPU machine

Number of per-CPU data structures	446 data structures
Number of callbacks when CPU set changes	35 callbacks
Frequency of global RCU operations	90 callbacks/second

- Inference: CPU dependences are widespread

Current solution: Linux Hotplug

- Hotplug allows dynamic addition/removal of a processor
  - Partitioning/virtualization
  - Physical repair
- Used for long-term reconfigurations
  - Assumes that processors, once off lined, never comes online
  - Notifies all relevant subsystems, creates/deletes all per-CPU state


CPU 3 going down
1
2
3
4
Time
CPU_DOWN_PREPARE
NOP loop
NOP loop
take_cpu_down
- disables interrupt
- remove cpu from cpu_online_mask
CPU_DYING
-schedule idle thread on this cpu
NOP loop
CPU_DEAD
CPU_POST_DEAD

Hotplug performance

Hotplug Operations	Cores	Latency (msec)
OFFLINE	1	25
	2	60
	3	137
ONLINE	1	106
	2	214
	3	331

- Good for virtualization but too slow for rapid reconfiguration

Outline

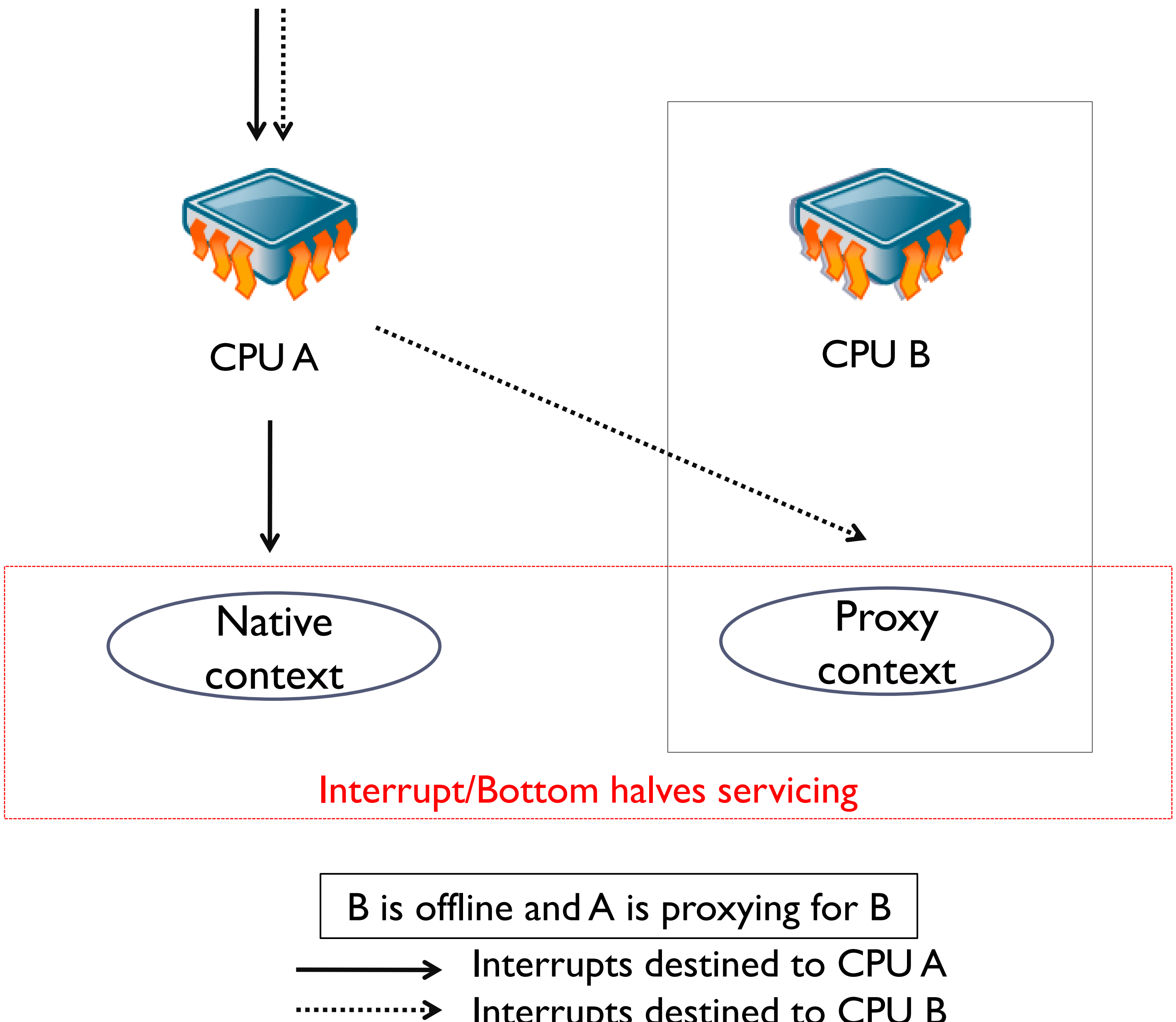
- Motivation
- Current Mechanisms
- Processor Proxies
- Deferred/Parallel hotplug

Our approach

- Strategy
  - Do very little for short-term changes
  - Do long-term changes off line, asynchronously and in parallel
- Solutions
  - Processor proxies address short-term reconfiguration
  - Deferred and Parallel hotplug reduces the frequency and latency of long-term reconfiguration

Processor Proxies

- A processor proxy is a fill-in for offline processor
- Provides separate execution context on the proxying CPU called the proxy context
- Participates in operations that requires the offline processor:
  - Servicing Inter Processor Interrupts (IPI)
  - Ensuring progress in RCU mechanism
- Does not execute threads


CPU A
CPU B
Native context
Proxy context
Interrupt/Bottom halves servicing
B is offline and A is proxying for B
Interrupts destined to CPU A
Interrupts destined to CPU B

Processor Proxy Evaluation Result

- Offline / Online performance compared to native

Hotplug Operations	Cores	Native (msec)	Proxy (msec)
OFFLINE	1	25	1.7
	2	60	4
	3	137	6.5
ONLINE	1	106	1.2
	2	214	2.8
	3	331	6

Deferred and Parallel Hotplug

- Processor proxies are not a long term solution
  - Threads don't run on a proxy
- If the reconfiguration is long lasting, move to a stable state
- Solutions:
  - Deferred hotplug: remove a CPU that is currently proxied
  - Parallel hotplug: reconfigure multiple CPUs simultaneously

Evaluation Results

Hotplug Operations	Cores	Native (msec)	Parallel (msec)
OFFLINE	1	25	25
	2	60	60
	3	137	130
ONLINE	1	106	106
	2	214	**111**
	3	331	**131**

- Performance of CPU online is greatly improved
  - Major time spent in initialization for CPU online
  - Initialization can happen in parallel

Conclusions

- Dynamic reconfiguration
  - Operating systems are not prepared
  - Hotplug mechanisms is too slow
- Low latency solutions
  - Processor Proxies
  - Deferred and Parallel hotplug
- Future work
  - Resource management