Mixed-Mode Multicore Reliability

Philip Wells, Koushik Chakraborty, Gurindar Sohi

pwells@google.com

kchak@engineering.usu.edu

sohi@cs.wisc.edu

ASPLOS ’09

Washington, D.C.

Overview

Circuit reliability decreasing as technology scales

- **Dual-modular redundancy (DMR): high coverage & cost**
- **Software has different reliability requirements**

  ➔You might want to run them at the same time

Mixed-mode multicore (MMM) reliability

- **Allow some applications to run in 'performance mode'**
- **Run others in DMR-based 'reliable mode'**

Conceptually simple, but:

- **Need to protect the integrity of apps in 'reliable mode'**
  - **Recheck TLB permission, run OS in reliable mode**
- **Desire to increase throughput by utilizing all cores**
  - **Leverage Multicore Virtualization techniques**

Outline

Background

- **Reliability trends**
- **DMR Overheads**

Mixed-mode multicore

- **Objectives & challenges**
- **Implementation**
- **Performance results**

Summary

Background: Reliability trends

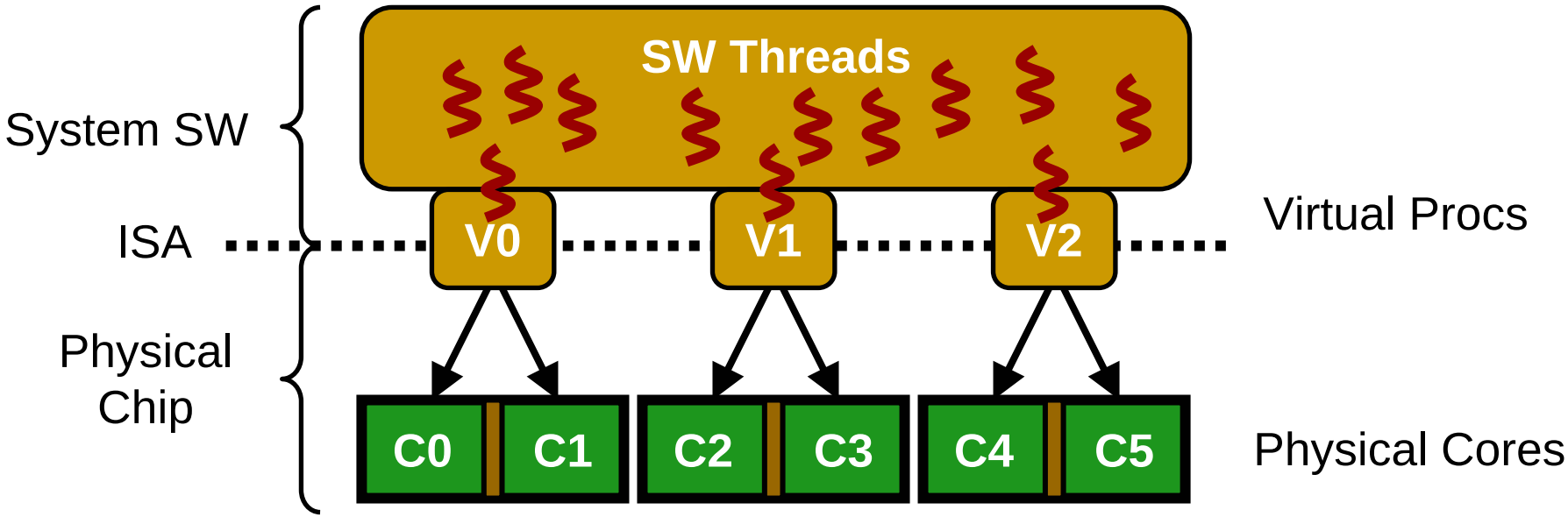
Circuits become more susceptible to:

- **Transient faults (e.g., particle strikes)**
- **Permanent faults (e.g., broken wires)**
- **Intermittent faults (e.g., thermal/power variation)**

Dual-modular redundancy (DMR)

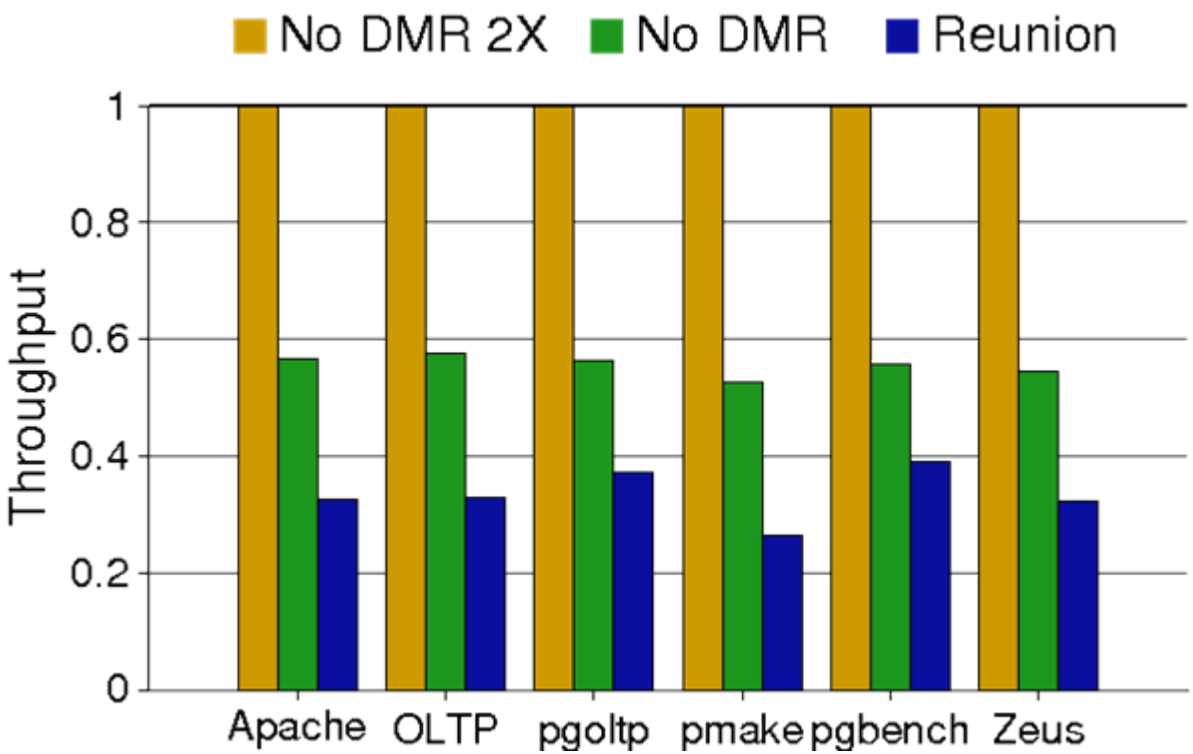
- **Join two cores to make a redundant pair**
- **Can provide very high coverage from a variety of faults**

Dual-modular redundancy (DMR)



We leverage Reunion [Smolens, 2007]

DMR overhead: Throughput



Simics full-system simulation (SPARC)

16-core system:

- **Reunion:**
  8 threads / 16 cores
- **No DMR:**
  8 threads / 8 cores
- **No DMR 2x:**
  16 threads / 16 cores

IPC overheads arise:

- **Window occupancy (Especially with SC)**
- **Serializing Instrs [HPCA 2008]**
- **C2C transfers**

Outline

Background

Mixed-mode multicore

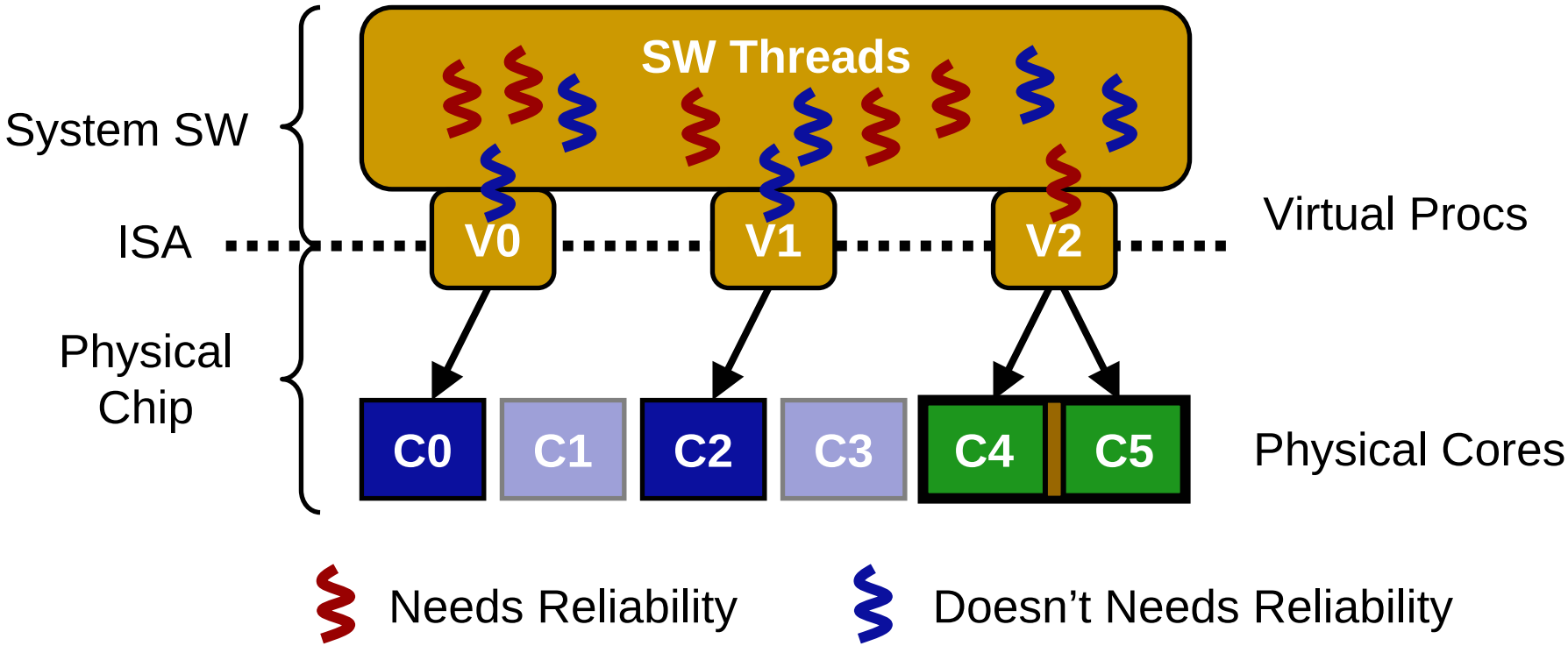
- Objectives & challenges
- Implementation
- Performance results

Summary

Mixed-mode objectives

Some apps need DMR, but many don't (web, media)

➔Allow some apps to run in 'performance mode'



Mixed-mode objectives

Some apps need DMR, but many don't (web, media)

➔Allow some apps to run in 'performance mode'

A performance app can erroneously change state

1) Protect the integrity of the system

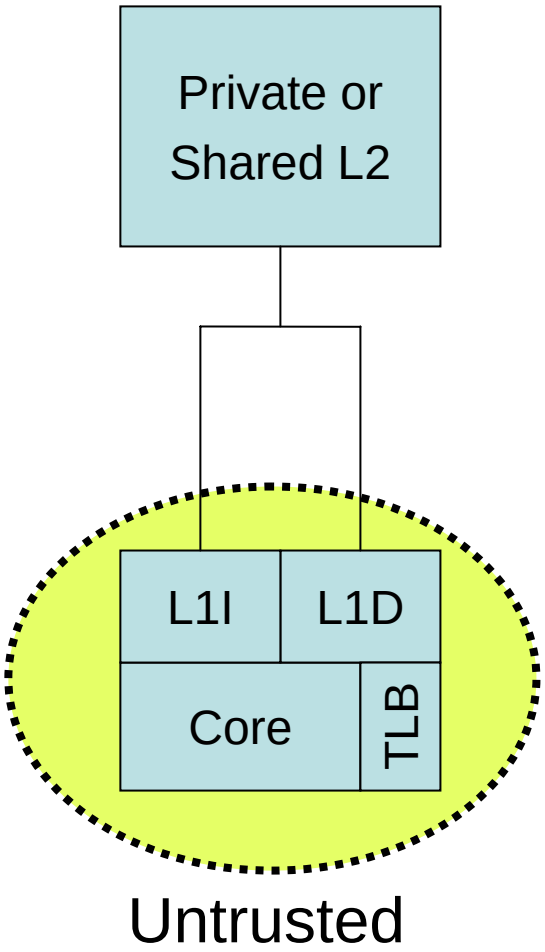
System software must run in 'reliable' mode

2) Creates frequent mode switches

Provide a simple interface to system software

3) But fully utilize cores to improve throughput

Challenge 1: Protect system integrity



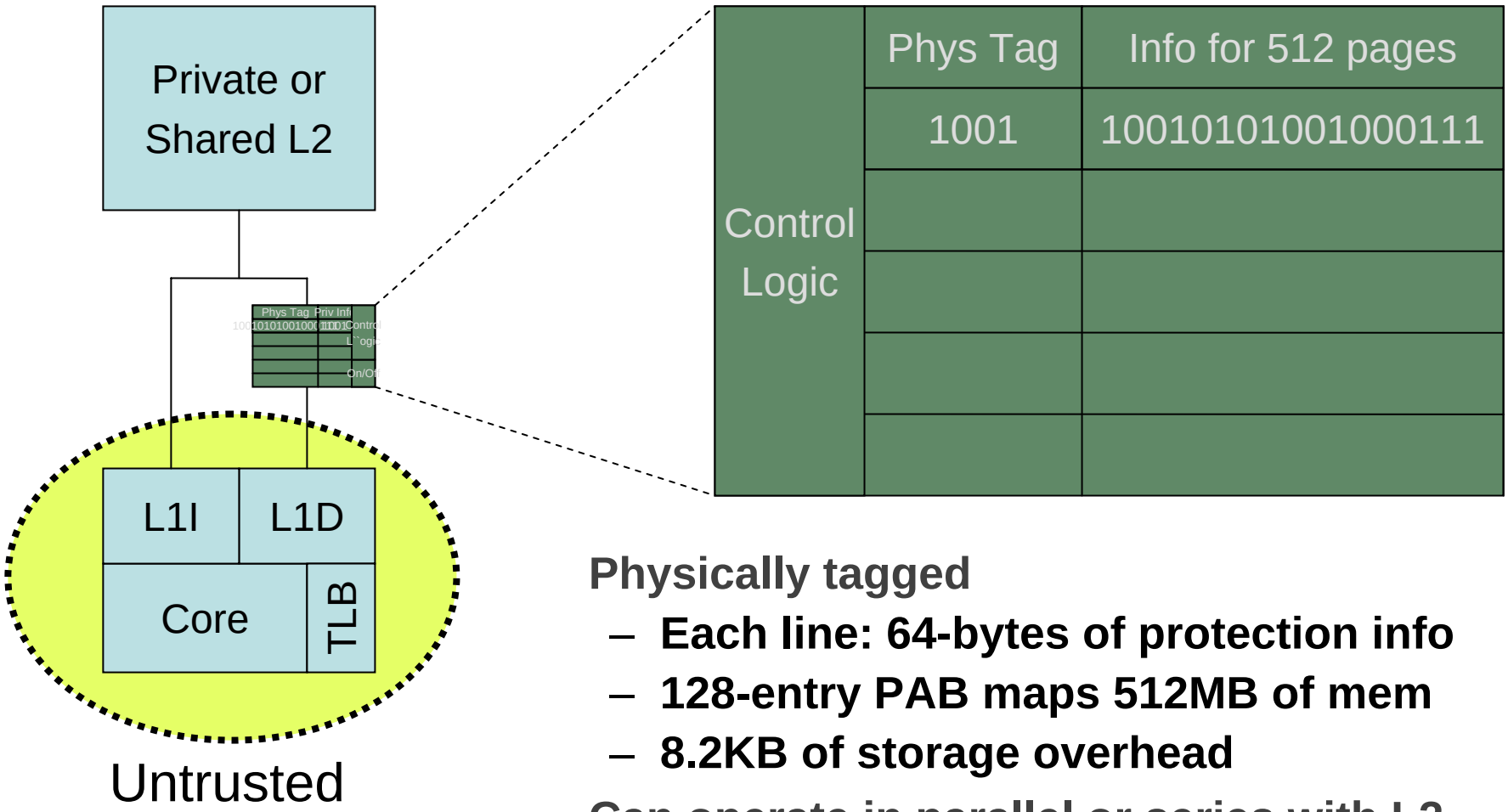
Fault in performance mode can corrupt 'reliable' state

- **Need to isolate faults to perf apps**

Memory state

- **Protection Assistance Table (PAT)**
  - **OS visible**
  - **1-bit/phys page, notes pages accessible in perf mode**
- **Protection Assistance Buffer (PAB)**
  - **Hardware**
  - **Cache of PAT, operates when core in non-DMR mode**
  - **Duplicates permission check of L1 write-throughs**

Protection Assistance Buffer (PAB)



Physically tagged

- **Each line: 64-bytes of protection info**
- **128-entry PAB maps 512MB of mem**
- **8.2KB of storage overhead**

Can operate in parallel or series with L2

Challenge 3: Improving throughput

Redundant cores used independently

- **Run more software threads**

Preserve a simple interface to system software:

- **For each VCPU, does it currently require DMR?**

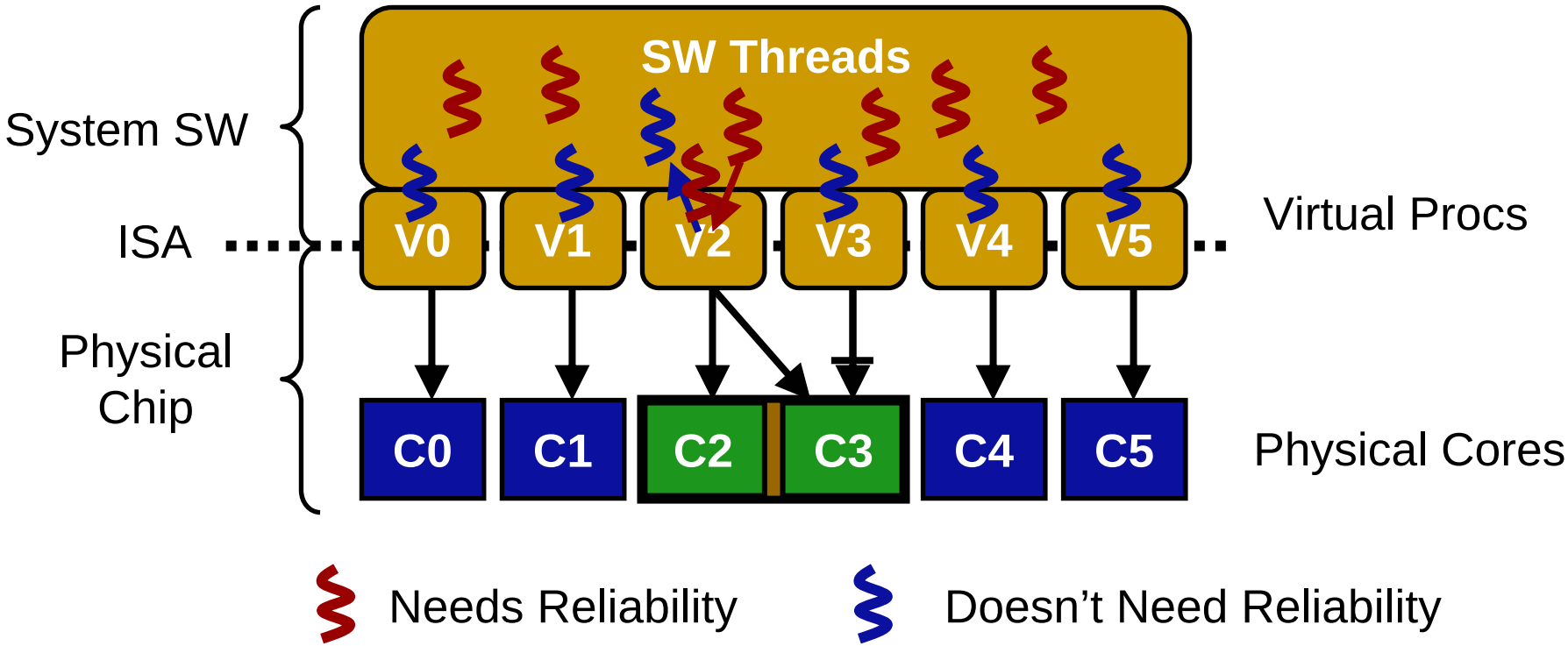
Simple interface allows:

- **Protect entire OS**
- **Abstract complex mechanisms**
- **Handle frequent mode changes**

Improving throughput cont…

Must expose 6 (instead of 3) VCPUs to OS

- **VCPUs dynamically and independently switch modes**



Solution: Overcommitted system

Multicore virtualization [PACT '06]

- **Flexibly map VCPUs onto cores**
- **Enable more OS-visible VCPUs than available cores**

HW spin detection heuristic

- **Determine when VCPU is not performing useful work**
  - **Preempt it in favor of one that is**
- **Incurs low overheads**

Mixed-mode performance

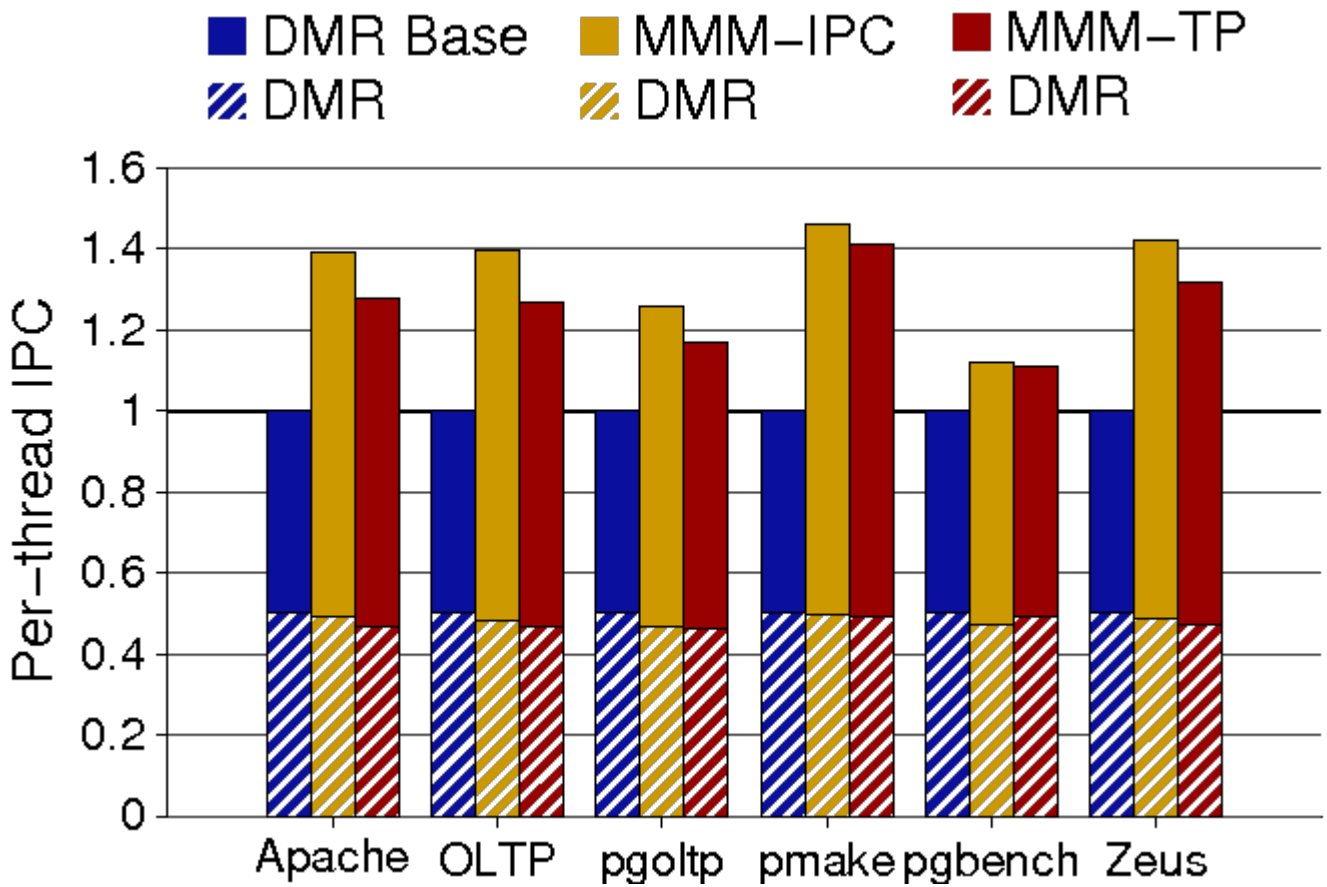
16 core system

- **Two guest VMs: One app needs DMR, the other doesn't**

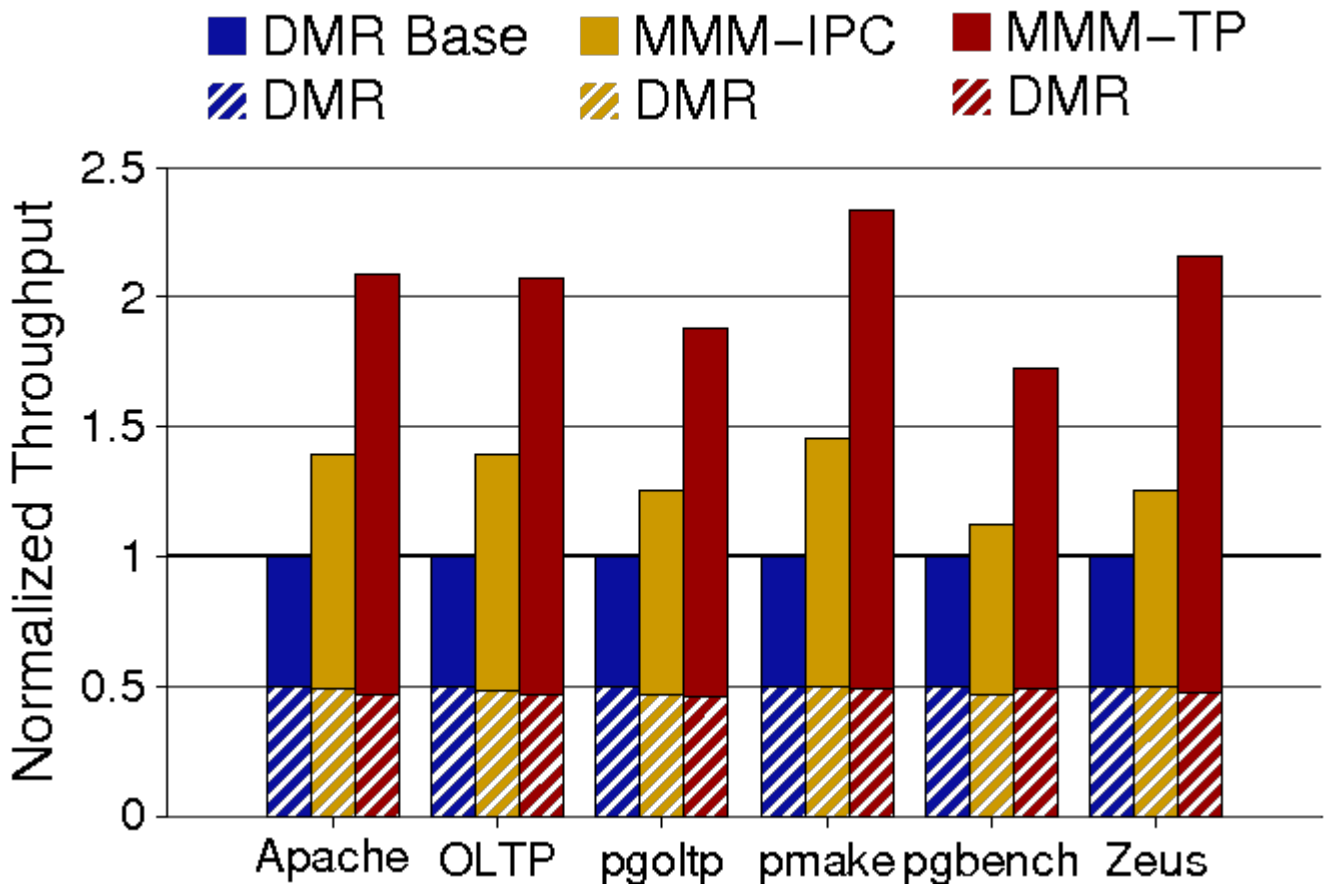
Three systems compared:

- **Reunion DMR baseline**
  - **Both apps must run in DMR mode because one needs to**
- **MMM-IPC**
  - **Turns off spare cores to improve IPC**
  - **Does not need multicore virtualization**
- **MMM-TP**
  - **Uses spare cores to execute 8 more VCPUs**
  - **Requires multicore virtualization**

Mixed-mode results: Per-thread IPC



Mixed-mode results: Throughput



Future work

Investigate different phases within a program

- **E.g., media player rendering vs. network authentication**
- **Proposed hardware interface already allows this**
  - **Need to define appropriate system API for marking pages and code regions as ‘performance’ or ‘reliable.’**

Security implications

- **E.g., relying on SPARC clean-win trap, or requiring loads to snoop PAB as well**

Mixed-mode summary

SW has different reliability requirements

- **Want to run both ‘reliable’ and ‘performance’ apps**

Mixed-mode multicore reliability

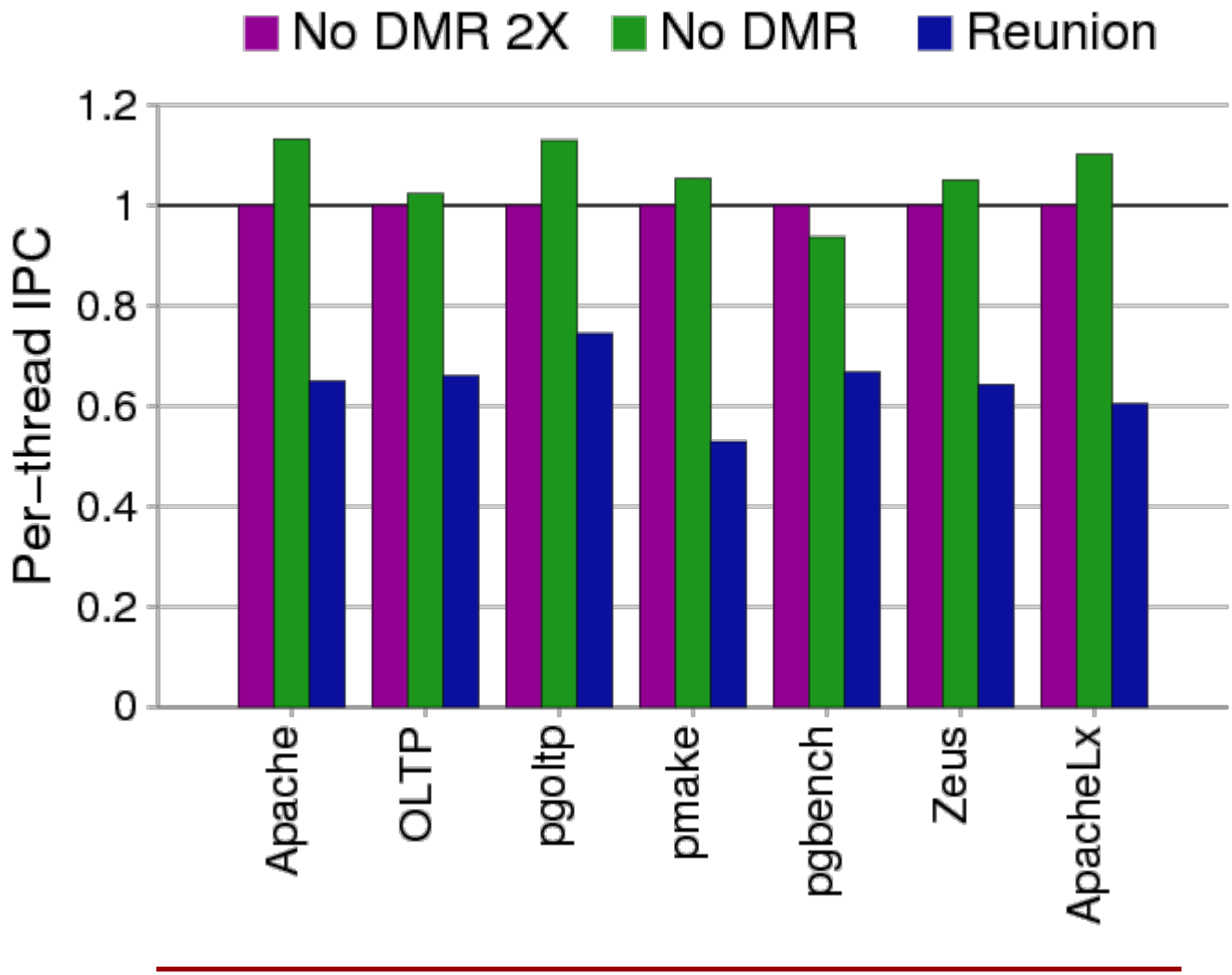
- **Provide reliability when needed**
- **Improves IPC and/or throughput when not**
- **Maintains system integrity & simple software interface**

Thank you!

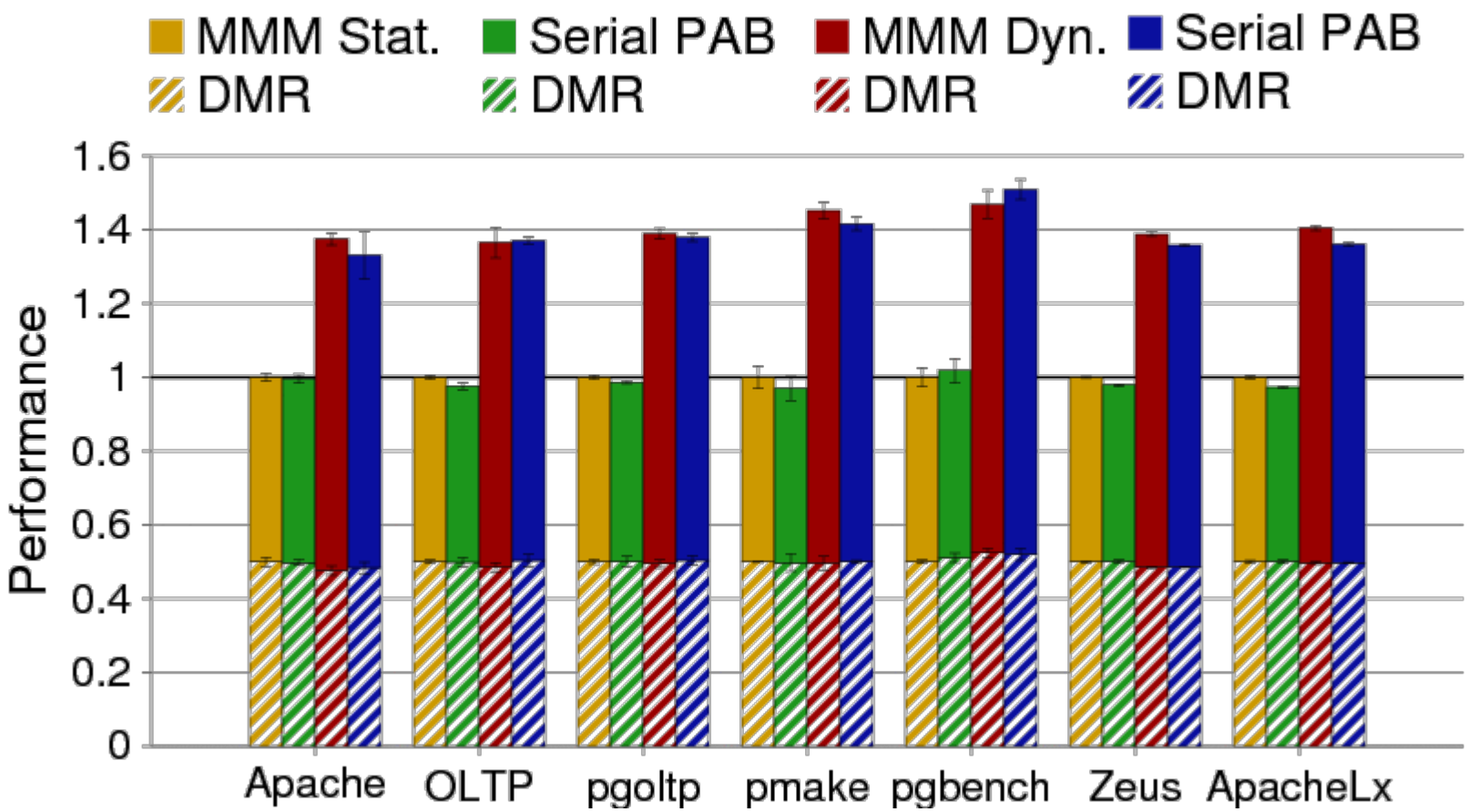
pwells@google.com
http://www.cs.wisc.edu/~pwells

Backup Slides

DMR overhead: Per-thread IPC



Results: Serial PAB lookup



Results: Mute coherence