Adapting to Intermittent Faults in Multicore Systems

Philip Wells, Koushik Chakraborty, Guri Sohi

{pwells, kchak, sohi}@cs.wisc.edu

ASPLOS 2008

Seattle, WA

Executive summary

Intermittent faults **on the horizon**

- **Occur in bursts due to physical & operating variation**
- **Resources become *temporary* unavailable**

How to *adapt* to intermittent availability of cores?

- **Three of the most logical techniques fall short**
- **We propose virtualization with *Overcommitted System***

This paper is *not* about detection or recovery

Intermittent faults: Overview

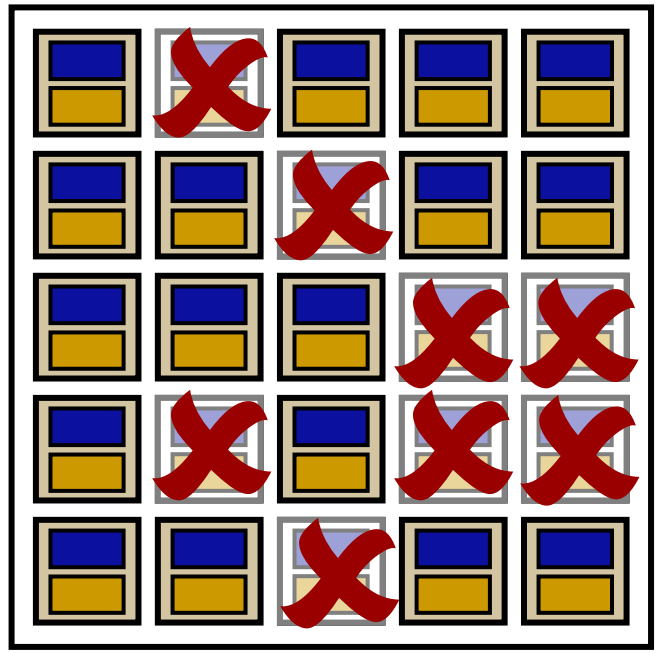
Arise due to combination of:

1) Physical variation

- **Manufacturing, wear-out**

2) Operating condition variation

- **Temp, voltage, SW phases**

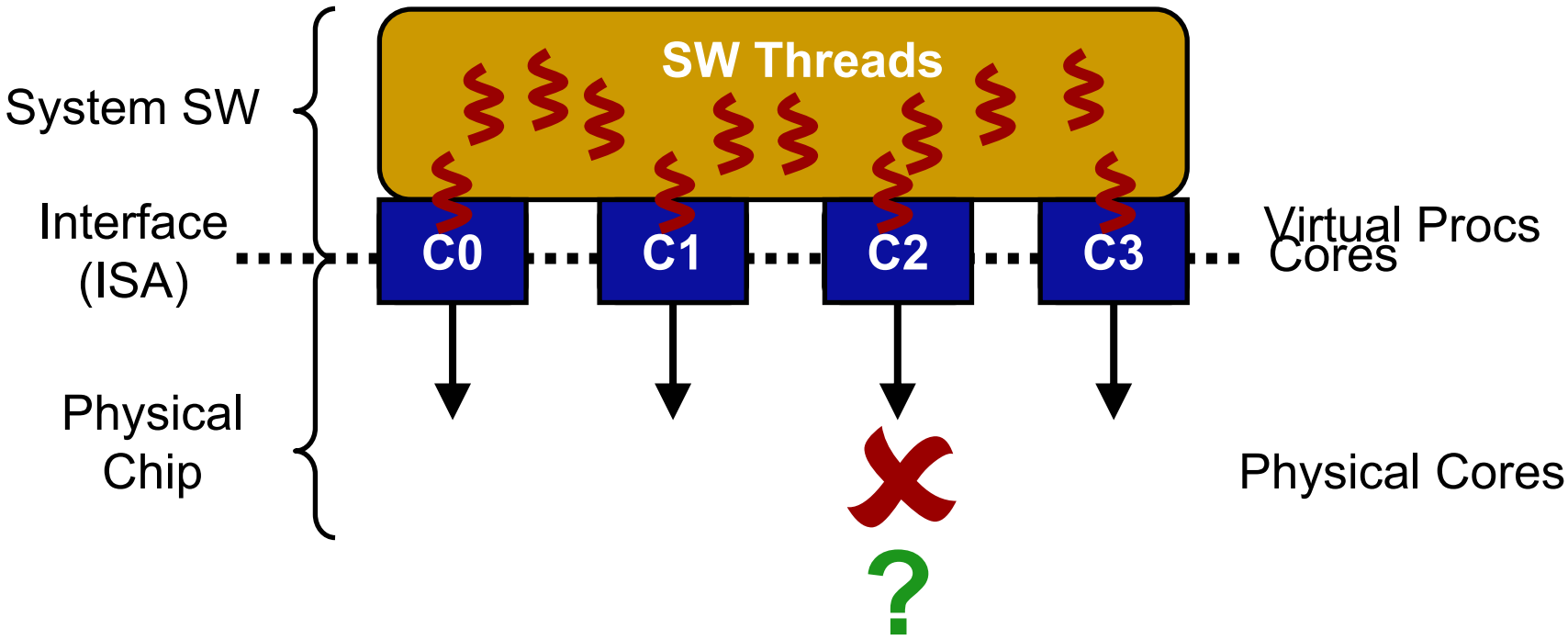


HW timing errors in bursts

➔Resources unavailable for a short period (cycles to secs)

How to adapt to intermittent resource availability?

Baseline system



What do we do?

Study methodology

Simics full-system simulation

- **8 OoO cores w/ private L2s and shared L3 cache**
- **Unmodified Solaris 9**
- **Several different workloads**

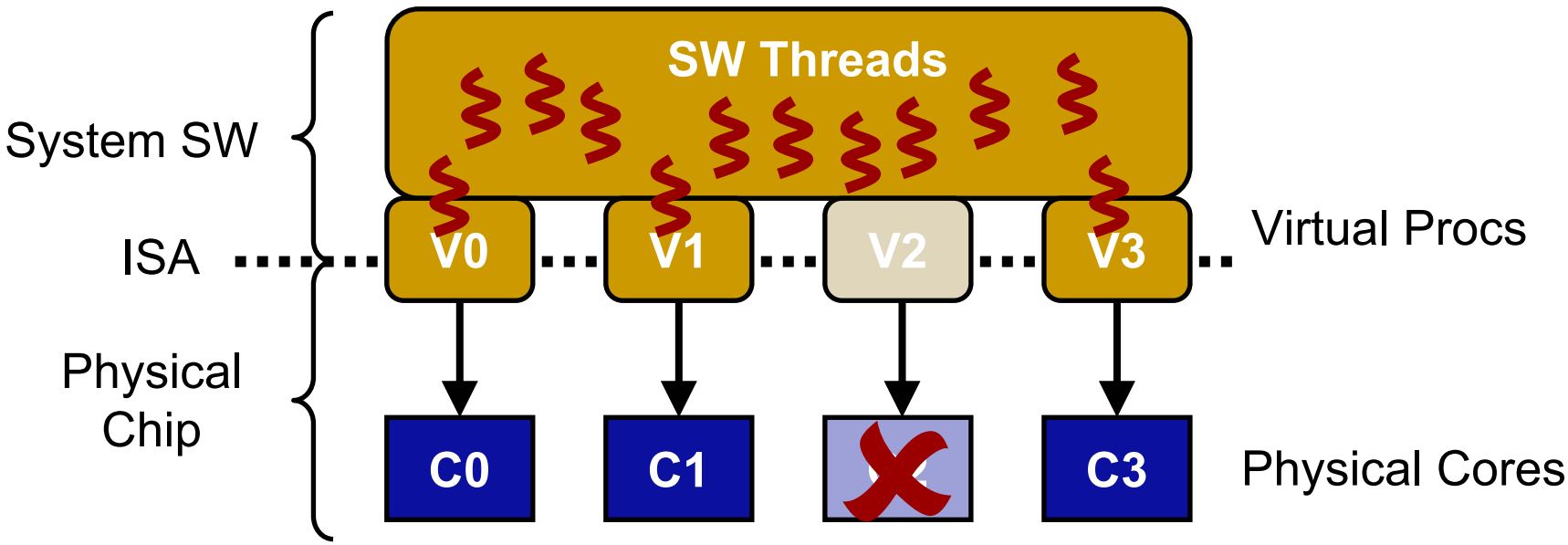
Experiments

- **Focus on system-level effects**
- **Assume 10k cycles to recover checkpoint**

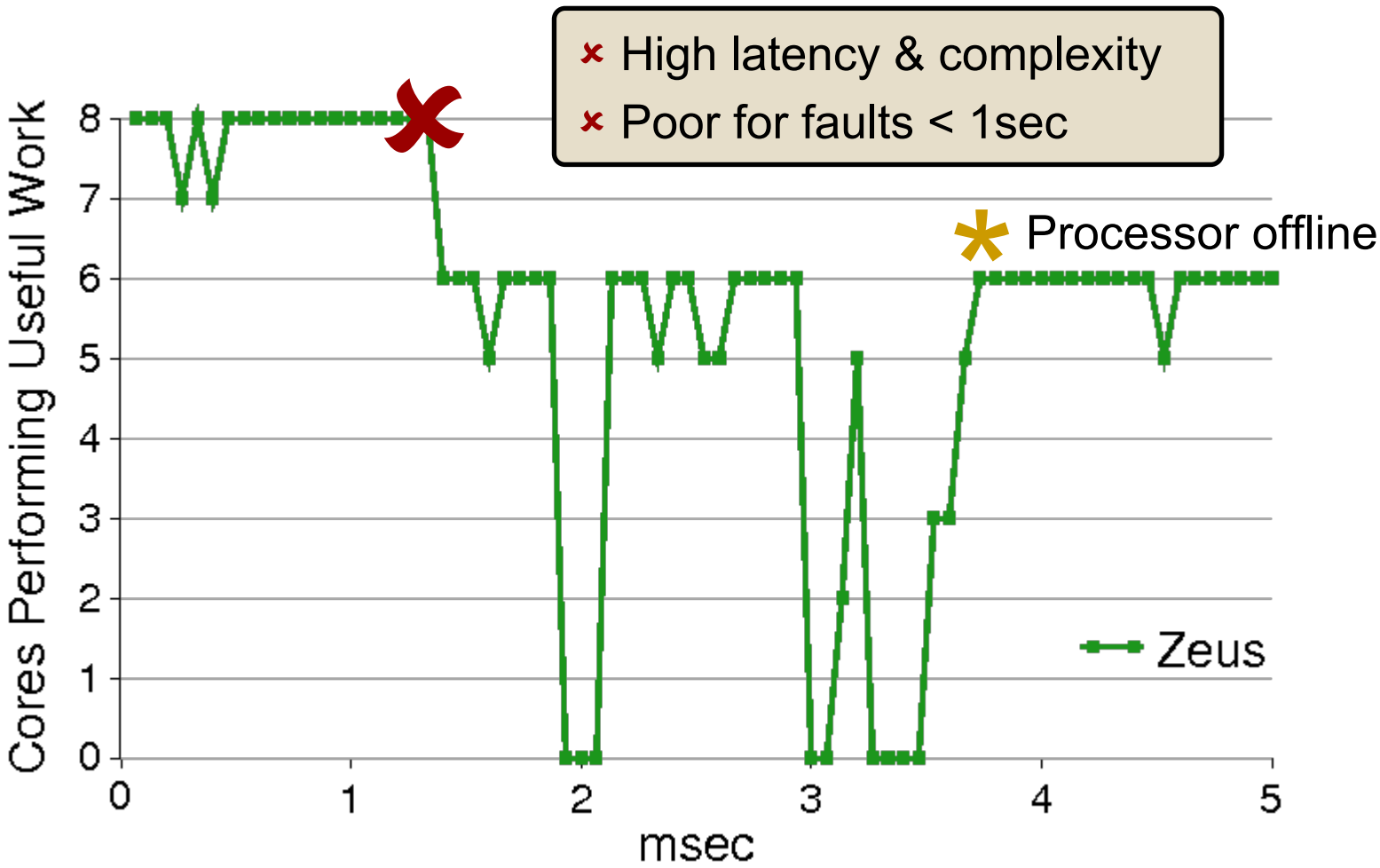
Technique 1: OS reconfiguration

Ask OS (or hypervisor) to stop using core

- **Send interrupt and use Solaris's *Dynamic Reconfig.***

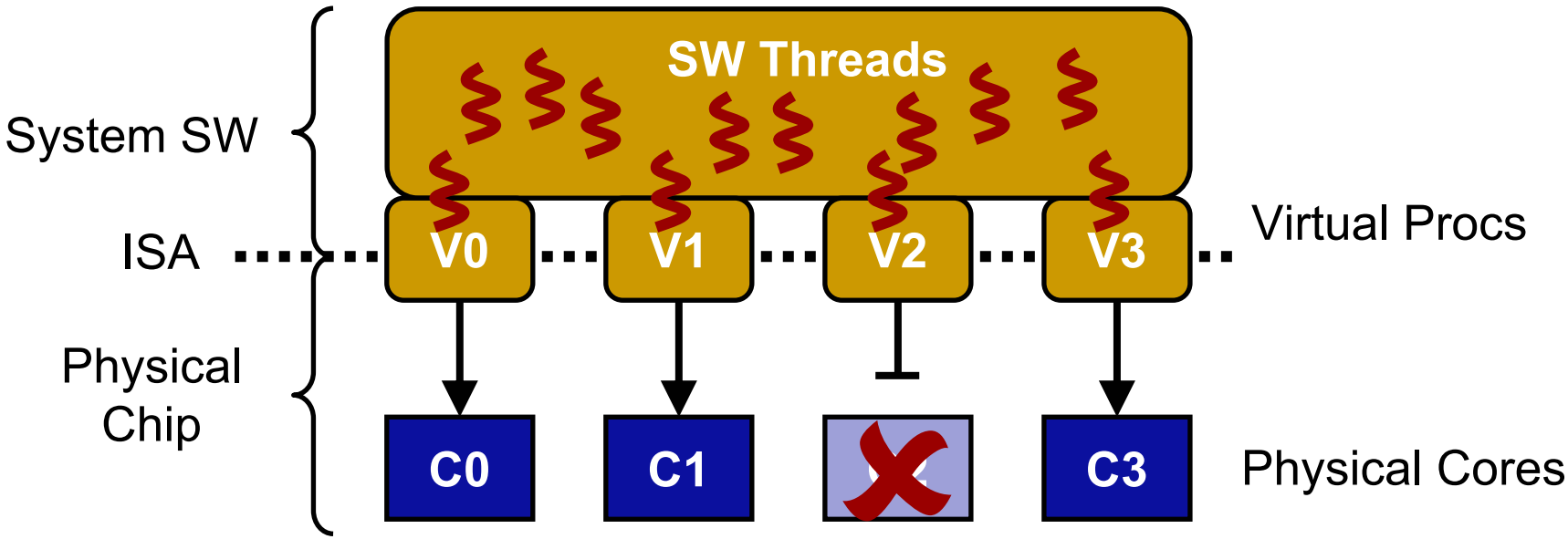


OS reconfiguration cont…

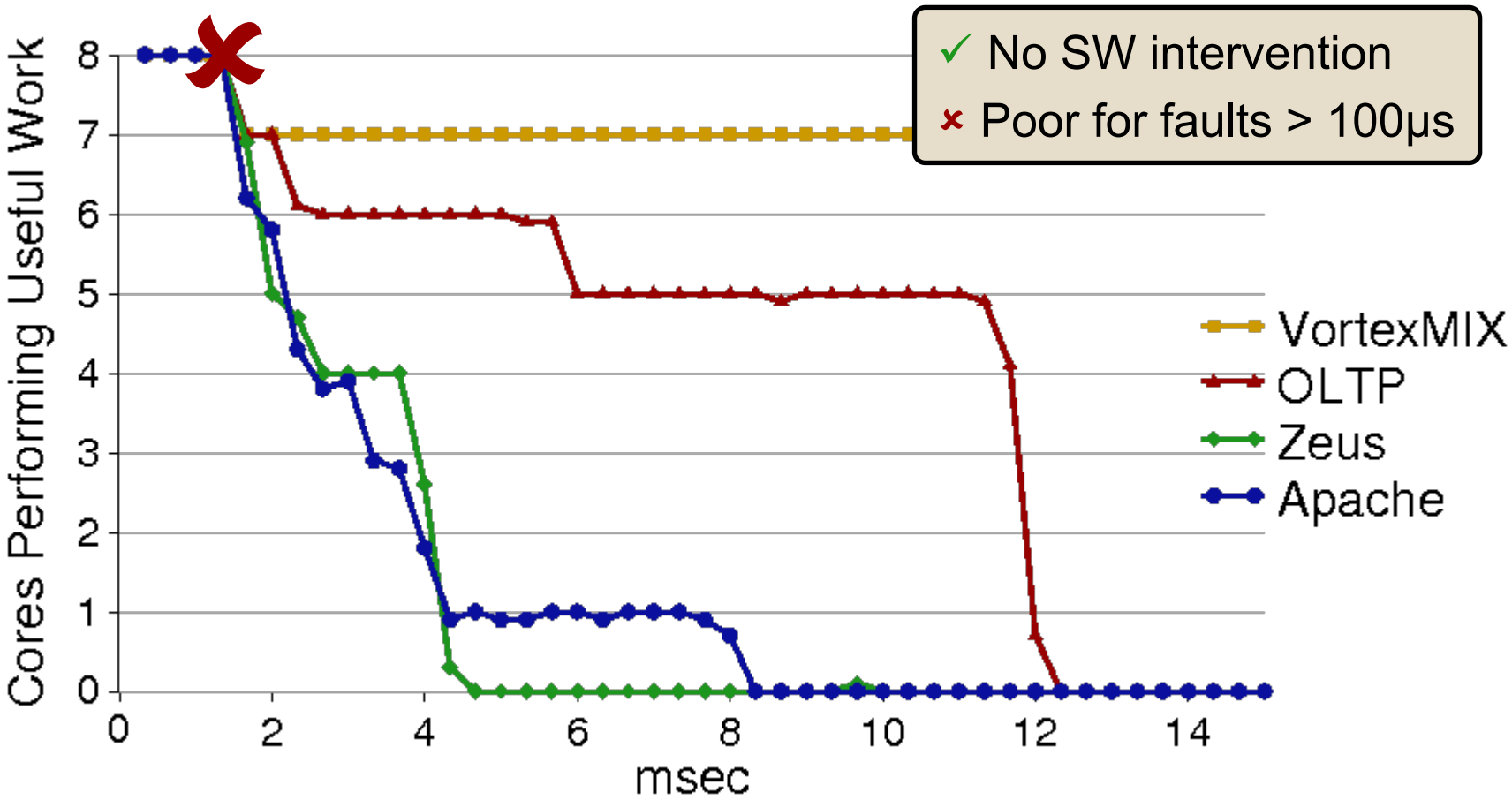


Technique 2: Pause execution

Simply stop executing instructions



Pause execution cont…



Where does that leave us?

Existing techniques leave a big gap: 100µs – 1 sec

- **Thermal variation: ≈ 1ms – 1sec**
- **Software phases: ≈ 10ms – seconds**

What we want:

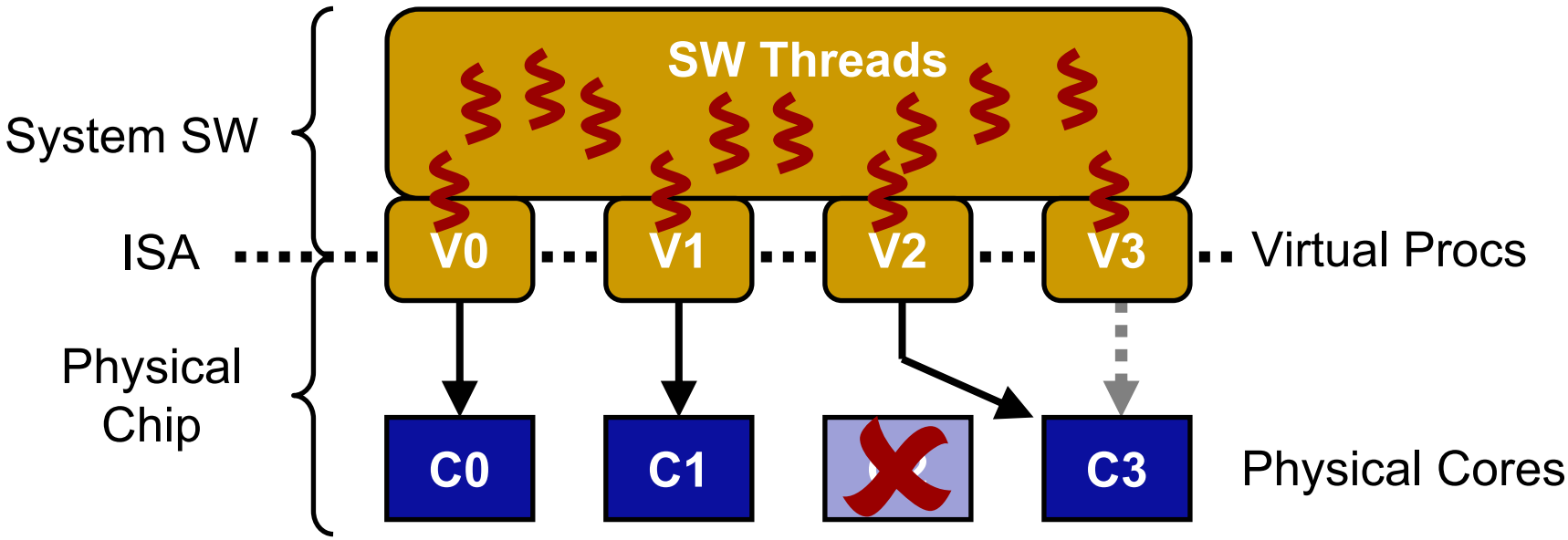
- **Flexible, dynamic mapping of *virtual procs* to cores**
- **But # of physical cores dynamically changes**

Overcommitted System: "OS is configured with more virtual processors than available physical cores"

Proposal: Overcommitted cont…

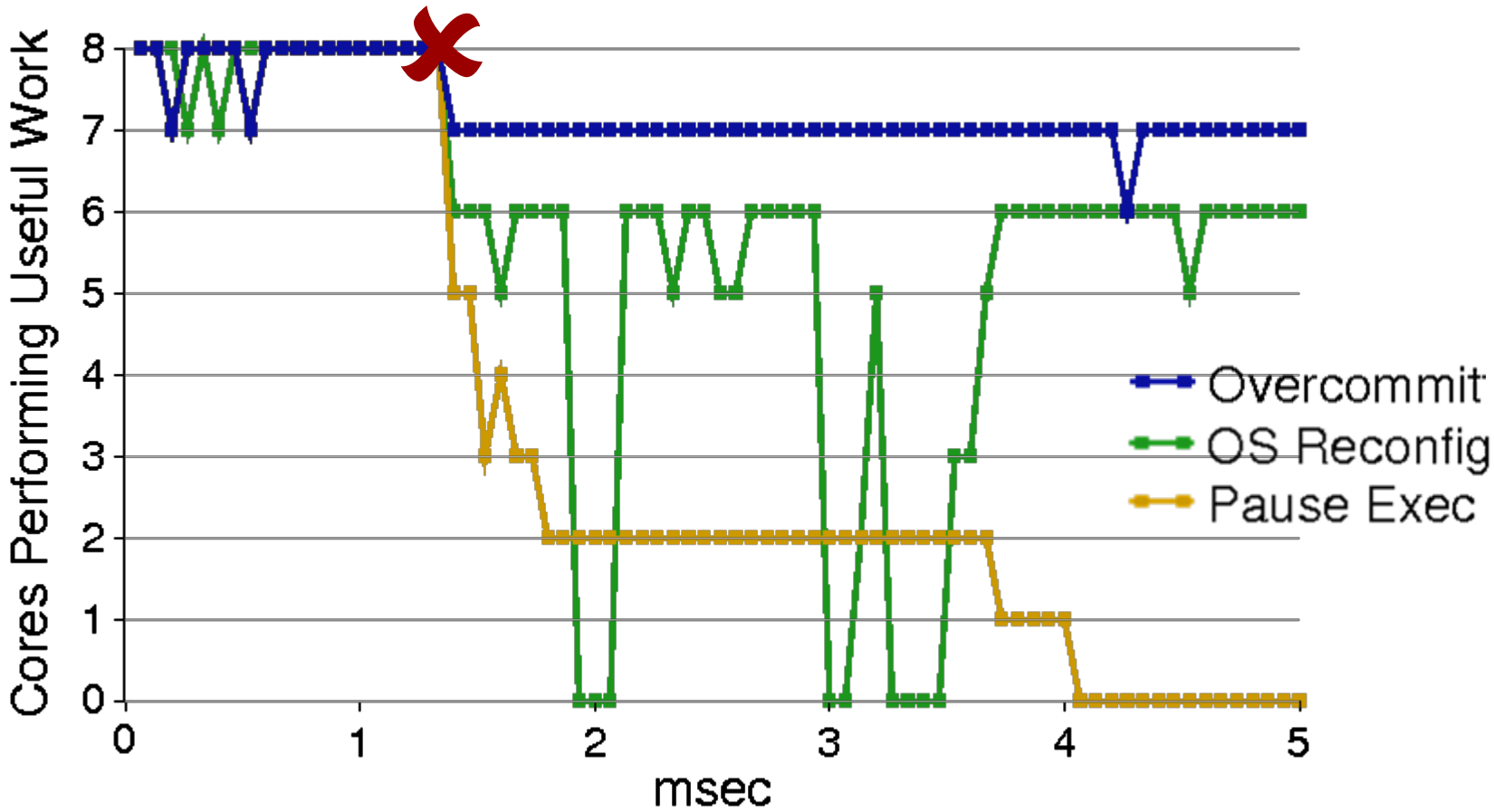
Virtualize physical cores

- **Provide thin HW/firmware layer to manage**
- **HW spin detection helps unmodified OSs [PACT '06]**



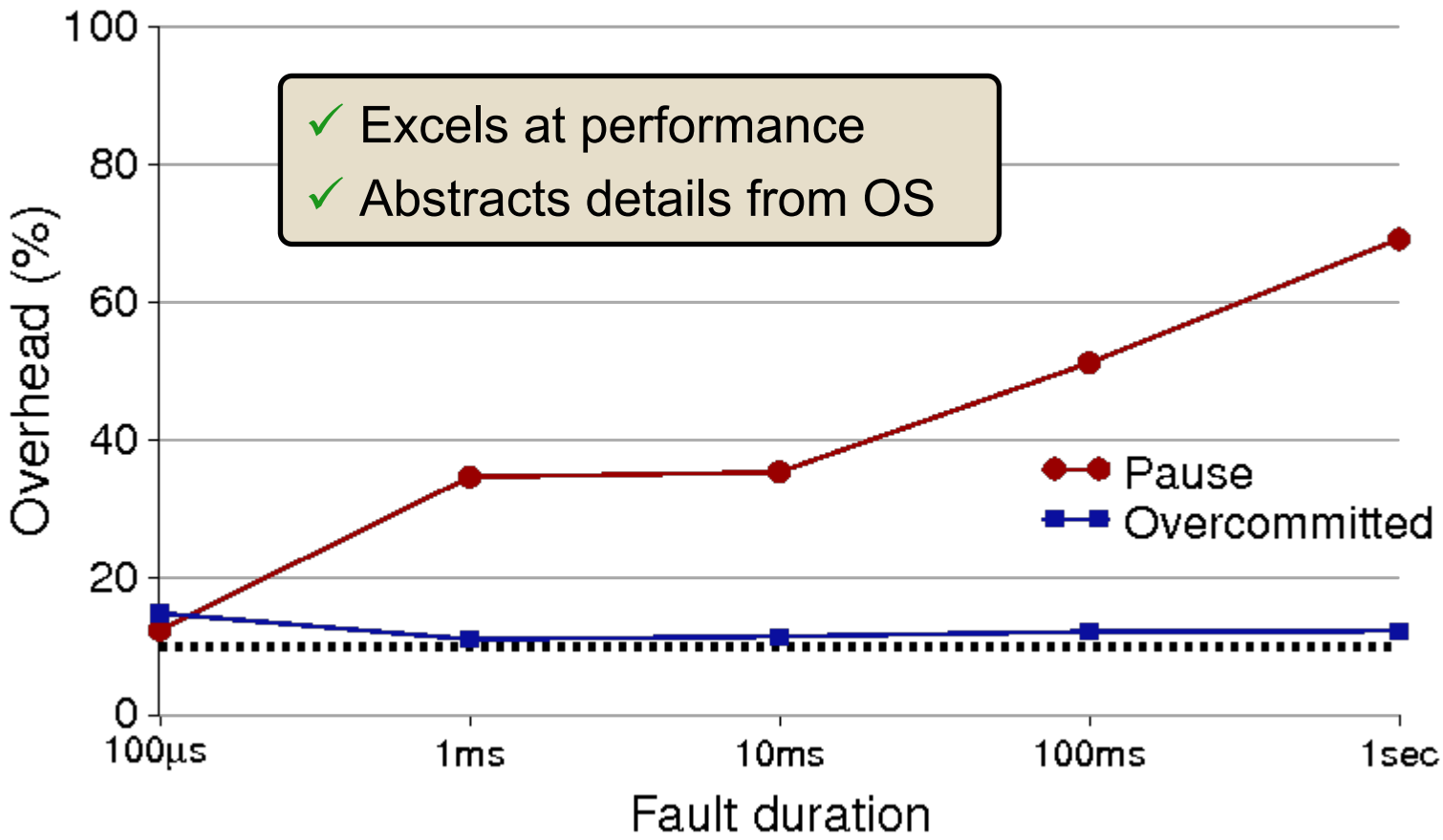
➔ ***Flexibly map virtual procs to physical cores***

Overcommitted results



Overcommitted results cont…

Randomly distributed faults (10% faulty on avg)



Also in the paper…

3rd technique: Spare cores

- **Tension: fault-free overhead vs. # concurrent faults**

More quantitative analysis

- **Throughput, latency, fairness**

More details on assumptions

- **How intermittent faults arise**
- **Why they cause resources to be unavailable**

Summary

Identified *intermittent faults* as important class

- **How to adapt to "intermittent" availability of cores?**
- **Three straightforward techniques fall short**

Propose virtualization with *Overcommitted System*

- **Only technique to excel at performance**
- **Abstract details of operation from SW**

Variety of factors will cause configurations to vary

- **Flexibility of our proposal will be useful**

Thank you!

Questions, comments:

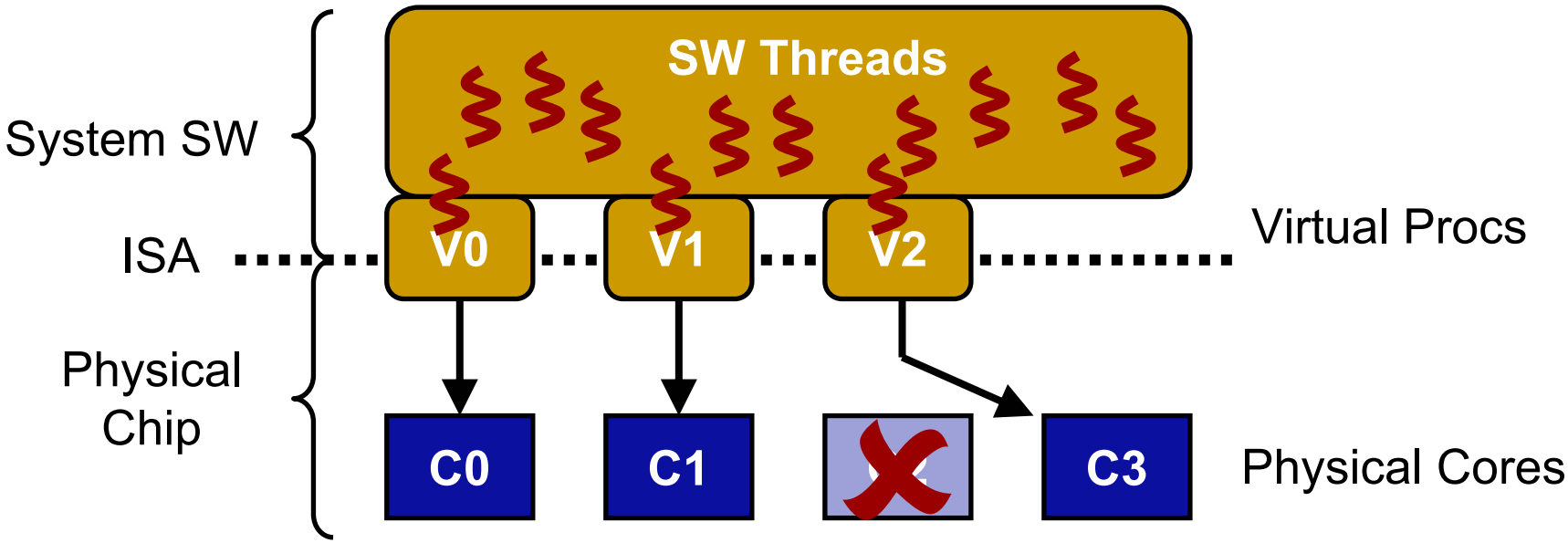
pwells@cs.wisc.edu

http://www.cs.wisc.edu/~pwells

Backup slides

Technique 3: Spare cores

Move computation to a spare core



Assumptions about intermittent faults

Bursty faults will occur 'frequently'

- **Rates from many sources are projected to increase**
  - **We don't know the future rates of faults**
- **Studies such as this can help influence technology trends**

Practical circuits cannot mask all intermittent faults

- **Ways faults manifest ↑**
- **Complexity & overhead of circuit reliability ↑**
- **Must have higher-level mechanisms too**

Suspending use of core

Reduces factors causing faults

- **Temp, voltages, etc. stabilize**

Reduced faults manifesting to higher layers

- **Kinds of faults hardest to protect with circuit techniques are most likely to occur during VT fluctuations**
- **Reducing opportunity for faults reduces number that must be corrected**

Likely to be useful for other purposes

- **E.g., ease fine-grained reconfig techniques**
- **Other undiscovered uses**

Technique summary

	Throughput	Latency	Fairness	Fault-free	Concurrent	Complexity	Timescales
Pause Execution	✗	✗	✗	✓	✓	L	✗
Spare Cores	✓	✓	✓	✗	✗	M	✓
SW Reconfig	✗	✗	✗	✓	✗	H	✗
Overcommitted	✓	✓	✓	✓	✓	M	✓