TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science Institute of System Architecture, Operating Systems Group

MIXED-MODE MULTICORE RELIABILITY

PHILIP M. WELLS, KOUSHIK CHAKRABORTY, GURINDAR S. SOHI

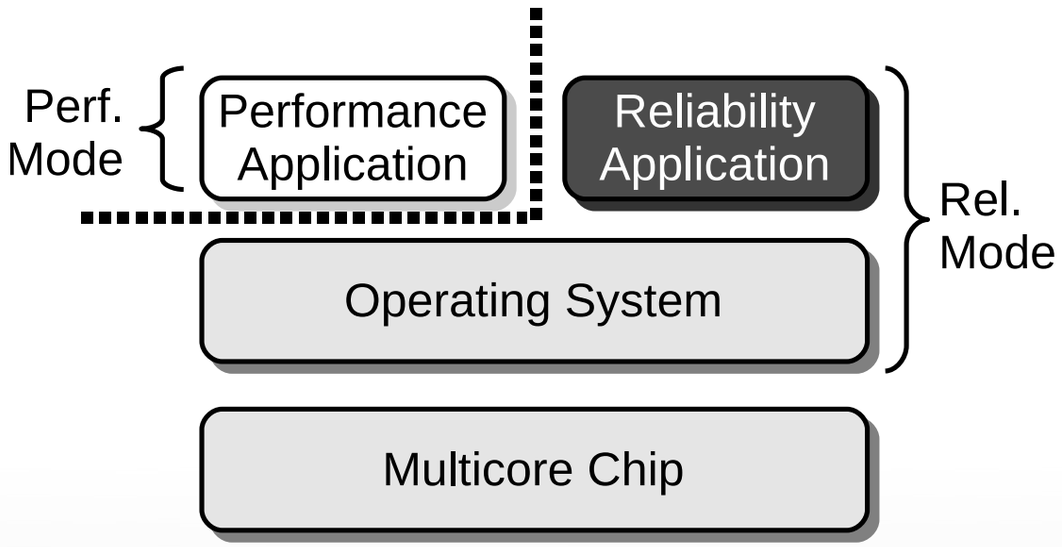
CARSTEN WEINHOLD

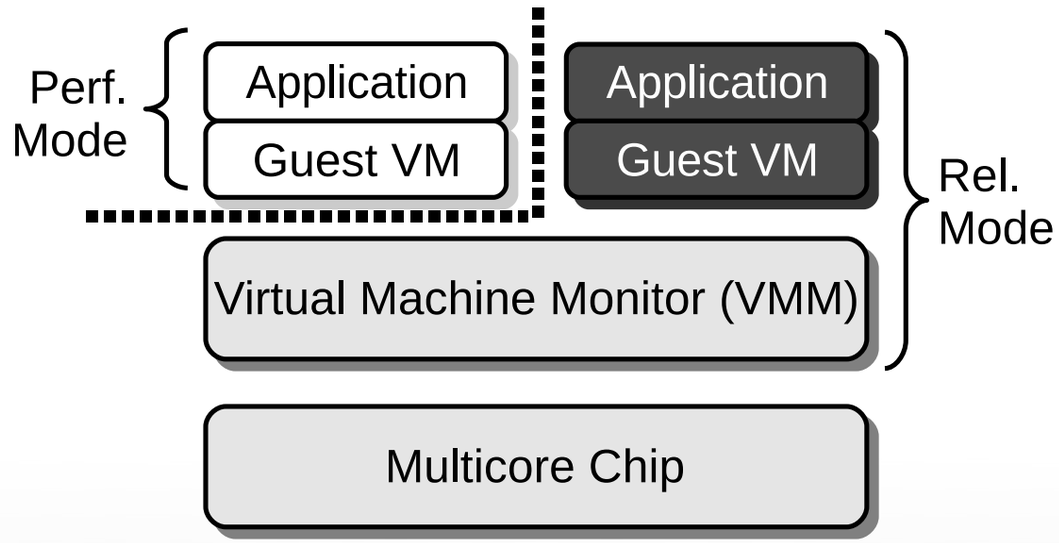
TECHNISCHE UNIVERSITÄT DRESDEN

MOTIVATION

- Expected hardware development:
  - Increased rate of hardware failures
  - CPUs become less reliable
- Use *Dual-Modular Redundancy (DMR)* to maintain high reliability
- Problems:
  - Expensive (up to 4x slowdown)
  - Some applications don't need it

ARCHITECTURE





CHALLENGES

- Paper proposes mixed multicore:
  - Redundancy support in hardware
  - Switch on / off as needed
- Simple idea, but two key challenges:
  - Protect high-reliability applications from faults occurring in non-DMR operation
  - Scheduling of cores complicated by desire to execute more high-performance threads

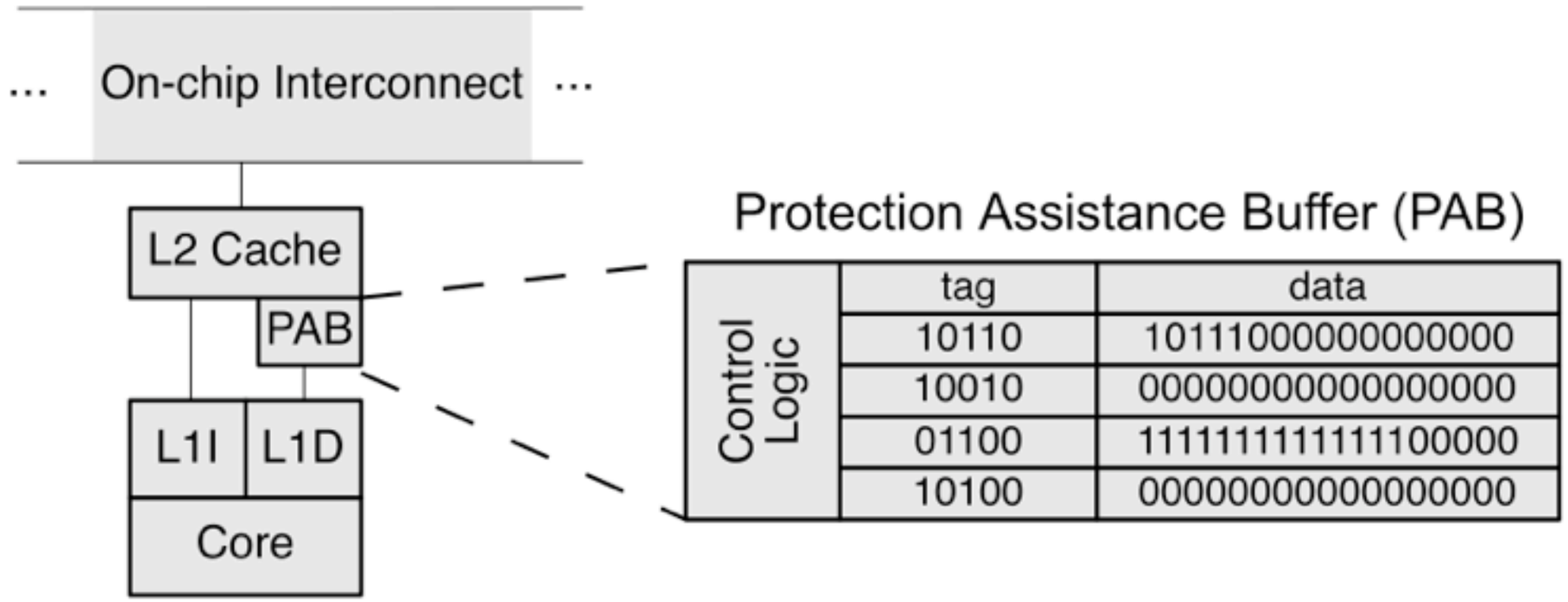
PROTECTION

- Memory and register state of DMR applications need to be protected
- Standard mechanisms (page tables, ...) only sufficient in DMR mode
- Some redundancy in non-DMR mode:
  - OS / VMM executed in reliable mode
  - Application executed without redundancy ...
  - ... except: permission for any store checked

HW SUPPORT

- Protection assistance table (PAT):
  - Similar to inverse page table
  - 1 permission bit per physical page (allow reliable mode only / any software)
- Protection Assistance Buffer (PAB):
  - Like TLB, caches PAT entries
  - Consulted in non-reliable mode only
  - Both TLB and PAB must indicate permission

PAB


... On-chip Interconnect ...
L2 Cache
PAB
L1I
L1D
Core
Protection Assistance Buffer (PAB)
Control Logic
tag
data
10110
1011100000000000
10010
0000000000000000
01100
1111111111100000
10100
0000000000000000

REUNION

- Based on previous work „Reunion“
- Loose lock stepping:
  - 2 cores: one vocal, one mute
  - Additional CHECK stage in pipeline
  - Fingerprint compared before commit
- Implemented below ISA
- VCPUs presented to system software

MMM-IPC: SWITCH

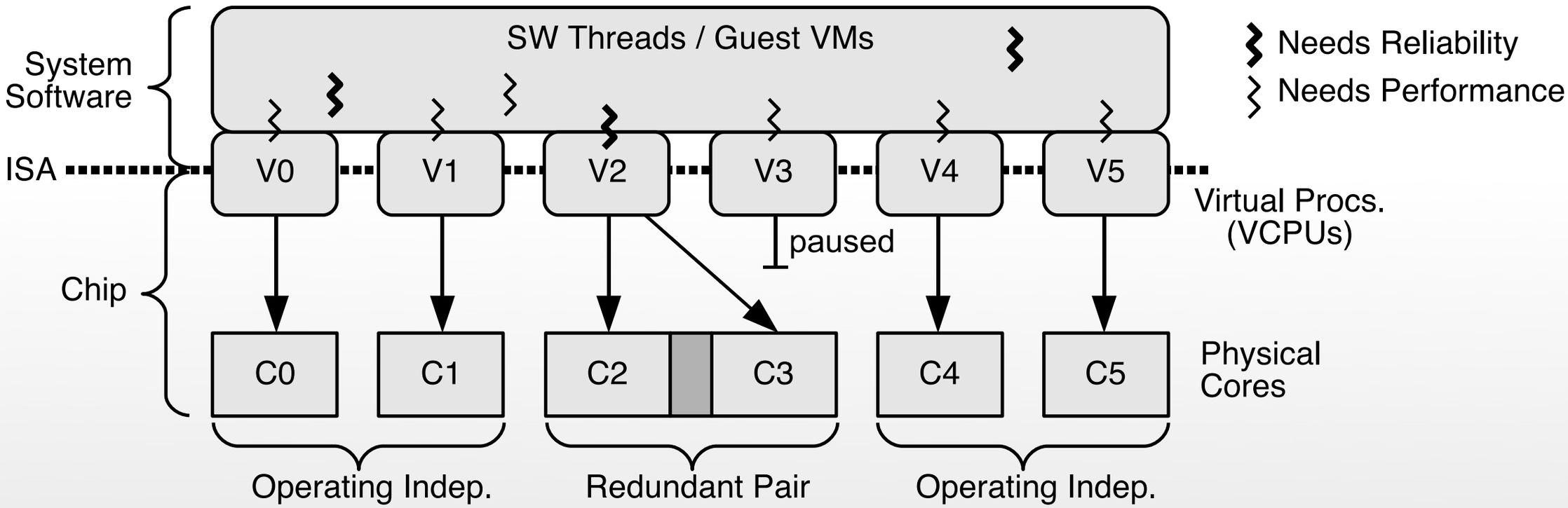
- Leaving DMR mode:
  - Both cores save copies of privileged state
  - Use separate „scratchpad space“ in physical memory

MMM-IPC: SWITCH

- Enter DMR mode:
  - Vocal core:
    - Stores current user state to cache
    - Loads previous privileged state
  - Mute core loads:
    - Its own copy of previous privileged state
    - Current user registers from cache
    - Vocal cores privileged sate for comparison
- Both privileged states must match

MMM-TP

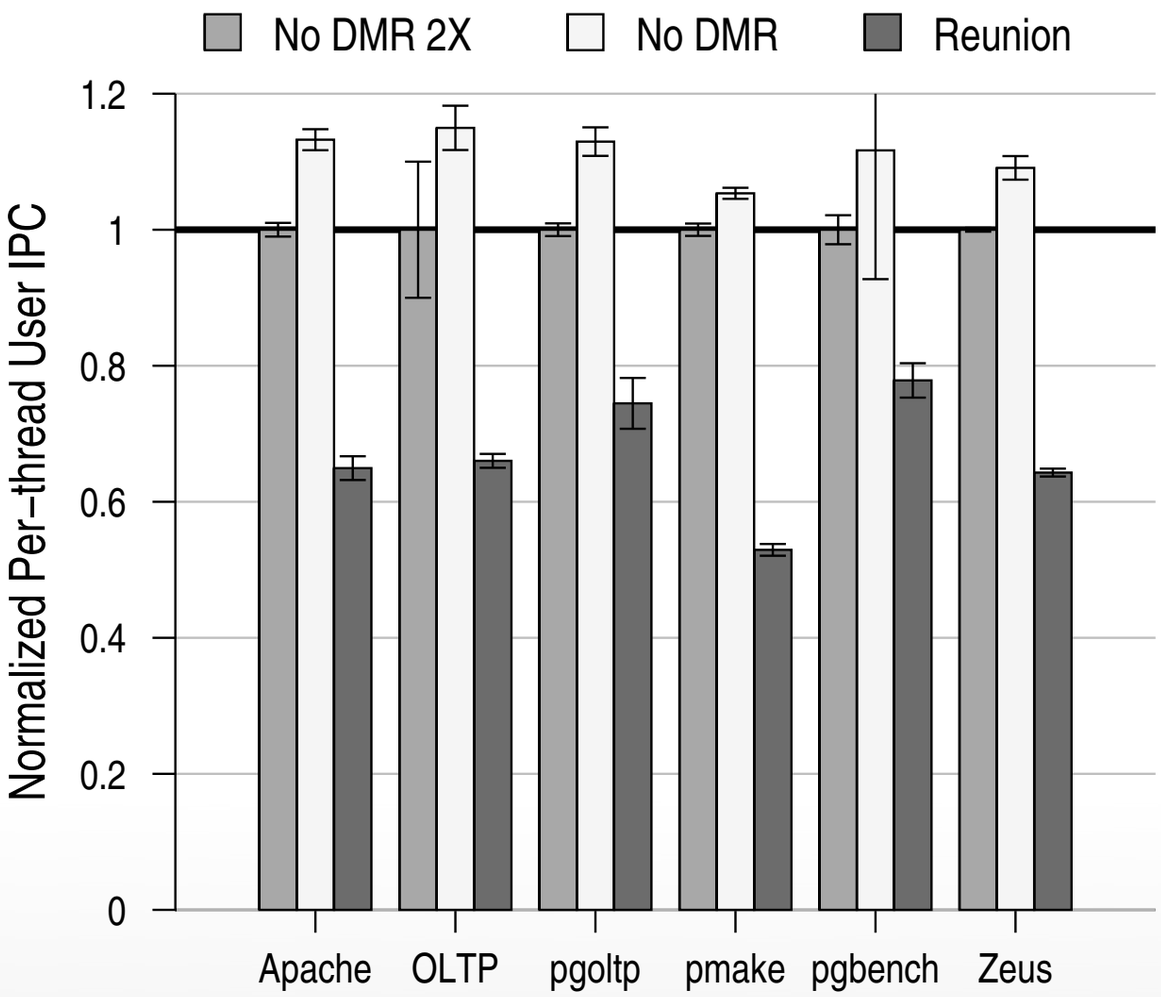
- MMM-TP solution can utilize all cores
- Overcommitting of VCPUs
- Redundant execution can force some VCPUs to pause



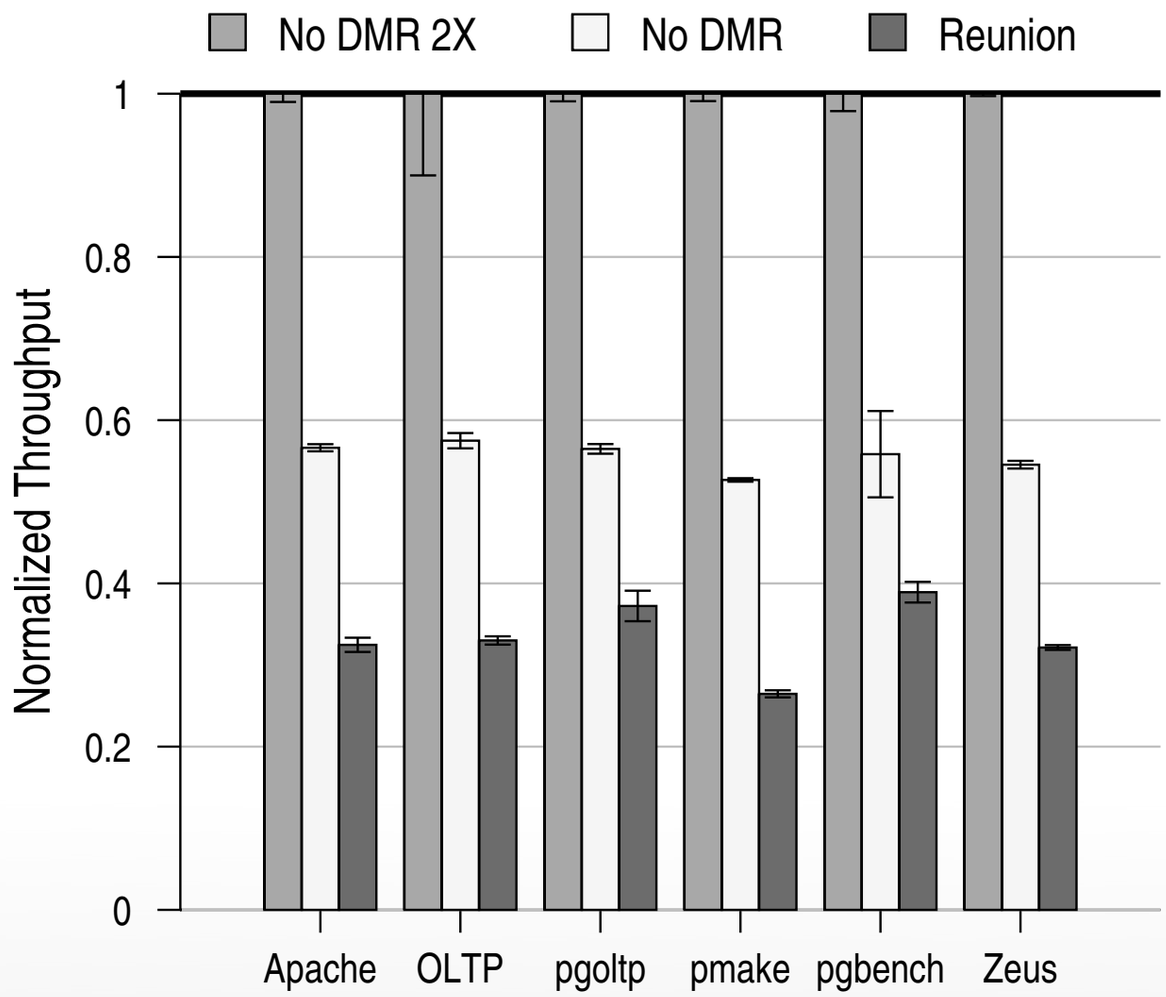
EVALUATION

- MMM-IPC and MMM-TP simulated:
  - „Simics MAI“ for model
  - Cycle-accurate module for processor and memory hierarchy
- Server benchmarks evaluated
- Only VM architecture benchmarked
- Estimates for architecture that runs OS + applications directly

DMR OVERHEAD

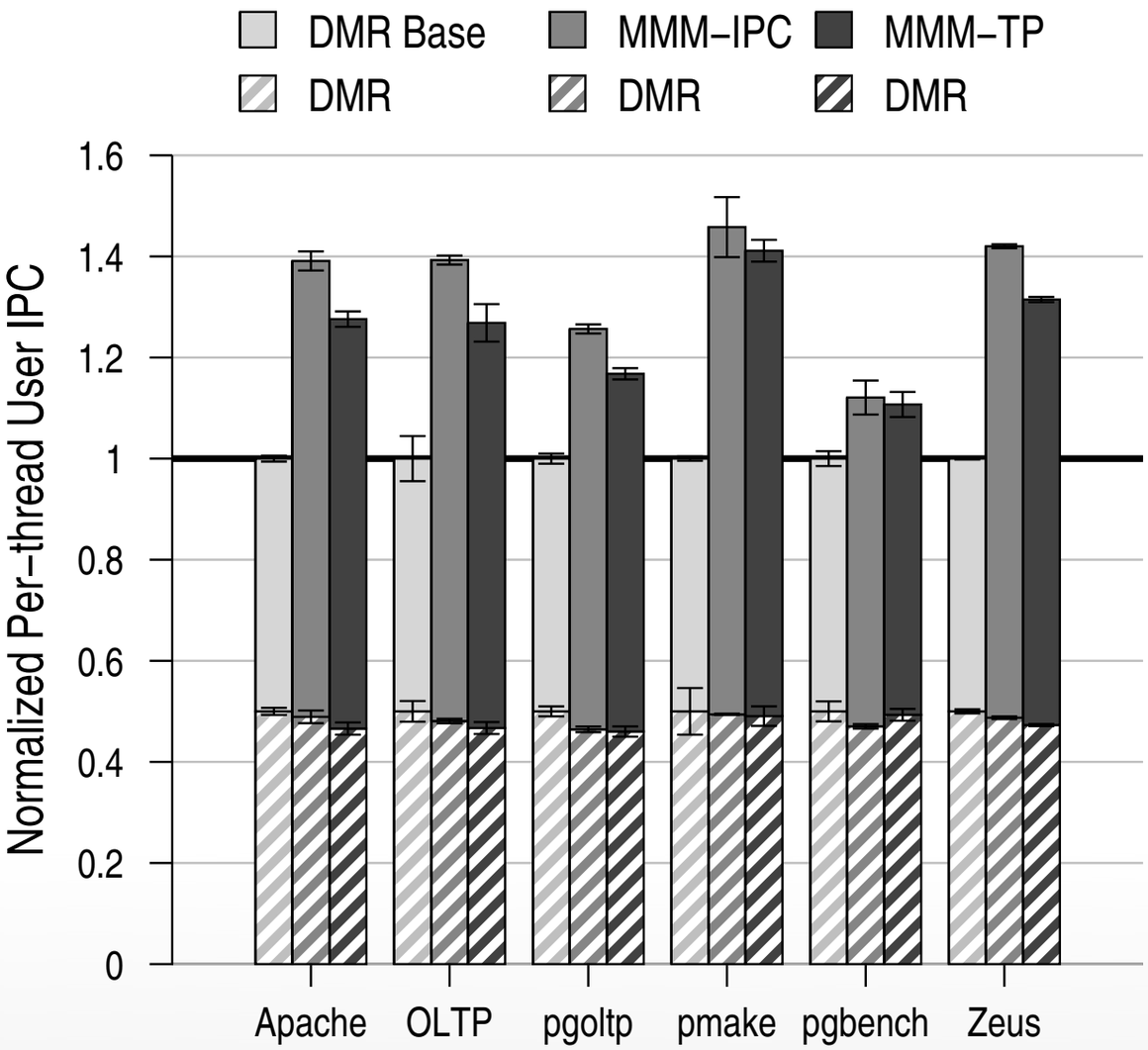


(a) Single Thread Performance

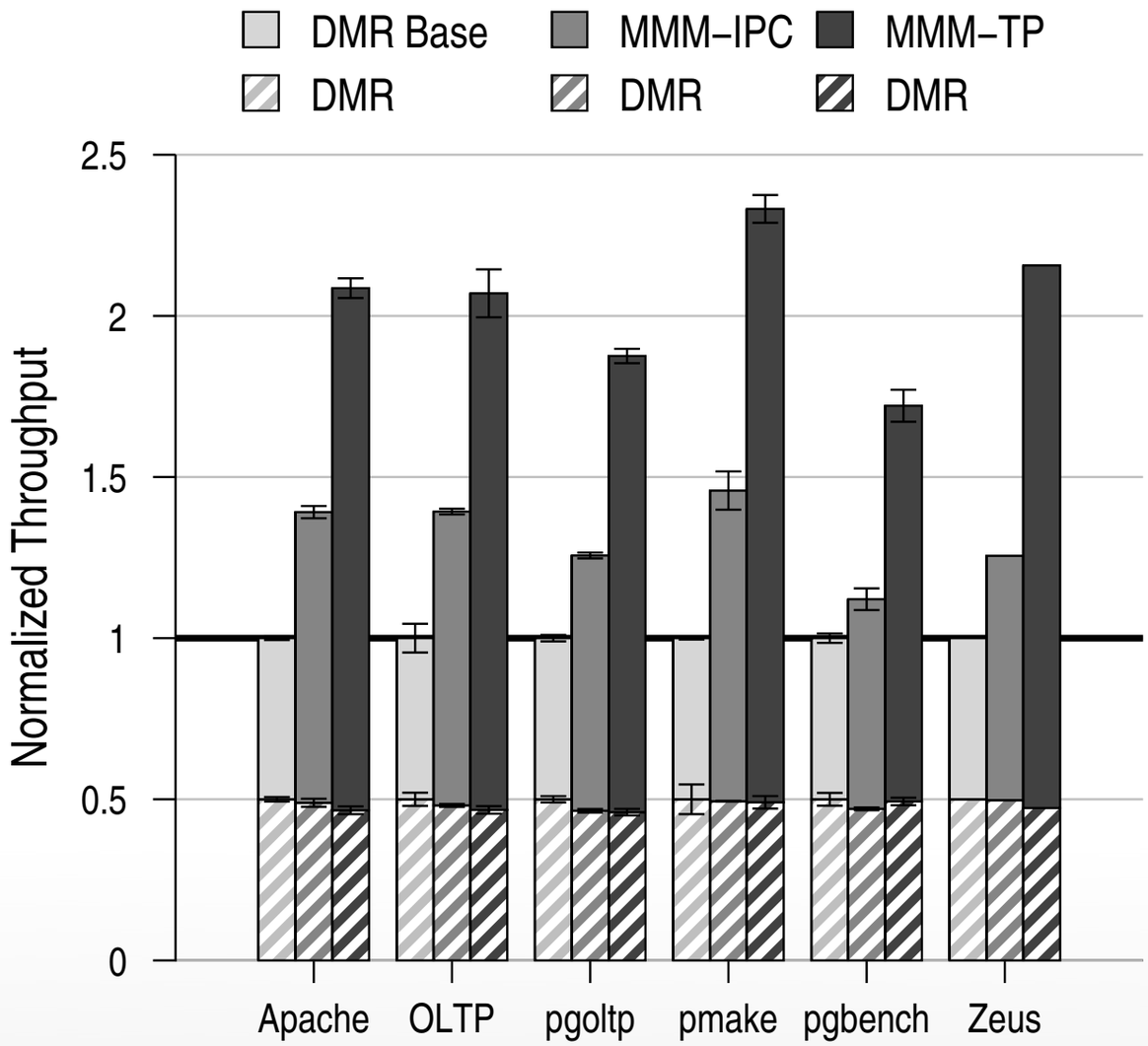


(b) Overall Throughput

MMM BENEFIT



(a) Single Thread Performance



(b) Overall Throughput

MMM OVERHEAD

	Enter DMR	Leave DMR
Apache	2.4k	10.4k
OLTP	2.4k	10.3k
pgoltp	2.3k	10.2k
pmake	2.2k	9.9k
pgbench	2.3k	10.2k
Zeus	2.4k	10.3k

Mixed-mode switching overhead

	User Cycles	OS Cycles
Apache	59k	98k
OLTP	218k	52k
pgoltp	210k	35k
pmake	312k	47k
pgbench	554k	126k
Zeus	65k	220k

Cycles between kernel–user mode switches

SUMMARY

- Multicore that supports mixed operation of DMR and non-DMR applications
- Improvements over standard DMR:
  - MMM-IPC: 34 - 48%
  - MMM-TP: 2.5 - 4x
- Transparent (VCPUs, HW scheduler)

DISCUSSION

- How expensive is this in a microkernel-based system?
  - Mostly evaluated for VM workloads
  - Estimates for mode transition costs high
- How does it look without the hardware scheduler / VCPU abstraction?
- Energy … ? Figures … ?