Leveraging Simultaneous Multithreading for Adaptive Thermal Control

James Donald & Margaret Martonosi

Princeton University

Outline

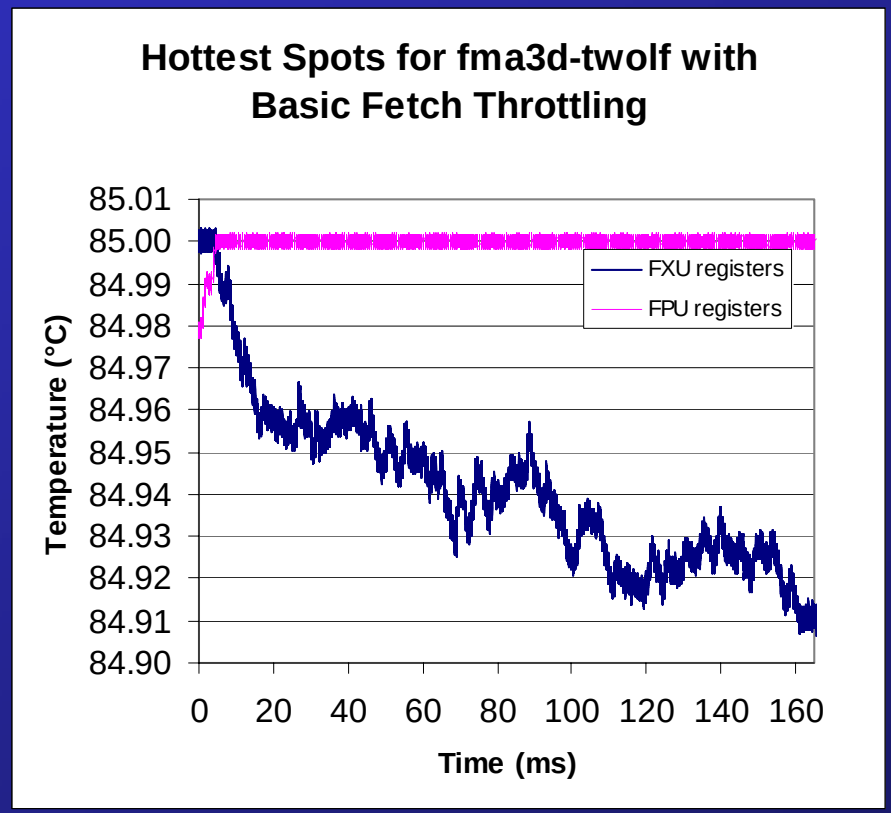
- Background & Problem Description
- Related Work
- Experimental Setup & Benchmarks
- Adaptive Fetch Algorithm
- Adaptive Register Renaming
- Future Work & Conclusions

Background

- Temperature-Aware Design
  - Increasingly relevant w/ greater transistor densities
  - Tools e.g. HotSpot
- Applied to multithreaded architectures (SMT, CMP)
  - Applicable to prevalent emerging architectures
  - Provides opportunity for smarter thermal management

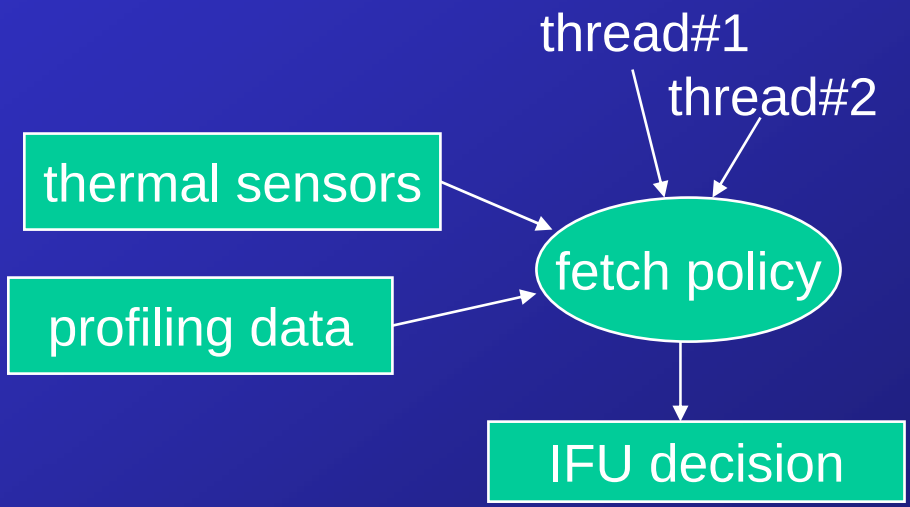
Localized Thermal Hotspots

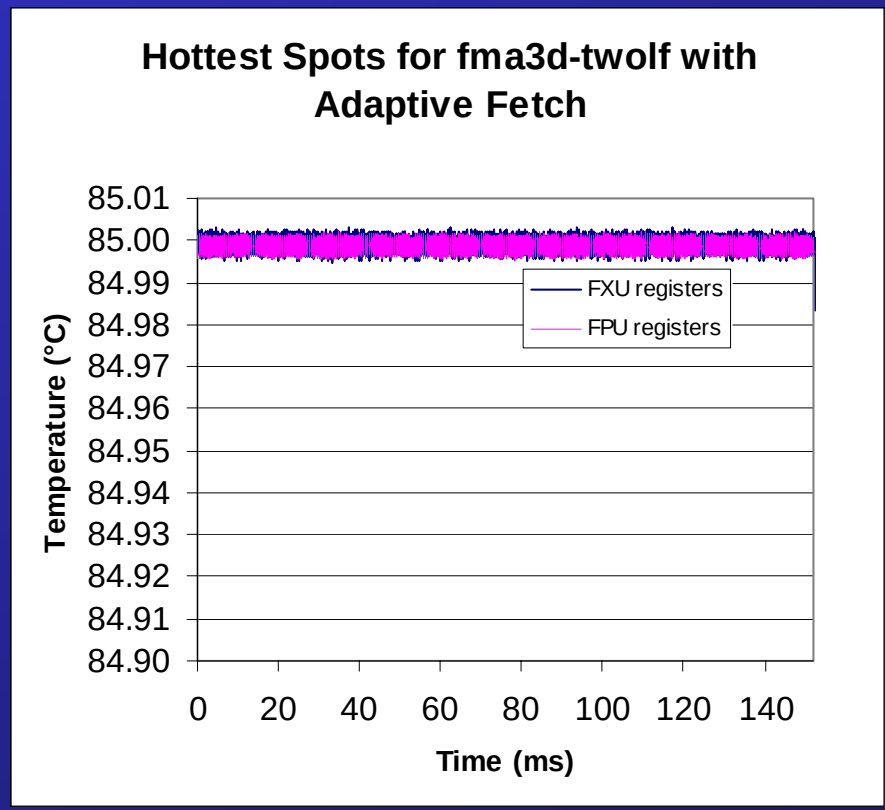
- A *single* hot spot limits performance.
- Global DTM (e.g. DVFS) handles this at expense of entire core's performance.



Proposed Solution

- Selectively fetch instructions to mitigate single hotspot's severity.
- Our experiments: two hotspots monitored.





Outline

- Background & Problem Description
- Related Work
- Experimental Setup & Benchmarks
- Adaptive Fetch Algorithm
- Adaptive Register Renaming
- Future Work & Conclusions

Related Work

- M. Powell, M. Gomaa and T. N. Vijaykumar. “Heat and Run: Leveraging SMT and CMP to Manage Power Density Through the Operating System”. ASPLOS 2004.
- Y. Li, Z. Hu, D. Brooks and K. Skadron. “Performance, Energy, and Thermal Considerations for SMT and CMP Architectures”. HPCA 2005.

Outline

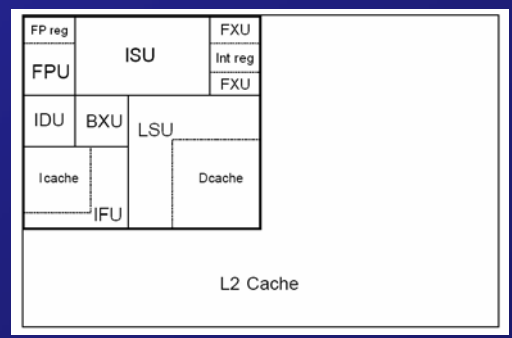
- Background & Problem Description
- Related Work
- Experimental Setup & Benchmarks
- Adaptive Fetch Algorithm
- Adaptive Register Renaming
- Future Work & Conclusions

Simulation Setup

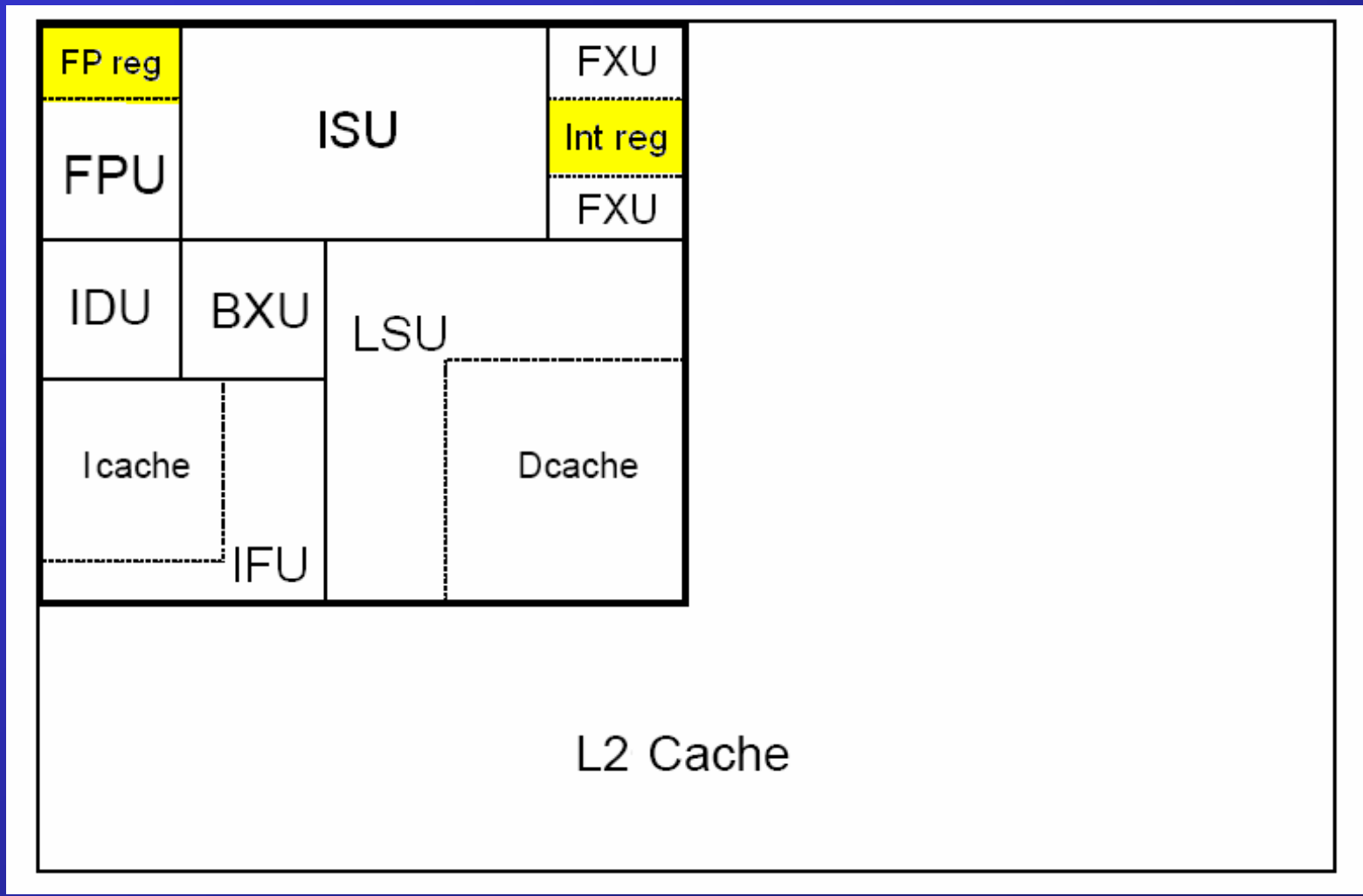
- Turandot PowerPC simulator w/ SMT
- PowerTimer + leakage power estimation
- HotSpot 2.0

Machine Parameters

Processor Configuration	
Process Technology	0.18 µ
Supply Voltage	1.2 V
Clock Rate	1.4 GHz
Organization	single-core, 2-context SMT
Functional Units	2 FXU, 2 FPU, 2 LSU, 1 BRU
Physical Registers	120 GPR, 90 FPR
Thermal Control	fetch throttling



Processor Floorplan



SPEC Benchmark Heat Behaviors

benchmark	type	function	FXUreg	FPUreg
ammp	FP	computational chemistry		
applu	FP	CFD/physics		
fma3d	FP	mechanical response		
galgel	FP	CFD		
gcc	int	compiler		
gzip	int	compression		
mcf	int	transportation sched		
mesa	FP	3-D graphics		
parser	int	word processing		
twolf	int	lithography routing		

Test Workloads

SMT workload	Mix type
ammp-gzip	FP-integer
ammp-mcf	FP-integer
applu-parser	FP-integer
applu-twolf	FP-integer
fma3d-galgel	FP-FP
fma3d-twolf	FP-integer
galgel-mesa	FP-integer
gcc-mesa	integer-FP
gcc-parser	integer-integer
gzip-mcf	integer-integer

- Tests a wide range of integer/floating-point and hot/hotter mixes.

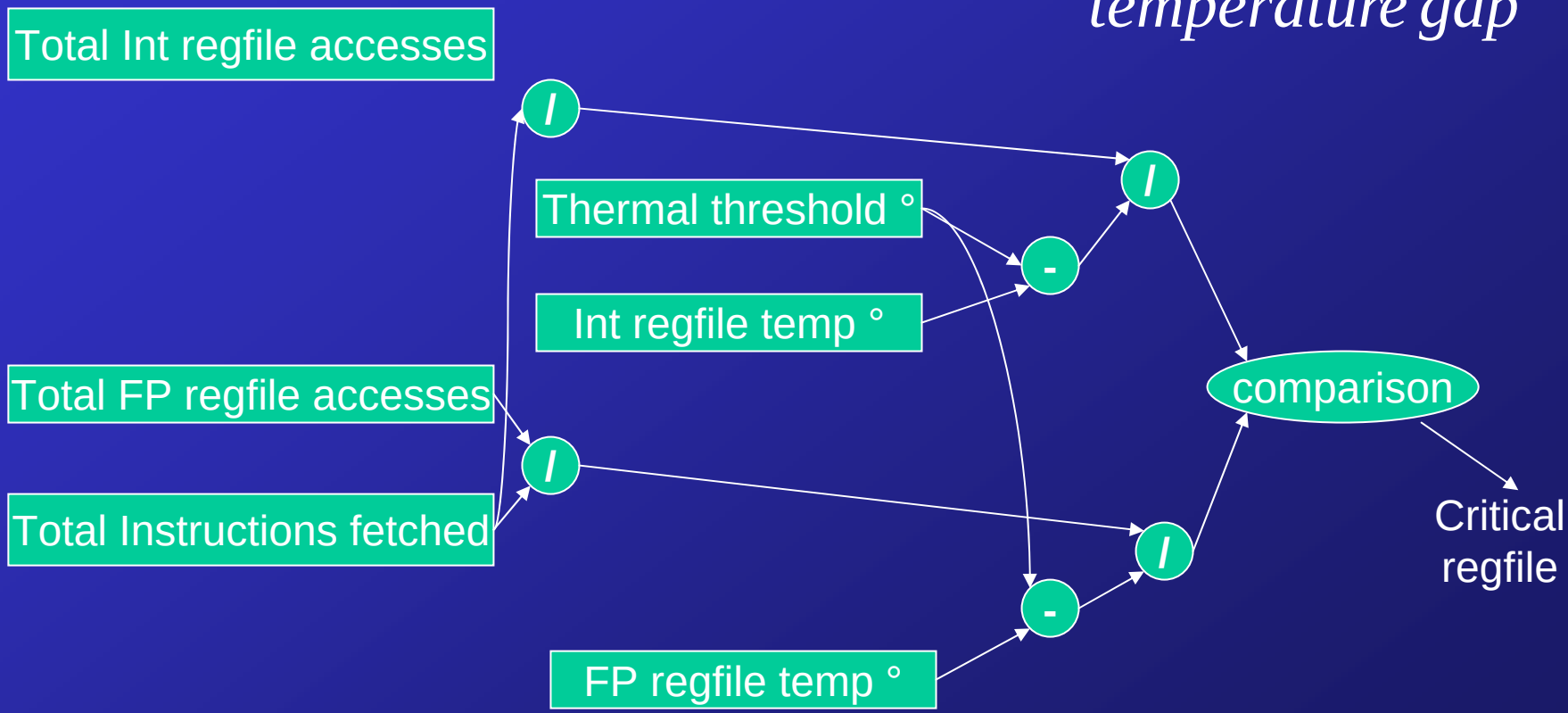
Outline

- Background & Problem Description
- Related Work
- Experimental Setup & Benchmarks
- Adaptive Fetch Algorithm
- Adaptive Register Renaming
- Future Work & Conclusions

Adaptive Fetch Algorithm (a)

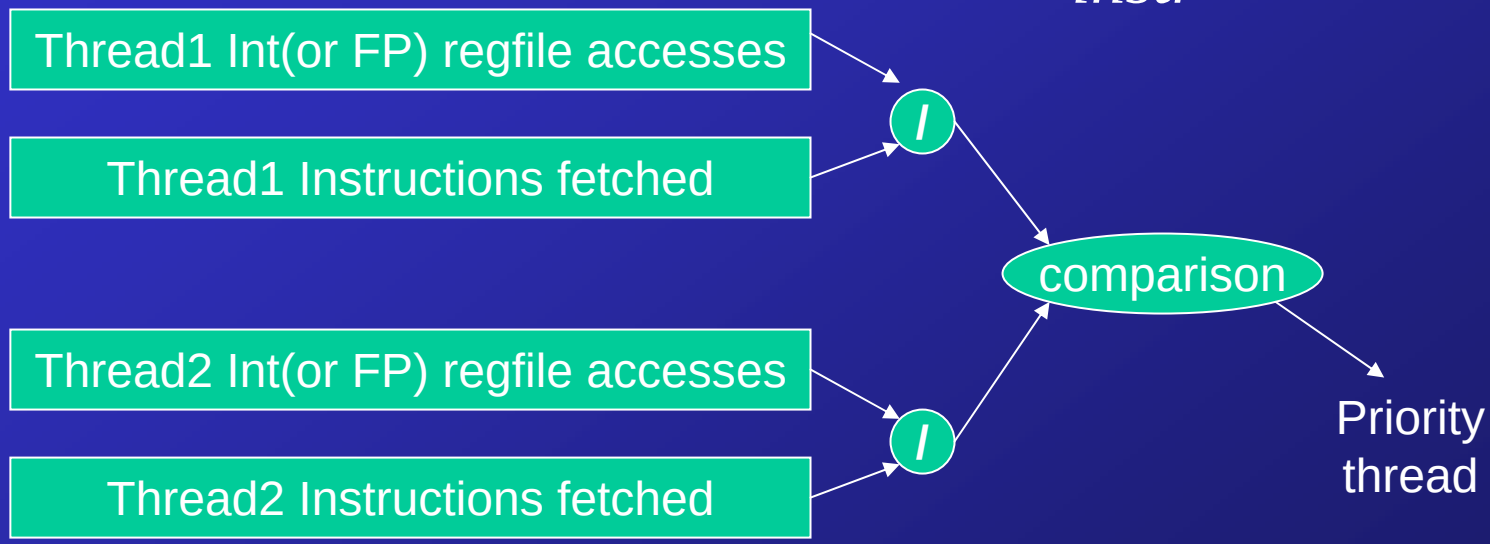
- Decide which hotspot to mitigate:

$$rate\ to\ overheat = \frac{regfile\ access\ rate}{temperature\ gap}$$



Adaptive Fetch Algorithm (b)

- Favor best thread for the identified hotspot.
  - Compare which thread has lower: $\frac{accesses}{instr}$



Moderate policy: 50% of cycles default to round-robin
Aggressive policy: 12.5%

Metrics

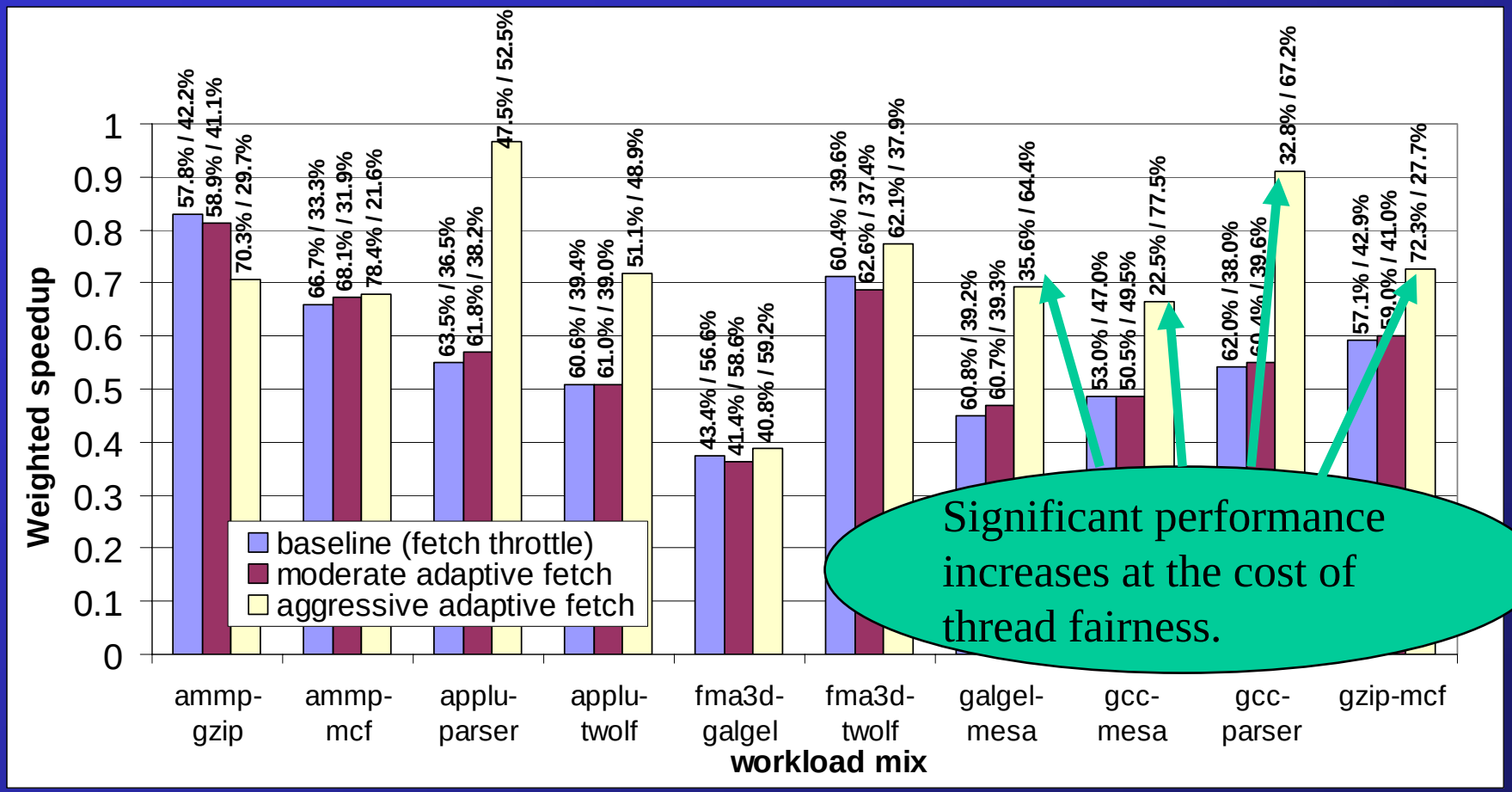
- Performance – IPC and weighted speedup

$$Weighted\ Speedup = \sum \frac{IPC_{SMT}[i]}{IPC_{normal}[i]}$$

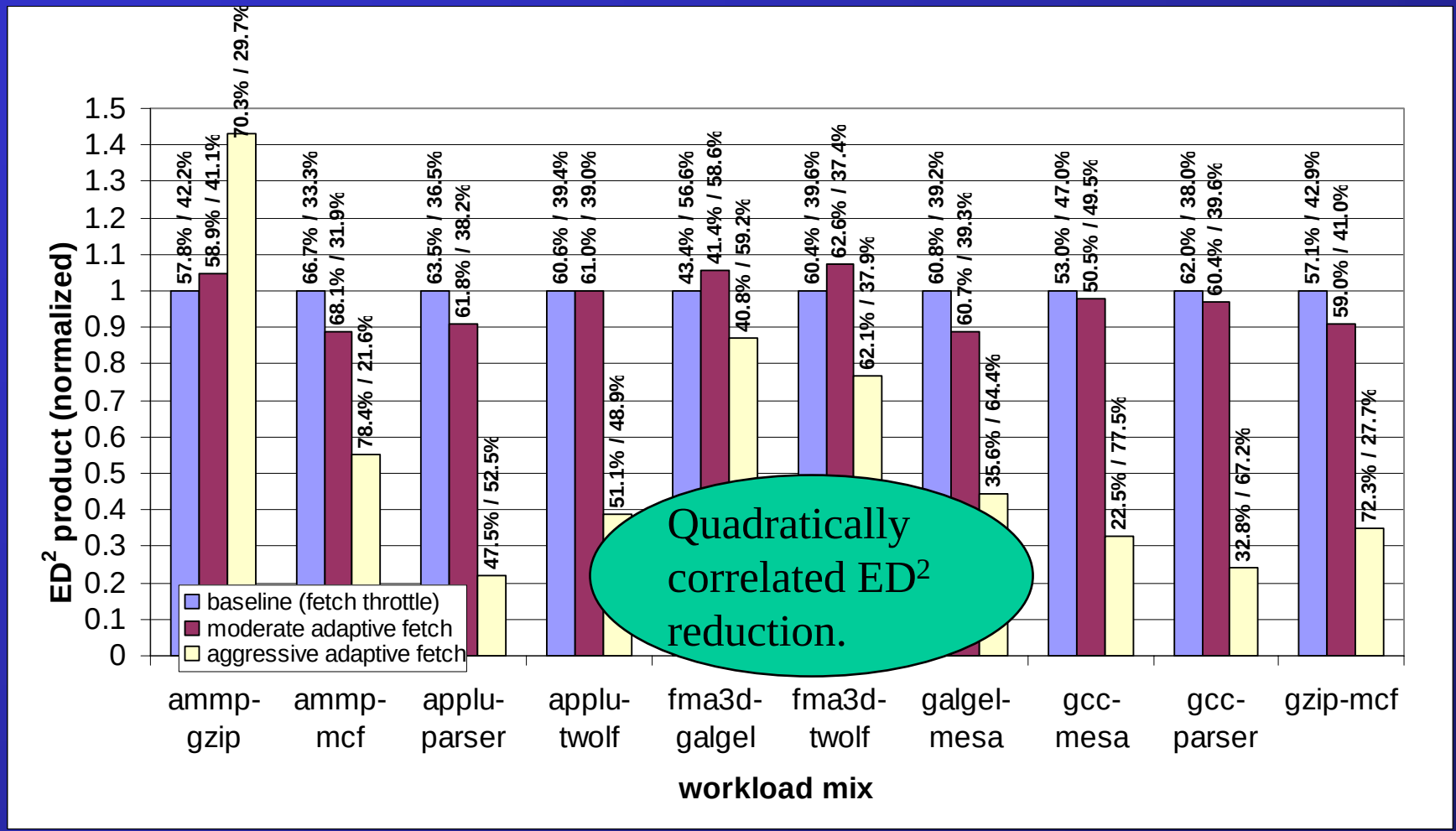
- Power-performance – ED^2 product

- Thread fairness (percentages of retired instructions)

Performance Results



Results – ED^2 Product



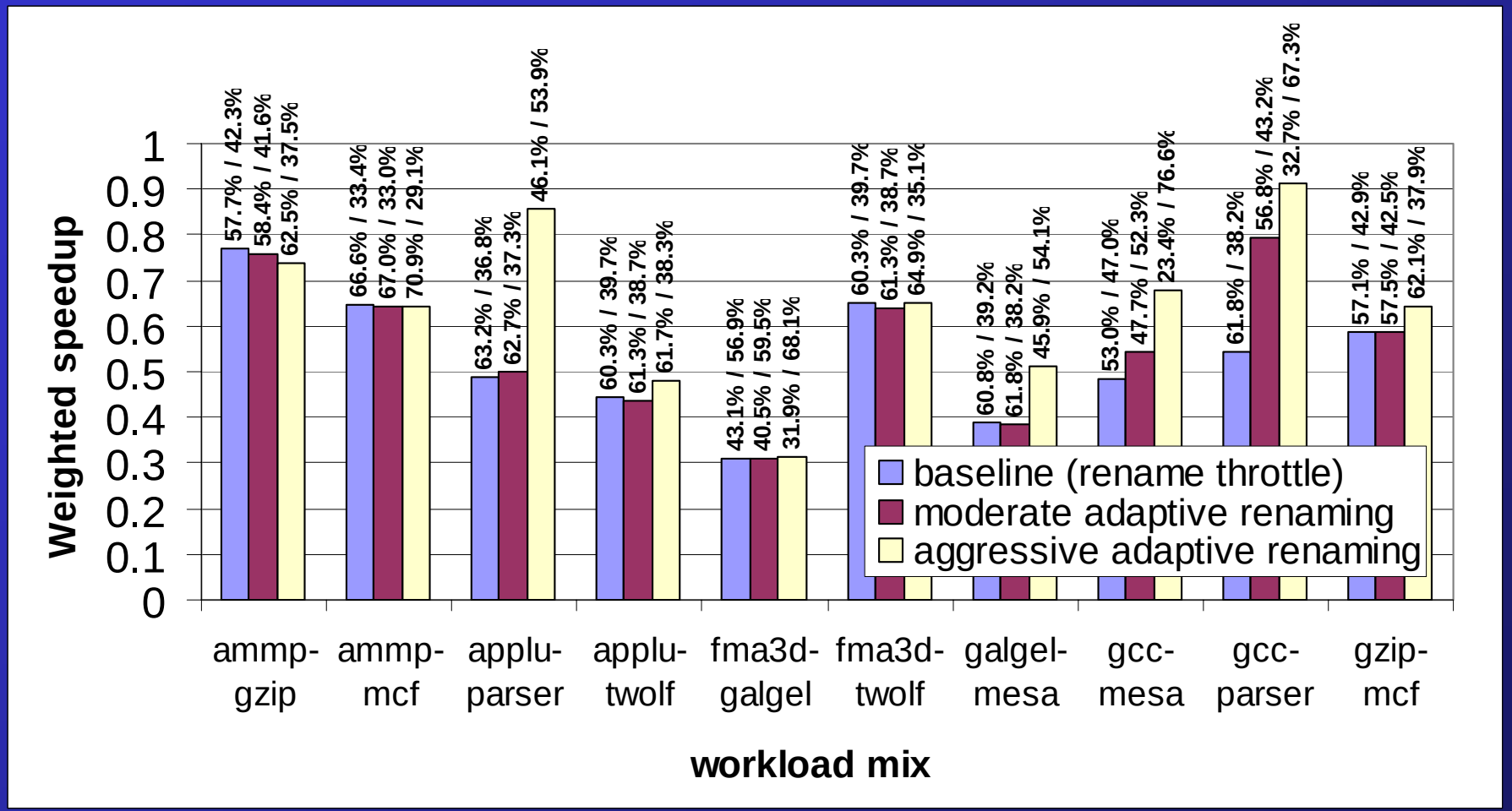
Outline

- Background & Problem Description
- Related Work
- Experimental Setup & Benchmarks
- Adaptive Fetch Algorithm
- Adaptive Register Renaming
- Future Work & Conclusions

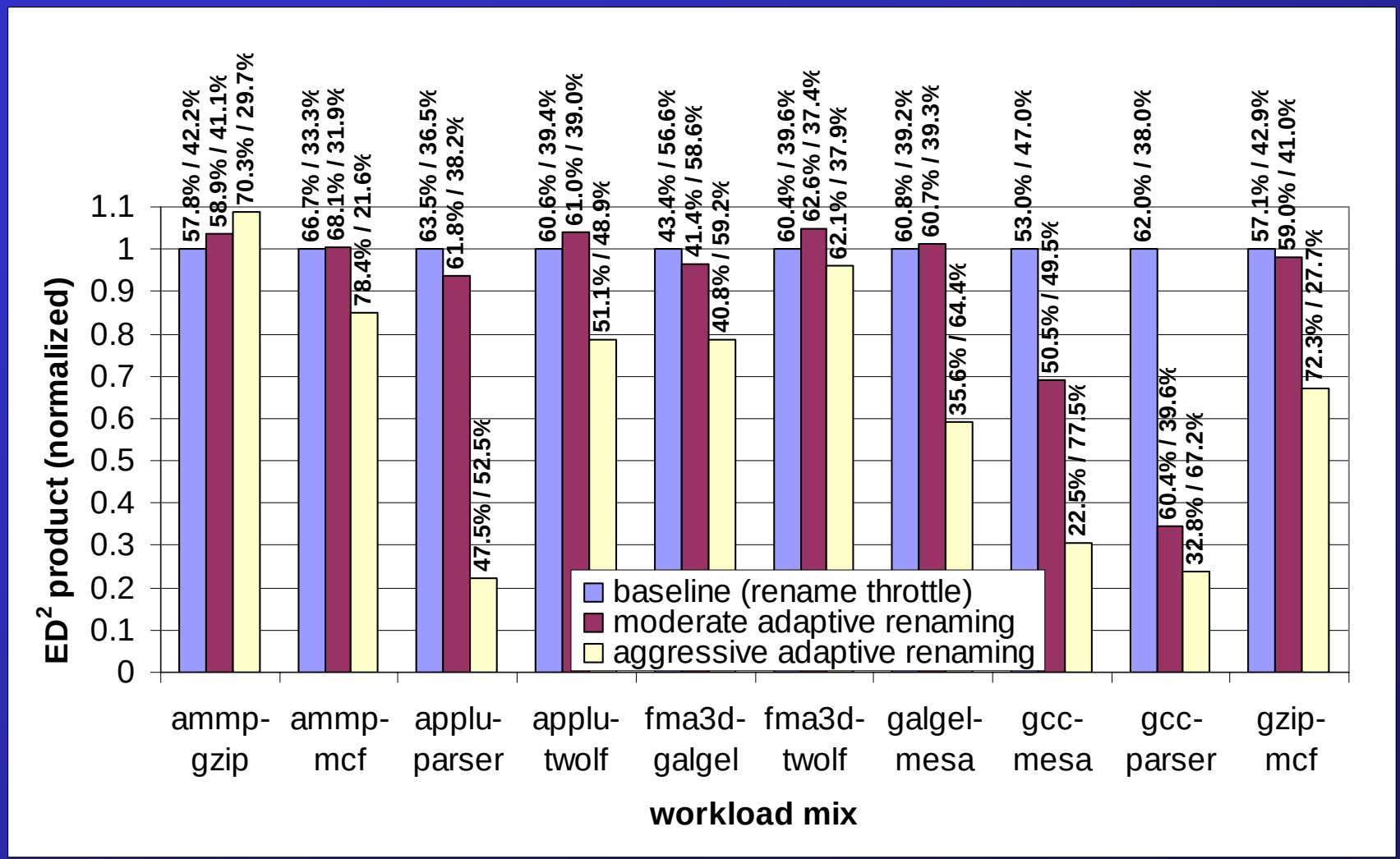
Adaptive Register Renaming

- Same counter-based thread priority decision algorithm.
- Difference: Decision affects thread priority for register mapping instead of instruction fetch.
- Disadvantage: Response at later pipeline stage already allows instructions in to waste resources.
- Advantage: Closer to point of interest (registers), allows more isolated, simpler implementation.

Speedup – Adaptive Renaming



ED^2 – Adaptive Renaming



Outline

- Background & Problem Description
- Related Work
- Experimental Setup & Benchmarks
- Adaptive Fetch Algorithm
- Adaptive Register Renaming
- Future Work & Conclusions

Future Work

- Implement in context of multicore (SMT combined CMP) processors.
- Comparison to real-world fetch policies (e.g. ICOUNT).
- Tradeoffs when in conjunction with global DTM strategies.

Conclusions

- Adaptive fetch: 30% performance improvement (44% ED^2 reduction)
- Adaptive rename: 23% performance improvement (35% ED^2 reduction)
- Benefits even in integer-only or FP-only mixes.
- Tradeoff between power-performance benefits and thread fairness.

Acknowledgements

- Special thanks
  - IBM
  - Yingmin Li
  - The anonymous reviewers