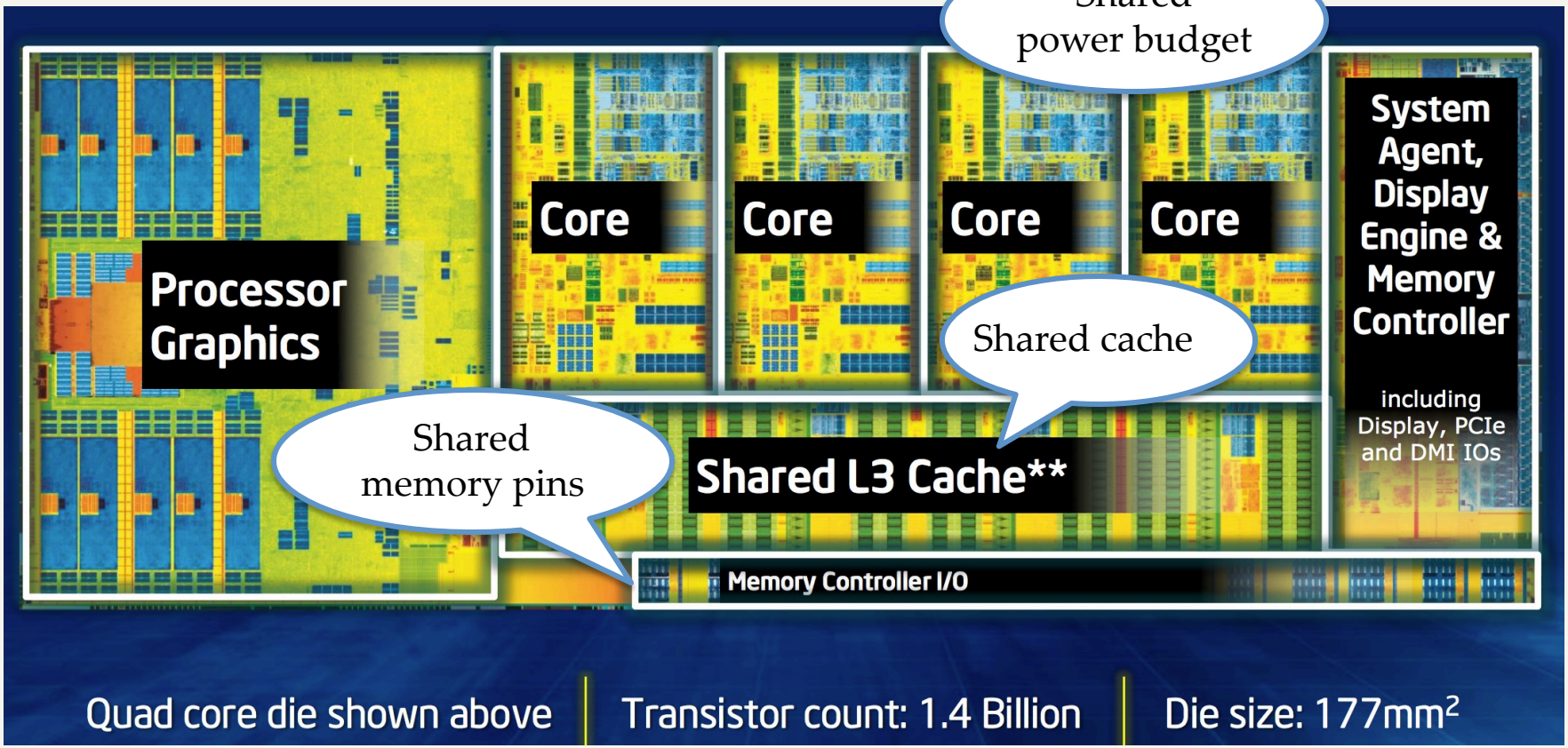
Cornell University
Computer Systems Laboratory

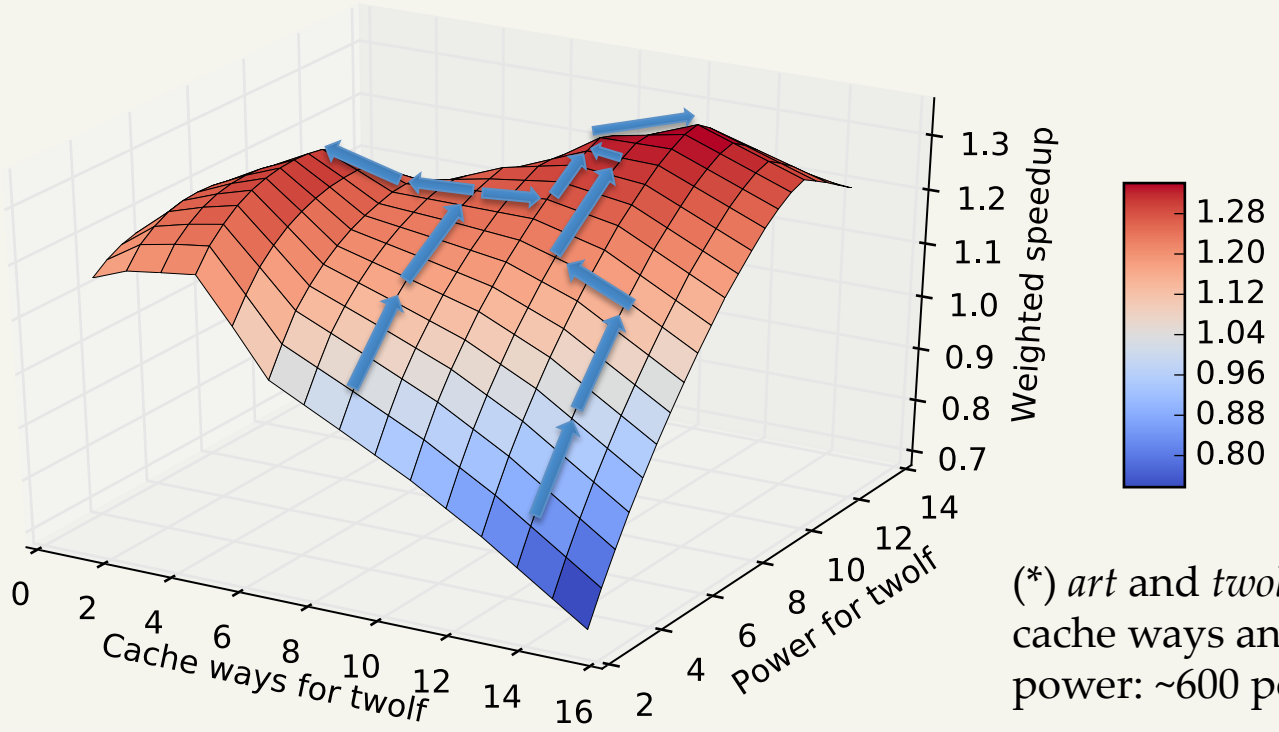
- Haswell's Floorplan



Source: Intel

COORDINATED RESOURCE ALLOCATION

- **Global allocation space very large**
  - Exponentially increasing with the number of cores, the number of resources, and the granularity of resources
  - Hill climbing unlikely to scale gracefully



(*) *art* and *twolf* share 16 cache ways and 20W power: ~600 points

COORDINATED RESOURCE ALLOCATION

- **Global allocation space very large**
  - Exponentially increasing with the number of cores, the number of resources, and the granularity of resources
  - Hill climbing unlikely to scale gracefully
- **Performance-resource relationship not trivial**
  - Not even convex for some resources (e.g., cache)
  - Phase changes
- **Balance between system throughput and fairness**

REAL-LIFE CUE: MARKET-LIKE BEHAVIOR

- **Compromise global optimality for simplicity**
  - Calculate global optimum is very expensive
  - Optimal outcome is not practical
  - **Simple, distributed mechanisms** can be reasonably good

Source: Wikipedia

A MARKET-BASED APPROACH

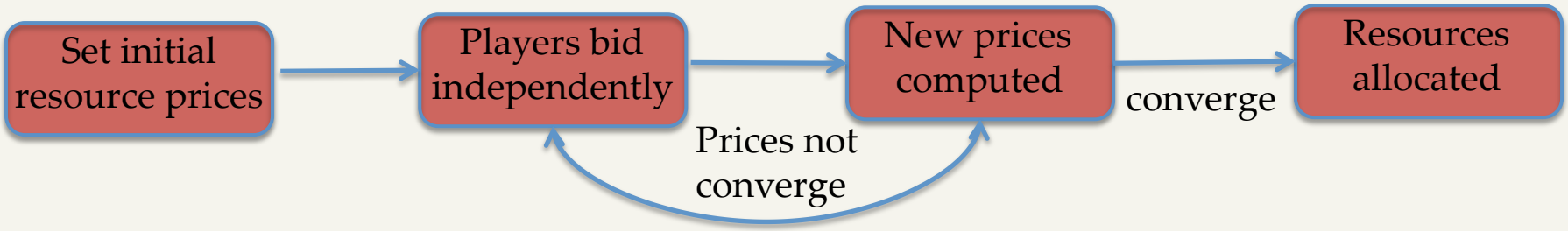
- **Central idea: Pricing CMP resources**
  - Every resource is assigned a price
  - Reflects supply and demand relationship
- **Market players (cores)**
  - Have finite budgets to purchase resources in the system
  - Try to maximize their own utility regardless of others
  - *Price-takers*: no monopolistic behavior
- **Market equilibrium**
  - Prices are such that supply = demand
  - **Our approach: Dynamic price discovery**

Strong ally: first welfare theorem

- **First welfare theorem**
  - Competitive market equilibrium is **Pareto optimal**, if players have monotonic increasing utilities

- **Pareto optimality**
  - An allocation is Pareto optimal if there is no way to reallocate goods so that someone is made better off without making someone else worse off
  - Caveat: Not "perfect world" by itself!

Dynamic price discovery



- **Market side**
  - How to set prices to satisfy demand from cores?
- **Player side (cores)**
  - What are my preferences (utility function)?
  - How do I bid to maximize my utility?

First take: a simple market

- **Market side: proportional pricing [Kelly, ETT 1997]**

$$price_j = \frac{\sum_i bid_{ij}}{total\_resource_j} \qquad resource_{ij} = \frac{bid_{ij}}{price_j}$$

- **Player side: linear utility function**
  - Find preferences: Sparse off- and/or on-line profiling

$$utility_i = \sum_j preference_{ij} \times resource_{ij}$$

  - Bidding strategy: [Wu and Zhang, STOC 2007]

$$bid_{ij} = \frac{preference_{ij} \times resource_{ij}}{utility_i} \cdot budget_i$$

First take: a simple market

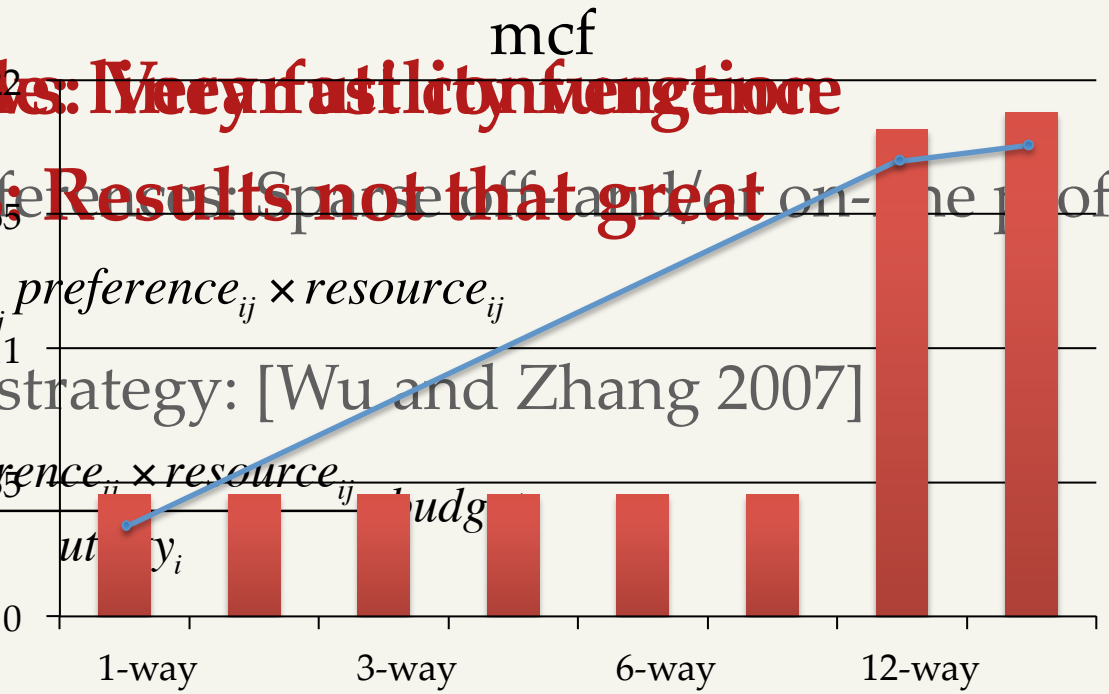
- **Market side: proportional pricing [F. Kelly]**

$$price_j = \frac{\sum_i bid_{ij}}{total\_resource_j} \qquad resource_{ij} = \frac{bid_{ij}}{price_j}$$

- **Good news: Very fast convergence**
- **Bad news: Results not that great**
- End-user side: Specify preferences declaratively on-line profiling
  - Bidding strategy: [Wu and Zhang 2007]

$$utility_i = \sum_j preference_{ij} \times resource_{ij}$$

$$bid_{ij} = \frac{preference_{ij} \times resource_{ij}}{utility_i} \times budget_i$$

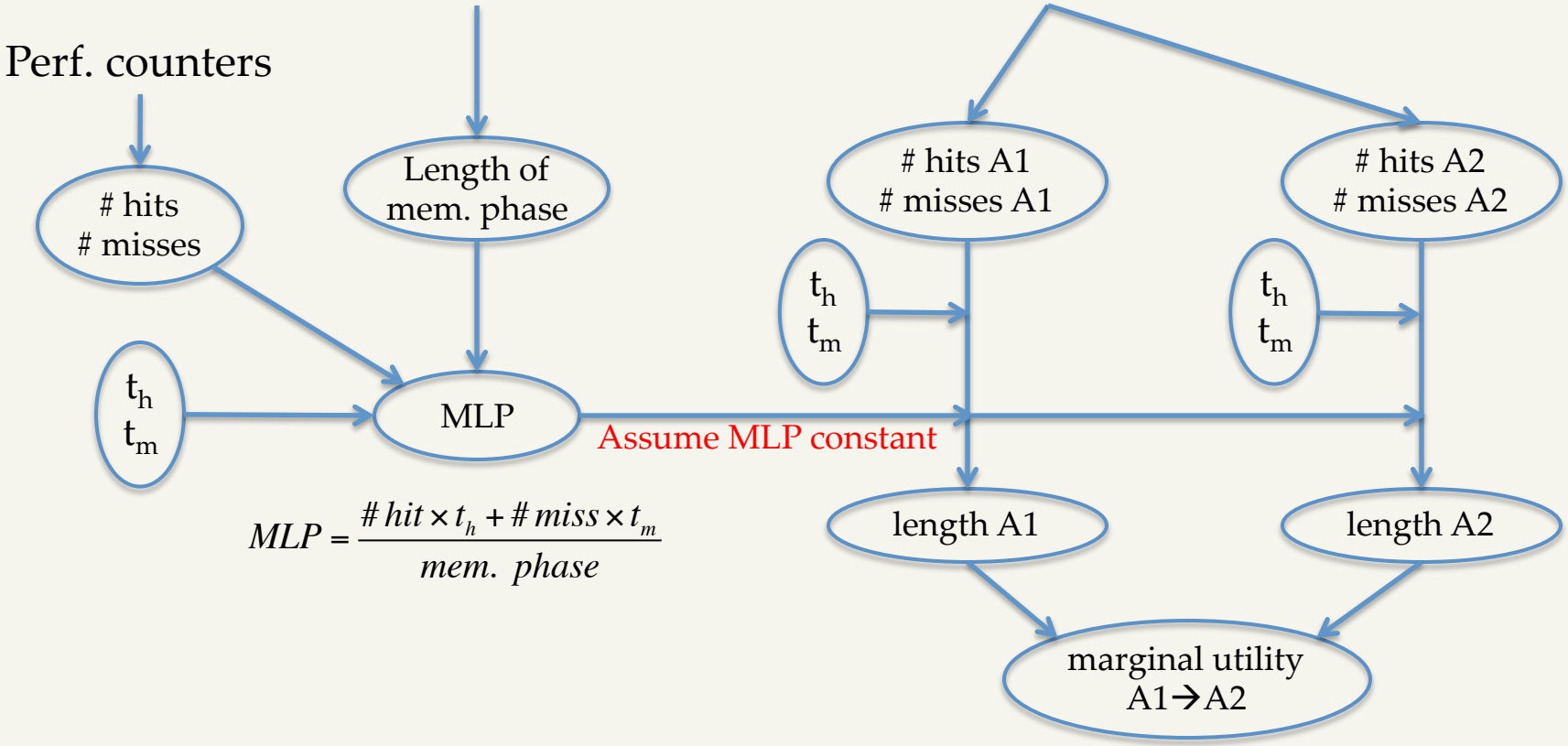


A HEURISTIC PLAYER MODEL

- **Separate memory and compute phases**
  - Memory phase: cache capacity, memory bandwidth, etc.
  - Compute phase: power budget, ROB, FUs, etc

- **We focus on allocating cache capacity and power budget in this paper**
  - Market framework can be applied to any other resources if an accurate utility model is built

- **Memory phase**
  - Combine Miftakhutdinov[1] and UMON[2]



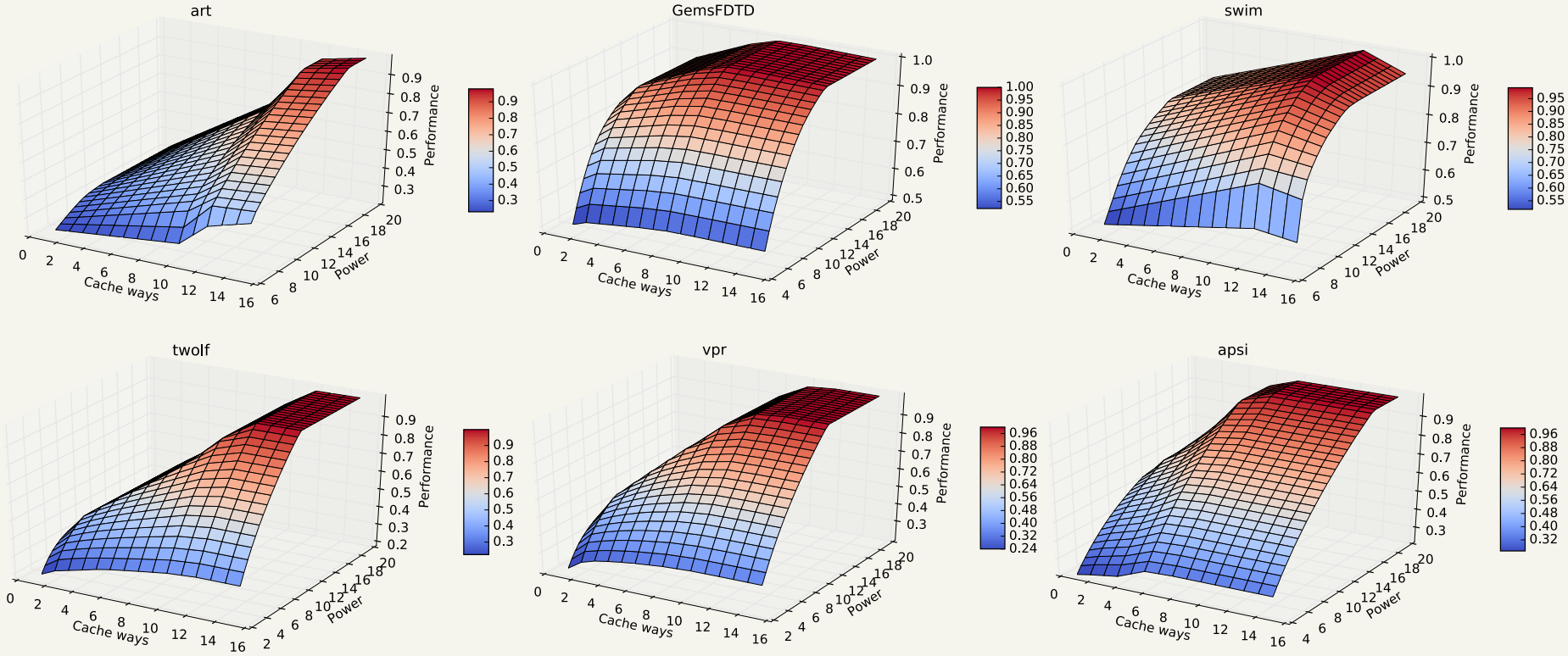
[1] Miftakhutdinov et al, MICRO 2012
[2] Qureshi and Patt, MICRO 2006

A HEURISTIC PLAYER MODEL

- **Memory phase**
  - Combine Miftakhutdinov[1] and UMON[2]
- **Compute phase**
  - Linear relationship between compute phase and frequency
  - Cubic relationship between power and frequency

[1] Miftakhutdinov et al, MICRO 2012
[2] Qureshi and Patt, MICRO 2006

- **Local hill-climbing**
  - All cores search through their utility function concurrently



WEALTH REDISTRIBUTION

- **Budget assignment depends on the definition of optimality**
  - **Fairness-oriented**: Give same budget to everyone
  - **Performance-oriented**: Assigning budgets in proportional to the performance gap between minimum and maximum allocation

Design issues

- **Convergence**
  - Detected through price fluctuation (<1%)
  - Fall-back mechanism after 30 iterations
    - Players quickly decide whether they prefer the current allocation or equal share
- **Bankruptcy**

WAIT—WHAT ABOUT THEORETICAL GUARANTEES?

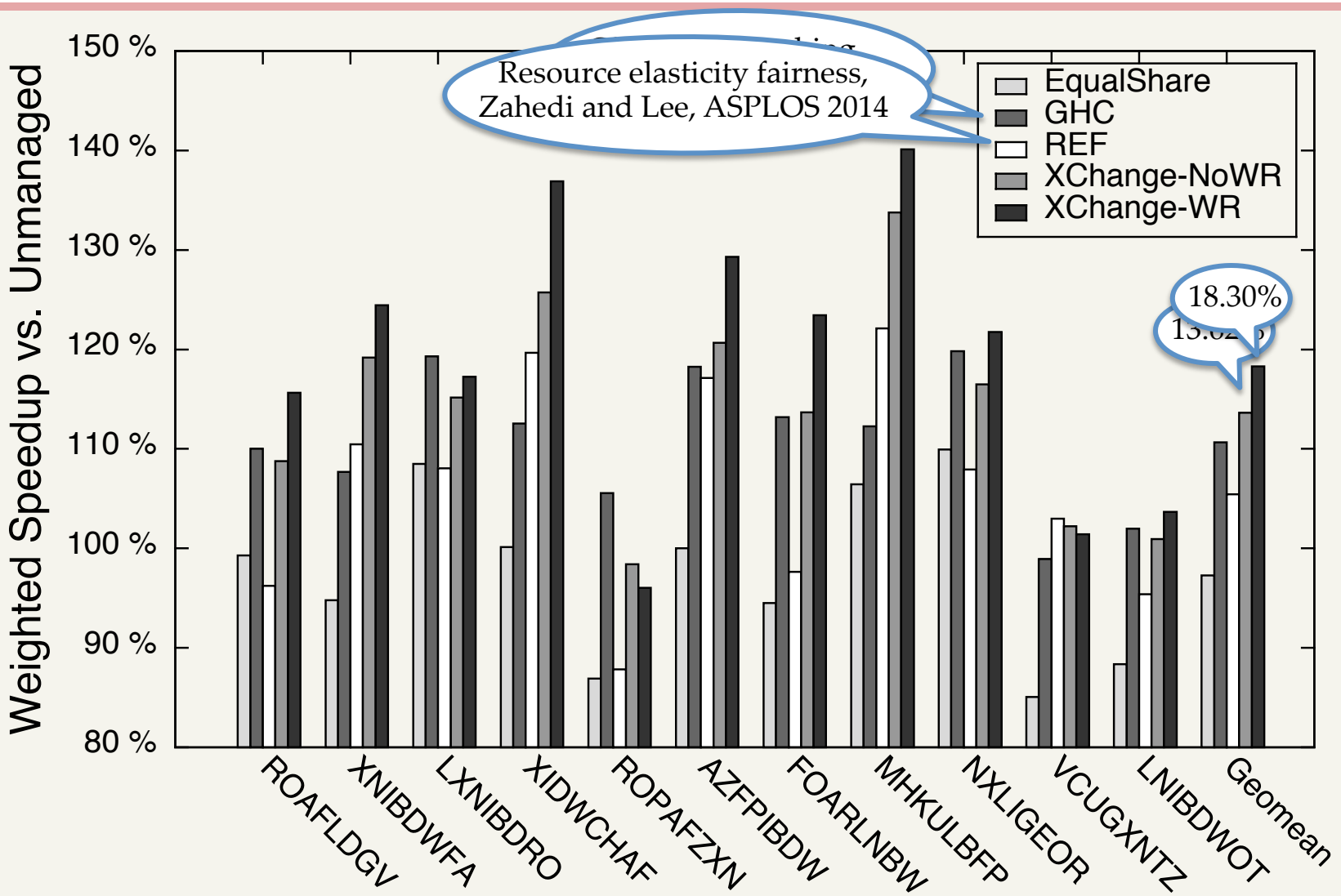
- **Pretty much out of the window**
  - Utility function approximation at best
    - Based on architecture heuristics
  - Bidding search not exhaustive
  - Predictive: past history = future performance

- **Nevertheless, reason for optimism**
  - Plenty of real-life examples that just work
  - Utility models captures application behavior well
  - First welfare theorem has weak conditions
  - We have reasonable fallback mechanism

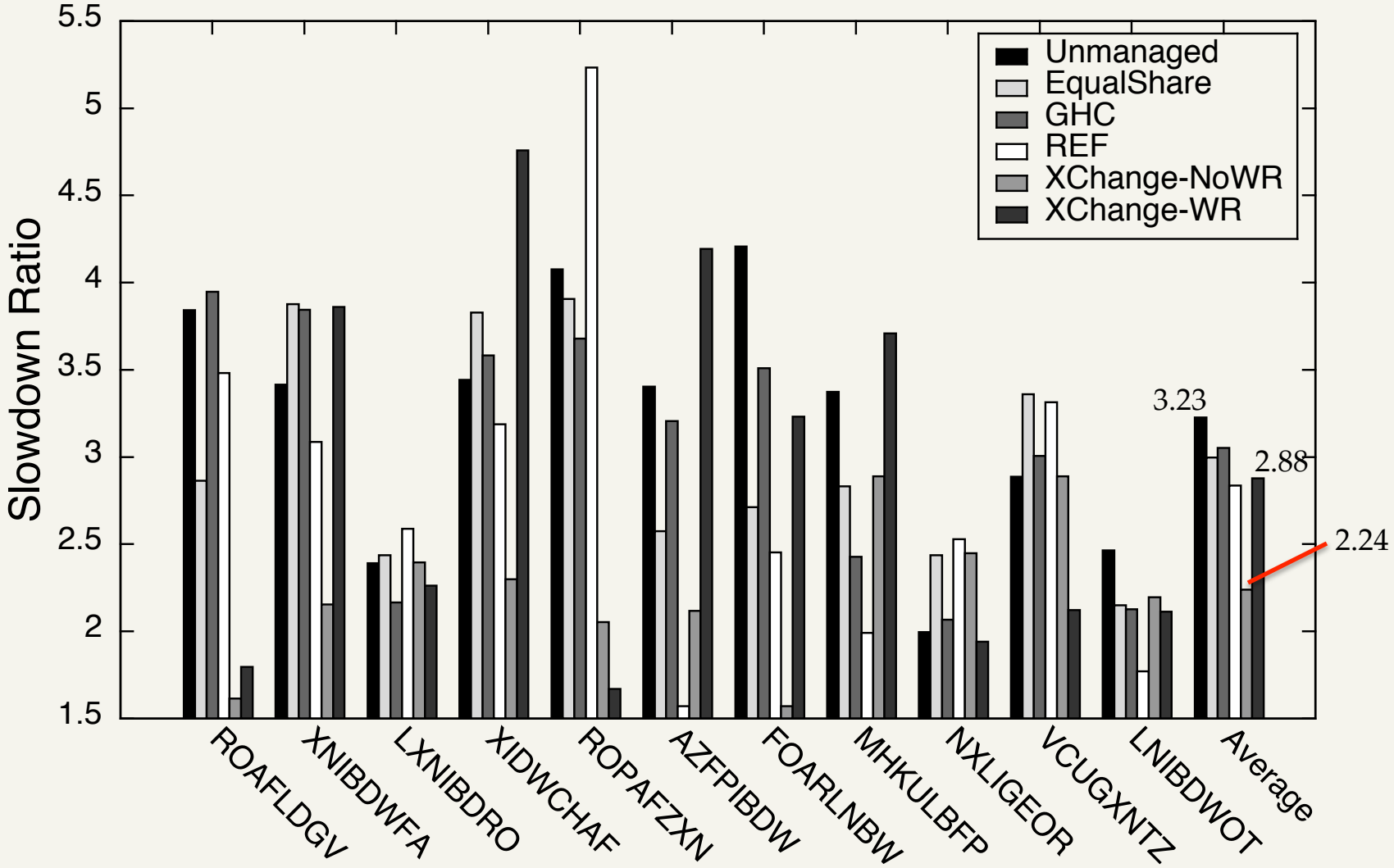
Experimental setup

- **Simulation setup**
  - 4 GHz 4-way OoO core, 32 kB i/d L1
  - 8- and 64-core CMP
  - 512kB L2, 10W per core as equal-share
  - DDR3-1600 channels, 4 ranks ea., 8 banks/rank
- **Performance analysis**
  - Mix of SPEC2000 and SPEC2006 multi-programed workloads
- **Comparison to state-of-the-art resource allocation**

System throughput (weighted speedup)



System fairness (maxmin slowdown)



- **Execution time for GHC to converge**

# cores	4	32	64	128	256
Cycles	43	484	1697	6418	24903
% interval	0.87%	9.69%	33.95%	128%	498%

- **Execution time for XChange-WR market-based technique to converge**

# cores	4	32	64	128	256
Cycles	9.47	12.49	15.89	22.64	52.70
% interval	0.19%	0.25%	0.32%	0.45%	1.05%

(*) Assume 5M cycle reallocation interval

Conclusions

- **Market-based approach very promising**
  - Fast and scalable
  - Solid results
  - Adjustable system throughput and fairness
- **Heuristic approach valid**
  - Plenty of real-life examples that just work
  - Utility models captures application behavior well
  - First welfare theorem has weak conditions
  - We have reasonable fallback mechanism

Cornell University
Computer Systems Laboratory

XChange:
A Market-based Approach to Scalable Dynamic Multi-resource Allocation in Multicore Architectures

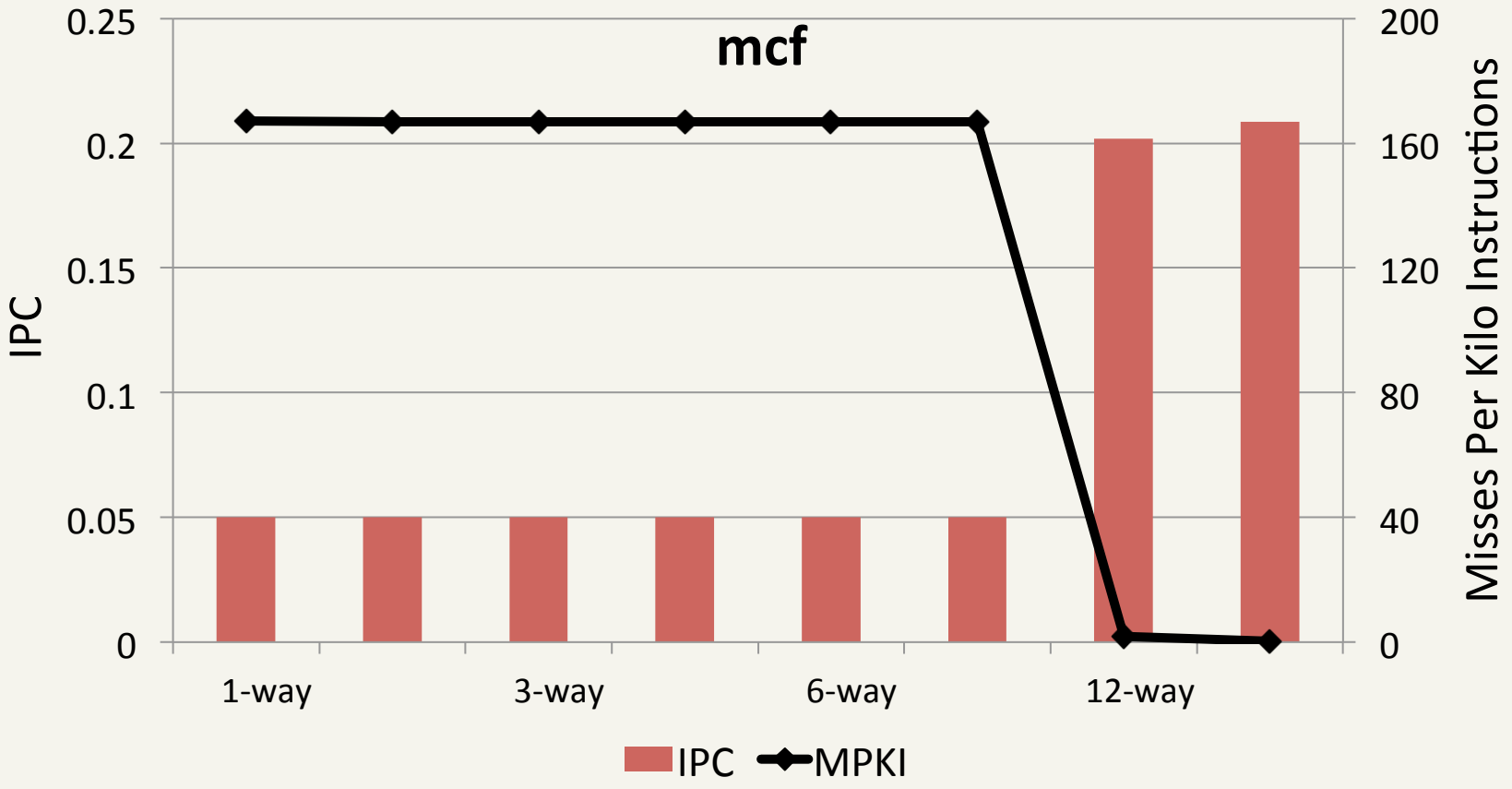
Xiaodong Wang
José F. Martínez

Computer Systems Lab
Cornell University

BACKUP SLIDES

PRIOR ART: EXAMPLE OF NON-CONVEXITY

- *mcf*'s IPC vs. cache allocation



Prior art

- **Key: Performance modeling + resource allocation**
- **Sampling + hill climbing [Choi and Yeung 2006]**
- **Predictive model + hill climbing**
  - Artificial neural network [Bitirgen et al. 2008]
  - Analytical model [Chen and John 2011]
- **Curve-fitting + elasticity-proportional (Zahedi and Lee 2014)**
  - Guarantee game-theoretic fairness at the cost of efficiency

- **Compute phase**
  - Power: assume compute phase is linear to frequency

$$t_{exe}(f) = t_{exe} \times \frac{f_{avg}}{f}$$

Measured (perf. counters)

$$power(f) = \frac{\frac{E_0}{V_0^2} \cdot V_f^2}{t_{exe}(f) + t_{mem}}$$

Miftakhutdinov

$$MU_{power}(f) = \frac{t_{exe}(f - \Delta f) - t_{exe}(f)}{power(f) - power(f - \Delta f)}$$

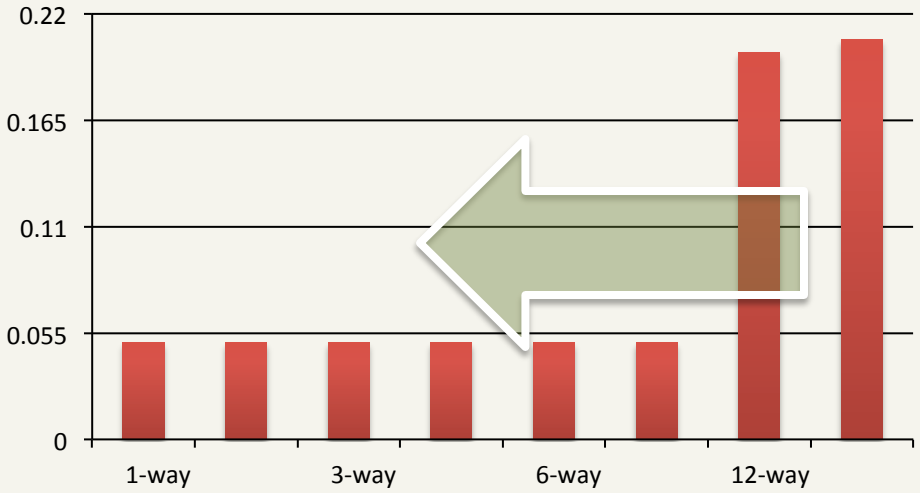
IMPLEMENTATION

- **Leverage Linux's APIC timer interrupt**
  - Every 1 ms, for kernel statistics update
  - Designate "master core" to post prices, collect bids

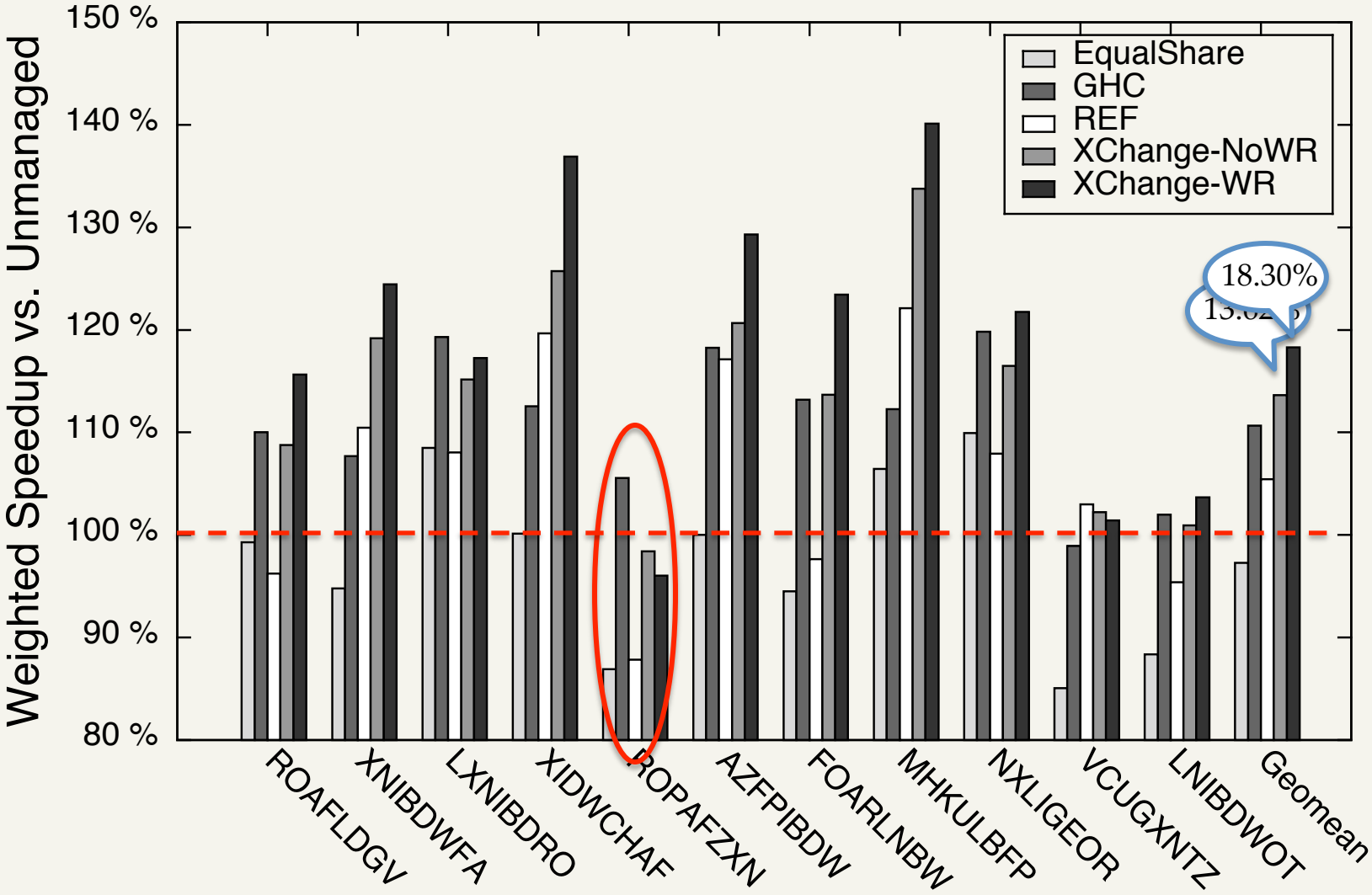
- **Modest hardware overhead**
  - ~ 4 kB/core (mostly UMON)

BIDDING STRATEGY

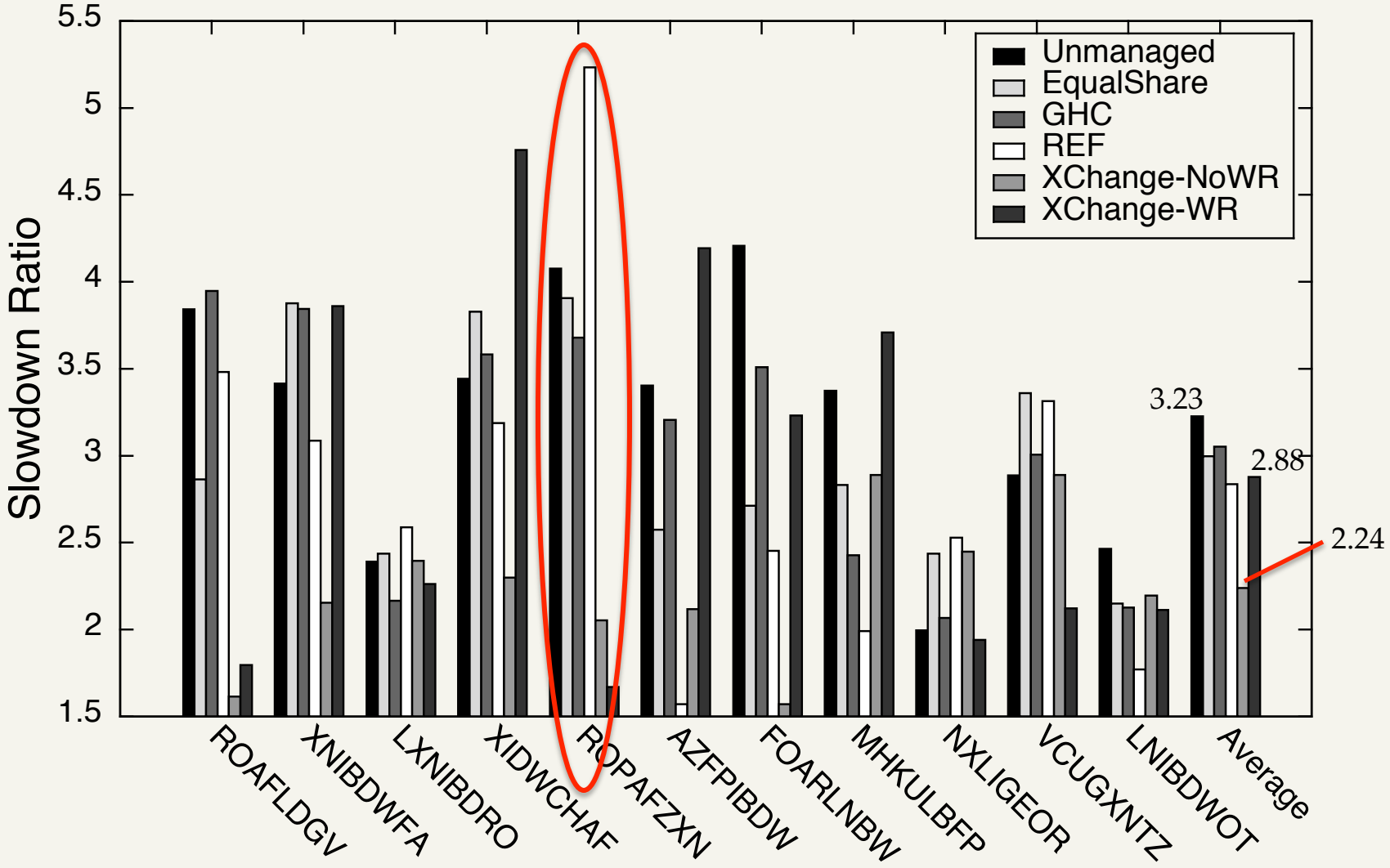
- **"Guided" hill-climbing**
  - Tries to go around cache non-convexity issue
  - Purchase minimum frequency allocation (800 MHz)
  - Bid all remaining money to cache
  - Progressively trade off cache ways for power
  - Caveat: works only when one resource is non-concave



System throughput (weighted speedup)



System fairness (maxmin slowdown)



System performance (harmonic speedup)