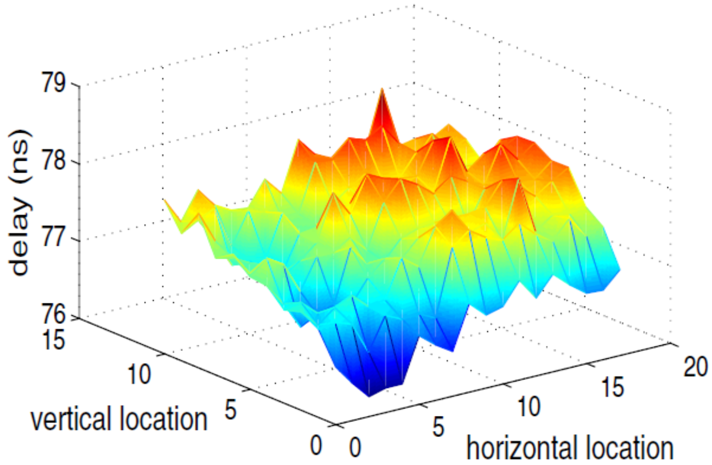
Fairness-oriented Scheduling Support for Multicore Systems

Changdae Kim and Jaehyuk Huh

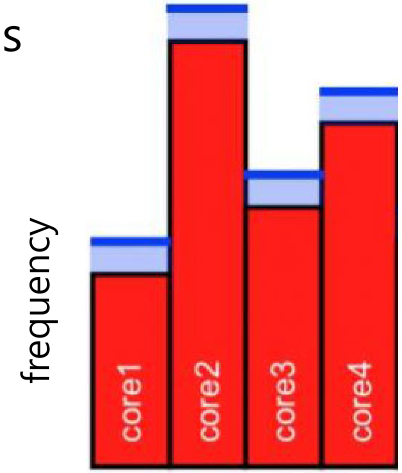
KAIST

Multicores have Uneven Capabilities

Asymmetric Multicores



Within-die Process Variation

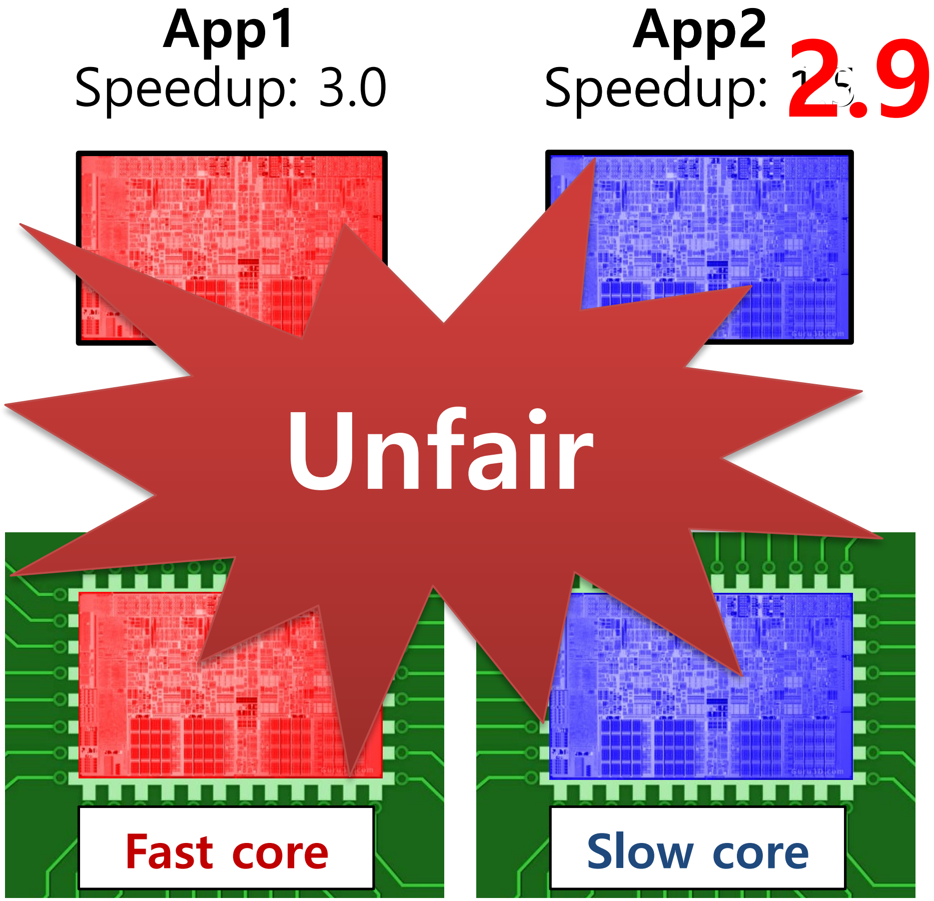


DVFS

Throughput-maximizing Scheduling

- **Max-perf: maximize the overall throughput**

  [Kumar'03], [Koufaty'10], [Kwon'11], [Saez'10], [Shelepov'09], [Craeynest'12]



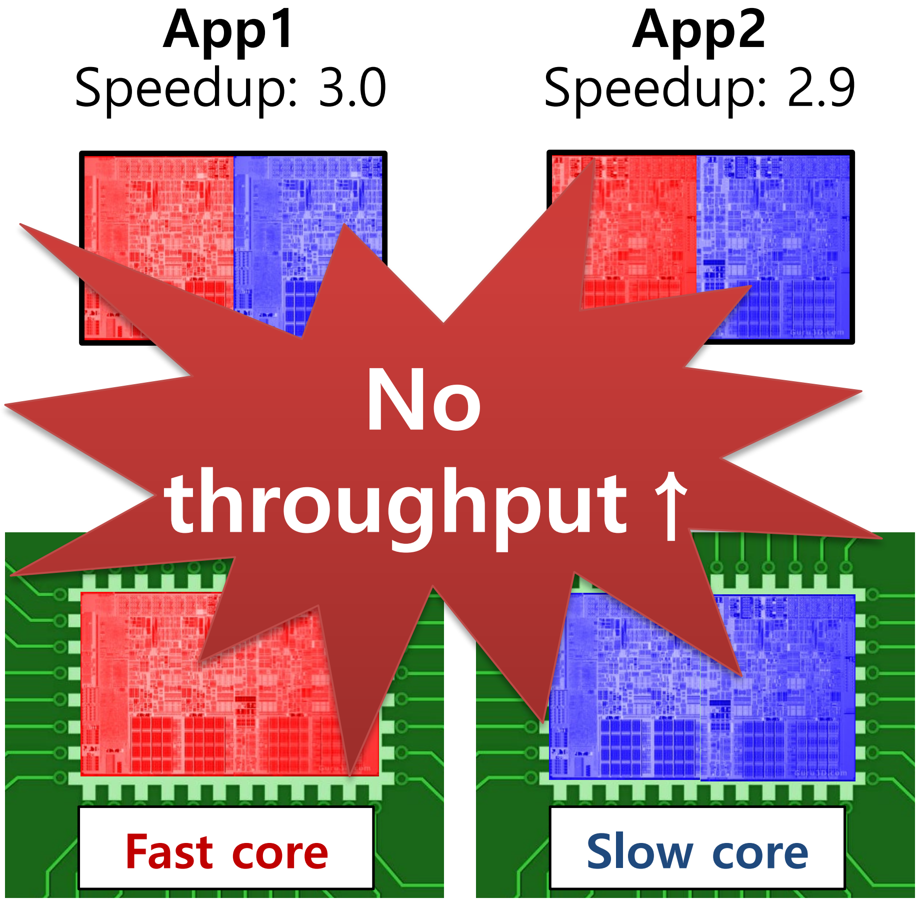
1) Estimate fast core speedup

$$\left(= \frac{\text{perf on fast core}}{\text{perf on slow core}} \right)$$

2) Assign fast core
   to highest speedup app

Fairness-Maximizing Scheduling

- **Max-fair: maximize the fairness** [Kwon'11]



App1
Speedup: 3.0

App2
Speedup: 2.9

No throughput ↑

Fast core

Slow core

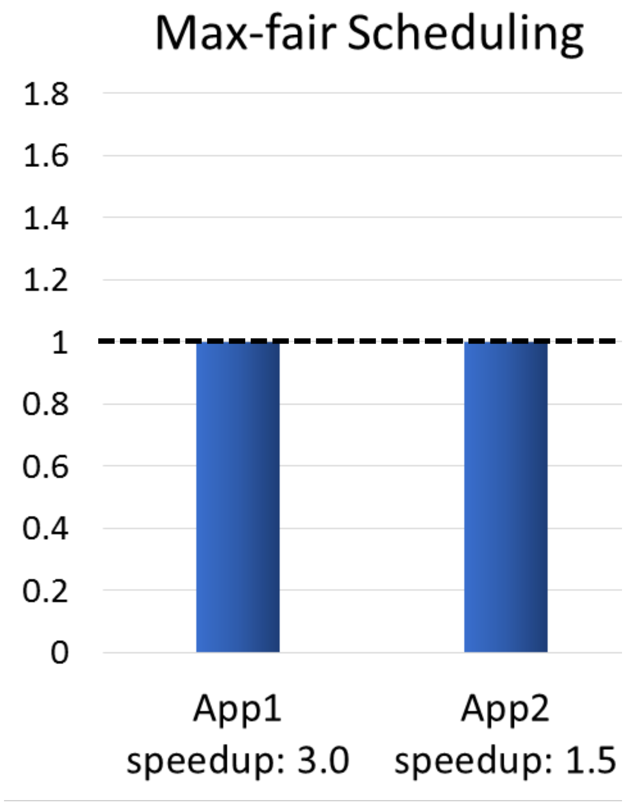
1) Equal share of fast core
   to all apps

2) Equal share of slow core
   to all apps

Max-Fair vs. Max-Perf

Fairness

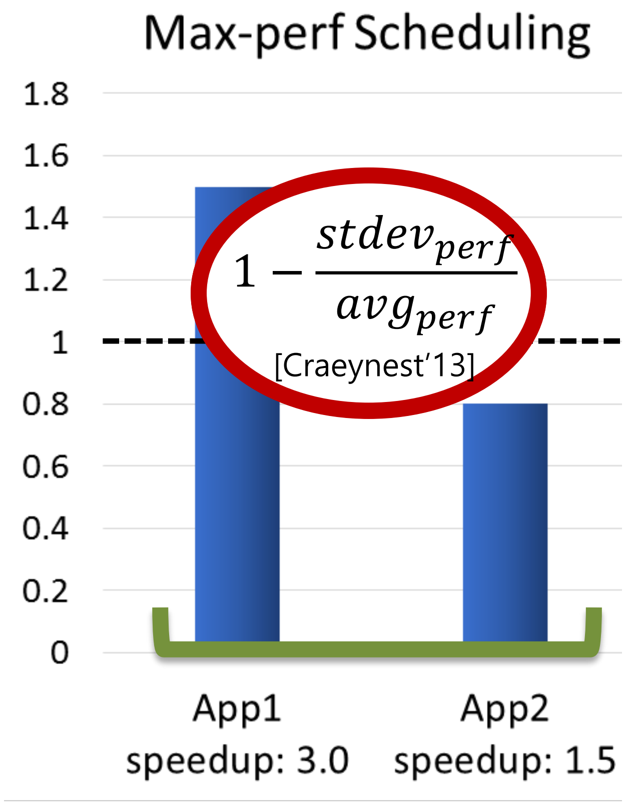
Throughput



Throughput
(average)
+15%

MinFairness
(minimum)
-20%

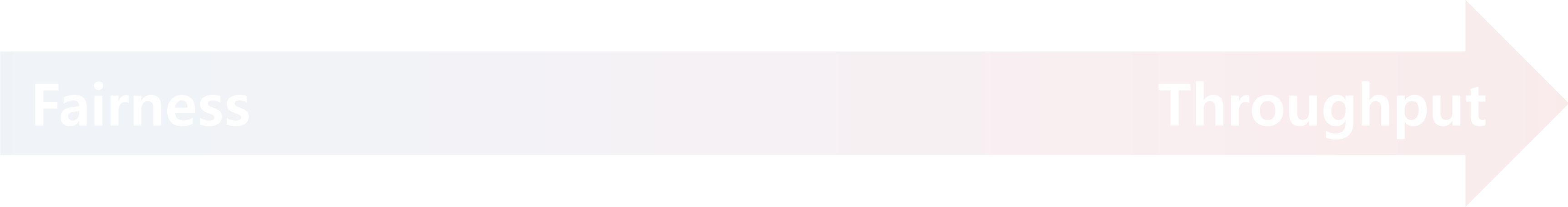
Uniformity
(variance)
-43%

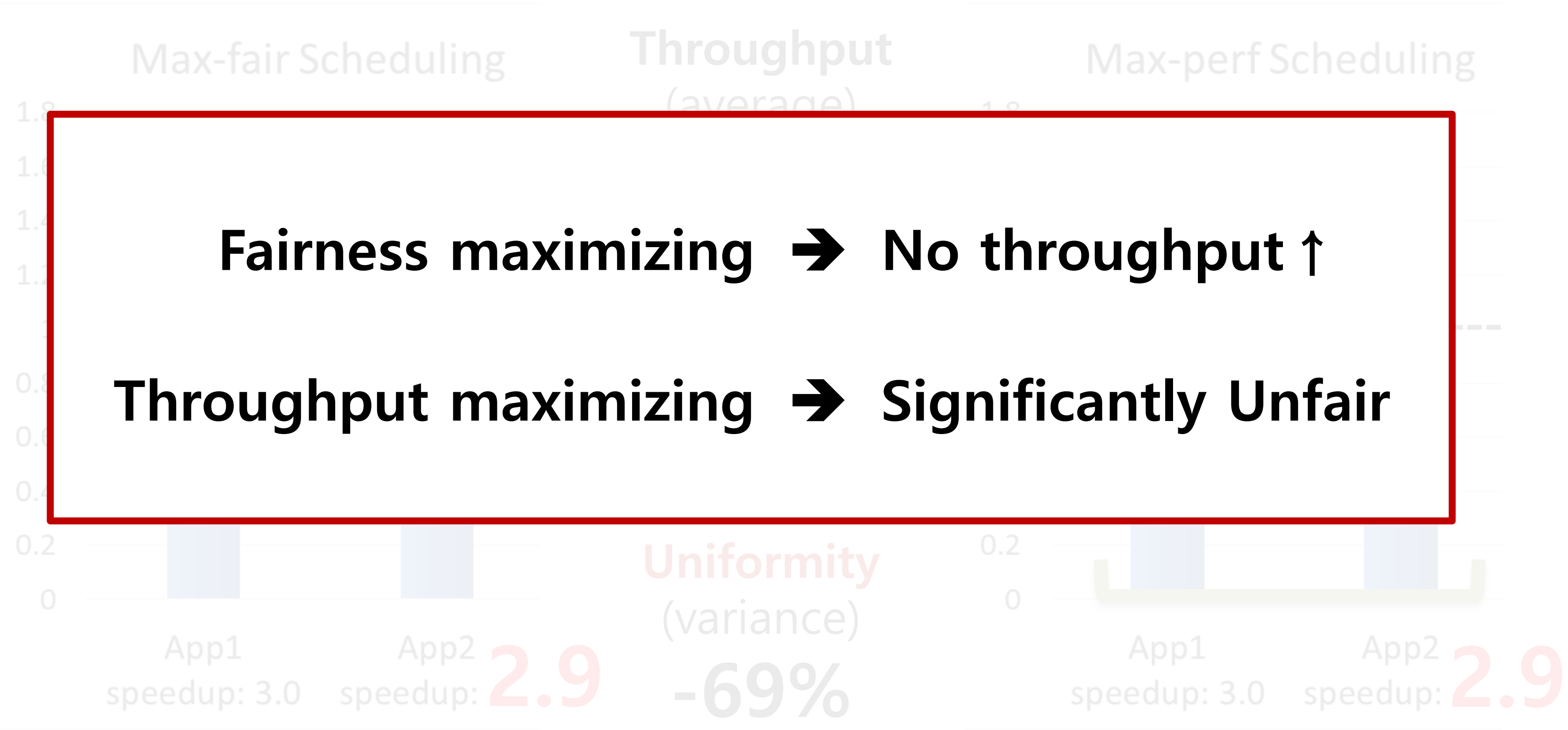


$$1 - \frac{stdev_{perf}}{avg_{perf}}$$

[Craeynest'13]

Max-Fair vs. Max-Perf

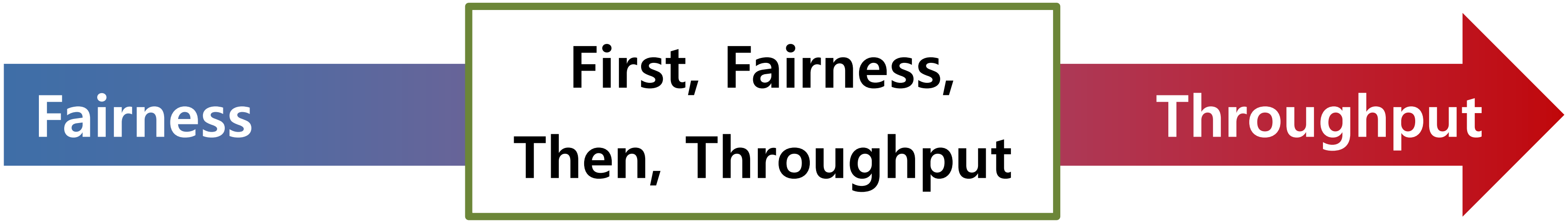




Fairness maximizing ➔ No throughput↑

Throughput maximizing ➔ Significantly Unfair

Fairness-oriented Scheduling



Policies

Min-fair: Guarantee MinFairness, then Throughput↑

Similar-min-fair: Min-fair + Improve Uniformity

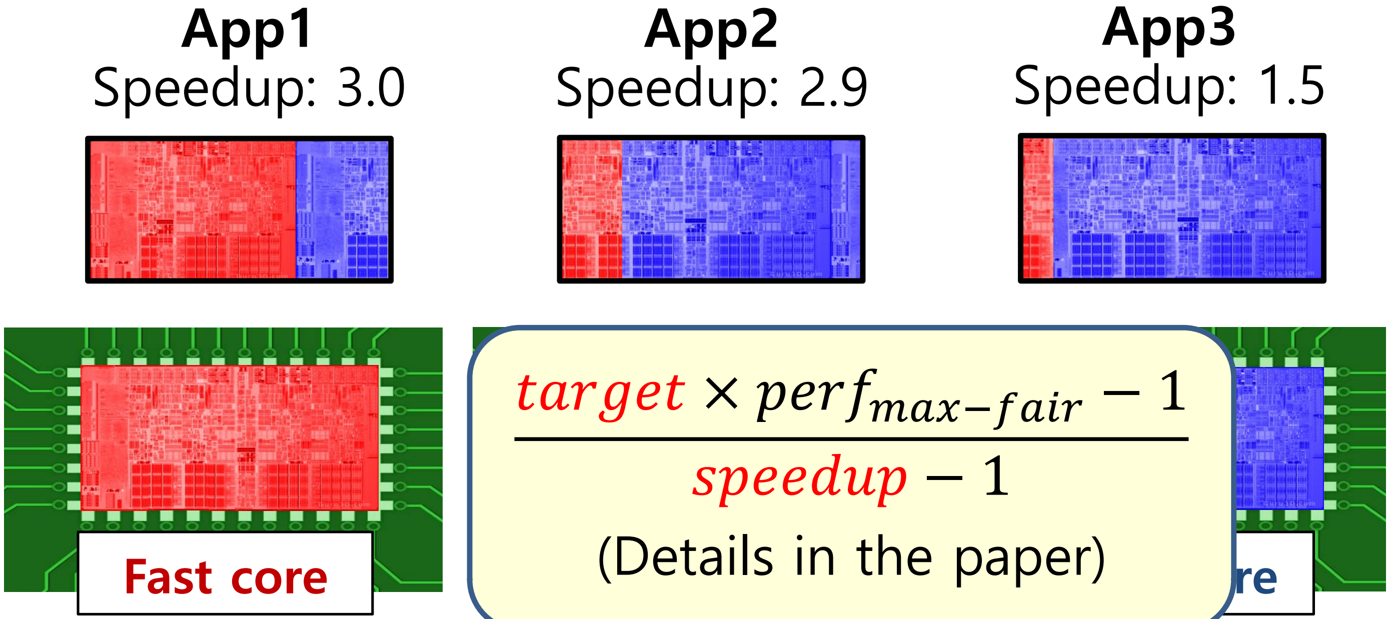
Implementation issues

Estimate accurate value of fast core speedup

Schedule as core share

Min-Fair Scheduling

- **Guarantee minimum performance as users want**



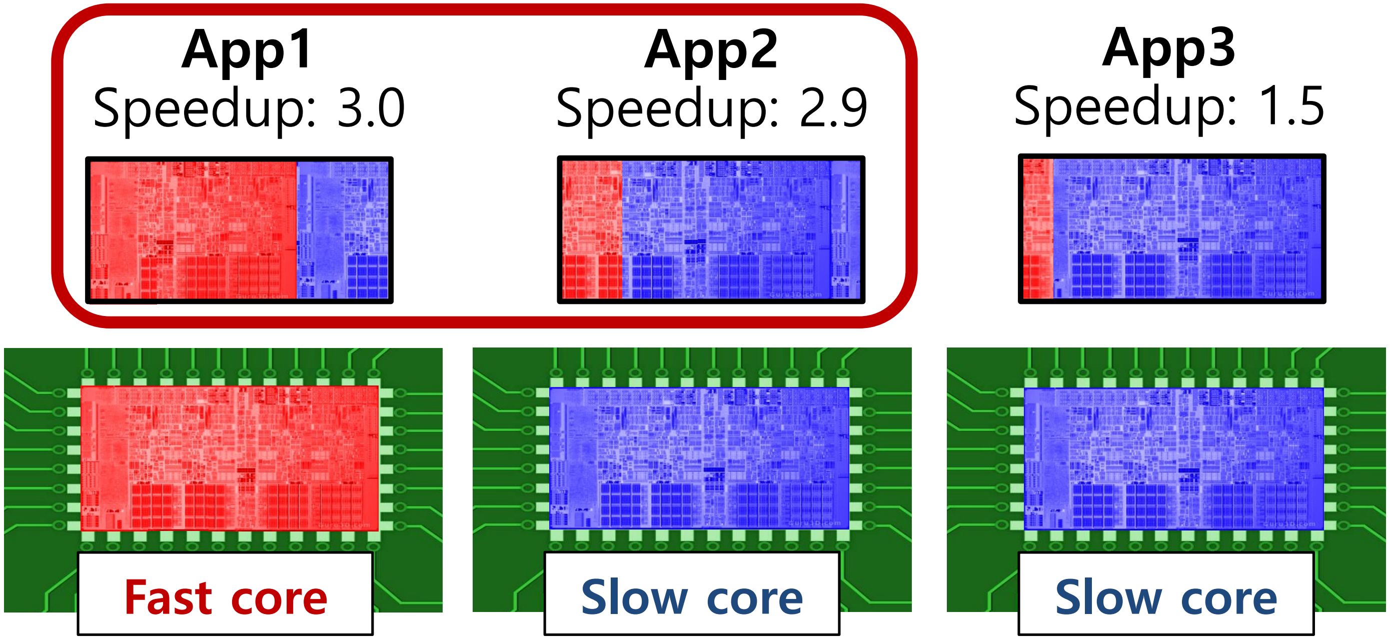
$$\frac{target \times perf_{max-fair} - 1}{speedup - 1}$$

(Details in the paper)

1) Administrator set ***target*** of MinFairness
2) Required fast core to all apps ➔ MinFairness > ***target***
3) Remaining fast core to high speedup app ➔ throughput↑
4) Distribute slow core to all apps

Similar-Min-Fair Scheduling

- **Similar speedup apps ➔ same throughput ↑**



App1
Speedup: 3.0

App2
Speedup: 2.9

App3
Speedup: 1.5

Fast core

Slow core

Slow core

1) Administrator sets *similarity* and *target* of MinFairness
2) Min-fair scheduling with *target*
3) Group similar apps (speedup difference < *similarity*)
4) Even fast core share out among apps in a group

Sampling-based Speedup Estimation

- Force to run on both types of cores for each interval
  - Similar to [Kumar '03]
- Measure performance using PMU (Performance Monitoring Units)
  - Performance metric: Instruction Per Seconds (IPS)

$$\text{speedup} = \frac{\text{moving average of IPS on fast core}}{\text{moving average of IPS on slow core}}$$

😄 Architecture independent mechanism

😄 Low error rate: avg. 2.70%, at worst 9.92%

😄 Must run on both types ➔ min-fair already does

😄 Incur frequent thread migration

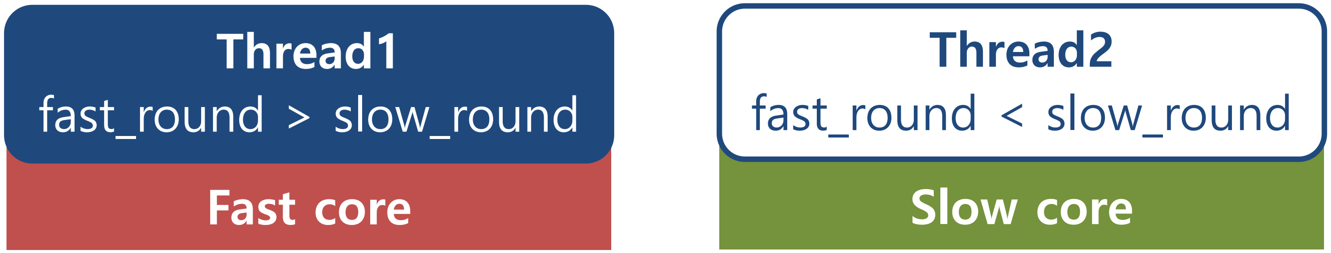
➔ Little impact on cores sharing a LLC [Craeynest '12]

➔ Less than 1.8% at worst

Implementation on Linux Kernel

- Add *fast_round* and *slow_round* on thread context
  - +1 when run on fast/slow core for fast/slow_core_share*30ms
  - Progress together ➔ schedule as core share ratio

- Swap threads to balance fast_round and slow_round



Thread1

fast_round > slow_round

Fast core

Thread2

fast_round < slow_round

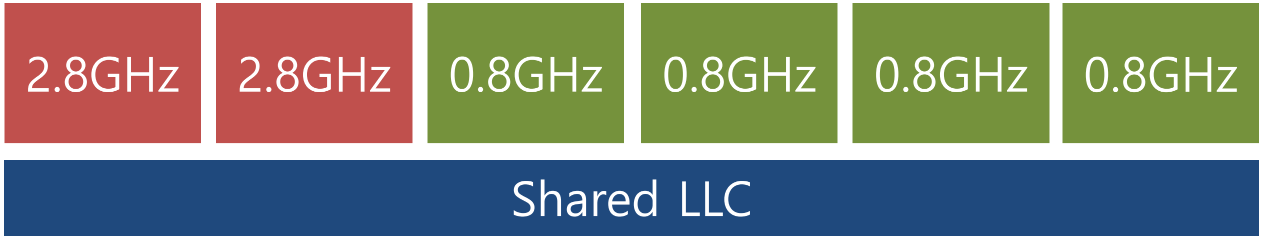
Slow core

- Measure IPS on each type of cores

- Core share update interval: 2 seconds

Evaluation Methodology

- Emulate uneven multicores using DVFS
  - AMD Phenom II: 2 fast cores and 4 slow cores



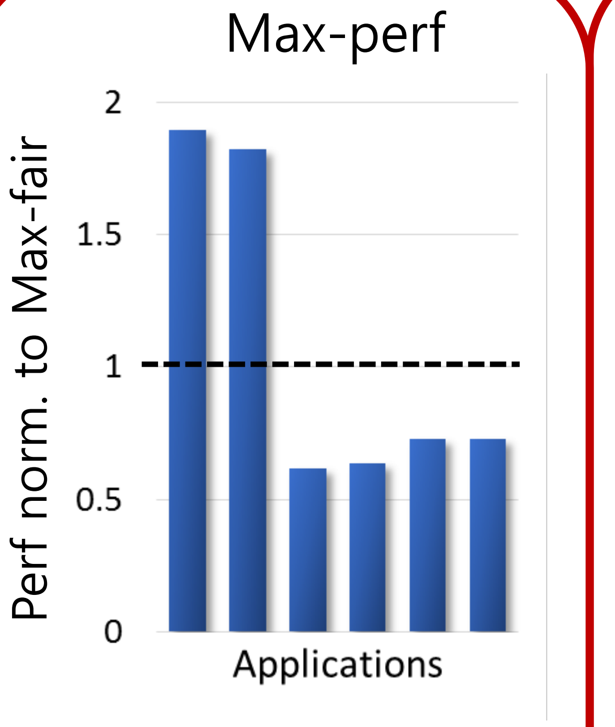
2.8GHz 2.8GHz 0.8GHz 0.8GHz 0.8GHz 0.8GHz

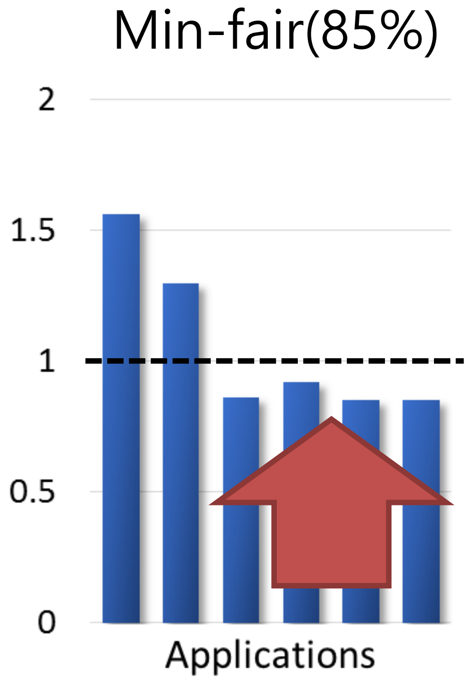
Shared LLC

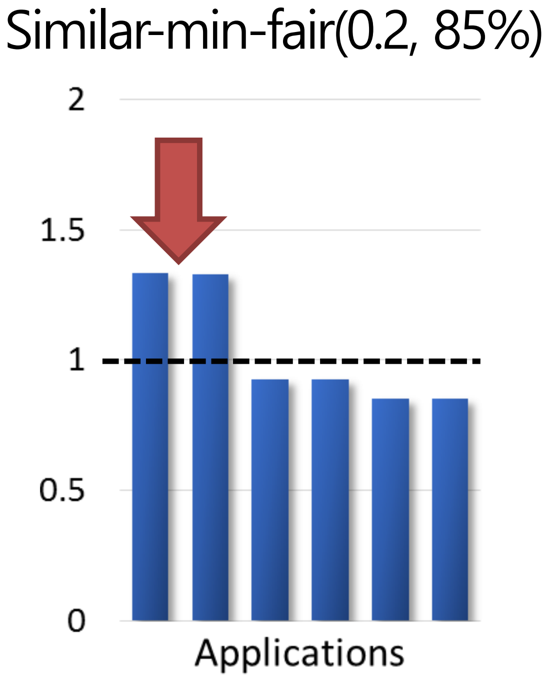
- Big.LITTLE architecture
  - Odroid-XU3 Lite: 4 big cores and 4 little cores
  - PMU not available ➔ use offline speedup value
- Workloads: mix of SPECCPU2006
  - Run repeatedly until all apps finish at least once
  - Performance: execution time of the first run

Result: Diverse Applications

- **Guarantee MinFairness and then improve throughput**



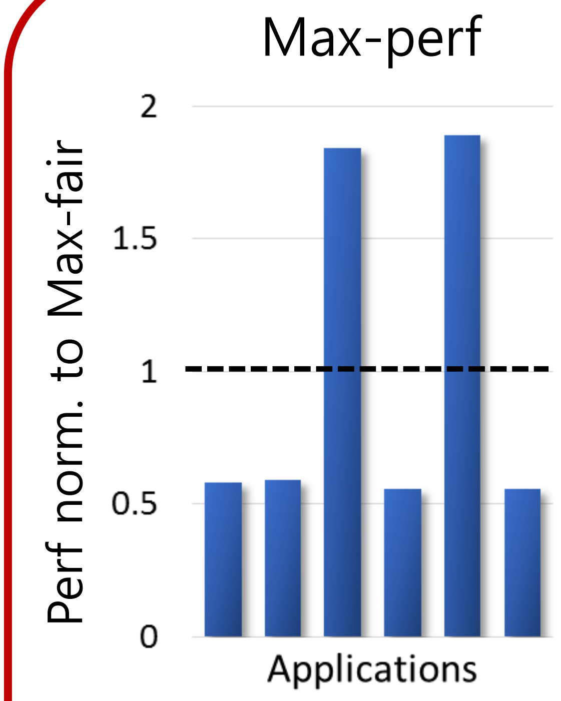




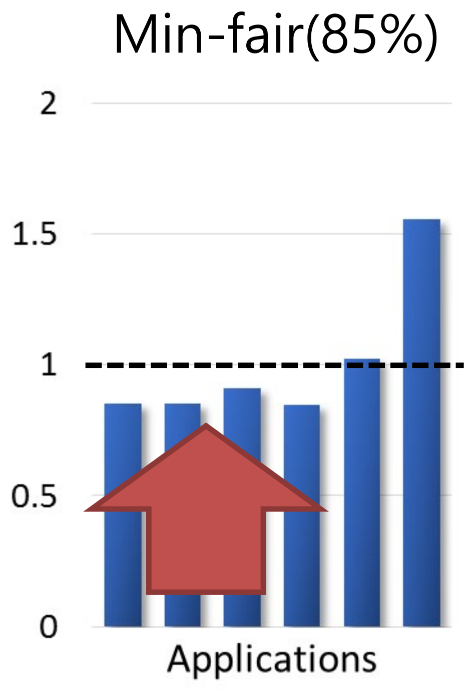
Throughput:	**107%**	**106%**	**104%**
MinFairness:	**62%**	✓ **85%**	**85%**
Uniformity:	**43%**	**72%**	✓ **78%**

Result: Similar Applications

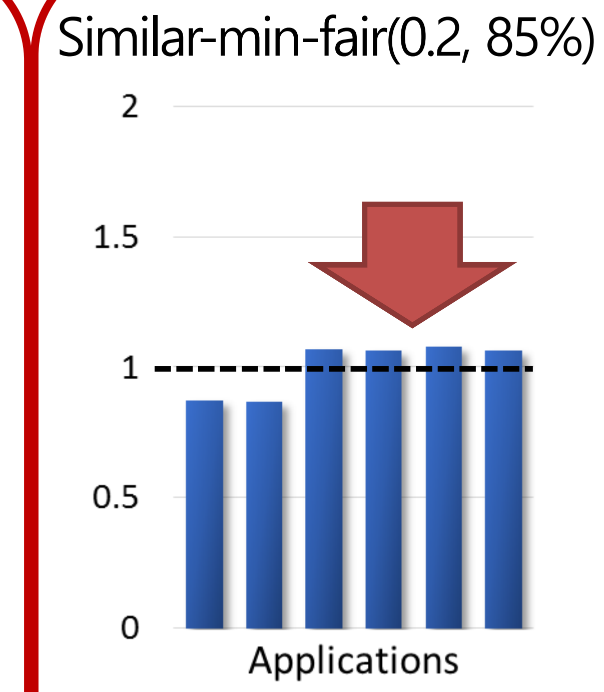
- **Improve fairness without effect on throughput**



Max-perf



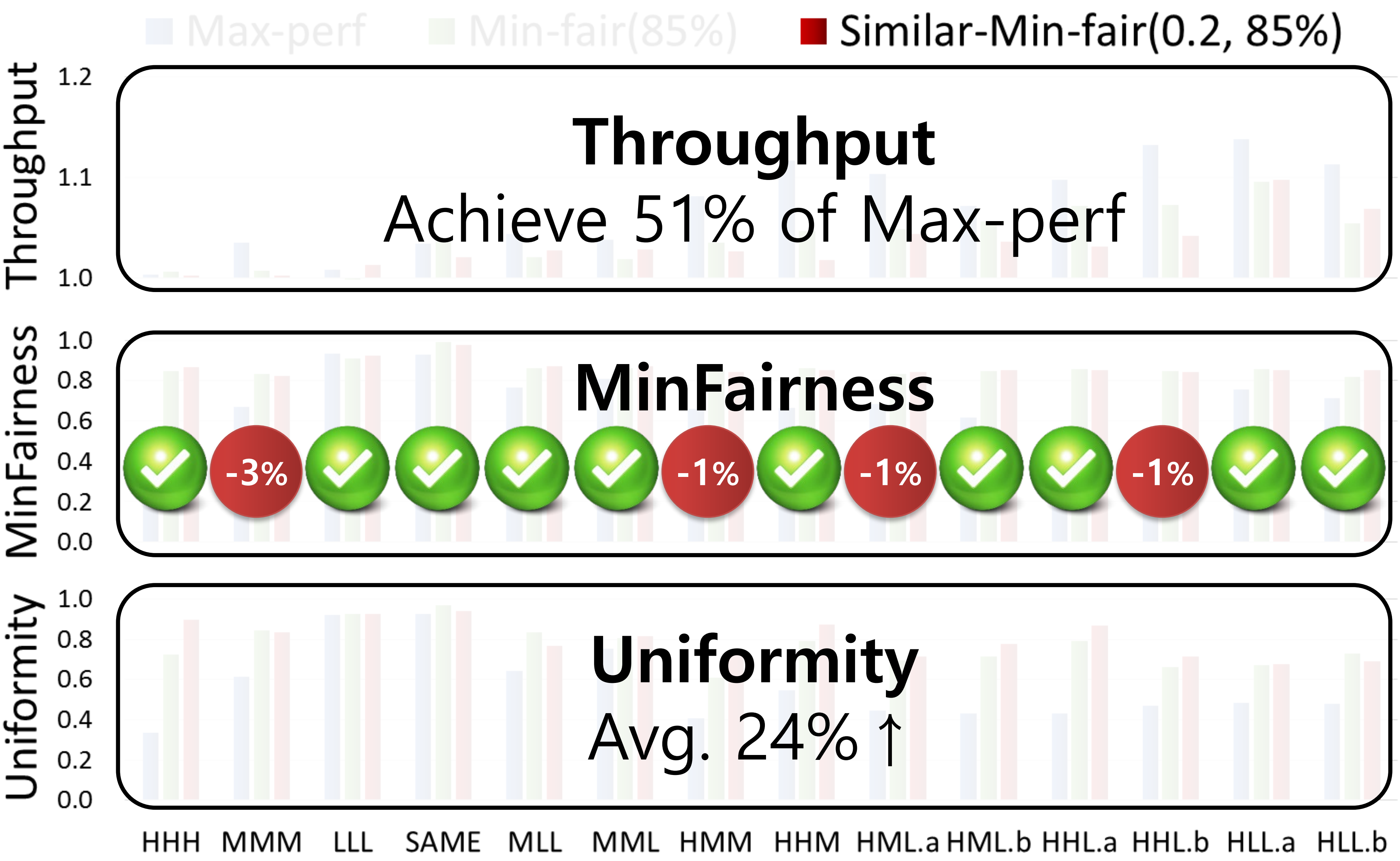
Min-fair(85%)



Similar-min-fair(0.2, 85%)

	Max-perf	Min-fair(85%)	Similar-min-fair(0.2, 85%)
Throughput:	**100%**	**100%**	**100%**
MinFairness:	**56%**	✓ **85%**	**87%**
Uniformity:	**33%**	**72%**	✓ **90%**

Result: All Mixes



Throughput

Achieve 51% of Max-perf

MinFairness

Uniformity

Avg. 24%↑

Conclusion

- Fairness-oriented scheduling for uneven multicores
  - First, Fairness, Then, Throughput
- Architecture independent speedup estimation
  - High accuracy and little overhead
- Implemented on linux kernel 3.7.3
- Real machine results
  - MinFairness: mostly guaranteed (missed less than 3%)
  - Uniformity: avg. 24% ↑
  - Throughput: achieve 51% of Max-perf