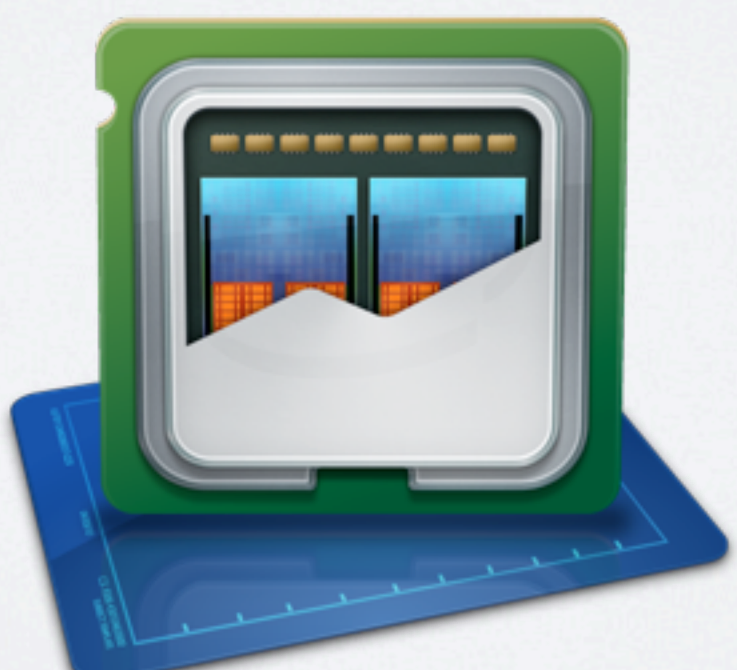
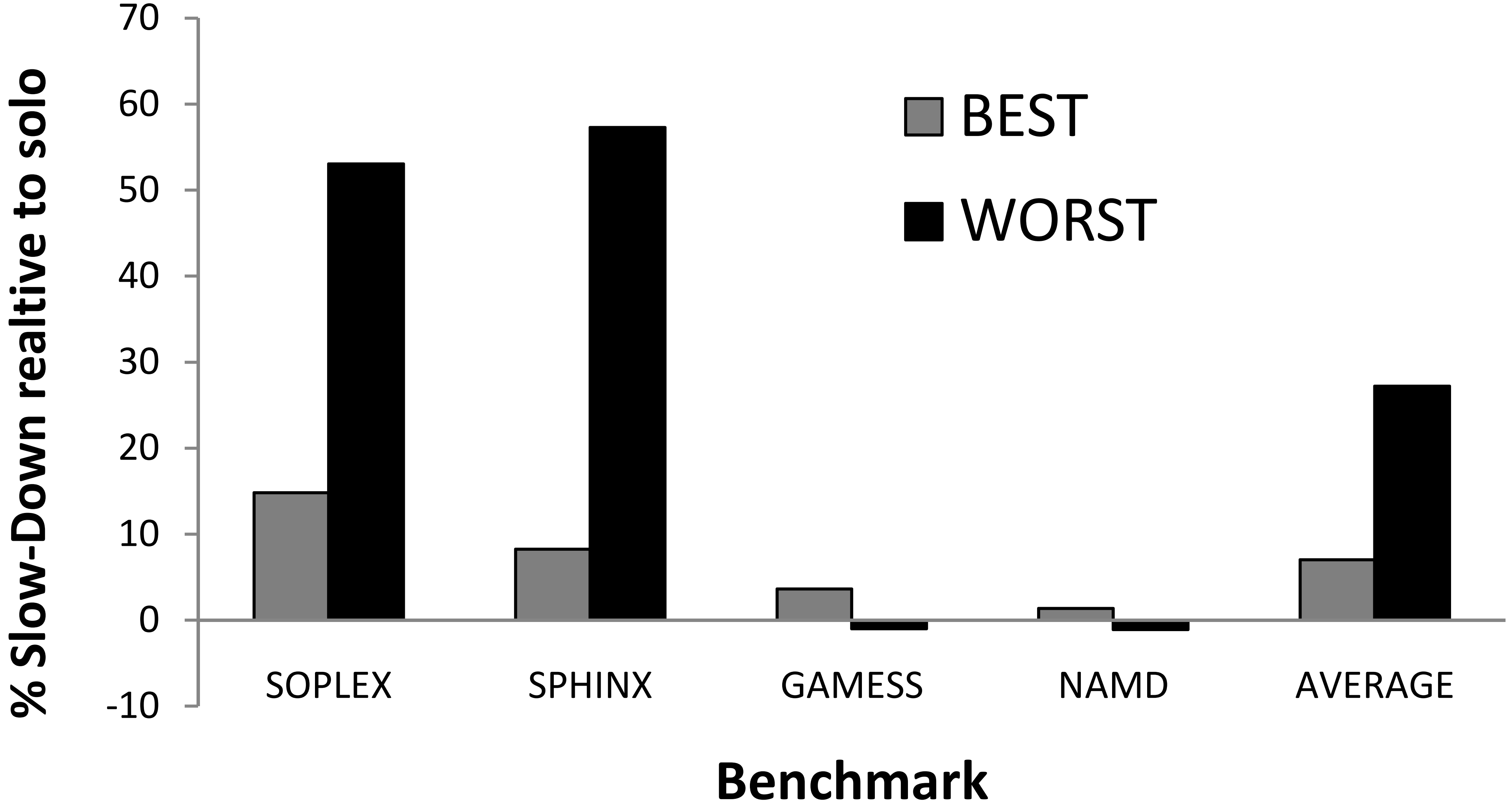
ADDRESSING SHARED RESOURCE CONTENTION IN MULTICORE PROCESSORS VIA SCHEDULING

Sergey & Sergey
Alexandra Fedorova



13% SPEED IMPROVEMENT
FOR FREE.
REALLY.


% Slow-Down realtive to solo
70
60
50
40
30
20
10
0
-10
BEST
WORST
SOPLEX
SPHINX
GAMESS
NAMD
AVERAGE
Benchmark

PERFORMANCE ON MULTICORES

- co-scheduled applications may **contend** for cache space
- … and memory controllers, busses, prefetch units
- previous solutions focus on **hardware** or **page coloring**
- mitigate contention by using **only scheduling**
- performance **improvement** and **isolation**

classification
scheme for
threads

scheduling
policy
assigning cores

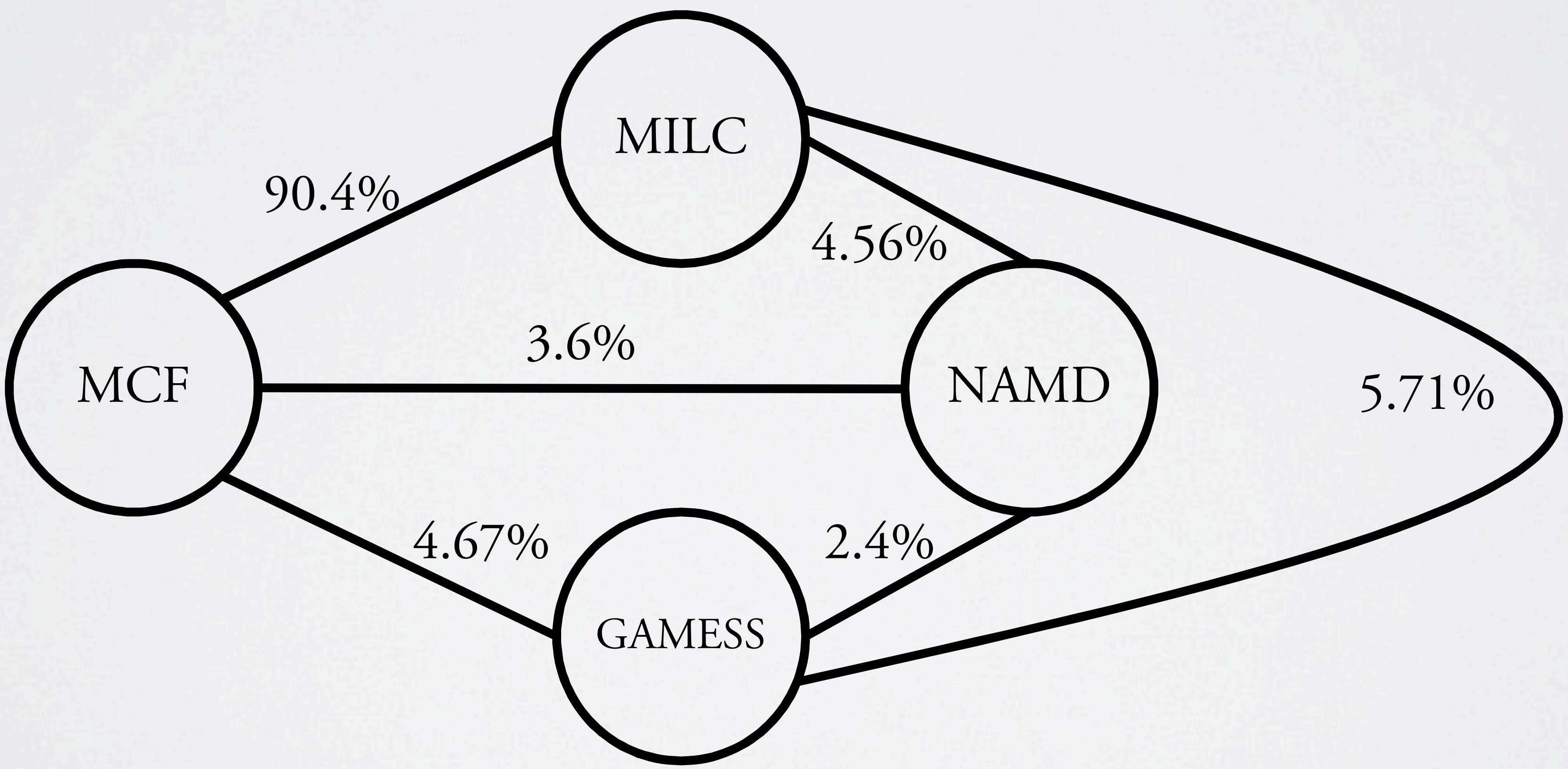
BASELINE: OPTIMAL ASSIGNMENT

Jiang's algorithm: measured co-run degradation + graph theory

	mcf	milc	gamess	namd
mcf	48.01%	65.63%	2.0%	2.11%
milc	24.75%	45.39%	1.23%	1.11%
gamess	2.67%	4.48%	-1.01%	-1.21%
namd	1.48%	3.45%	-1.19%	-0.93%

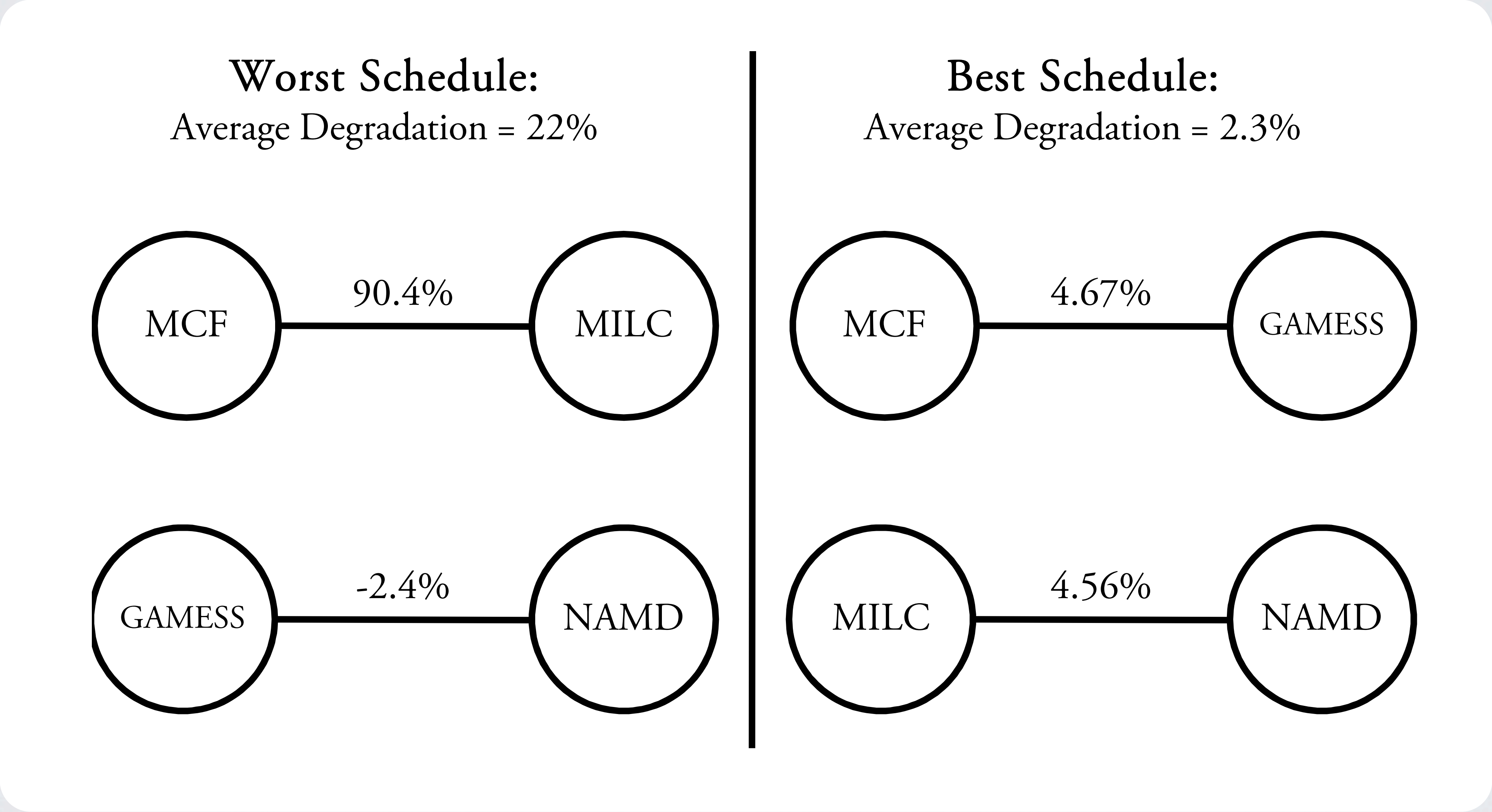
BASELINE: OPTIMAL ASSIGNMENT

Jiang's algorithm: measured co-run degradation + graph theory



BASELINE: OPTIMAL ASSIGNMENT

Jiang's algorithm: measured co-run degradation + graph theory



EVALUATING A CLASSIFICATION

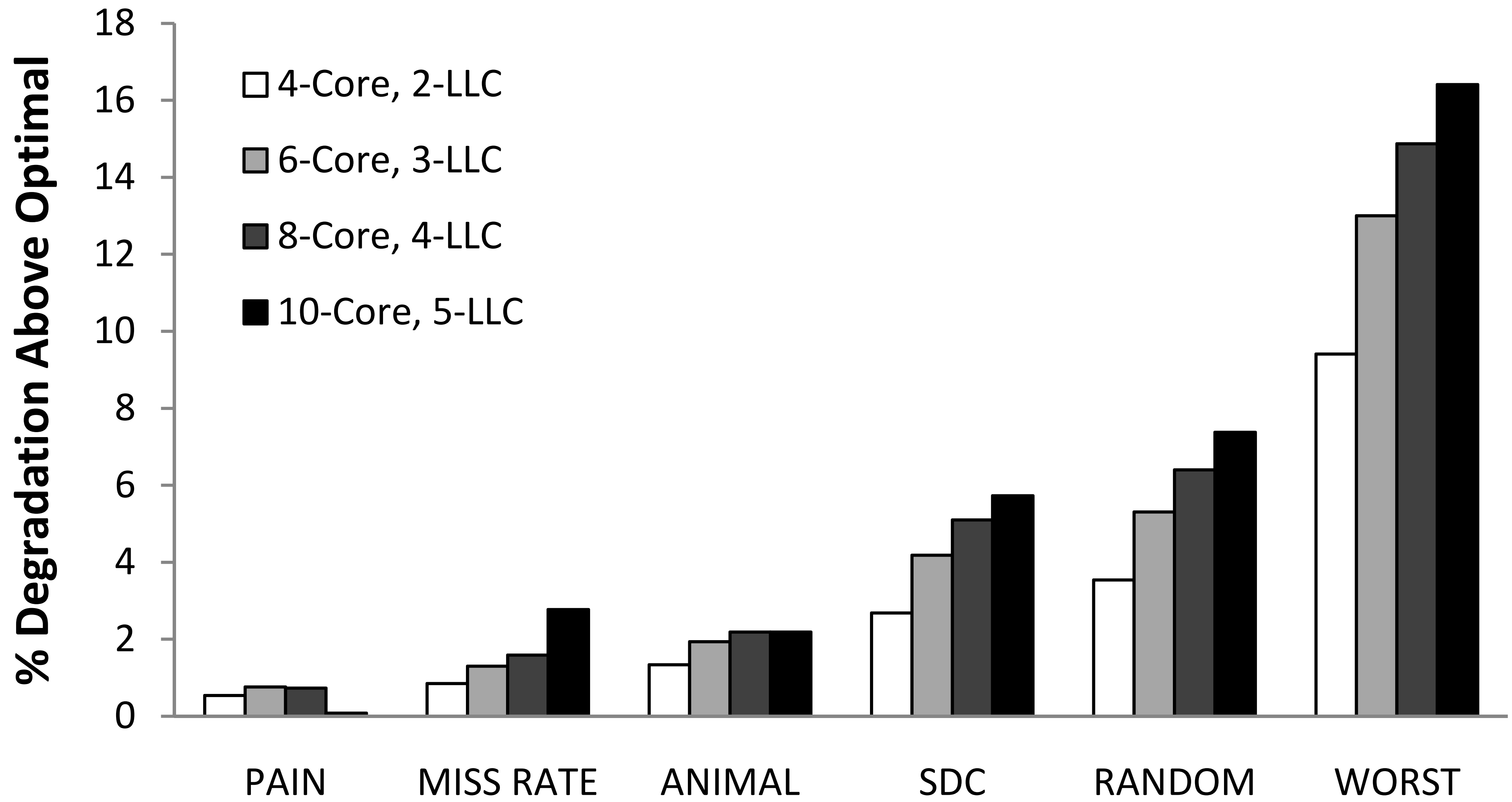
1. Baseline: optimal schedule with measured degradation
2. Contestant: estimated best schedule using classification
3. Compare performance degradation of both runs

STACK DISTANCE PROFILE

0	0	0	0		0

CANDIDATE CLASSIFICATIONS

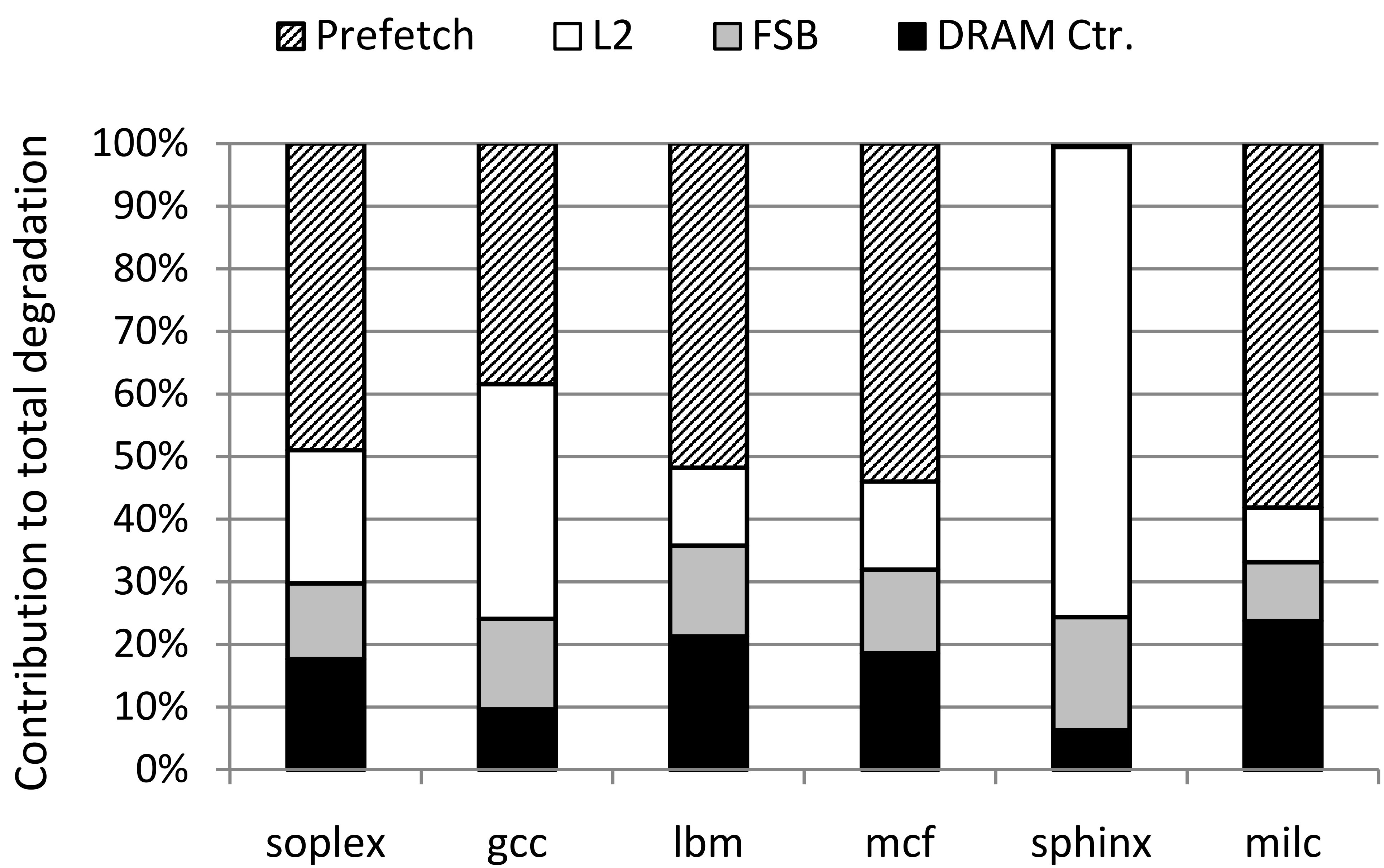
- Stack Distance Competition
- Animal Classes
- Miss Rate
- Pain



% Degradation Above Optimal
18
16
14
12
10
8
6
4
2
0
4-Core, 2-LLC
6-Core, 3-LLC
8-Core, 4-LLC
10-Core, 5-LLC
PAIN
MISS RATE
ANIMAL
SDC
RANDOM
WORST

SURPRISE

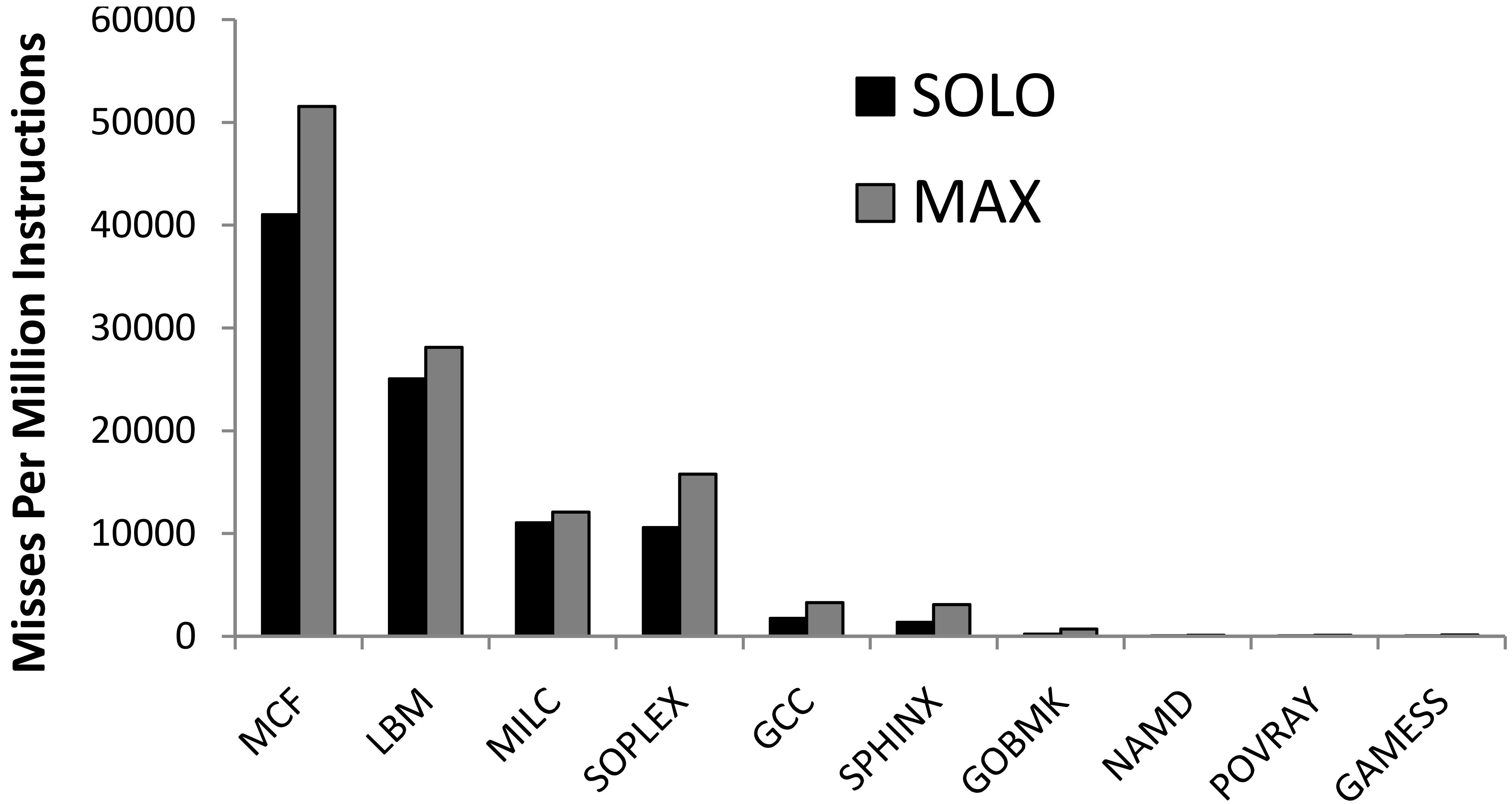
high miss rate → poor cache reuse → indifferent to contention


Prefetch
L2
FSB
DRAM Ctr.
Contribution to total degradation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
soplex
gcc
lbm
mcf
sphinx
milc

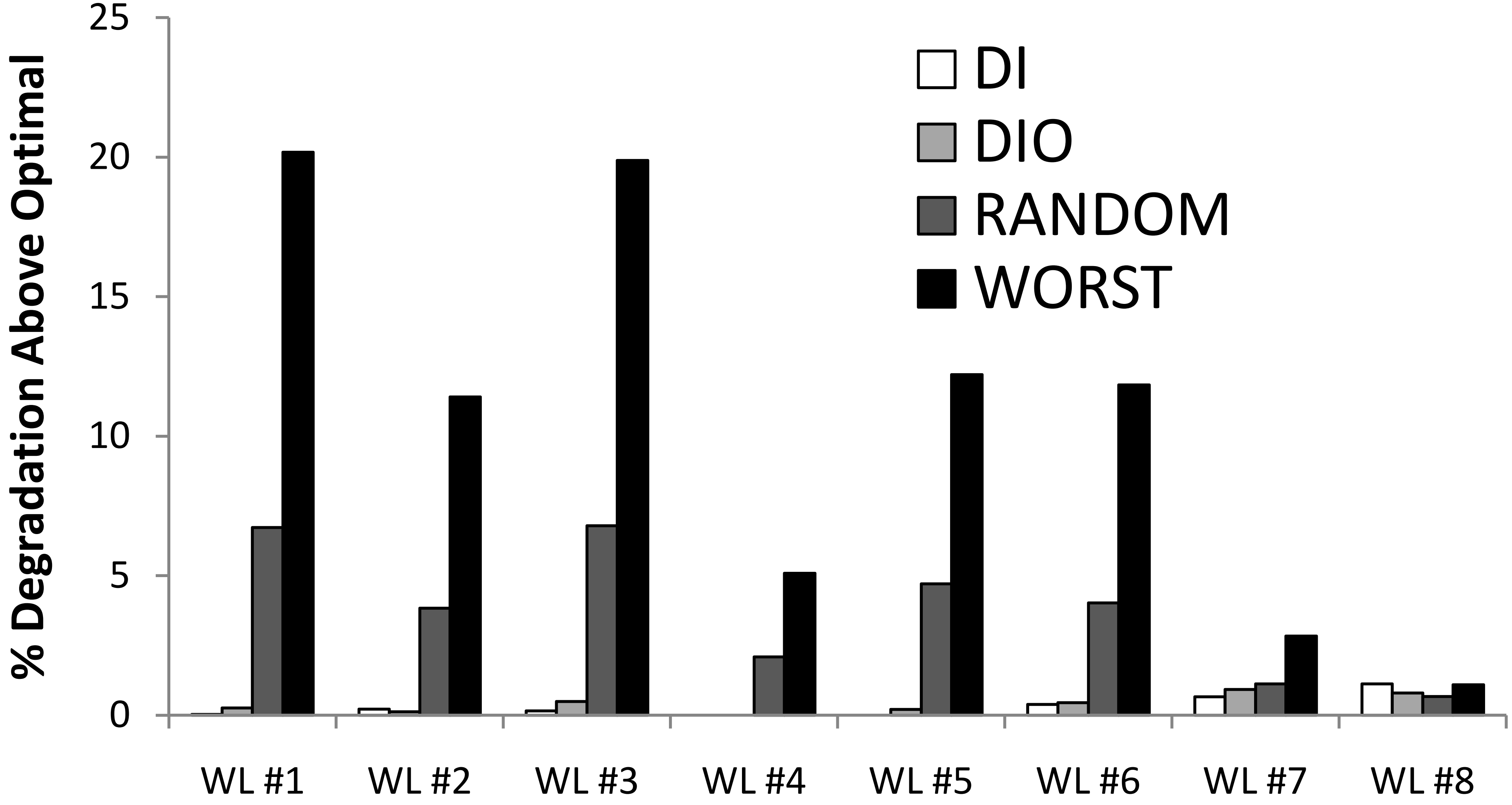
SCHEDULING ALGORITHM

1. Sort threads according to classifier.
2. Assign to cores using centralized sort.

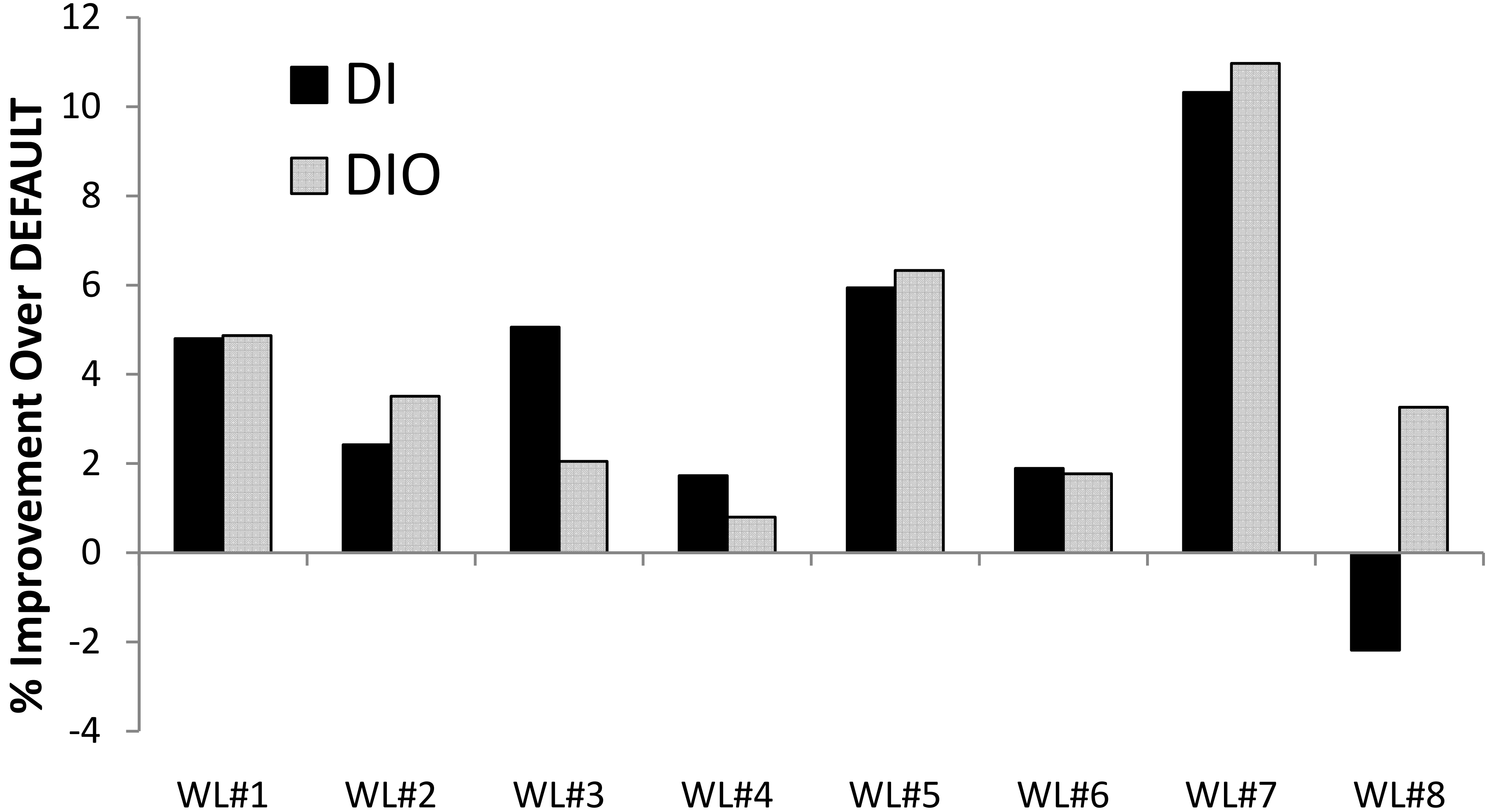
- Distributed Intensity (DI):
  miss rate determined from stack distance profile
- Distributed Intensity Online (DIO):
  miss rate determined using performance counters



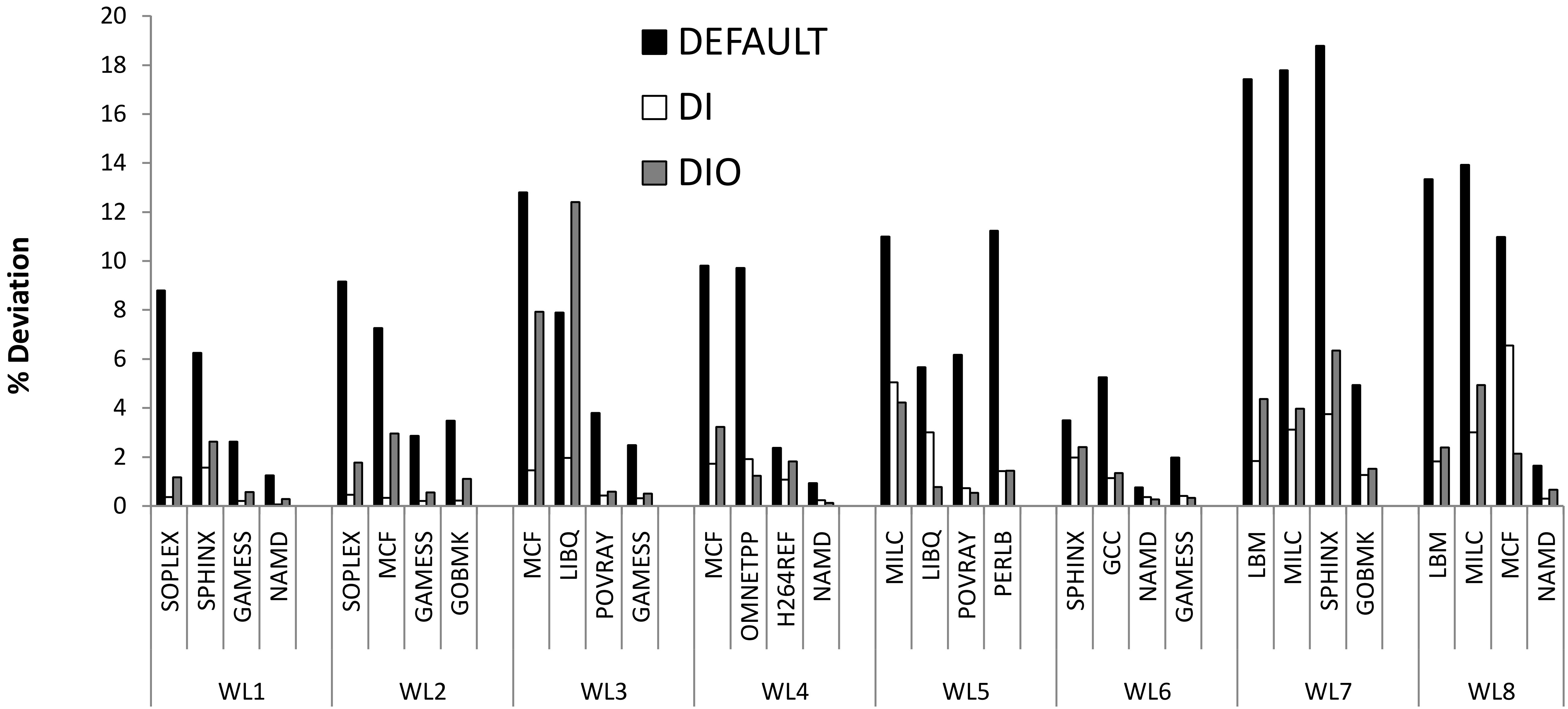
Misses Per Million Instructions
60000
50000
40000
30000
20000
10000
0
SOLO
MAX
MCF
LBM
MILC
SOPLEX
GCC
SPHINX
GOBMK
NAMD
POVRAY
GAMESS



% Degradation Above Optimal
25
20
15
10
5
0
WL #1
WL #2
WL #3
WL #4
WL #5
WL #6
WL #7
WL #8
DI
DIO
RANDOM
WORST



% Improvement Over DEFAULT
12
10
8
6
4
2
0
-2
-4
DI
DIO
WL#1
WL#2
WL#3
WL#4
WL#5
WL#6
WL#7
WL#8



DEFAULT
DI
DIO
% Deviation
20
18
16
14
12
10
8
6
4
2
0
SOPLEX
SPHINX
GAMESS
NAMD
WL1
SOPLEX
MCF
GAMESS
GOBMK
WL2
MCF
LIBQ
POVRAY
GAMESS
WL3
MCF
OMNETPP
H264REF
NAMD
WL4
MILC
LIBQ
POVRAY
PERLB
WL5
SPHINX
GCC
NAMD
GAMESS
WL6
LBM
MILC
SPHINX
GOBMK
WL7
LBM
MILC
MCF
NAMD
WL8

SUMMARY

- cache space is **not** the single most important bottleneck
- cache **miss rate** is a good predictor for contention
- contention-aware thread-core assignment improves **performance** and reduces **variability**
- using **performance counters** yields a practical contention-aware scheduler