School of Electrical & Computer Engineering
College of Engineering

Phase-Aware Scheduling for Heterogeneous Systems from Multicore Processors to the Cloud

Lina Sawalha

Dr. Ronald D. Barnes

Dr. Monte P. Tull

This work was supported in part by NSF award no. 1018771

Outline

- Introduction
- Goals
- Motivation
- Phase-identification based scheduling
  - Phase-IPC scheduling method
  - Phase-Sampling scheduling method
- Results for heterogeneous multicore processors
- Scheduling jobs in the cloud
- Conclusions

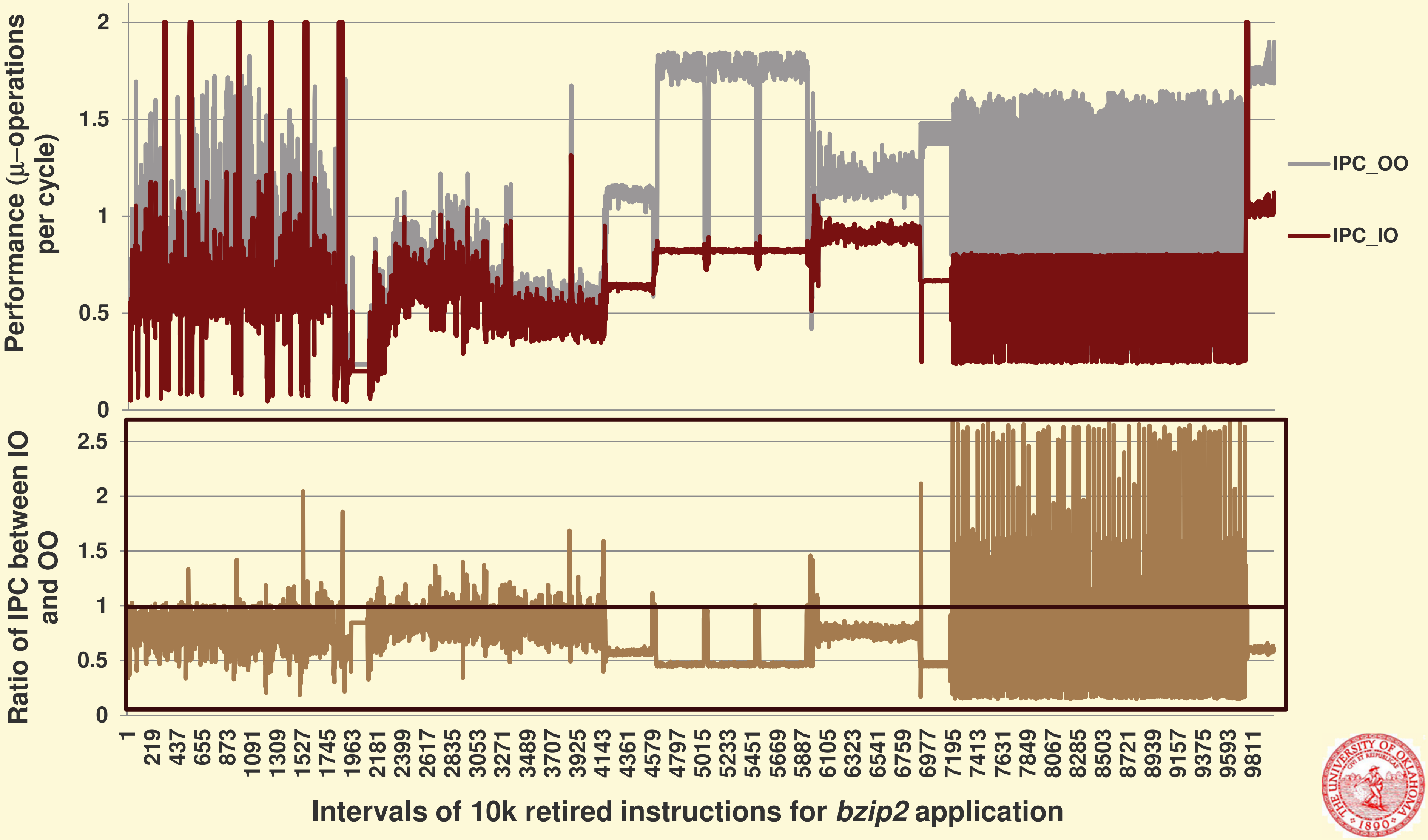
Introduction

- ♦ Heterogeneous multiprocessor systems offer advantages in terms of both performance and power consumption.
- ♦ Assigning applications to the different types of cores is complicated.
- ♦ Asymmetric cores ➡ different performance
- ♦ Different phases of execution for each application

Introduction

- Correlation between executing phases and program behavior
- Dynamic Scheduler:
  - Identifies program phases
  - Stores information about phases
  - Recognizes occurrences of the same phases
  - Reuse stored information for scheduling
- Extending phase-based scheduling to the cloud

Motivation


Performance (μ–operations per cycle)
2
1.5
1
0.5
0
IPC_OO
IPC_IO
Ratio of IPC between IO and OO
2.5
2
1.5
1
0.5
0
1
219
437
655
873
1091
1309
1527
1745
1963
2181
2399
2617
2835
3053
3271
3489
3707
3925
4143
4361
4579
4797
5015
5233
5451
5669
5887
6105
6323
6541
6759
6977
7195
7413
7631
7849
8067
8285
8503
8721
8939
9157
9375
9593
9811
Intervals of 10k retired instructions for *bzip2* application

Related Work

- Static Approaches:

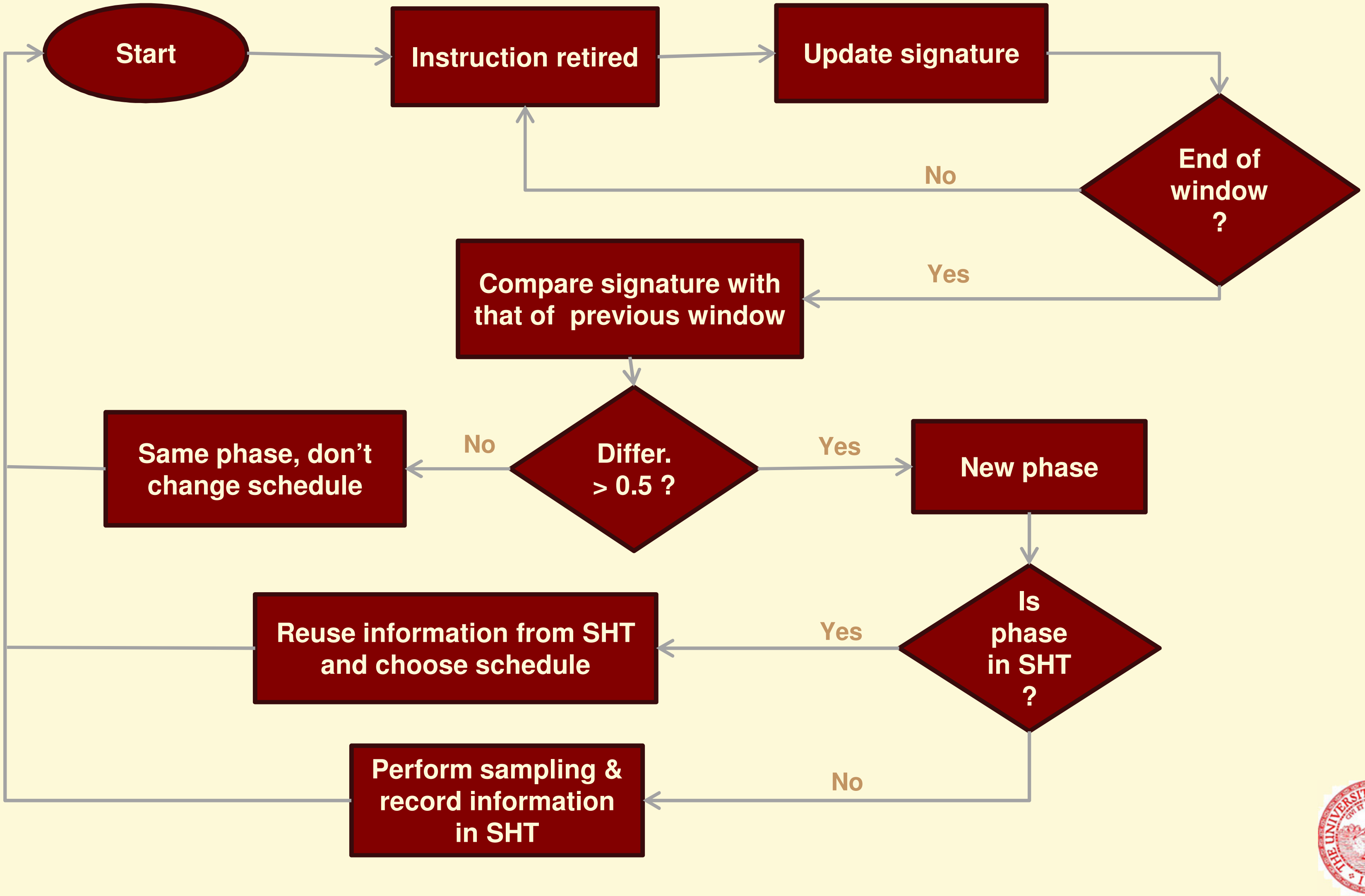
  [Chen’09], [Shelepov’08 & 09], [Lakshminarayana’09]
- Dynamic Approaches:
  - Heuristic Sampling [Kumar’04], [Becchi’06]
  - History-aware scheduler [Jooya’09]
- Static/Dynamic: Phase-based approach [Sondag’11]

Phase Identification Method

- Working set signatures [Dhodapkar’02]*
  - Working set: Compressed representation of program behavior
  - Non-overlapping windows of retired instructions
  - Signature calculated by hashing some bits from program counter to identify a working set

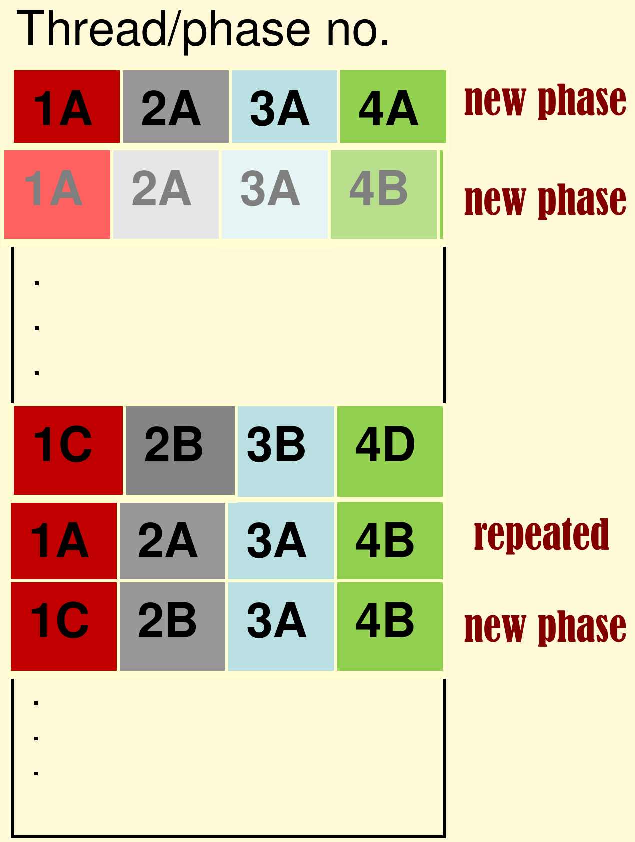
* A. S. Dhodapkar and J. E. Smith, “Managing multi-configuration hardware via dynamic working set analysis,” ACM SIGARCH Computer Architecture News, vol. 30, no. 2, pp. 233–244, May 2002.

Phase-Identification Based Scheduling


Start
Instruction retired
Update signature
End of window ?
No
Yes
Compare signature with that of previous window
Same phase, don’t change schedule
No
Differ. > 0.5 ?
Yes
New phase
Reuse information from SHT and choose schedule
Yes
Is phase in SHT ?
Perform sampling & record information in SHT
No

Phase-Sampling

- ♦ Sampled performance evaluation
- ♦ New *set of phases* ➡ sampling
- ♦ Select the highest throughput schedule & record in the SHT
- ♦ Reuse the recorded schedule when encountering the same *set of phases* again

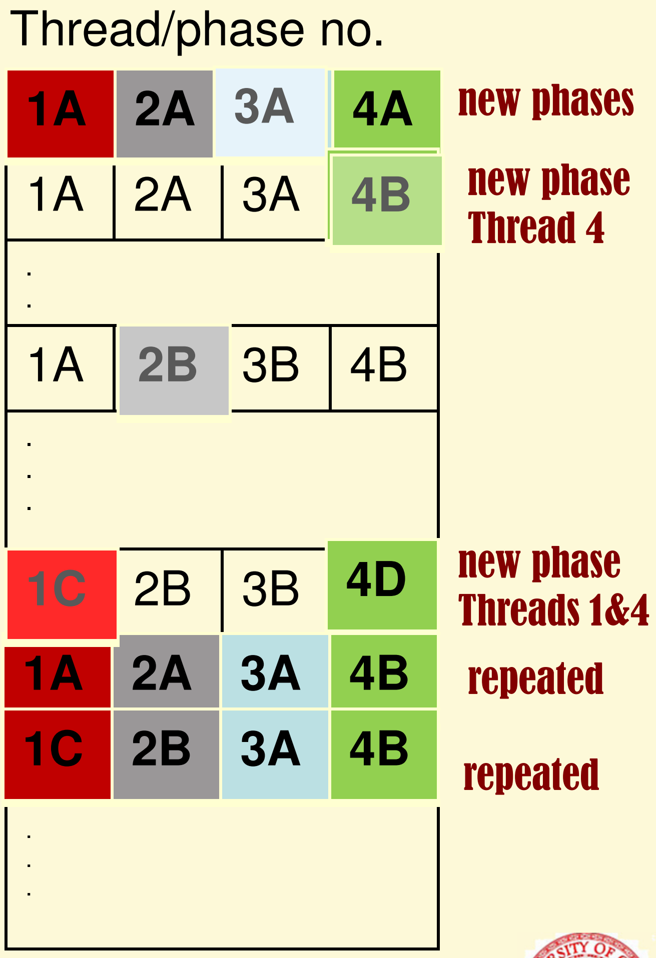


Thread/phase no.

1A	2A	3A	4A	new phase
1A	2A	3A	4B	new phase
⋮				
1C	2B	3B	4D	
1A	2A	3A	4B	repeated
1C	2B	3A	4B	new phase
⋮				

Phase-IPC

- New phase for one thread
  - ➡ Sampling for that thread only
- Record IPC for each phase on each core along with the signature
- Best schedule predicted based on estimated throughput of all the different combinations



Thread/phase no.

1A	2A	3A	4A	new phases
1A	2A	3A	4B	new phase Thread 4
⋮				
1A	2B	3B	4B	
⋮				
1C	2B	3B	4D	new phase Threads 1&4
1A	2A	3A	4B	repeated
1C	2B	3A	4B	repeated
⋮				

Evaluation Metrics

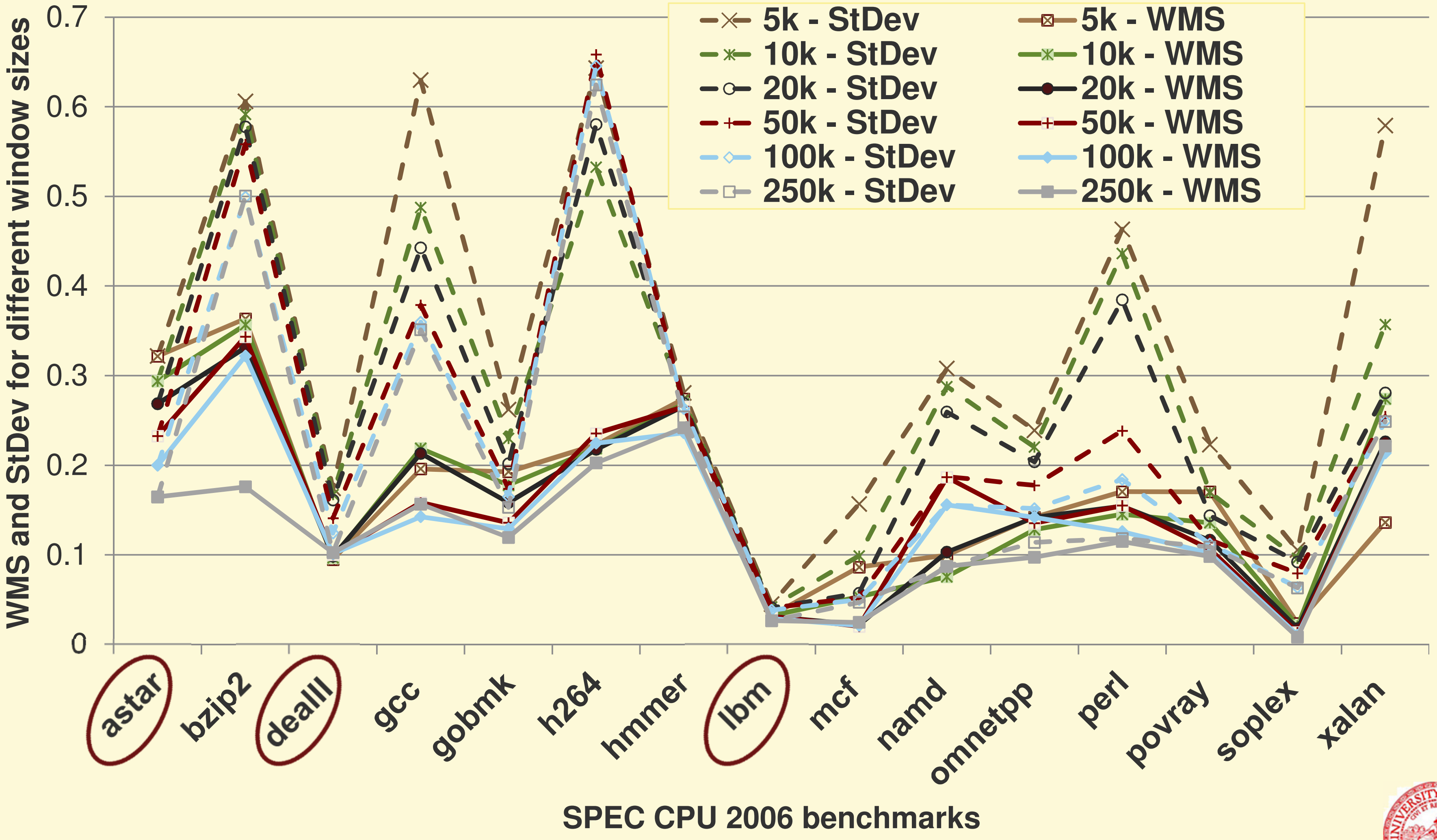
- Need a metric that balances throughput and individual thread performance
- Instructions per cycle (IPC)
- Weighted Speedup: $\frac{IPC_{\ actual\ core}}{IPC_{\ fastest\ core}}$
  - Requires oracle knowledge of best IPC
  - Not suitable input for scheduling heuristic
  - Used for comparing our approaches with other methods

Methodology

- Soonergy Simulator: A cycle-accurate architectural and micro-architectural simulator
- 15 integer and floating-point benchmarks from SPEC CPU2006 benchmark suite
- 250 million x86 instructions
- Four different core configurations:

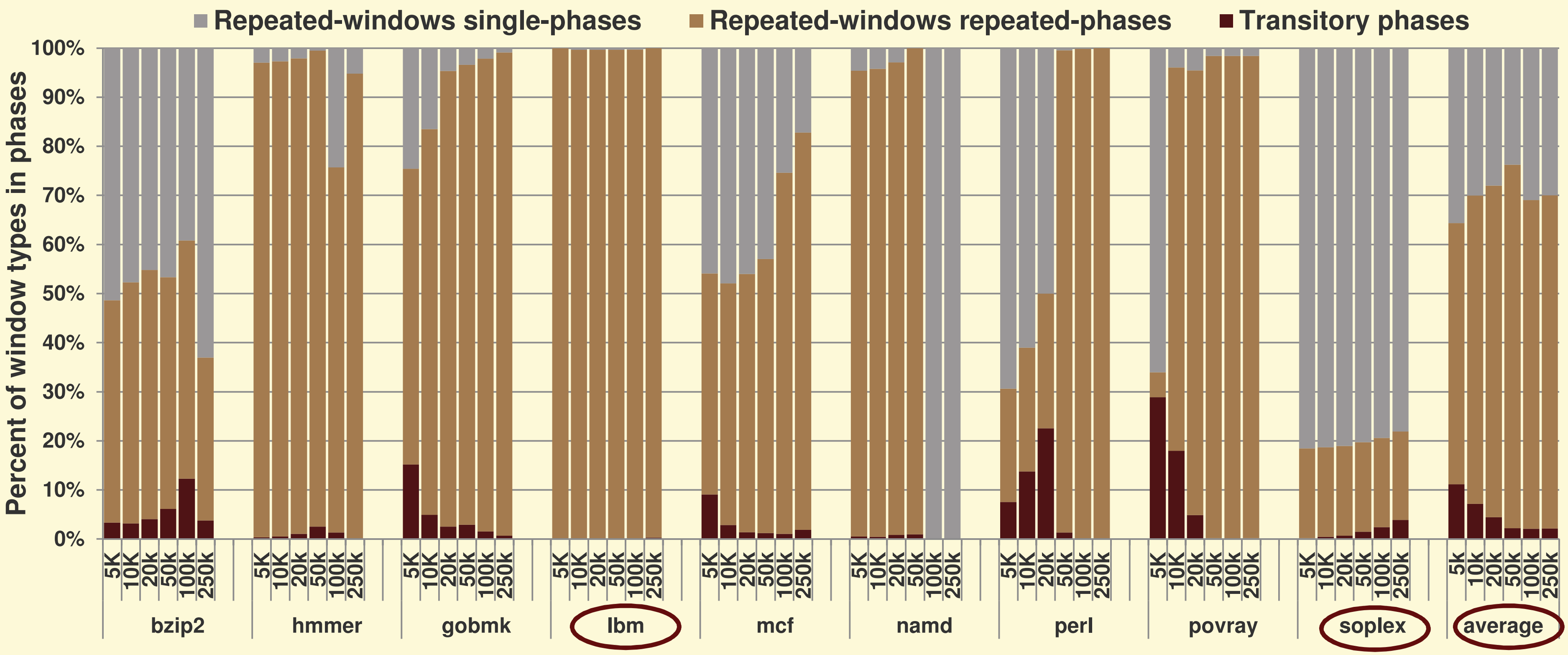
Parameter	Core 0	Core 1	Core 2	Core 3
Execution	IO	OO	OO	OO
Issue width	4	4	3	2
L1 cache	32KB	32KB	16KB	16KB
ROB	N/A	128	96	64
RS	N/A	32	24	16

Results

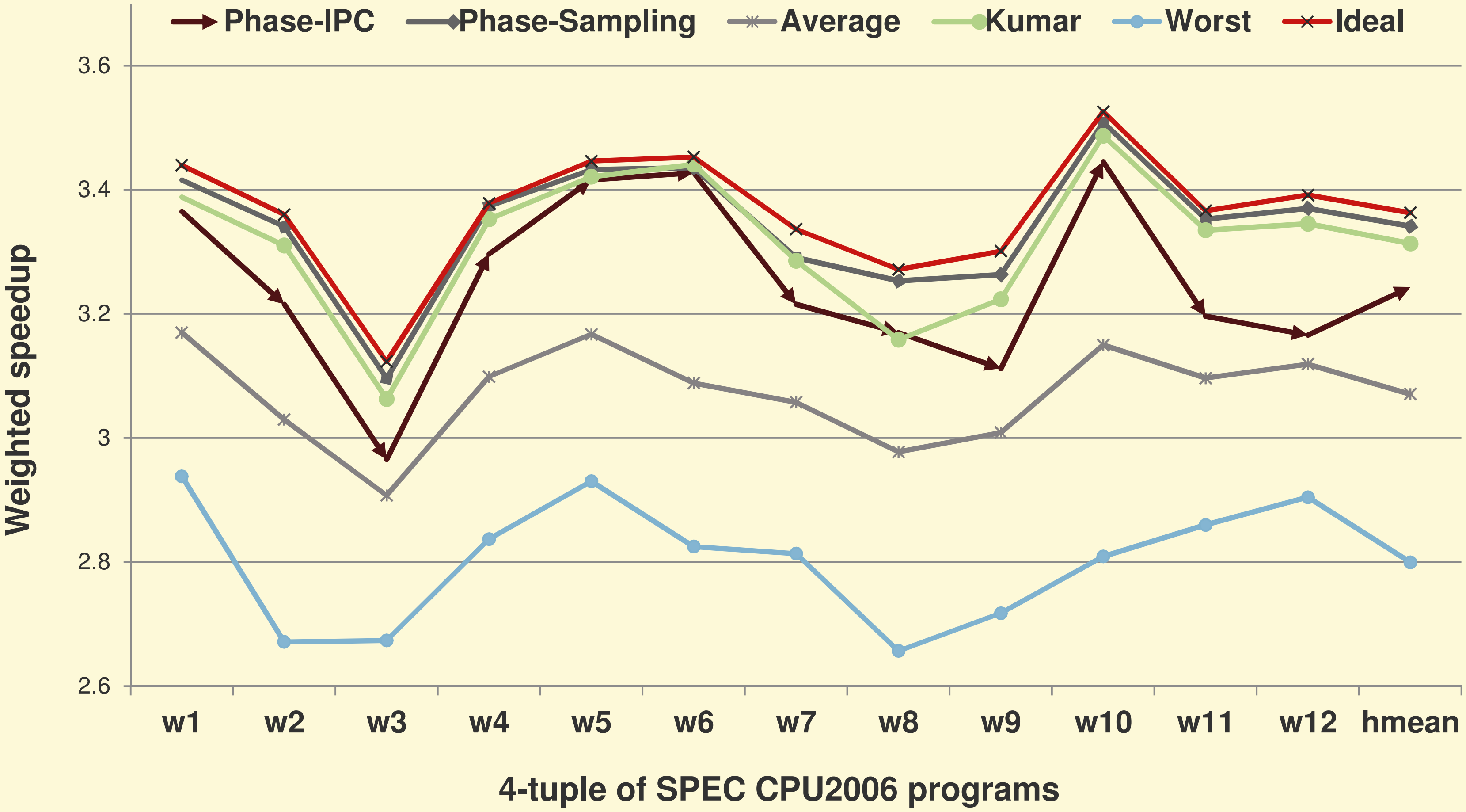


WMS and StDev for different window sizes
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5k - StDev
5k - WMS
10k - StDev
10k - WMS
20k - StDev
20k - WMS
50k - StDev
50k - WMS
100k - StDev
100k - WMS
250k - StDev
250k - WMS
astar
bzip2
dealII
gcc
gobmk
h264
hmmer
lbm
mcf
namd
omnetpp
perl
povray
soplex
xalan
SPEC CPU 2006 benchmarks

Results


Repeated-windows single-phases
Repeated-windows repeated-phases
Transitory phases
Percent of window types in phases
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
5K
10K
20k
50k
100k
250k
bzip2
hmmer
gobmk
lbm
mcf
namd
perl
povray
soplex
average

Performance Results



HOW MIGHT THIS WORK FOR THE CLOUD?

Cloud Computing

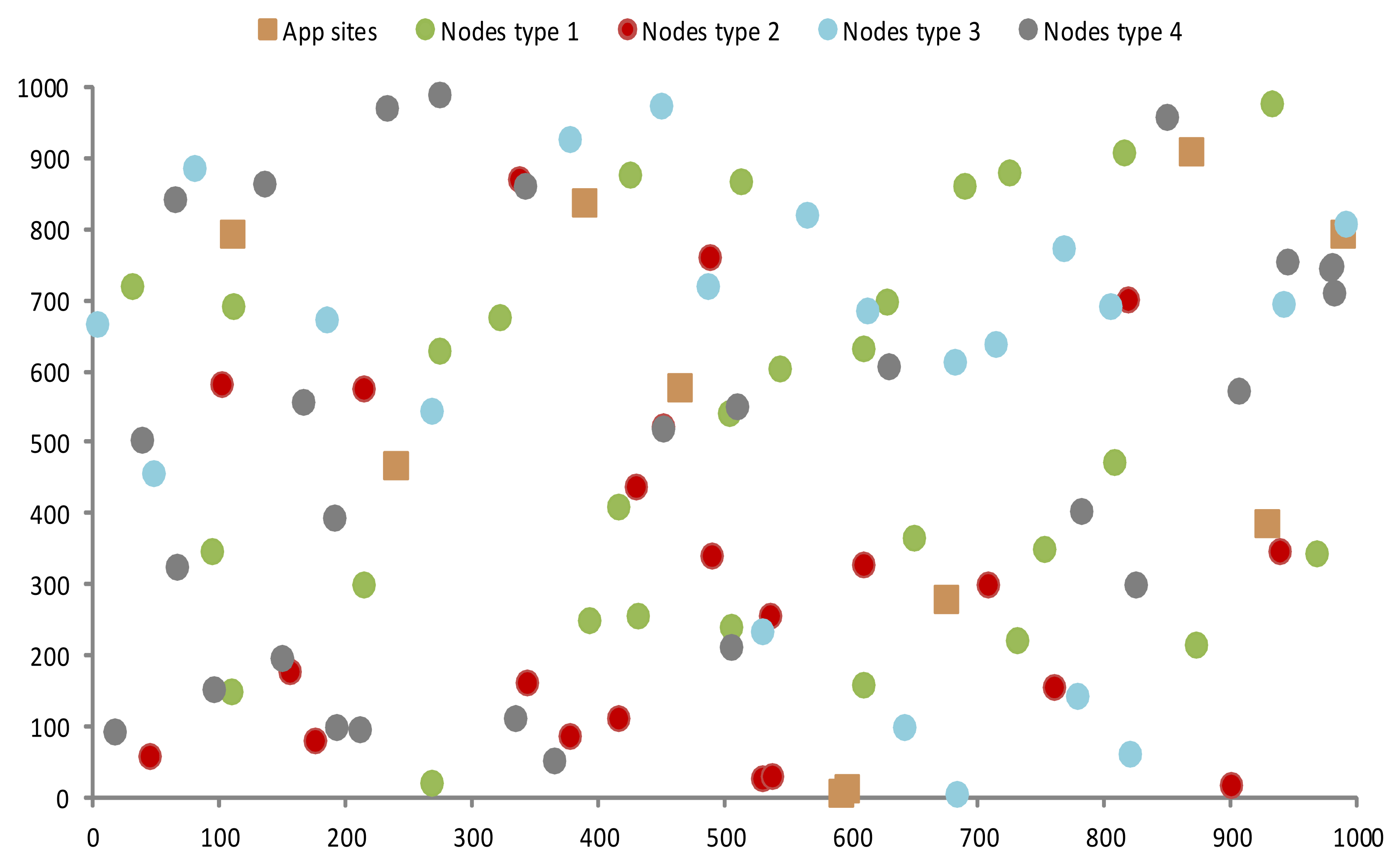
- Distributed computing
- Computing nodes spread over different places
- Heterogeneous computing nodes

- Need to find the best job to node map
- Use phase-aware scheduling to re-schedule jobs during runtime

Cloud World Description

- World:
  - 1000 km x 1000 km
  - 100 computing nodes
  - 10 submission sites
- Jobs
  - SPEC CPU 2006 benchmarks
  - Random exponential arrival time
  - Random exponential length
  - Communication cost by distance

World Map


App sites
Nodes type 1
Nodes type 2
Nodes type 3
Nodes type 4
1000
900
800
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000

Random Scheduling

- The scheduler assigns a free computing node randomly to each job.
- Distance of nodes is not considered
- Jobs are not rescheduled dynamically
- Wait list contains jobs waiting for free nodes
- Not efficient scheduling method

Proposed Phase-Guided Scheduling

- Each execution phase evaluated on the different node types
- If available free nodes of different types, replicate job on the different node types
  - After window elapsed choose the best performing node
  - Kill jobs on other nodes

Phase-Based Scheduling

- If no free nodes available for evaluation
  - switch with closest job not in evaluation period
- Evaluate current job and switched jobs
- Choose the assignment that leads to the best overall performance for the current and switched jobs.

Future Work

- More on scheduling jobs for the cloud
- Approach can be extended to fully multithreaded multi-program workload
- Fast context switch for multicore processor
- Power consumption

Conclusions

- ♦ Dynamic Scheduler:
  - ♦ Identifies program phases
  - ♦ Stores information about phases
  - ♦ Recognizes occurrences of the same phases
  - ♦ Reuse stored information for scheduling
- ♦ Phase-Sampling outperforms Phase-IPC and previous scheduling methods but incurs more sampling
- ♦ Phase-IPC requires many fewer sampling intervals and no permutation of threads across each core type

QUESTIONS?