MLP-Aware Dynamic Instruction Window Resizing for Adaptively Exploiting Both ILP and MLP

Yuya Kora
Kyohei Yamaguchi
Hideki Ando

Nagoya University

Single-Thread Performance

- In the past: Pollack’s law
  - Performance improvement $\propto \sqrt{area}$
- Recently: little improve
  - Despite ever increasing transistor budget

- Use increased transistor budget effectively to improve single-thread performance

Memory Wall

- Large speed gap between processor and main memory
- Conventional solutions:
  - Large cache
    - Expensive: several MB is not enough
  - Hardware prefetcher
    - Effective only for regular access patterns

Aggressive Out-of-Order Execution

- Significantly increase #in-flight instructions through extensive instruction window resources
  - Issue queue, reorder buffer, load/store queue

- Allow parallel memory accesses by executing cache-miss loads as early as possible
  - Reduce effective memory access latency
  - MLP: memory-level parallelism

Advantage and Disadvantage

- Advantage
  - Data fetch is accurate
  - Simple extension of conventional processor

- Disadvantage
  - Large resources lengthen clock cycle time

Outline

- Tradeoff Involved in Enlarging Window Resources for Aggressive Out-of-Order Execution
- Dynamic Instruction Window Resizing
- Evaluation
- Conclusion

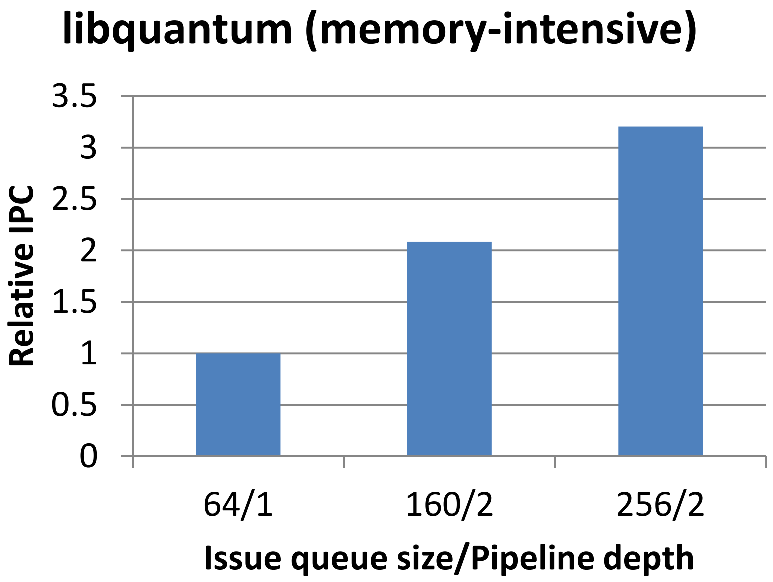
Performance Problem in Enlarged Window Resources

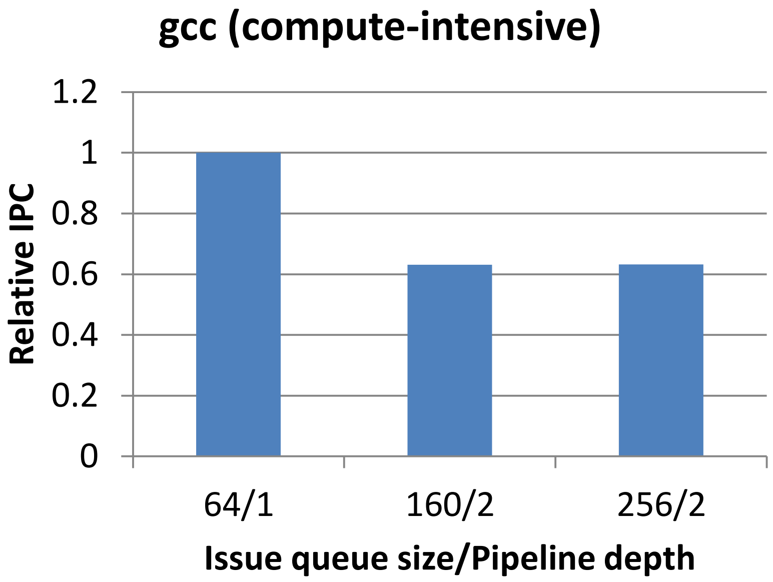
- Large window resources lengthen the clock cycle time
  - Can be solved by pipelining resources

- Prevent ILP from being exploited
  - Pipelined issue queue cannot issue dependent instructions back-to-back
  - Reduce IPC

- Tradeoff
  - Large window: beneficial for MLP, but harmful for ILP
  - Small window: beneficial for ILP, but cannot exploit MLP

Example Illustrating Tradeoff





- Large window is beneficial, even if it is pipelined
- MLP is exploited

- Large window is harmful, because IQ is pipelined

Outline

- Tradeoff Involved in Enlarging Window Resources for Aggressive Out-of-Order Execution
- Dynamic Instruction Window Resizing
- Evaluation
- Conclusion

Dynamic Instruction Window Resizing

- Adapt window size to available parallelism
  - ILP or MLP
- As more exploitable MLP is predicted
  - Window resources are enlarged and pipeline depth is increased
- If prediction indicates less MLP is exploited (= ILP is more valuable)
  - Window resources are shrunk and pipeline depth is decreased

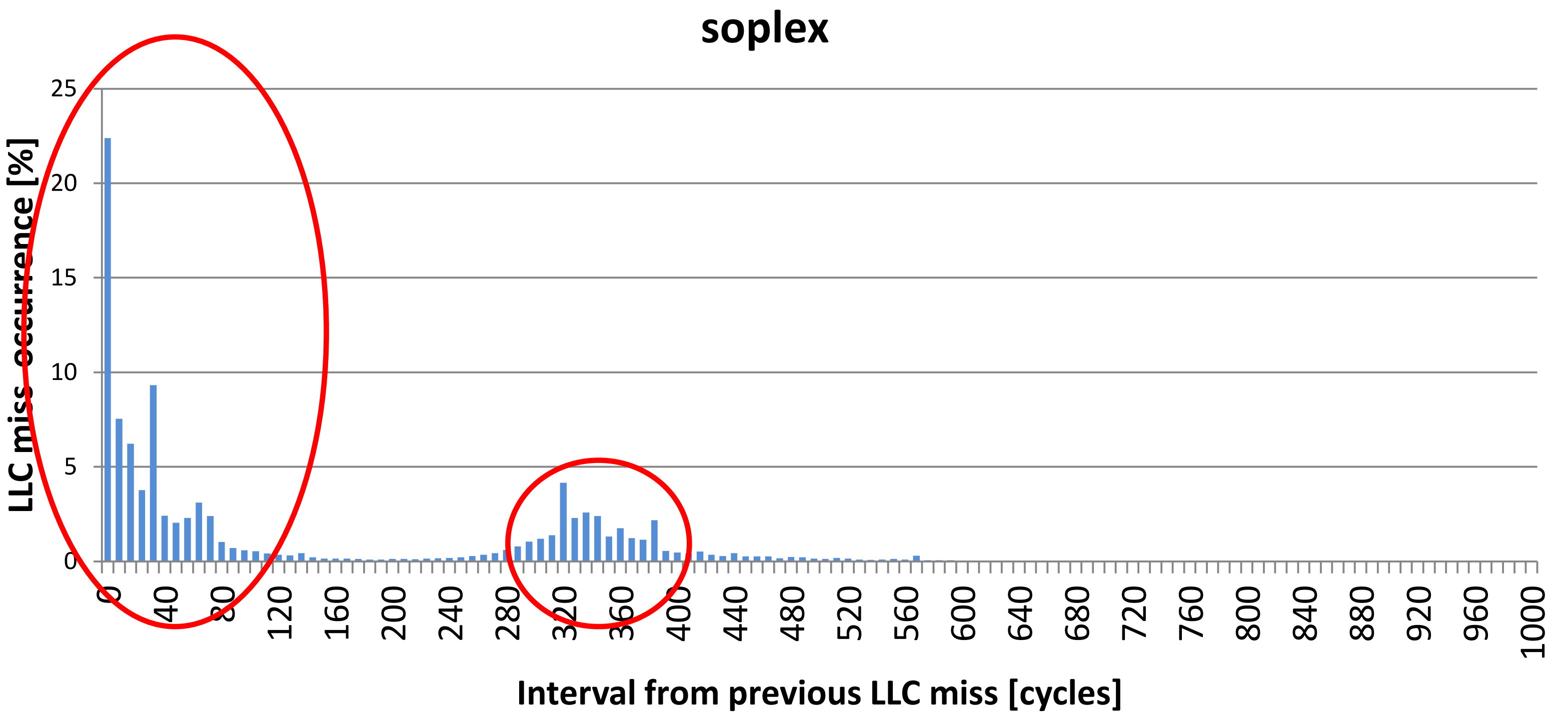
Prediction when MLP is Exploitable

- If an LLC miss occurs once
  - Predict that MLP is exploitable for a while

- If memory latency has lapsed after the last LLC miss
  - Predict that MLP will not be exploitable

Rationale of Prediction

- LLC misses are typically clustered

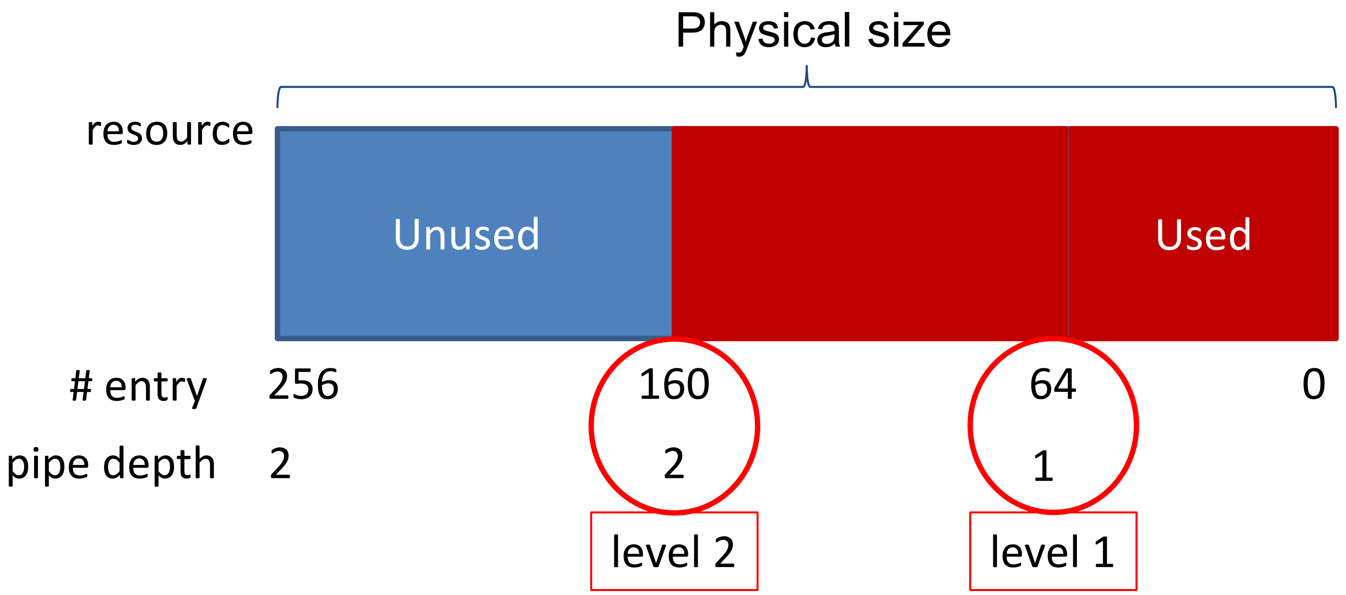


Term: *level*

- Instruction window resource *level*
  - {size, pipeline-depth} of the resource

- As level number increases
  - Size increases
  - Each resource is pipelined so that it does not increase the clock cycle time

What is Level Transition?



When the current size is 64 entries with 1-stage pipeline, and we enlarge to 160 entries with 2-stage pipeline, we say that the level is increased or transits from 1 to 2.

Algorithm: Put it Together

- If LLC miss occurs:
  - Increase level of the resources

- If memory latency has lapsed after last LLC miss:
  - Check if resources are all shrinkable
  - If so, it decreases level
  - Otherwise, it stops resource allocation to increase vacancies in resources, and postpones level decrease until shrinkable

Outline

- Tradeoff Involved in Enlarging Window Resources for Aggressive Out-of-Order Execution
- Dynamic Instruction Window Resizing
- Evaluation
  - Environment
  - IPC
  - Energy Efficiency
  - Cost Efficiency
  - Comparison with Runahead Execution
- Conclusion

Configuration of Base Processor

Architecture type	Intel P6
Issue width	4
ROB	128 entries
IQ	64 entries
LSQ	64 entries
L1 I-cache	64KB, 2-way, 32B line
L1 D-cache	64KB, 2-way, 32B line, 2 ports, 2-cycle hit latency
L2 cache	LLC, 2MB, 4-way, 64B line, 12-cycle hit latency
Main memory	300-cycle minimum latency
Data prefetcher	stride-based, 4K-entry, 4-way table, 16-data on L2 miss

Size and Pipeline Depth of Resources

resource	parameter	level			
		1	2	3	
IQ	entries	64	160	256	
	pipe depth	1	2	2	HSPICE
ROB	entries	128	320	512	
	pipe depth	1	2	2	CACTI
LSQ	entries	64	160	256	
	pipe depth	1	2	2	

- Has physically 4x lager window resources than base
- Can be configured to one of three levels
- Clock cycle time is determined by IQ delay in base processor

Environment for Performance Evaluation

- Simulator based on SimpleScalar 3.0a

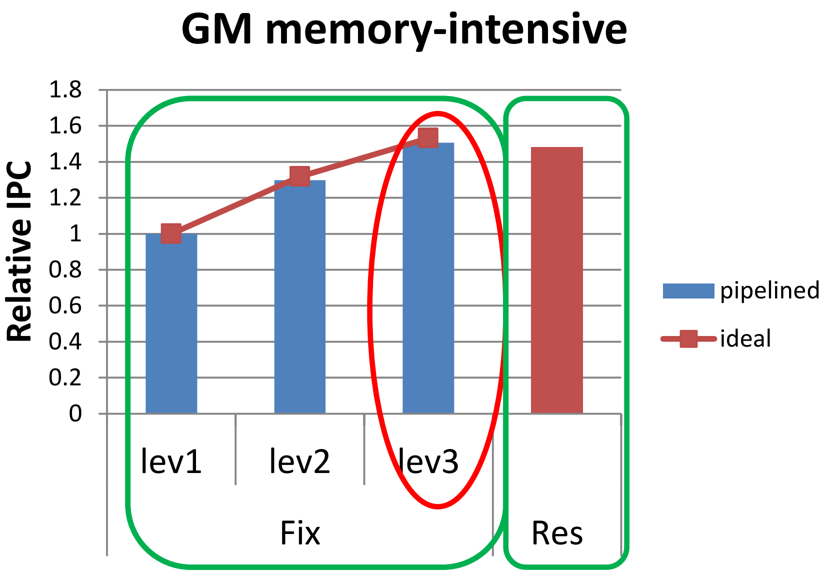
- Alpha ISA

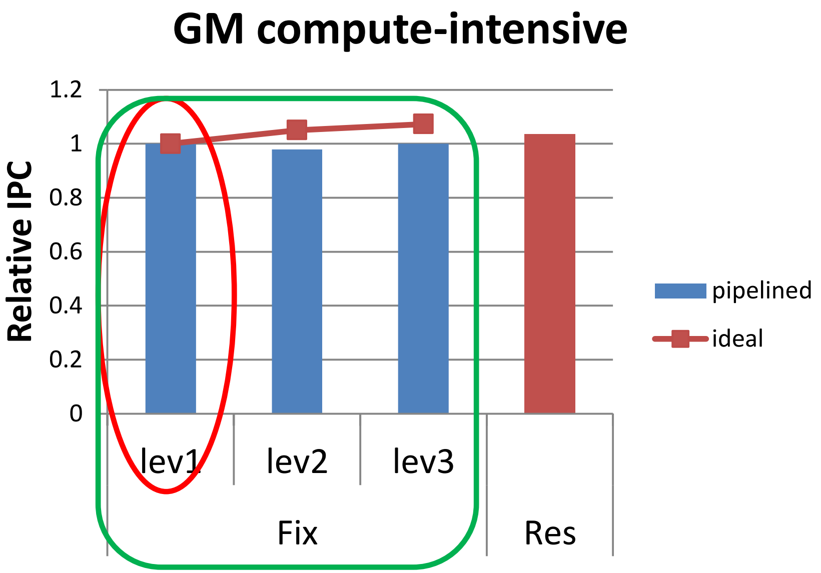
- SPEC2006 benchmark programs
  - Mem-intensive: AVG load latency ≥ 10 cycles
  - Comp-intensive: AVG load latency < 10 cycles

IPC: Evaluation Models

- **Fixed size model**: Size of window resources is FIXED during execution. At levels 2 and 3, resources are pipelined, which causes issue bubble and extra branch misprediction penalty.
- **Dynamic resizing model**: Size of window resources is DYNAMICALLY RESIZED. At levels 2 and 3, resources are pipelined, which causes issue bubble and extra branch misprediction penalty.
- **Ideal model**: Same as the fixed size model, but NOT pipelined. No penalty related to pipelining is imposed.

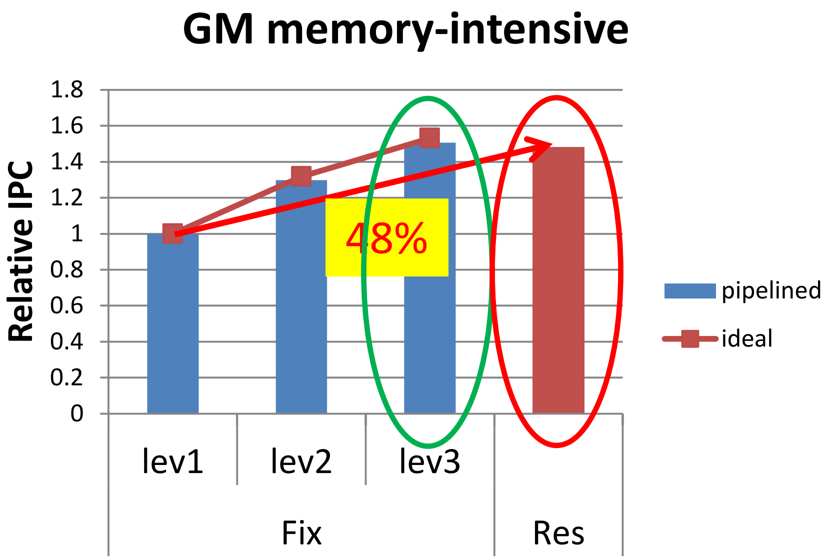
IPC: Fixed Size Model

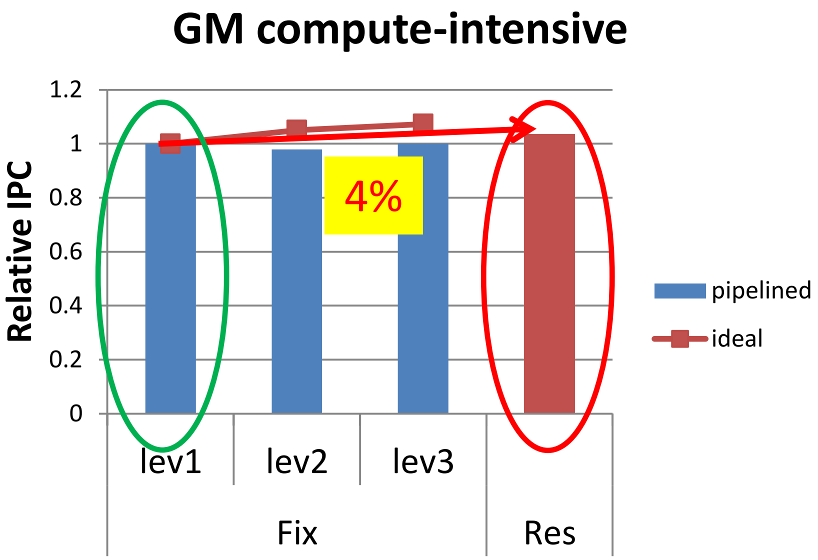




- **mem-intensive programs**: Level 3 achieves best performance. MLP is exploited aggressively with a large window.
- **comp-intensive programs**: Performance is not so sensitive to level, but level 1 is the best.
- Best resource level is different depending on program, when the size is fixed.

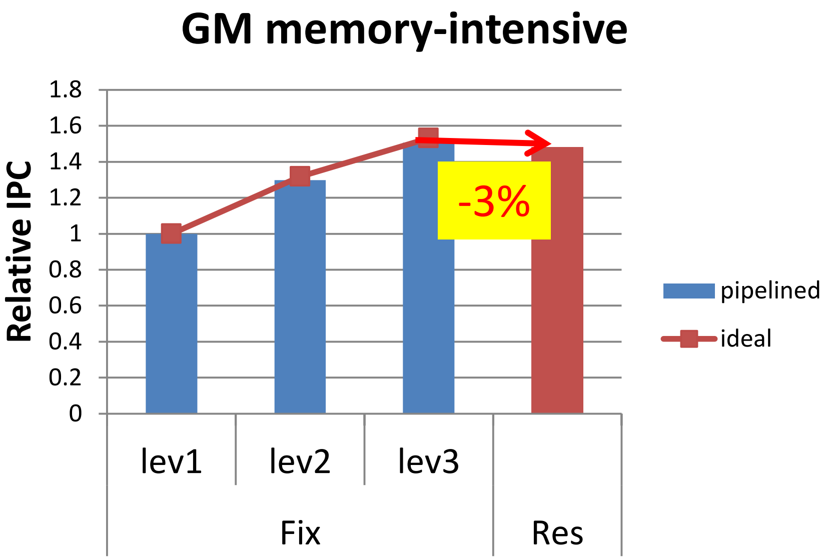
IPC: Dynamic Resizing Model

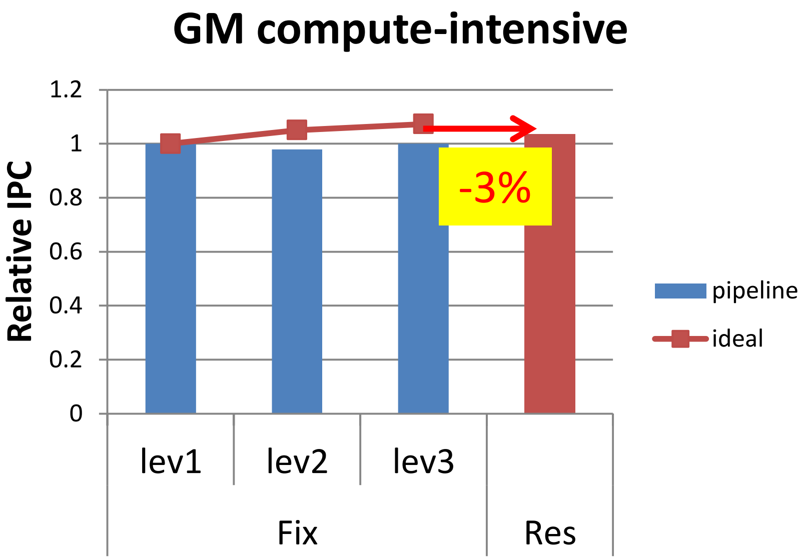




- Dynamic resizing model achieves as good as best performance for levels 1 to 3 of fixed size model.
- Imply good adaptability
- 21% speedup for all programs

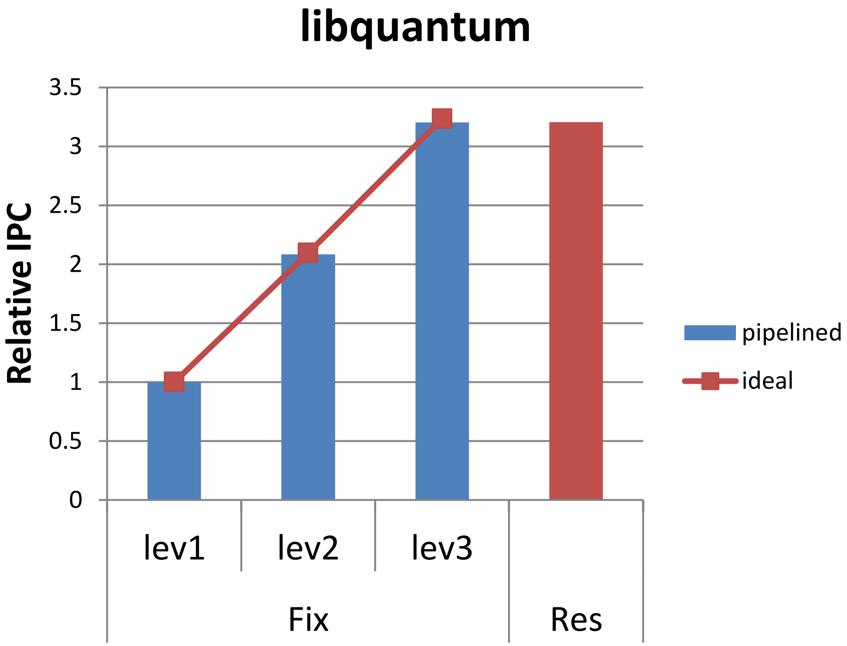
IPC: Ideal Model

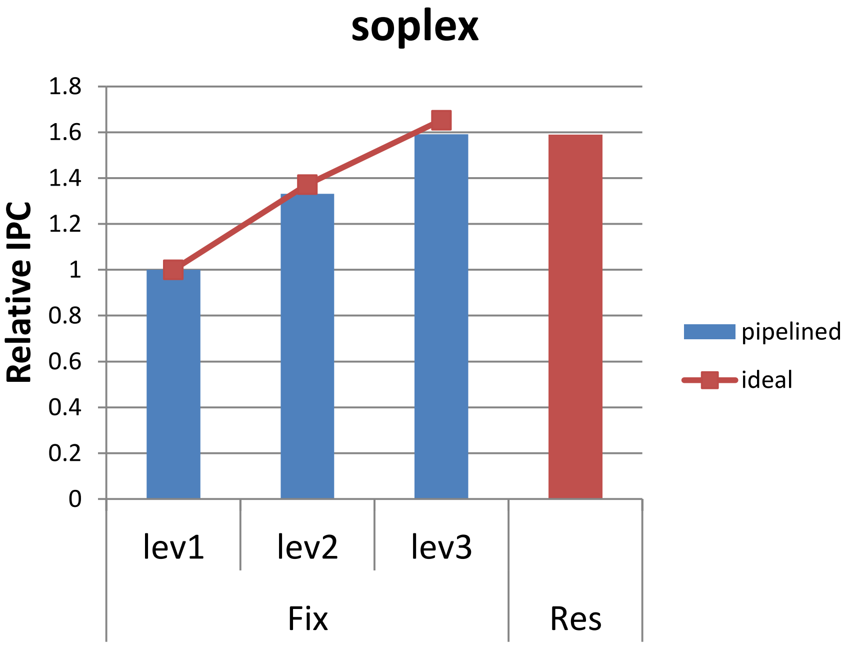




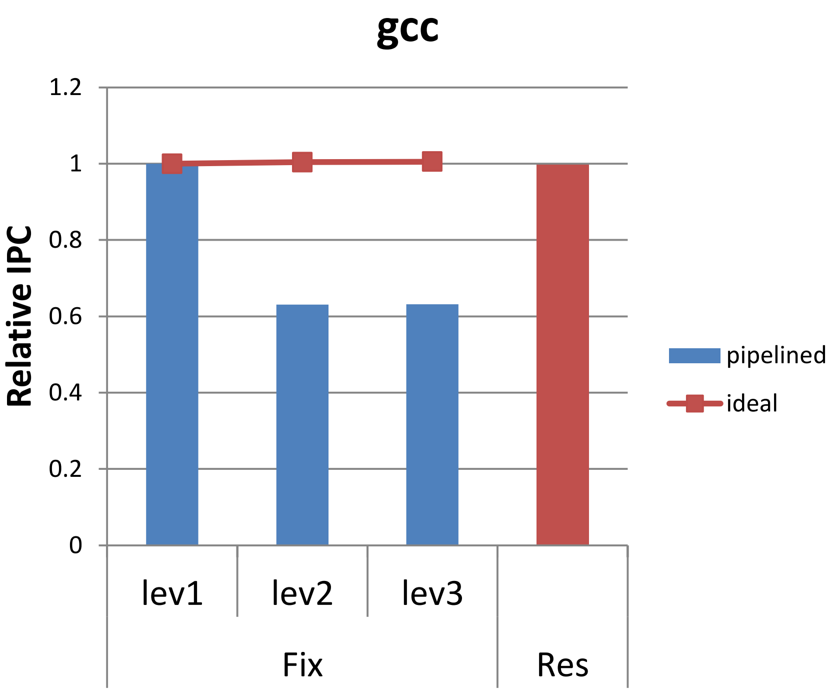
- No significant degradation in dynamic resizing model
- Imply good adaptability

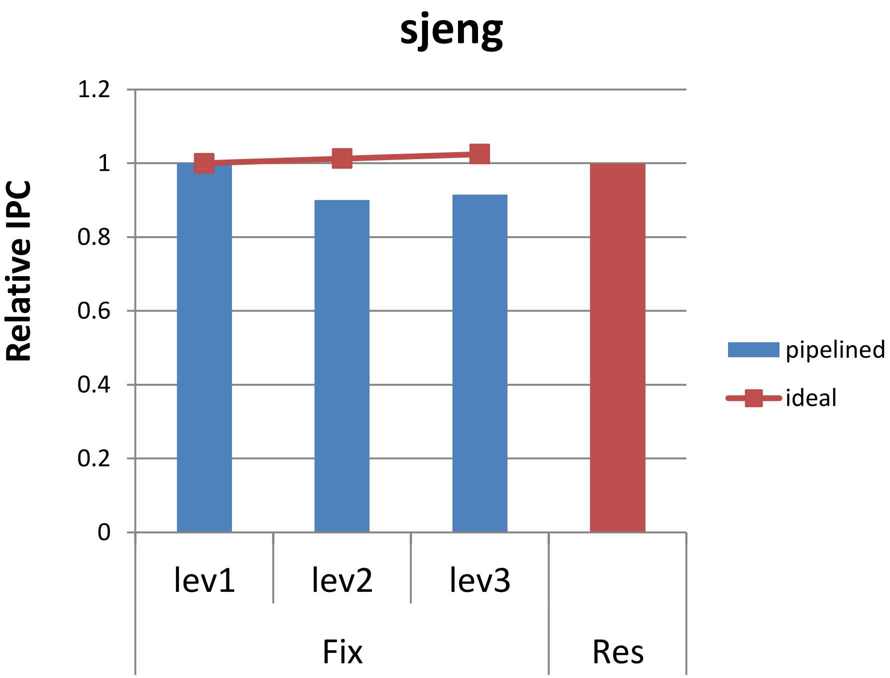
Samples from Memory-intensive Programs





Samples from Compute-intensive Programs

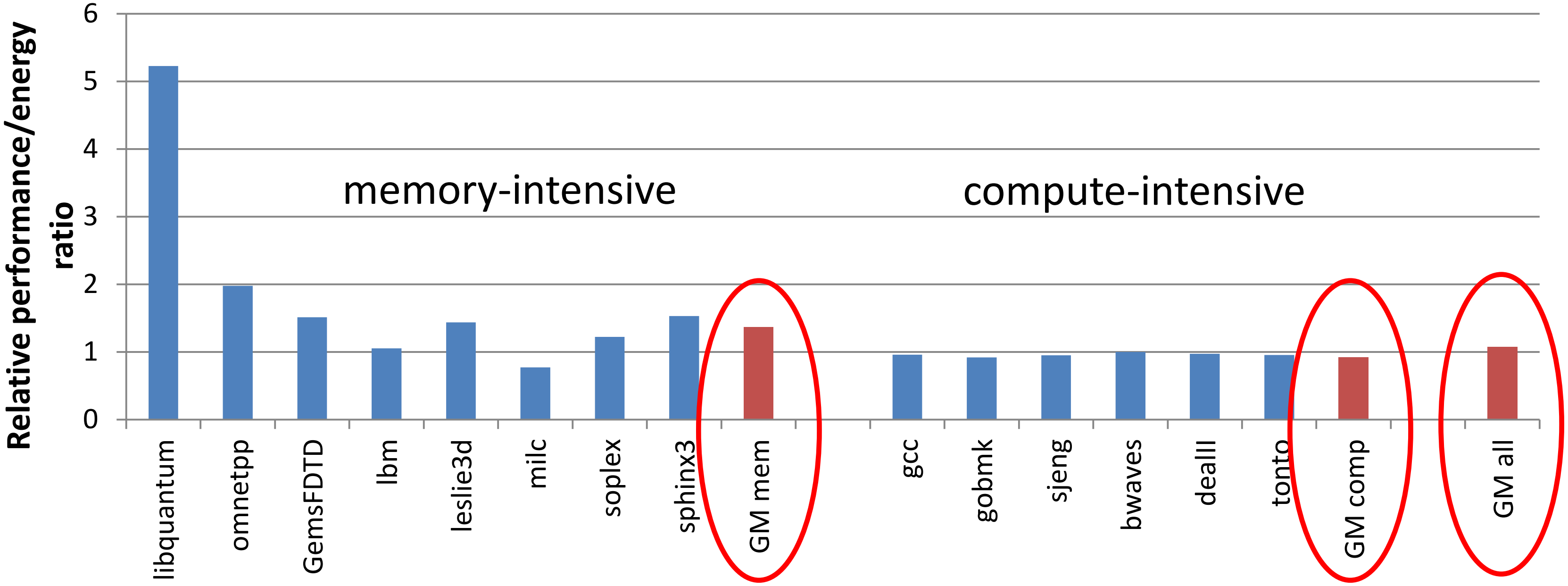




Energy Efficiency: Assumption

- Performance/Energy $\propto$ 1/EDP
- Consumed energy is derived using McPAT
- 32nm LSI technology
- Temperature of 350K

Energy Efficiency: Results



- Power is increased, but perf is improved ⇒ Better energy efficiency
- Memory-intensive: 36% better
- Compute-intensive: 8% worse
- Overall: 8% better

Cost Efficiency: Assumption

- Calculate area using McPAT

- 32nm LSI technology

Cost Efficiency: Results

Additional cost	value (per core)	1.6mm^2
	vs. base core	6%
	vs. Sandy Bridge core	8%
	vs. Sandy Bridge chip	3%
Speedup	achieved	21%
	expected by Pollack's law	3%
	augmented L2 cache	1%

2MB, 4-way → 2.5MB, 5-way
(increased cost is 1.3x greater than the additional cost)

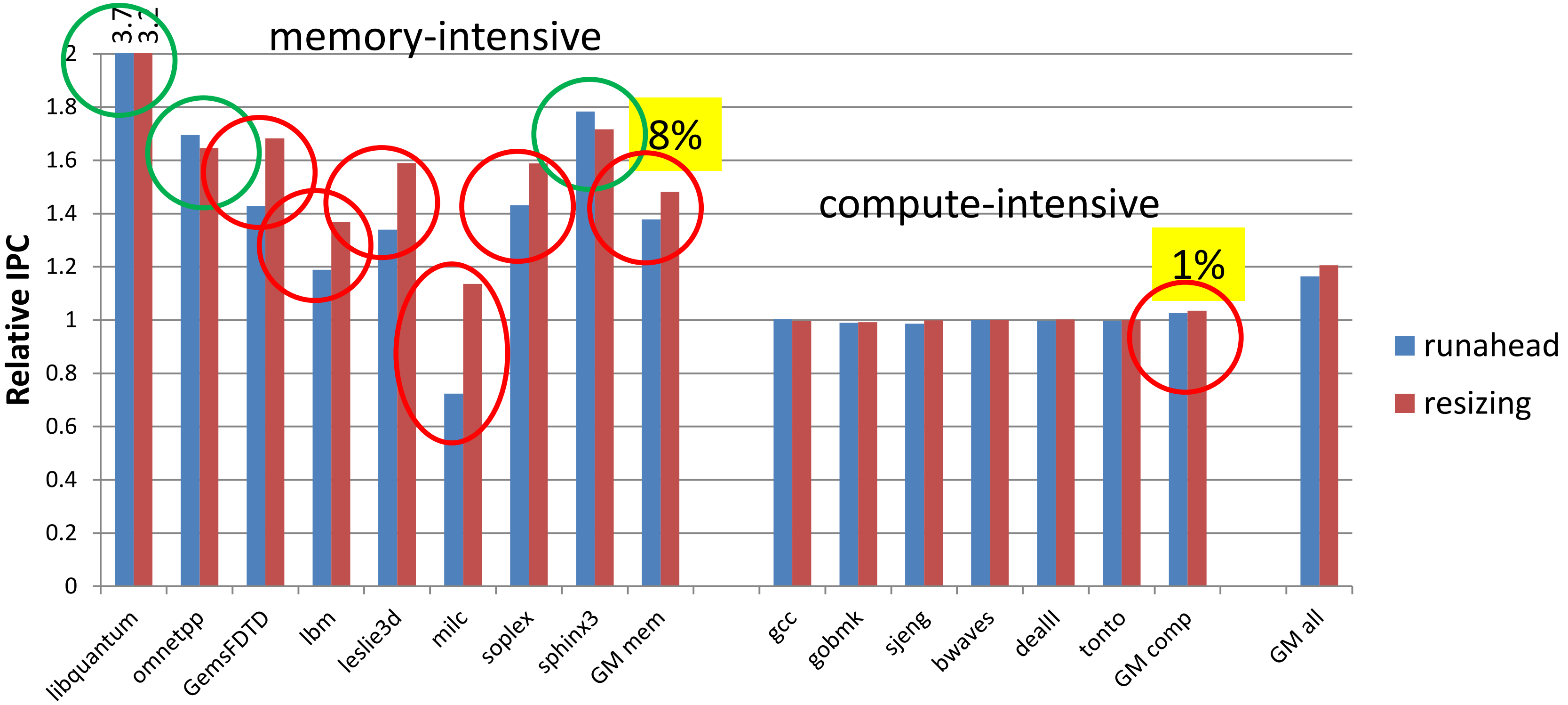
Good cost/performance ratio,
that far exceeds that based on Pollack's law

Background on Runahead Execution

- Features
  - Exploit MLP by pre-execution
  - Can also exploit ILP because it requires only a small instruction window
- When an L2 cache miss occurs
  - Checkpoint architecture state and enter runahead mode
- In runahead mode
  - Instructions are speculatively executed
  - If another L2 miss occurs, MLP is exploited by overlapping main memory access with the triggered load access
- Runahead mode ends when the original L2 miss is resolved, and normal execution restarts from the checkpoint

- Practical
  - Simple
  - Accommodate existing architecture
  - Adopted in SunMicro Rock and IBM Power6

Comparison with RA: Results



- Runahead that has an EXCLUSIVE period for MLP exploitation achieves better performance in very mem-intensive programs.
- Resizing achieves better performance in moderately mem-intensive because it can exploit ILP and MLP SIMULTANEOUSLY.

Conclusion

- Dynamic instruction window resizing
  - Exploit ILP and MLP adaptively
  - Based on prediction of available parallelism
- Features
  - Very simple
  - Very practical
- Our scheme achieves
  - Performance level similar to the best performance achieved with fix-sized resources
  - 21% speedup
  - 6% extra cost of a core, or 3% of an entire proc chip
  - 8% better energy efficiency