The Effects of OoO Execution on The Memory System

Ph.D. Research Proposal

Aamer Jaleel

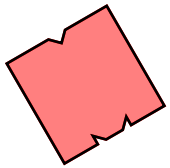
ajaleel@eng.umd.edu

Proposal Committee:

Dr. Rajeev Barua {barua@eng.umd.edu}

Dr. Bruce Jacob {blj@eng.umd.edu}

Dr. Donald Yeung {yeung@eng.umd.edu}



Overview

- **Motivation**
- **The Problem**
  - **Effects on Replay Traps**
  - **Effects on Cache Performance**
- **A New Metric - *Disorder***
  - **Absolute Disorder**
  - **Relative Disorder**
- **Proposed Work**
  - **Sensitivity Studies**
  - **Dynamic Mechanisms**

Out-of-Order Execution

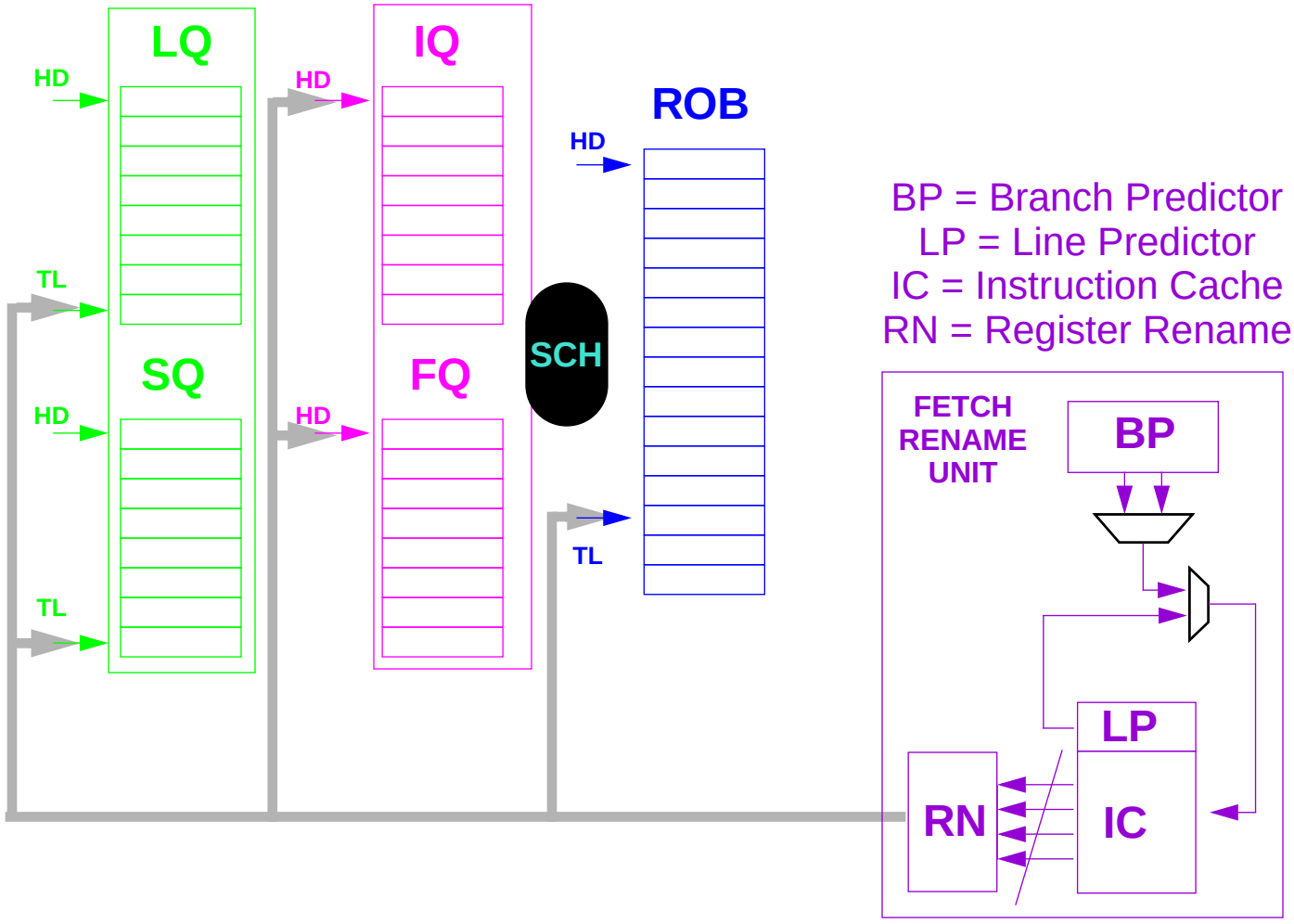
- **Overlap idle time caused by long latency operations with *possible* useful work**
  - **Floating point latency: 10-15 cycles**
  - **Memory latency: 100-2000 cycles**
- **Widely held belief that a processor's OoO efficiency depends on the number of instructions it views at a given time**

Large Instruction Windows / ROBs!!!

More Instruction Level Parallelism!!!!!!!

OoO Hardware Structures

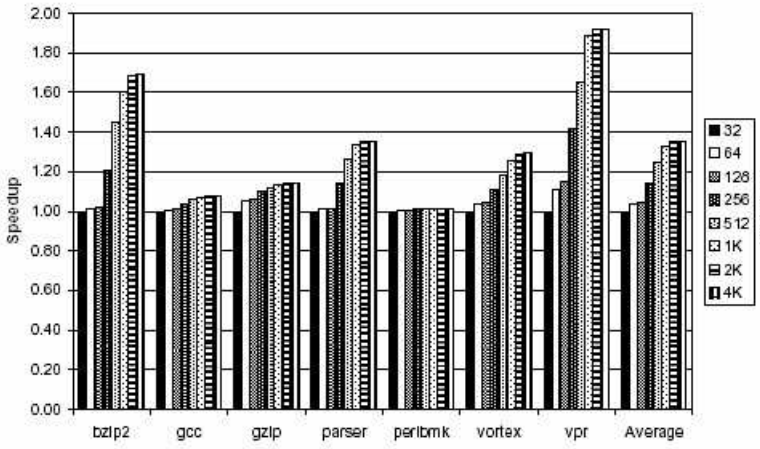
- **ROB, Integer and Floating Point Issue Queues, Load and Store Queues**

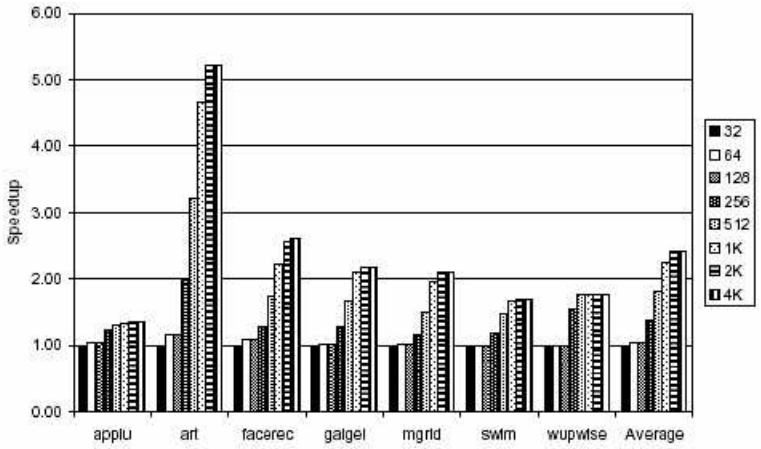


Research Trend in Increasing ILP

- **Proposals to build large ROBs**
  - **Akkary et al., Lebeck et al., Skadron et al.**
- **Circuit techniques to build large ROBs with out affecting clock cycle time**
  - **Brown et al., Henry et al., Onder et al.**
- **Techniques to allow for more load/store queue communications**
  - **Park et al., Akkary et al.**

Large Instruction Windows





- **Lebeck et al. show 35-250% performance improvements with large ROBs**
- **However, when one considers the discounted real effects, most of these performance improvements disappear**
  - **Replay Traps**
  - **Cache Misses**

Replay Traps - Background

- **Replay traps are required to**
  - **Force accesses to a particular memory location in order**
  - **Handle different-sized accesses to the same memory location**
- **Two recovery schemes**
  - **Flush pipeline, and then re-fetch and re-execute all instructions from the replay trap causing instruction**
  - **No flush, re-execute ONLY replay trap causing instruction and all other instructions that directly or indirectly depend on it**

Types of Replay Traps

- **Load-Store Replay**

1. **LD BYTE A (1)**
2. **ST BYTE A (3)**
3. **LD BYTE A (2)**
4. **LD BYTE B (4)**

(a) Load-Store Replay

Types of Replay Traps

- **Load-Store Replay**
- **Wrong Size Replay**

1. **LD BYTE A (1)**
2. **ST BYTE A (3)**
3. **LD BYTE A (2)**
4. **LD BYTE B (4)**

(a) Load-Store Replay

1. **LD BYTE A (1)**
2. **ST BYTE A (2)**
3. **LD HALF A (3)**
4. **LD BYTE B (4)**

(b) Wrong Size Replay

Types of Replay Traps

- **Load-Store Replay**
- **Wrong Size Replay**
- **Multi-Processor**
  - **Load-Load Replay**

1. **LD BYTE A (1)**
2. **ST BYTE A (3)**
3. **LD BYTE A (2)**
4. **LD BYTE B (4)**

(a) Load-Store Replay

1. **LD BYTE A (1)**
2. **ST BYTE A (2)**
3. **LD HALF A (3)**
4. **LD BYTE B (4)**

(b) Wrong Size Replay

P1	P2
1. **LD BYTE A (4)**	*3.* **LD BYTE A (1)**
2. **ST BYTE A (2)**	*4.* **LD BYTE B (3)**

(c) Load-Load Replay

Types of Replay Traps

- **Load-Store Replay**
- **Wrong Size Replay**
- **Multi-Processor**
  - **Load-Load Replay**
  - **Load-Miss Load Replay**

1. LD BYTE A (1)
2. ST BYTE A (3)
3. LD BYTE A (2)
4. LD BYTE B (4)

(a) Load-Store Replay

1. LD BYTE A (1)
2. ST BYTE A (2)
3. LD HALF A (3)
4. LD BYTE B (4)

(b) Wrong Size Replay

P1	P2
1. LD BYTE A (4)	*3.* LD BYTE A (1)
2. ST BYTE A (2)	*4.* LD BYTE B (3)

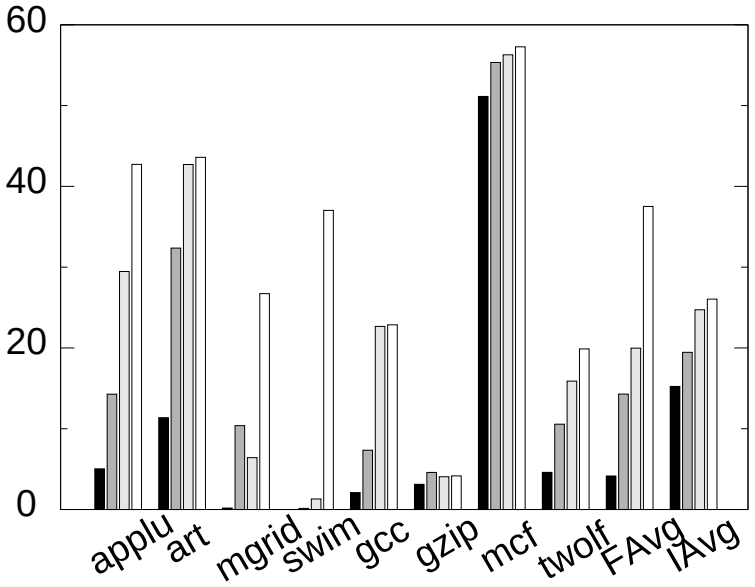
(c) Load-Load Replay

P1	P2
1. +LD BYTE A (1)	*2.* ST BYTE A (2)
3. LD BYTE A (3)	*4.* LD BYTE B (4)

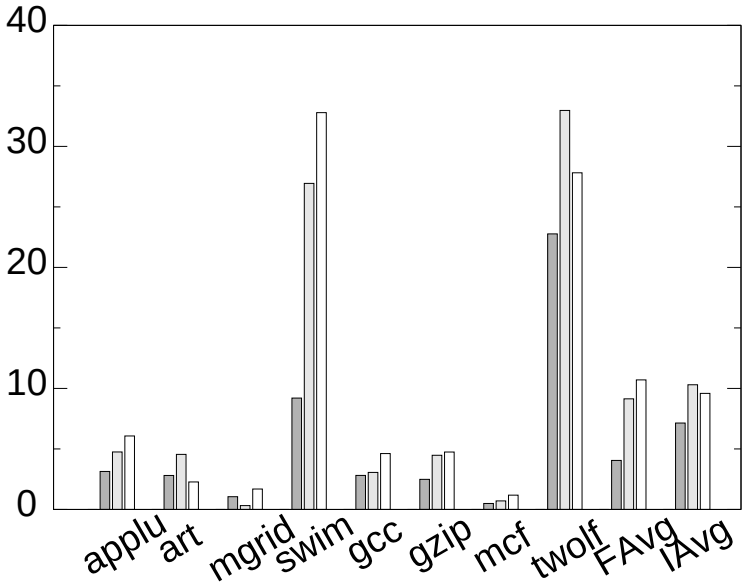
(d) Load-Miss Load Replay

The Problem

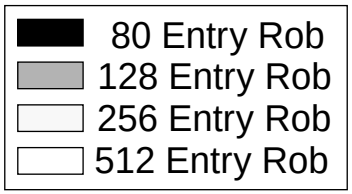
- **Aggressive OoO techniques result in an**
  - **Increase in frequency of *replay traps***
  - **Increase in number of L1 *cache misses***



% Total Execution Time Spent in Handling Traps



% Increase in L1 Cache Misses Compared to 80-Entry ROB



Hypothesis

- **Increasing ROB sizes reorder ALU and memory instructions**
- **Re-ordering of ALU instructions poses little or no threats, BUT re-ordering of memory instructions causes most of the negative effects**
- **3 Processor Configurations**
  - **ALU-in / MEM-in: ALU and memory instructions issued in-order (*sequential*)**
  - **ALU-out / MEM-in: ALU instructions issued out-of-order, memory instructions issued in-order (*more speculation than ALU-in/MEM-in*)**
  - **ALU-out / MEM-out: Both ALU and memory instructions are issued OoO (*superscalar*)**

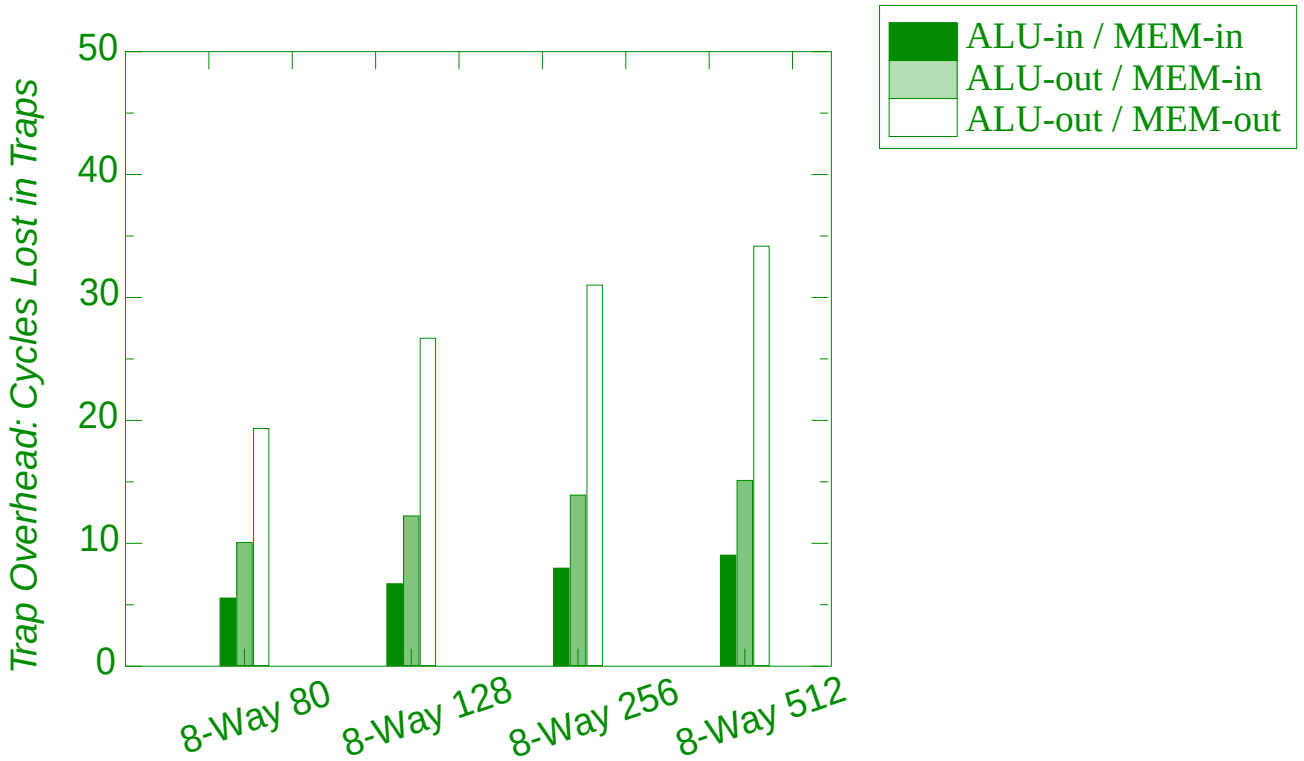
Experimental Framework

Configuration Name	ROB Size	Issue Width	IssueQ Size INT/FP	# Functional Units**	LQ/SQ Size
Alpha-80	80	2-32 Way	20/15	(4/4/1/1) x1	32/32
Alpha-128	128	2-32 Way	40/30	(4/4/1/1) x2	64/64
Alpha-256	256	2-32 Way	80/60	(4/4/1/1) x4	128/128
Alpha-512	512	2-32 Way	160/120	(4/4/1/1) x8	256/256

**INT ALU/INT MULT/FP ALU/FP MULT

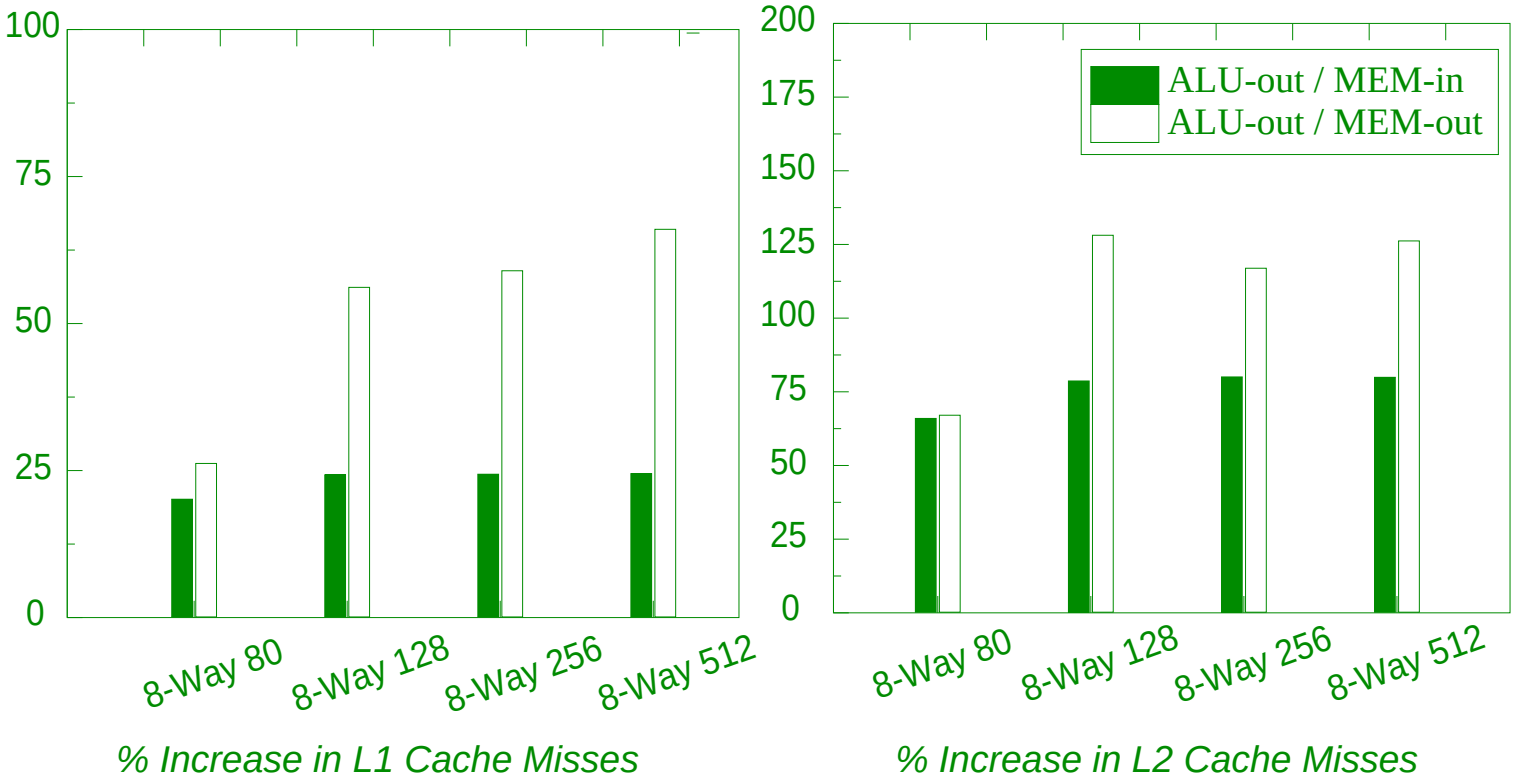
- **Sim-Alpha**
  - **Validated Execution Driven Simulator**
  - **Detailed DRAM Simulator**
  - **64K 2-way IL1/DL1 and 2MB 4-way Unified L2**
  - **8 MSHRS per cache**
  - **1024-entry store-wait data structure**
  - **Supports No/Sequential/Stride Data Prefetch**
- **Benchmarks**
  - **SPEC2000 & Olden**

OoO And Replay Trap Overhead



- **ALU-in/MEM-in --> ALU-out/MEM-out**
  - **Factor 8-15 increase in trap frequency**
  - **15-30% increase in trap overhead**

OoO And Cache Misses



% Increase in L1 Cache Misses

% Increase in L2 Cache Misses

Normalized to # L1 Cache Misses of ALU-in/MEM-in

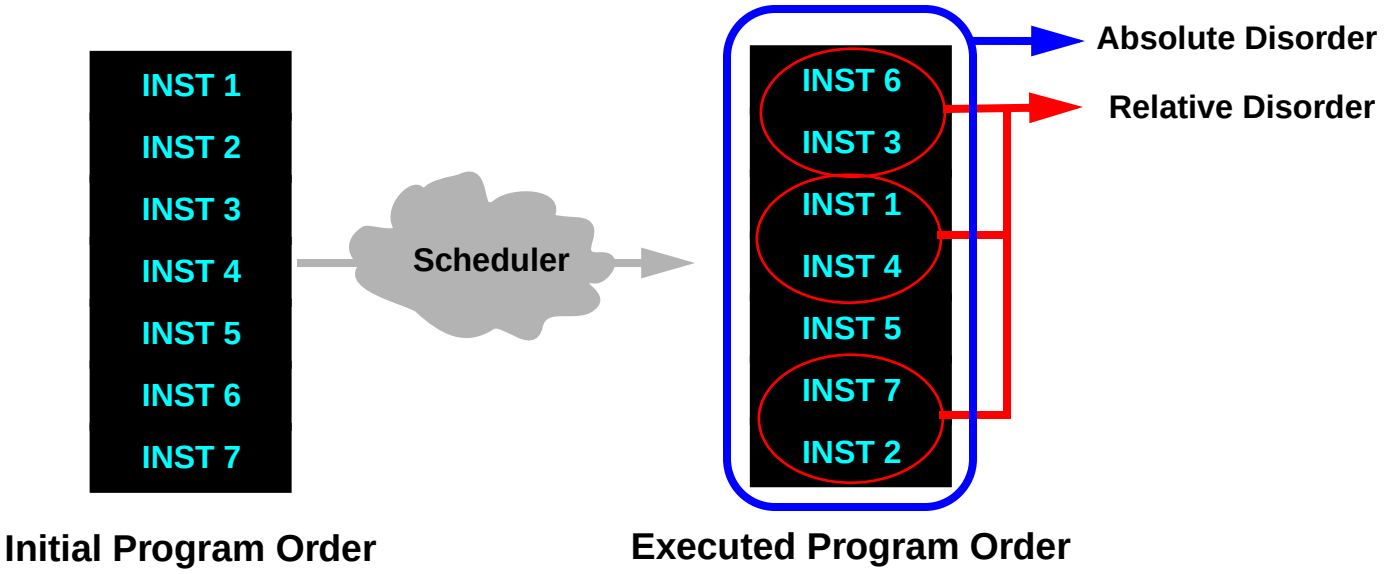
- **ALU-in/MEM-in --> ALU-out/MEM-out**
  - **25-75% increase in L1 Cache Misses**
  - **80-125% increase in L2 Cache Misses**

What Have We Learned?

- **In general, increasing a processor's OoO capability conflicts with the memory ordering requirements and cache performance**
- **Limiting the memory instructions to be issued in program order aids in reducing these negative effects**
  - **Downside of MEM in-order: memory ILP hurt**
- **Future aggressive OoO processors need a mechanism to throttle the degree by which they issue memory instructions out-of-order**

Introducing A New Metric

- *Disorder* — The degree by which an instruction is issued out-of-order
- Two types of disorder
  - Absolute disorder — On a program order perspective
  - Relative disorder — On a per instruction executed perspective



Disorder Measurements

- We measure disorder ONLY for memory instructions
- When a memory instruction is fetched, it is assigned a sequential ID, i.e. the first memory instruction fetched gets sequential ID 1, next gets sequential ID 2, and so on.
- In the event of a pipeline flush, the sequential ID is restored to the last successfully retired sequential ID + 1
- Disorder is computed after a memory instruction has its dependencies resolved and is issued to execute

Absolute Disorder

- **Degree by which an instruction is issued OoO compared to fetch order**
- **Computed by finding the difference between the instruction issued and the instruction that *should've been issued* were the core in-order**

PROGRAM ORDER	*ISSUE ORDER*	*ABSOLUTE DISORDER*
MEM_1	MEM_1	0
MEM_2	MEM_3	1
MEM_3	MEM_5	2
MEM_4	MEM_7	3
MEM_5	MEM_8	3
MEM_6	MEM_2	- 4
MEM_7	MEM_{10}	3
MEM_8	MEM_4	- 4
MEM_9	MEM_9	0
MEM_{10}	MEM_{11}	1
MEM_{11}	MEM_6	- 5

4 - Way Issue Order

Cycle 101: 1, 3
Cycle 105: 5, 7, 8
Cycle 126: 2, 10
Cycle 139: 4
Cycle 213: 9, 11
Cycle 224: 6

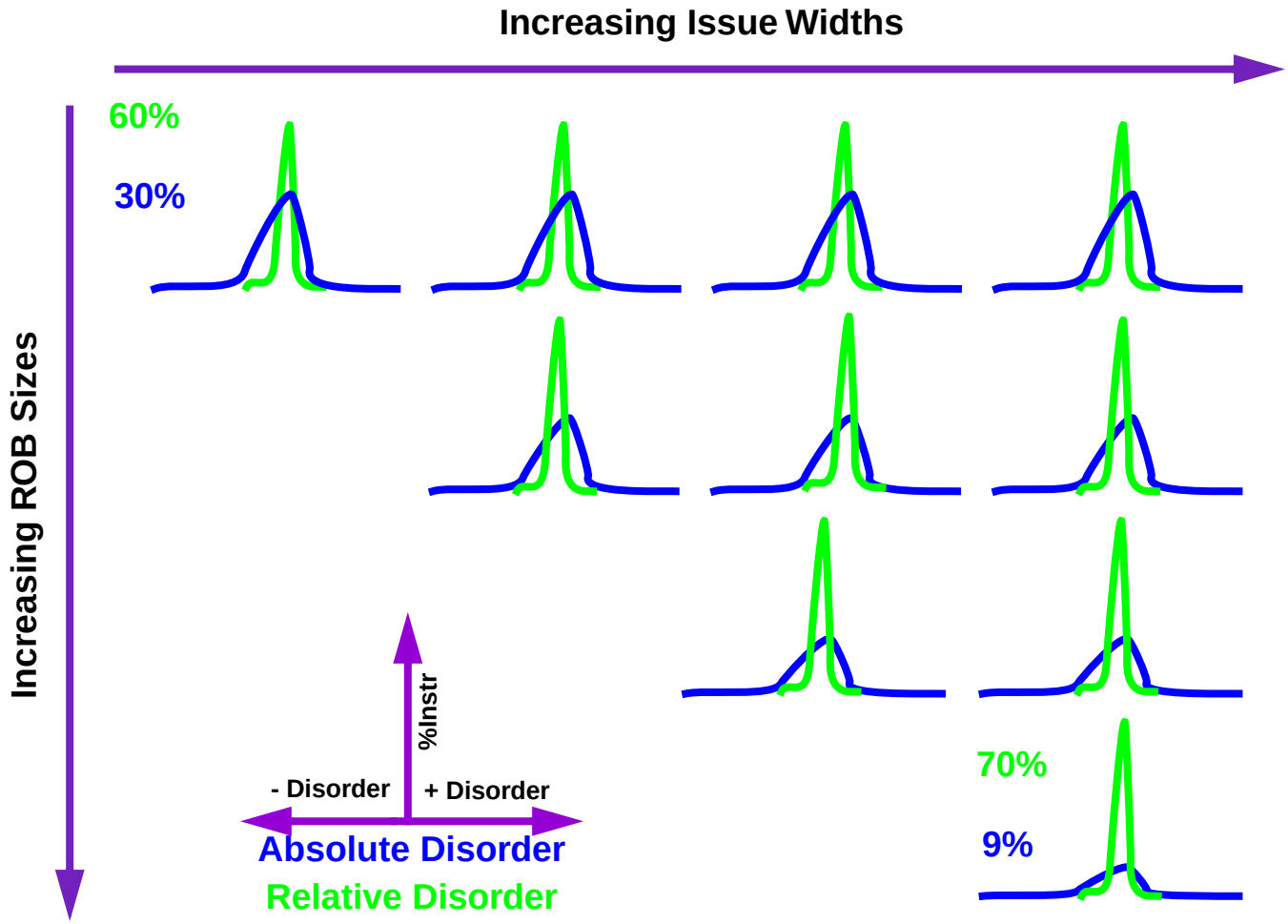
Relative Disorder

- **Degree by which an instruction is issued OoO compared to other instructions**
- **Computed by finding the minimum absolute disorder between an instruction and all other instructions issued in the current and a previous cycle**

	4 - Way Issue Order	*Relative Disorder of MEM10*	*RELATIVE DISORDER*
MEM_1			1
MEM_2			-3
MEM_3			2
MEM_4			2
MEM_5	Cycle 101: 1, 3		2
MEM_6	Cycle 105: 5, 7, 8		3
MEM_7	Cycle 126: 2, 10		-1
MEM_8	Cycle 139: 4		1
MEM_9	Cycle 213: 9, 11		-2
MEM_{10}	Cycle 224: 6	10 - 2 = 8 10 - 5 = 5 10 - 7 = 3 10 - 8 = 2 Minimum = 2	2
MEM_{11}			2

Disorder Trends

- **Vary Issue Widths from 4 to 32 wide**
- **Vary ROB Sizes from 80 to 512 entries**



Disorder Results

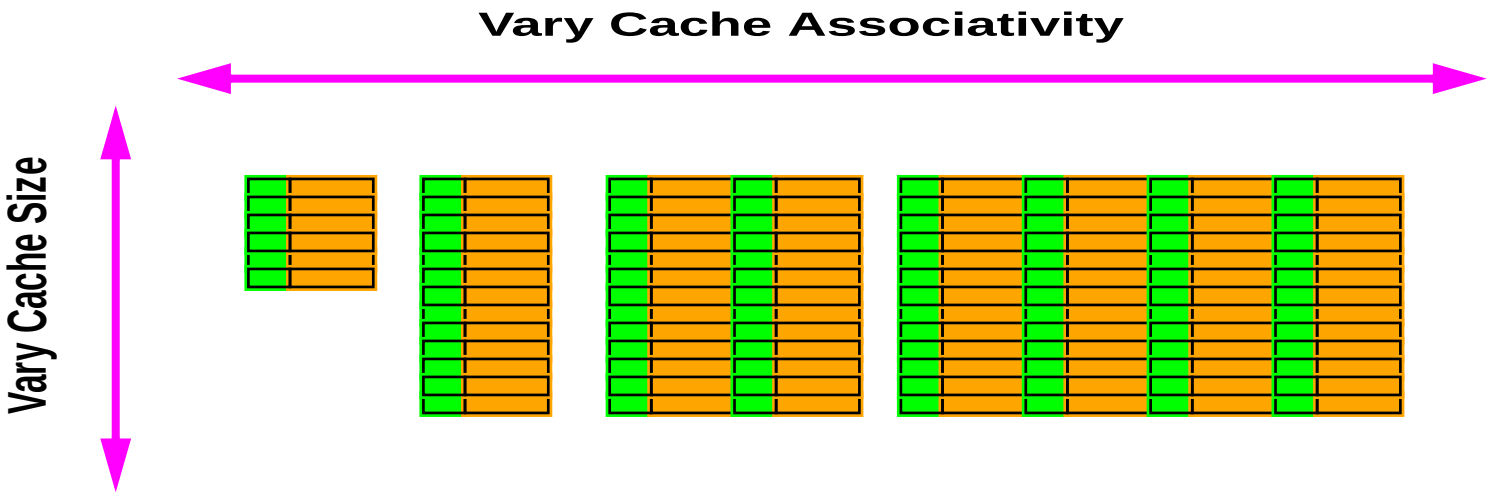
- **Low disorder is due to dependency stalls and L1 miss penalty. High disorder is due to L2 miss penalty**
- **Absolute Disorder**
  - **Increasing ROBs and not issue widths cause an increase in absolute disorder**
  - **< 1/3rd of TOTAL memory instructions executed are issued in actual program order**
- **Relative Disorder**
  - **60 – 70% of memory instructions issued are in close proximity with each other in the instruction stream**
  - **Spatial issue among memory instructions, i.e. “when a processor speculates it continues to speculate”**

Proposed Work

- **Sensitivity Studies**
  - **Cache Organization Parameters**
  - **Load/Store Queue Studies**
- **Dynamic Mechanisms**
  - **Windowing of Load/Store Queue**
  - **?????**

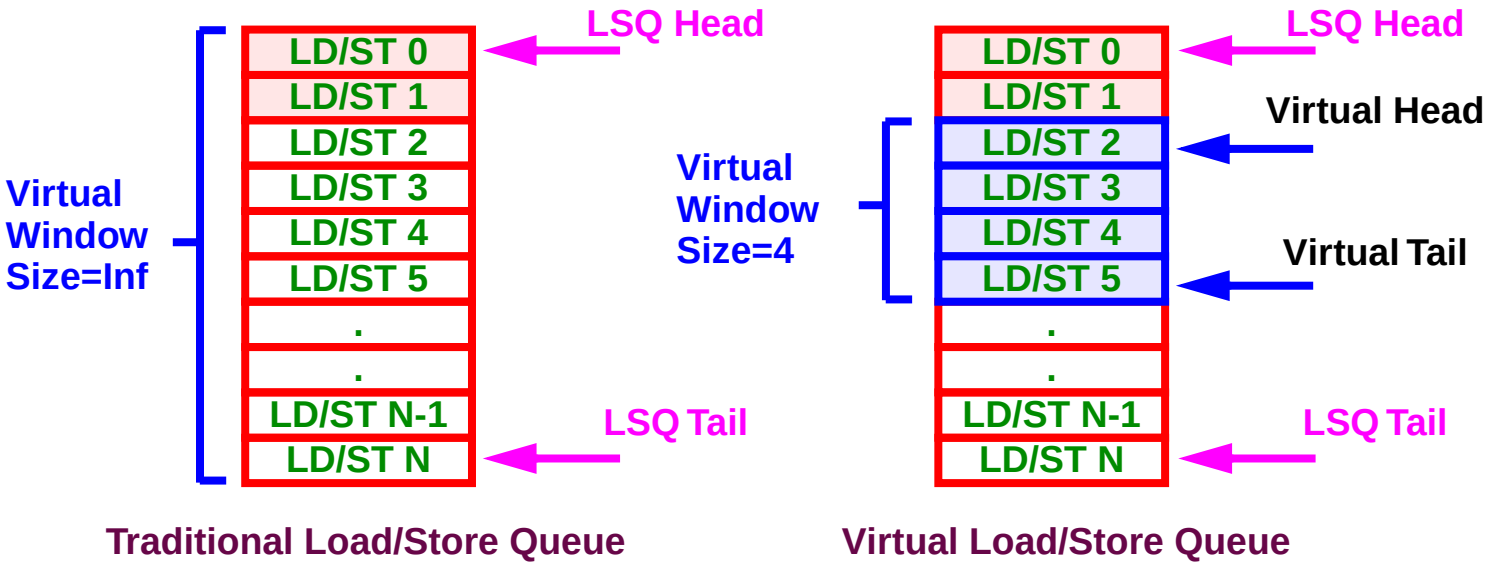
Disorder Vs. Cache Organization

- **Cache Parameters**
  - **Cache Size**
  - **Cache Line Size**
  - **Cache Associativity**



Disorder Vs. Load/Store Queue

- Introduce a virtual window into the load/ store queue
- Only instructions residing within the virtual window may be issued
- Others wait until the virtual window slides onto them



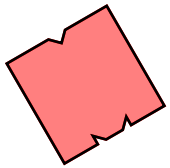
Windowing of Load/Store Queue

- **Static Mechanism (Sensitivity Study):**
  - **Statically set the size of the virtual window based on profile information of the application in concern**
  - **Drawback: Memory ILP is lost during periods of application execution where negative effects do not exist**
- **Dynamic Mechanism:**
  - **Dynamically vary the size of the virtual window based on application behavior during execution**
  - **Virtual window initially starts at infinity but as certain thresholds are reached the virtual window size is reduced**

Conclusions

- **Though the well known mechanism of increasing ILP improves performance, it can cause side effects in the memory system**
- **Characterized the problem in terms of**
  - **Increased Replay Traps**
  - **Increased Cache Misses**
- **Source of problem is the reordering of memory instructions**
- **Proposing to study mechanisms to throttle the degree by which memory instructions are issued OoO**

Questions?????



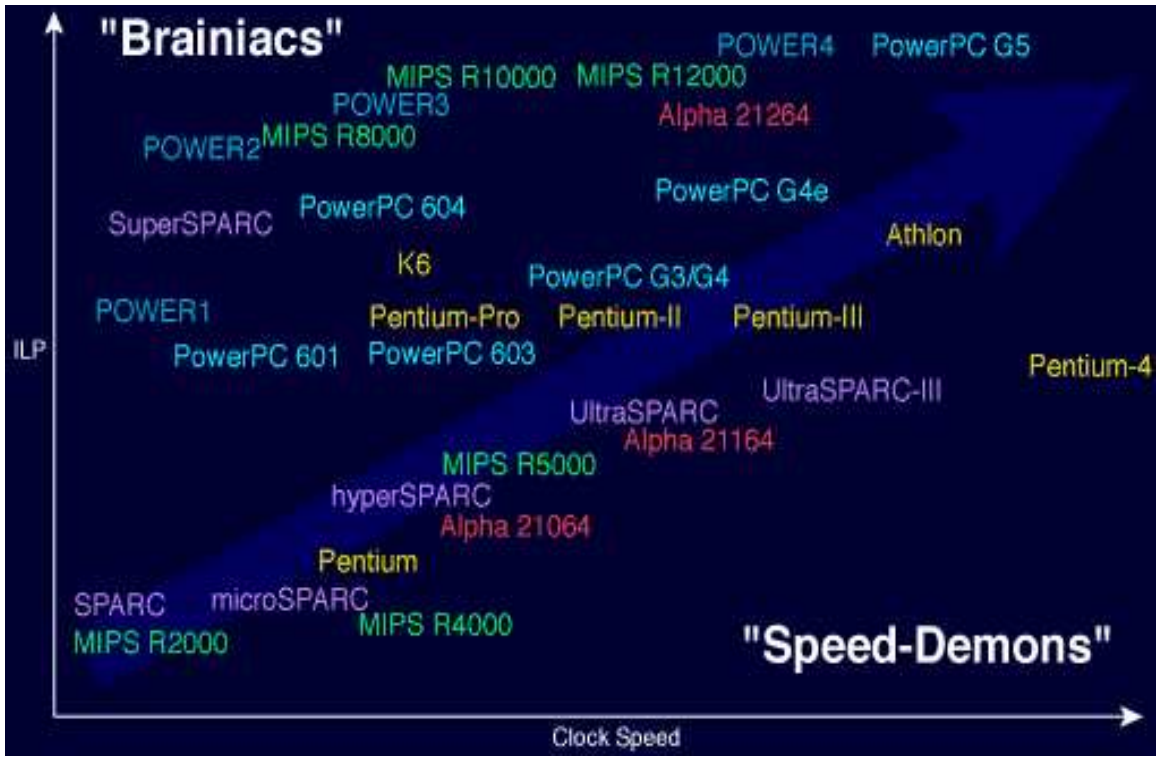
Improving Performance

Execution Time = Cycle time * CPI * Inst Cnt

- **Instruction Level Parallelism (ILP)**
  - **Pipelining**
  - **Multiple Issue Width**
  - **Out-of-Order Execution**
- **Speculation**
  - **Cache Line Prediction**
  - **Branch Prediction**
- **Prefetching**
  - **Hardware Prefetching**
  - **Software Prefetching**

Industry Trends

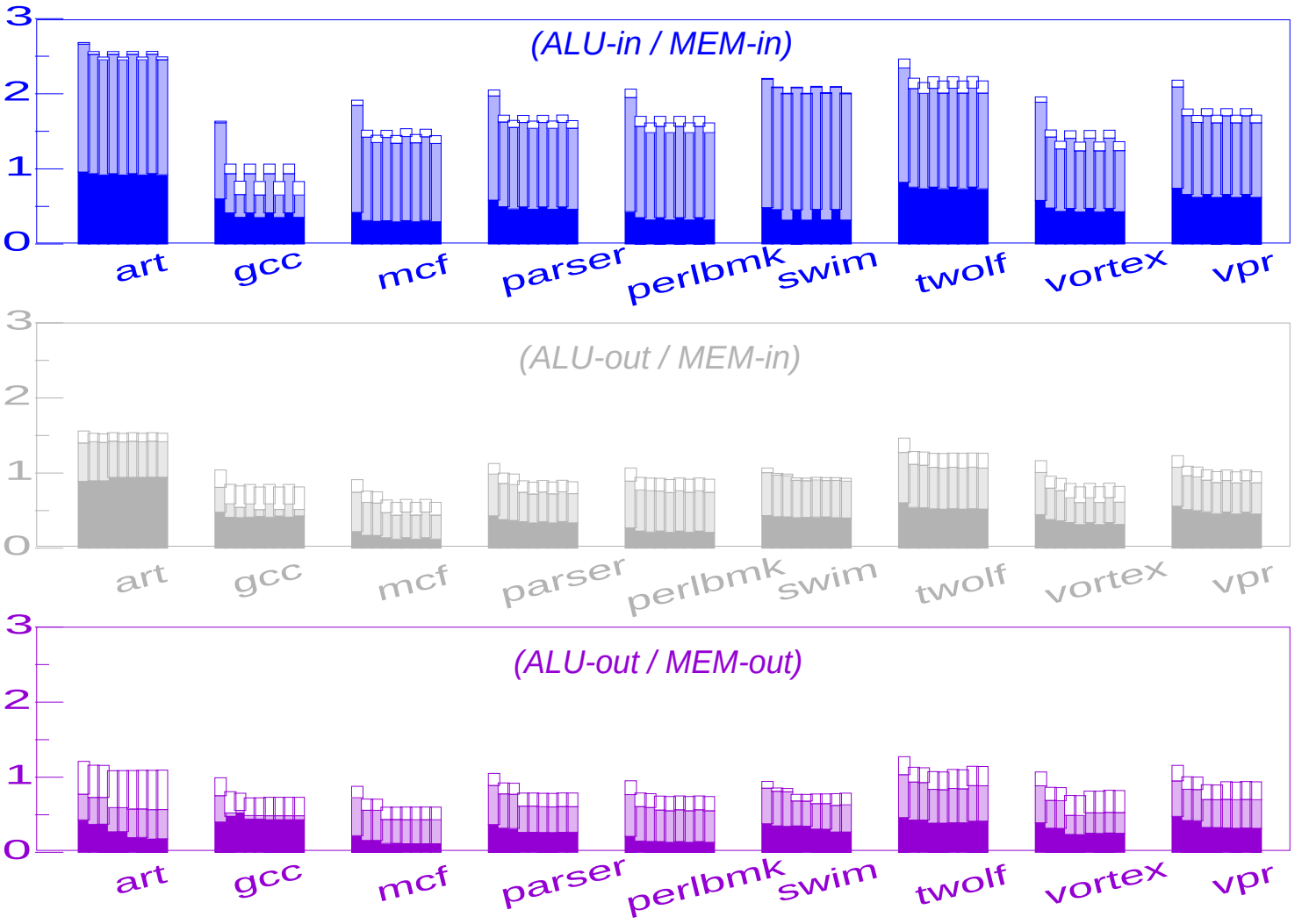
- **Two Design Philosophies**
  - ***Brainiacs*: Improve microprocessor performance by increasing ILP**
  - ***Speed Demons*: Improve microprocessor performance by increasing clock speeds**



Why Use a Virtual Window?

- **Provide a mechanism to allow for large ROBs to exploit ALU instruction ILP yet provide the benefits of smaller ROBs**
- **Since memory instructions held in load/ store queues (LSQ), could have large ROB sizes and small LSQs.**
  - **Effective in limiting number of memory instructions in flight, hence reduces disorder**
  - **Inefficient design methodology as it can under utilize the ROB space**
- **Allow for large ROB and LSQ sizes but create a *virtual window* in LSQ**
  - **Statically or dynamically vary the size of virtual window**

Performance Graphs



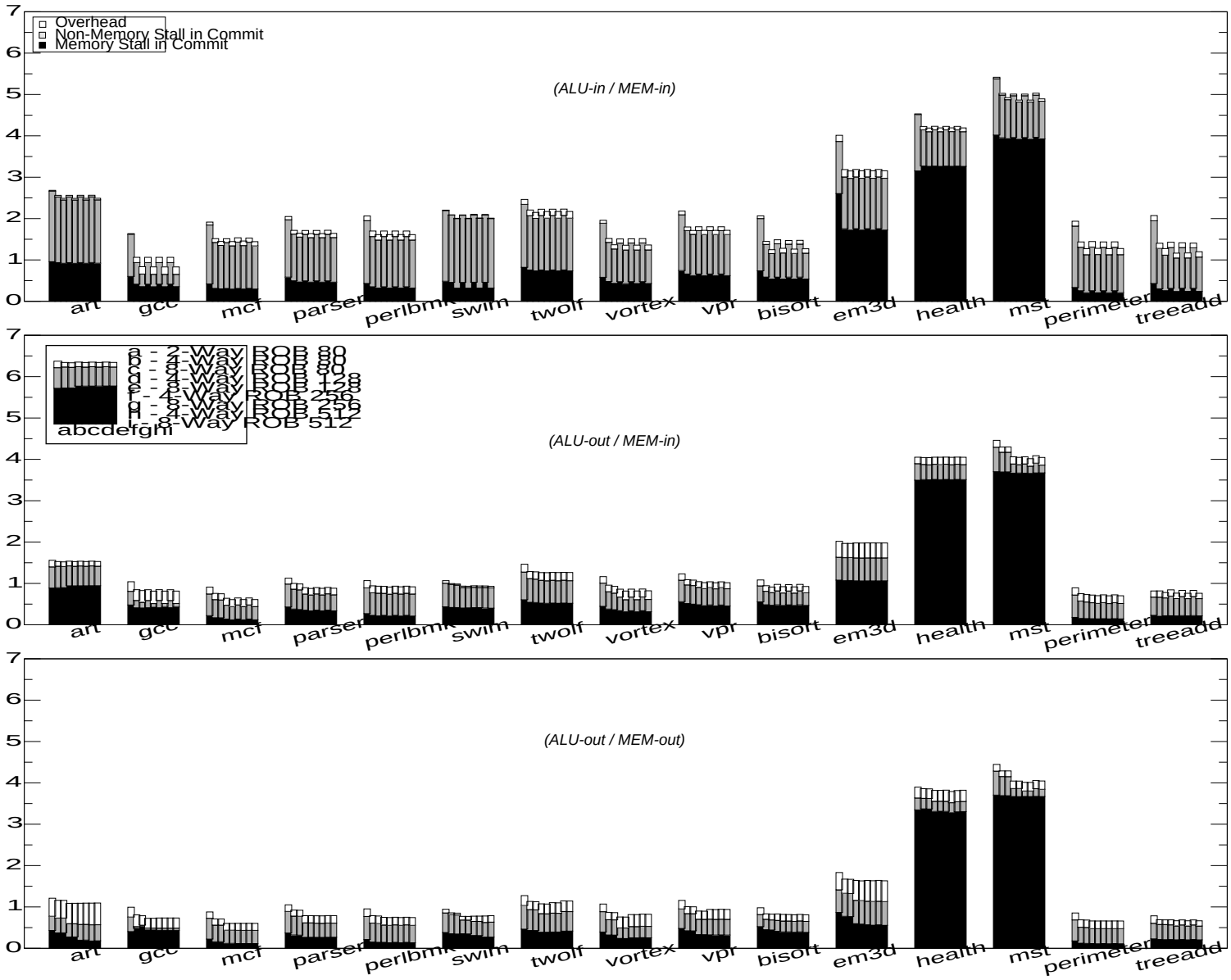
PhD Research Proposal

The Effects Of OoO Execution On The Memory System

Aamer Jaleel

University of Maryland
ECE Dept.

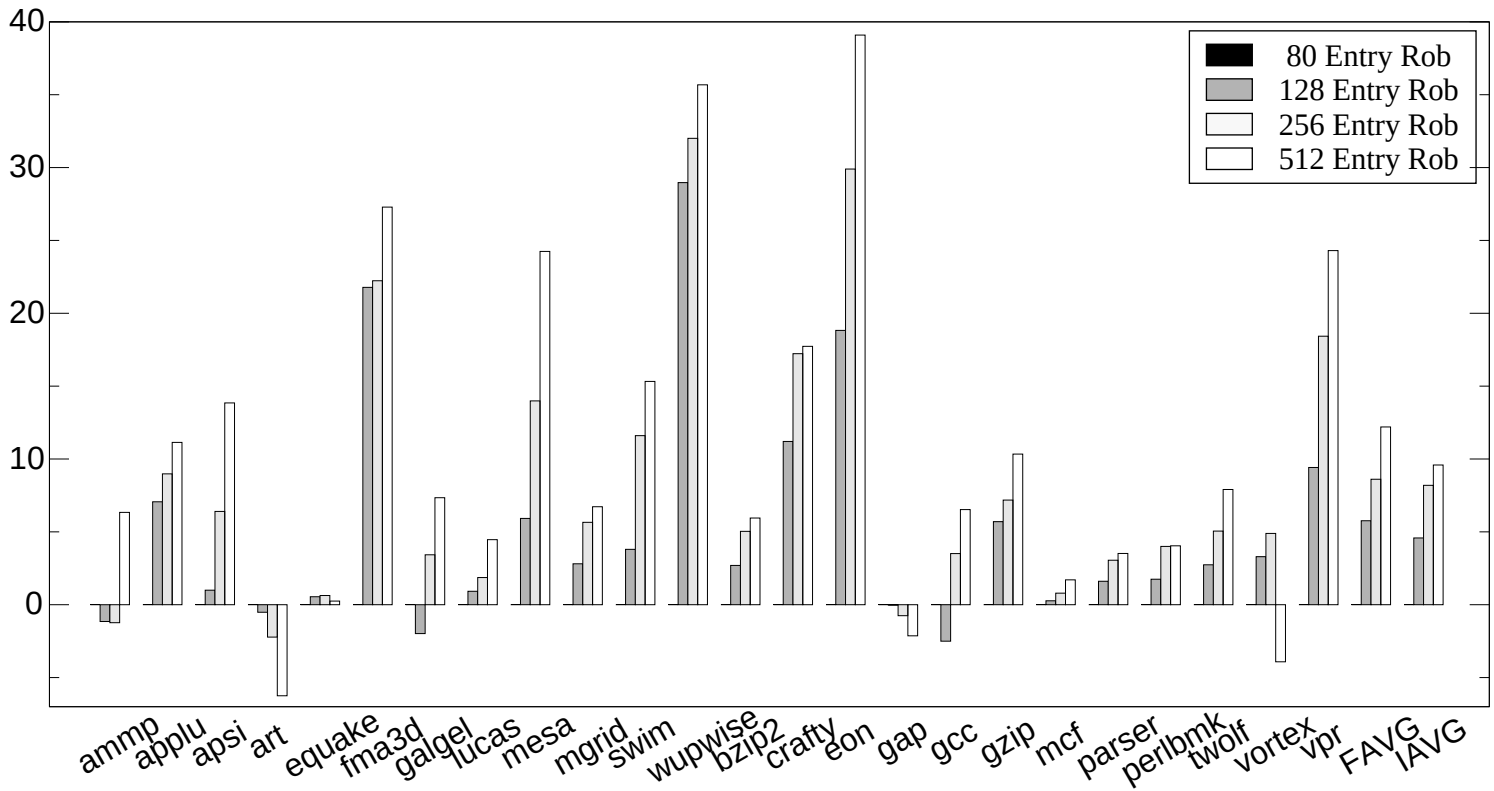
Performance Graphs



UNIVERSITY OF MARYLAND

The Problem - Cache Misses

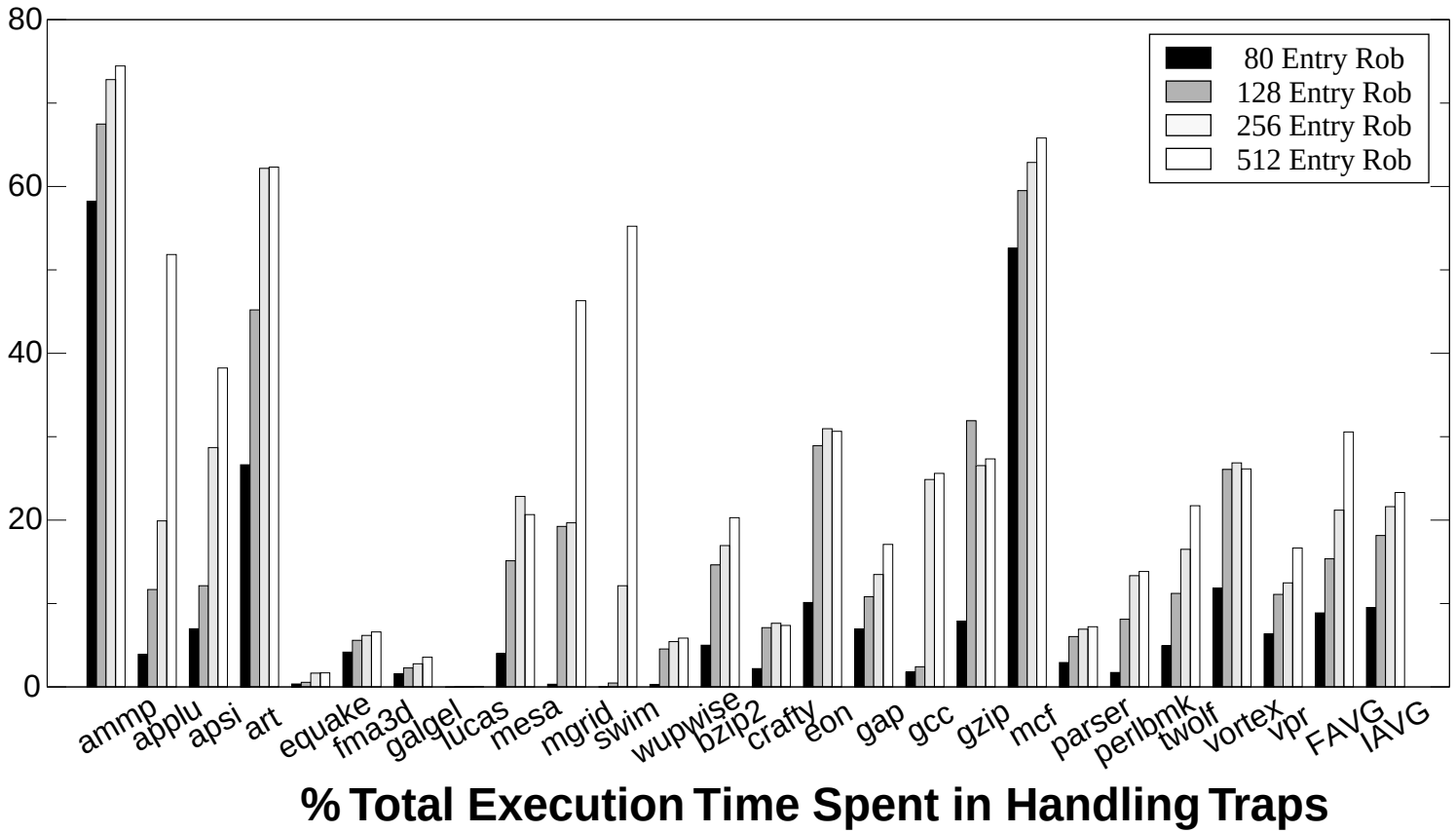
- **Aggressive OoO techniques result in an increase in application *cache misses***

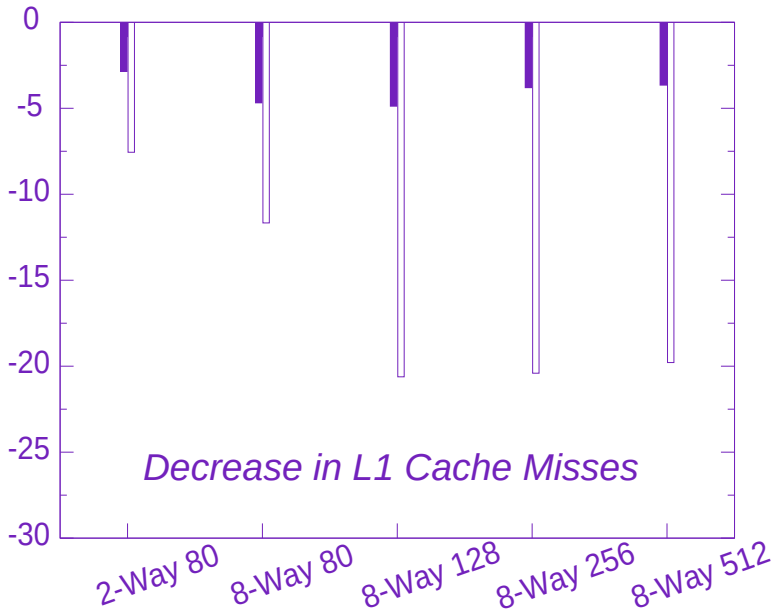


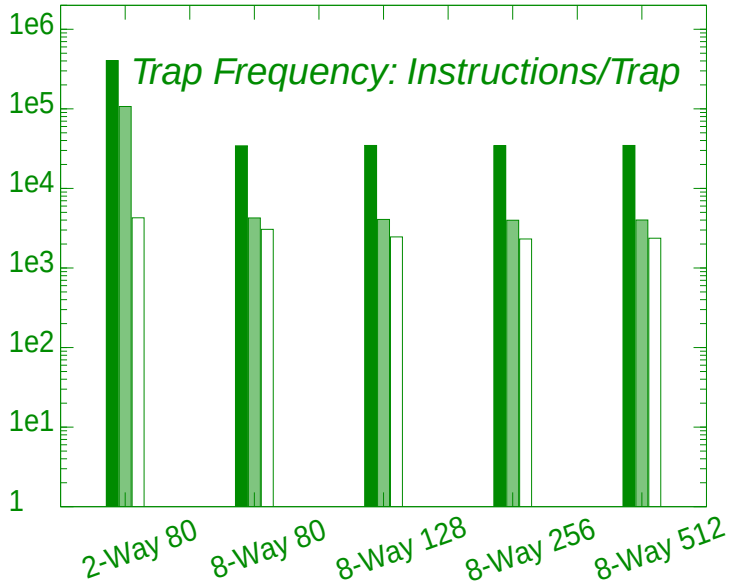
% Increase in L1 Cache Misses Compared to 80-Entry ROB

The Problem - Replay Traps

- **Aggressive OoO techniques result in an increase in *replay traps***
- **Aggravate with increasing ROB sizes.**




0
-5
-10
-15
-20
-25
-30
Decrease in L1 Cache Misses
2-Way 80
8-Way 80
8-Way 128
8-Way 256
8-Way 512


1e6
1e5
1e4
1e3
1e2
1e1
1
Trap Frequency: Instructions/Trap
2-Way 80
8-Way 80
8-Way 128
8-Way 256
8-Way 512