

# Evaluating and Mitigating Bandwidth Bottlenecks Across the Memory Hierarchy in GPUs

Saumay Dublsh, Vijay Nagarajan, Nigel Topham

The University of Edinburgh

ISPASS 2017

25<sup>th</sup> April

Santa Rosa, California

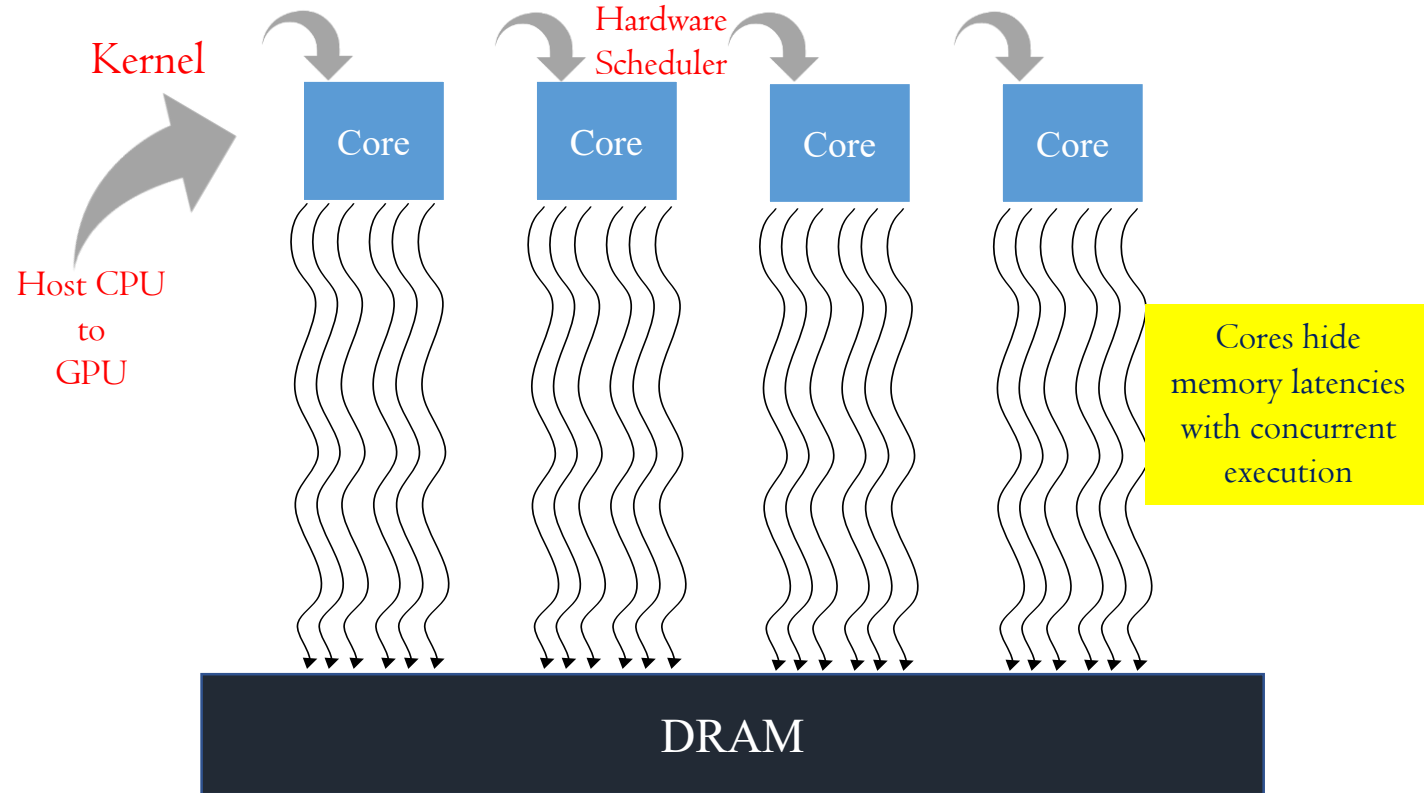


Institute for Computing  
Systems Architecture

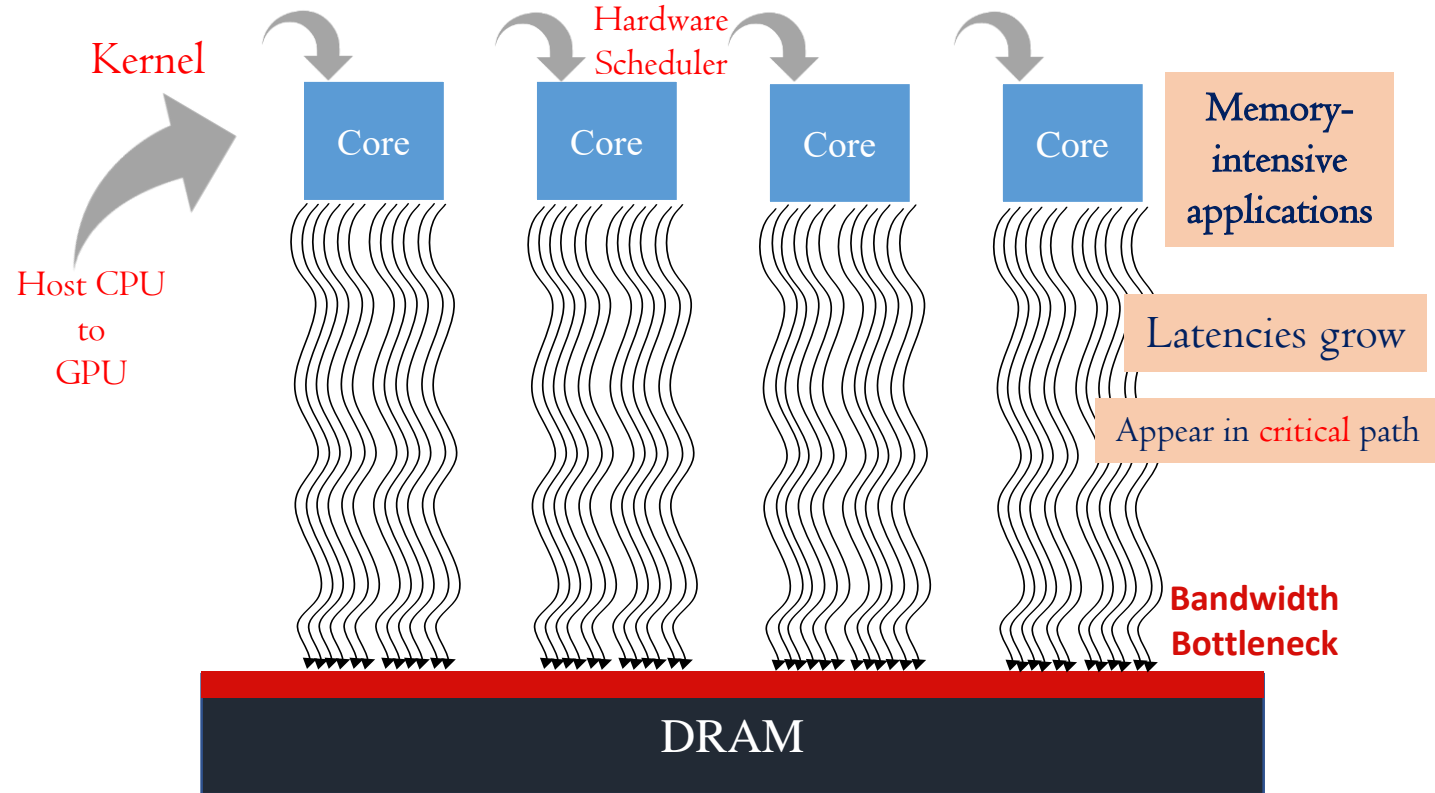


THE UNIVERSITY  
*of* EDINBURGH

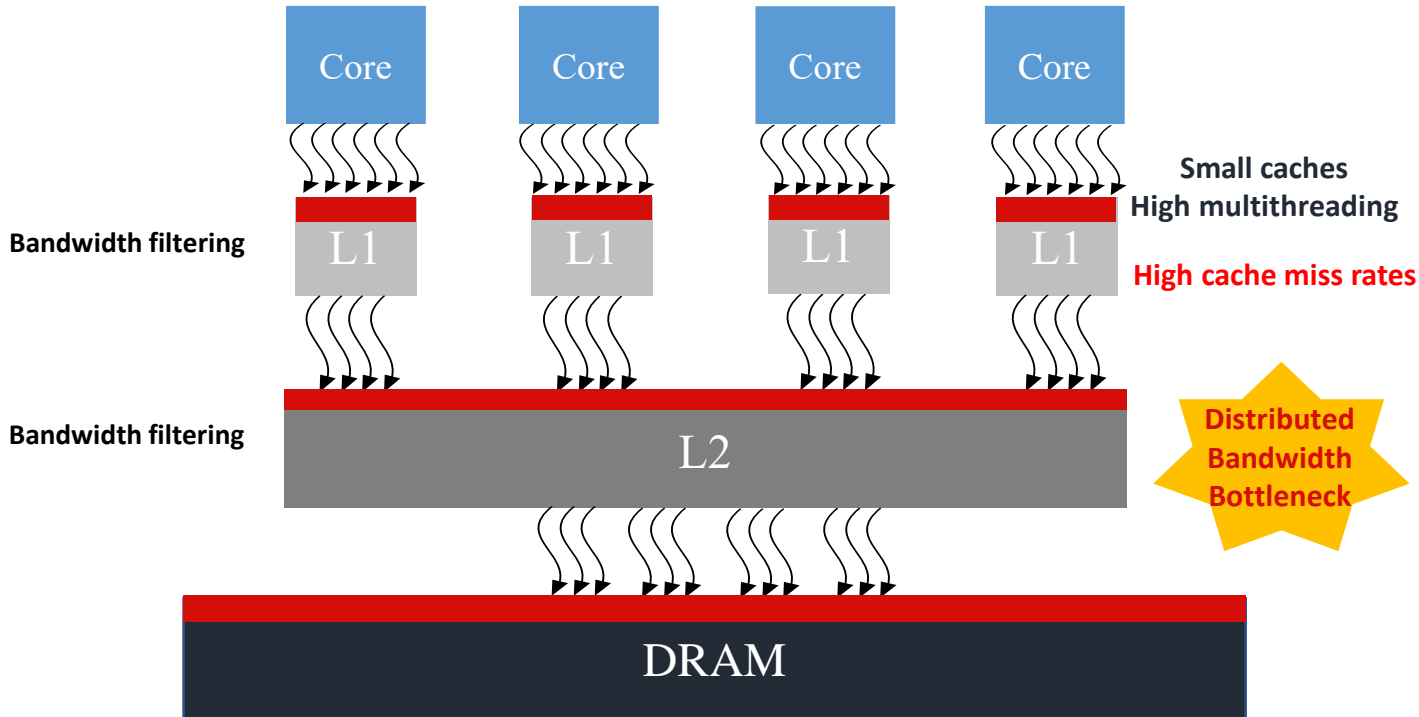
# Multithreading on GPUs



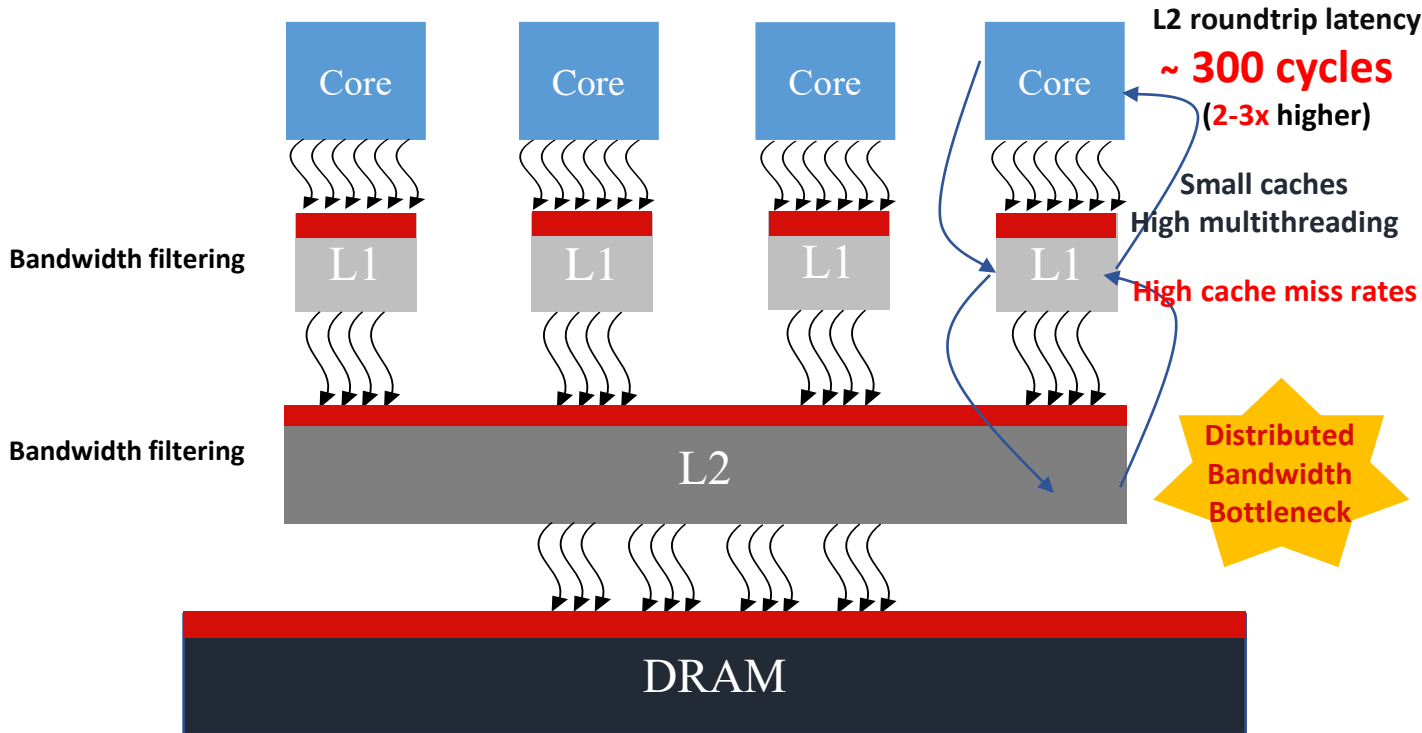
# Multithreading on GPUs



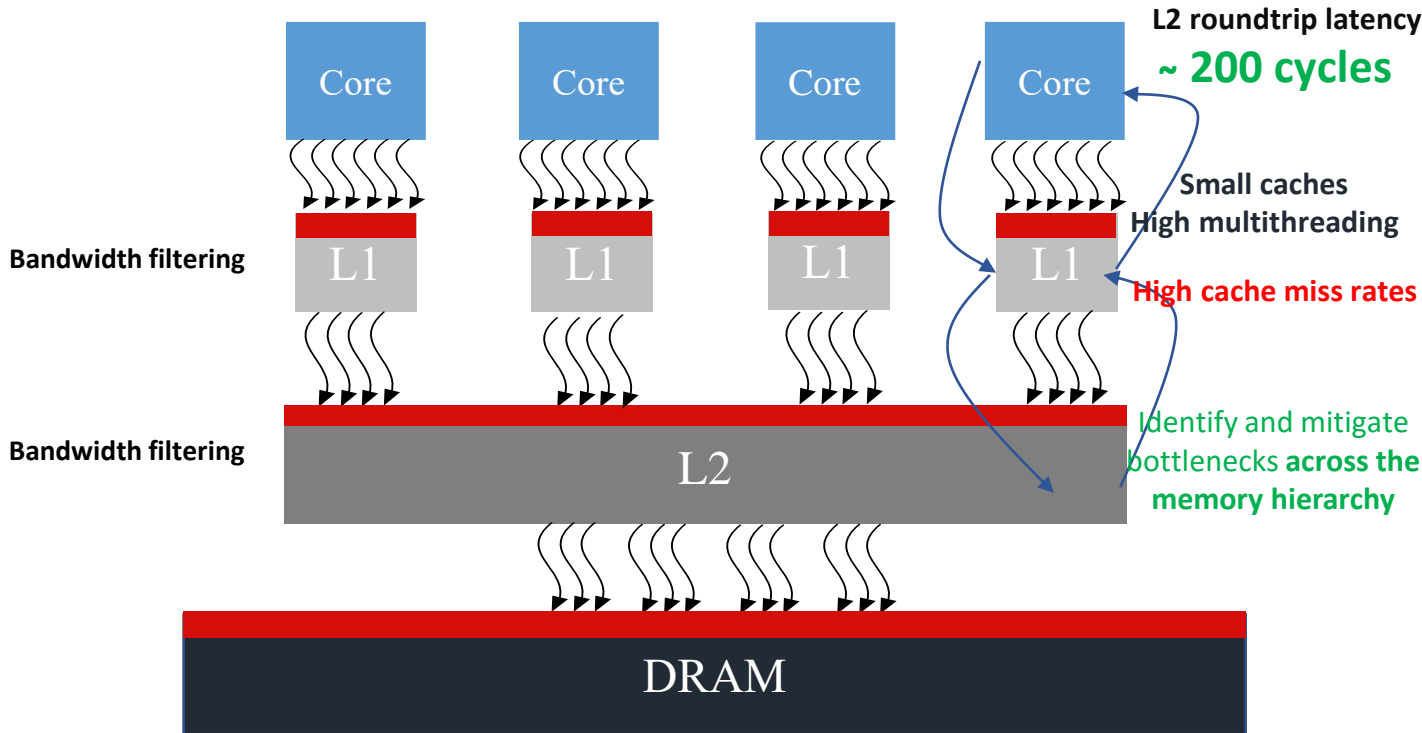
# Deeper Memory Hierarchy



# Deeper Memory Hierarchy



# Deeper Memory Hierarchy



# Goals

---

- Characterize : Understand the bandwidth bottlenecks across different levels of the memory hierarchy such as L1, L2 and DRAM
- Cause : Investigate the architectural causes for congestion
- Effect : Design-space exploration to evaluate the effect of mitigating congestion
- Proposal: Use **cause** and **effect** analysis to present **cost-effective configurations** of the memory hierarchy

# Experimental Environment

---

- **Platform**
  - GPGPU-Sim (v3.2.2)
  - GPUWattch (McPAT)
- **Benchmark Suites**
  - Rodinia
  - Parboil
  - MapReduce

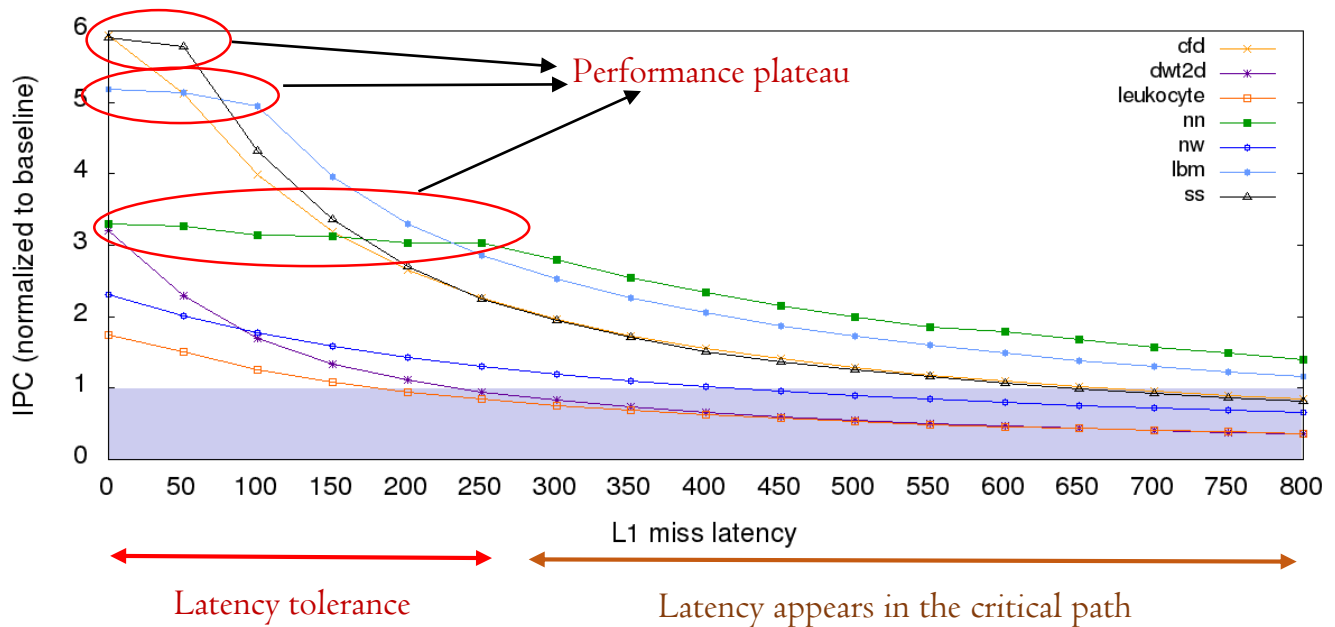


# Baseline Configuration

---

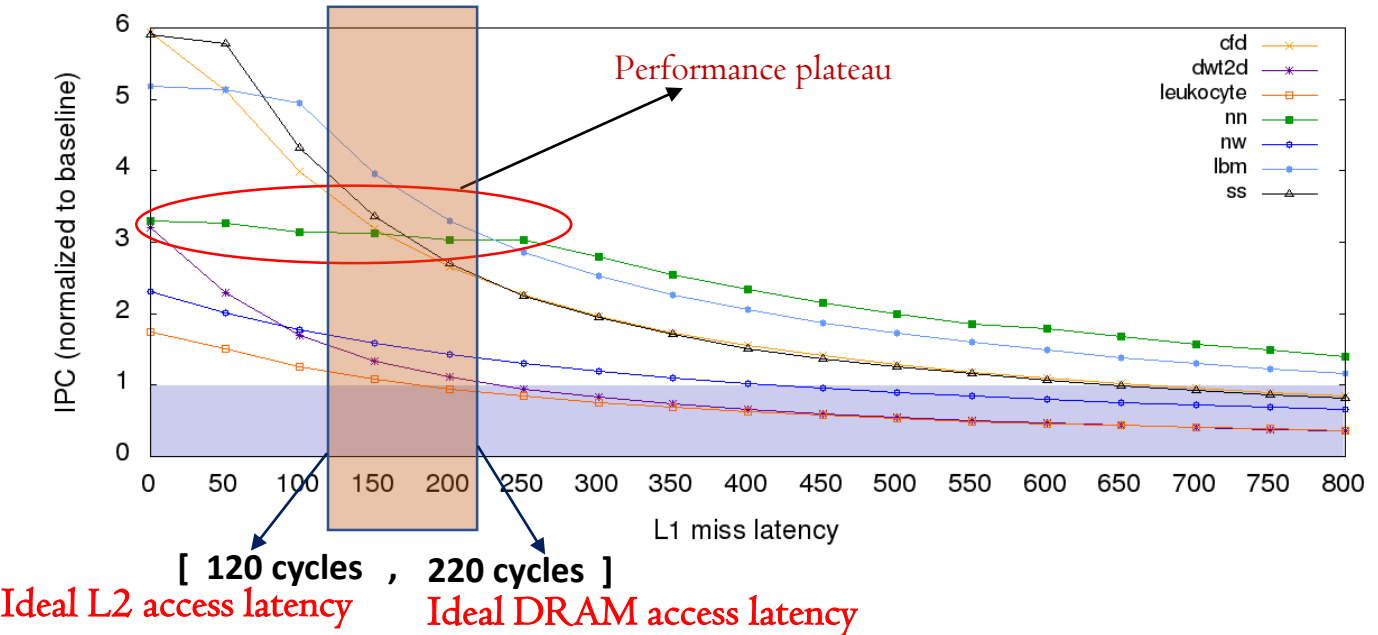
- GTX 480 NVIDIA GPU
  - 15 SMs
  - Private L1 Data Cache (16 KB; 32 MSHRs)
  - Shared L2 Cache (768 KB; 32 MSHRs/bank)
  - L1-L2 Interconnect (Crossbar; 32+32 bytes)
  - DRAM (384 bits bus width)

# Latency Tolerance



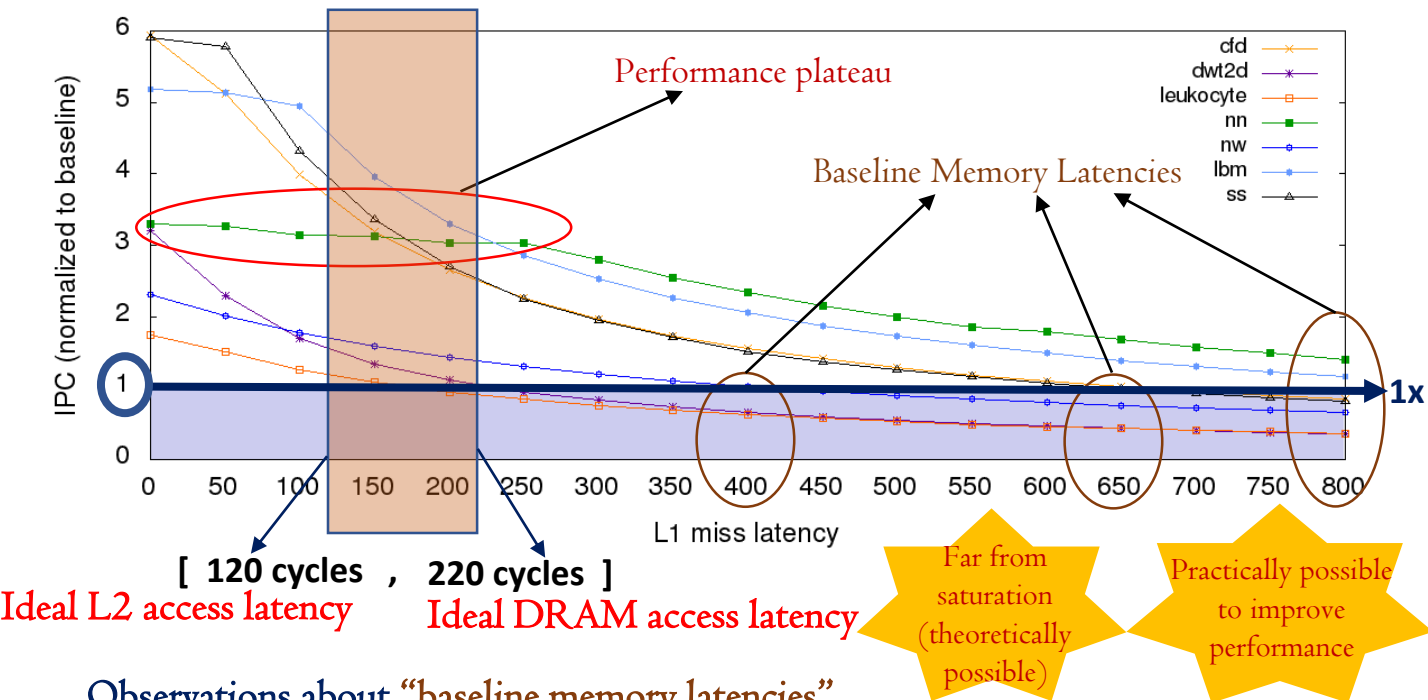
Performance versus Latency curve for memory-intensive benchmarks

# Latency Tolerance



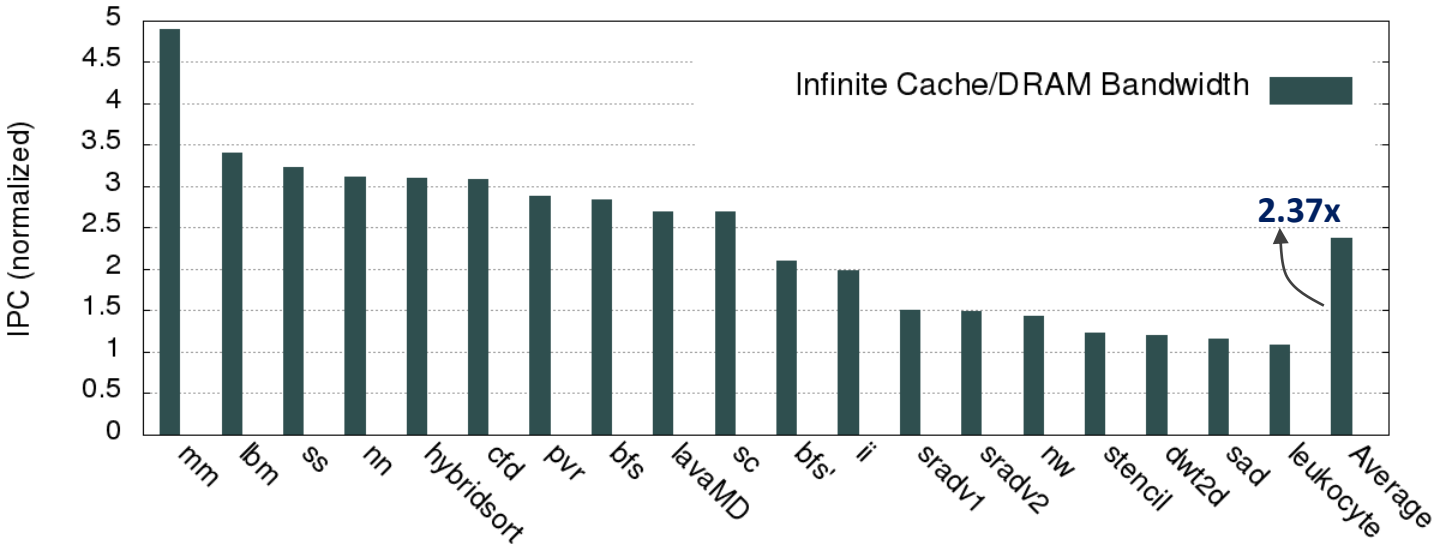
Added latencies due to increasing congestion

# Latency Tolerance

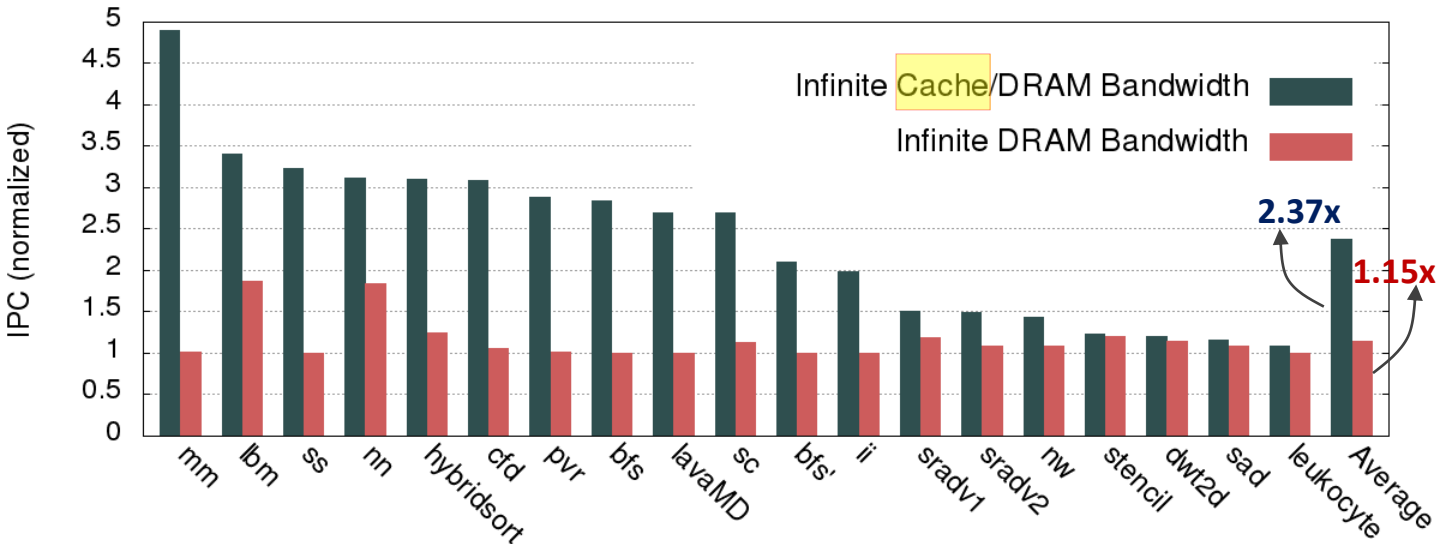


1. Baseline memory latencies critically higher than performance plateau latencies
2. Baseline memory latencies critically higher than ideal access latencies to L2/DRAM

# Infinite Bandwidth

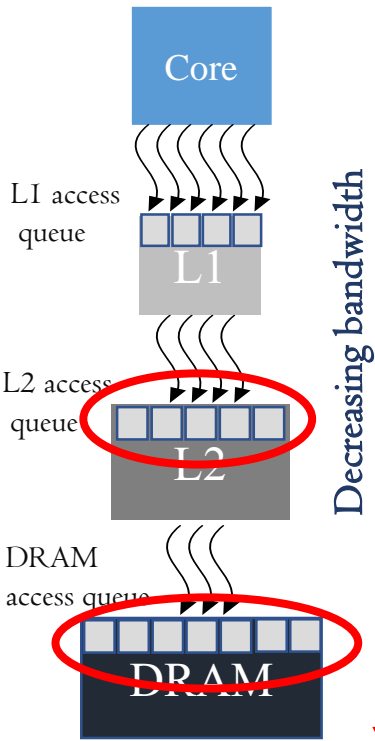


# Infinite Bandwidth



Significant congestion in the cache hierarchy

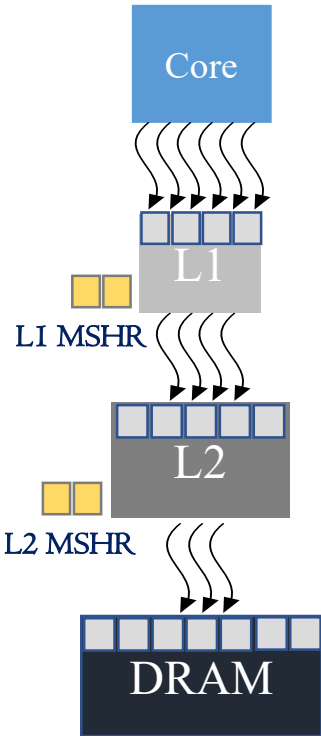
# Understanding Bandwidth Bottleneck



- While the **bandwidth provided** decreases in the lower levels of the memory hierarchy, **bandwidth demand** does not reduce proportionally.
- This leads to a **bandwidth skew** between adjacent levels.
- As a result, requests **queue up** in the memory hierarchy for long durations, causing congestion.
- **L2 access queues** are **full** for **46%** of its usage lifetime.
- **DRAM access queue** are **full** for **39%** of its usage lifetime

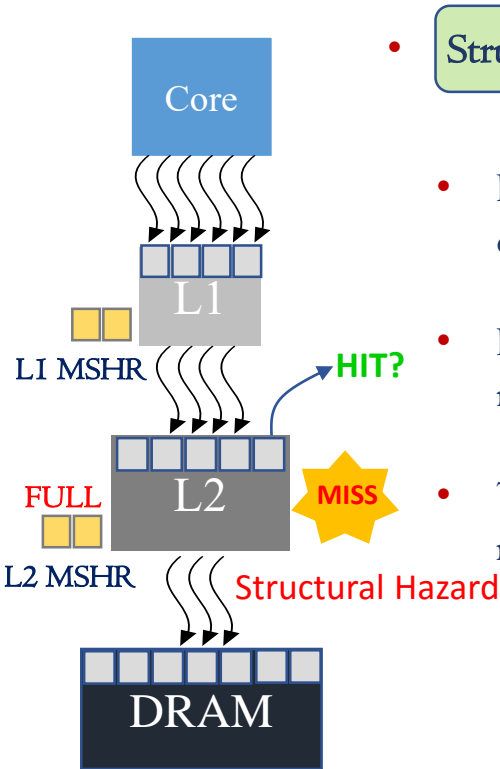
# Causes of congestion

- Structural Hazards
- Back Pressure





# Causes of congestion

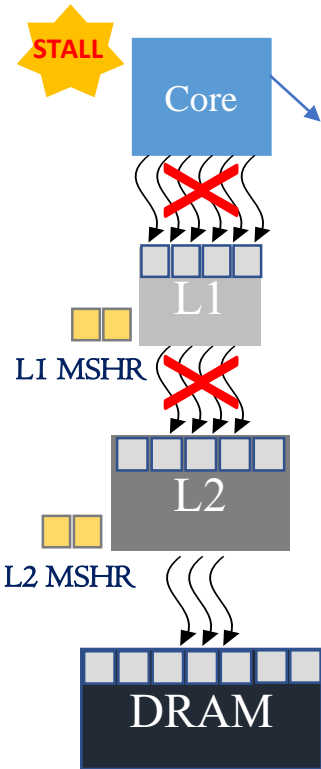


## • Structural Hazards • Back Pressure

- Prolonged **contention** for **cache resources** such as MSHRs or replaceable cache lines.
- Pending requests must **complete and relinquish** the resources.
- Therefore, new miss requests get **serialized**, increasing the memory latencies even more.

High cache hit latencies

# Causes of congestion



- Structural Hazards
- Back Pressure

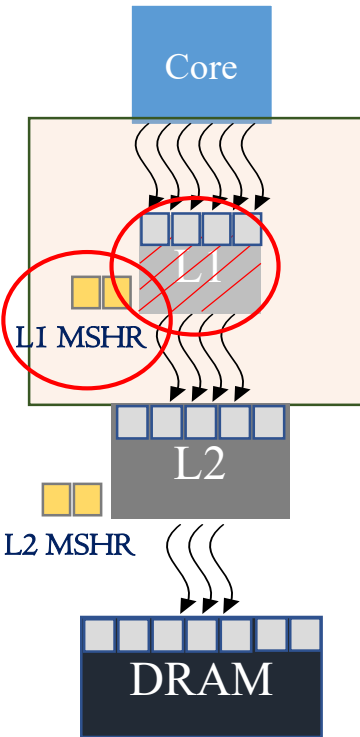
Independent compute?

- Cascading effect of structural hazards
- Higher level gets throttled
- Eventually throttles core performance

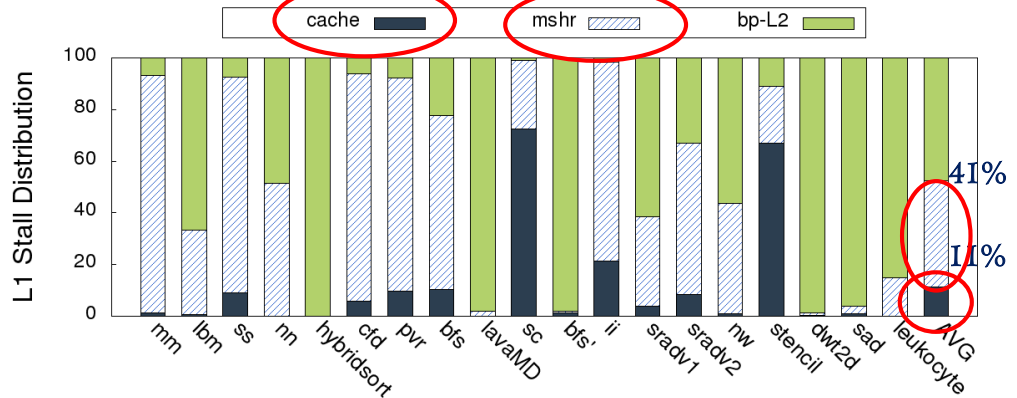
Restricted parallelism on cores

# Causes of congestion

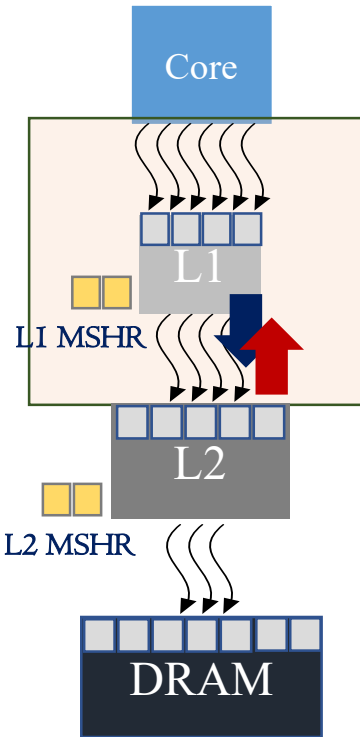
- Structural Hazards
- Back Pressure



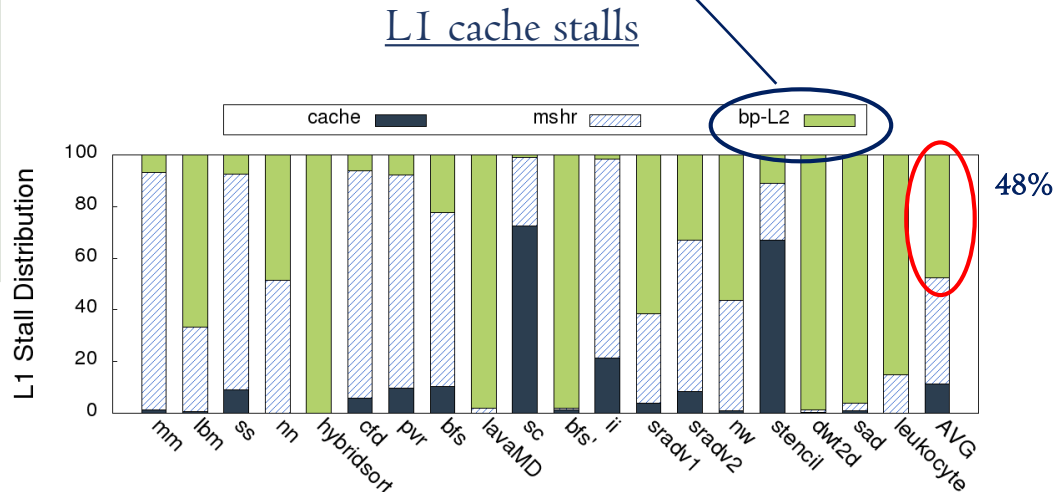
L1 cache stalls



# Causes of congestion



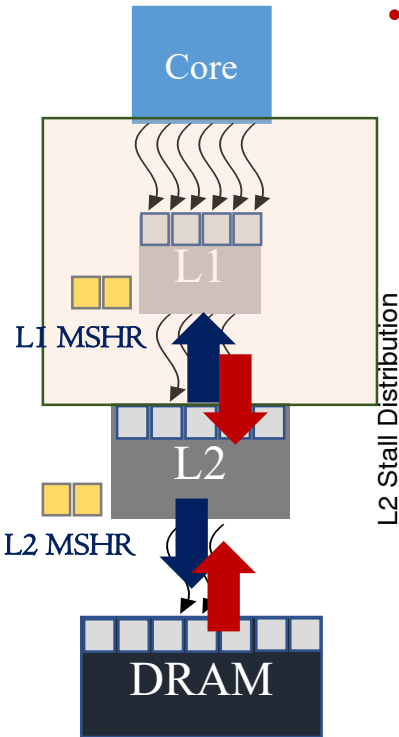
- Structural Hazards
- Back Pressure



## Major causes of stalls at L1

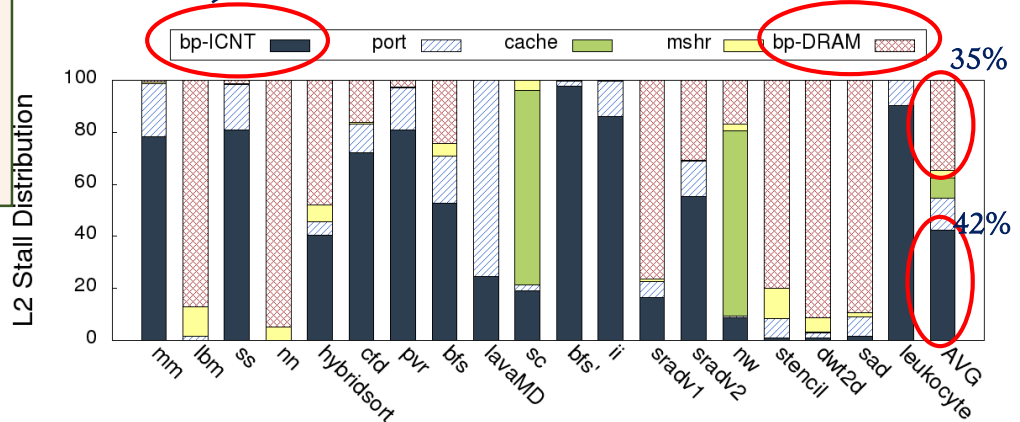
1. **L1 MSHR : 41%** (Structural Hazards)
2. **L2 back pressure : 48%** (Back pressure)

# Causes of congestion



- Structural Hazards
- Back Pressure

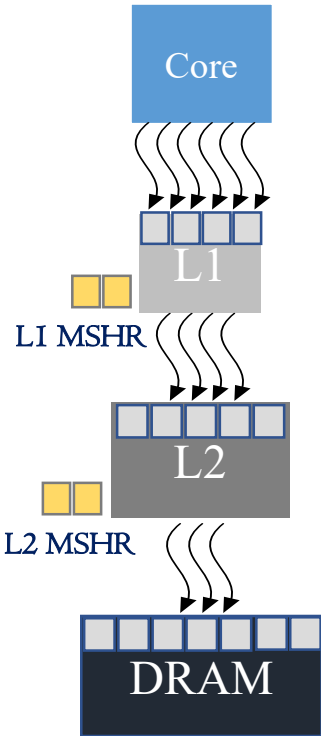
## L2 cache stalls



## Major causes of stalls at L2

1. Crossbar (response path) : 42% (Back pressure)
2. DRAM : 35% (Back pressure)

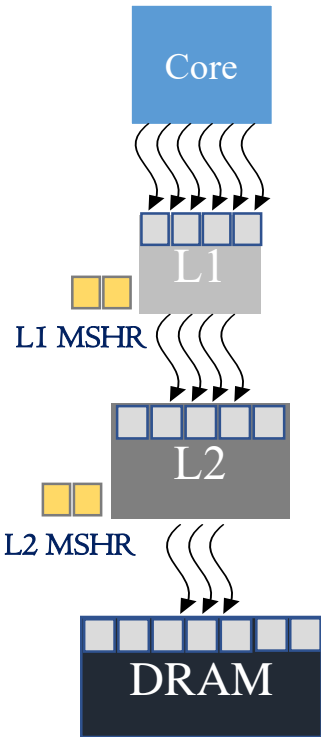
# Mitigating congestion







## Classifying the Design Space

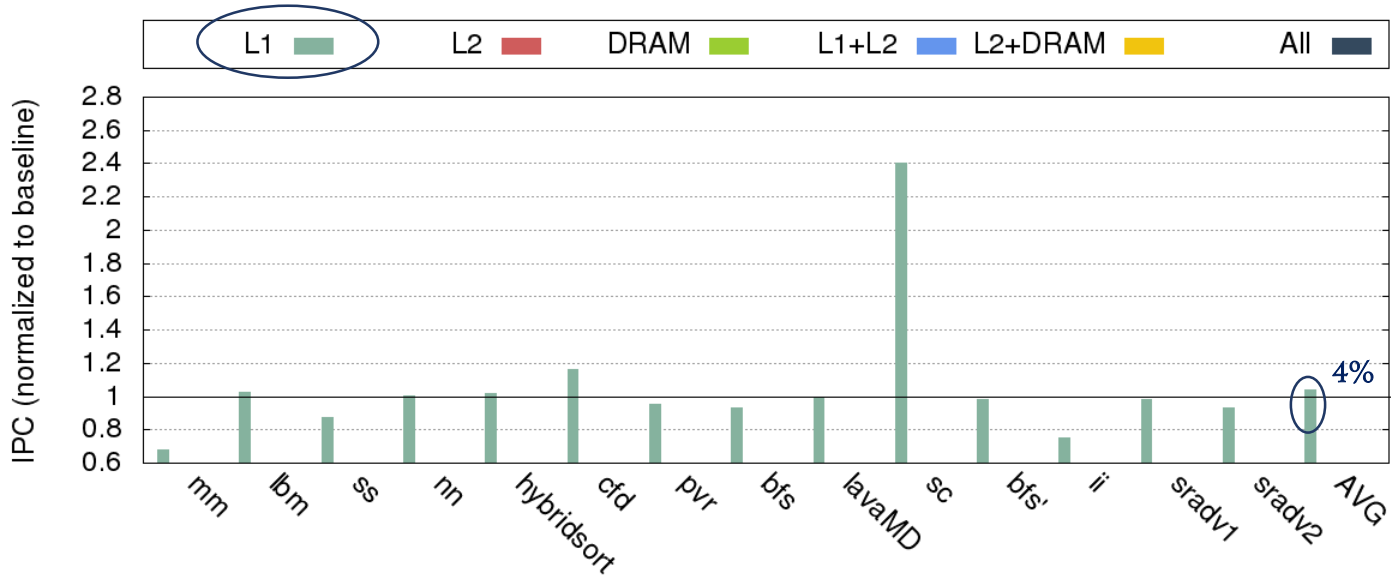
- **Category-1: Operate at peak throughput**
  - Minimize stalls by exploiting existing peak throughput
  - e.g. MSHRs, Access Queue size
- **Category-2: Increase peak throughput**
  - Minimize stalls by increasing the peak throughput
  - e.g. Crossbar flit size, DRAM bus width

# Identifying the Design Space



- **L1 parameters**
  - LI Miss Queue
  - LI MSHR
  - Memory pipeline width
- **L2 parameters**
  - L2 Miss/Response Queue
  - L2 MSHR
  -  L2 Data Port Width
  -  L2 Banks
  -  Flit Size (Crossbar)
- **DRAM parameters**
  - Scheduler Queue
  - Banks
  -  Bus width

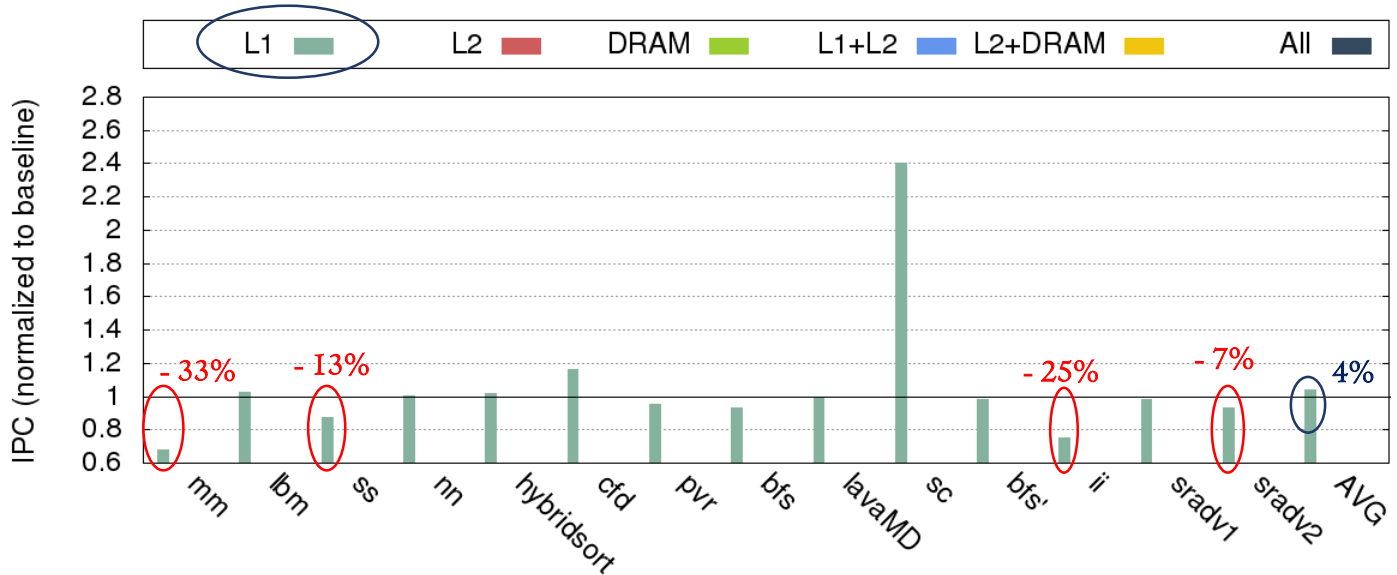
# Mitigating congestion



Scaling L1 parameters by 4x



# Mitigating congestion

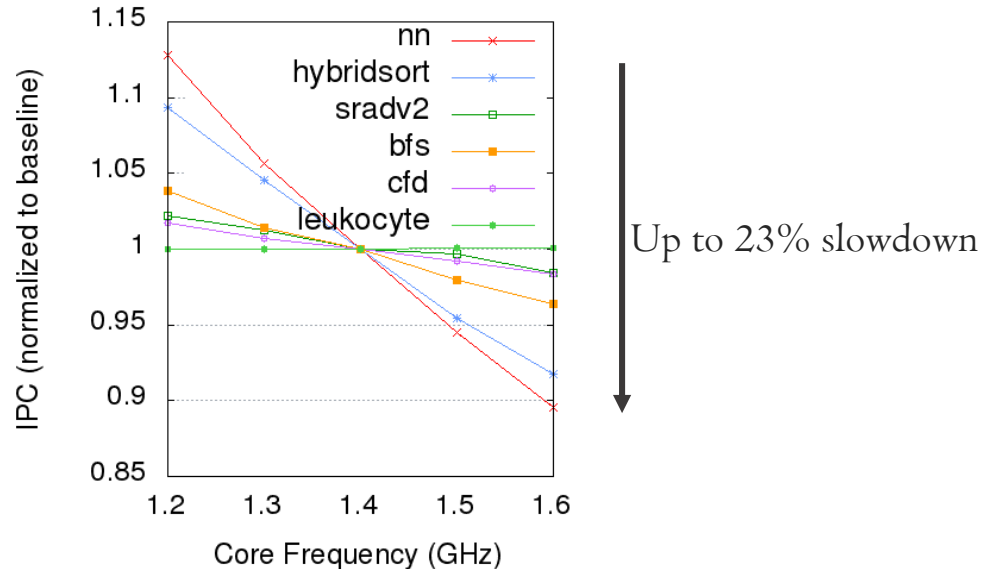


Scaling L1 parameters by 4x

Improving bandwidth in isolation can lead to even more congestion at the lower levels

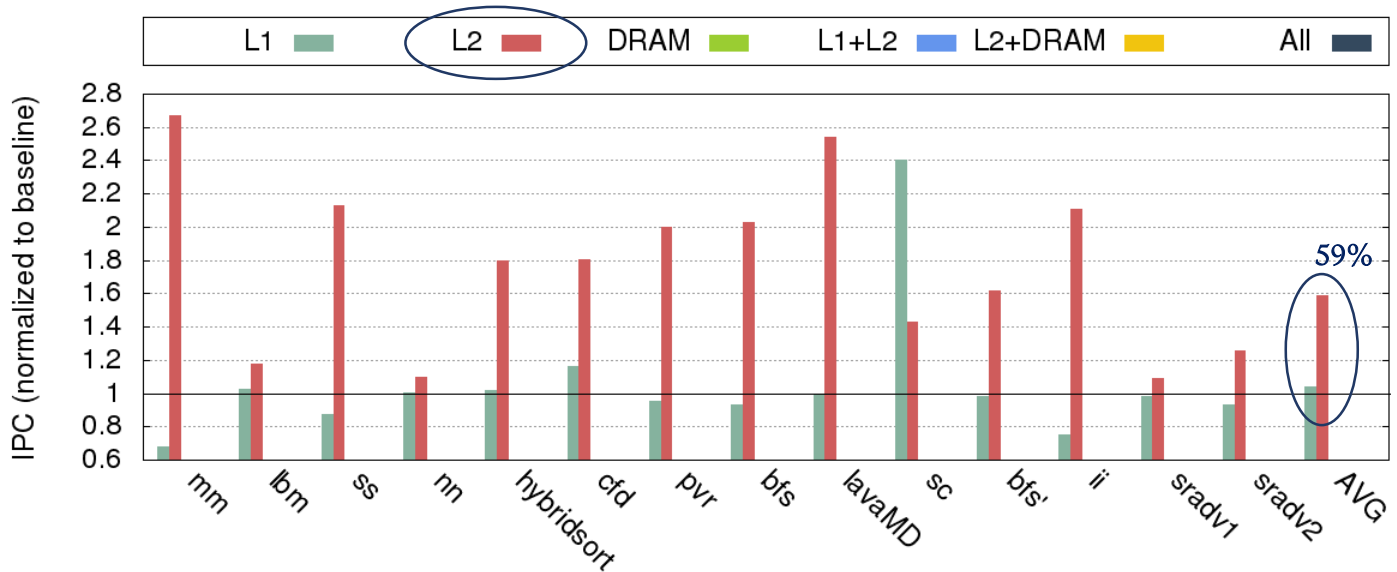
# Mitigating congestion

## Core frequency scaling on real GTX 480



Improving bandwidth in **isolation** can lead to even more congestion at the lower levels

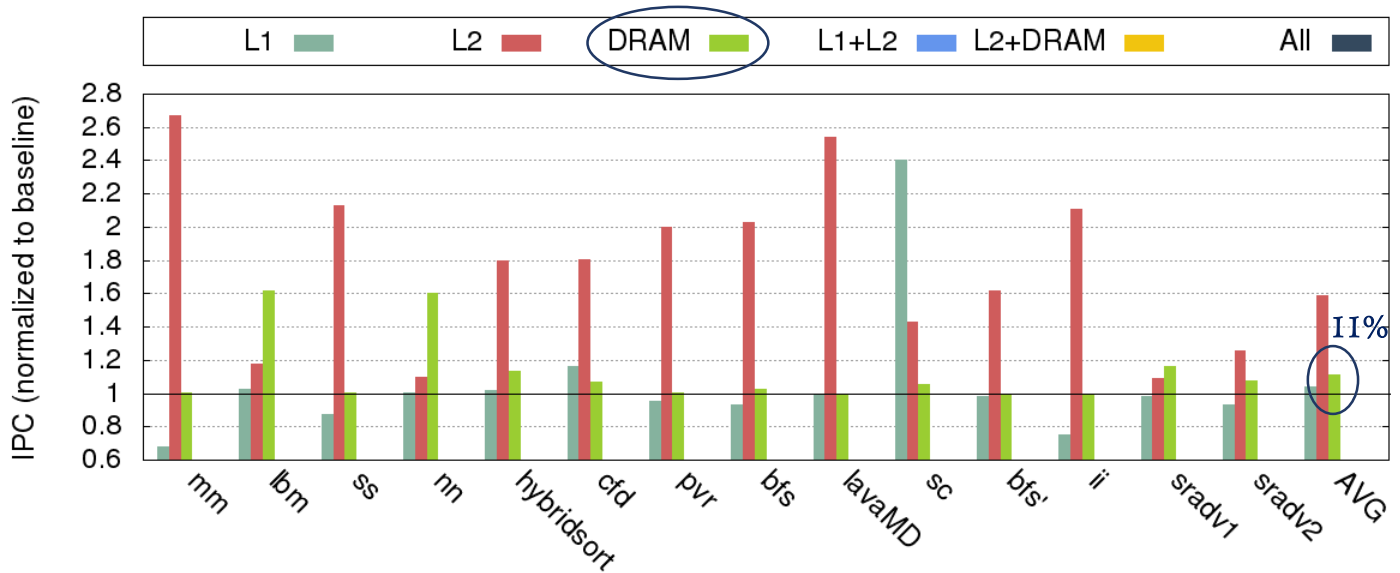
# Mitigating congestion



Scaling L2 parameters by 4x

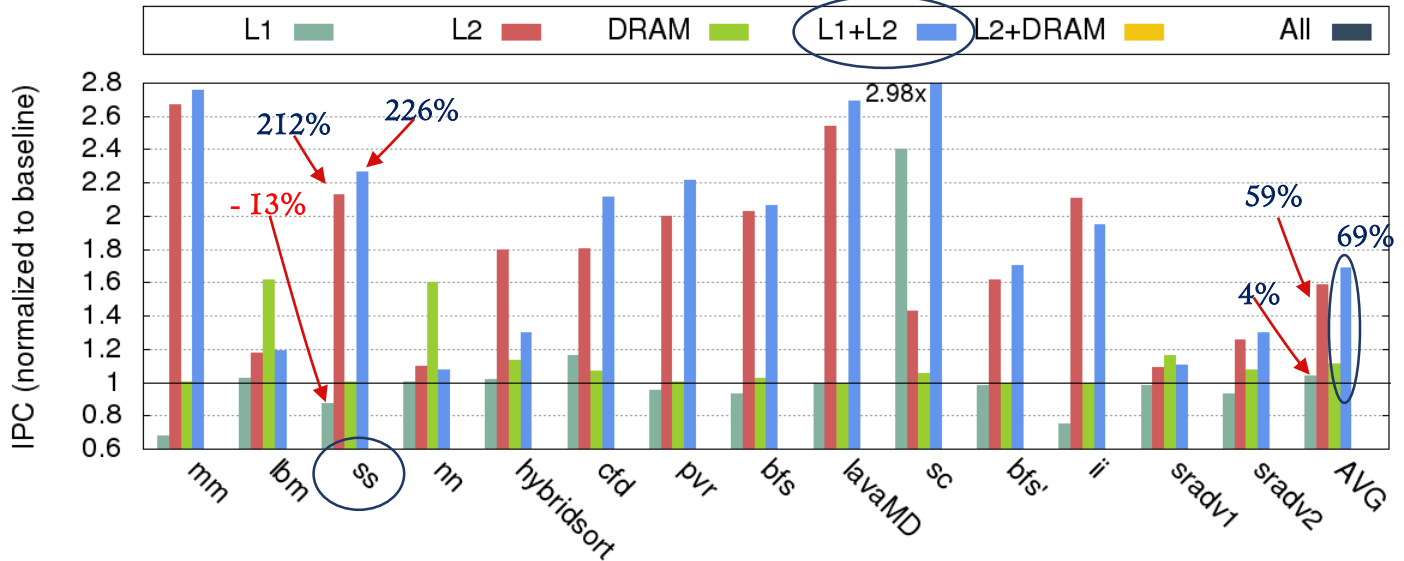
Shows the criticality of the L2 bandwidth

# Mitigating congestion



Scaling DRAM parameters by 4x  
(HBM)

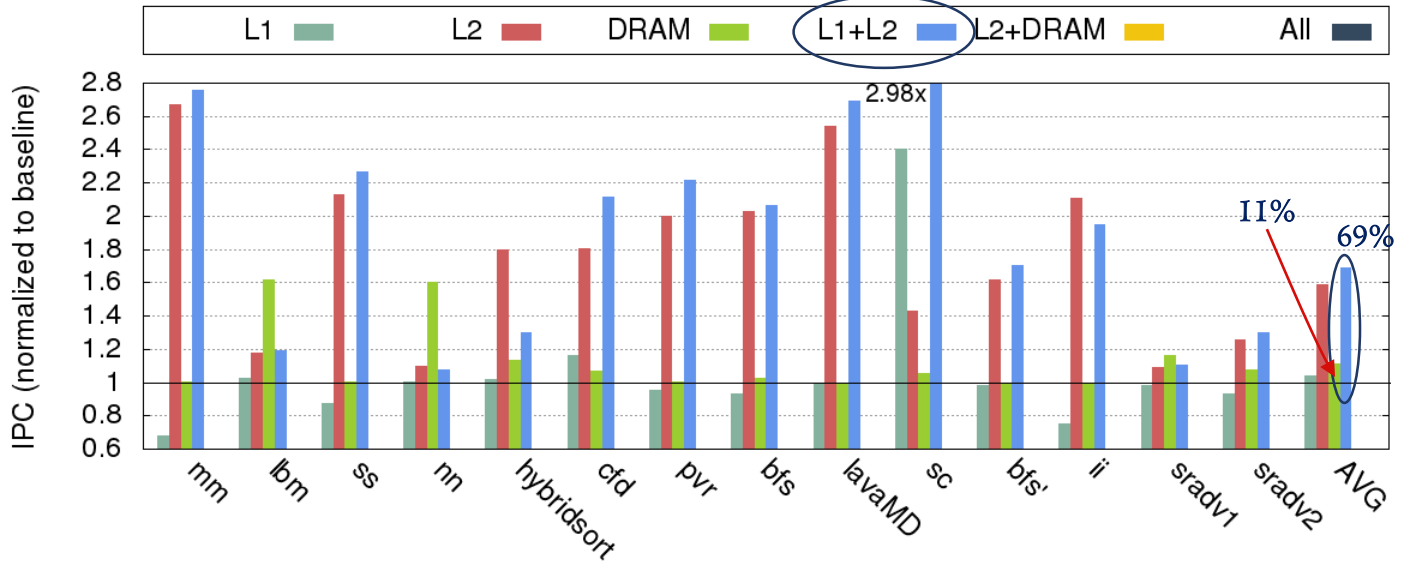
# Mitigating congestion



Scaling L1 and L2 parameters by 4x

A case for synergistic scaling!

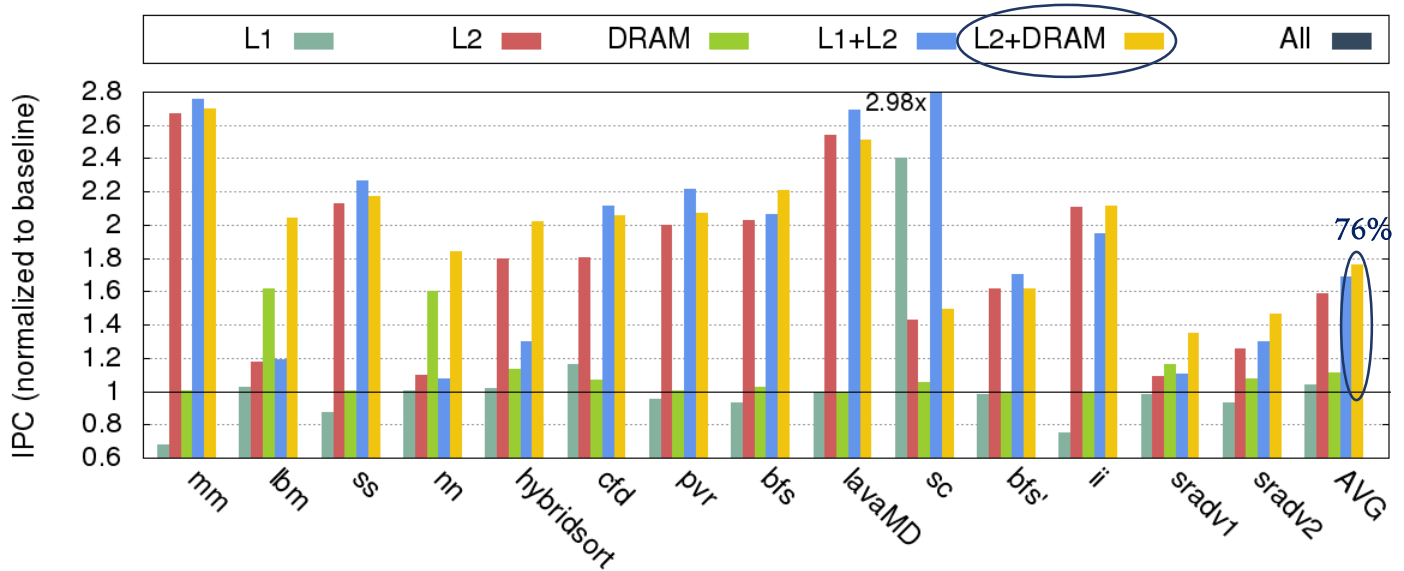
# Mitigating congestion



Scaling L1 and L2 parameters by 4x

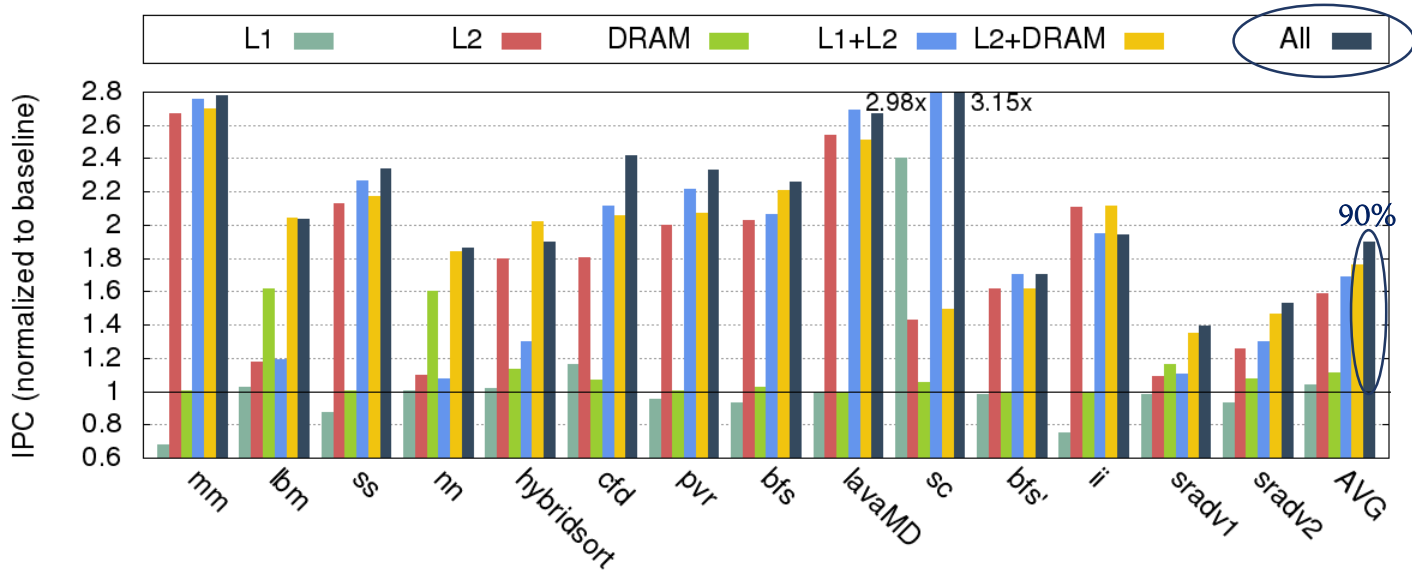
Higher speedup on mitigating congestion in the **cache hierarchy** compared to DRAM (as done in **HBM**)

# Mitigating congestion



Scaling L2 and DRAM parameters by 4x

# Mitigating congestion



Scaling the entire memory hierarchy by 4x



# Pruning the Design Space

---

- Scaling all architectural parameters by 4x impractical
- Need to prune the design space
- We now know the ...
  - Causes of congestion (at each memory level)
  - Effects of reducing congestion (at different memory levels)

## Cost effective configuration

Mitigate causes where the effect is maximum

Boost bandwidth resources where it hurts most!

# Cost-effective Design Space

---

- L1 parameters
  - L1 Miss Queue
  - L1 MSHR
  - Memory pipeline width
- L2 parameters
  - L2 Miss/Response Queue
  - ~~L2 MSHR~~
  - ~~L2 Data Port Width~~
  - ~~L2 Banks~~
  - Flit Size (Crossbar)
- DRAM parameters
  - ~~Scheduler Queue~~
  - ~~Banks~~
  - ~~Bus width~~

# Cost-effective Design Space

- L1 parameters

- LI Miss Queue
- LI MSHR
- Memory pipeline width

- L2 parameters

- L2 Miss/Response Queue



Simple Buffers

Minimal cost of scaling

Scale by 4x

- Flit Size (Crossbar)

# Cost-effective design-space

- L1 parameters

- LI Miss Queue
- LI MSHR
- Memory pipeline width



Fully Associative  
Array

Moderate cost of scaling

Scale by 1.5x

- L2 parameters

- L2 Miss/Response Queue

- Flit Size (Crossbar)

# Cost-effective design-space

- L1 parameters

- LI Miss Queue
- LI MSHR
- Memory pipeline width

- L2 parameters

- L2 Miss/Response Queue

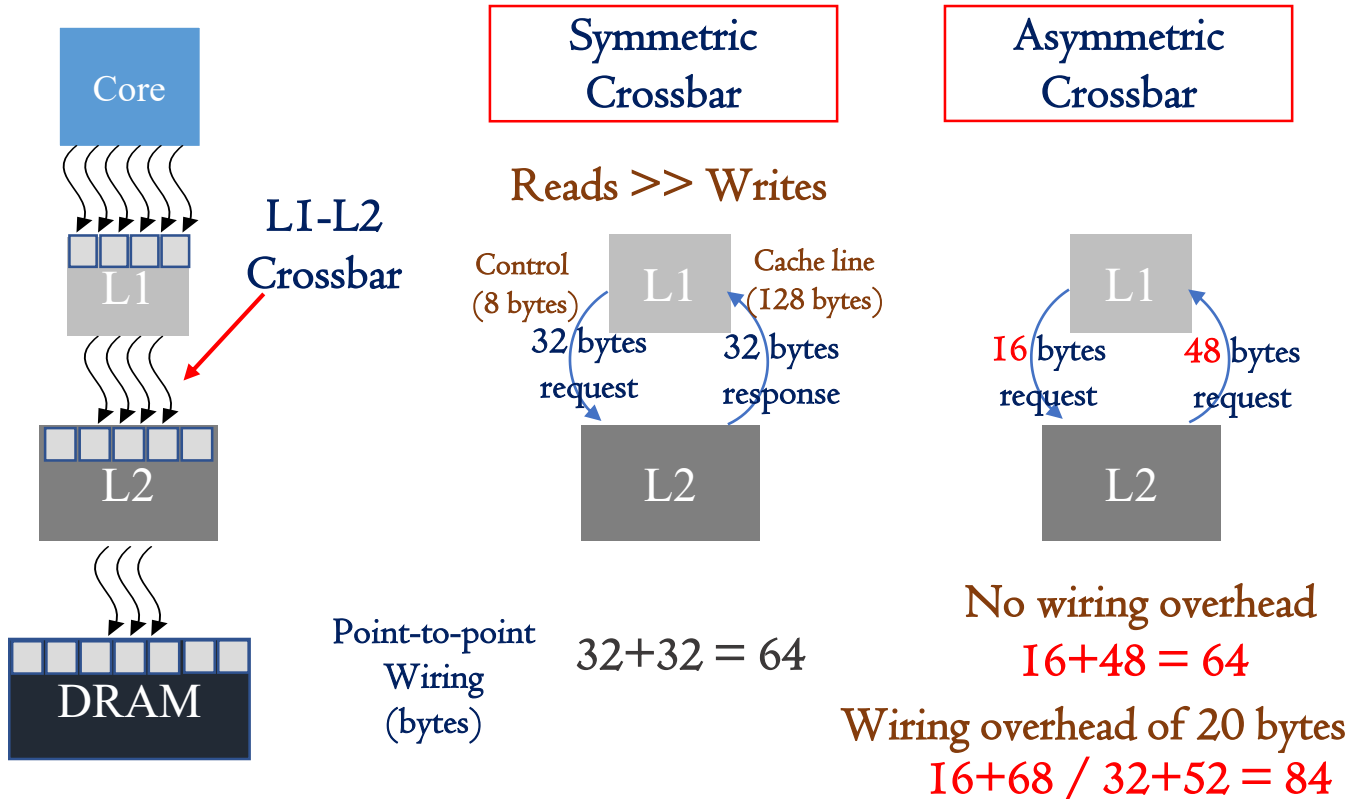
- Flit Size (Crossbar)

32+32 Baseline Crossbar

Scales quadratically with flit size

**“Asymmetric Crossbar”**

# Asymmetric Crossbar



# Cost-effective Design Space: Summary

## L1 Cache

- L1 Miss Queue : 8 entries → 32 entries
- Memory pipeline width : 10 wide → 40 wide
- L1 MSHR : 32 entries → 48 entries

## L2 Cache

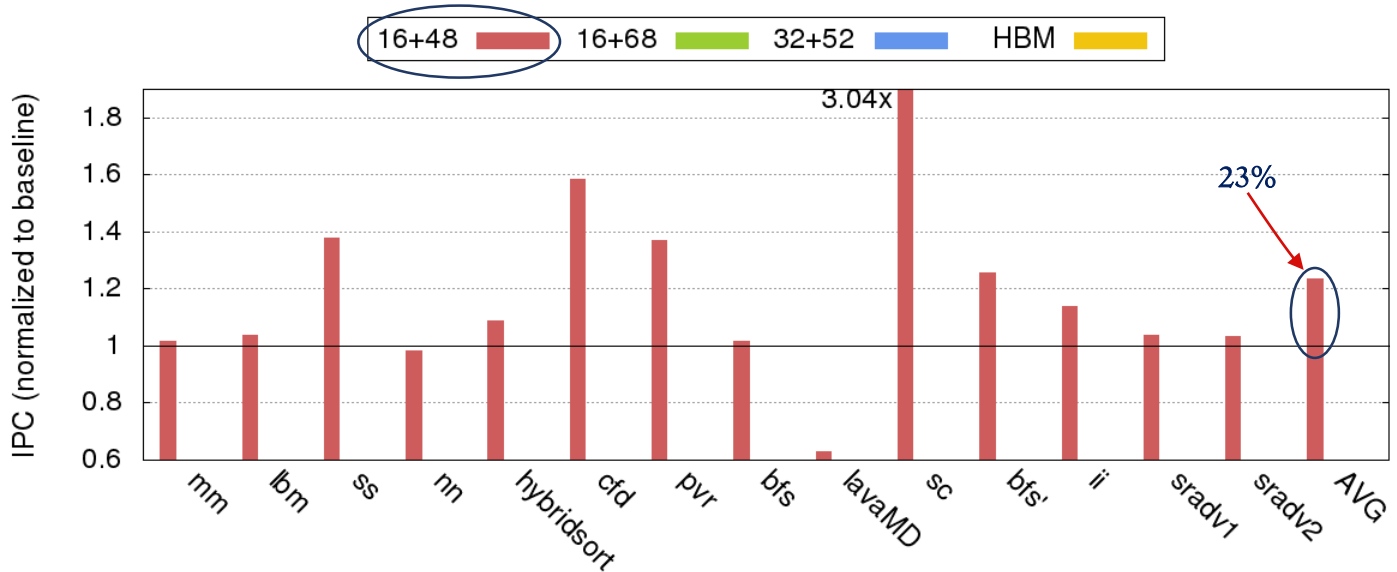
- L2 Miss/Response Queue : 8 entries → 32 entries
- Flit Size (Crossbar) : 32+32 → 16+48 (=64), 16+68, 32+52 (=84)

Evaluate 3 cost-effective configurations:

16+48    16+68    32+52

# Results

## Cost-effective configurations



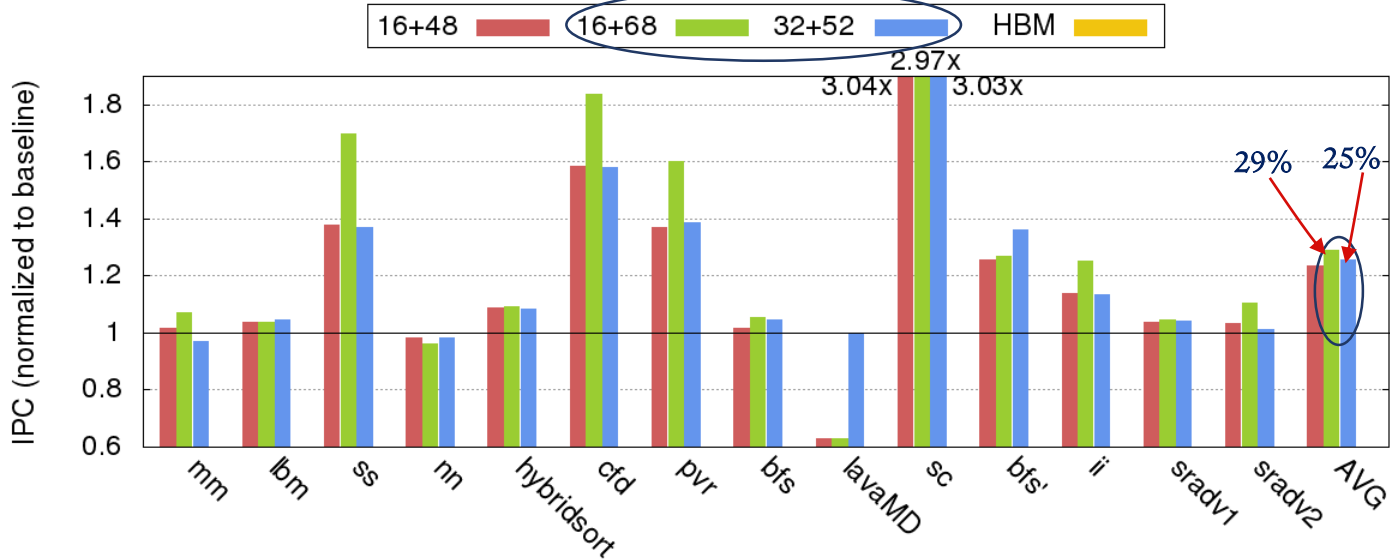
Area overhead: 1.1%

Point-to-point wires remains same as baseline



# Results

## Cost-effective configurations

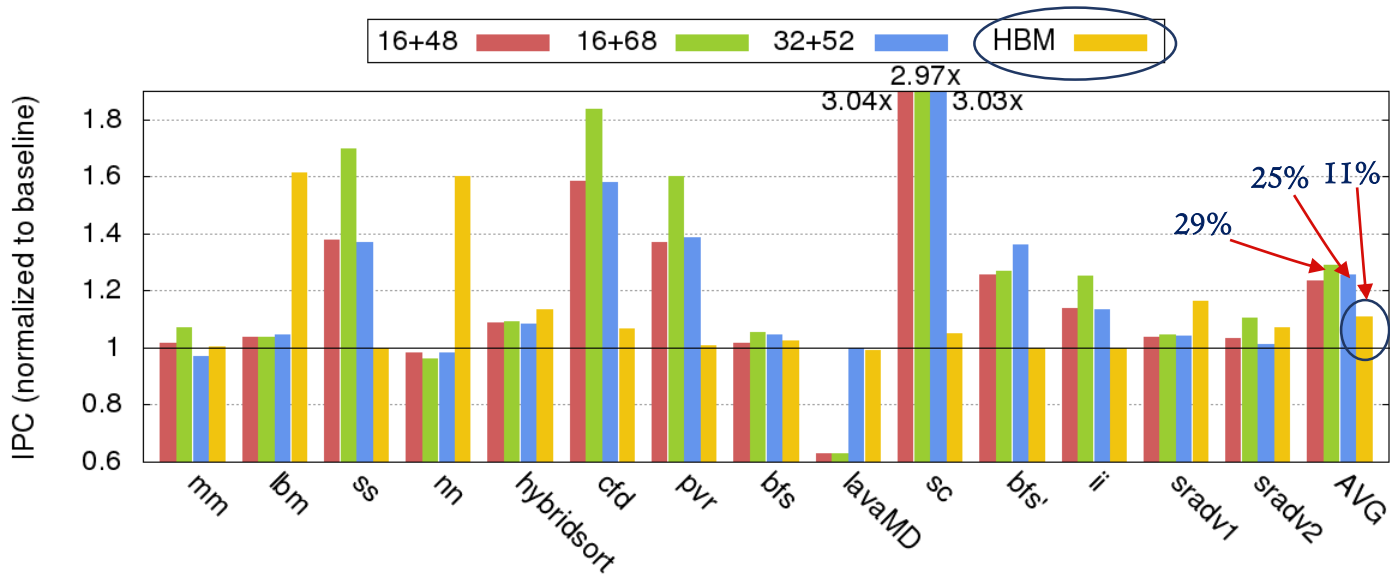


Area overhead: 1.6%

Investing in the **response path** gives better returns

# Results

## Cost-effective configurations



Higher speedup on resolving bandwidth bottleneck in cache hierarchy

Configuration with synergistic scaling (of L1 and L2) and asymmetric crossbar with higher response bandwidth (16+68) performs best

# Conclusion

---

- Problem

- High congestion **across** the memory hierarchy
- Congestion leads to high memory latencies (**both to L2 and DRAM**)
- High latencies **appear in the critical path** for memory-intensive applications, causing slowdown

- Observation

- Characterize **stalls** and develop **insights** about bandwidth bottleneck
- Significant bandwidth bottleneck in the **cache hierarchy**
- Addressing bandwidth problem in **isolation** can even lead to **slowdown**

- Proposal

- **Synergistic scaling** of bandwidth of L1 and L2 cache
- **Asymmetric scaling** of bandwidth of crossbar
- **23%** speedup with **1.1%** area overhead (no additional wires in crossbar)
- **29%** speedup with **1.6%** area overhead (additional wiring in crossbar)

# Questions?

---

**Saumay Dublish**

saumay.dublish@ed.ac.uk

<http://homepages.inf.ed.ac.uk/s1433370/>



Institute for Computing  
Systems Architecture



THE UNIVERSITY  
*of* EDINBURGH