MLP-Aware Instruction Queue Resizing: The key to Power-Efficient Performance

Pavlos Petoumenos, Georgia Psychou, Stefanos Kaxiras, Juan Manuel Cebrian Gonzalez and Juan Luis Aragon

University of Patras, Greece & University of Murcia, Spain

Introduction

- Out-of-order cores exhibit low power efficiency
- Instruction Queue: a major source of inefficiency
  - Common target for power-aware techniques
  - IQ is downsized unless ILP is reduced
- IQ-resizing also affects Memory Level Parallelism (MLP)
  - No existing IQ-resizing technique deals with MLP

Introduction

- Out-of-order cores exhibit low power efficiency
- Instruction Queue: a major source of inefficiency
  - Common target for power-aware techniques
  - IQ is downsized unless ILP is reduced
- IQ-resizing also affects Memory Level Parallelism (MLP)
  - No existing IQ-resizing technique deals with MLP

- Contributions:
  - MLP = main factor of performance loss
  - Integration of MLP-awareness in IQ-resizing techniques

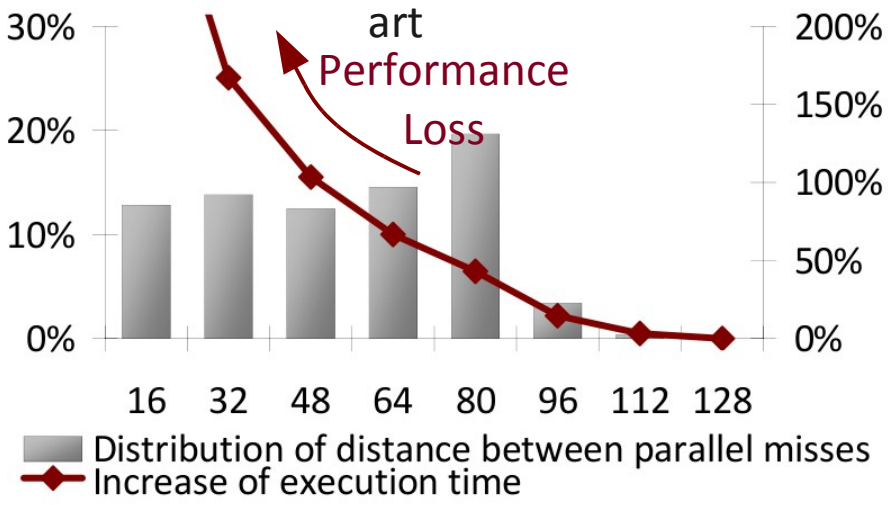
Instruction Queue Resizing

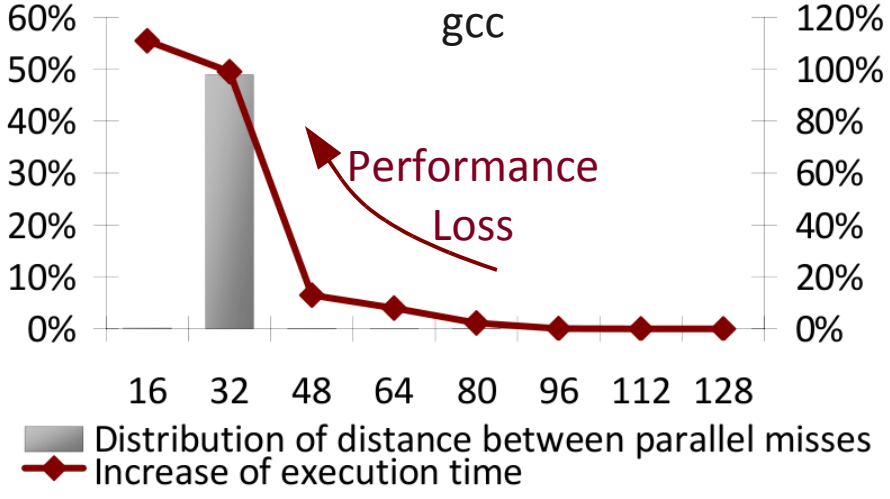
- Why IQ Resizing?
  - Power hungry structure
  - Built to support peak performance
- Significant prior work:
  - Buyuktosunoglu et al
  - Kucuk et al
  - Folegnani and Gonzalez
  - *Resizing triggered by ILP*

Memory Level Parallelism

- Memory wall
  - One LLC miss costs hundreds of cycles → stalls core
  - MLP = parallel (overlapping) misses in the IQ
    - → Changes the apparent cost of misses
- MLP-Aware techniques:
  - Qureshi et al: Keep non-parallel misses in the cache
  - Eyerman & Eeckhout: Don't stall fetch, if it hurts MLP

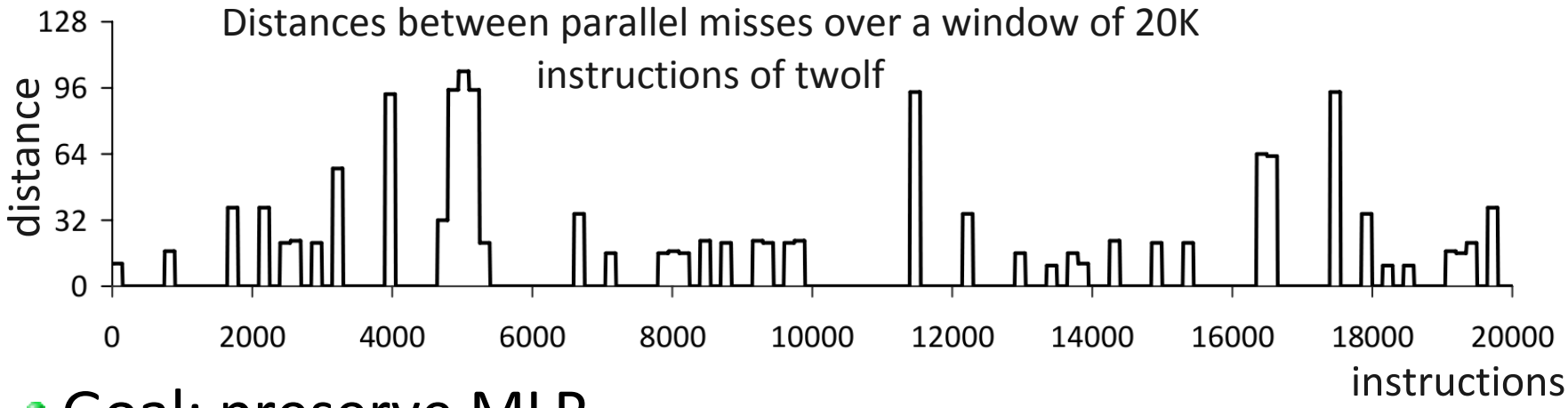
MLP and IQ Resizing





- Resizing the IQ
  - Smaller IQ turns parallel misses into isolated misses
  - MLP loss disproportionally hurts performance!

MLP and IQ Resizing

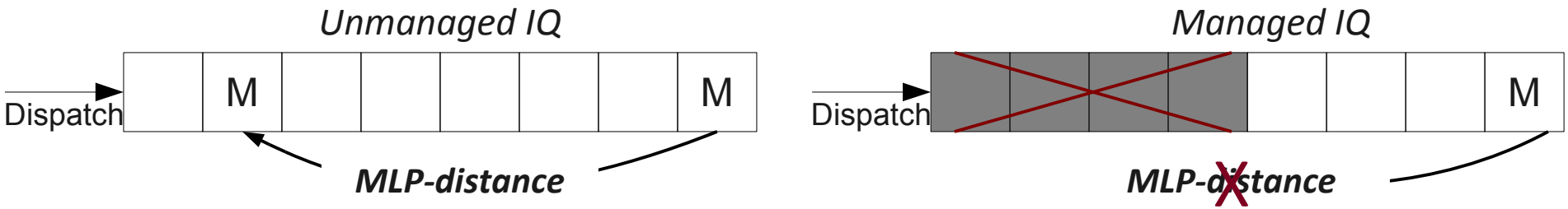


- Goal: preserve MLP
  - IQ size = large enough to keep parallel misses parallel
- But: existing techniques cannot handle MLP
  - Only ILP & IQ occupancy drive these techniques
  - MLP is fast-changing → Window-based management too coarse-grained to handle it
    - Average/max MLP not good enough

MLP-aware IQ Management

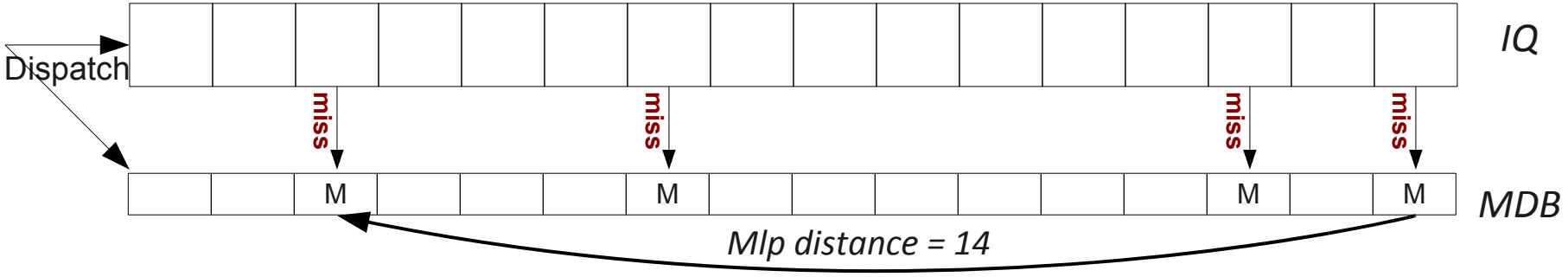
- Technique overview:
  - Quantify MLP information
  - Associate this information with parts of the instruction stream
  - Upon seeing the same instructions, make MLP-aware management decisions

Quantifying MLP



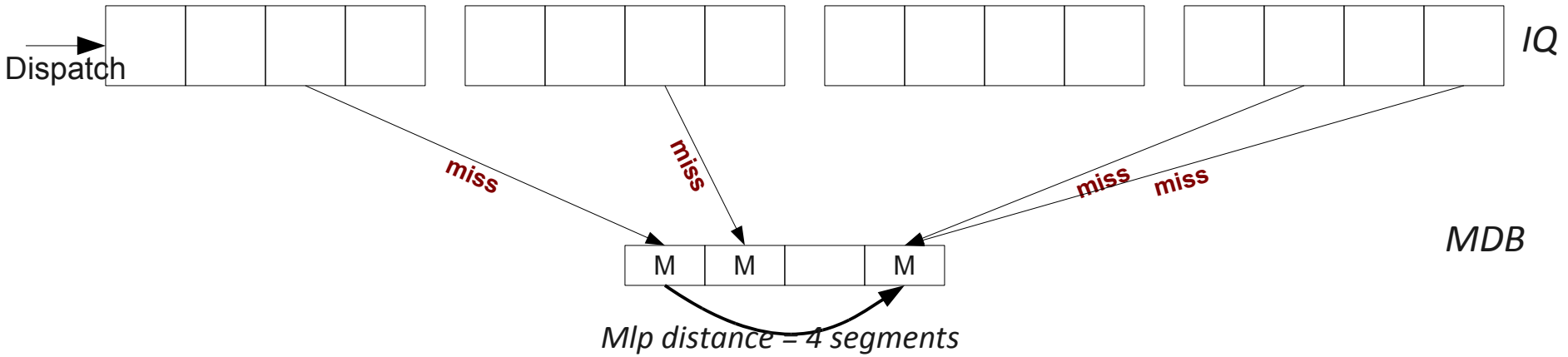
- MLP-distance = all instructions between 2 LLC misses
  - Remeber: 2 misses can overlap if MLP-distance < IQ size
- Measuring MLP-distances:
  - Straightforward way
    - For each serviced miss, find the youngest instruction in the IQ waiting for data from the memory
    - Problem: Can only identify overlapping misses for the current IQ size

Quantifying MLP



- MLP Distance Buffer – MDB:
  - Cyclic FIFO buffer *as big as the unmanaged IQ*
  - LLC miss → the corresponding MDB entry is marked
  - When an entry is retired, we search for the youngest MDB entry which has caused a miss

Quantifying MLP



- Efficient MDB:
  - Management works on segments
  - Management-related information does not need more resolution
  - One MDB entry for each IQ segment
  - Small and power-efficient

Associating MLP-distance with code

- Basic Blocks?
- Superpaths:
  - Aggregation of sequentially executed basic blocks
    - Uniquely identified by its starting address
  - Size comparable to the IQ size
    - Longer → fails to capture fast-changing MLP-distances
    - Shorter→ too short to manage the whole IQ
- We keep MLP-distance information for each superpath
  - We update it when all instructions of the superpath retire
  - Update value: the longest MLP-distance among MDB entries associated with the superpath

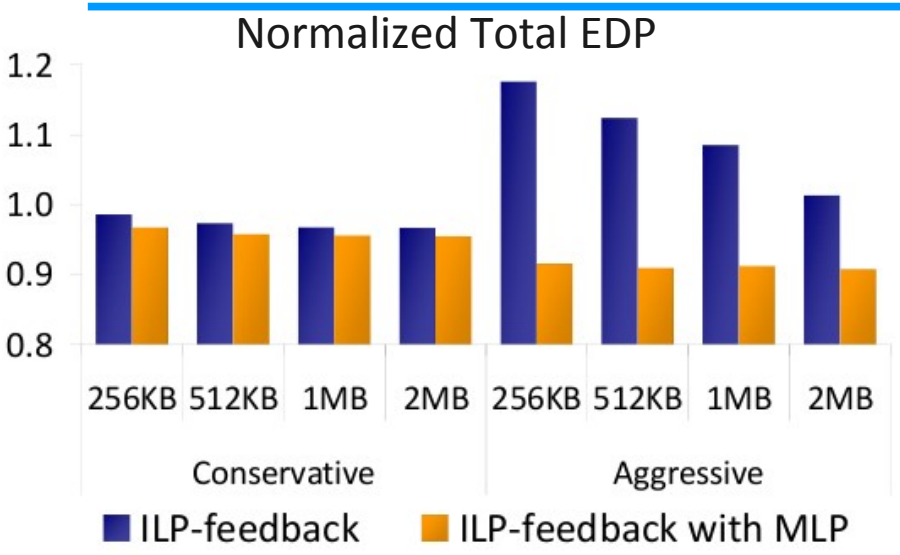
IQ Resizing Policy

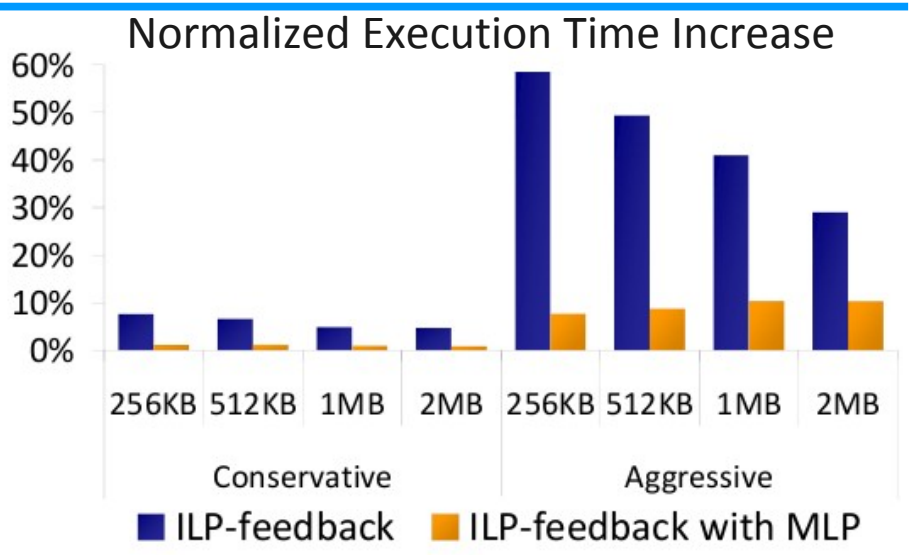
- Must protect both MLP and ILP
- MLP:
  - In dispatch, we track superpaths
  - If a superpath has stored MLP-distance information:
    - IQ target size >= MLP-distance
- ILP:
  - Any existing technique would work
  - We used the ILP-feedback approach of Folegnani & Gonzalez
    - If the youngest IQ segment does not contribute ”much” to IPC, turn it off
    - A threshold regulates what ”much” means

Evaluation

- Basic Setup:
  - Cycle accurate simulator
  - WATTCH power models
  - Memory intensive SPEC2000 benchmarks
  - 256KB-2MB unified L2 caches
  - Unified IQ-ROB: 8 sixteen-entry segments
- Comparison between the ILP-aware and the ILP/MLP-aware technique
- All results normalized to that of an unmanaged IQ

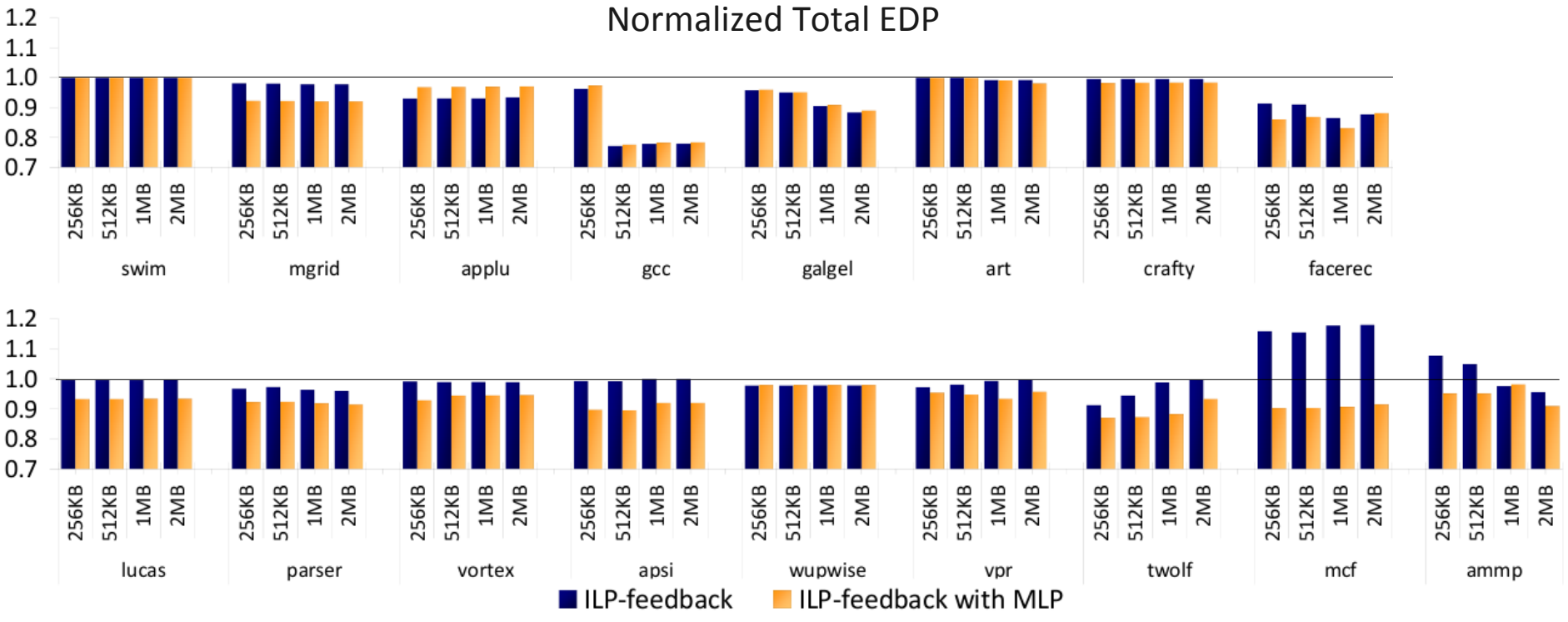
Effects of MLP-awareness





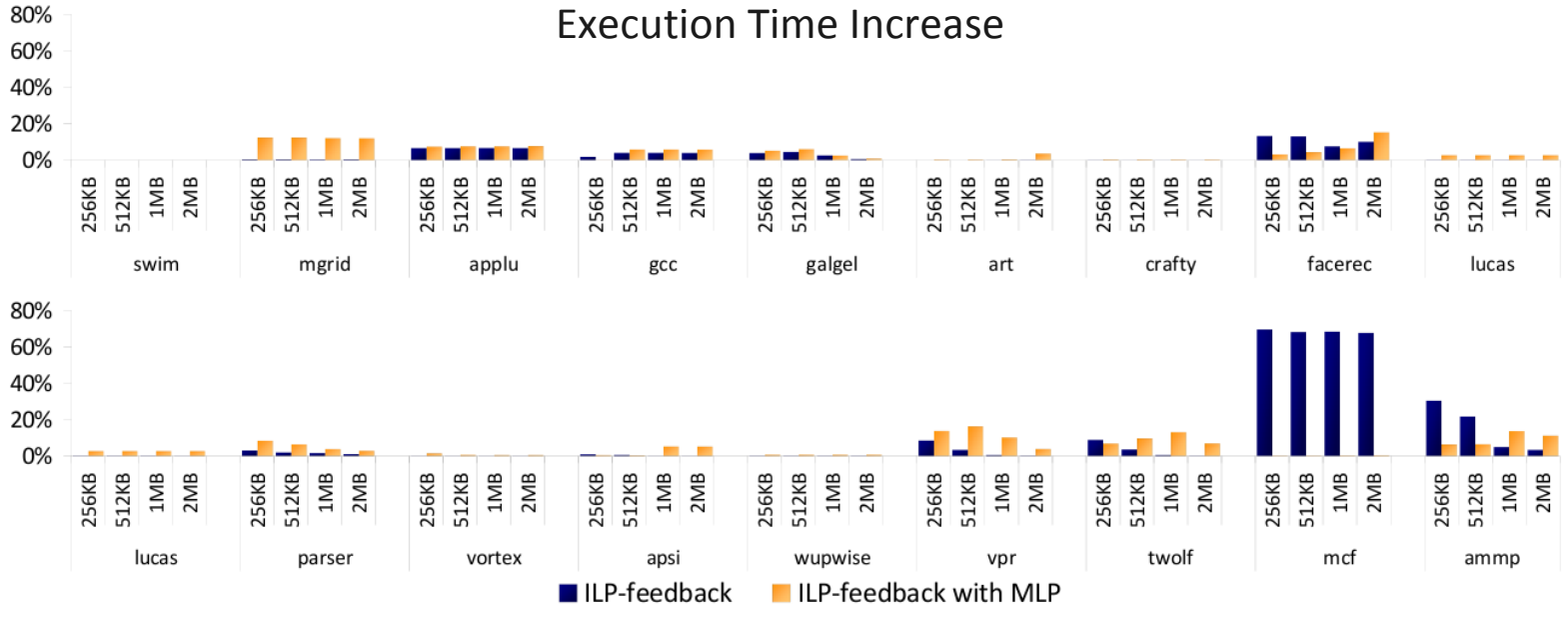
- ILP-feedback
  - Yields benefits when it doesn't hurt MLP
  - But has to stay conservative to achieve that
- ILP/MLP-feedback
  - Can be both ILP-aggresive and MLP-friendly

Direct Comparison - EDP



- ILP/MLP technique: consistently better than ILP
  - 6.1-7.2% EDP improvement vs 1.4-3.4% on average

Direct Comparison – Execution Time



- ILP/MLP technique:
  - Comparable performance degradation to the ILP-feedback technique while resizing is more aggresive
  - Except for mcf (~70% execution time increase for ILP)

Conclusions

- Prior IQ resizing approaches focus on ILP
  - Lack of MLP-awareness bounds their effectiveness
- Our technique provides this MLP-awareness
  - Tracks and predicts possible MLP
  - Overrides decisions of ILP-targeted mechanisms if they hurt MLP
- Consistently improves total processor EDP