The 2nd International Workshop on OpenCL (IWOCL'14)

MAP-Driven Performance Analysis for Local Memory Usage

Jianbin Fang[+], Henk Sips[+], Ana Lucia Varbanescu[++]

+ Delft University of Technology, the Netherlands

++ University of Amsterdam, the Netherlands

May 12-13, 2014

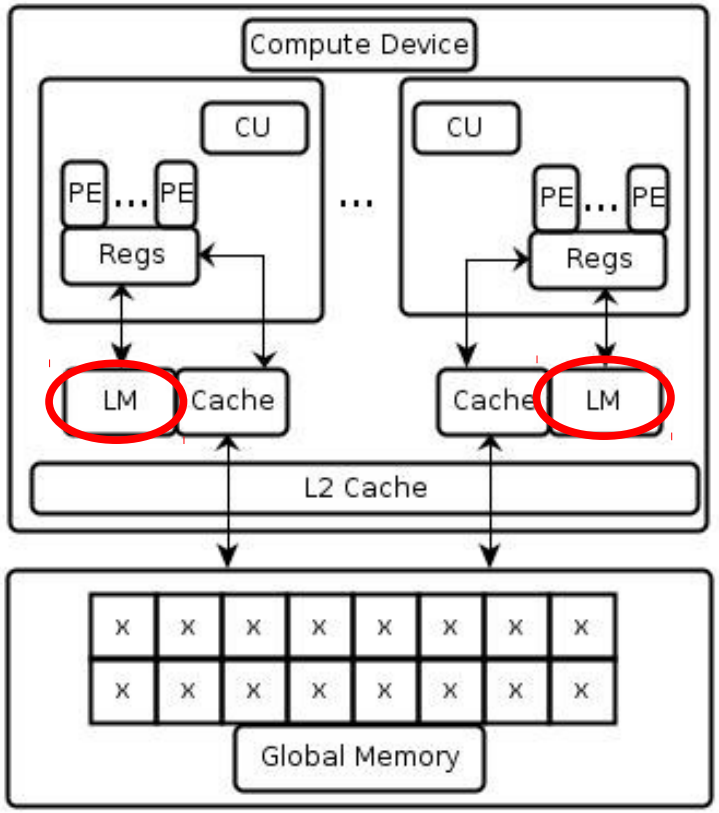
Bristol, England

Outline

- Introducing local memory (background)
- Our research question (why)
- Our approach (how)
- Our key findings (results)

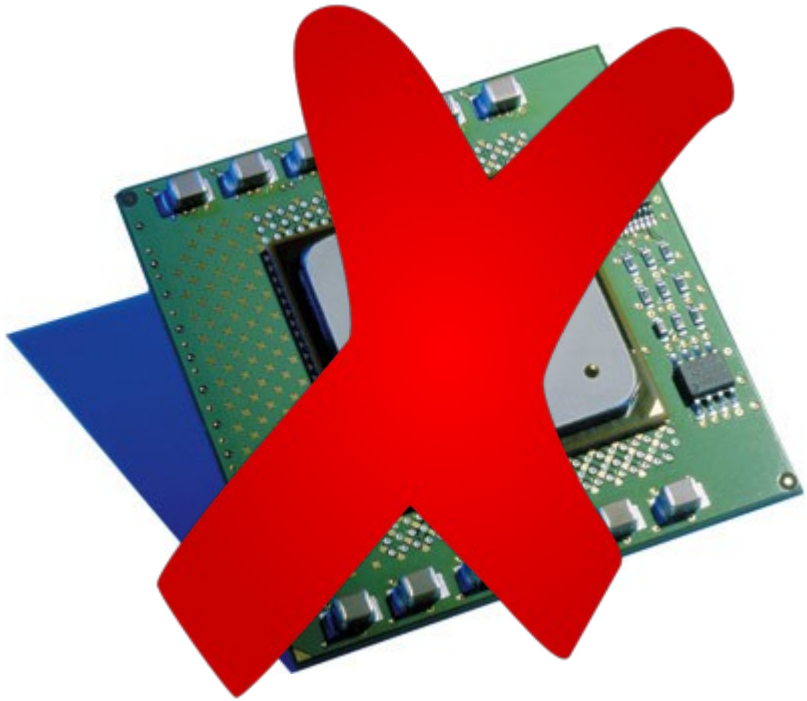
OpenCL and Local Memory

- Like-a-cache: on-chip and faster
- Not-a-Cache: user-managed
- Data elements are shared by work-items of a work-group



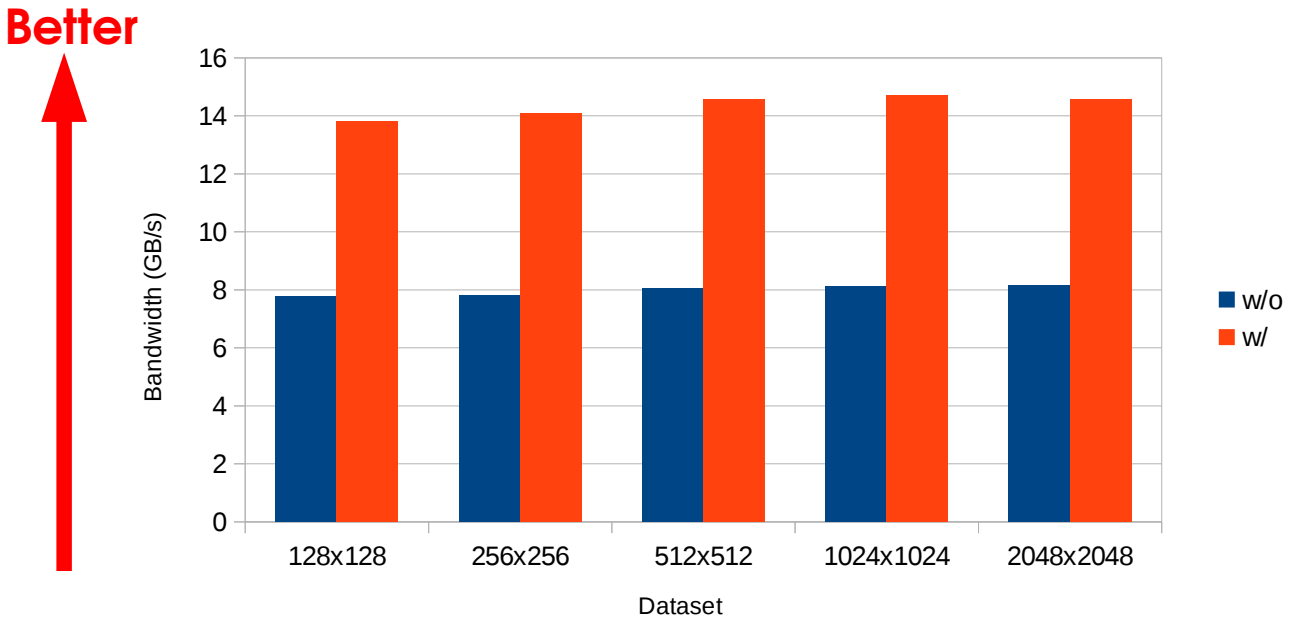
Rules of Thumb

- Using local memory on GPUs is preferred (e.g., data reuse)
- Using local memory on CPUs is not recommended



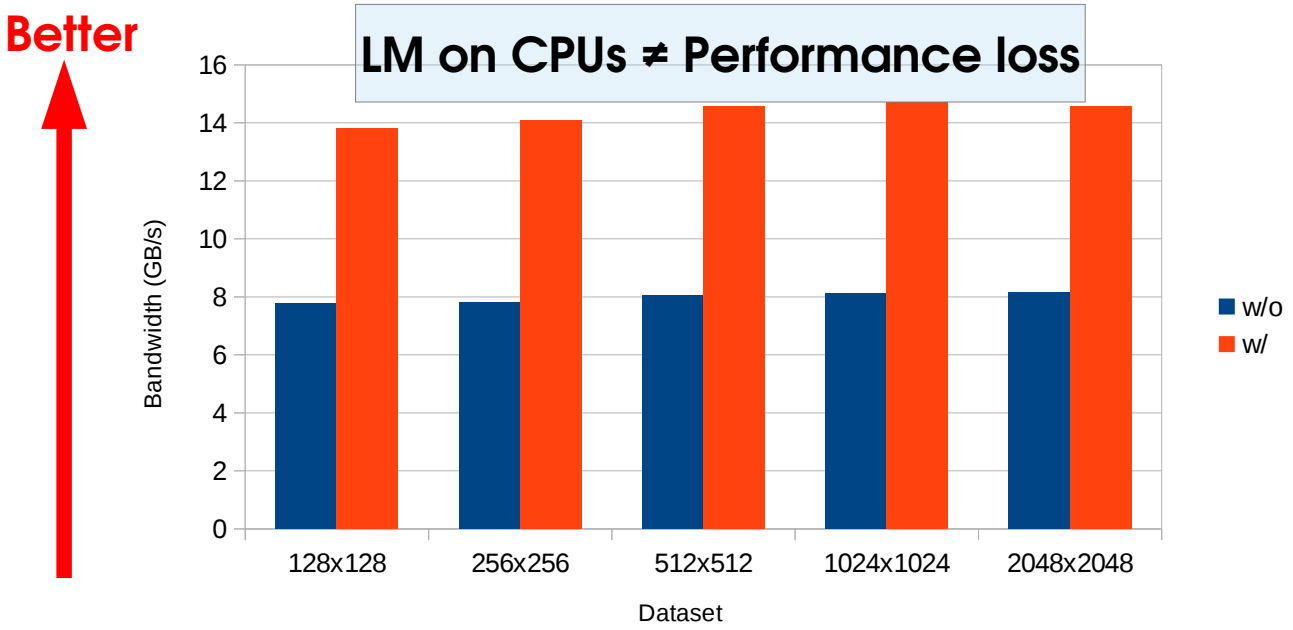
The Reality is ...

- A counter-intuitive example
  - 3x3 convolution
  - On Intel Xeon E5620 (6 cores)



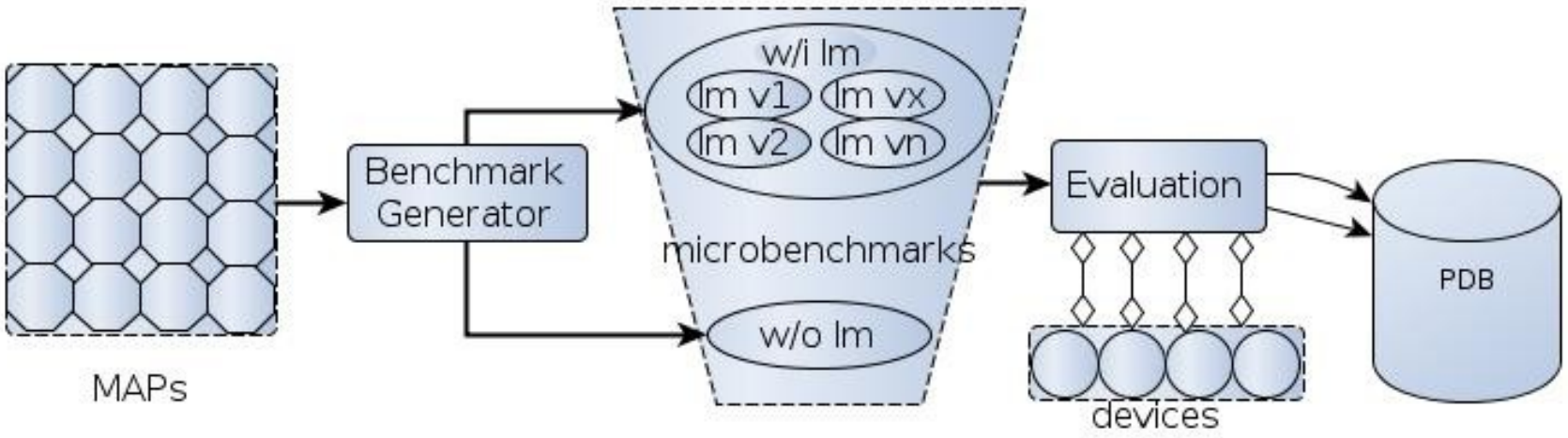
The Reality is ...

- A counter-intuitive example
  - 3x3 convolution
  - On Intel Xeon E5620 (6 cores)



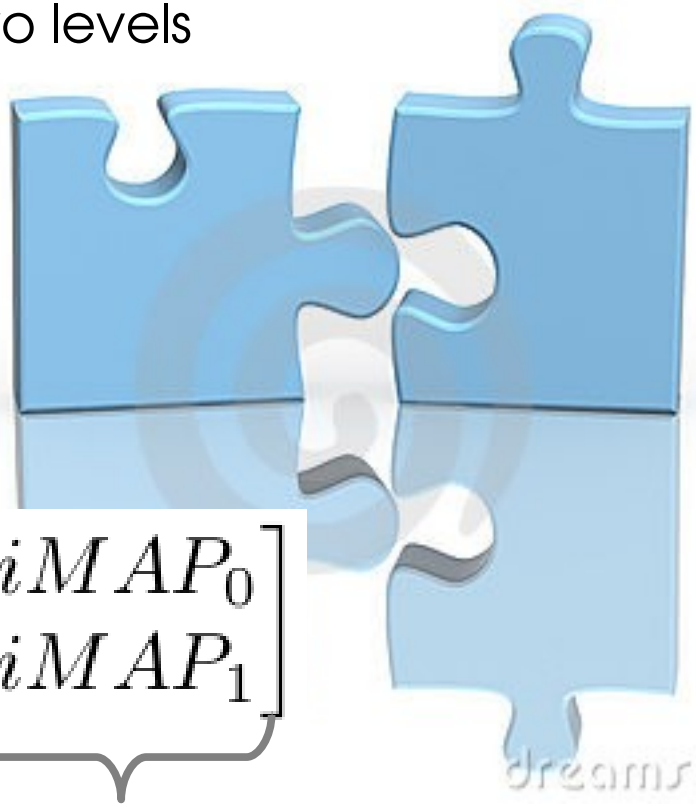
When to Use Local Memory?

Our Approach



MAP Description

- OpenCL organizes parallelism at two levels
- We describe a MAP at two levels
  - eMAP: work-group level
  - iMAP: work-item level

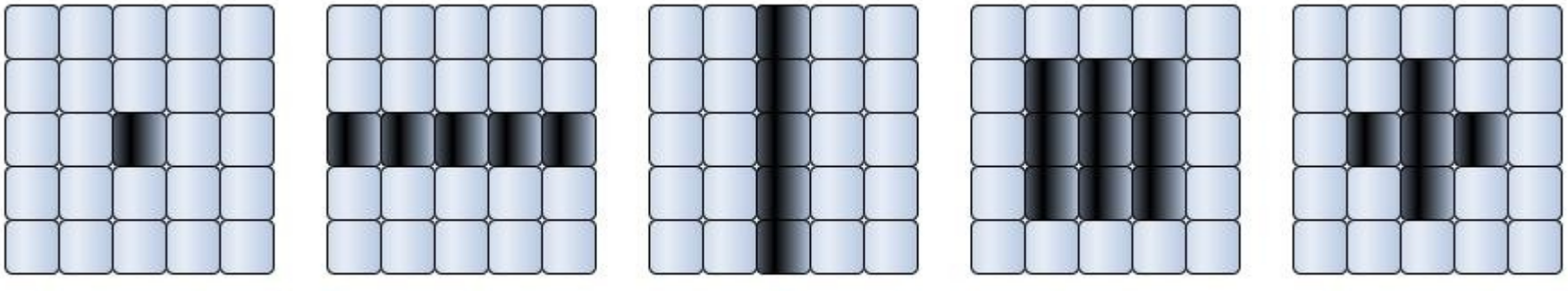


$$\vec{s} = \underbrace{\begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix} \begin{bmatrix} t_y \\ t_x \end{bmatrix}}_{\text{eMAP}} + \underbrace{\begin{bmatrix} iMAP_0 \\ iMAP_1 \end{bmatrix}}_{\text{iMAP}}$$

33 MAP Citizens

$$\vec{s} = \underbrace{\begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix} \begin{bmatrix} t_y \\ t_x \end{bmatrix}}_{\text{eMAP}} + \underbrace{\begin{bmatrix} iMAP_0 \\ iMAP_1 \end{bmatrix}}_{\text{iMAP}}$$

- eMAP
  - M00, M01, M10, M11∈{0,1}
- iMAP
  - Single, Row, Column, Block, Neighbor



Micro-Benchmarks

- We generate 2 micro-benchmarks (in OCL) for each MAP
- The kernel code
  - Local space allocation
  - Local data staging
  - Local data access (specified by the MAP)
- We provides a tool (Aristotle*) to facilitate this process

*https://github.com/haibo031031/aristotle

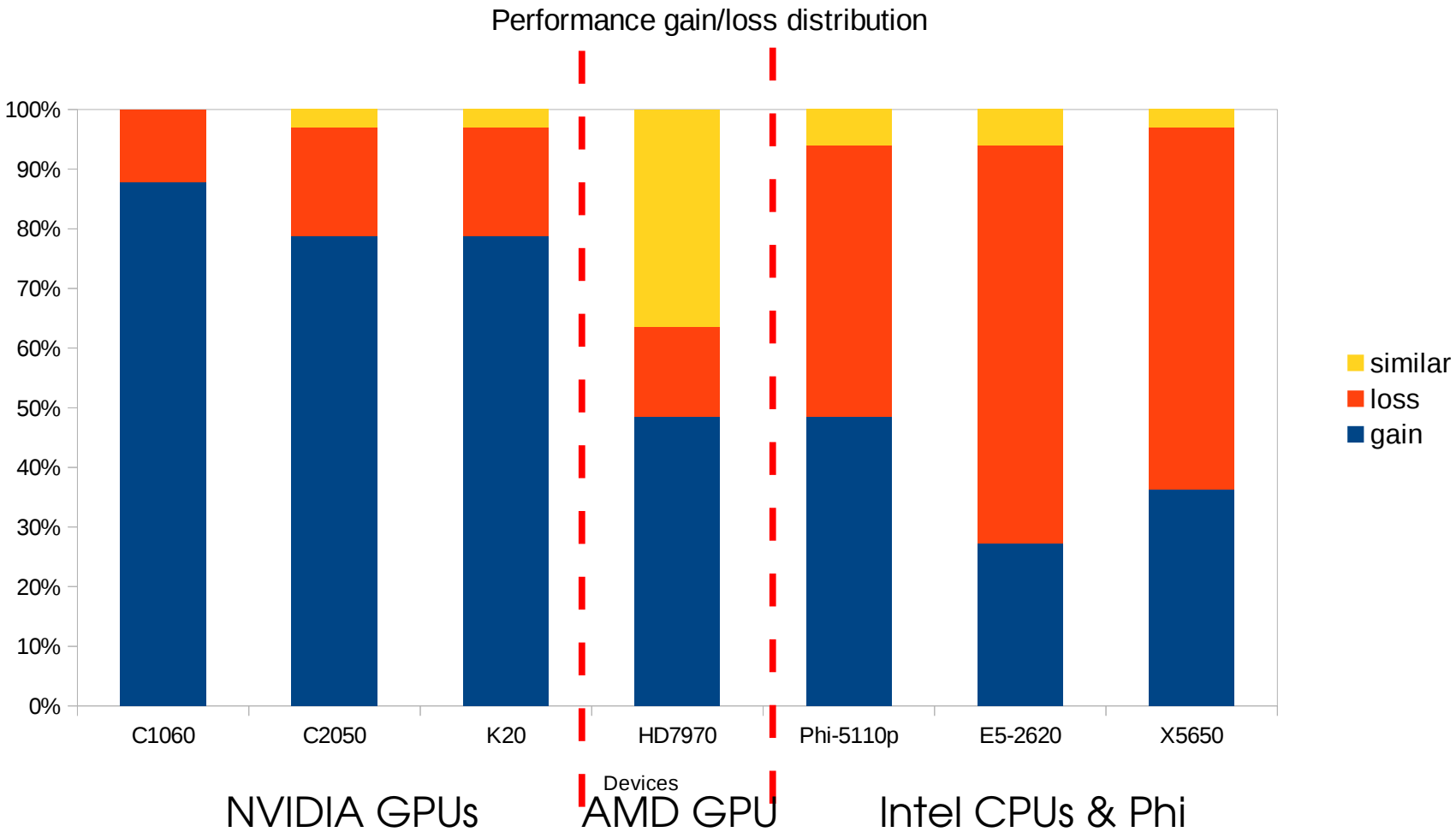
Experimental Platforms

- Devices
  - SPM-only: NVIDIA C1060
  - SPM+Cache: AMD HD7970, NVIDIA C2050, K20
  - Cache-only: Intel Xeon X5650. E5-2620, Intel Phi 5110P
- Software environments
  - AMD APP v2.8
  - Intel OpenCL SDK v3.0
  - NVIDIA CUDA v5.5 (not updated for long time )

Experimental Setup

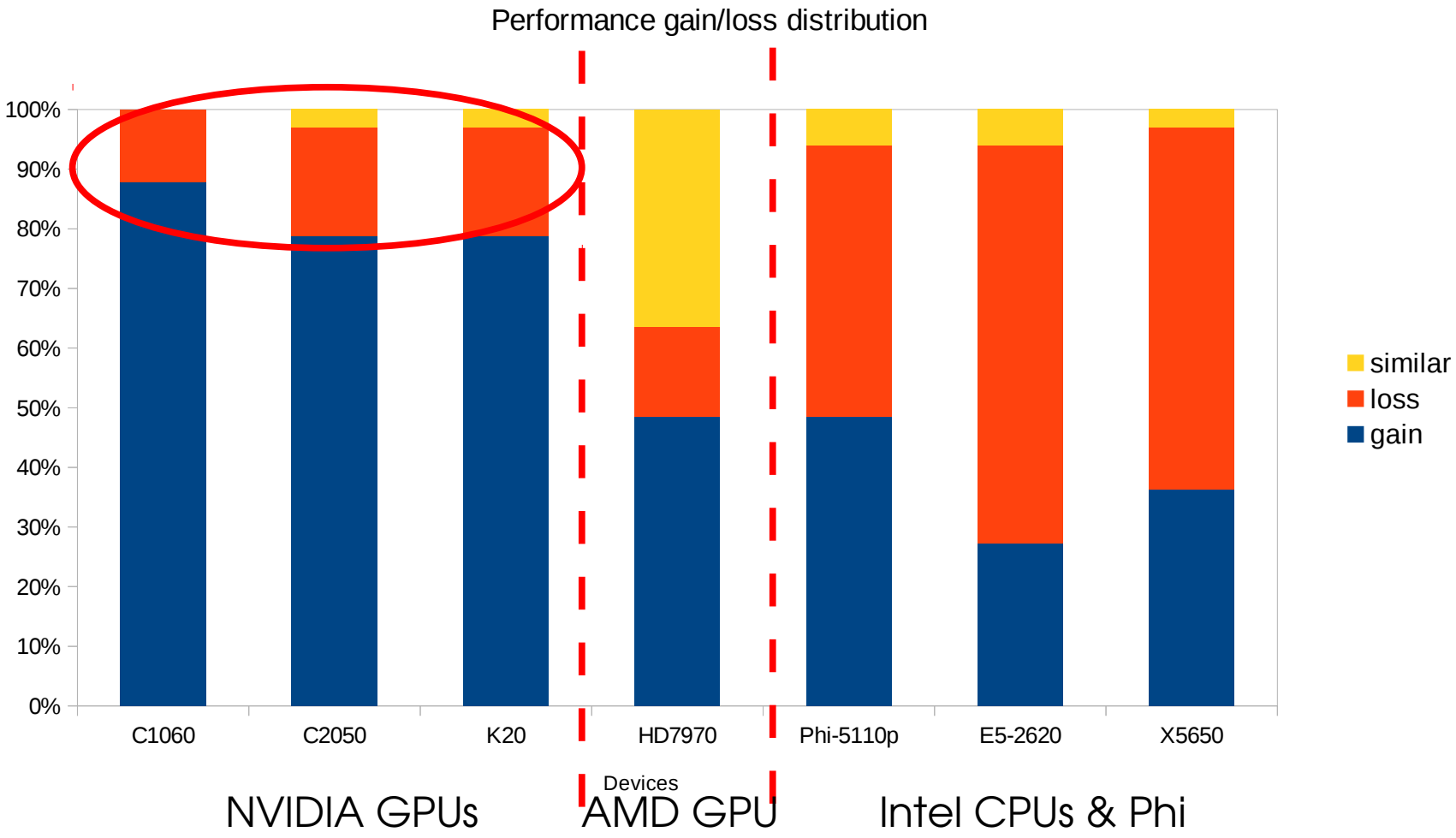
- Metrics: bandwidth
- Datasets
  - 128, 256, 512, 1024, 2048, 4096
  - Block MAPs: r=3
- Run each measurement for 21 iterations
  - 1 iteration to warm-up
  - Measure 20 iterations
  - Flush caches between iterations

A Bird's-eye View



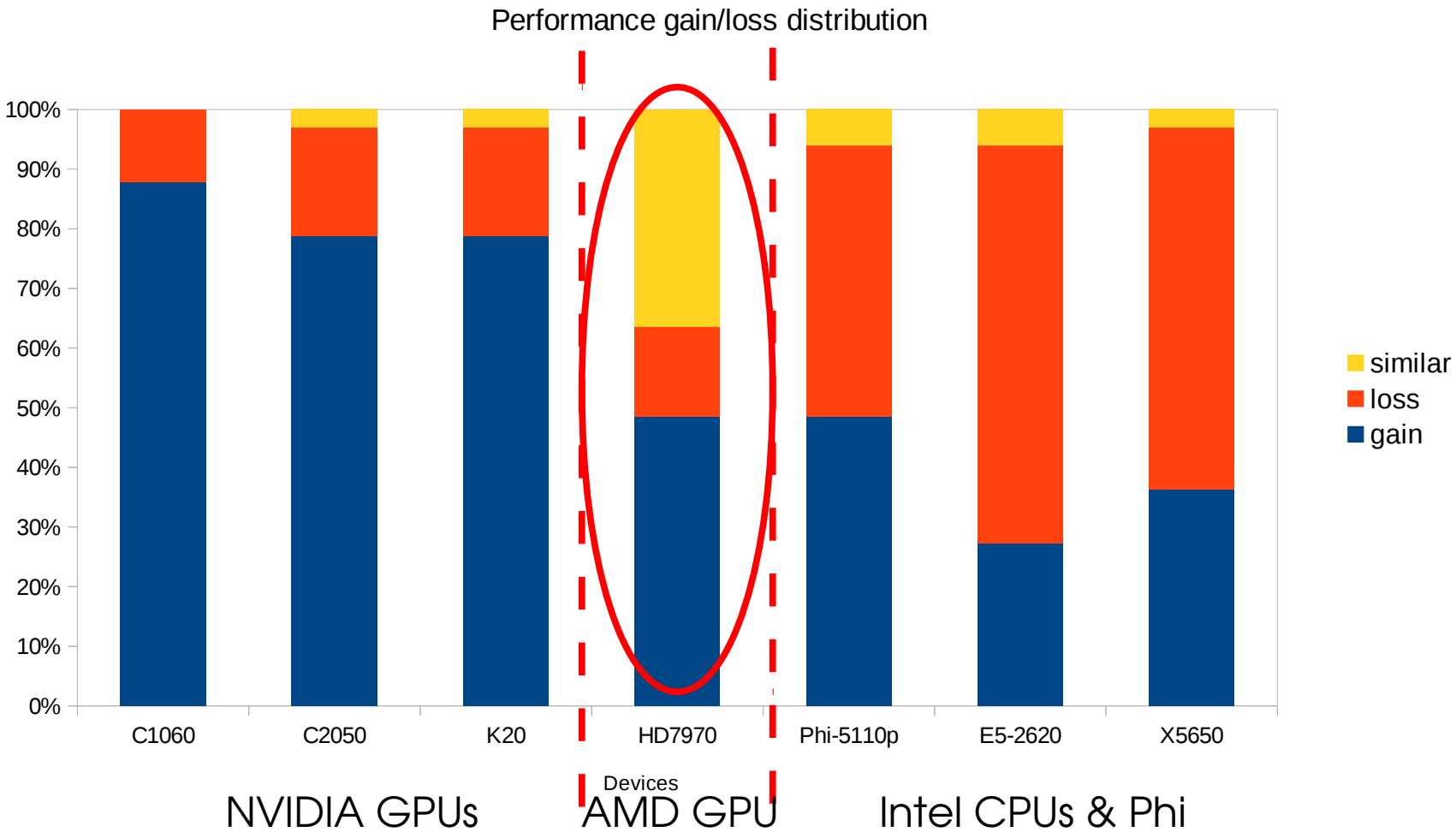
Datasets: 4096x4096

A Bird's-eye View



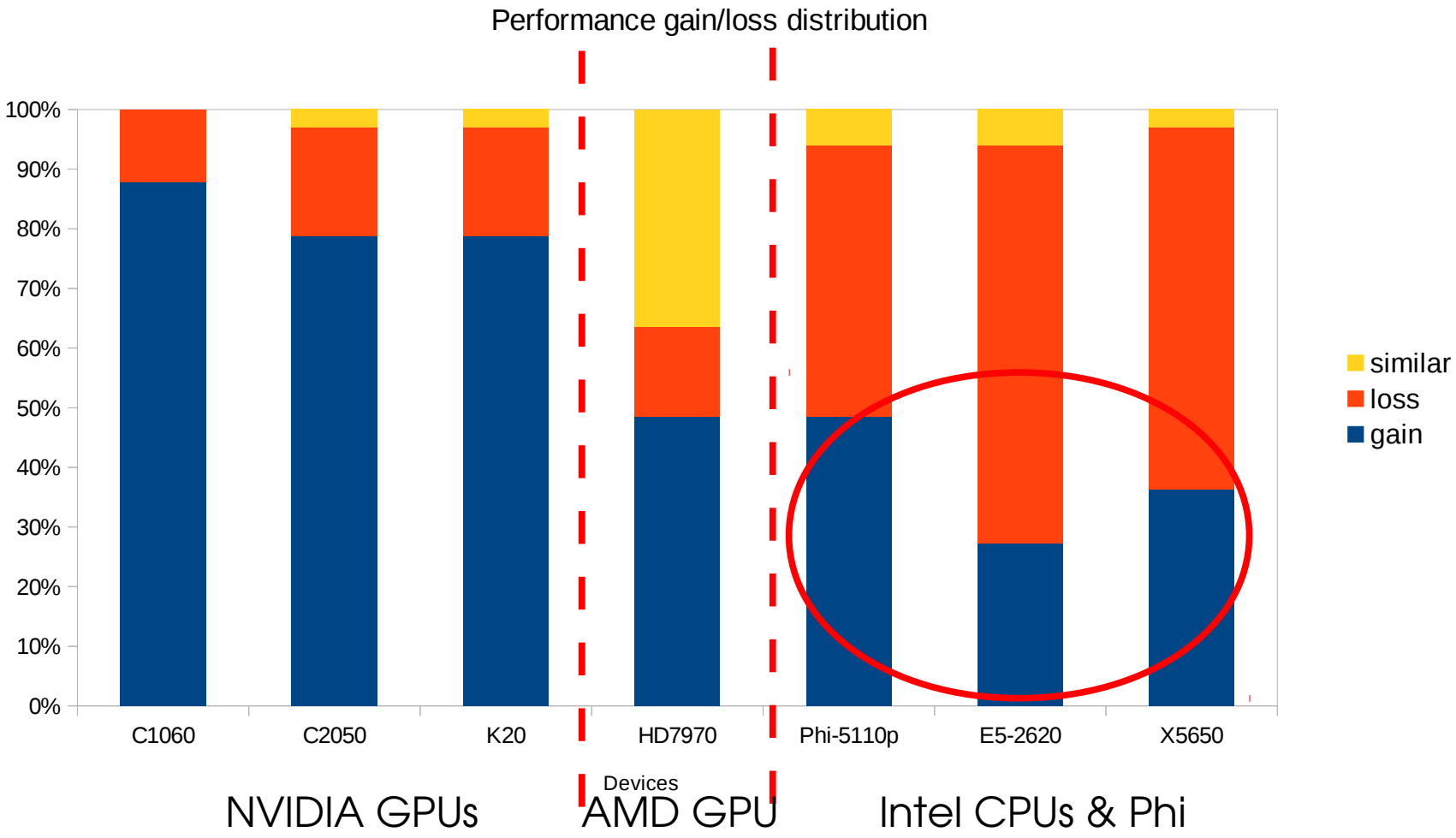
Datasets: 4096x4096

A Bird's-eye View



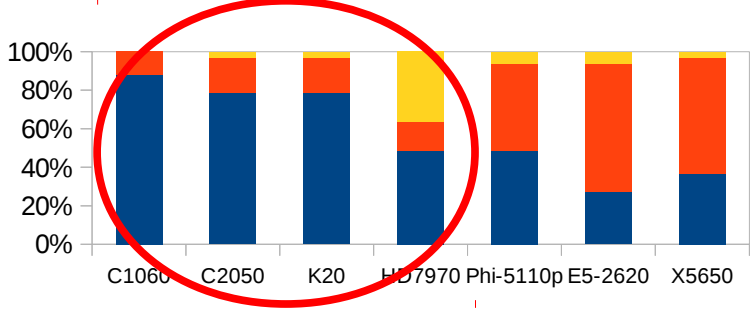
Datasets: 4096x4096

A Bird's-eye View

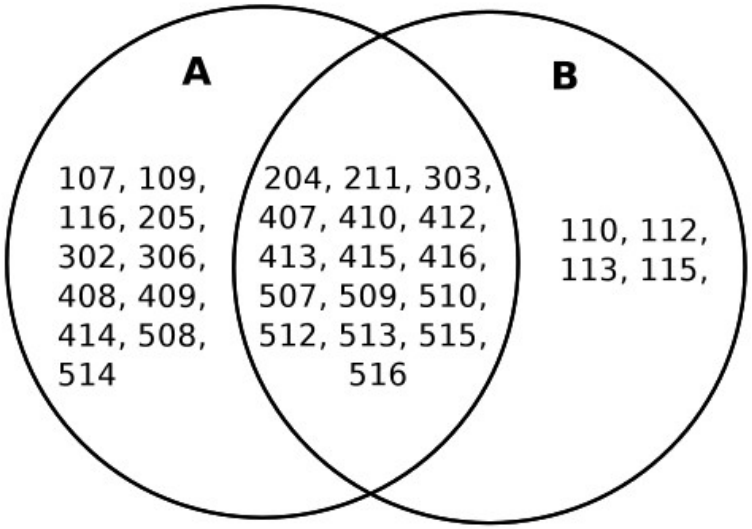


Datasets: 4096x4096

SPM Processors

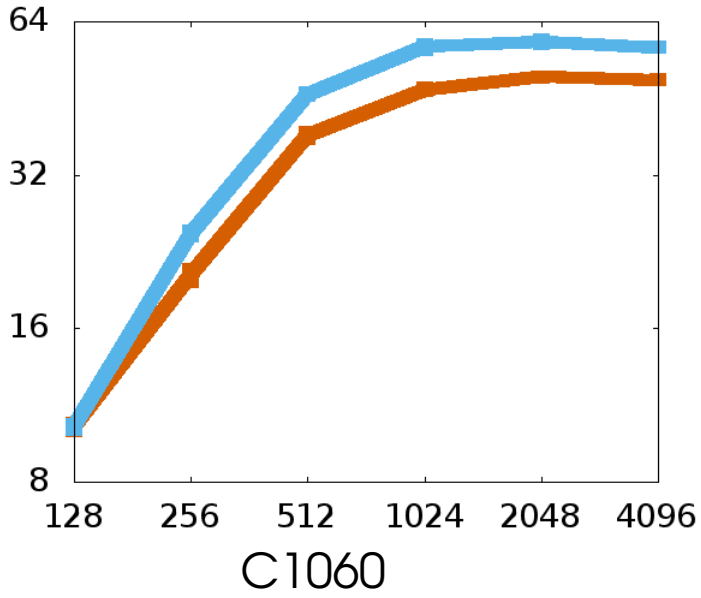


- Memory bandwidth increase factors
  - Data reuse (A)
  - Changed memory access orders (B)

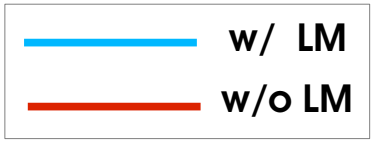


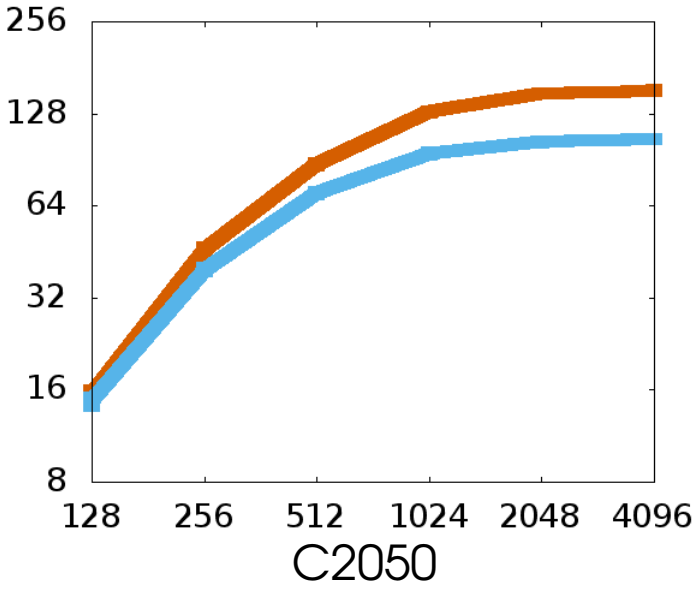
SPM Processors: w/o Caches VS. w/ Caches

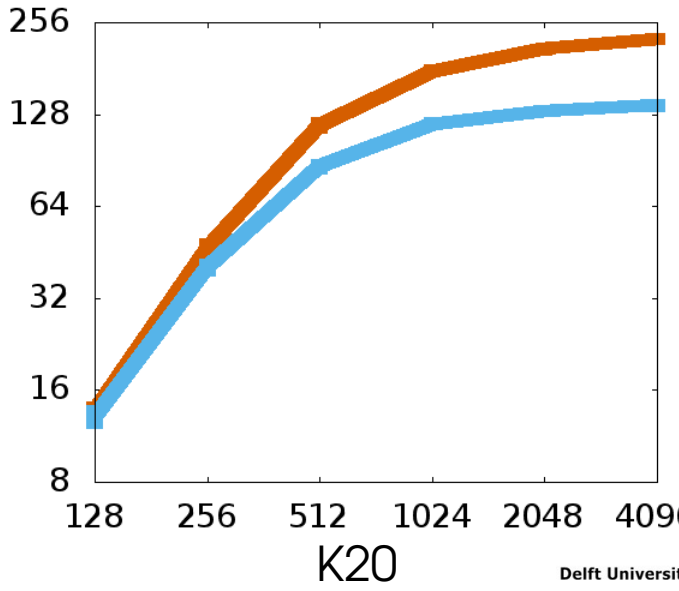
Better



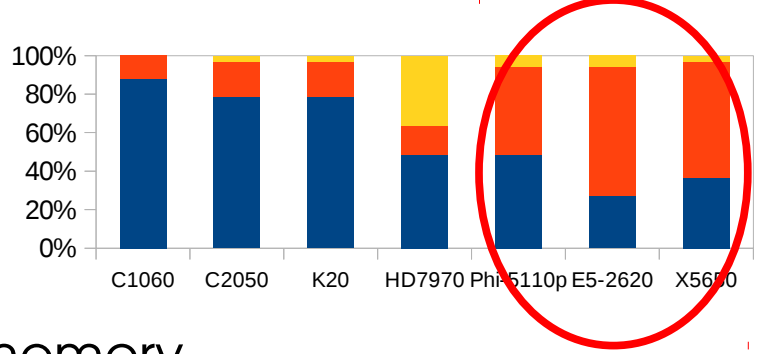
- The performance gain canceled
- Disabling local memory is better
  - MAP-514



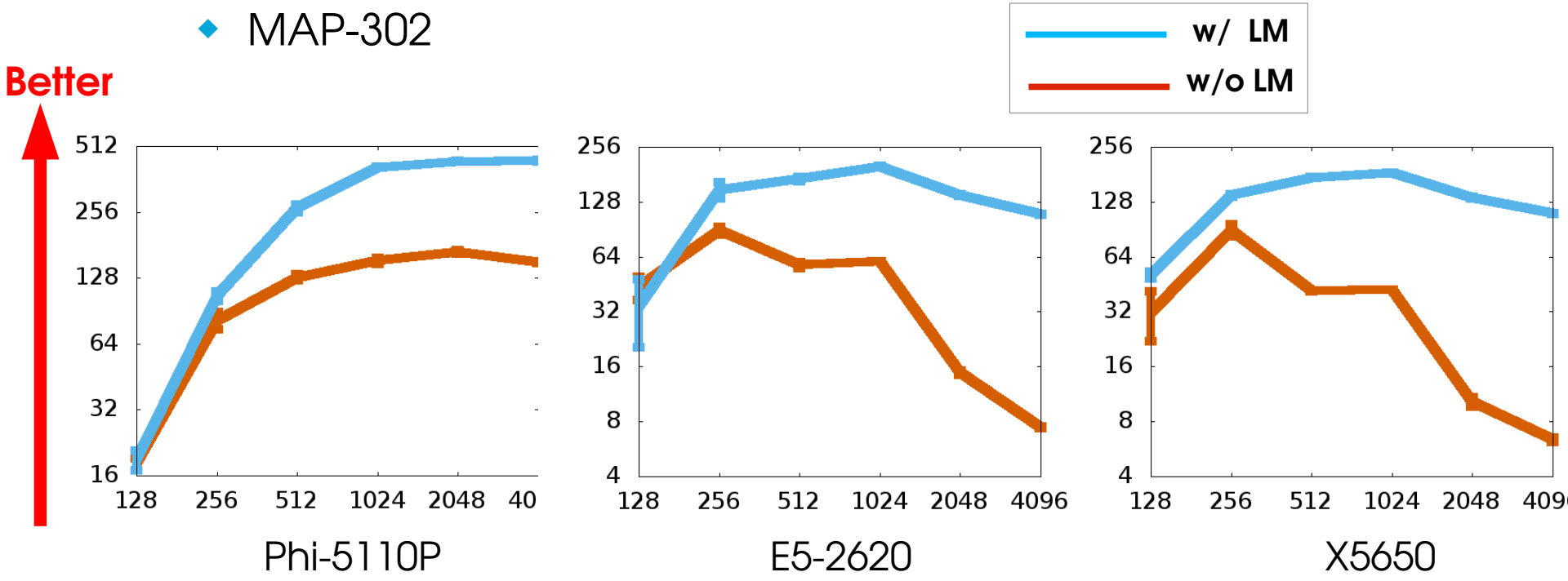




Cache-only Processors

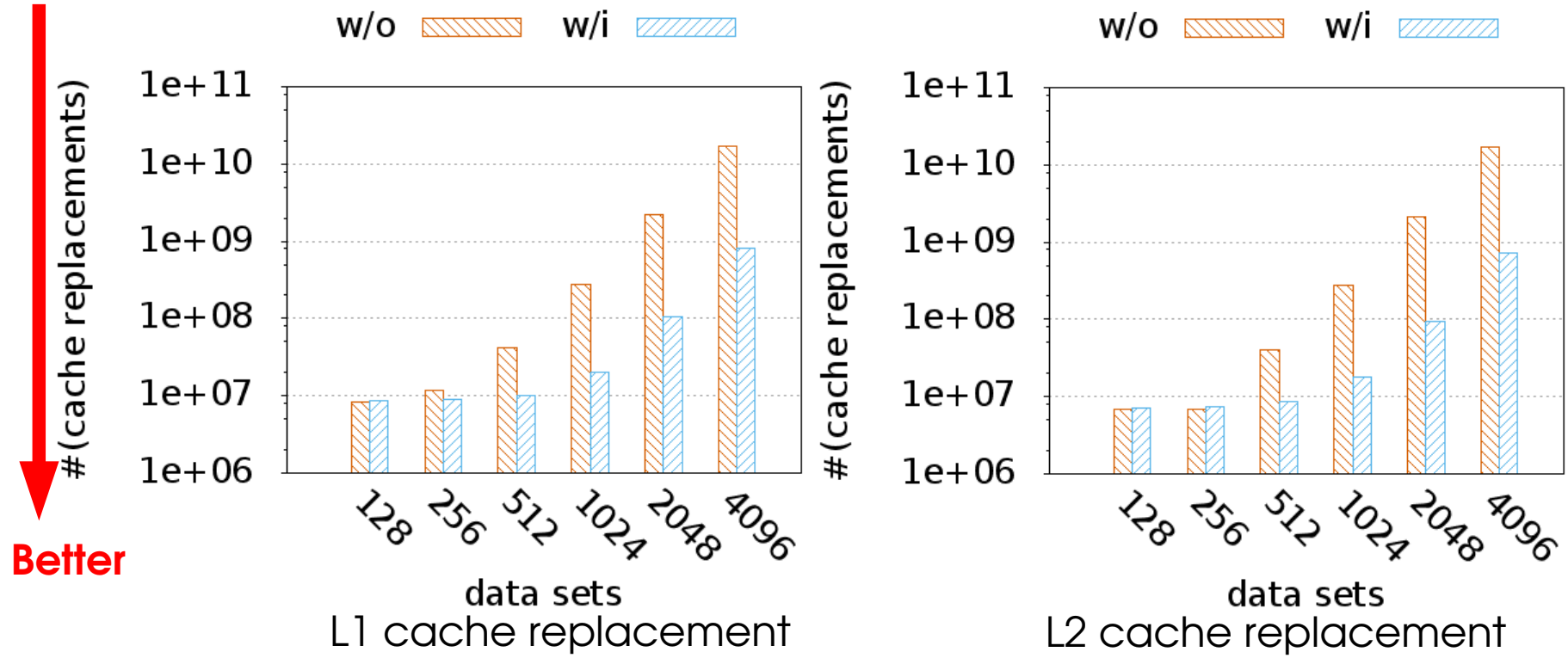


- Emulating local memory on global memory
- Using local memory might utilize caches better
  - MAP-302

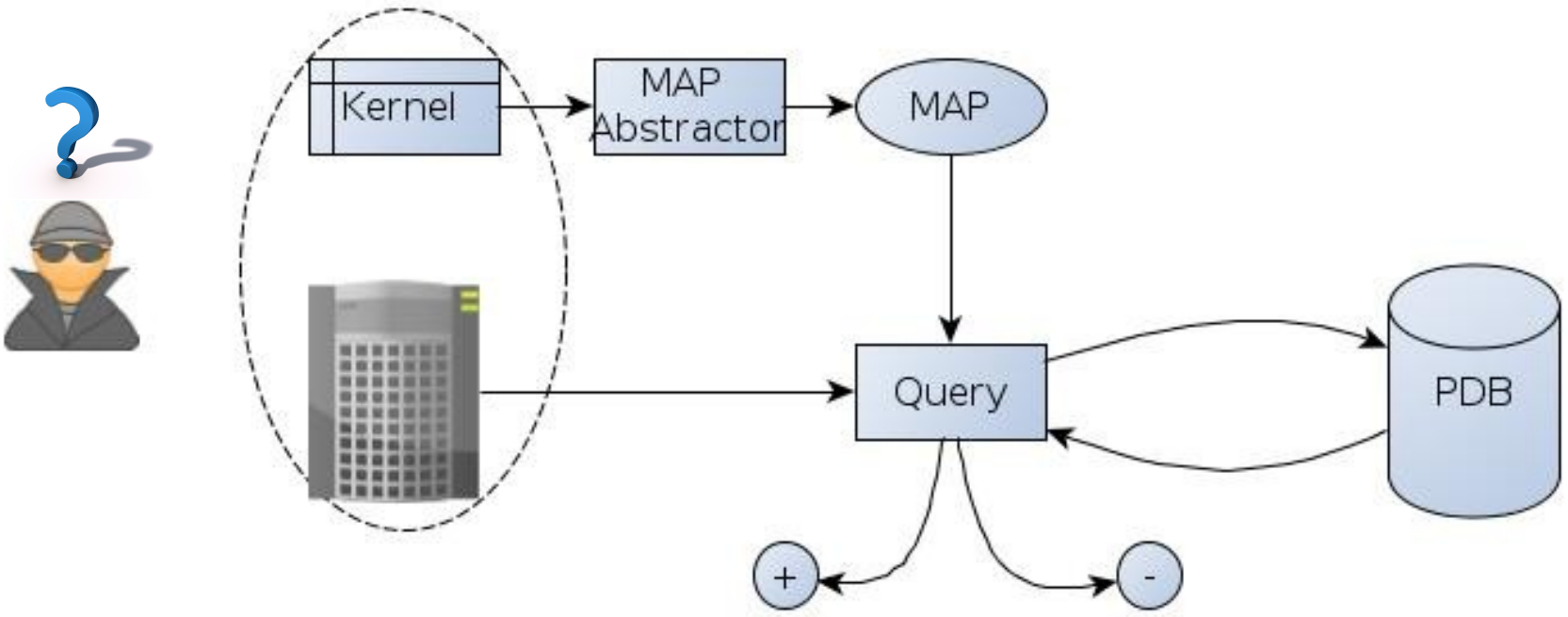


Cache-only Processors

- Using local memory on MAP-302 leads to a BW increase
- Profile the number of cache-line replacements on E5-2620



Performance Database Use-Scenario



Summary

- Data reuse and access order changes are positive factors
- Unpredictable local memory performance is due to caches
- Our query-based approach to decide local memory usage
- Architecture design indications
  - SPM and caches co-exist !?

Follow-up Questions

- Evaluations on more platforms, e.g., tiny GPUs
- How to identify MAPs for a given kernel?
  - Visual inspection
  - Automatic tools
- How to 'predict' the performance impact of using local memory in the presence of multiple MAPs?
- How to enable/disable local memory usage?
  - Enabler*: w/o → w/
  - Disabler**: remove the use of local memory

*J. Fang, et al., "ELMO: A User-Friendly API to enable local memory in OpenCL kernels," in PDP2013.
**J. Fang, et al., "Grover: Looking for Performance Improvement by Disabling Local Memory Usage in OpenCL Kernels," ICPP2014 (in submission).

Questions

Jianbin Fang
PhD student at TU Delft
j.fang@tudelft.nl

Jianbin Fang Ana Lucia Varbanescu Henk Sips