Quantifying the Performance Impacts of Using Local Memory for Many-Core Processors

Jianbin Fang[1], Ana Lucia Varbanescu[2], Henk Sips[1]

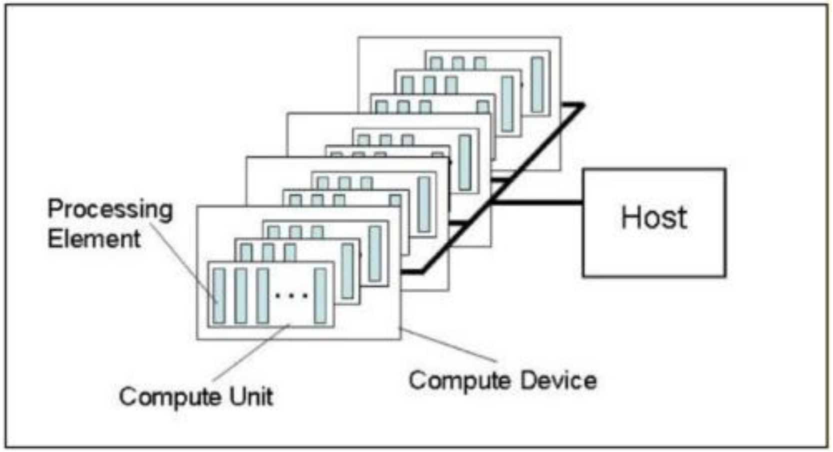
1 Delft University of Technology

2 University of Amsterdam

The Netherlands

Looking Back on OpenCL

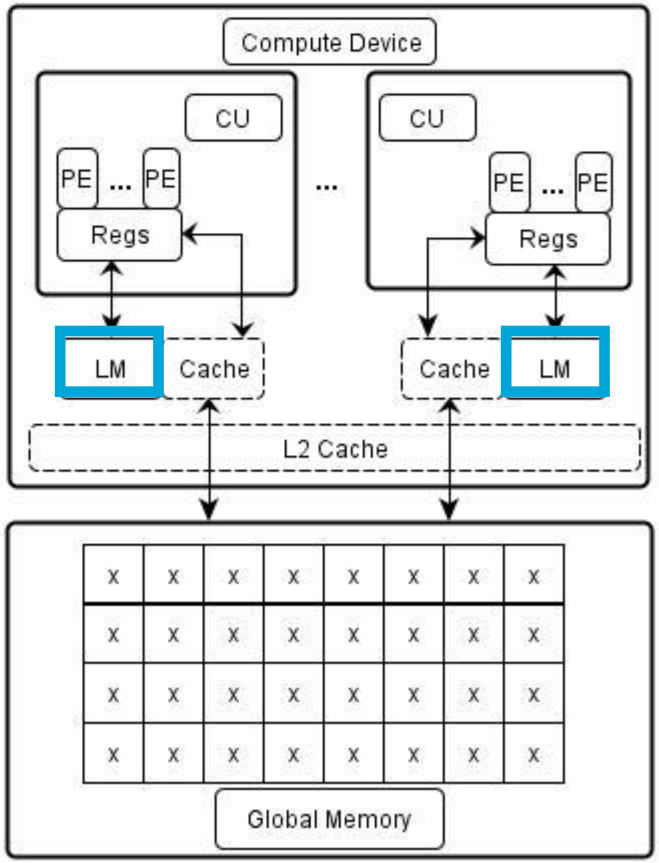
- OpenCL- Open Computing Language
  - ✓ An open programming framework by Khronos group
  - ✓ Heterogeneous platforms CPUs, GPU, MIC, FPGA, DSPs, …
- OpenCL platform model



- An OpenCL program
  - ✓ Kernel: a language based C99
  - ✓ Host: a set of APIs
- Adopted by many vendors
  - ✓ Current version: v2.0 (July 2013)

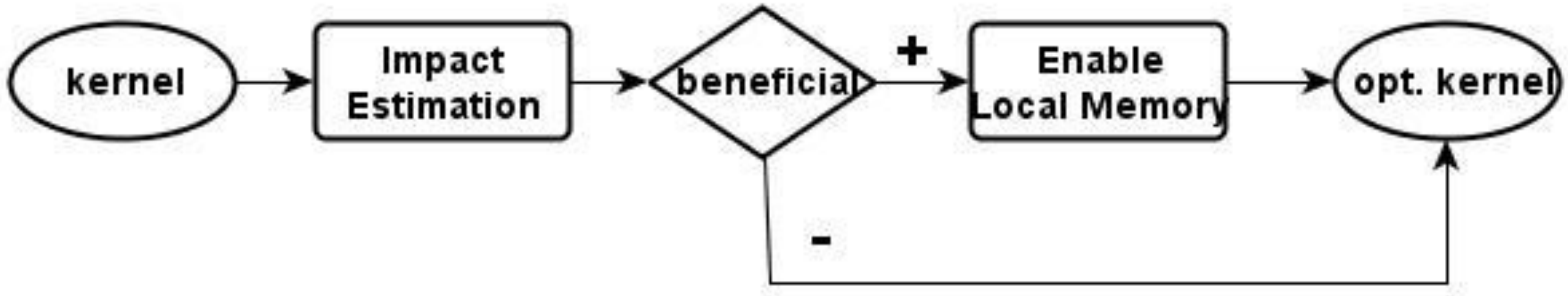
OpenCL and Local Memory

- Local memory is a key performance factor
  - ✓ FAST: On-chip
  - ✓ Not-a-Cache: User-managed
- Current status: using local memory is a trial-and-error process
  - ✓ Work hard to enable it …
  - ✓ and hope for performance gain.



Our Idea

- Performance impact estimation



- How can we estimate the benefits of using local memory?
  - ✓ Assess the necessity of using local memory
  - ✓ Facilitate performance modeling of OpenCL platforms

Local Memory "Myths"

- Local memory assumptions for performance gain:
  - ✓ Data sharing is mandatory
  - ✓ Using LM on GPUs is mandatory
  - ✓ Using LM on CPUs must be avoided

- We contradict these myths!
  - ✓ Data reuse is not equivalent with LM performance gain
  - ✓ Enabling LM on GPUs can be skipped
  - ✓ Enabling LM on CPUs can be beneficial

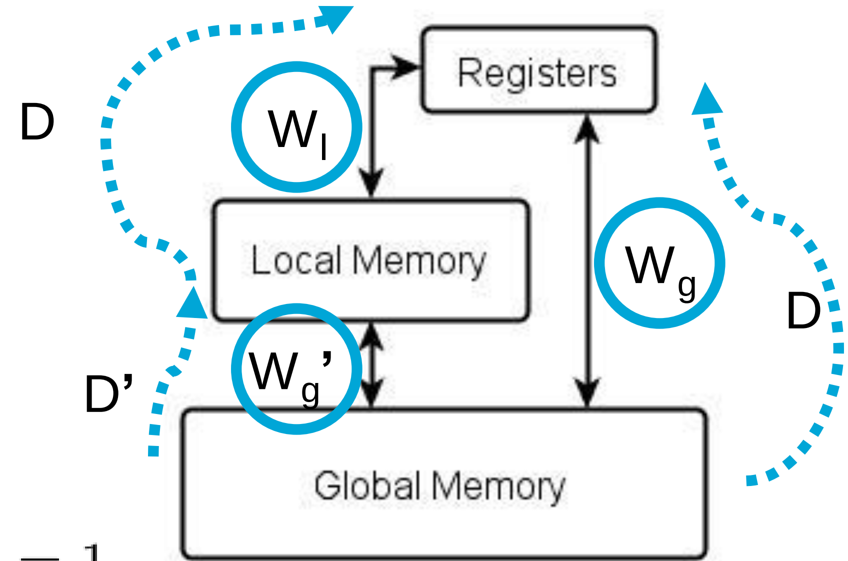
Data reuse ≠ Performance gain

- NBody on GTX580
  - ✓ Threads share exactly the same data set
- Results (in GB/s)

	64x64	128x128	256x256	512x512	1024x1024
$LM_{w/o}$	613.50	636.43	646.06	616.42	589.95
$LM_{w/i}$	512.44	495.28	516.04	518.61	520.64
Loss(%)	16.47	22.18	20.13	15.87	11.75

- Conclusion
  - ✓ Using local memory performs worse by 18% on average

No data reuse ≠ Performance loss



- Describe analysis

$$T_{w/o} = \frac{D}{W_g},$$

$$T_{w/i} = \begin{cases} \frac{D}{W_l} + \frac{D'}{W'_g} & N = 1 \\ max(\frac{D}{W_l}, \frac{D'}{W'_g}) = \frac{D'}{W'_g} & N > 1 \end{cases}$$

- Conclusion
  - ✓ Besides data reuse (D↓), access order change matters (W↑)!
  - ✓ Matrix transpose is a good example.

LM on CPUs ≠ Performance loss

- Image convolution on CPU
  - ✓ Intel Xeon E5620 (6 cores)
  - ✓ Filter radius is 3
- Results (in GB/s)

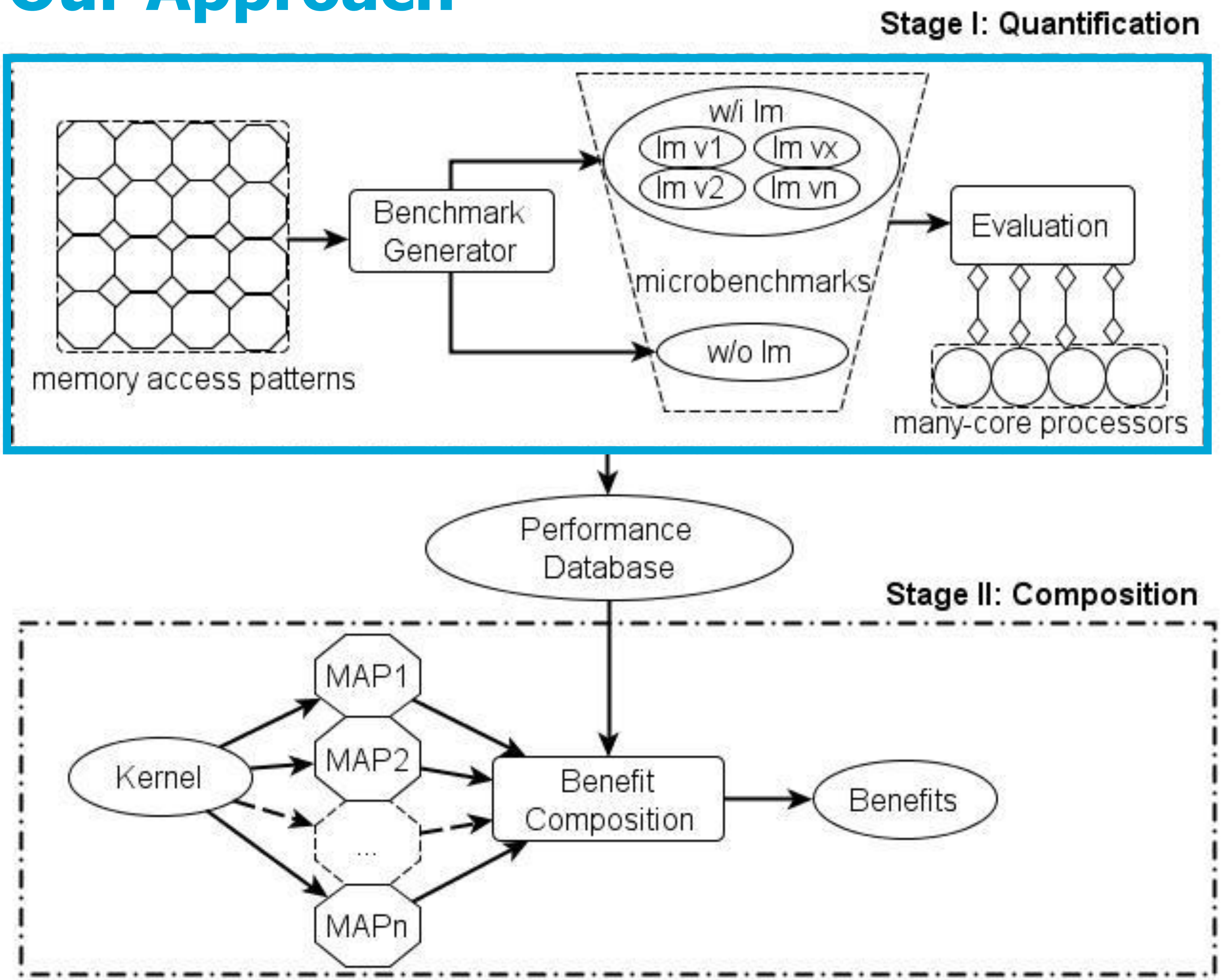
	64x64	128x128	256x256	512x512	1024x1024	2048x2048
$LM_{w/o}$	6.81	7.77	7.81	8.06	8.13	8.15
$LM_{w/i}$	12.23	13.81	14.08	14.56	14.70	14.56

- Conclusion
  - ✓ Using local memory delivers (2x) better performance

Performance Impact Estimation

- Not an easy job
  - ✓ No assumptions hold for all cases
  - ✓ Application-dependent
  - ✓ Platform-dependent
- Our approach:
  1. Enumerate and analyze all feasible memory access patterns
  2. Quantify and log local memory impacts for each MAP on each platform (in terms of bandwidth)
  3. Model applications as (compositions of) MAPs
  4. Quantify application's gain by search and compose

Our Approach



Stage I: Quantification

MAP Description:

MAP=eMAP+iMAP

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

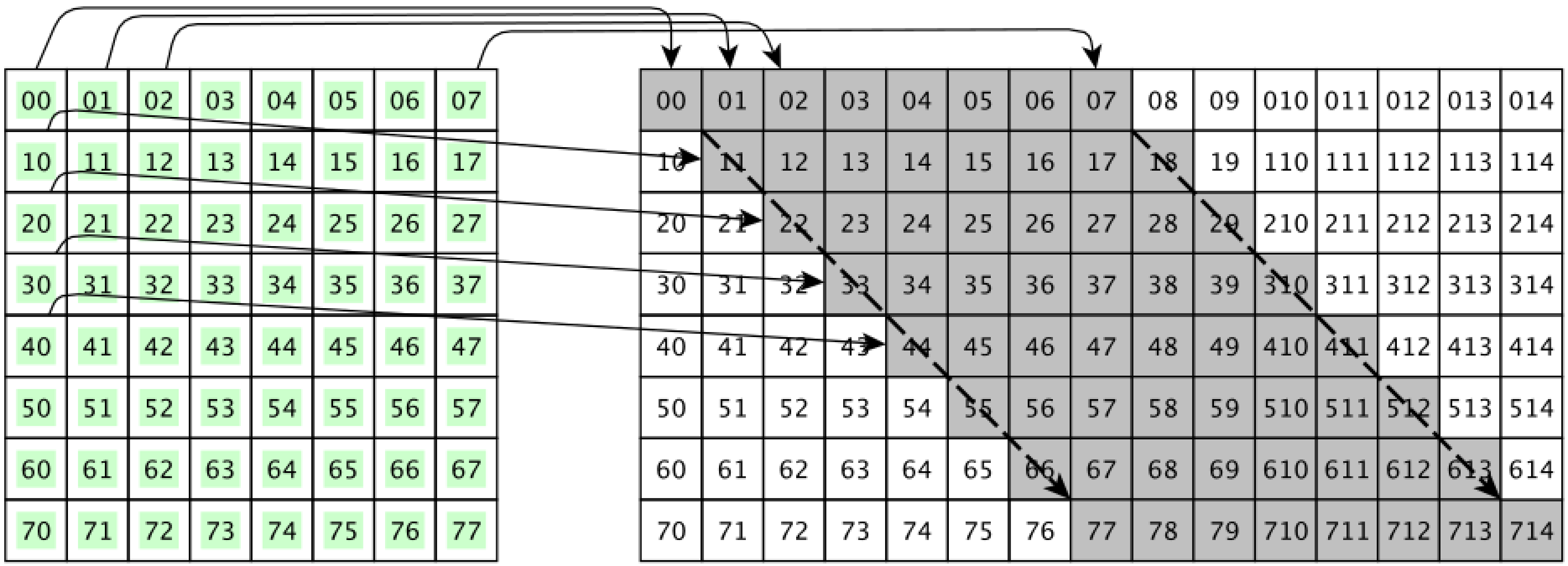
16 cases

$$M_{00}, M_{01}, M_{10}, M_{11} \in \{0, 1\}$$

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} t_y \\ t_x \end{bmatrix}$$

eMAP-14

Stage I: Quantification



Work-group: 8 x 8 work-items

Data set: 15 x 8

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} t_y \\ t_x \end{bmatrix} \rightarrow \vec{s} = \begin{cases} d_y = t_y \\ d_x = t_y + t_x \end{cases}$$

eMAP-14

Stage I: Quantification

34 patterns

MAP Description:

MAP=eMAP+iMAP

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

16 cases

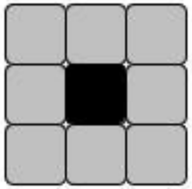
$$M_{00}, M_{01}, M_{10}, M_{11} \in \{0, 1\}$$

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} t_y \\ t_x \end{bmatrix}$$

MAP-14

5 cases

Single, Row, Column, Block, Neighbor



Block (4)

Stage I: Quantification

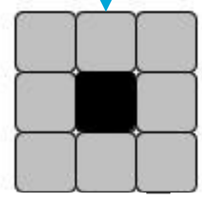
Generating Benchmarks (MAP-407)

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

eMAP-07

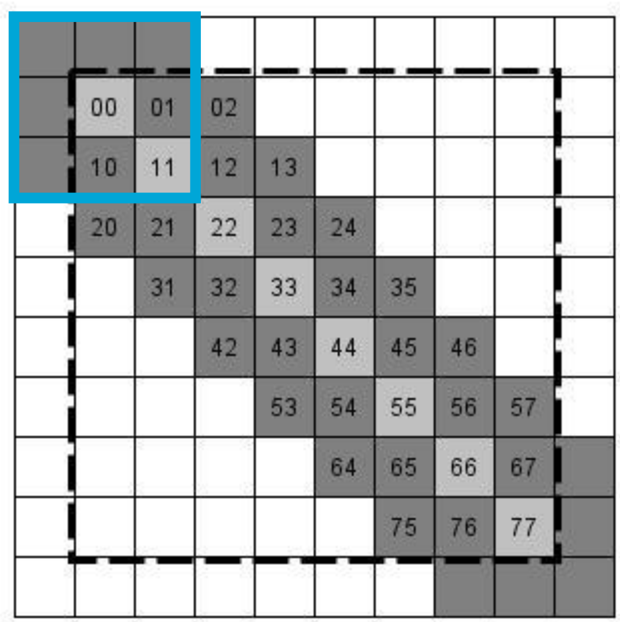
$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_y \\ t_x \end{bmatrix} \rightarrow \vec{s} = \begin{cases} d_y = t_x \\ d_x = t_x \end{cases}$$

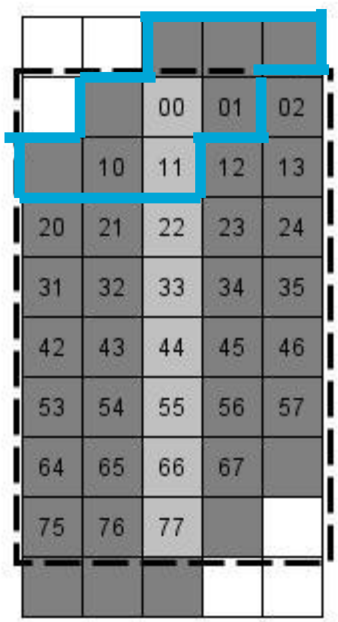
Block (4), r=1



Max vs. Min:

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

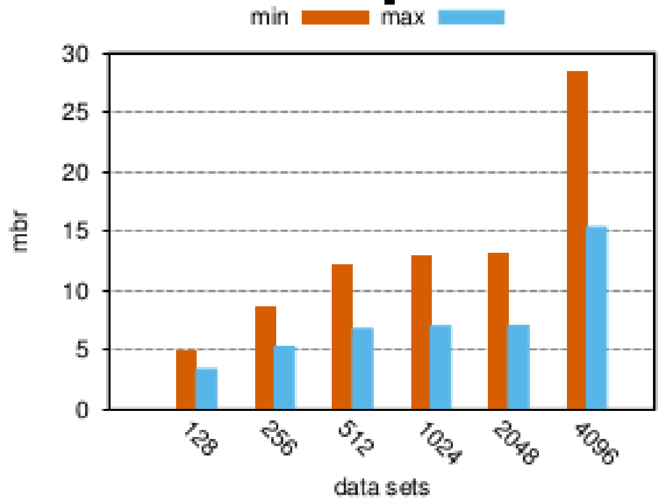




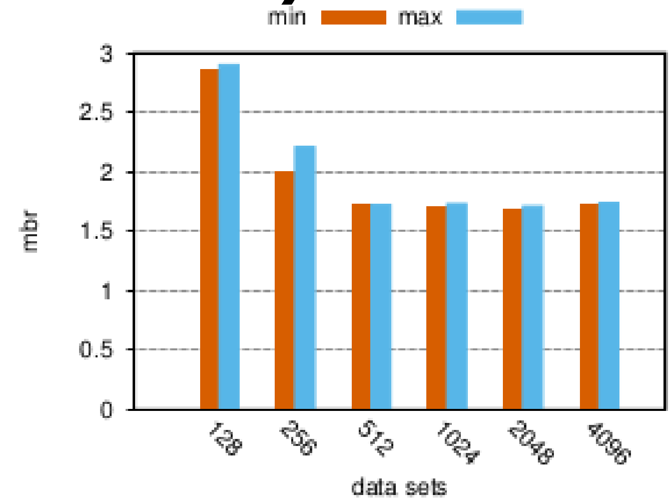
Stage I: Quantification
Min/Max Comparison (MAP-407)

mbr=B/b

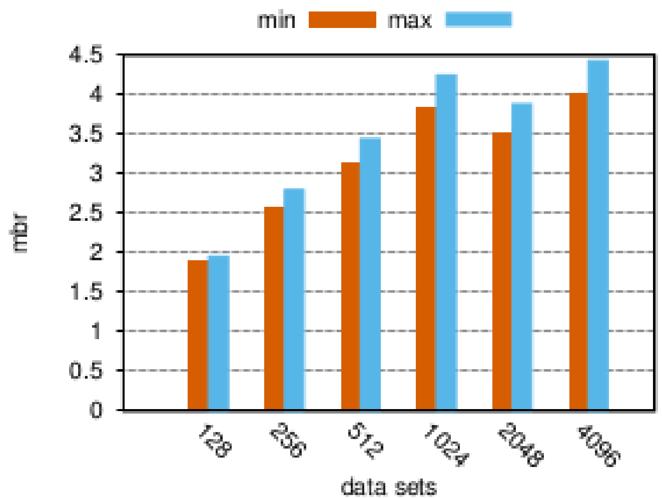
better



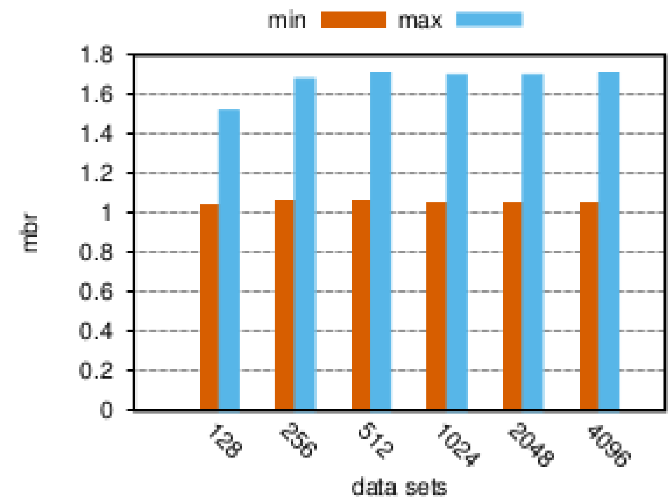
(a) GTX280



(b) HD6970



(c) GTX580

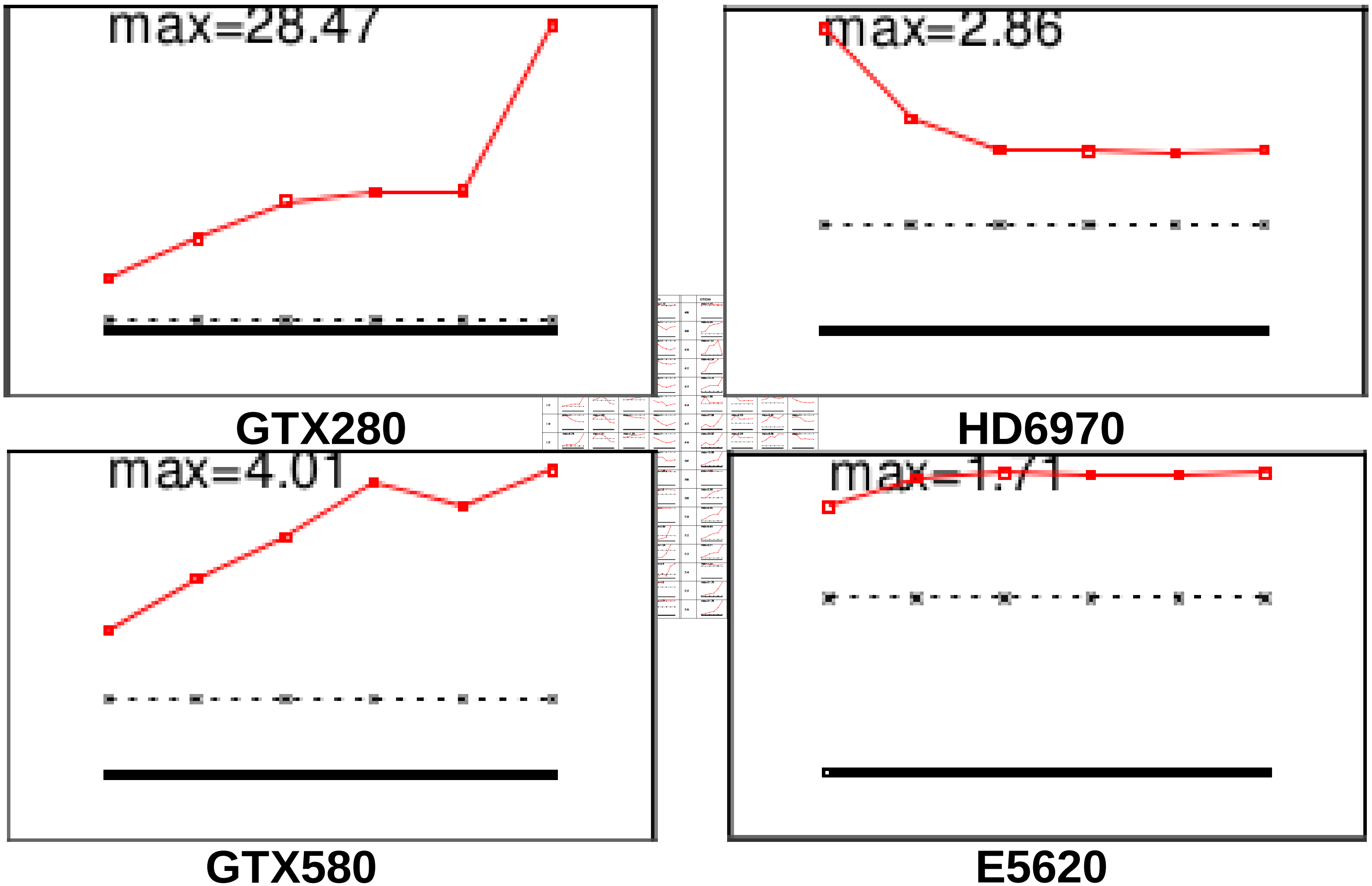


(d) E5620

Stage I: Quantification

Performance Database Overview

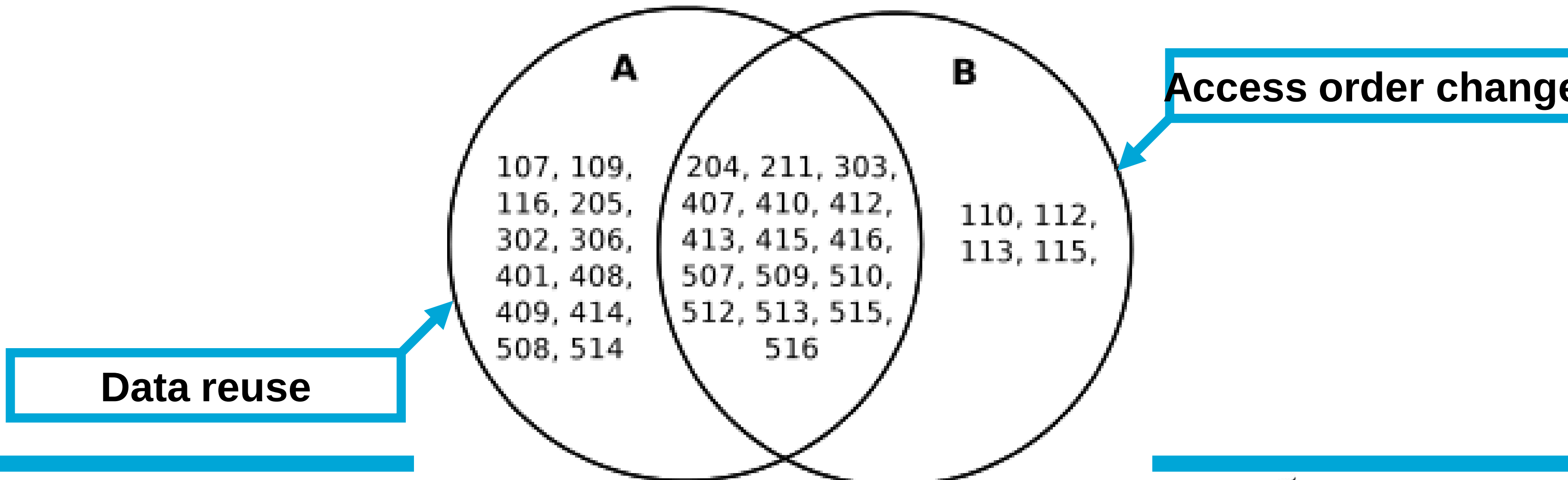
Performance Database (MAP-407)



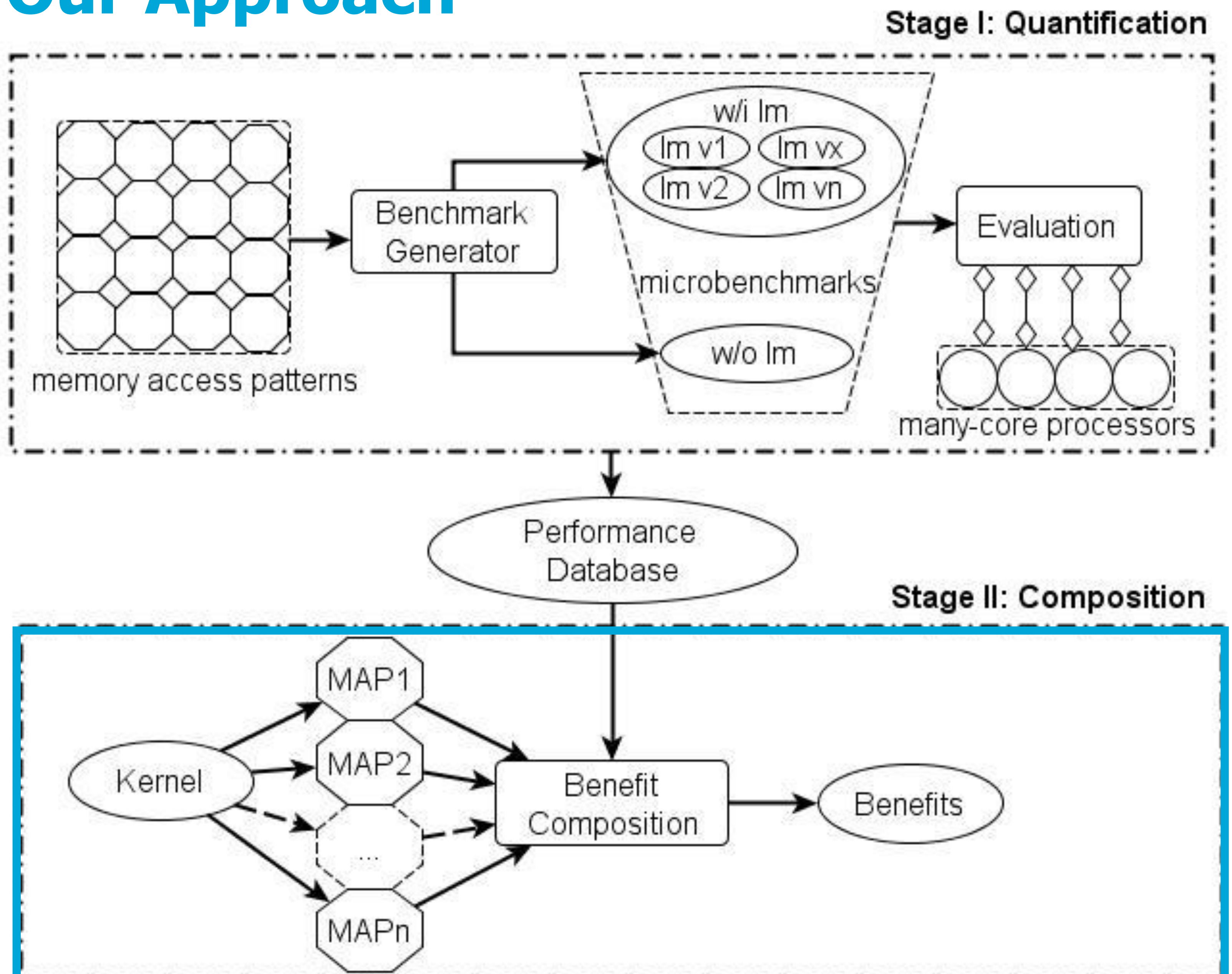
Stage I: Quantification

Performance Database Summary

	GTX280	HD6970	GTX580	E5620
Gain Always	30	23	29	4
Loss Always	2	0	4	25
Varied	2	11	1	5



Our Approach



Stage II: A Query-based Performance Prediction

- Kernel performance gain due to LM = memory bandwidth ratio before (b) and after (B) using LM

$$mbr_p = \frac{B_1 \bigotimes B_2 \bigotimes B_3 \bigotimes \ldots \bigotimes B_m}{b_1 \bigoplus b_2 \bigoplus b_3 \bigoplus \ldots \bigoplus b_m}$$

- Predicting bandwidth when using LM
  - ✓ Identify MAPs (manually)
  - ✓ Query bandwidth information (B, b) from DB
  - ✓ Compose the bandwidths of individual MAPs
- IC, MM, MT, SOR on GTX580

Stage II: A Query-based Performance Prediction

- Case I: MT, SOR
  - ✓ The kernel has one input matrix (and MAP)
  - ✓ Use the corresponding mbr in DB
- Case II: MM

$$mbr_p = \frac{(n_A \times B_A + n_B \times B_B)/(n_A + n_B)}{min(b_A, b_B)}$$

- Case III: IC
  - ✓ Assume the filter is small and allocated on on-chip memory
  - ✓ Use mbr of MAP-408

Stage II: A Query-based Performance Prediction

	MT	SOR	MM	IC
MAPs	(110)	(508)	(205,302)	(408)
mbr_p	1.18	0.96	2.40	1.83
T (ms)	10.63	14.06	1897.95	145.56
T_p^{LM}(ms)	9.01	14.64	790.81	79.54
T_m^{LM}(ms)	9.12	14.21	766.11	94.60
Accuracy(%)	1.26	3.07	3.22	15.91

Conclusion

- Quantifying the performance impact of using local memory on many-cores is possible
  - ✓ Not easy expected => well-known assumptions don't always hold
  - ✓ MAP-based => application-agnostic
  - ✓ Query-based => prediction-friendly
  - ✓ Database-based => easy to extend
  - ✓ Composition-based => applicable for fairly complex kernels

On-going Work

- More MAPs and tests (on more diverse platforms, e.g. MIC)
- Investigate further the performance interference between MAPs
- An auto-tuner to automatically enable local memory

Questions

Jianbin Fang

PhD student at TU Delft

Email: j.fang@tudelft.nl

WWW: http://www.pds.ewi.tudelft.nl/fang/