Benchmarking SpMV on Many-Core Architecture

Biwei Xie and **Zhen Jia**

Institute of Computing Technology
Chinese Academy of Sciences

Princeton University

Why Benchmarking?

"To Measure Is To Know"

-- William Thomson (Lord Kelvin)

We have an implementation

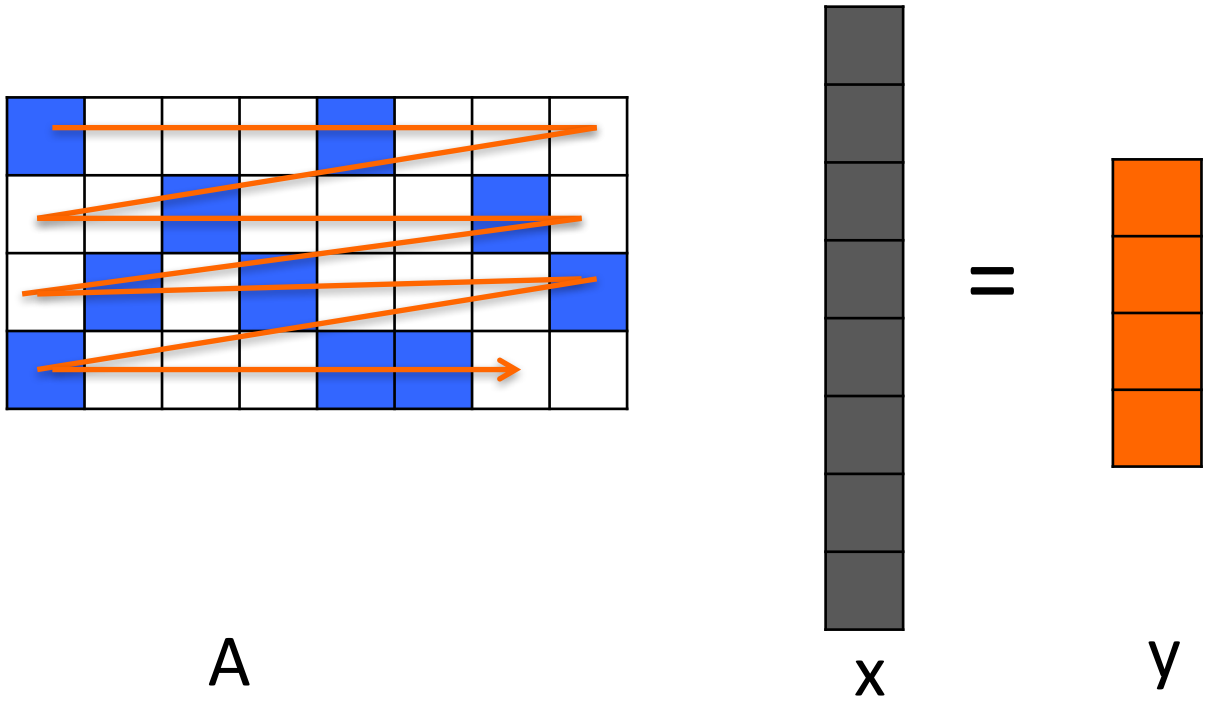
But, what if …

Why SpMV?

SpMV: Sparse matrix-vector multiplication

Given a *sparse* m × n matrix A and an dense n × 1 vector x:

$$y = A \bullet x:$$



SpMV Application Scenarios

- HPC
  - Various linear algebra algorithms: CG (Conjugate Gradients) …
- Graph Computing
  - Page Rank, BFS, etc.

Google twitter amazon

- CNN
  - Convolution

Factors important to SpMV performance

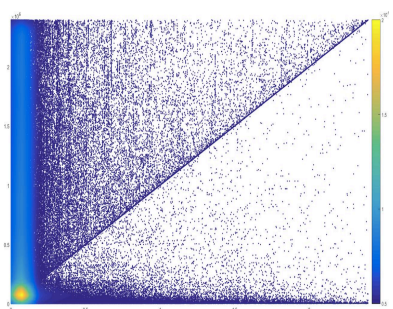
- Characteristics of the matrix
  - Data scale
  - Sparsity
  - Sparse Pattern
- SpMV Methods
  - CSR, CSC, DIA, COO, CSR5, CVR, ELL, ESB, Merge … (dozens)
  - Parallelization, Vectorization, Blocking …
- Platform
  - X86, ARM, GPU …

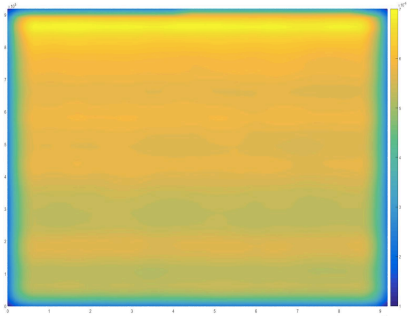
How to locate the best SpMV method for a given sparse matrix on a specific architecture?

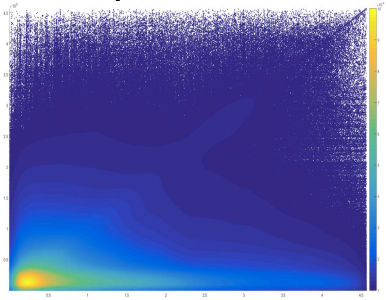
Benchmarking Methodology

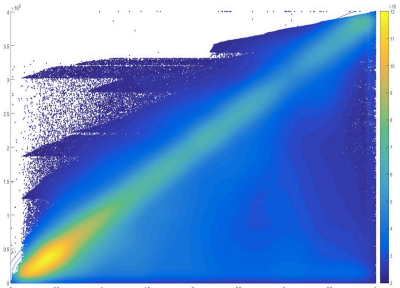
Sparse Matrix (Data Set)

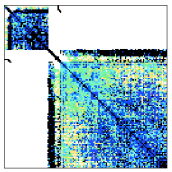
- Sparse Matrix
  - 1,500+ sparse matrices from UFL (discard small matrices)
  - Various sparse patterns and data scales

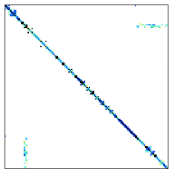

Wikipedia -Talk

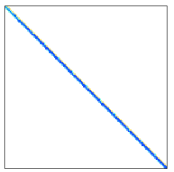

Web-Google

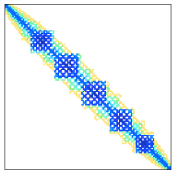

higgs-Twitter

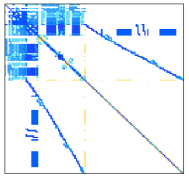

Amazon-0312

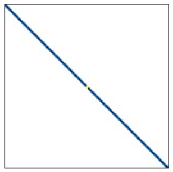

ASIC-100k


Wind Tunnel


Economics

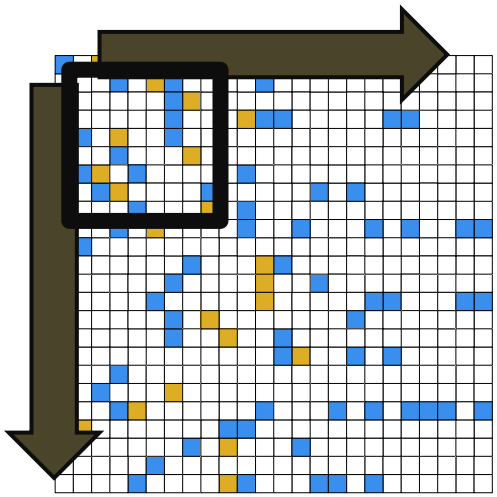

Ga41As41H72


Circuit5M


Cantilever

SpMV Methods

- SpMV algorithms:
  - 27 SpMV implementations.
  - From high-quality research (ICS, SC, CGO …)
  - From commercial/open-source packages
  - Widely used (CSR, COO …)

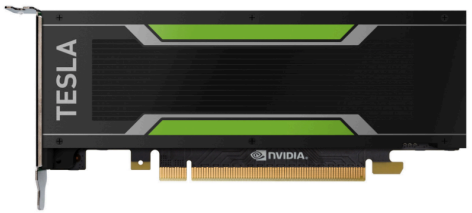


CSR: By rows
CSC: By columns
BSR: By blocks
DIA: By diagonals
CVR: By multi-rows
ELL: By blocks & rows
… …

ID	SpMV Methods	Input Formats	Package	Platform	OpenSource	Short Name
1	COO	COO	MKL-intel [17]	X86	No	
2	CSR	CSR	MKL-intel	X86	No	
3	CSC	CSC	MKL-intel	X86	No	
4	DIA	DIA	MKL-intel	X86	No	
5	IE	CSR	MKL-intel	X86	No	
6	BSR	BSR [18]	MKL-intel	X86	No	
7	ESB Dynamic	ESB [19]	MKL-intel	X86	No	ESB-d
8	ESB Static	ESB [19]	MKL-intel	X86	No	ESB-s
9	CVR	CVR	CGO'18 [20]	X86	Yes	
10	CSR5	CSR5	ICS'15 [21]	X86	Yes	
11	VHCC	VHCC	CGO'15 [22]	X86	Yes	
12	CSRcuSparse	CSR	cuSparse [23]	GPGPU	No	CSRcu
13	BSRcuSparse	BSR	cuSparse	GPGPU	No	BSRcu
14	HYBcuSparse	HYB	cuSparse	GPGPU	No	COOcu
15	COOCUSP	COO	CUSP	GPGPU	Yes	COOCU
16	CSRCUSP	CSR	CUSP	GPGPU	Yes	CSRCU
17	ELLCUSP	ELLPACK [24]	CUSP	GPGPU	Yes	ELLCU
18	HYBCUSP	HYB	CUSP	GPGPU	Yes	HYBCU
19	DIACUSP	DIA	CUSP	GPGPU	Yes	DIACU
20	CSRMagma	CSR	Magma [25]	GPGPU	Yes	CSRMA
21	BSRMagma	BSR	Magma	GPGPU	Yes	BSRMA
22	ELLMagma	ELLPACK	Magma	GPGPU	Yes	ELLMA
23	SELLPMagma	SELLP	Magma	GPGPU	Yes	SELLP
24	ELLPKMagma	ELLPacket	Magma	GPGPU	Yes	ELLPK
25	ELLRTMagma	ELLRT	Magma	GPGPU	Yes	ELLRT
26	Merge	CSR	SC'17 [26]	GPGPU	Yes	
27	CSR5GPU	CSR5	ICS'15 [21]	GPGPU	Yes	CSR5g

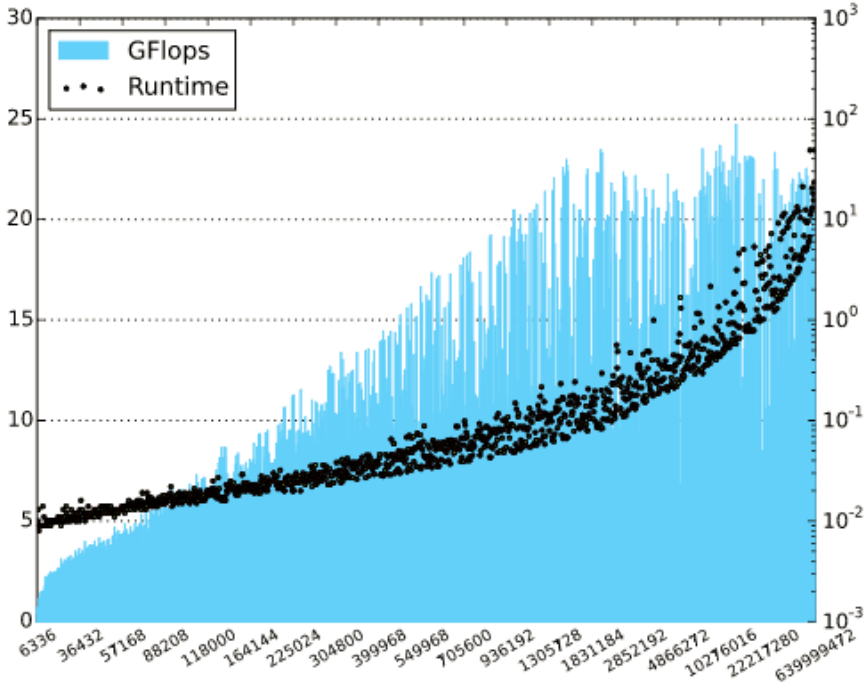
Platforms (Architectures)

- Architecture (Many-core)
  - Representative many-core architectures: Intel Xeon Phi, GPGPU
  - Much different architecture design:
    - Intel Xeon Phi: Knights Landing, CMP+ SIMD
    - GPGPU: NVidia Tesla M40, SIMT

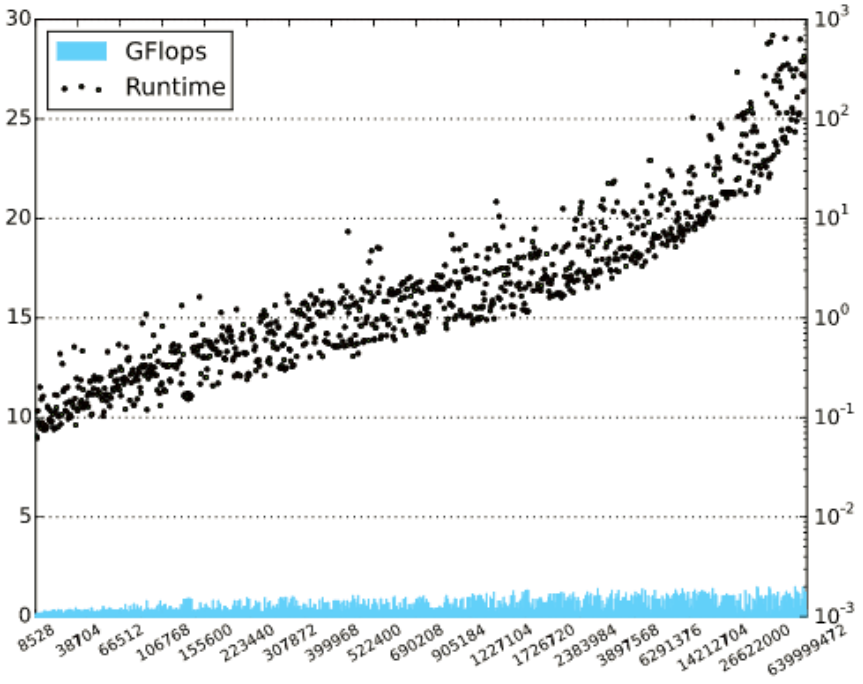


	Xeon Phi	GPU
Architecture	Knights Landing	Tesla
Model	7250	M40
Peak perf(float)	6.1 TFLOPS	7 TFLOPS
Details	# of cores: 68	# of SMs: 24
	# of HT/core: 4	# of coes/SM: 128
	SIMD width: 512	warp size: 32
	L1 cache: 32KB(D)+ 32KB(I)	shared memory/SM: 48KB
	L2 cache: 1024KB/two coes	L2 cache: 3MB
	Memory: 16GB MCDRAM 96GB DDR4	Global Memory: 24GB

Performance on CPU

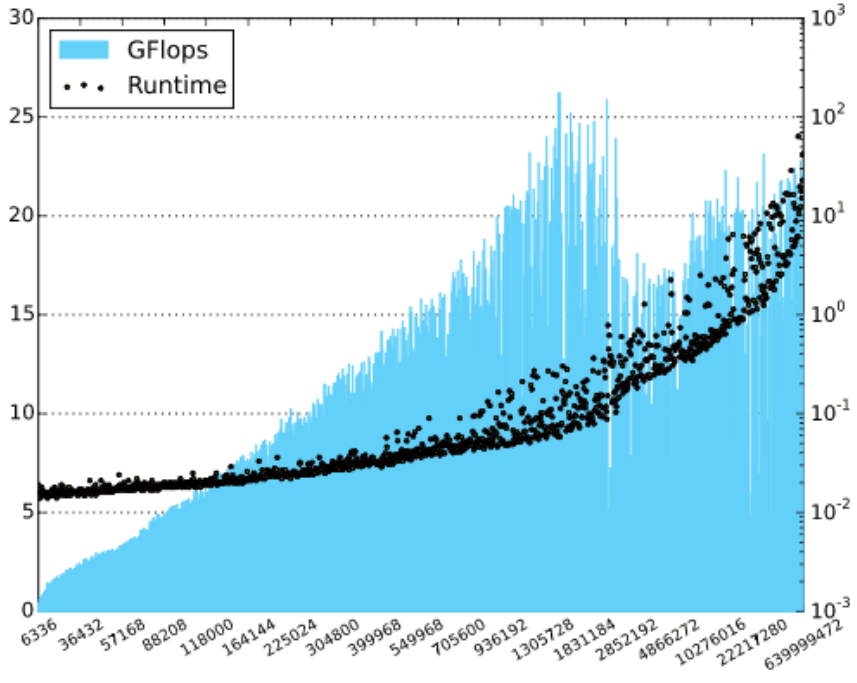


(i) CVR on Phi

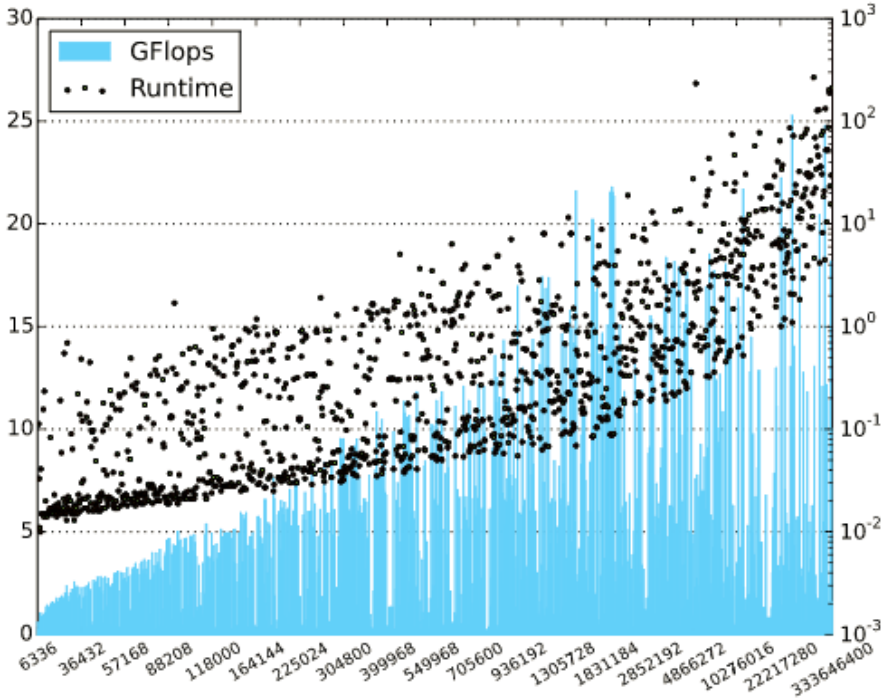


(c) CSC on Phi

Performance on CPU (Continued)



(j) CSR5 on Phi

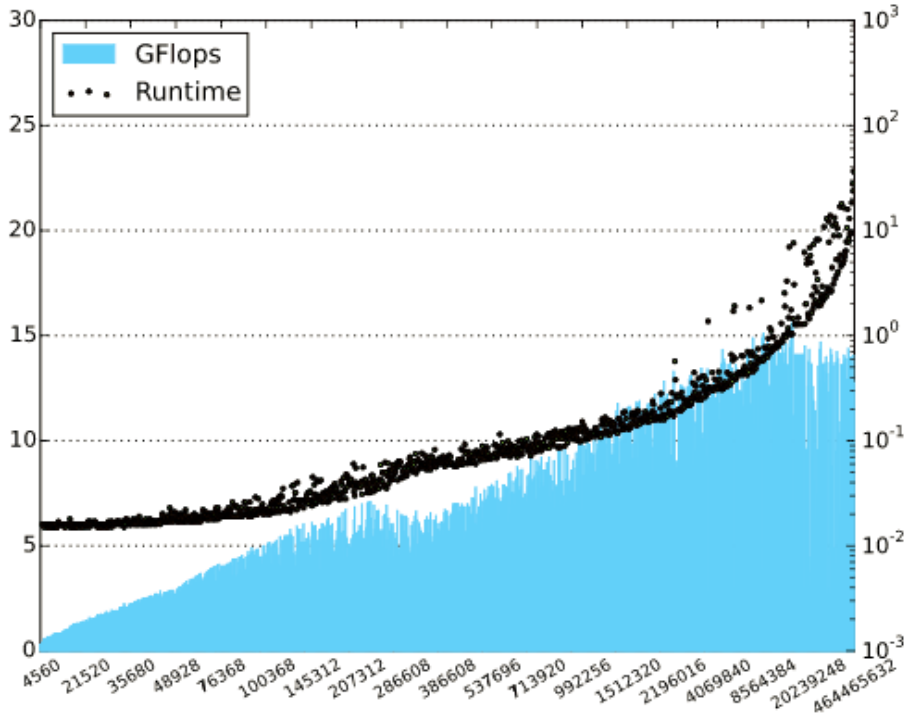


(g) ESB-s on Phi

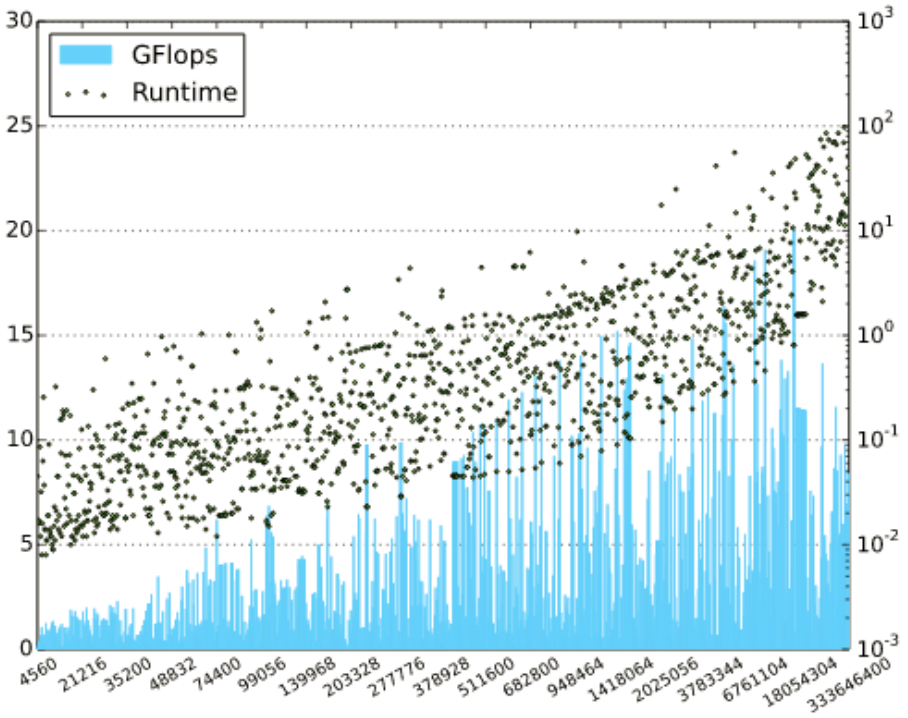
Observations on CPU

- CSR, IE, CSR5 and CVR, show good performance on both small and large sparse matrices with various sparse patterns.
- COO, CSC, and DIA, which are widely used in real-world scenarios, show much poorer performance.
- BSR, ESB-d and ESB-s are sensitive to the sparse patterns

Performance on GPU



(k) Merge on GPU



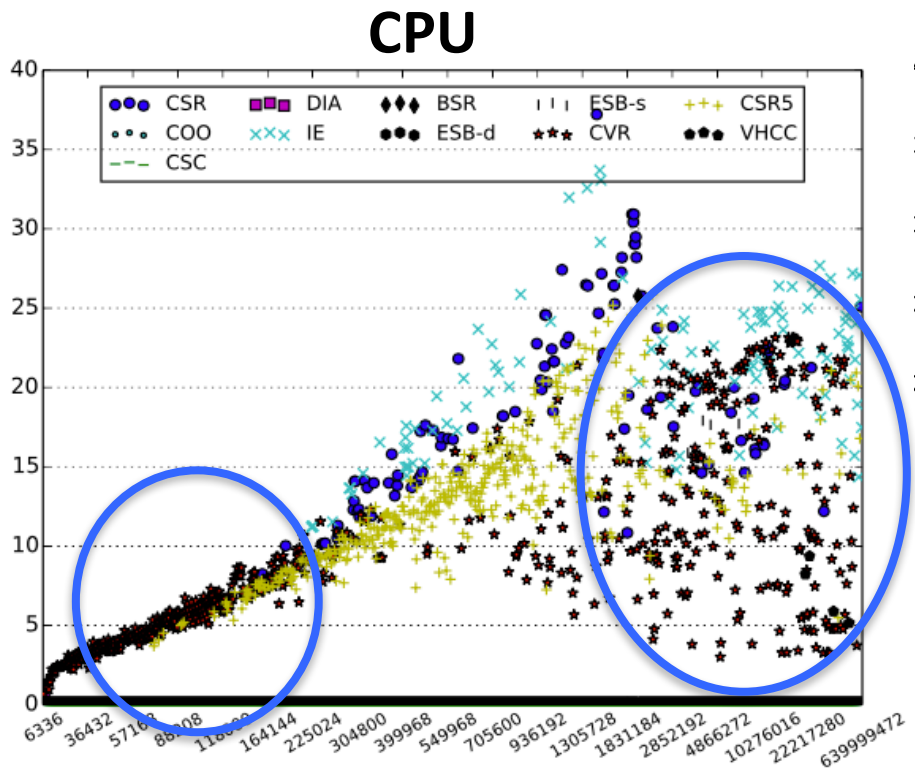
(f) BSRMA on GPU

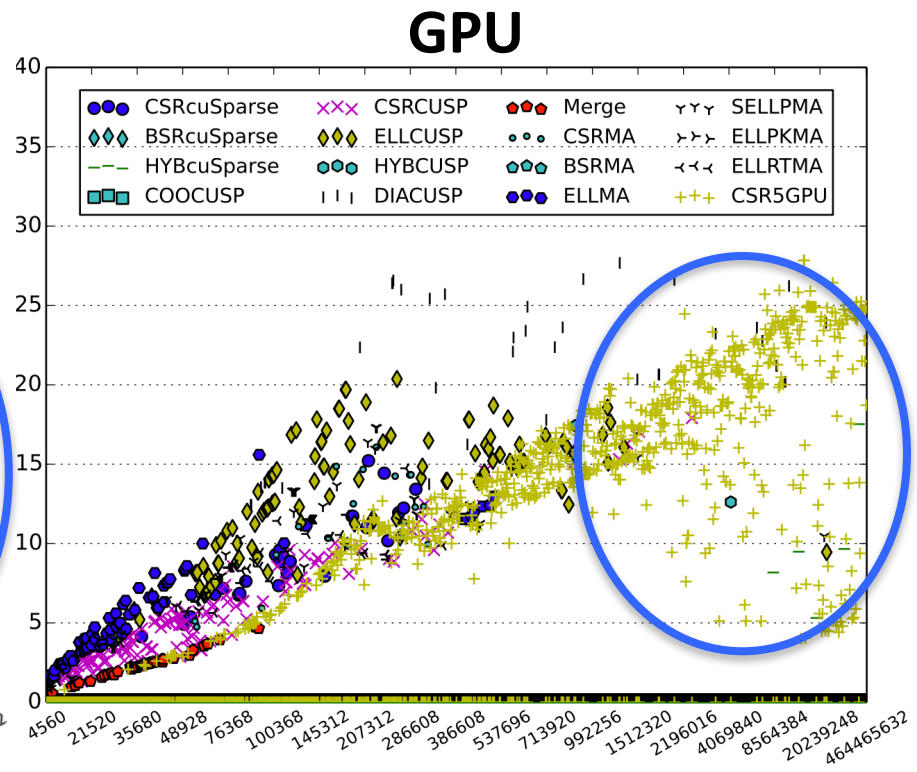
Observations on GPU

- BSR, HYB, ELL and DIA are sensitive to the sparse patterns
- Merge method is stable and insensitive to sparse patterns

Best Methods Distribution

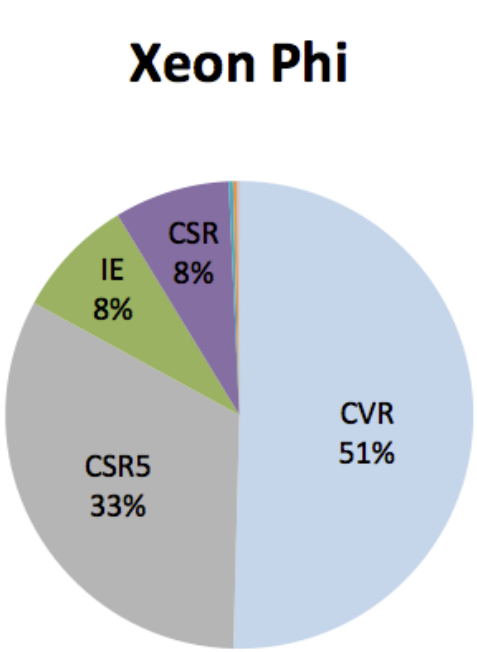
- No single SpMV method is suitable for all sparse matrices
- Some methods show much better performance than others

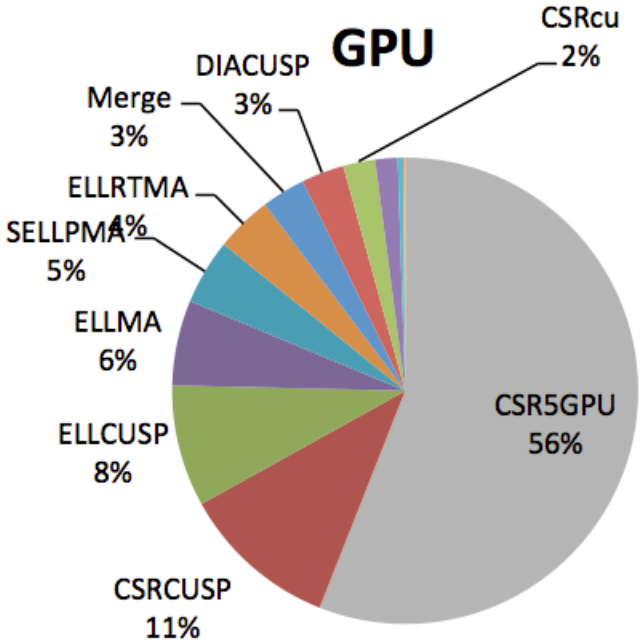




Optimal Methods Distribution

- On Phi:
  - CVR and CSR5 are the optimal for more than 84% data sets
- On GPU:
  - The optimal method is quite scattered. CSR5 occupies 56%.



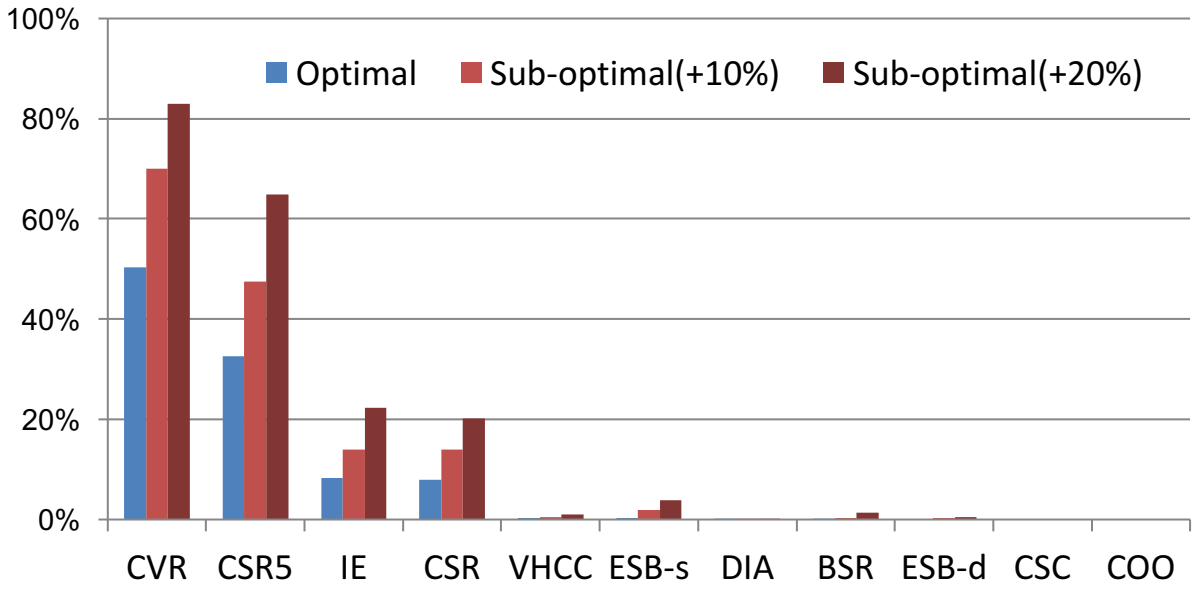


Sub-optimal

- Sub-optimal methods:
  - Slightly worse than the best performance

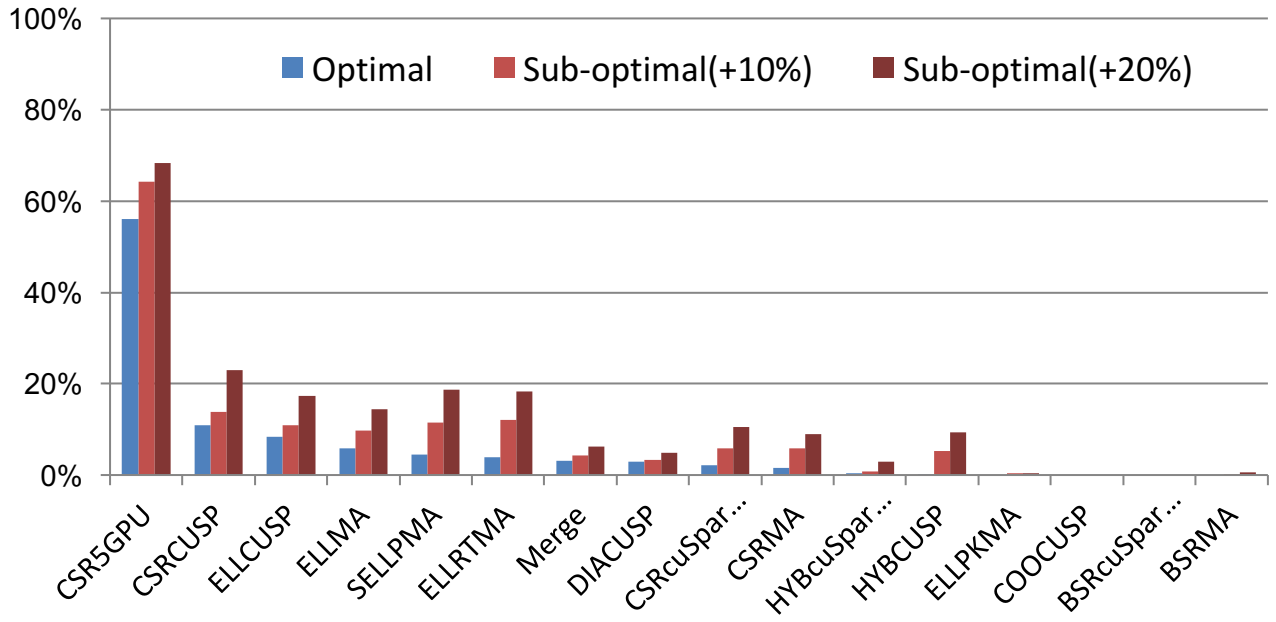
CPU

- Xeon Phi:
  - CVR is the best on more than 82% matrices with less than 20% performance loss to the optimal.
  - Widely used SpMV methods, like CSR, CSC, COO, DIA is not as good as expected.



GPGPU

- Tesla:
  - CSR5 achieves sub-optimal on 65% sparse matrices with less than 20% performance loss.
  - ELL and its derivative show modest performance



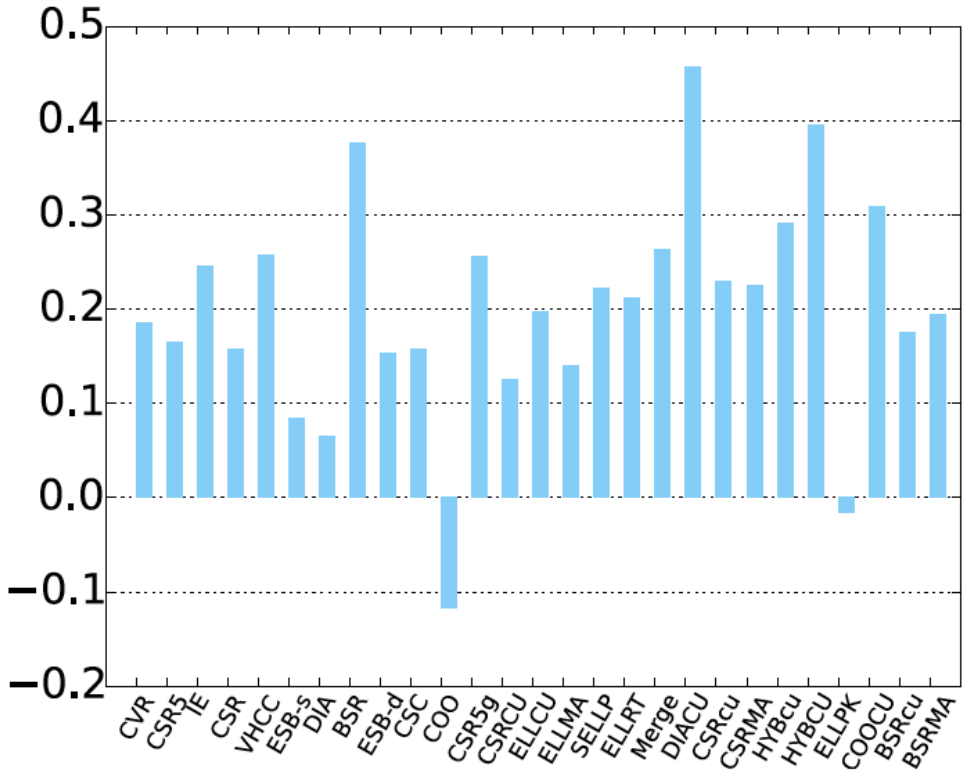
Correlation Analysis

- Correlation :
  - Analyze the correlation between SpMV performance and the features(data scale, sparse pattern and etc.)
  - Pearson correlation coefficient range -1 to 1:
    - 1: positive linear correlation,
    - 0: no linear correlation,
    - −1: total negative linear correlation

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}$$

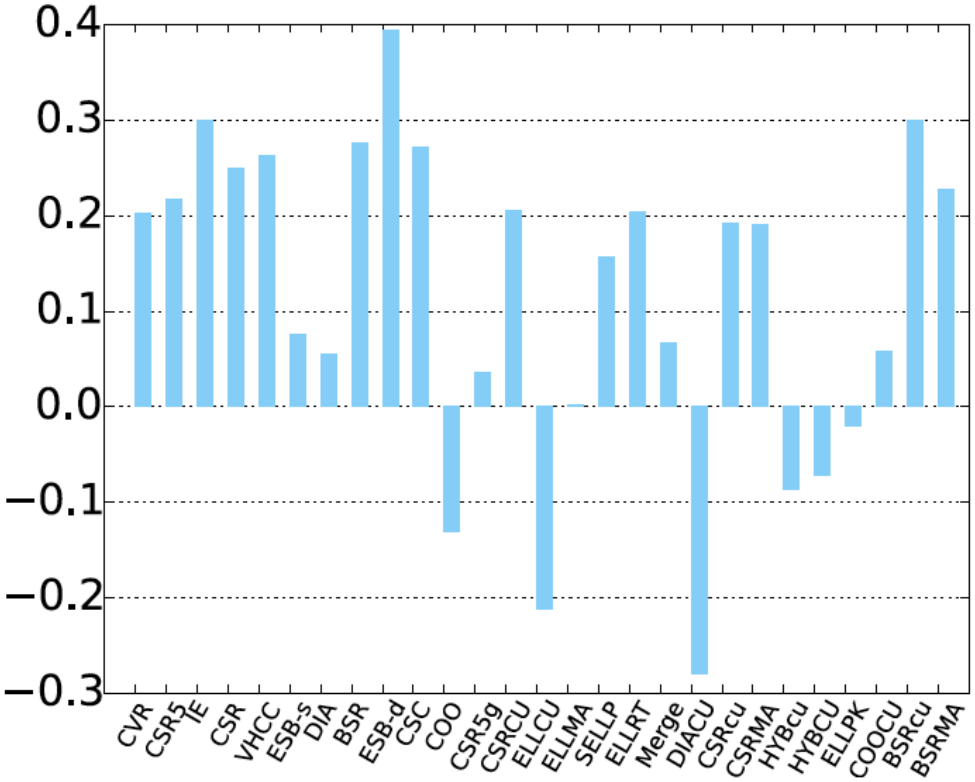
Correlation Analysis

- Data scale:
  - With the number of non-zero elements increasing, the performance of most SpMV methods increase.



Correlation Analysis

- Density:
  - Most SpMV methods show lower throughput when the matrix is sparser.



Conclusion

- Three factors:
  - Sparse matrix: sparse pattern, data scale …
  - SpMV method: parallelization, vectorization, blocking…
  - Hardware platform: Xeon Phi, GPGPU …
- Taking away:
  - Certain methods can achieve good performance for most data sets
  - Some widely used methods, i.e., CSR, CSC, are not as good as emerging ones
  - For most SpMV methods, sparser matrix results in lower throughput
- Open-source project:
  - https://github.com/puckbee/pava: a benchmarking framework, which supports almost all SpMV methods on Intel Xeon Phi and GPGPU.

More result in the paper

Coming soon…

Thanks!

Q&A

Slides for Defending