Auto-Tuning Strategies for Parallelizing Sparse Matrix-Vector (SpMV) Multiplication on Multi- and Many-Core Processors

Kaixi Hou, Wu-chun Feng

{kaixihou, wfeng}@vt.edu

Shuai Che

Shuai.Che@amd.com

Virginia Tech

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

SyNeRG synergy.cs.vt.edu

AMD

AsHES Workshop 2017, Florida, USA

Sparse Matrix Storage

- Sparse matrix: the majority of entries are zeros
- An efficient storage only records nonzero entries
  - Need to ignore zero entries and put all nonzeros together

A 4x4 Matrix

1	0	0	0
3	0	2	0
0	0	0	0
0	5	8	4

Dense Representation

1			
3		2	
	5	8	4

Ignore all the zeros

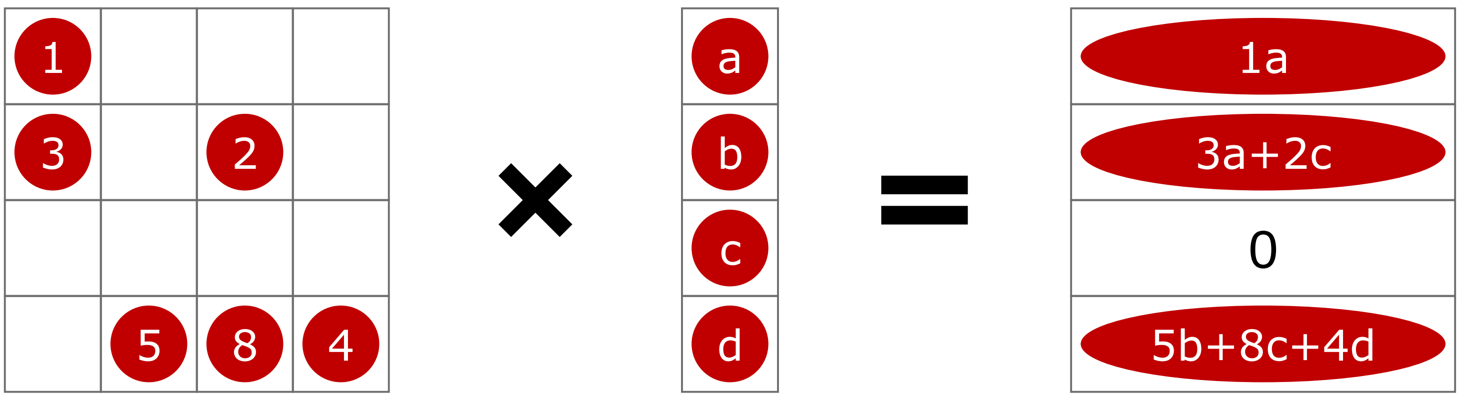
row index	0	1	1	3	3	3
column index	0	0	2	1	2	3
value	1	3	2	5	8	4

Only store the 6 nonzeros

Sparse Representation

Sparse Matrix-Vector Multiplication (SpMV)

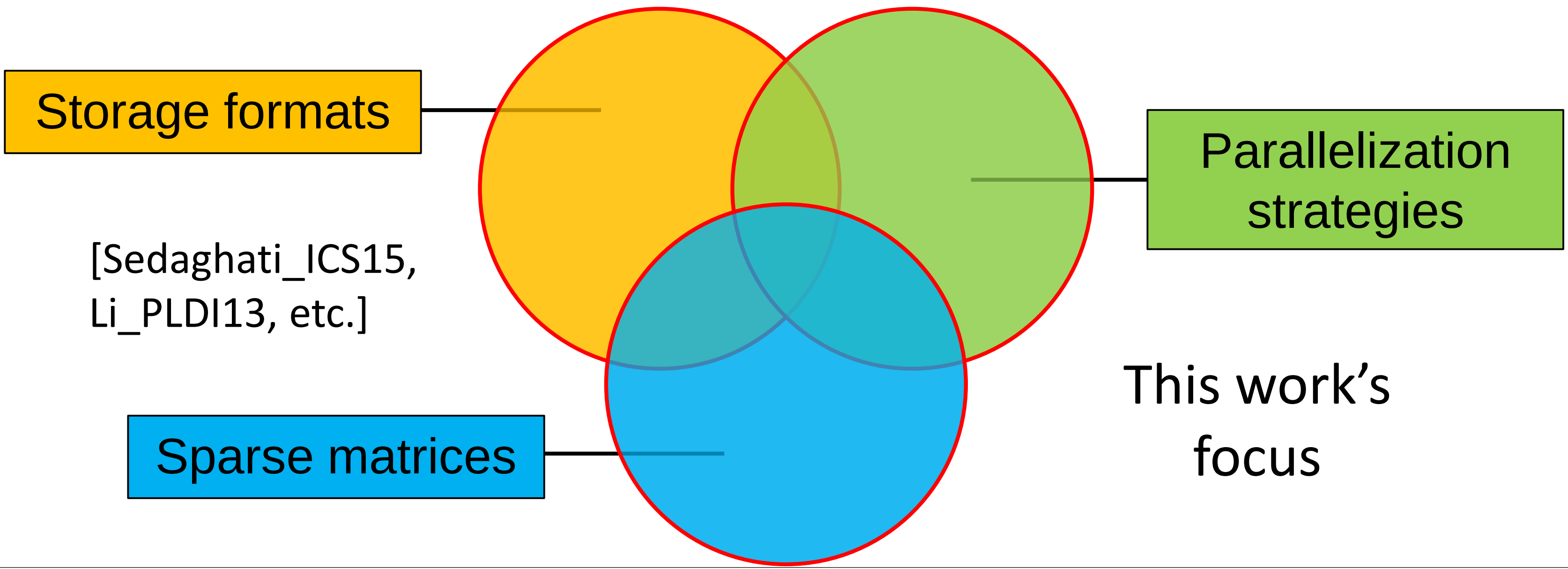
- Problem definition: multiply a sparse matrix ***A*** and a dense vector ***x***, and return the result as a dense vector ***y***



Sparse matrix **A** Dense vector **x** Dense vector **y**

Factors Affecting SpMV Performance

- Storage formats of sparse matrix A
  - CSR, ELLPACK, DIA, COO, BCCOO, BRC, CSR5, etc.
- Parallelization strategies
  - Different formats correspond to different algorithms
  - Even same format can lead to different parallel strategies, e.g., granularities of parallelism, optimizations, etc.
- Input sparse matrices themselves



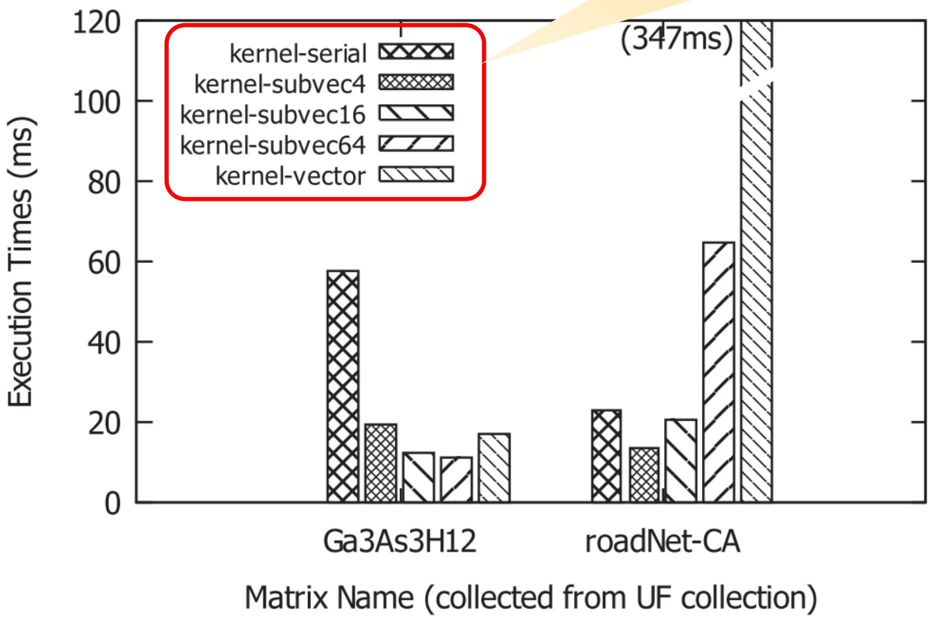
Motivating Examples

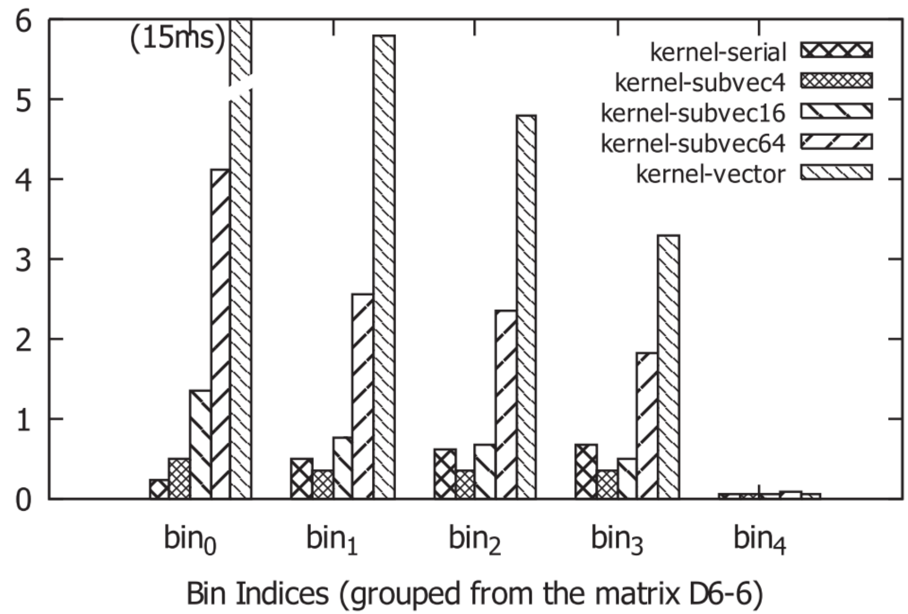
- How input sparse matrices and parallelization strategies affect performance?

Motivating Examples

- How input sp… …nd parallelization strategies affect perfor…

Different parallel strategies





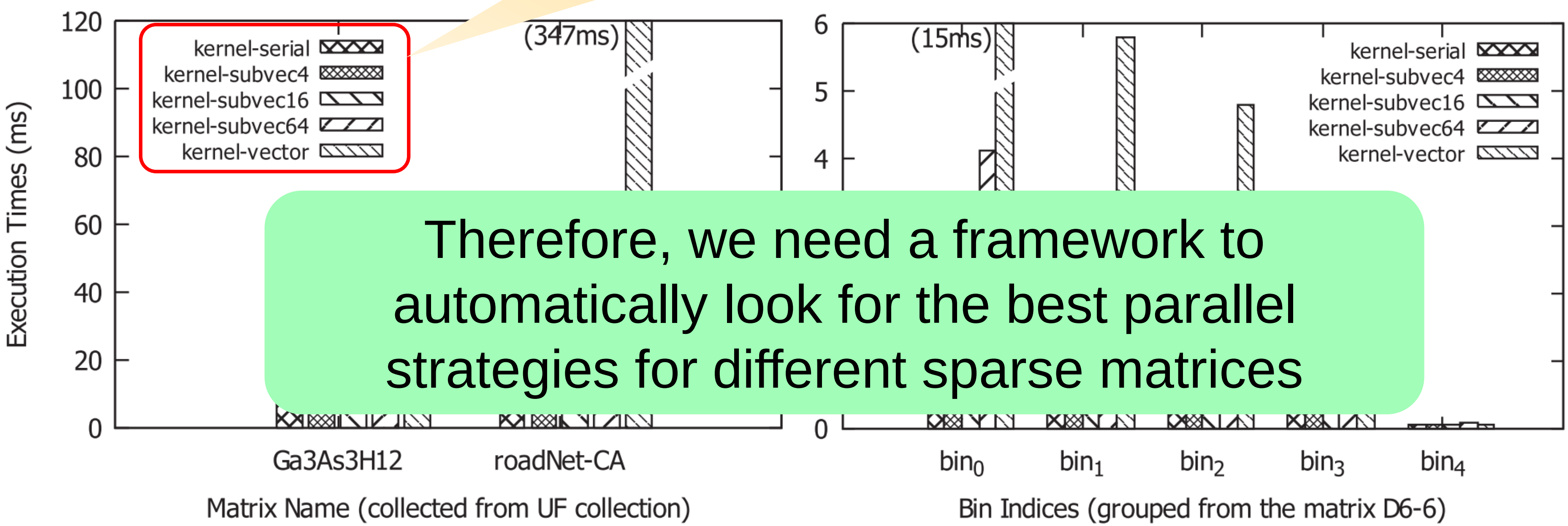
Different Input Sparse Matrices prefer different parallelization strategies

Different Nonzero Row Sizes prefer different parallelization strategies

Motivating Examples

- How input sparse matrices and parallelization strategies affect performance

Different parallel strategies



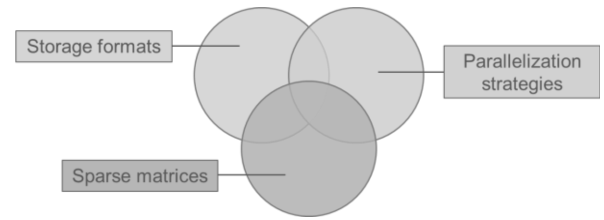
Therefore, we need a framework to automatically look for the best parallel strategies for different sparse matrices

Different Input Sparse Matrices prefer different parallelization strategies

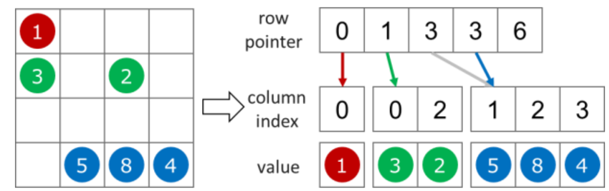
Different Nonzero Row Sizes prefer different parallelization strategies

Outline

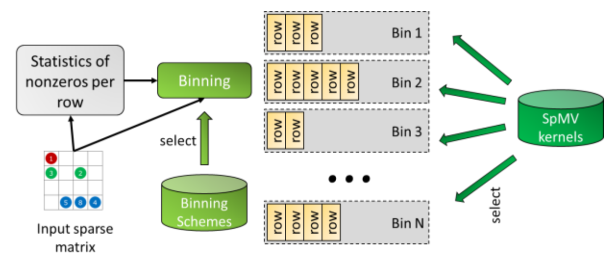
- Sparse Matrix and SpMV
- Motivation



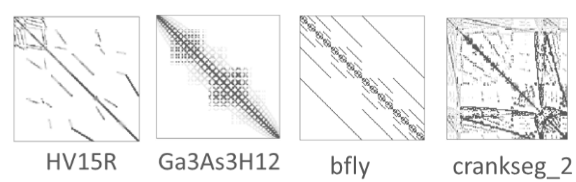
- Background
  - CSR format



- Reconfigurable SpMV Method
  - Binning schemes
  - Kernel choices
- SpMV Data Mining Framework

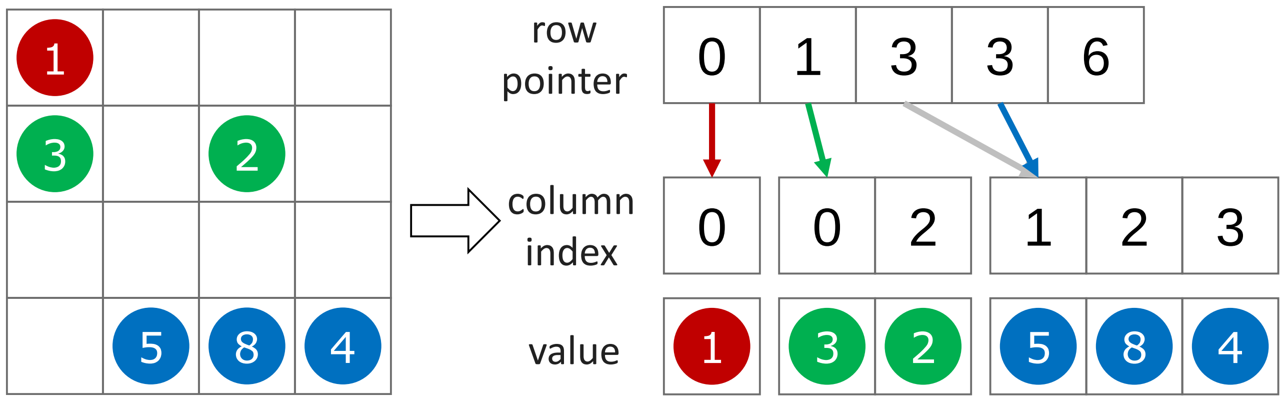


- Evaluations & Conclusion



Compressed Sparse Row (CSR) Format

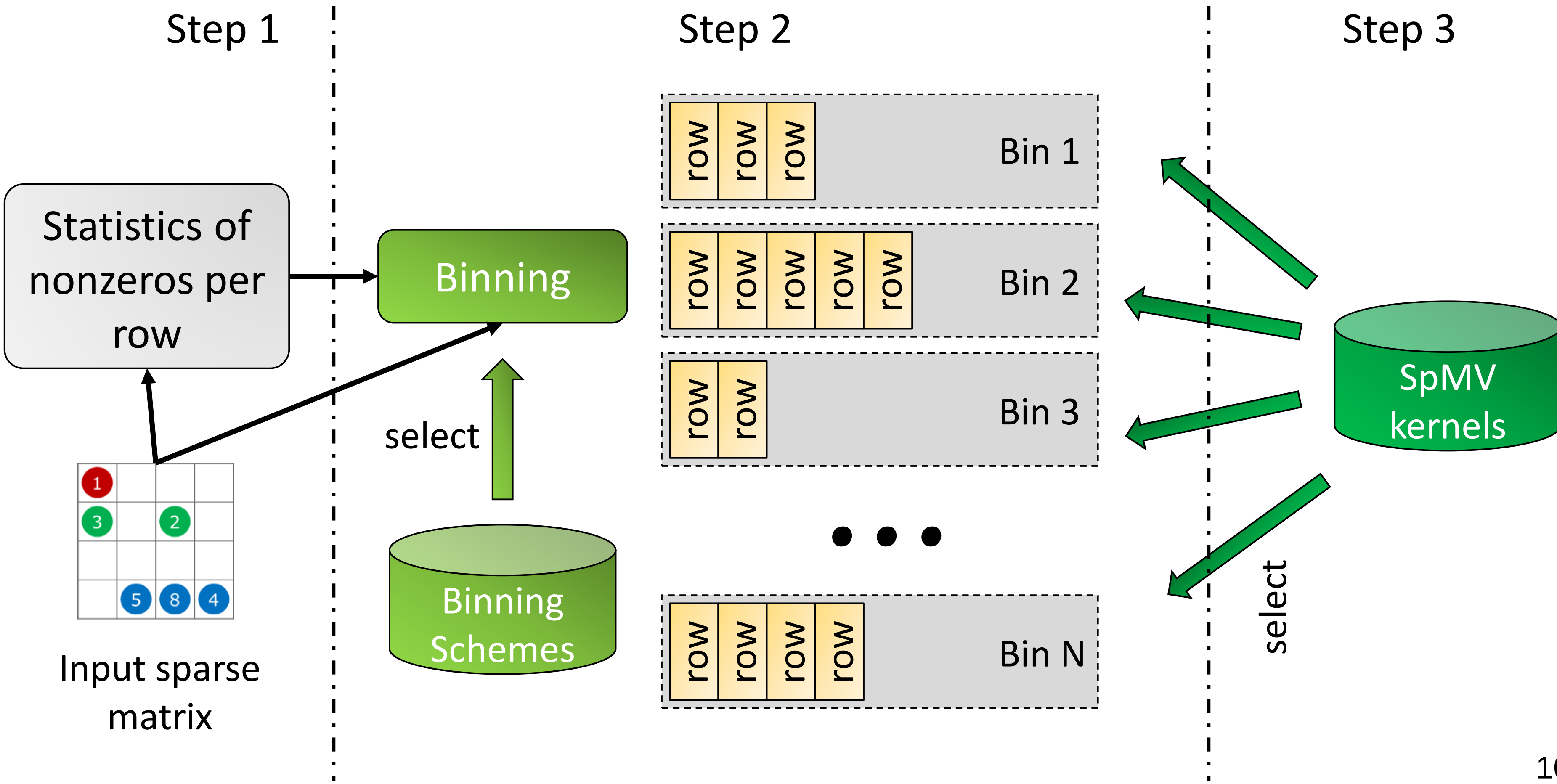
- Widely-used sparse matrix format
  - Store row pointers, column indices, and nonzero values



CSR Representation

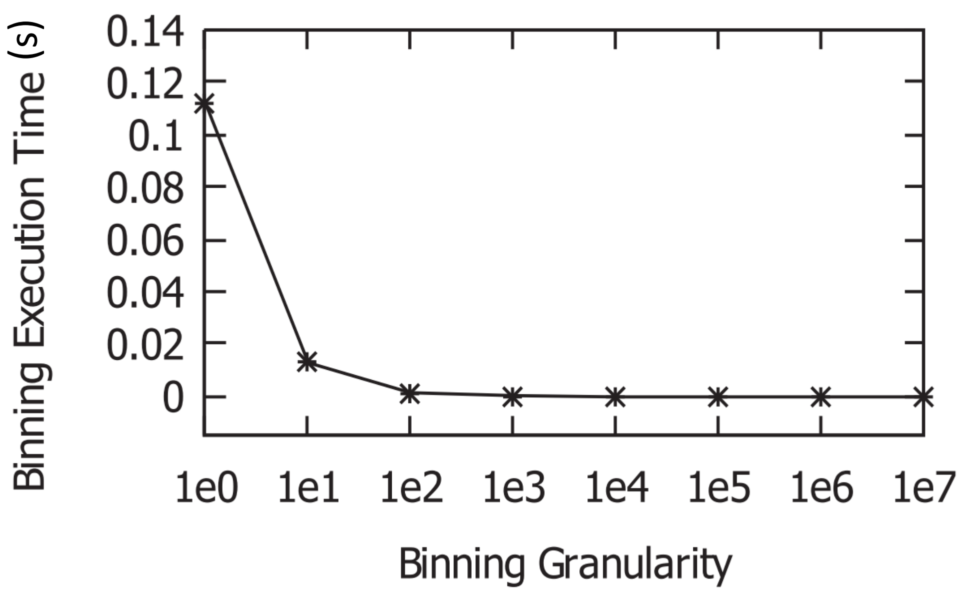
Reconfigurable SpMV Method

- Overview of our SpMV method
  - Binning schemes and kernels can be customized



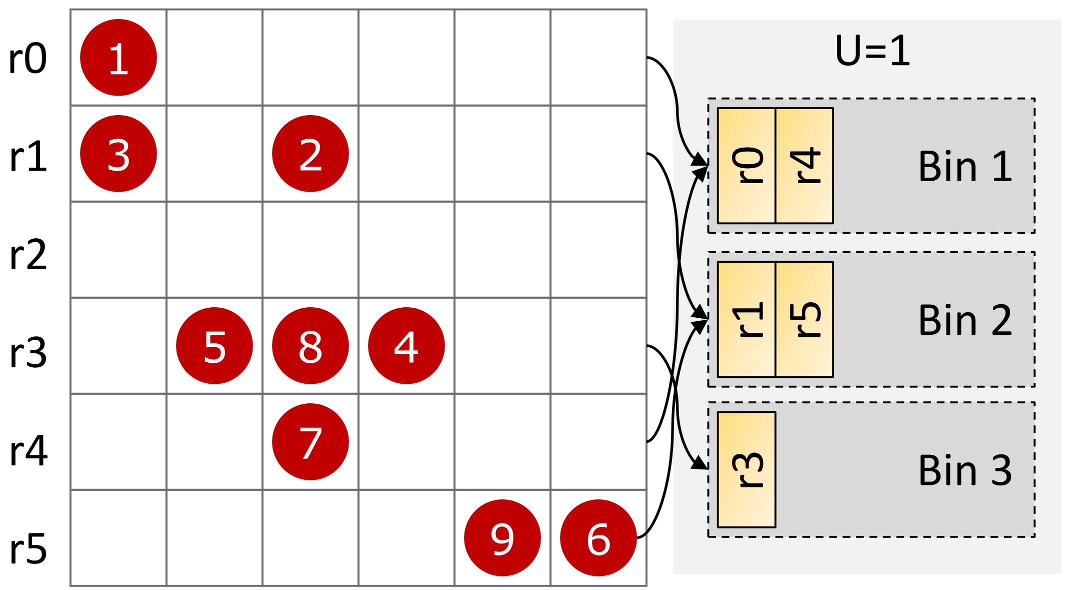
Binning Schemes

- For load balance, we group (permute) rows into different bins, according to their nonzero numbers
- However, how to choose correct granularities for binning?
  - Small granularities lead to high binning overhead
  - Large granularities lead to high row variance in the same bin



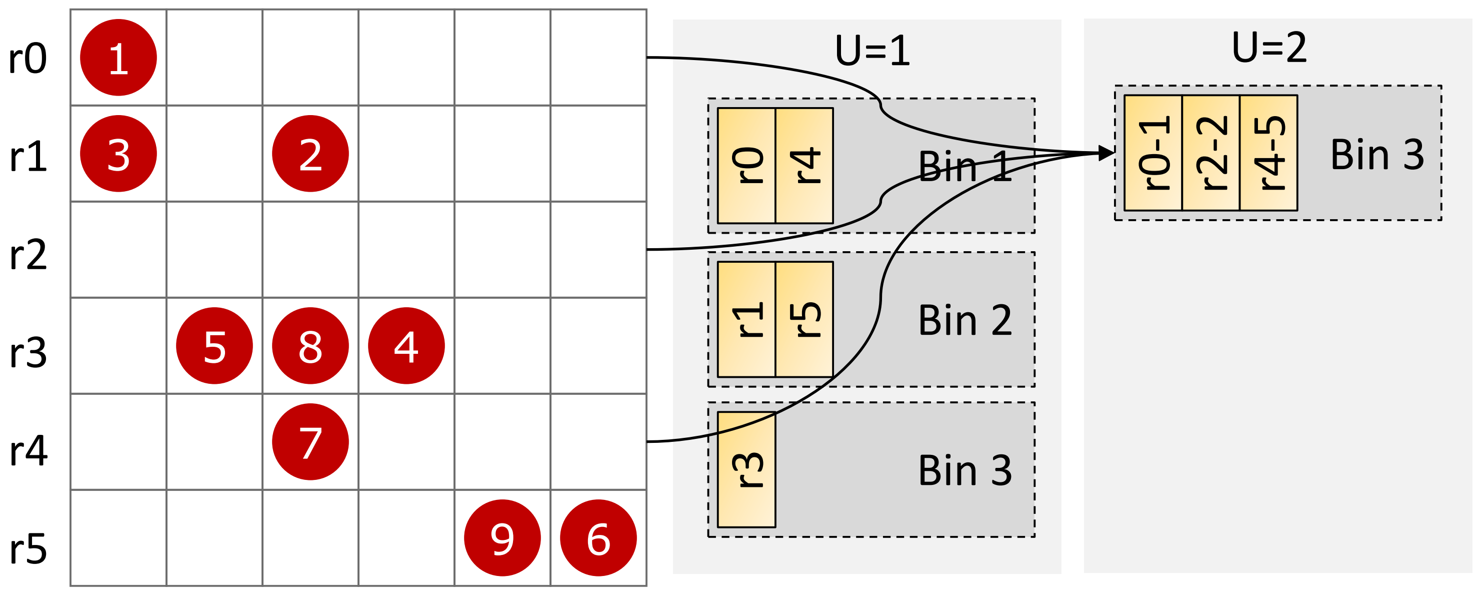
Binning Schemes

- In our method, we treat multiple neighboring rows as a single “virtual” row
  - We have a set of candidate granularity units (denoted as ***U***) to determine the number of neighboring rows



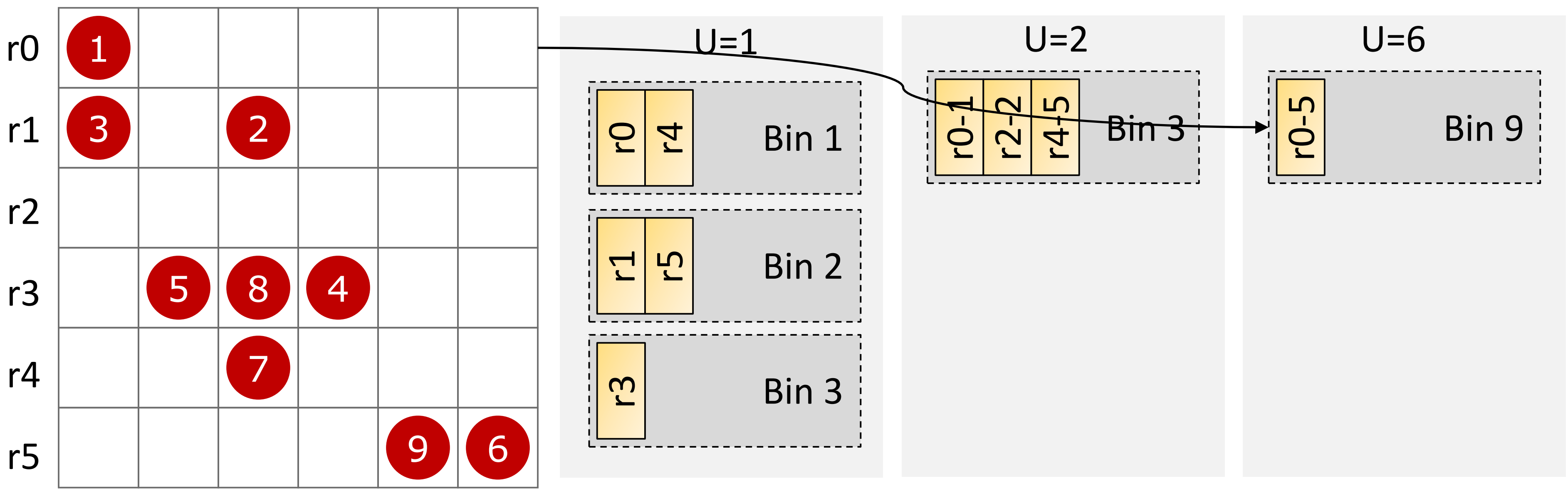
Binning Schemes

- In our method, we treat multiple neighboring rows as a single “virtual” row
  - We have a set of candidate granularity units (denoted as ***U***) to determine the number of neighboring rows



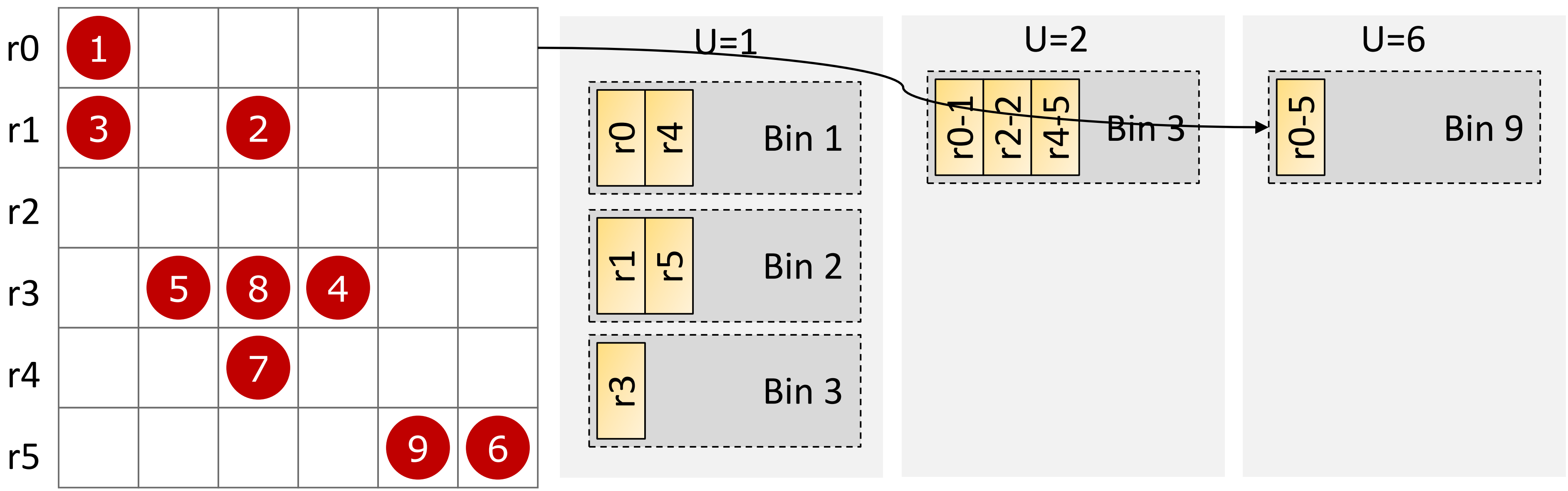
Binning Schemes

- In our method, we treat multiple neighboring rows as a single "virtual" row
  - We have a set of candidate granularity units (denoted as ***U***) to determine the number of neighboring rows



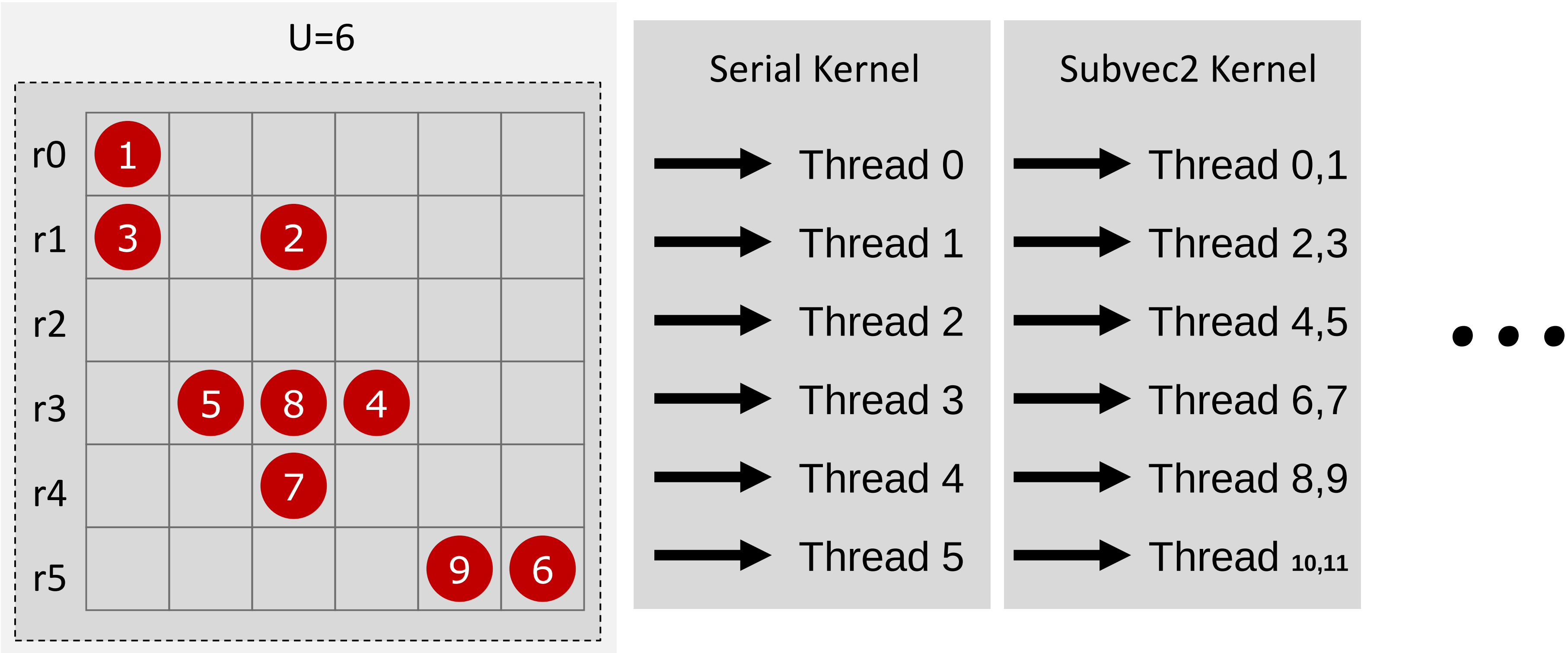
Binning Schemes

- In our method, we treat multiple neighboring rows as a single “virtual” row
  - We have a set of candidate granularity units (denoted as ***U***) to determine the number of neighboring rows
- Better locality, throughput, etc.



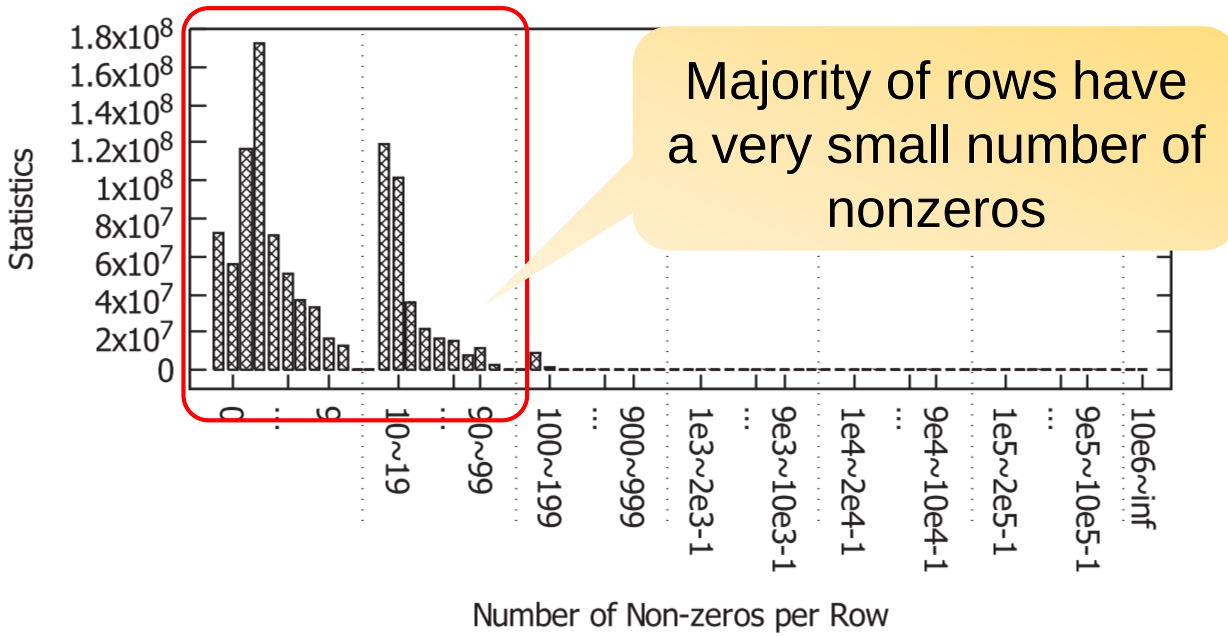
Kernel Choices

- Different SpMV kernels to process different types of rows
  - Assign a row to one thread
  - Assign a row to multiple threads (wavefront-level)
  - Assign a row to multiple wavefronts (thread-block-level)



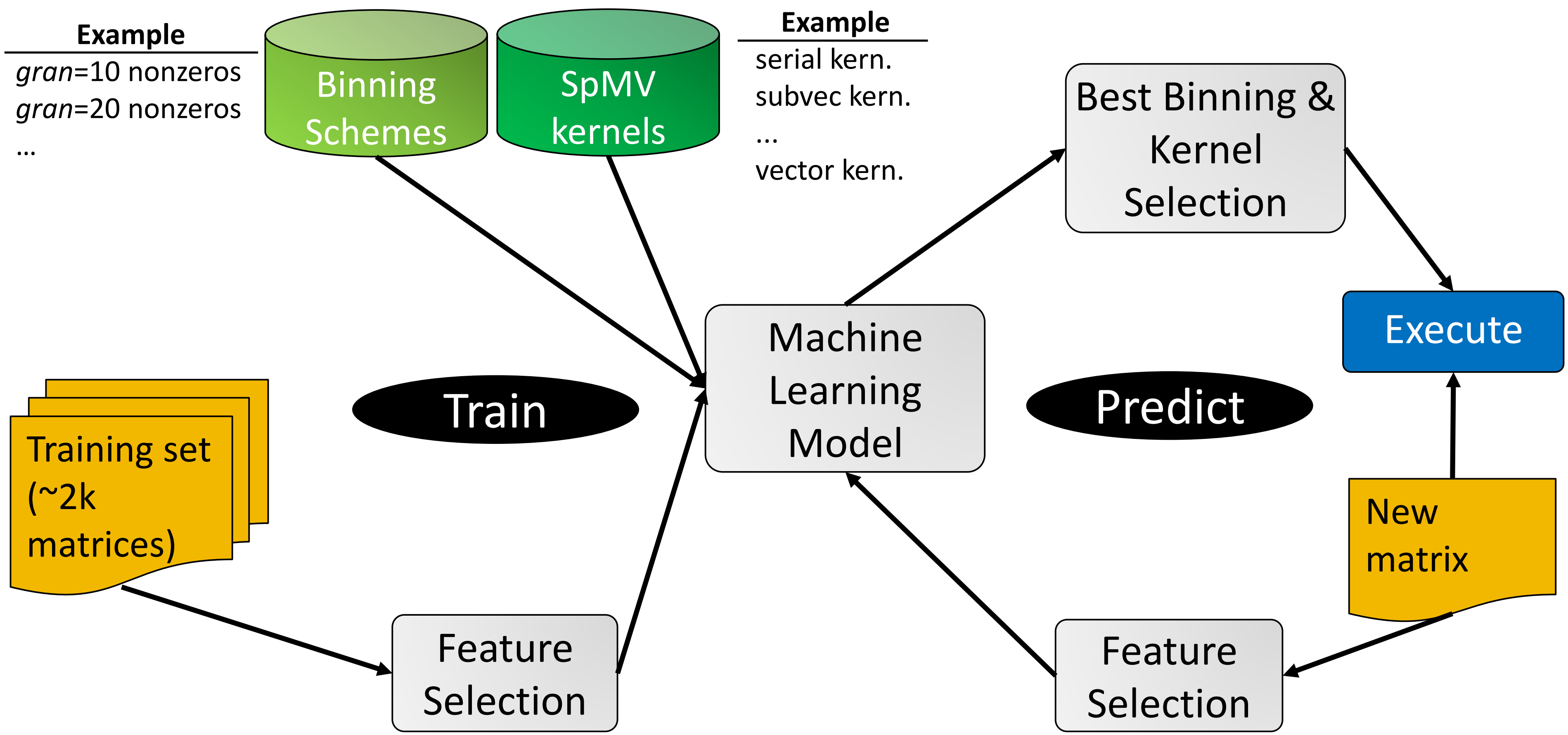
Kernel Choices

- We use up to a thread block to process one nonzero row
- Current work only focuses on short and medium row sizes
  - Our bin-based method can easily be extended to support long rows (e.g., dynamic parallelism based method)



SpMV Data Mining Framework

- Overview of our data mining framework to look for the optimal binning policies and SpMV kernels



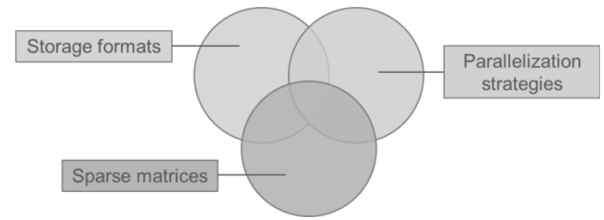
Classification Tools

- The classification tool is *C5.0* for data mining *
- We select over 2K sparse matrices from *the University of Florida sparse matrix collection* as the training set
  - 75% are used for training
  - The rest are used for testing
- The error rate of learning is 5~15%
  - 1st stage of learning (for binning schemes) is around 5%
  - 2nd stage (for parallelization strategies) is less than 15%
- Finally, we have two generated rule-sets
  - One is for ***how to select binning schemes***
  - Another is for ***how to select kernels for each bin***

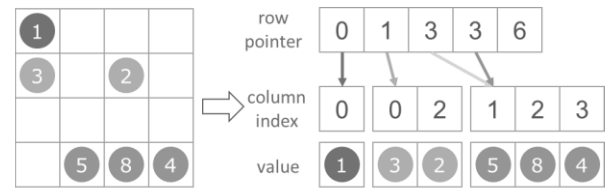
* https://www.rulequest.com/see5-info.html

Outline

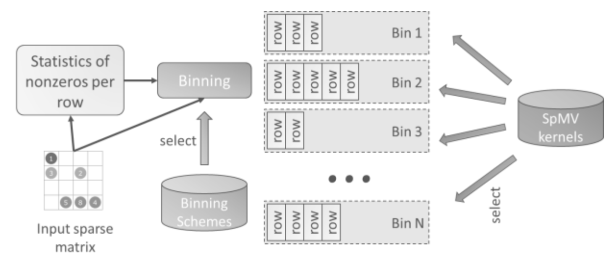
- Sparse Matrix and SpMV
- Motivation



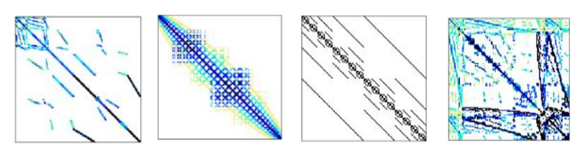
- Background
  - CSR format



- Reconfigurable SpMV Method
  - Binning schemes
  - Kernel choices
- SpMV Data Mining Framework



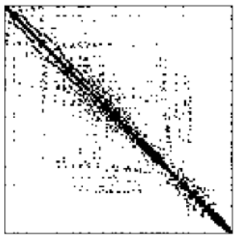
- **Evaluations & Conclusion**



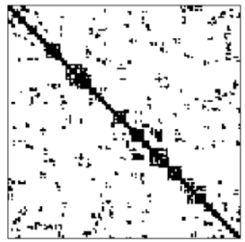
HV15R Ga3As3H12 bfly crankseg_2

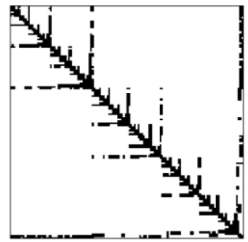
Benchmark Suite

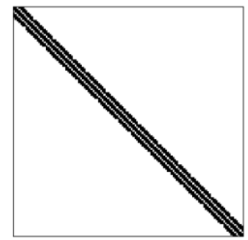
- We select 16 matrices from *the University of Florida sparse matrix collection*

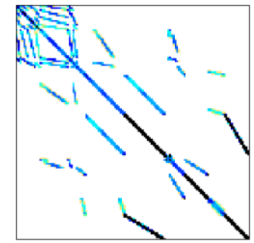

whitaker3_dual

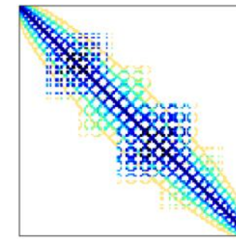

shar_te2-b2


roadNet-CA

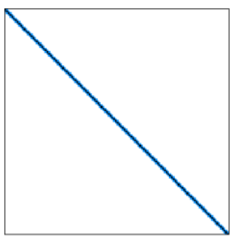

pkustk14


pcrystk02

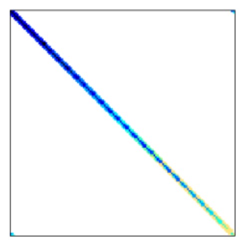

HV15R

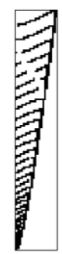

Ga3As3H12

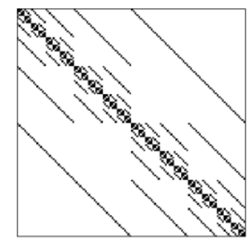

europe_osm

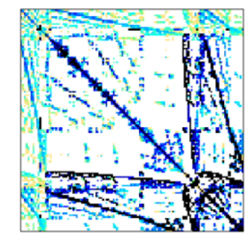

denormal

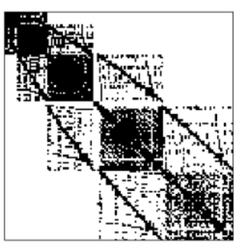

D6-6

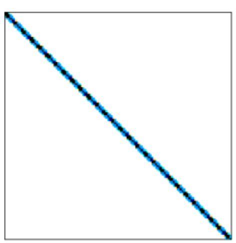

cryg10000


ch7-9-b3


bfly


crankseg_2


dictionary28


apache1

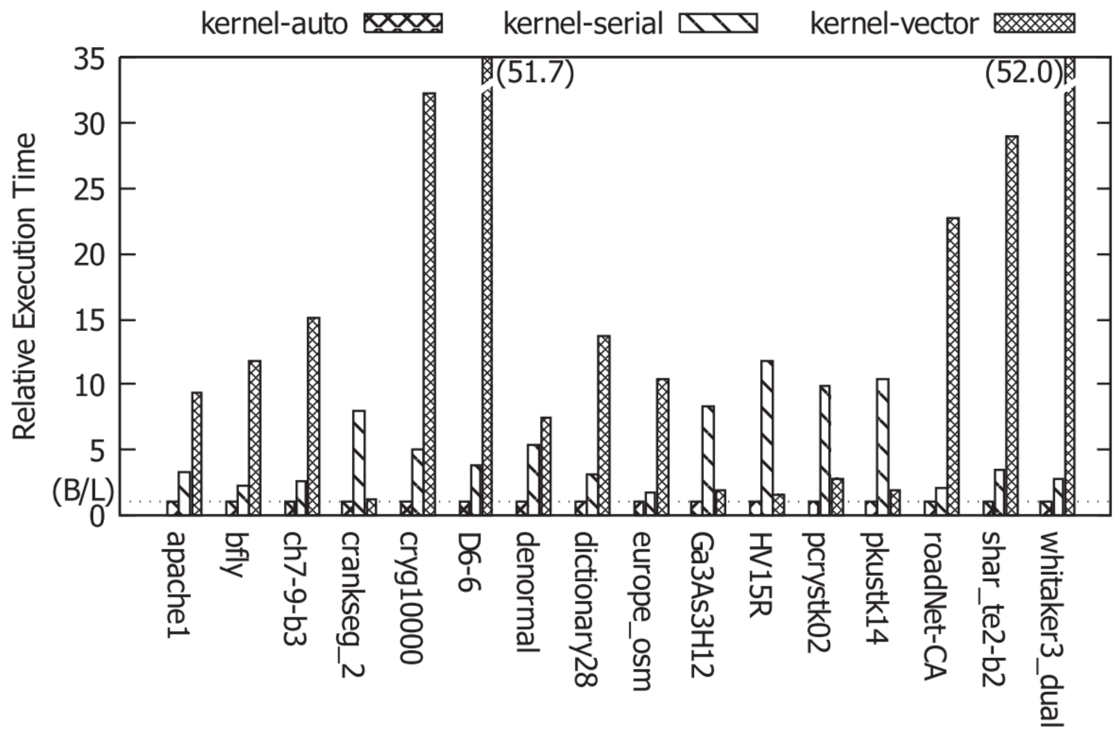
… from ***structural*** problem, ***undirected graph*** *sequence*, ***combinatorial*** problem, ***materials*** problem, ***counter-example*** problem, ***theoretical/quantum chemistry*** problem, ***CFD*** problem, ***duplicate materials*** problem, ***2D/3D*** problem

Experiment Platform

- AMD A10-7850K APU: A real HSA hardware
- It features **four 3.7 GHz CPU cores** and **eight 720MHz GPU compute units**
- Our system is equipped with 16 GB memory
- We use AMD Heterogeneous System Architecture (HSA) - Linux amdkfd v1.4 release
- We use CL Offline Compiler CLOC V0.9.5 (HSA 1.0F) with SNACK support

Performance Evaluation

- Speedups from our framework

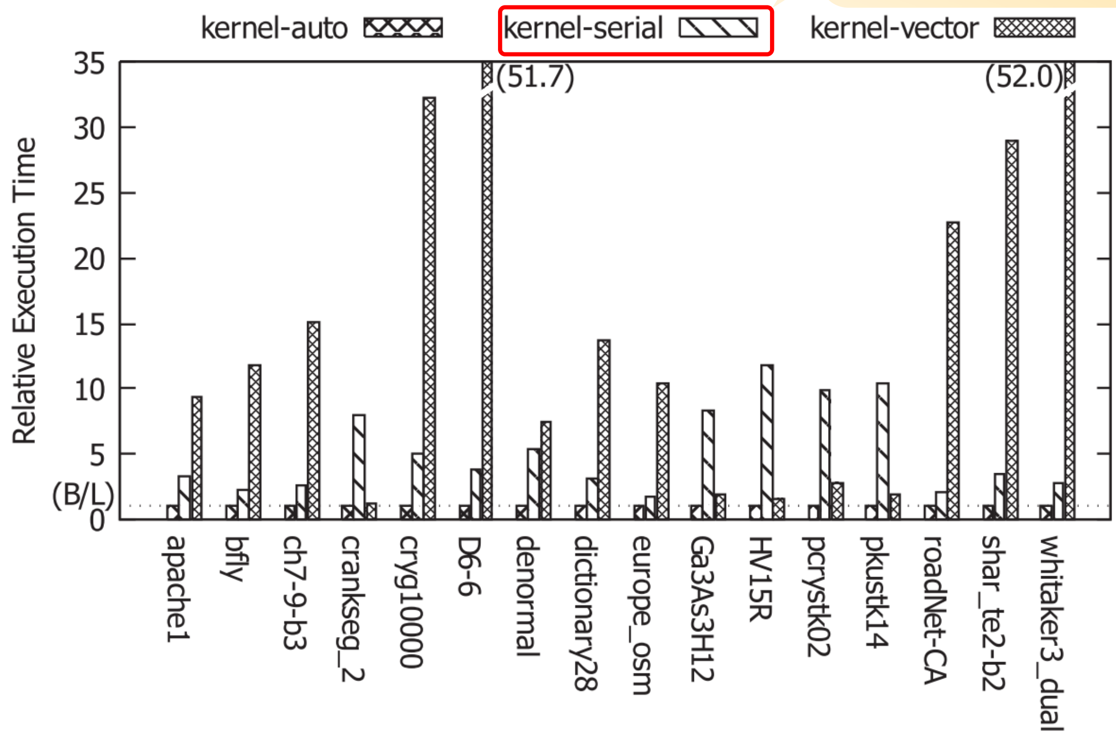


  - **Kernel-auto** is the kernel from our SpMV framework by automatically selecting binning and parallelization strategies

Performance Evaluation

Assign one row to each thread

- Speedups from our framework

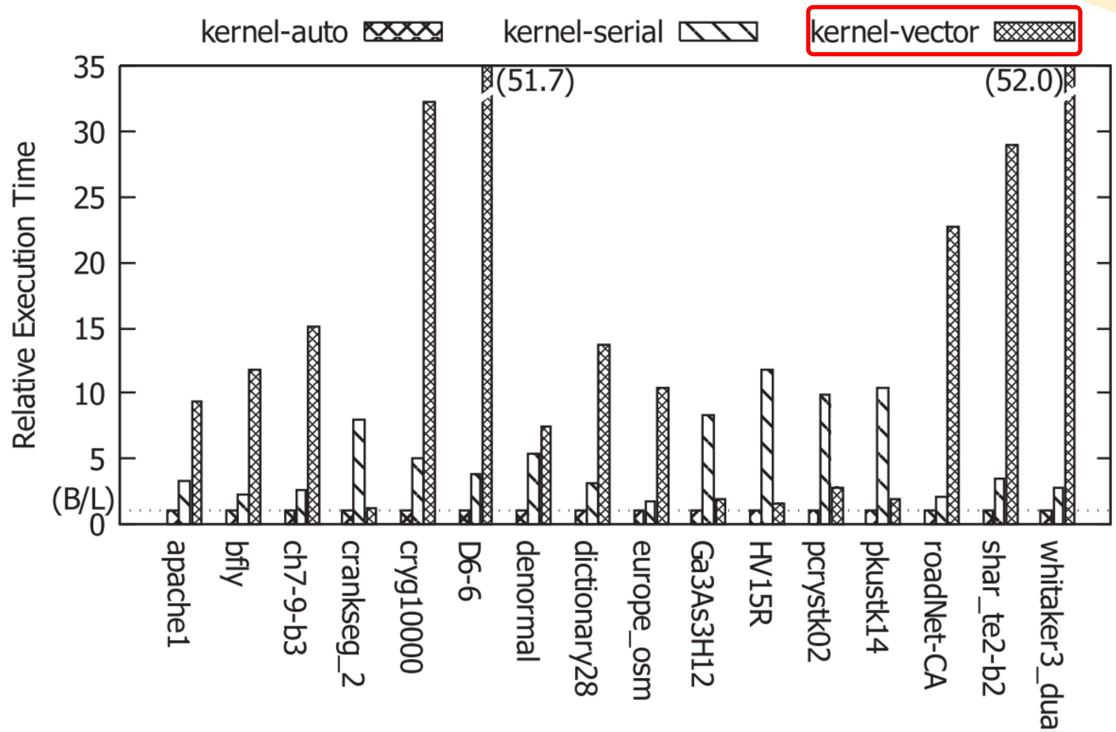


  - **Kernel-auto** is the kernel from our SpMV framework by automatically selecting binning and parallelization strategies
  - Compared to **kernel-serial**, we can achieve 1.7x to 11.9x speedups

Performance Evaluation

Assign one row to each thread-block

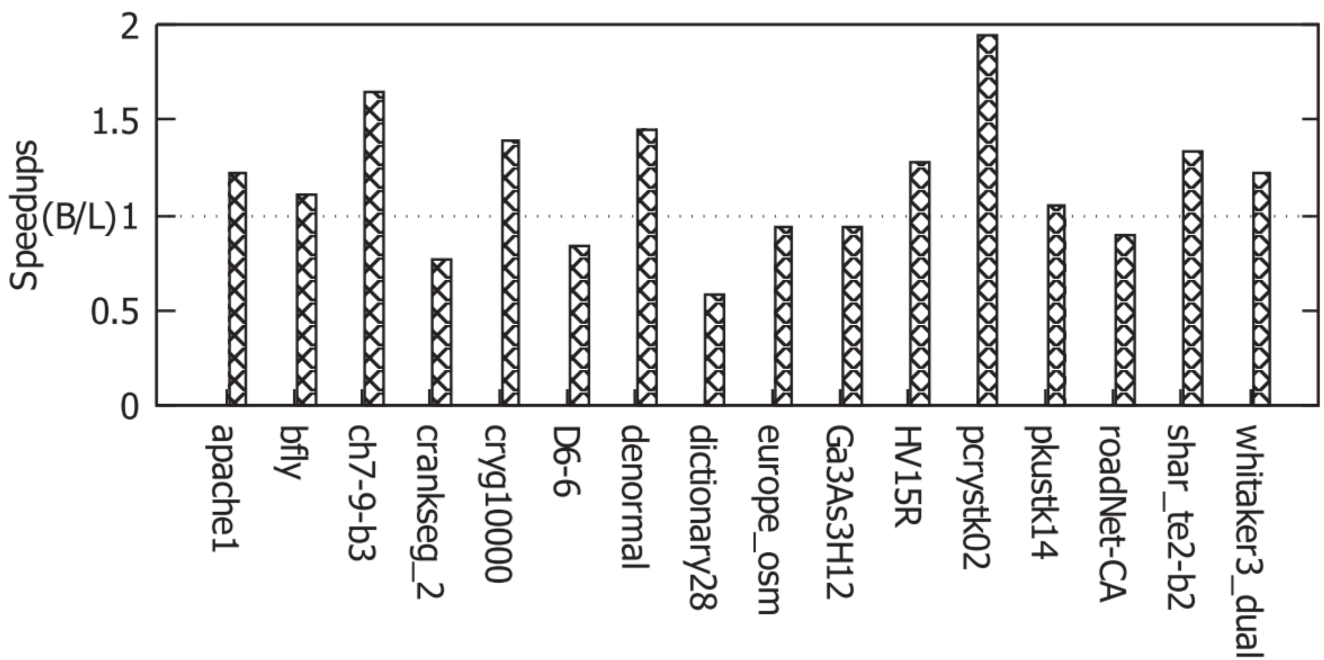
- Speedups from our framework



  - **Kernel-auto** is the kernel from our SpMV framework by automatically selecting binning and parallelization strategies
  - Compared to **kernel-serial**, we can achieve 1.7x to 11.9x speedups
  - Compared to **kernel-vector**, we can get 1.2x to 52.0x speedups

Performance Evaluation

- Speedups from the prior state-of-the-art GPU SpMV "CSR-Adaptive"*



  – Our SpMV can yield better performance over 10 out of 16 sparse matrices and achieve up to 1.9x speedups

* J. Greathouse, M. Daga, "Efficient Sparse Matrix-vector Multiplication on GPUs Using the CSR Storage Format", SC 2014

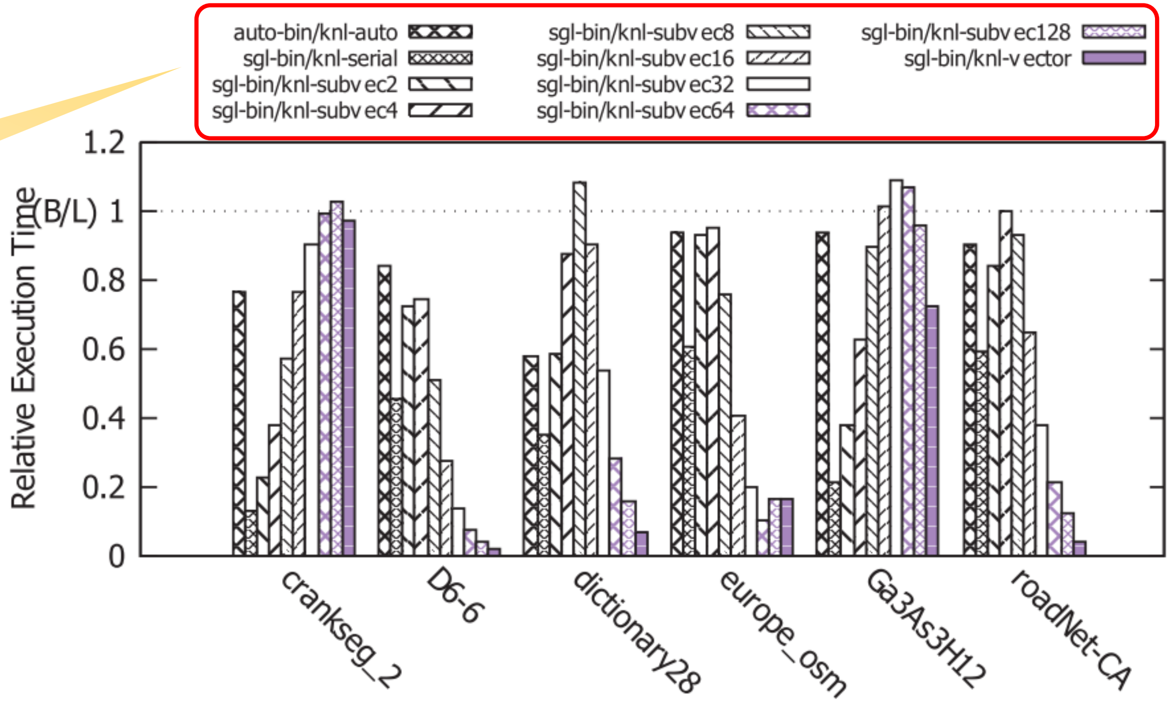
Conclusion

- Proposed a SpMV framework using the machine learning model to automatically find the optimal parallel strategies
  - Focusing on the CSR format
  - Choosing the appropriate grouping policy to organize independent rows (as “virtual” rows) into different bins
  - Looking for the suitable kernels to process the bin rows
- Achieved significant performance improvements over the SpMV kernels using single kernel
- Achieved up to 1.9x speedups over other state-of-the-art SpMV kernels

Discussion & Future Work

- Grouping all rows to a single bin
  - Need more features of matrix to identify when to put all rows into a single bin

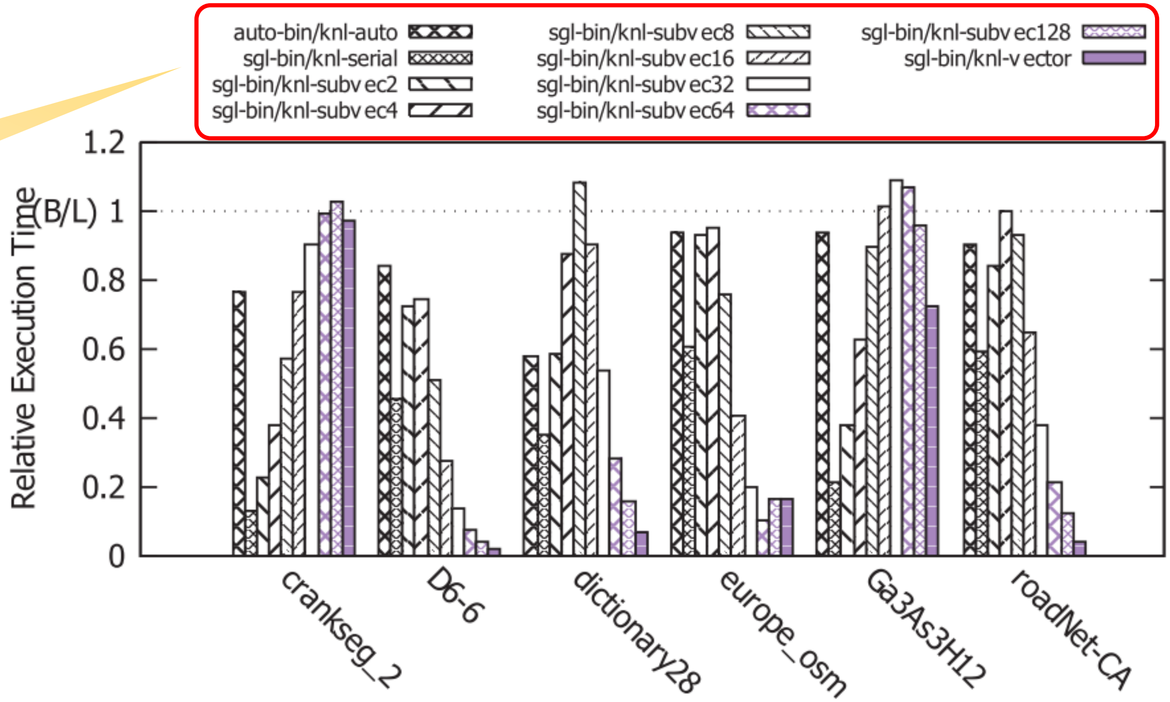
Different parallel strategies



Discussion & Future Work

- Grouping all rows to a single bin
  - Need more features of matrix to identify when to put all rows into a single bin

Different parallel strategies

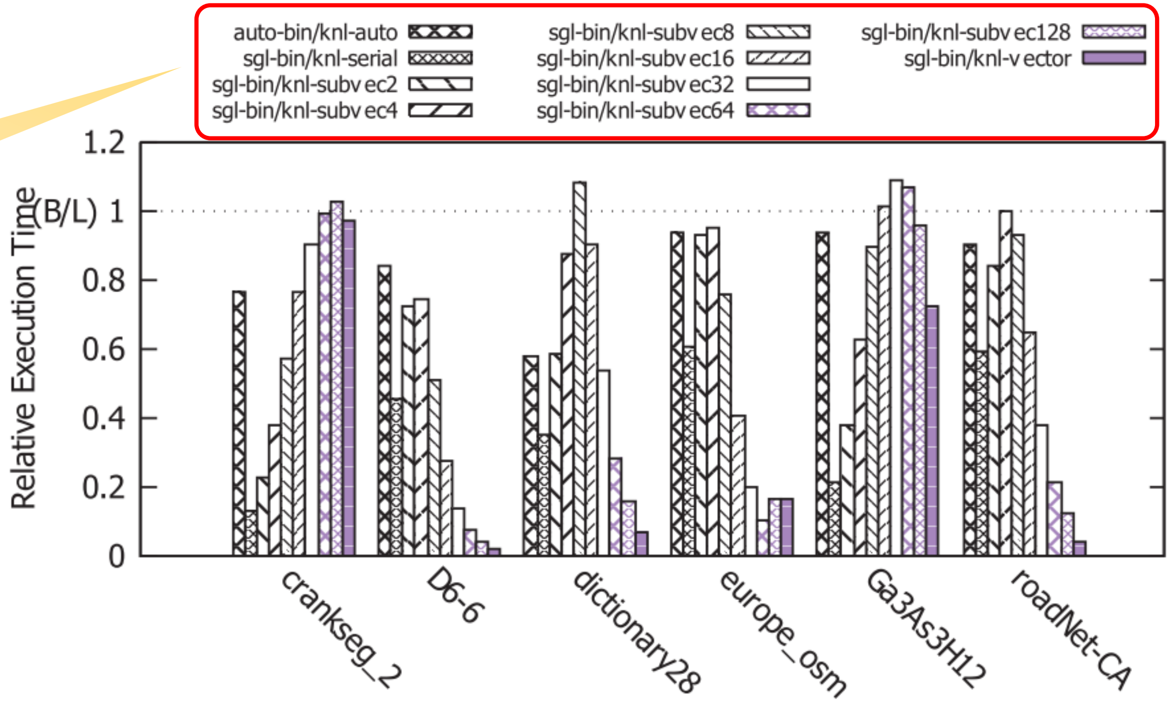


- Extending our work to fully utilize both CPU and GPU
  - High-volume bins on throughput-oriented processors
  - Low-volume bins on latency-oriented processors

Discussion & Future Work

- Grouping all rows to a single bin
  – Need more features of matrix to identify when to put all rows into a single bin

Different parallel strategies



- Extending our work to fully utilize both CPU and GPU
  –
  –

THANK YOU! More info: kaixihou@vt.edu