Parallel Segmented Merge and Its Applications to Two Sparse Matrix Kernels

Weifeng Liu, Norwegian University of Science and Technology, Norway
Hao Wang, Ohio State University, USA
Brian Vinter, University of Copenhagen

June 8th, 2018, Bergen, Norway

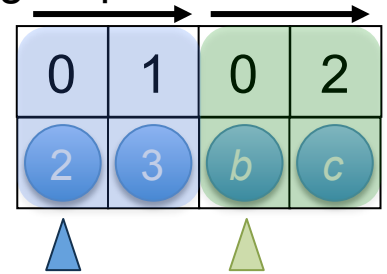
Background

1. Merge (sort) and its use
2. A definition of segmented merge
3. Merge in parallel

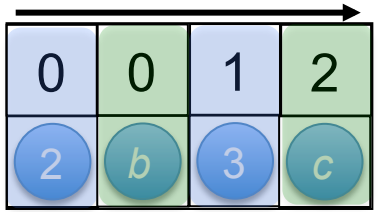
Merge (sort)

- We consider to merge two sorted key-value ascending/descending sequences into one. A serial algorithm first creates two pointers running along the two sequences, compares the entries pointed by them, and saves the smaller/larger one to the resulting sequence.

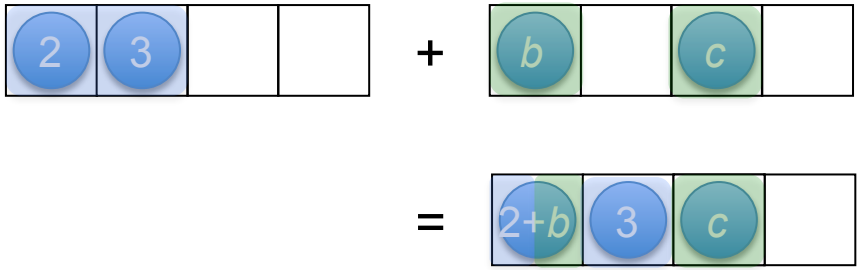
Two sorted sequences



One sorted resulting sequence

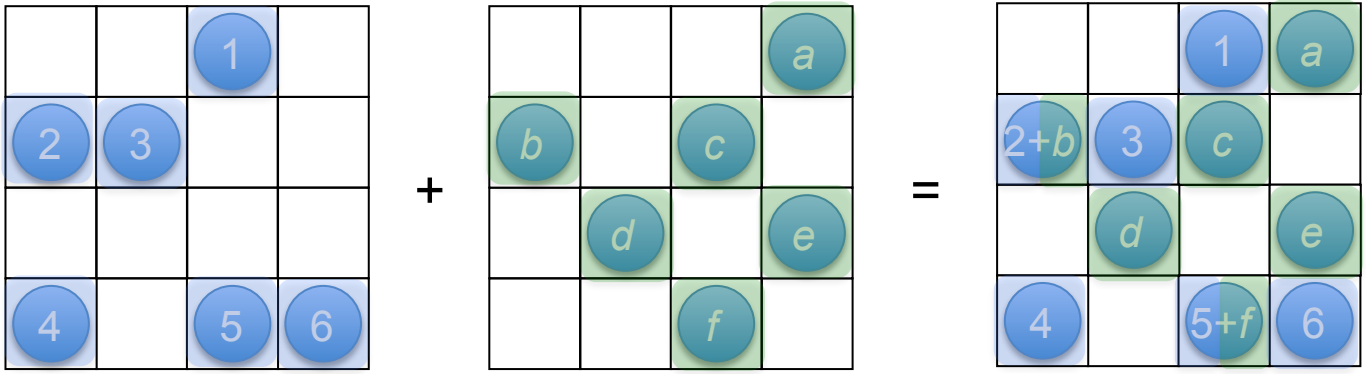


Applying the left merging to sparse vector-vector addition:



Use case 1: sparse matrix-matrix addition

- Add a sparse matrix *A* by a sparse matrix *B*, and obtain a resulting sparse matrix *C*.



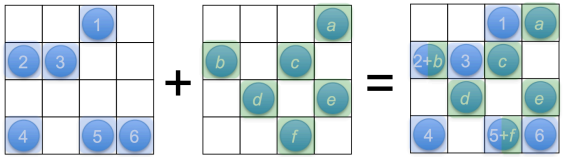
A (4x4) sparse, 6 nonzeros

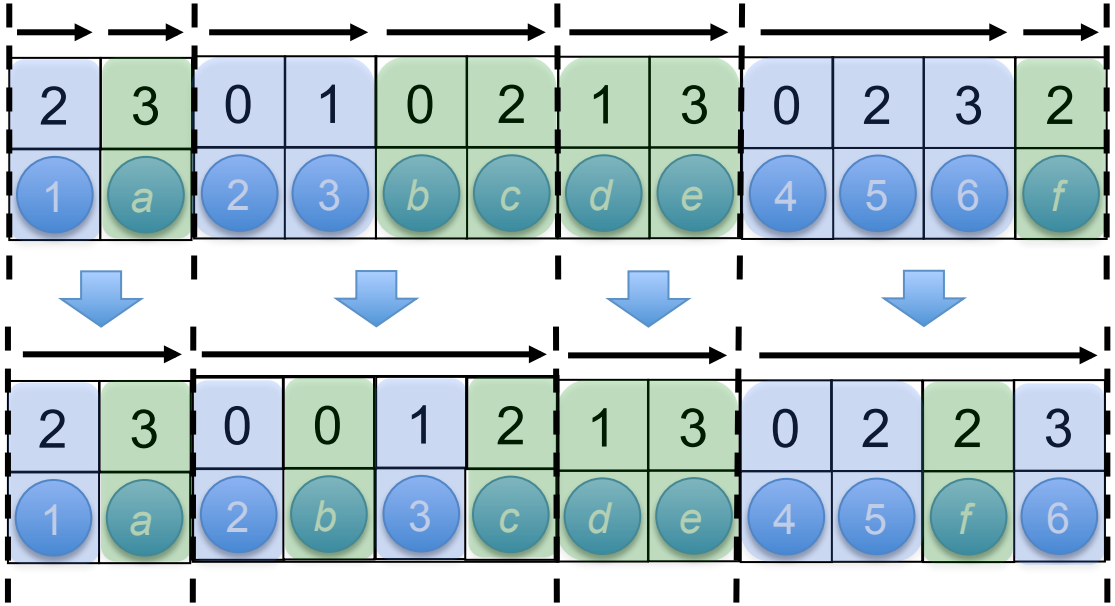
B (4x4) sparse, 6 nonzeros

C (4x4) sparse, 10 nonzeros

Use case 1: sparse matrix-matrix addition

- Each resulting row is merged from two rows.

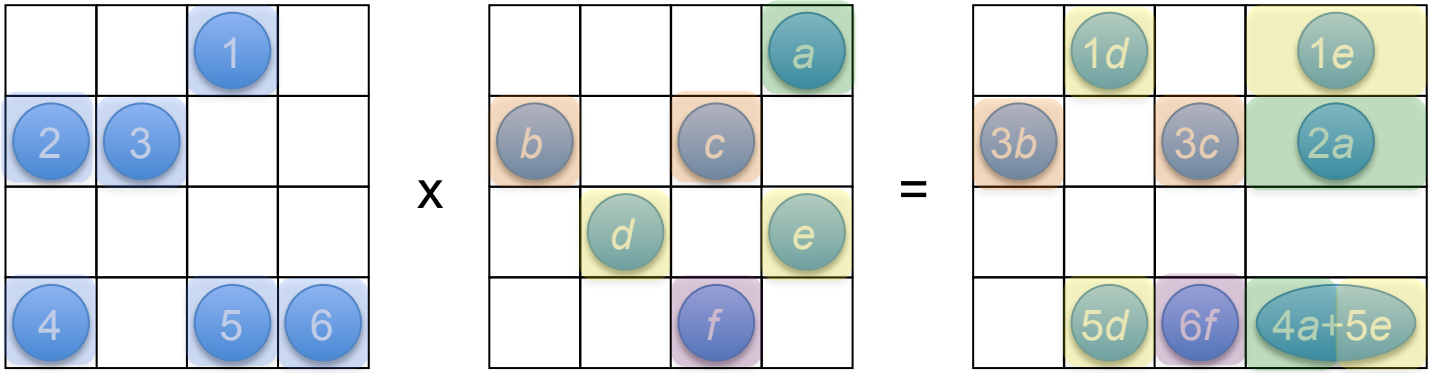




NTNU

Use case 2: sparse matrix-matrix multiplication

- Multiply a sparse matrix *A* by a sparse matrix *B*, and obtain a resulting sparse matrix *C*.



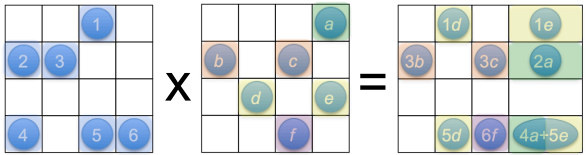
A (4x4) sparse, 6 nonzeros

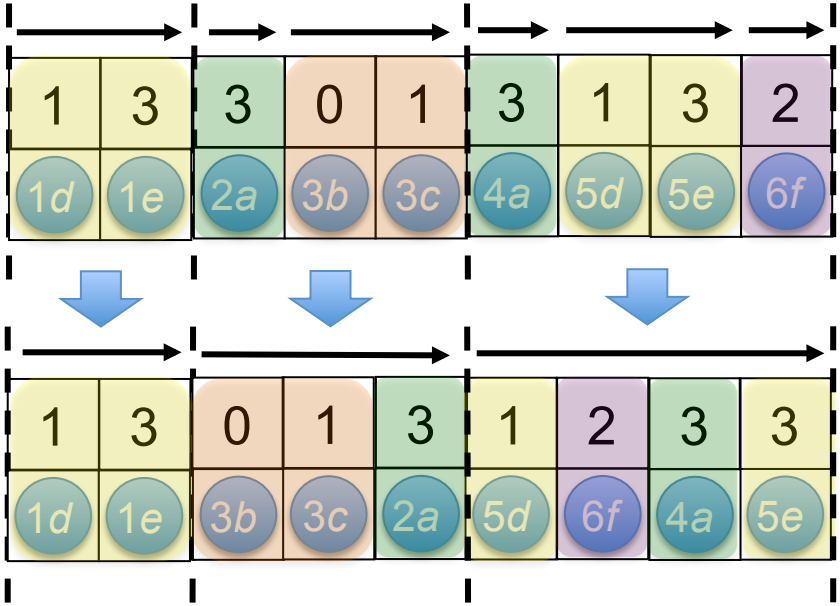
B (4x4) sparse, 6 nonzeros

C (4x4) sparse, 8 nonzeros

Use case 2: sparse matrix-matrix multiplication

- Each resulting row is merged from multiple rows.



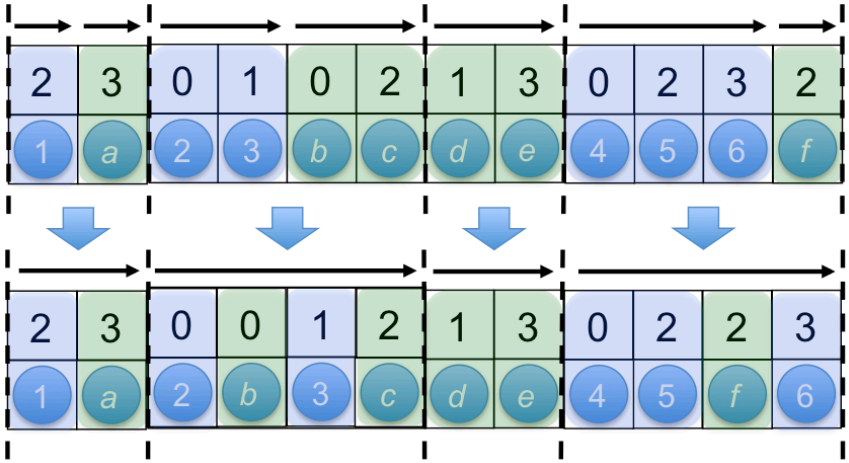


Background

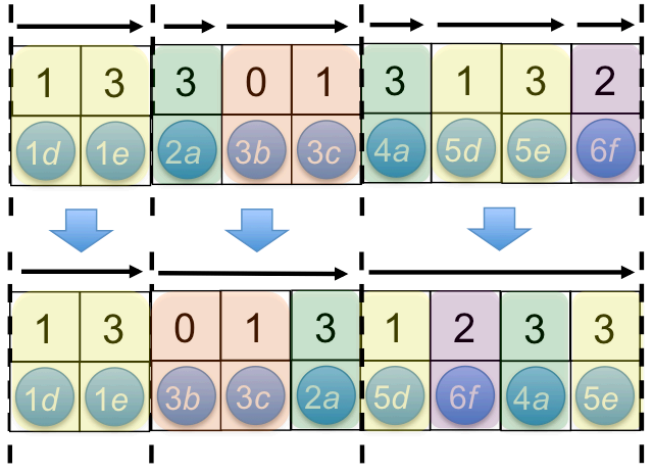
1. Merge (sort) and its use
2. A definition of segmented merge
3. Merge in parallel

A definition of segmented merge (segmerge)

- **Segmented merge** operation merges *q* sub-segments into *p* segments.



q = 7, *p* = 4



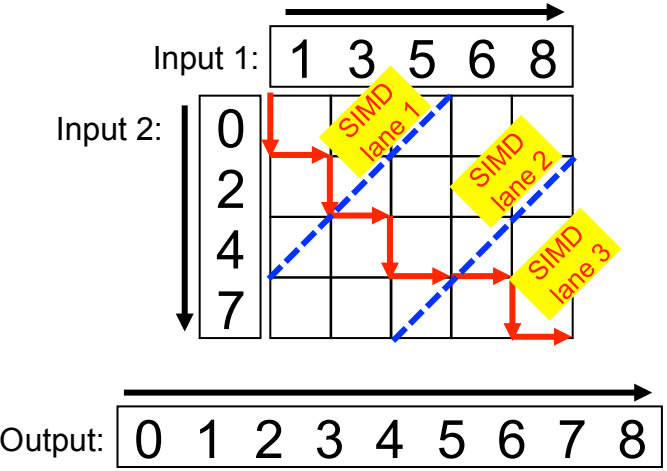
q = 6, *p* = 3

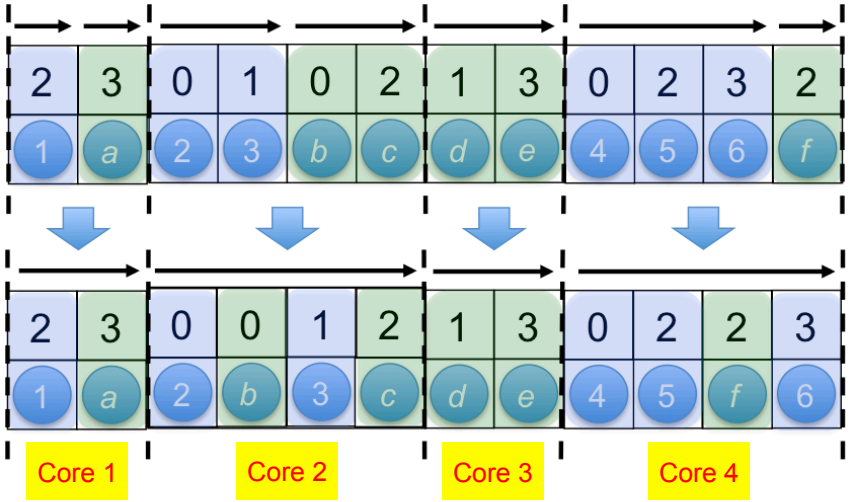
Background

1. Merge (sort) and its use
2. A definition of segmented merge
3. Merge in parallel

Merge in parallel

- There are two parallel merge methods:
  - (1) **mergeing two sub-segments**: merge-path works in a vectorized and balanced way,
  - (2) **mergeing multiple segments**: assigning each segment to a core





Though each segment can be merged in a balanced way, cores may have imbalanced workload due to very diverse segment sizes.

This motivates our segmerge algorithm.

Green, McColl, Bader. GPU Merge Path: a GPU Merging Algorithm. ICS '12.

NTNU

Parallel Segmented Merge

1. Algorithm description
2. Experiments: SpTRANS and SpGEMM
3. Micro-benchmarks

Overview of our segmerge

Step 1. create level-1 and level-2 pointers

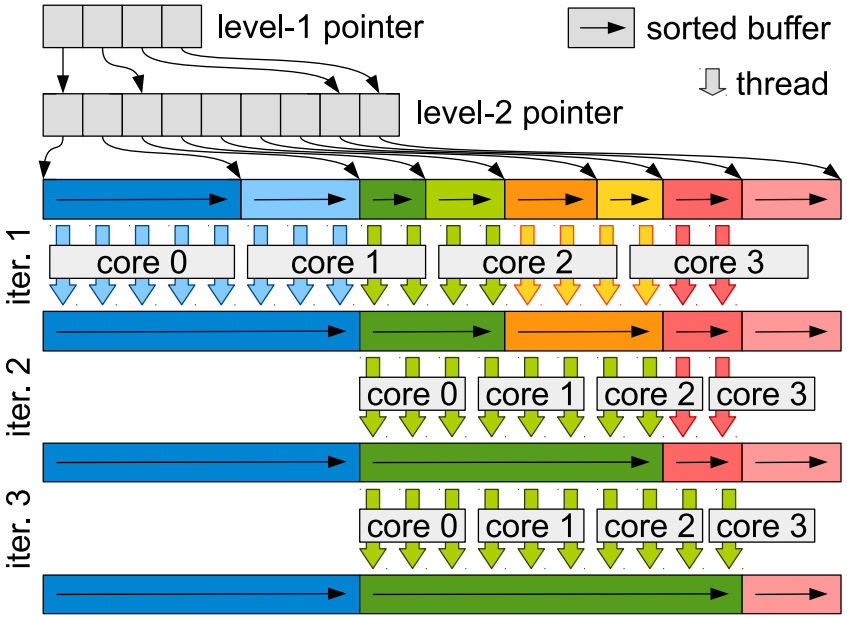
Step 2. fix each thread's workload (#entries) and compute #threads

Step 3. sub-segments scatter info to threads

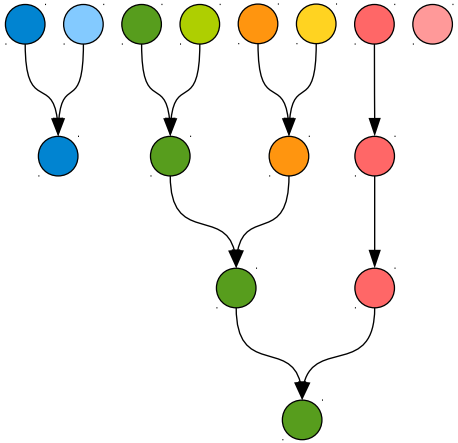
Step 4. threads gather info to find workload (similar to merge path)

Step 5. threads are evenly assigned to cores

Step 6. iterate steps 2- 5 until finish



Merge tree of the left segments

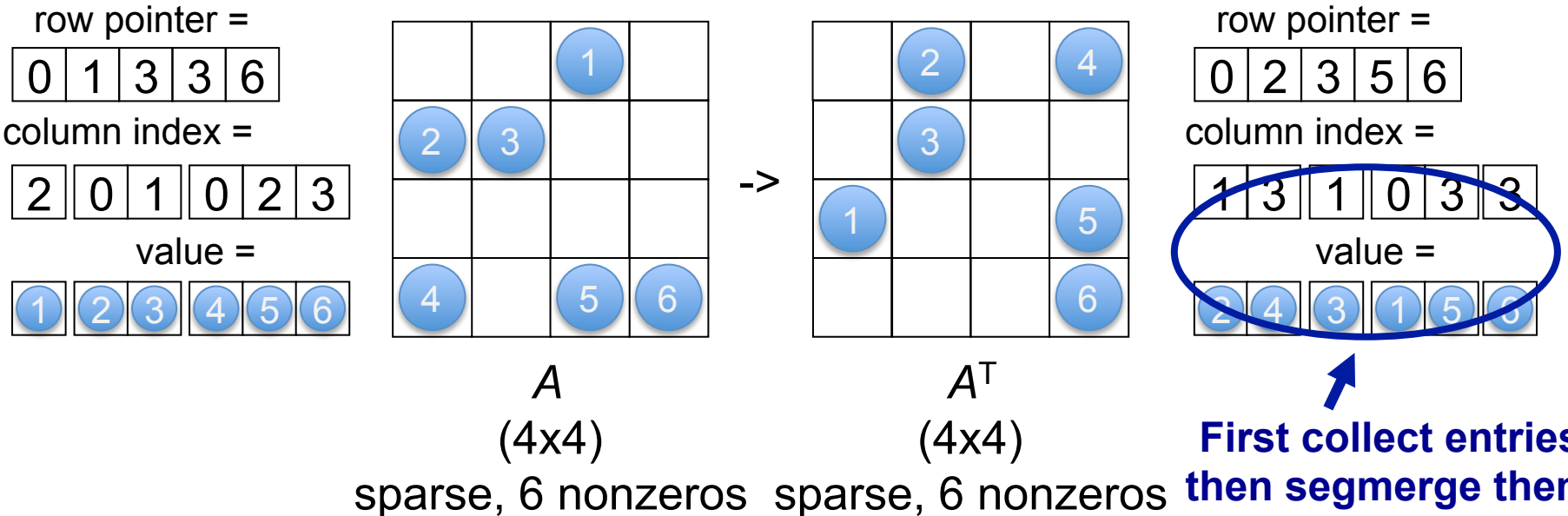


Parallel Segmented Merge

1. Algorithm description
2. Experiments: SpTRANS and SpGEMM
3. Micro-benchmarks

Experiment 1: sparse matrix transposition

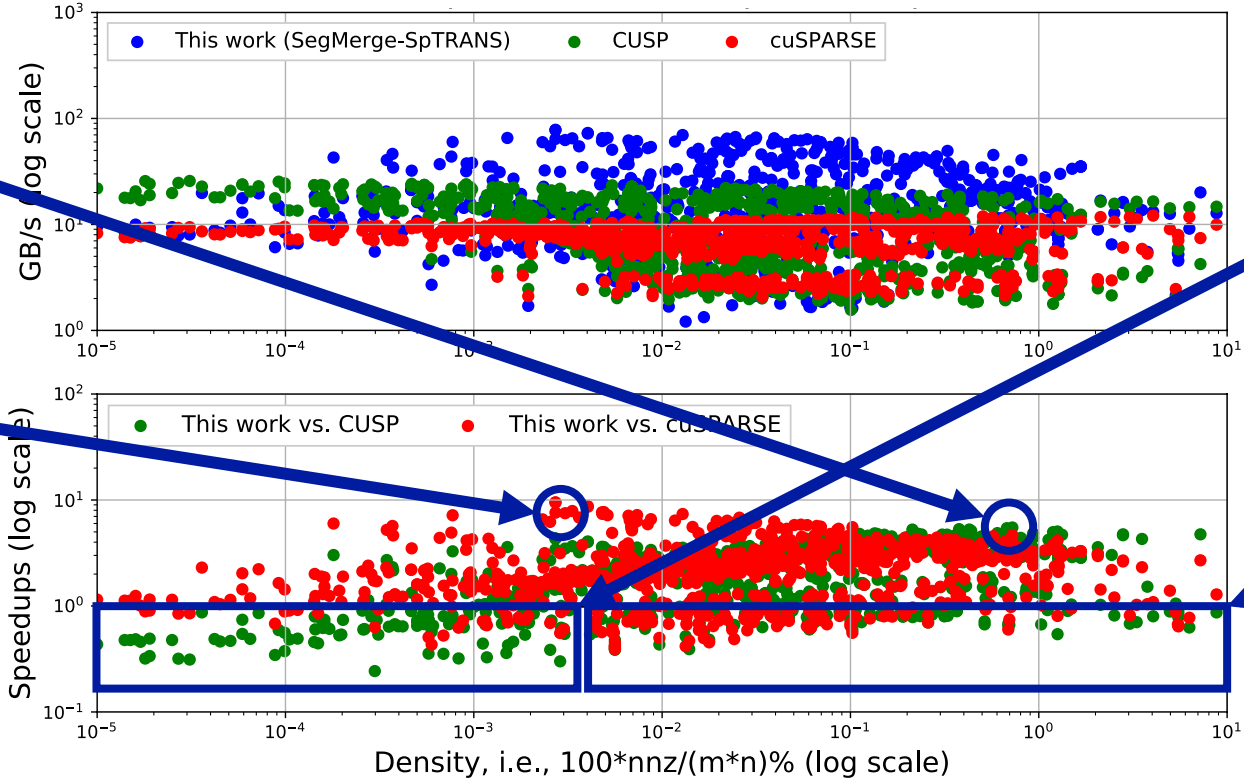
- Transpose a sparse matrix *A* (in CSR) to a sparse matrix *B* (i.e., A^T, in CSR). This is equivalent to a conversion from CSR to CSC, or vice versa.



Experiment 1: sparse matrix transposition

~7x over CUSP

~10x over cuSPARSE



CUSP behaves better when very sparse

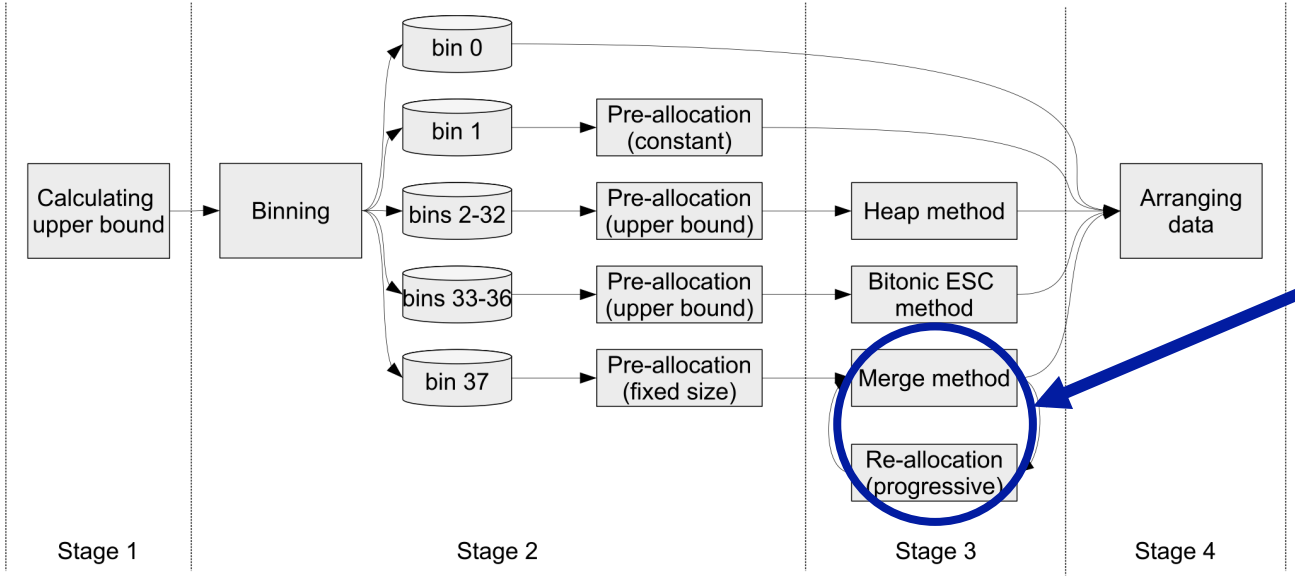
In some cases slower than CUSP and cuSPARSE

GPU: Titan X

#matrices: 956

NTNU

Experiment 2: sparse matrix-matrix multiplication



Only use segmerge for `long' rows

Weifeng Liu, Brian Vinter. **A Framework for General Sparse Matrix-Matrix Multiplication on GPUs and Heterogeneous Processors**. *JPDC. 2015.* (extended from *IPDPS14*)

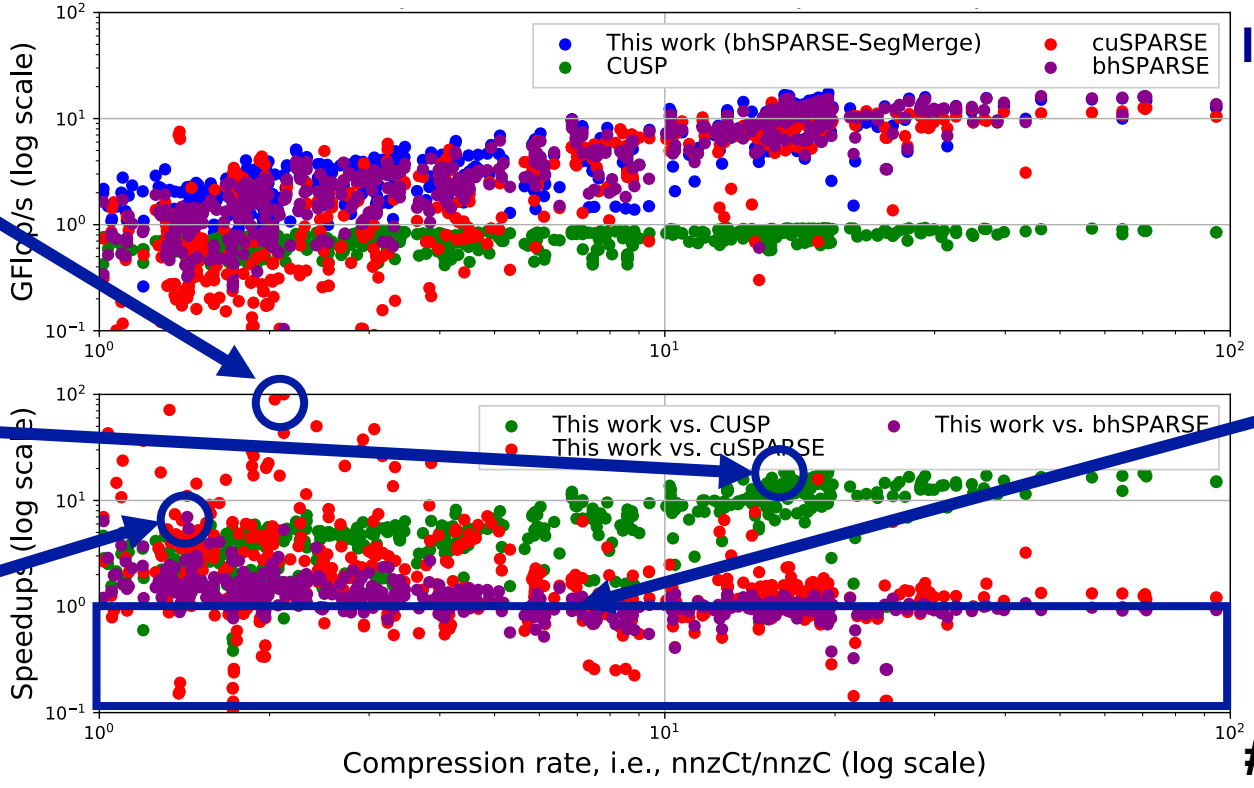
Weifeng Liu, Brian Vinter. **An Efficient GPU General Sparse Matrix-Matrix Multiplication for Irregular Data**. *IPDPS14*.

Experiment 2: sparse matrix-matrix multiplication

~100x over cuSPARSE

~20x over CUSP

~7x over bhSPARSE



In some cases, cuSPARSE is faster, and bhSPARSE rarely gives higher perf.

GPU: Titan X

#matrices: 956

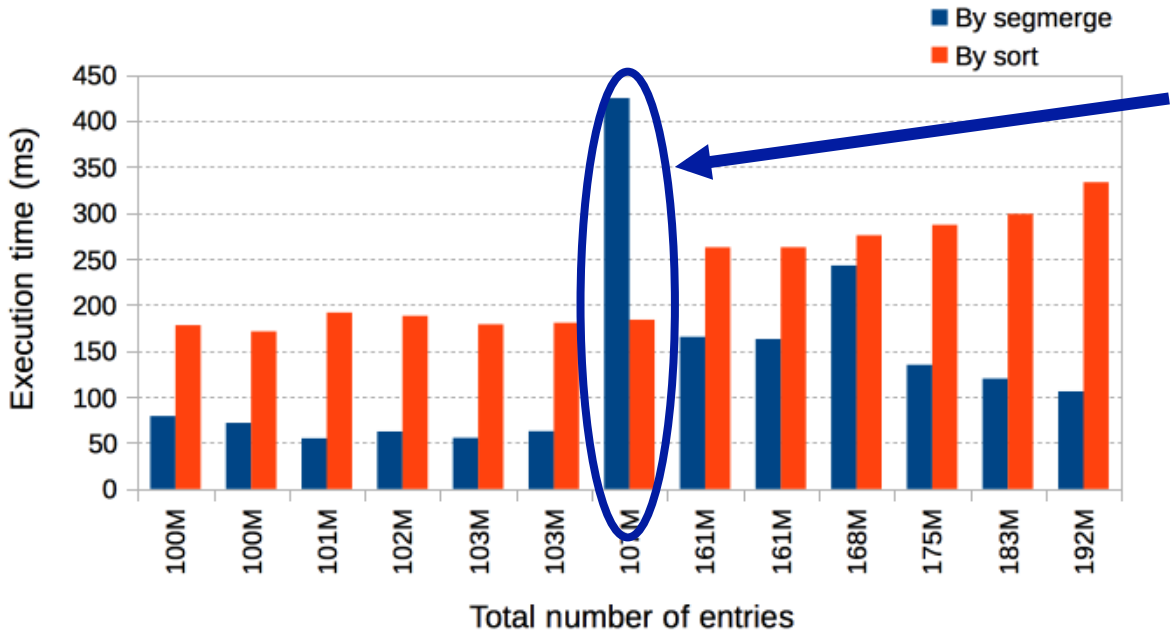
NTNU

Parallel Segmented Merge

1. Algorithm description
2. Experiments: SpTRANS and SpGEMM
3. Micro-benchmarks

Micro-benchmark 1: segmerge vs sort

- In the SpGEMM test, if we replace segmerge with CUDA thrust-sort:

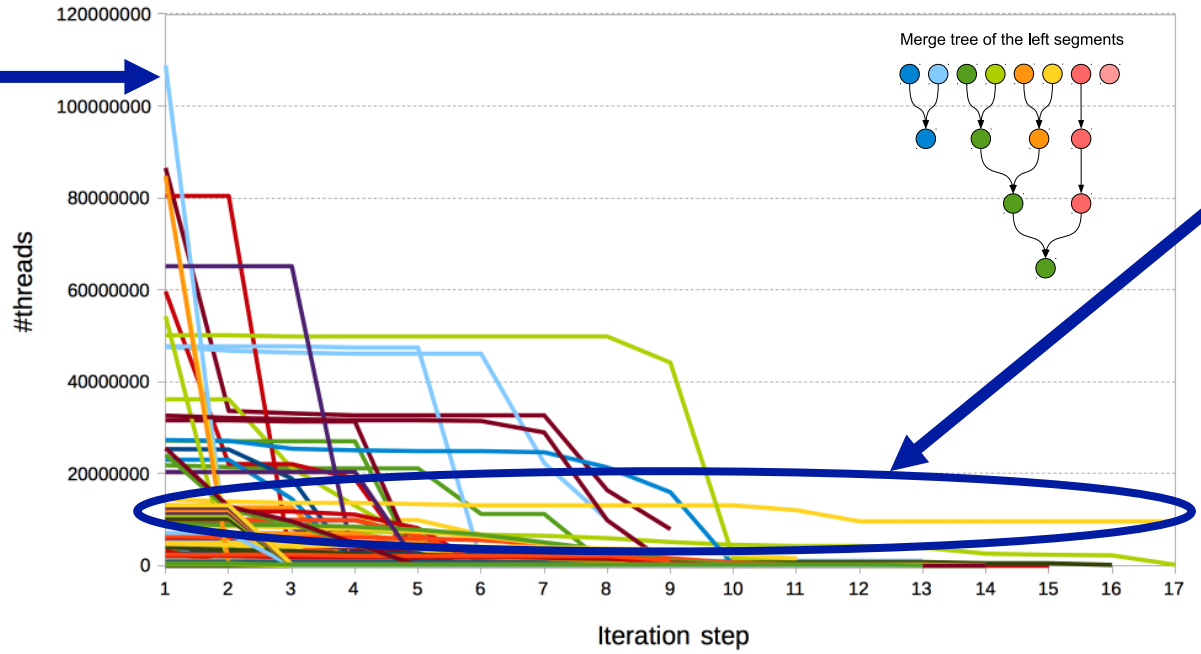


In this case, sort is faster than segmerge

Micro-benchmark 2: #threads in iterations

- In the SpGEMM test, we record #threads issued in each iteration step:

108813822
552638
547690
515739
349187
32396
12215
6863
5987
5971
5963
5959
5956



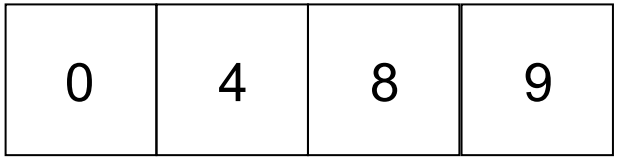
14406767
13862538
13592550
13520724
13306832
13139154
13130214
13125706
13123376
13121288
11973924
9579396
9579268
9579188
9579132
9579116
9579088

NTNU

Conclusion

- Though some parallel segmented operations (segsum[Blelloch TR93, Liu PARCO15], segsort [Hou ICS17], etc.) exist, parallel segmerge has been largely ignored.
- Our segmerge algorithm always evenly distributes entries to threads, thus runs in a very balanced way on GPU.
- SpTRANS and SpGEMM are largely accelerated by using segmerge at proper stages.
- According to micro-benchmarks, there are still some optimizations can be done.

T k u !



A y Q s n s ?