Cache-aware Sparse Matrix Formats for Kepler GPU

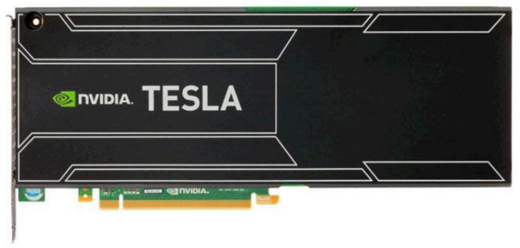
Yusuke Nagasaka, Akira Nukada, Satoshi Matsuoka
Tokyo Institute of Technology

Sparse Matrix

- Generated by FEM, being as the graph data
  - Often require solving sparse linear equation fast
- Iterative method : CG method, BiCG method
  - Level-1 BLAS (Dot product + AXPY)
    - Sequential memory access
  - Sparse matrix vector multiplication (SpMV)
    - Using sparse matrix format
    - Random memory access

      Performance depends on cache hit rate

SpMV computation on GPU

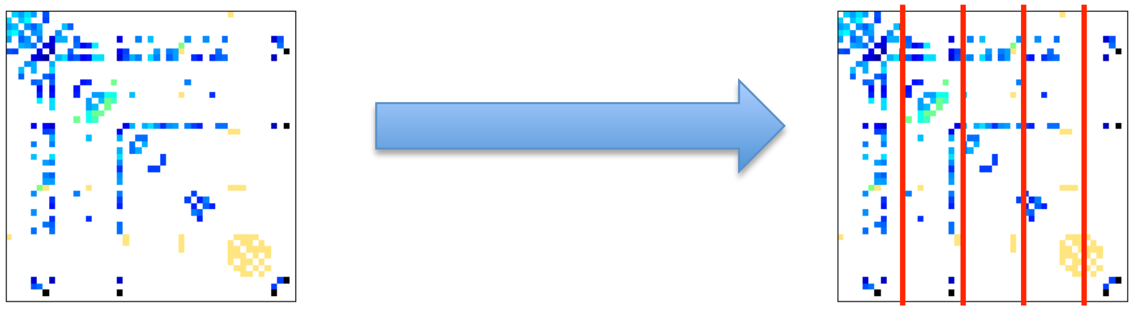


- High memory bandwidth and parallelism enable high performance
- Latency is hidden with SMT
- Available cache per thread is small
  - Controlling the cache is difficult
  - => Lower cache hit rate compared to CPU

	Intel Xeon Processor E5-2620 v2	NVIDIA Tesla K20X
Cache size	L1 cache : 192KB (instruction / data) L2 cache : 1.5MB L3 cache : 15MB	Read-only cache : 12KB * 4 / SMX L2 cache : 1.5MB
Max threads	12 threads	28672 threads

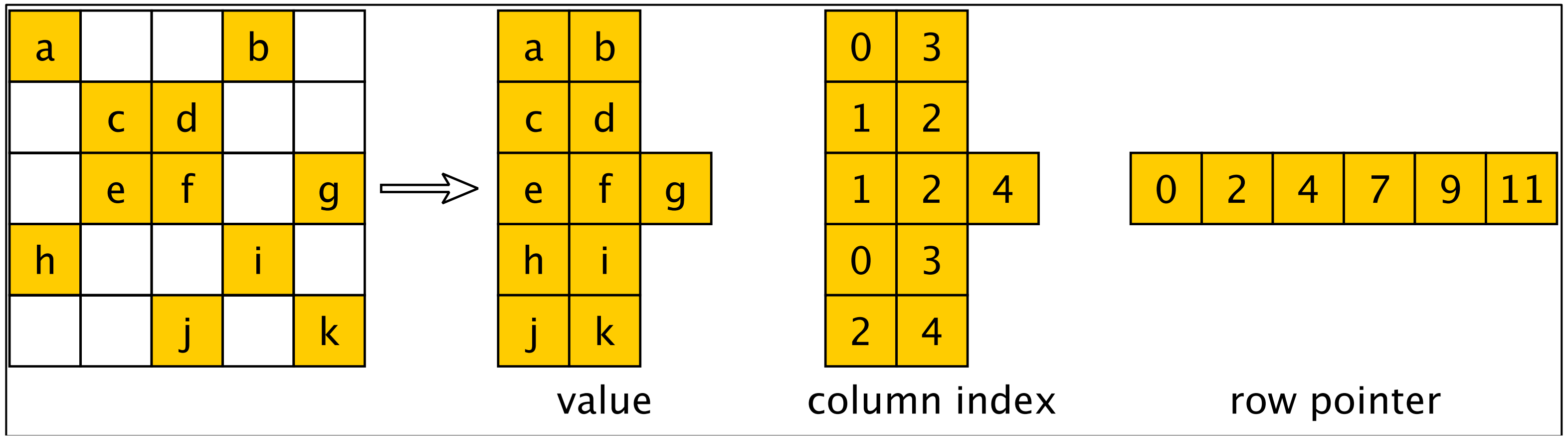
Contribution

- We propose a family of cache-aware formats for GPU
  - Segmentation along the column
  - Segmented formats, Non-Uniformly Segmented formats
    - SpMV computation consists in 2 phases
  - Achieve speedups of up to
    - x2.0 for real datasets in SpMV
    - x3.0 for the random matrices in SpMV
    - x1.2 for real datasets in CG
    - x1.68 for multi-node CG



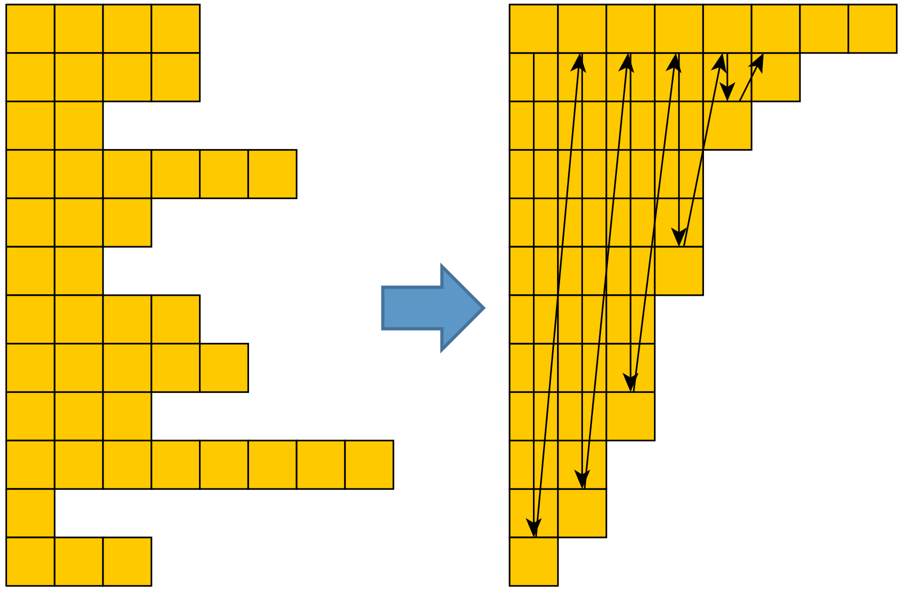
Sparse Format

- Compressing the needless zero elements
  - Reduce memory usage
  - Eg.) COO, CSR
- Efficient memory access to matrix data depends on architecture
  - Vector machine, GPU : column major format
    - JDS, ELLPACK, SELL-C-σ



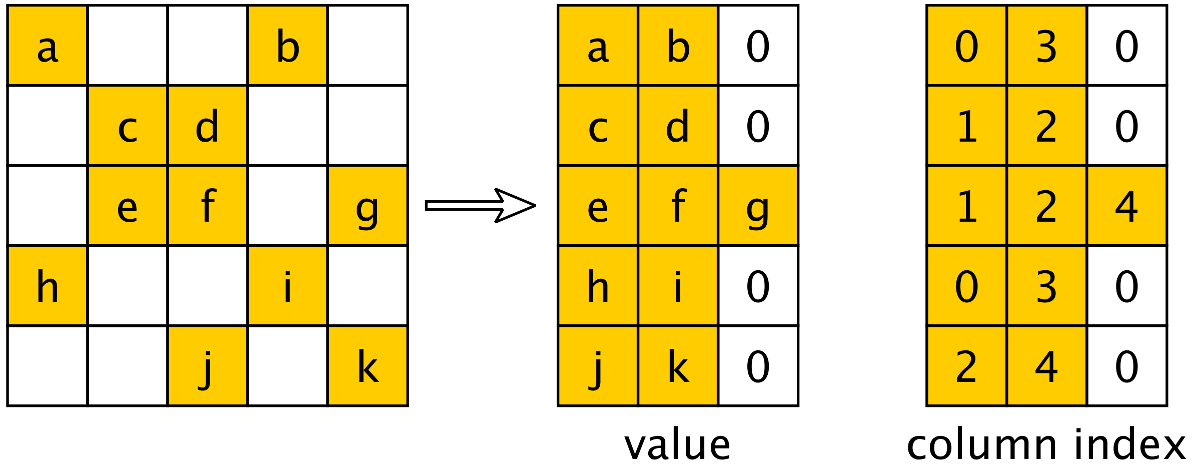
(Existing Sparse Format) JDS

- Reordering the rows by the number of non-zero elements per row
  - Generate column major format
    - Favorable for vector machine and many core architectures



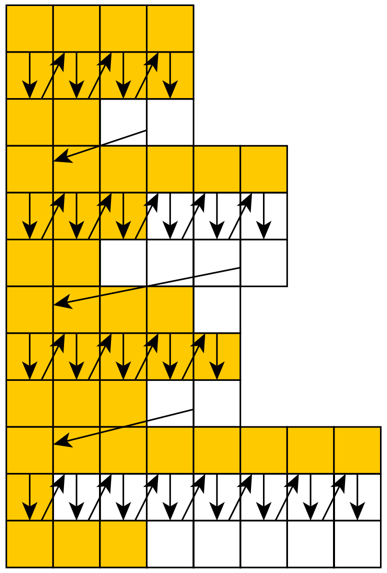
(Existing Sparse Format) ELLPACK

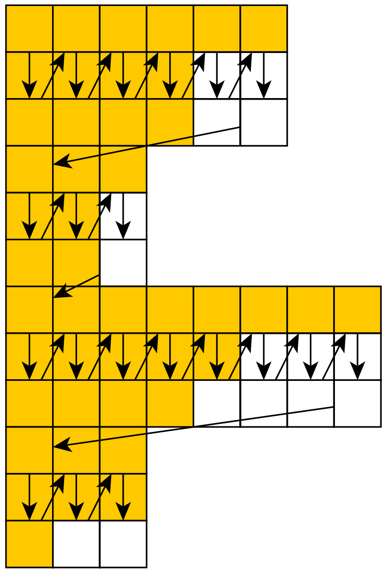
- Stored in column major ordering
- Number of elements is same for all rows
  - For smaller rows, zeros are filled in
  - Large variance of the number of elements per row
    - => Waste memory usage and increase additional computations



(Existing Sparse Format) SELL-C-σ [Kreutzer, 2013]

- Converting ELLPACK each row block (Sliced ELLPACK)
  - Reduce the zero filling
  - C is block size
    - C = WARP size
- Sorting each σ rows
  - Tradeoff between the zero fill and the cost of sorting


SELL-3-1


SELL-3-6

Cache Hit Rates of Existing Sparse Formats

- NVIDIA Tesla K20X
- The dataset is taken from the University of Florida Sparse Matrix Collection
- JDS (Reordering) format
  - Read-only cache is assigned to input vector cache
  - Coalesced access to matrix data

Matrix	Size	L2 Cache Hit Rate [%]	Read-only Cache Hit Rate [%]
Audikw_1	943,695	82.864	51.420
Crankseg_2	63,838	98.338	66.540
mouse_gene	45,101	99.912	8.298

PROPOSAL FORMATS

Column size and cache hit rate

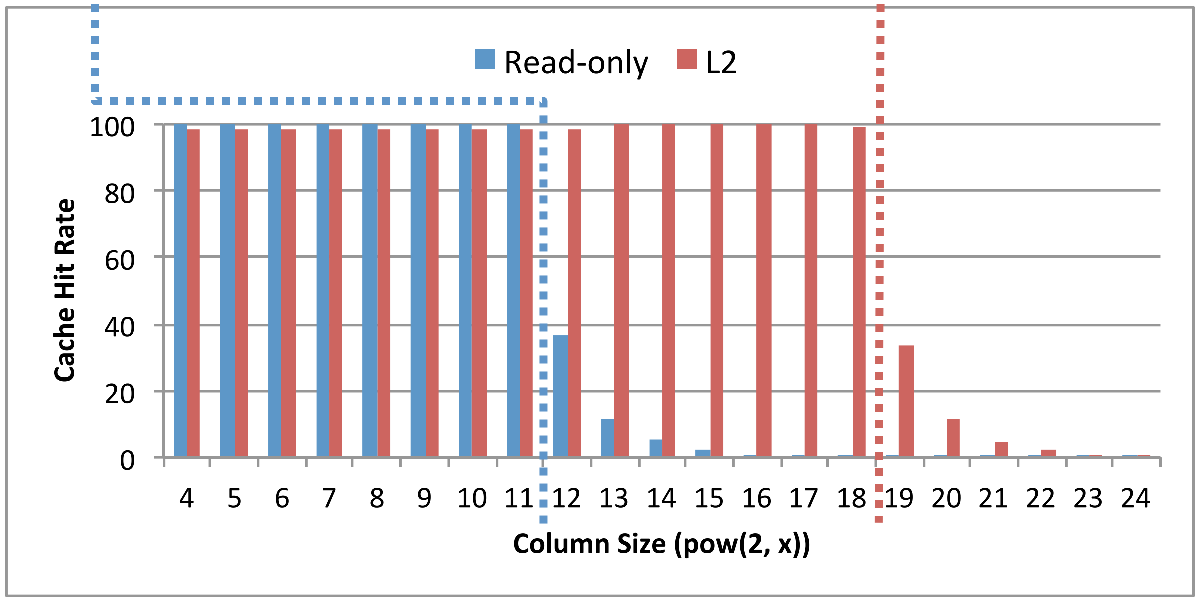
Column size where the cache hit rate drops corresponds to each cache size

- Read-only cache : 12KB - L2 cache : 1.5MB

⇒ Segmenting the matrix and the input vector enable to achieve high cache hit rate

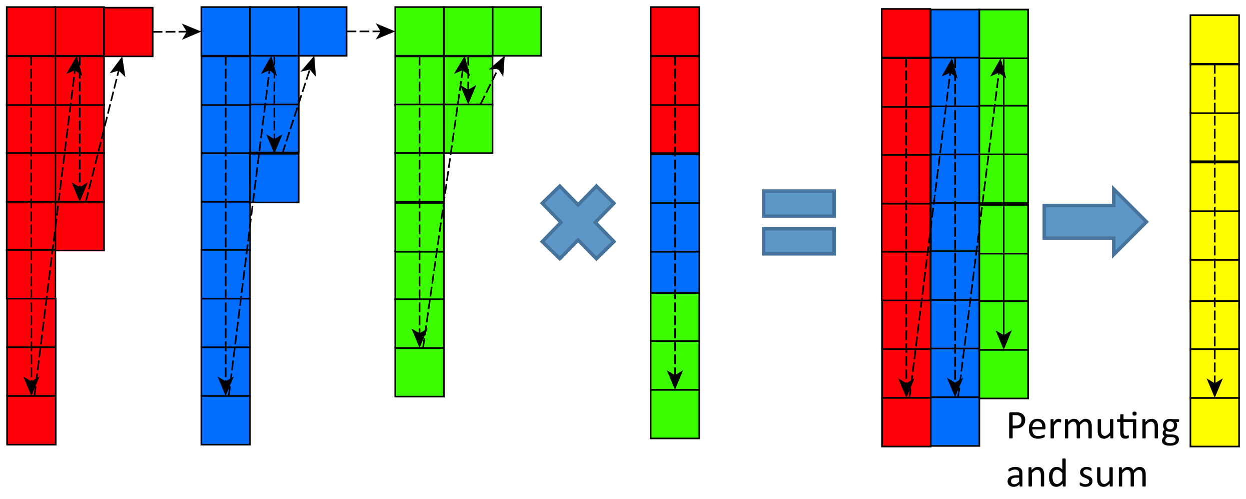
– Single precision

– Using JDS format



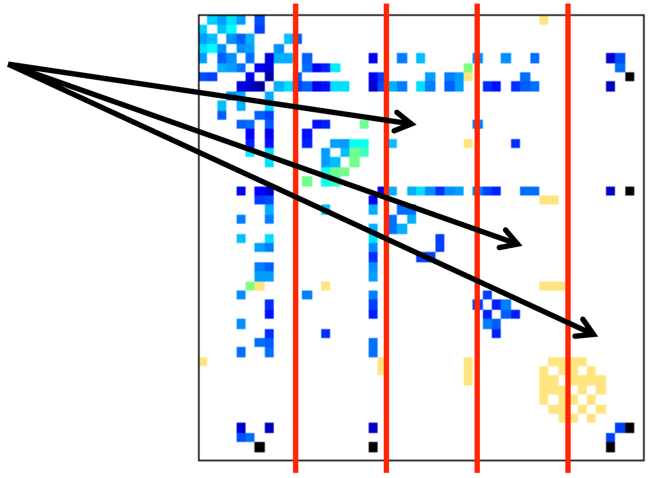
Segmented Formats

- Column-wise segmentation
  - Each segment is converted to JDS or SELL-C-σ
- SpMV computation consists of 2 phases
  - 1st phase : Computing SpMV for each sub-matrix and sub-vector, and storing the result into the memory
  - 2nd phase : Accumulation of the intermediate vectors



Segmented Formats disadvantages

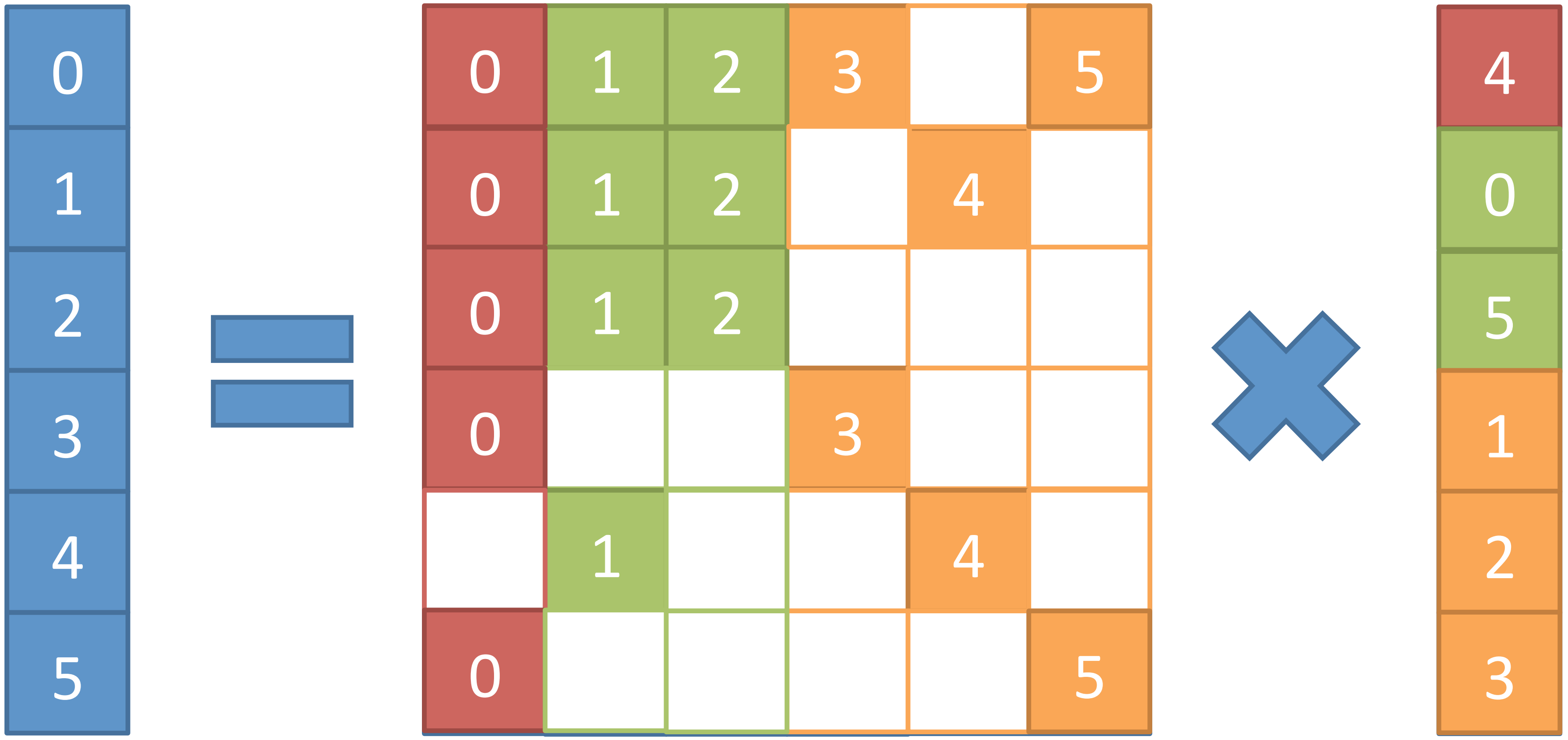
- Increase memory accesses
  - Sequential memory write in 1st phase
  - Random memory read in 2nd phase
- Generate the segments having few non-zero elements
  - Improvement of reusability < Overhead of segmenting
  - => Low efficiency



Non-Uniformly Segmented Formats (NUS Formats)

- Mixing the multi level segmentation size
  - Large segmentation width for the low density area
  - => Reduce the number of segments
- Sorting by the number of non-zero elements of column
  - Set the high density column to left side and high reusability vector elements to the top

Converting NUS Format



Matrix index : column index   Vector index : original row index

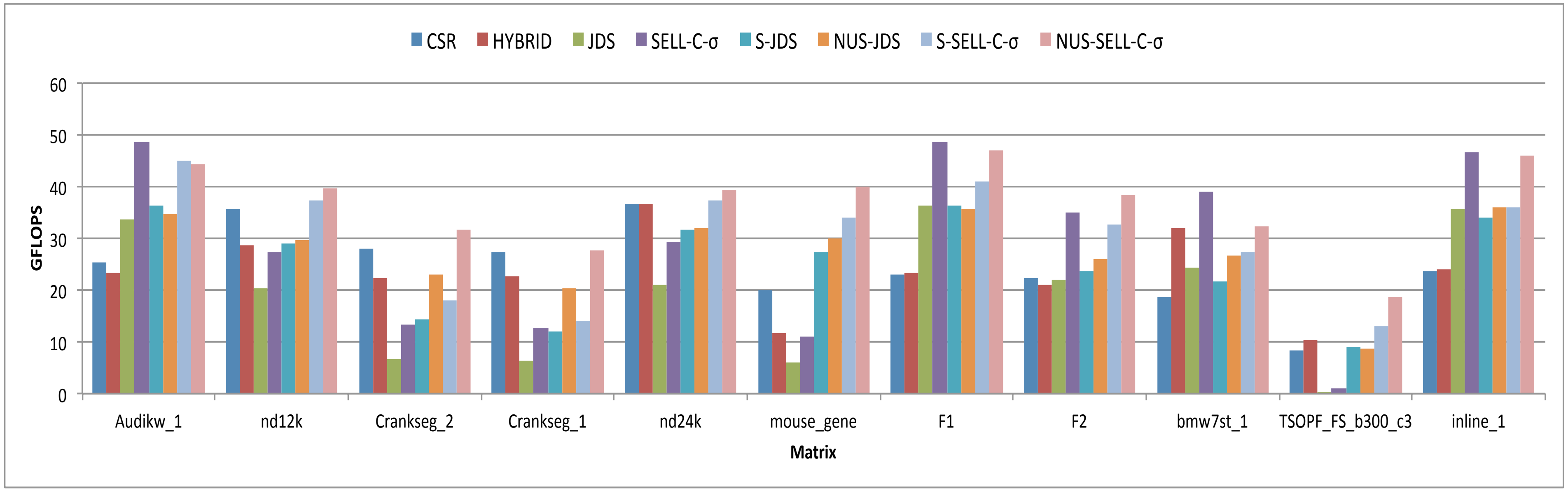
PERFORMANCE EVALUATION

Experiment Environment

- TSUBAME-KFC
  - CPU : Intel Xeon E5-2620 v2 2.10GHz x 2
  - GPU : NVIDIA Tesla K20X x 4
    - Single precision peak performance : 3.95 [TFLOPS]
    - Bandwidth : 250 [GB / sec]
    - Memory size : 6 [GB]
    - L2 cache : 1.5 [MB]
    - Read-only cache : 12 * 4 [KB / SMX]
  - OpenMPI 1.7.2
  - FDR InfiniBand network
- CUDA 5.5
- cuSPARSE : provided by NVIDIA
  - CSR format, HYBRID format

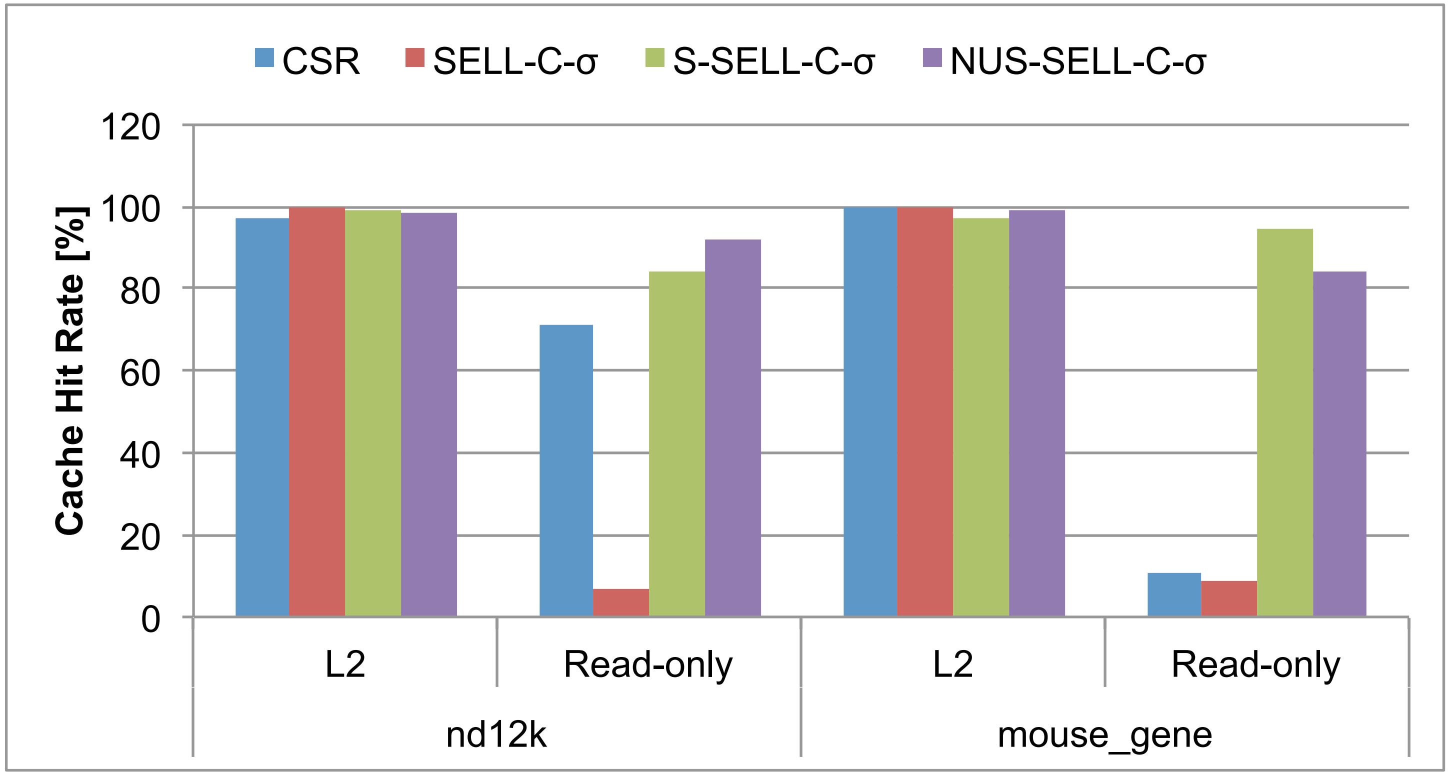
Performance Evaluation SpMV (Florida data sets)

- CSR and (NUS-)SELL-C-σ show good performance
- Our formats show
  - speedup of x0.83 ~ x2.01 compared to non-segmented format
  - Stable performance



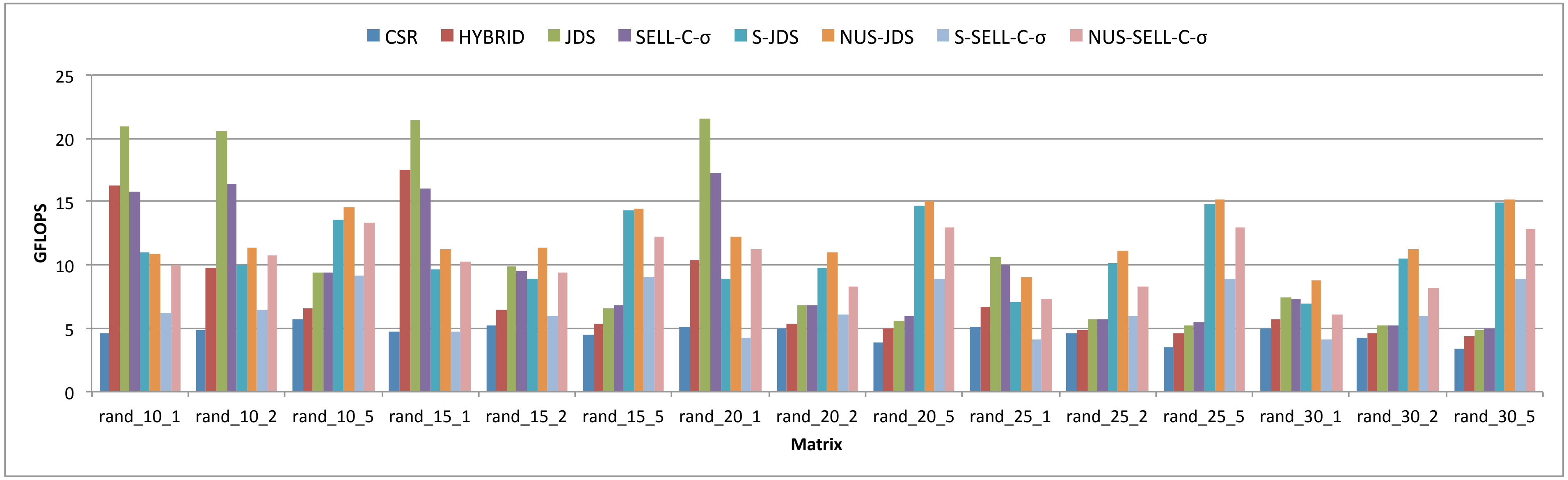
Performance Evaluation
Cache Hit Rate of SpMV

- Segment size suits to read-only cache
  - Improvement of cache hit rate from non-segmented formats



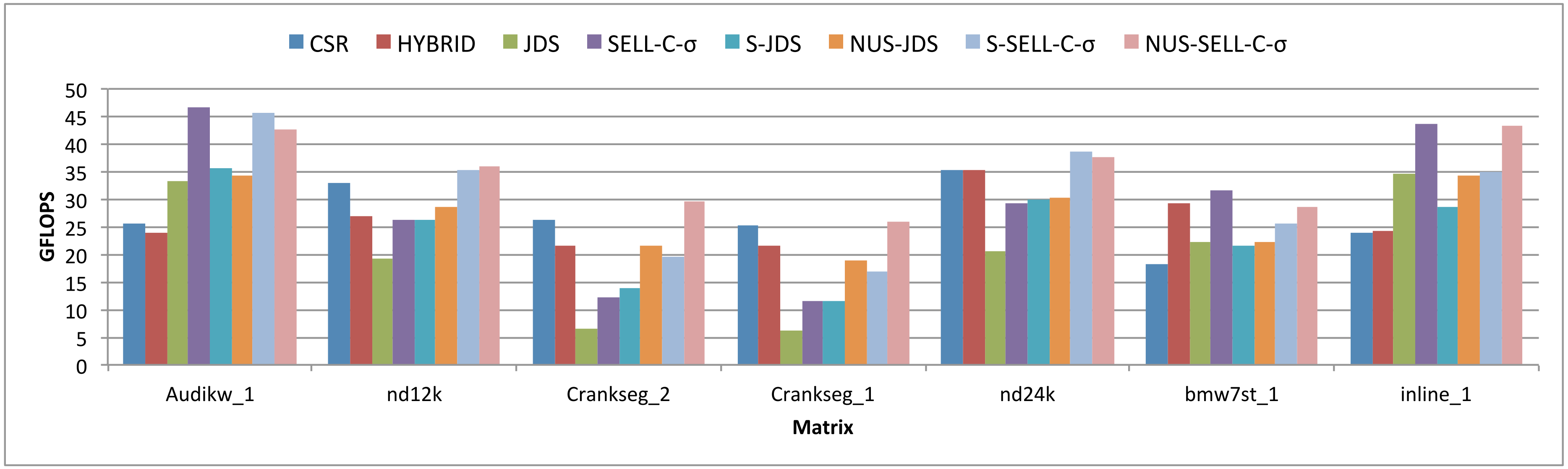
Performance Evaluation SpMV (Randomly generated matrix)

- Investigating larger matrices
  - Number of rows : 1 M ~ 3 M
  - Non-zero density : 0.0001%, 0.0002%, 0.0005%
- Speedup of up to x3.0
  - Our formats become better choice in denser matrix



Performance Evaluation Conjugate Gradient method

- CG computation for positive definite matrices
  - Similar performance improvement to SpMV
  - Speedup of x1.2

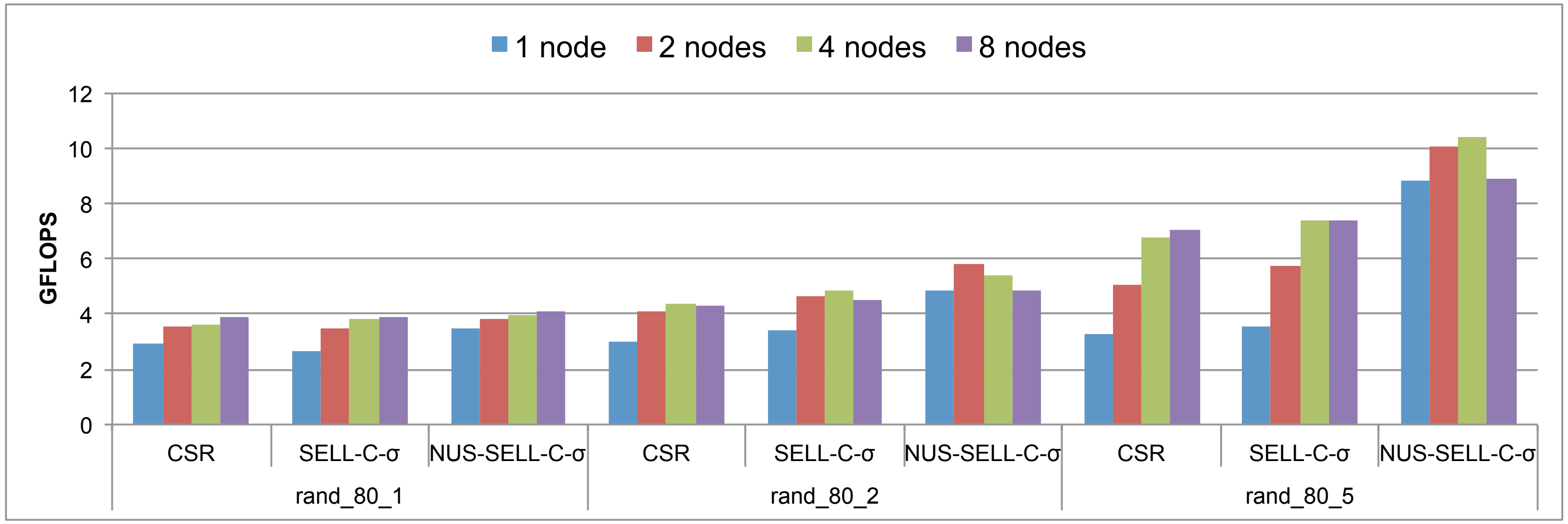


Performance Evaluation
Multi-node CG method

- Strong scaling
  - Assign row block to each node
    - Each row block has fewer non-zero elements
    - => Cause performance degradation
- Generate larger random matrices
  - Row size : 8 M
  - Non-zero density : 0.0001%, 0.0002%, 0.0005%

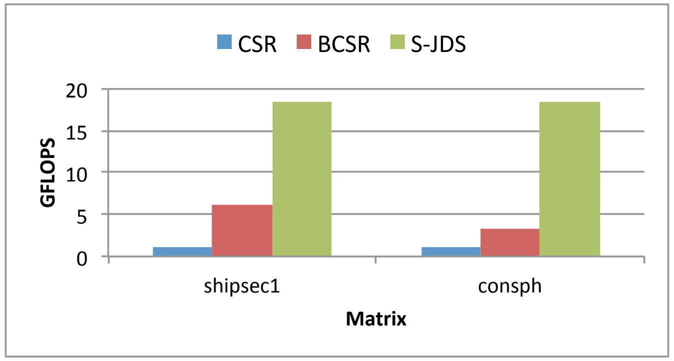
Performance Evaluation
Multi-node CG method

- NUS-SELL-C-σ shows superiority to CSR and SELL-C-σ
  – Speedup of up to x1.68
  – In lower density matrix, data transfer time between nodes takes relatively longer
    - Performance difference between formats is not noticeable



Related Works

- 2D cache blocking for SpMV
  - For CPU
    - Eun-Jin Im et al. (International Journal of High Performance Computing Applications, 2004)
    - Improving the locality of input and output vector
    - Controlling the cache is easier compared to GPU
  - for GPU : BCSR format
    - Weizhi Xu et al. (SNPD 2012)
    - Synchronization for each column block
    - Large overhead of synchronization
      - SJDS show better performance



Related Works (cont'd)

- Blocked format focusing on load balancing
  - ELLPACK sparse block format
    - Liu et al. (ICS'13)
    - Target architecture : Intel MIC
    - No matrix reordering
  - Blocked format for GPU : BRC format
    - Ashari et al. (ICS'14)
    - Set the block size without considering the cache size

Conclusion

- Segmented formats and Non-Uniformly Segmented formats using column-wise segmentation improve the cache hit rate and SpMV performance
- NUS formats achieved speedups of up to
  - x2.0 for real data set in SpMV
  - X3.0 for randomly generated matrix in SpMV
  - X1.2 for real data set in CG
  - x1.68 for multi-node CG

Future Work

- Future work
  - Applying the format to other devices
    - Intel MIC, AMD Radeon GPU, Multi-core CPU
  - Performance modeling
    - Enable to select best format, segment size and the number of segments
  - Evaluation of the cost of format converting