



*the 17th International Symposium on Advanced
Parallel Processing Technology*



ARROW: Adaptive Row Reorganization for Warp-Balanced SpMV

Yuxiang Zhang, **Jianhua Gao**, Weixing Ji

July 30, 2026

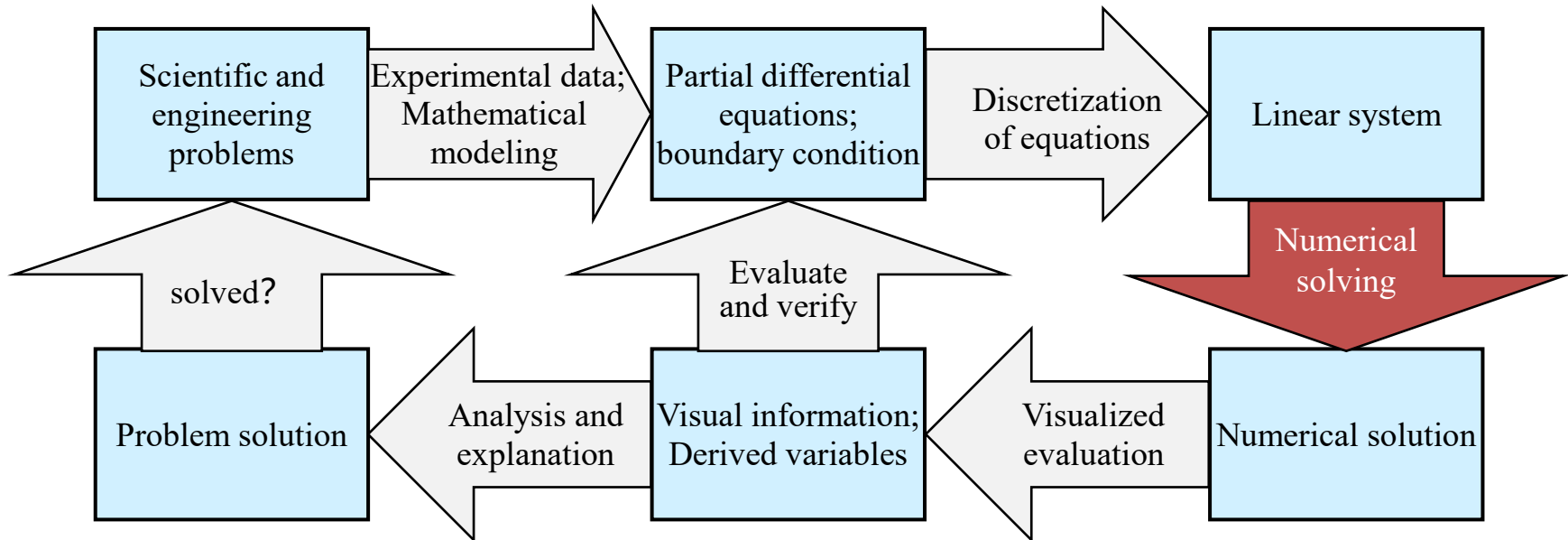


Beijing Normal University

- Background and Motivation
- Design and Implementation
- Experimental Evaluation
- Conclusion

- Background and Motivation
- Design and Implementation
- Experimental Evaluation
- Conclusion

- In the solving of many scientific and engineering problems, numerical solving is one of the most dominant steps.



General solving workflow of scientific and engineering problems

- Conjugate Gradient (CG) and Generalized Minimum Residual (GMRES) methods are most popular iterative algorithms
- SpMV is one of the most dominant operators.

ALGORITHM 1: Conjugate gradient algorithm

Input: coefficient matrix A , right-hand-side vector b , initial guess x_0 , maximum iterations MAX

Output: approximate solution $\hat{x}$

```

1  $r_0 = b - Ax_0$ ;
2  $rOld = r_0^T r_0$ ;  $p_0 = r_0$ ;
3 for ( $i = 1$ ; ( $\sqrt{rOld} \geq \|b\|\epsilon$ ) && ( $i \leq MAX$ );  $i++$ ) do
4    $y_i = Ap_{i-1}$ ;
5    $\alpha = \frac{rOld}{(p_{i-1}^T y_i)}$ ;
6    $x_i = x_{i-1} + \alpha p_{i-1}$ ;  $r_i = r_{i-1} - \alpha y_i$ ;
7    $rNew = r_i^T r_i$ ;  $\beta = \frac{rNew}{rOld}$ ;  $rOld = rNew$ ;
8    $p_i = r_i + \beta p_{i-1}$ ;
9 end
10  $\hat{x} = x_{i-1}$ ;
```

Algorithm 1: Restarted GMRES algorithm.

Input: Sparse matrix A , initial guess x_0 , right-hand-side vector b , relative residual threshold ϵ

Output: approximate solution $\hat{x}$

```

1  $r_0 = b - Ax_0$ ;
2  $\beta = \|r_0\|$ ;  $v_1 = r_0/\beta$ ;
3 for  $j = 1, 2, \dots, m$  do // Arnoldi process
4    $w_j = Av_j$ ; // SpMV in inner iteration
5   for  $i = 1, 2, \dots, j$  do
6      $H_{ij} = (w_j, v_i)$ ;  $w_j = w_j - H_{ij}v_i$ ;
7   end
8    $H_{j+1,j} = \|w_j\|$ ;
9   if  $H_{j+1,j} == 0$  or Stopping criteria satisfied then
10     $m = j$ ; break;
11  end
12   $v_{j+1} = w_j/H_{j+1,j}$ ;
13 end
14  $V_m = [v_1, \dots, v_m]$ ;
15 find  $y^m$  that minimizes  $\| \beta e_1 - H_m y \|$ ;
16  $x_m = x_0 + V_m y^m$ ;
17  $r_m = b - Ax_m$ ; // SpMV in outer iteration
18 if  $\|r_m\| \leq \epsilon \|b\|$  then
19    $\hat{x} = x_m$ ; break;
20 else // Restart
21    $r_0 = r_m$ ;  $x_0 = x_m$ ; Goto 2;
22 end
```

- Conjugate Gradient (CG) and Generalized Minimum Residual (GMRES) methods are most popular iterative algorithms
- SpMV is one of the most dominant operators.

ALGORITHM 1: Conjugate gradient algorithm

Input: coefficient matrix A , right-hand-side vector b , in maximum iterations MAX

Output: approximate solution $\hat{x}$

```

1  $r_0 = b - Ax_0$ ;
2  $rOld = r_0^T r_0$ ;  $p_0 = r_0$ ;
3 for ( $i = 1$ ; ( $\sqrt{rOld} \geq \|b\|\epsilon$ ) && ( $i \leq MAX$ );  $i++$ ) do
4    $y_i = Ap_{i-1}$ ;
5    $\alpha = \frac{rOld}{(p_{i-1}^T y_i)}$ ;
6    $x_i = x_{i-1} + \alpha p_{i-1}$ ;  $r_i = r_{i-1} - \alpha y_i$ ;
7    $rNew = r_i^T r_i$ ;  $\beta = \frac{rNew}{rOld}$ ;  $rOld = rNew$ ;
8    $p_i = r_i + \beta p_{i-1}$ ;
9 end
10  $\hat{x} = x_{i-1}$ ;
```

Algorithm 1: Restarted GMRES algorithm.

Input: Sparse matrix A , initial guess x_0 , right-hand-side vector b , relative residual threshold ϵ

Output: approximate solution $\hat{x}$

```

1  $r_0 = b - Ax_0$ ;
2  $\beta = \|r_0\|$ ;  $v_1 = r_0/\beta$ ;
3 for  $j = 1, 2, \dots, m$  do // Arnoldi process
4    $w_j = Av_j$ ; // SpMV in inner iteration
5   for  $i = 1, 2, \dots, j$  do
6      $H_{ij} = (w_j, v_i)$ ;  $w_j = w_j - H_{ij}v_i$ ;
7   end
8    $H_{j+1,j} = \|w_j\|$ ;
9   if  $H_{j+1,j} == 0$  or Stopping criteria satisfied then
10     $m = j$ ; break;
11  end
12   $v_{j+1} = w_j/H_{j+1,j}$ ;
13 end
14  $V_m = [v_1, \dots, v_m]$ ;
15 find  $y^m$  that minimizes  $\|\beta e_1 - H_m y\|$ ;
16  $x_m = x_0 + V_m y^m$ ;
17  $r_m = b - Ax_m$ ; // SpMV in outer iteration
18 if  $\|r_m\| \leq \epsilon \|b\|$  then
19    $\hat{x} = x_m$ ; break;
20 else // Restart
21    $r_0 = r_m$ ;  $x_0 = x_m$ ; Goto 2;
22 end
```

■ Memory bounded

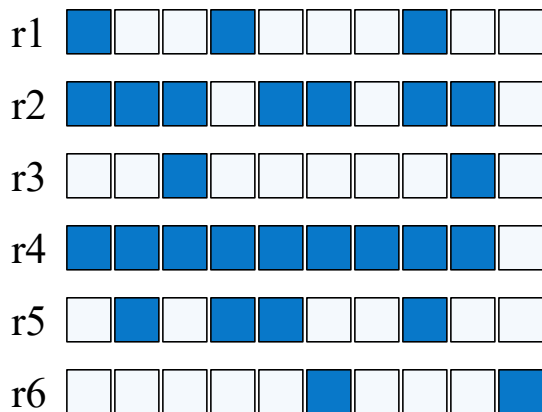
- Low arithmetic intensity

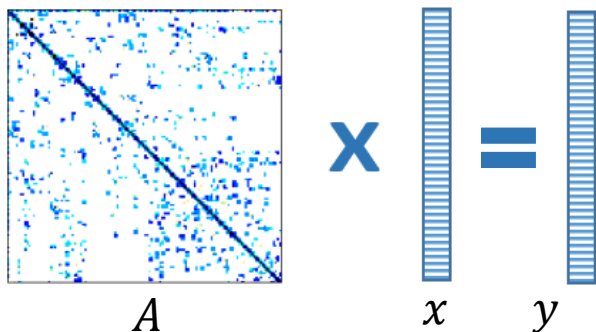
■ Indirect access

- Irregular vector reads

■ Uneven work

- Different row lengths



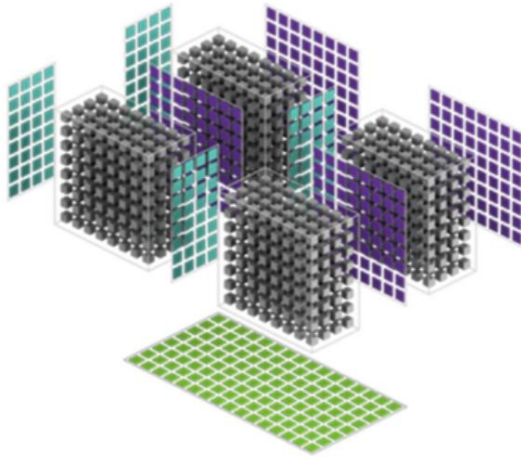


$A \times x = y$

Sparse matrix-vector multiplication (SpMV)

**Irregular memory access + uneven work
→ low GPU utilization**

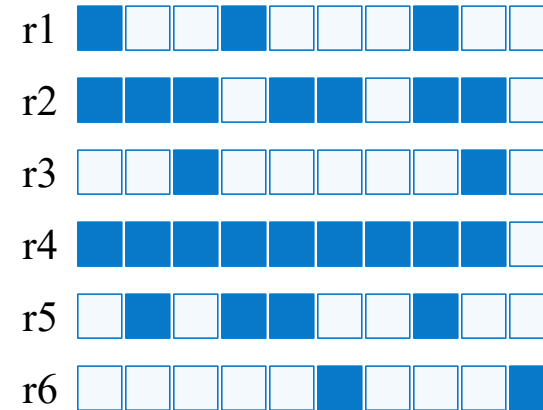
Tensor Core fragment



Fixed size & Dense

≠
Mismatch

SpMV structure



Irregular & Sparse

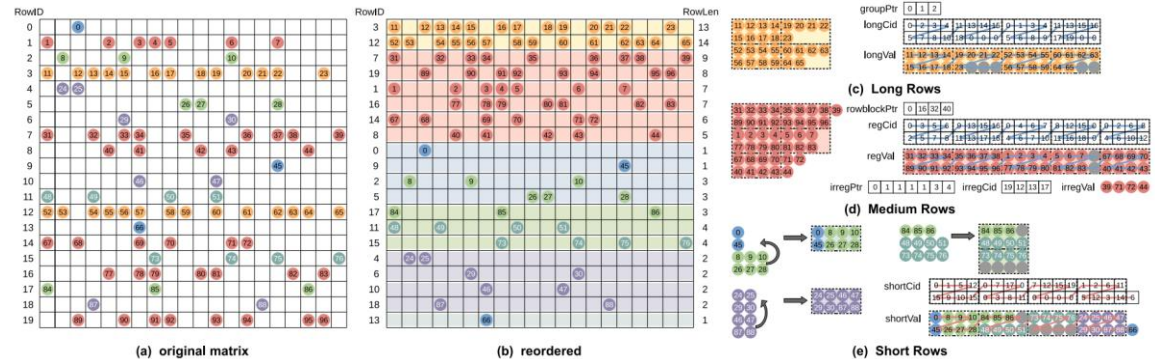
Direct mapping introduces padding + redundant accumulation

Related Work

■ DASP^[1]

Reorder rows and group them according to NNZ per row

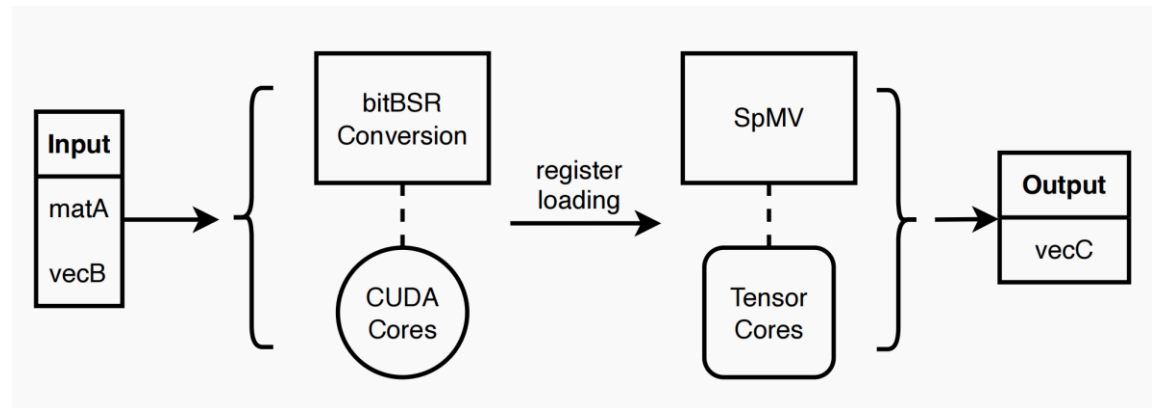
Cons: High padding ratio



■ Spaden^[2]

Bitmap conversion and bitBSR storage-based SpMV

Cons: Load imbalance & high useless computing



[1] Lu Y, Liu W. DASP: Specific dense matrix multiply-accumulate units accelerated general sparse matrix-vector multiplication[C], SC 2023.

[2] Chen Y A, Yu J X. Bitmap-based sparse matrix-vector multiplication with tensor cores[C], ICPP 2024.

- Background and Motivation
- Design and Implementation
- Experimental Evaluation
- Conclusion

Data Layout

Sort rows
Extract dense tiles
Stored in new storage

+

TC Execution

Register-aware MMA
Double buffering

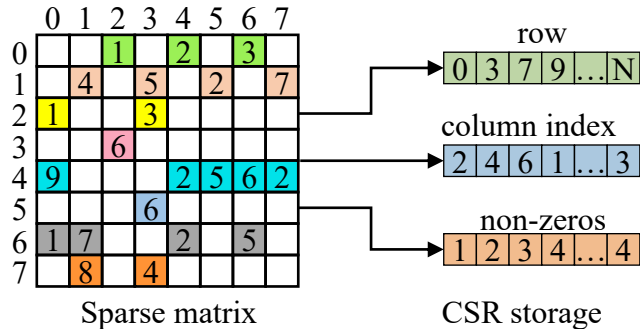
+

Load Balance

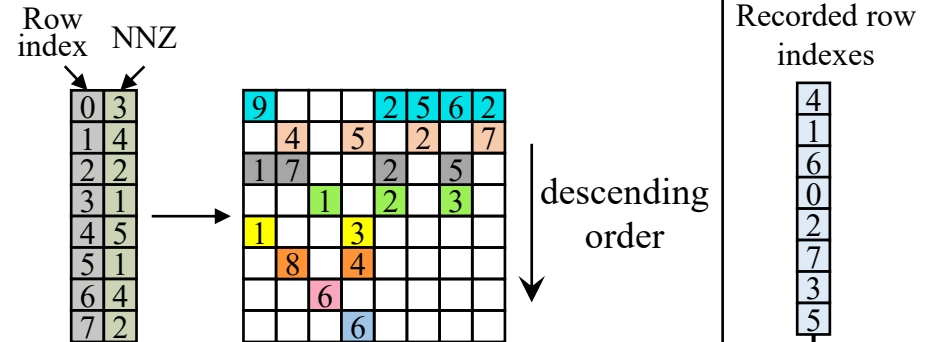
Long-row specified kernel
Merge small row groups

Sparse Format Design

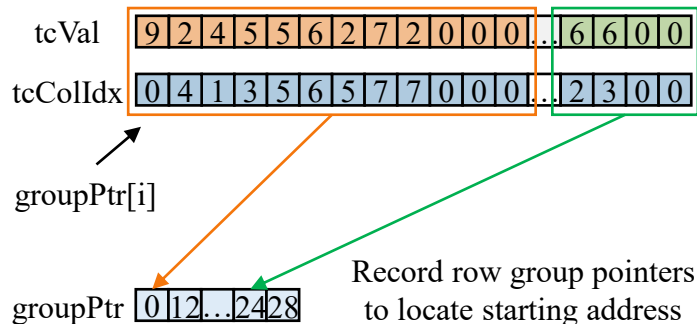
INPUT: Sparse matrix stored in CSR format



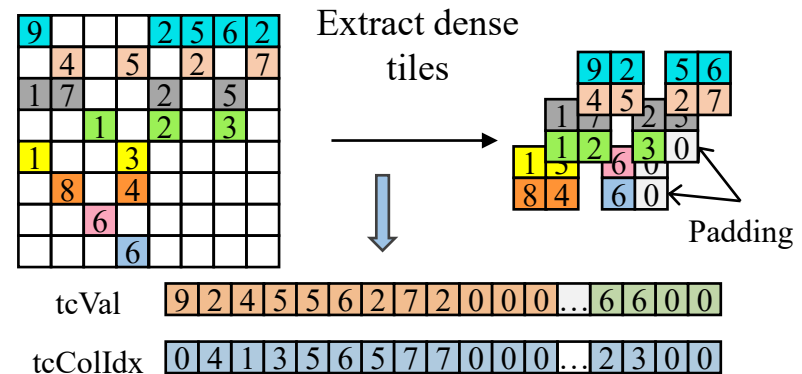
STEP 1: Reorder rows according to NNZ per row



STEP 3: Generate row group indexes

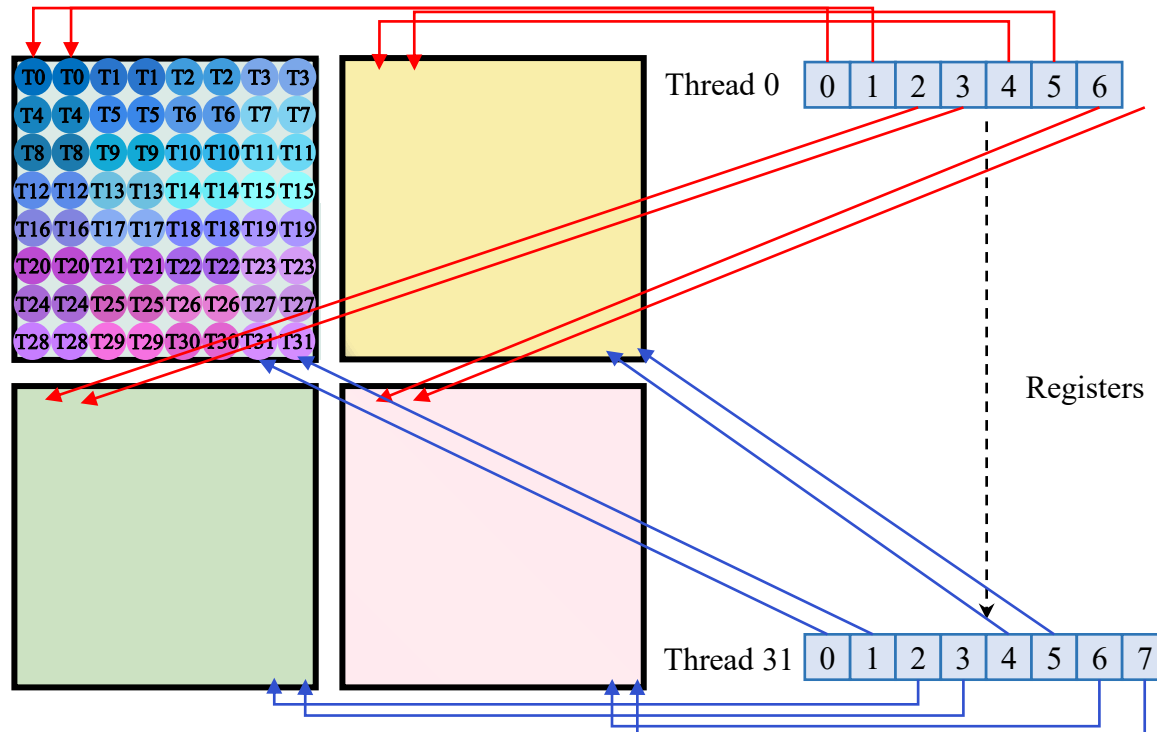


STEP 2: Extract dense tiles



Register-Aware Tensor Core Execution

- Load matrix tiles into the register layout of the left-hand MMA fragment
- Based on *tcColIdx*, gather and reorganize the input vector into the right-hand MMA fragment
- Execute MMA operations



- Extract only the results that contribute to the final SpMV output according to the following rules
- Double register buffers are used to support MMA pipeline

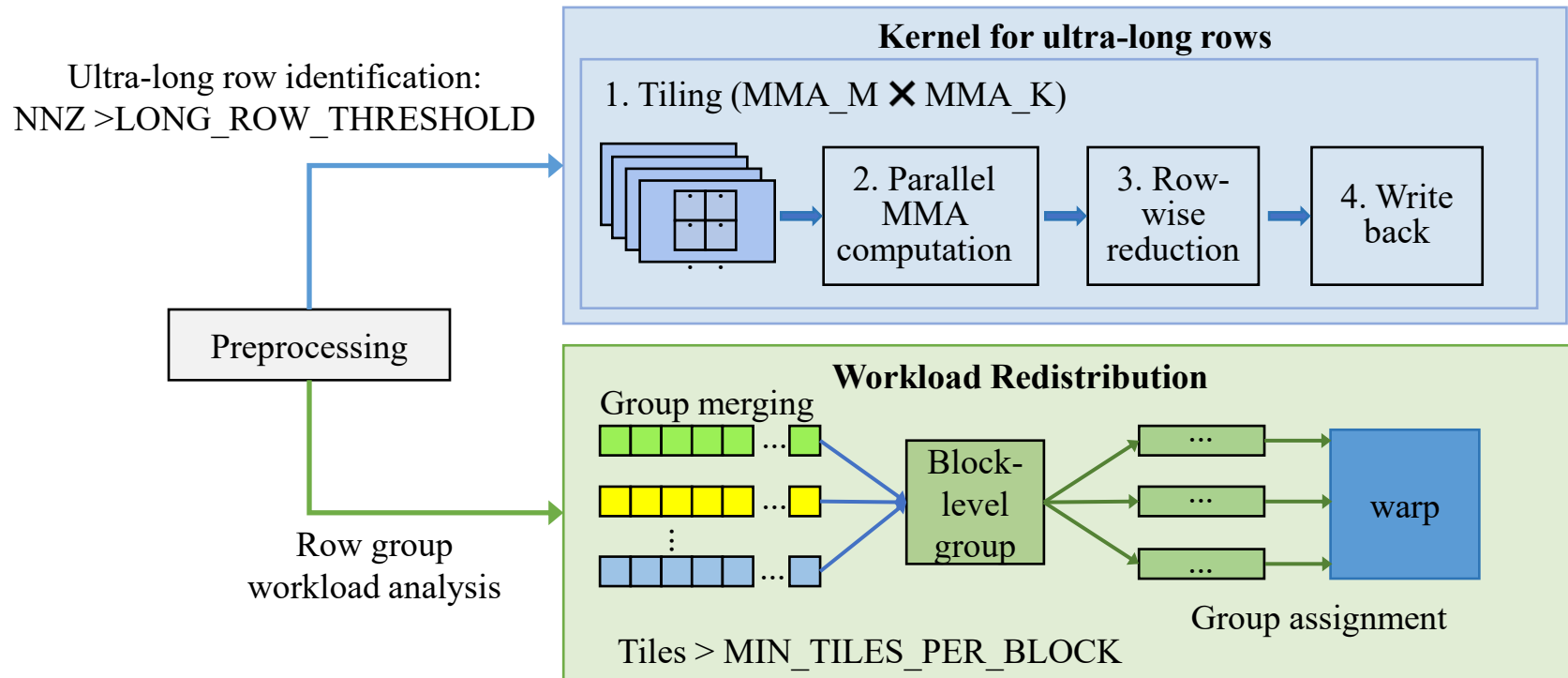
$$i = laneId \gg 2, \quad j = (laneId \bmod 4) \ll 1$$

$$Y_{out}[i] \leftarrow Y_{out}[i] + \begin{cases} C_{frag}[0], & \text{if } i = j, \\ C_{frag}[1], & \text{if } i = j + 1, \\ 0, & \text{otherwise,} \end{cases}$$

$$Y_{out}[MMA_M/2 + i] \leftarrow Y_{out}[MMA_M/2 + i] + \begin{cases} C_{frag}[6], & \text{if } i = j, \\ C_{frag}[7], & \text{if } i = j + 1, \\ 0, & \text{otherwise.} \end{cases}$$

Load-Balancing for Irregular Matrices

- Extract ultra-long rows to call the specific kernel
- Merge consecutive lightweight row groups until the cumulative tile count reaches the predefined threshold



- Background and Motivation
- Design and Implementation
- **Experimental Evaluation**
- Conclusion

■ Hardware and software

- GPU 1: GeForce RTX 3090 with 24 GB device memory
- GPU 2: A100 with 40 GB device memory
- Ubuntu 22.04
- CUDA 12.0
- GCC 11.4.0

■ Sparse matrices

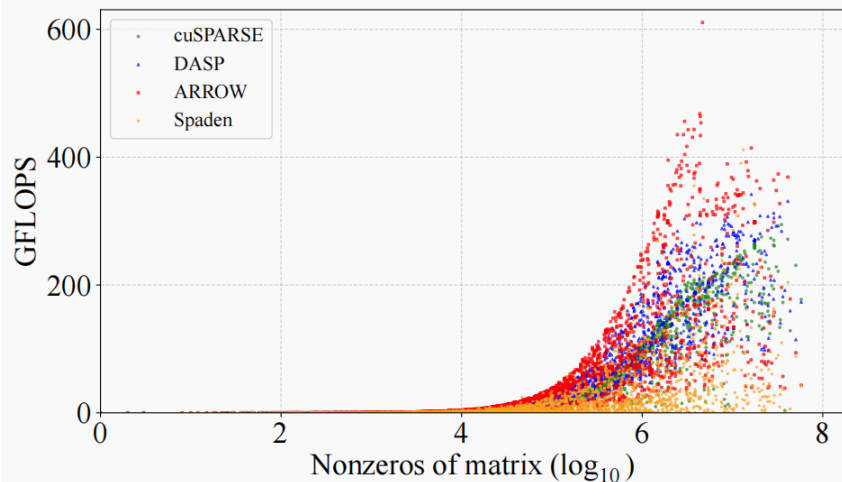
- 2800 matrices from SuiteSparse Matrix Collection
- 14 representative matrices for case studies

■ Compared methods

- cuSPARSE
- DASP
- Spaden

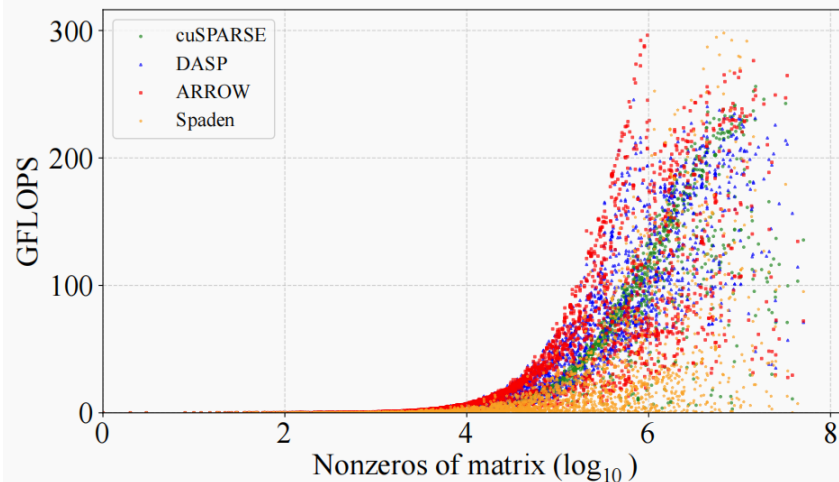


GFLOPS Comparison



(a) GFLOPS on A100

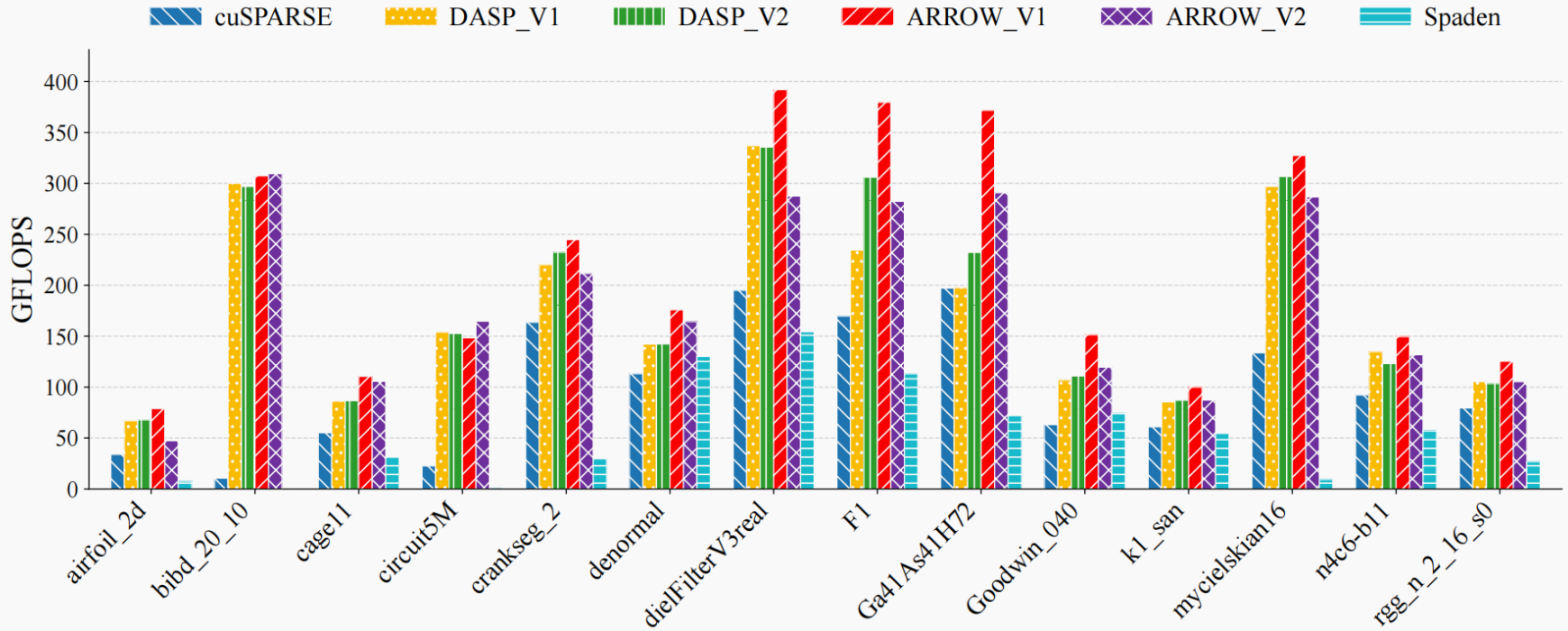
A100
1.78× vs. cuSPARSE
1.10× vs. DASP
3.22× vs. Spaden



(b) GFLOPS on RTX 3090

RTX 3090
1.57× vs. cuSPARSE
1.06× vs. DASP
3.58× vs. Spaden

Results on Representative matrices

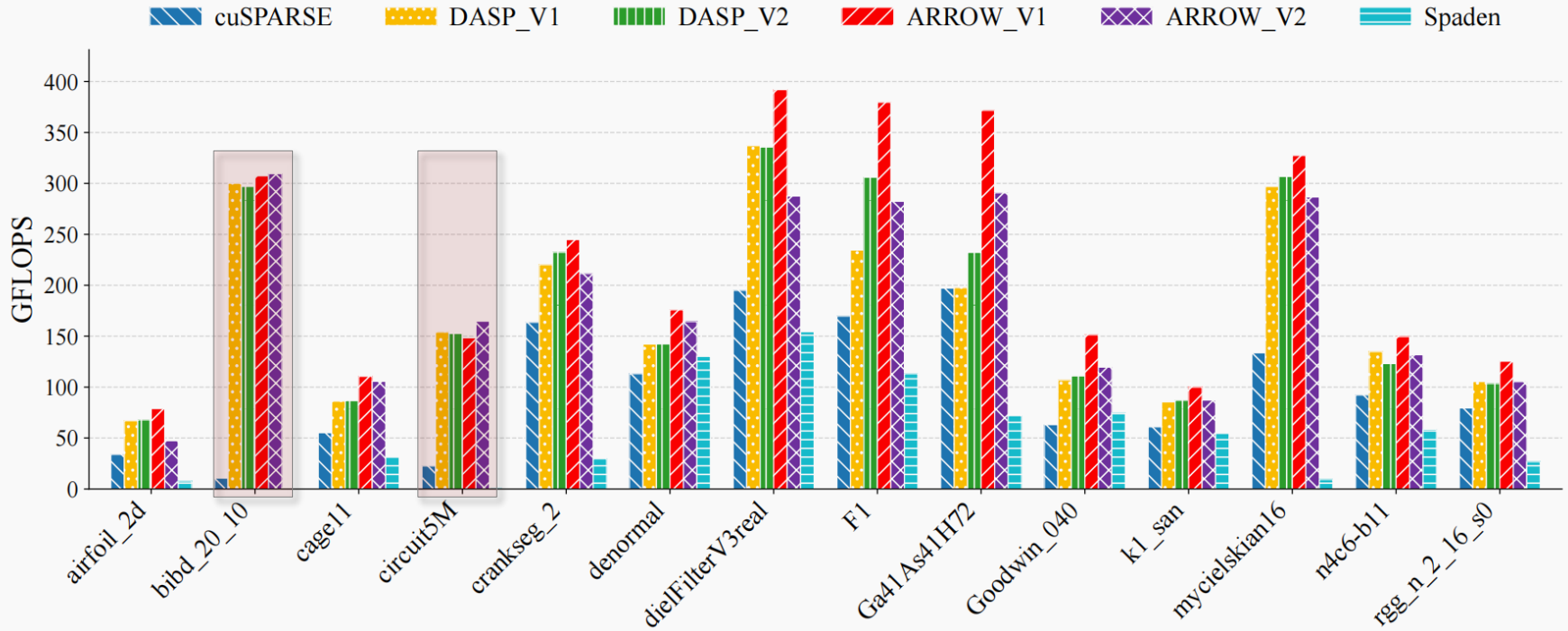


GFLOPS comparison of different methods for representative matrices on A100

ARROW_V1: without load-balanced optimization

ARROW_V2: with load-balanced optimization

Results on Representative matrices

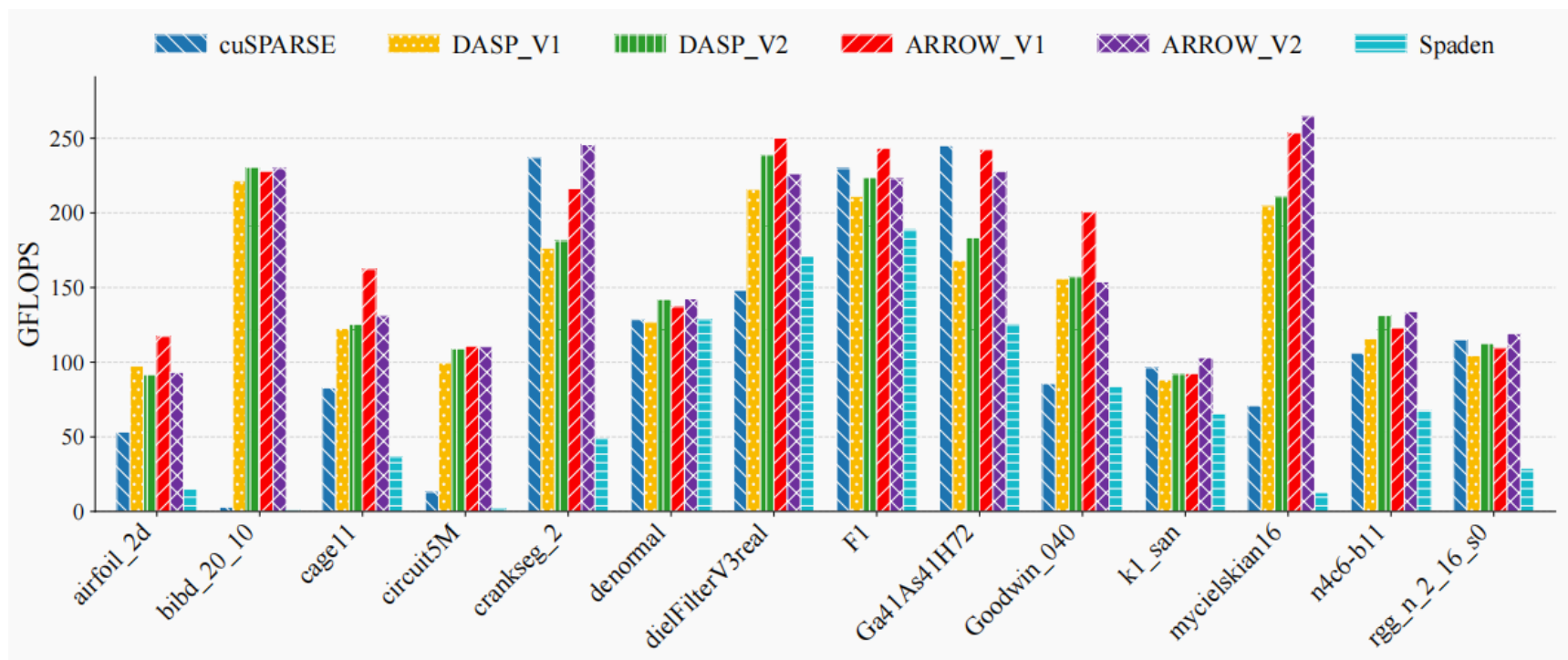


GFLOPS comparison of different methods for representative matrices on A100

ARROW_V1: without load-balanced optimization

ARROW_V2: with load-balanced optimization

Results on Representative matrices

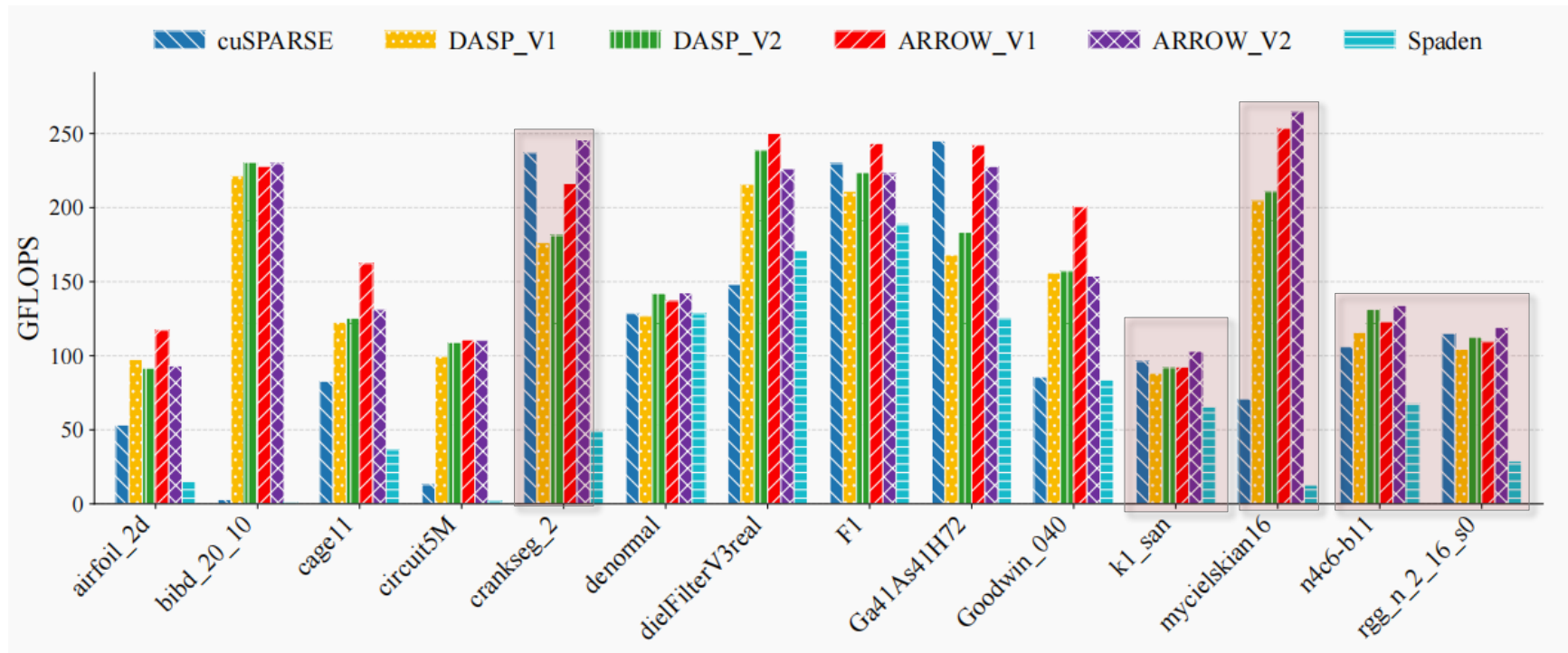


GFLOPS comparison of different methods for representative matrices on RTX 3090

ARROW_V1: without load-balanced optimization

ARROW_V2: with load-balanced optimization

Results on Representative matrices



GFLOPS comparison of different methods for representative matrices on RTX 3090

ARROW_V1: without load-balanced optimization

ARROW_V2: with load-balanced optimization

- Background and Motivation
- Design and Implementation
- Experimental Evaluation
- Conclusion

■ Problem

- Mismatch between SpMV and Tensor Cores

■ Arrow

- New data layout design
- Efficient MMA powered by register mapping and double buffering
- Load balancing by processing long rows and short rows differently

■ Results

- Superior to three SOTA methods on two typical GPU platforms



*the 17th International Symposium on Advanced
Parallel Processing Technology*



Q&A



Beijing Normal University