EURO-PAR
CONFERENCE 2024

SKLP 处理器芯片全国重点实验室
STATE KEY LAB OF PROCESSORS, ICT, CAS

中国科学院计算技术研究所

中国科学院大学
University of Chinese Academy of Sciences

中国石油大学（北京）
CHINA UNIVERSITY OF PETROLEUM

Accelerating Large-scale Sparse LU Factorization For RF Circuit Simulation

Guofeng Feng[1,2], Hongyu Wang[1,2], Zhuoqiang Guo[1,2], Mingzhen Li[1*],
Tong Zhao[1], Zhou Jin[3], Weile Jia[1], Guangming Tan[1], Ninghui Sun[1]

1. SKLP, Institute of Computing Technology, CAS, Beijing 100190, China
2. University of Chinese Academy of Sciences, Beijing, Beijing 100190, China
3. SSSLab, China University of Petroleum-Beijing, Beijing 102249, China

30 August 2024　　*Madrid Spain*

OUTLINE

- Background
- Motivation
- Design
- Evaluation
- Conclusion

Background: RF Circuit Simulation

- **Radio Frequency Circuit:**
  - At the frequency around 20 kHz to 300 GHz
  - A wide range of applications:
    - Radio broadcasting
    - Mobile phones
    - Satellite

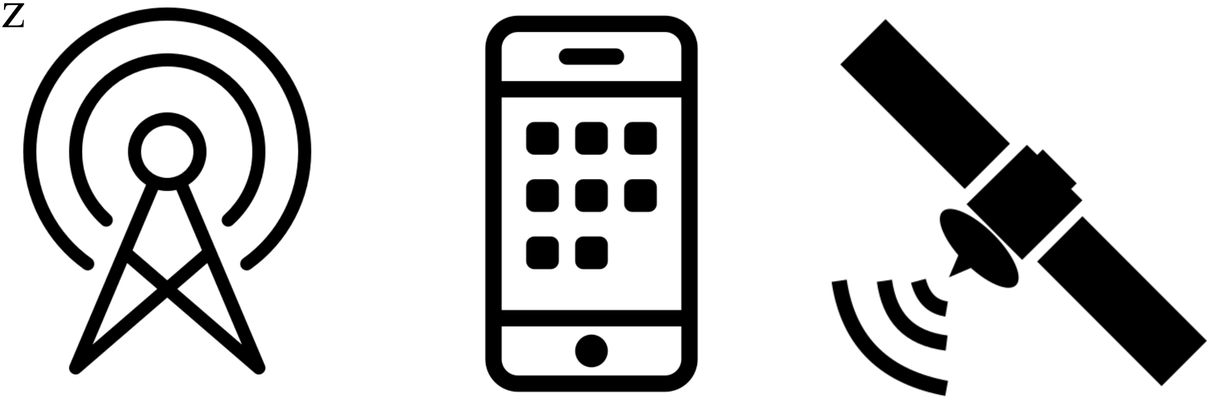


Image source: Wikimedia Commons

- **Circuit Simulation**
  - A vital step in circuit design
    - Simulation before the expensive fabrication of circuits
  - To matrices and factorization
    - Solving a system of linear equations

Background: LU Factorization

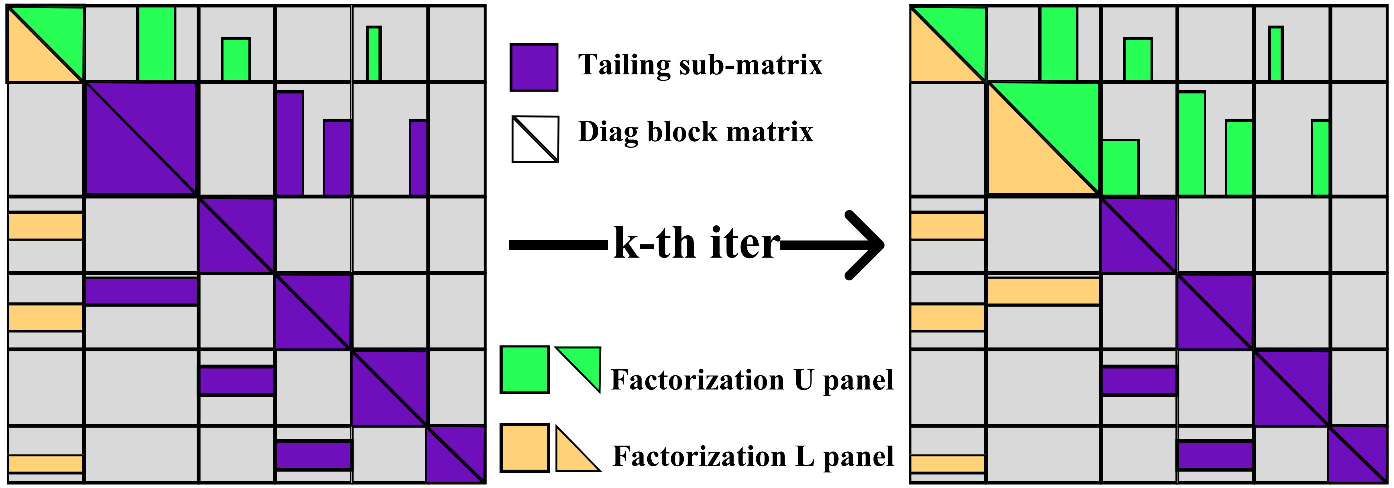
- $\boldsymbol{A = LU}$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

LU factorization decomposes a matrix ***A*** as the product of a lower triangular matrix (***L***) and an upper triangular matrix (***U***), which is key component of direct method solver (Gaussian Elimination)

Background: Sparse LU Factorization

- **When A is sparse**
  - Require to improve computation and memory efficiency (3 phase: preprocessing, symbolic, numeric)
- **Supernodal**
  - Grouping columns with similar non-zero structures to form a supernode, leveraging high-performance dense linear algebra subprograms (BLAS)



Background: Sparse Solver

- **Related work**
  - Large-scale simulation requires better leveraging the power of current distributed machines (commonly equipped with heterogeneous accelerators)

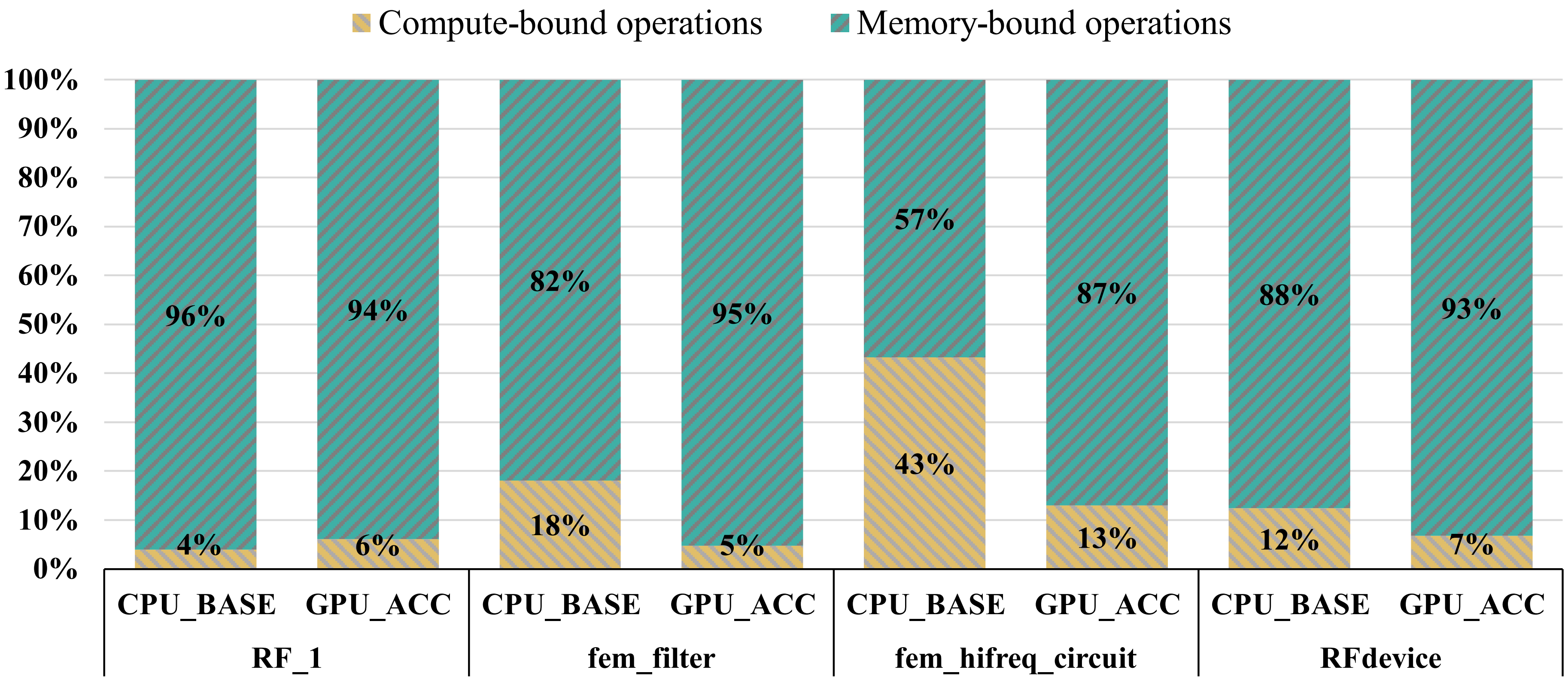
Solver	Distributed	Heterogeneous
PARDISO	×	×
KLU	×	×
GLU	×	√
NICSLU	×	√
MUMPS	√	×
SuperLU_DIST	√	√

OUTLINE

- **Background**
- **Motivation**
- **Design**
- **Evaluation**
- **Conclusion**

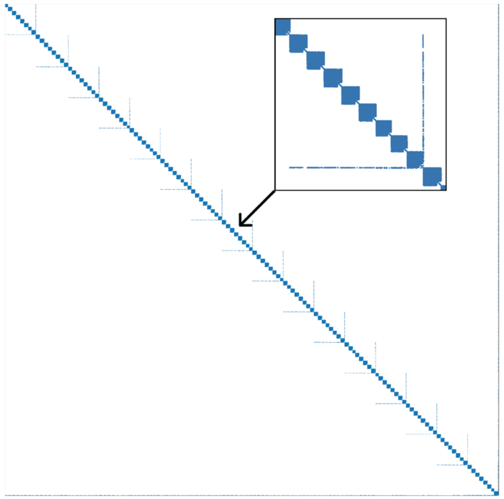
Motivation: Performance Bottleneck

- **Hot Spot: Memory-bound Operations**
  - The compute-bound operations (GEMM) are widely optimized and reaching the performance ceiling

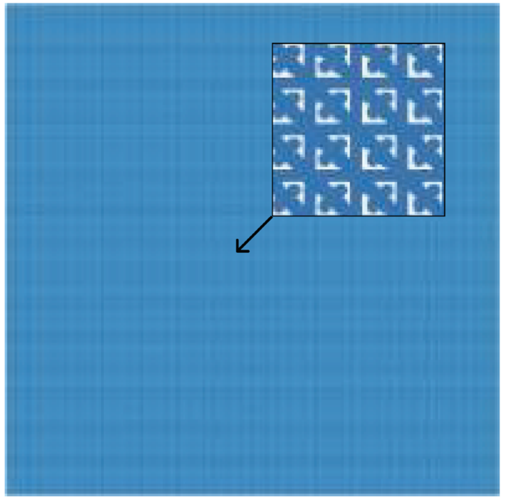


Motivation: Application Features

- **Application Features**
  - RF circuit matrices are composed of structured dense blocks even after the preprocessing
  - The pattern of structured dense blocks can be exploited to alleviate the memory bound



(a) RF_1 (row (col) = 5,809,680, nnz = 194,348,610)



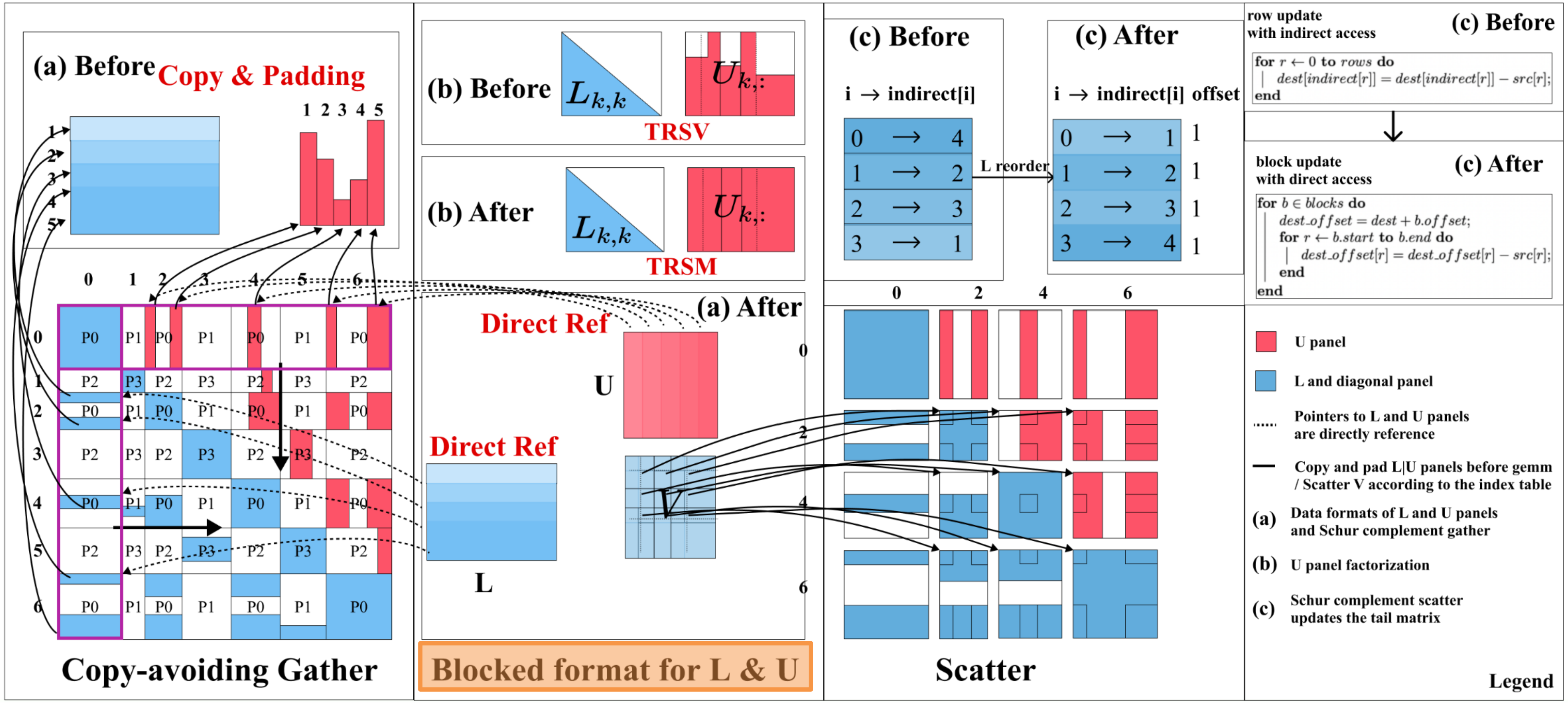
(b) RF_2 (row (col) = 418,718, nnz = 921,708,586)

OUTLINE

- Background
- Motivation
- Design
- Evaluation
- Conclusion

Design: Blocked Format

- Reorganize non-zeros:



(c) Before

i	→	indirect[i]
0	→	4
1	→	2
2	→	3
3	→	1

(c) After

i	→	indirect[i]	offset
0	→	1	1
1	→	2	1
2	→	3	1
3	→	4	1

row update with indirect access — (c) Before

```
for $r \leftarrow 0$ to $rows$ do
    $dest[indirect[r]] = dest[indirect[r]] - src[r];$
end
```

↓

block update with direct access — (c) After

```
for $b \in blocks$ do
    $dest\_offset = dest + b.offset;$
    for $r \leftarrow b.start$ to $b.end$ do
        $dest\_offset[r] = dest\_offset[r] - src[r];$
    end
end
```

🟥 U panel

🟦 L and diagonal panel

...... Pointers to L and U panels are directly reference

— Copy and pad L|U panels before gemm / Scatter V according to the index table

(a) Data formats of L and U panels and Schur complement gather

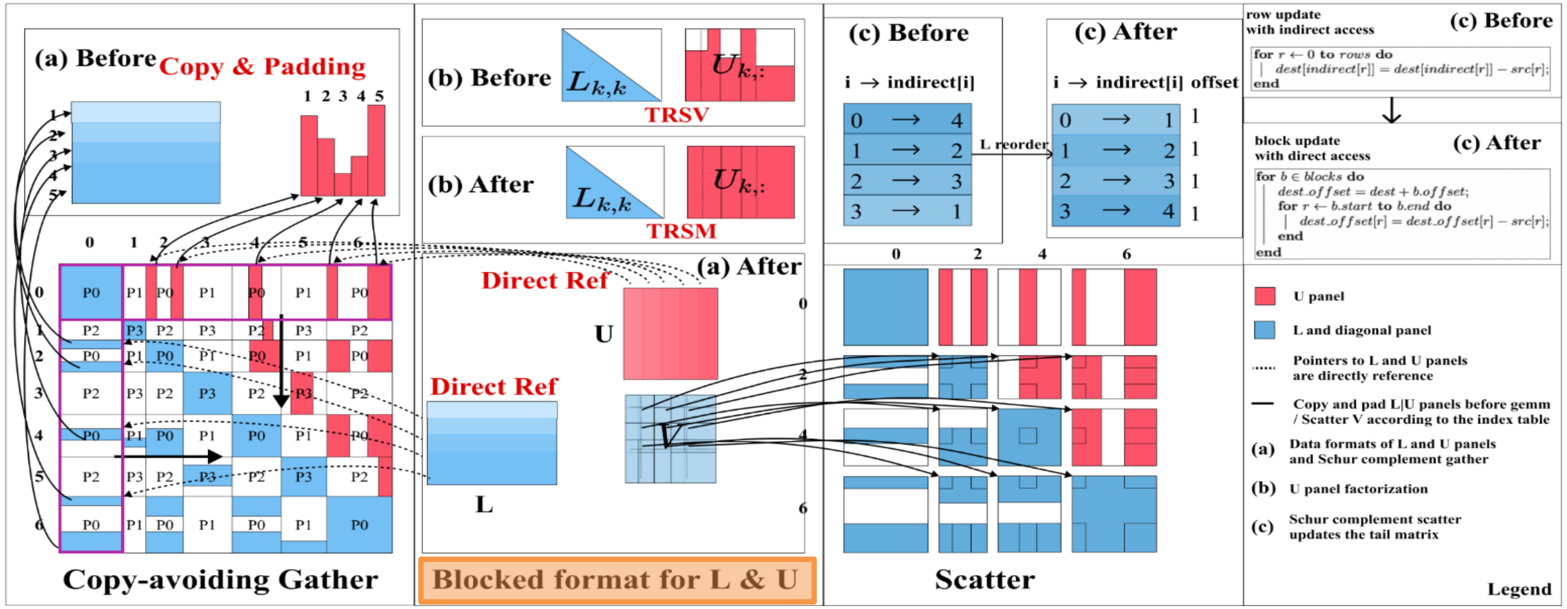
(b) U panel factorization

(c) Schur complement scatter updates the tail matrix

Legend

Design: Blocked Format

- **Reorganize nonzeros:**
  - RF matrices exhibit a block-sparse structure, it is feasible to eliminate redundant zero values along with the corresponding redundant computations introduced by zero padding.
  - Subsequent operations can benefit from the continuous storage and improved locality



(c) Before

i	→	indirect[i]
0	→	4
1	→	2
2	→	3
3	→	1

(c) After

i	→	indirect[i]	offset
0	→	1	1
1	→	2	1
2	→	3	1
3	→	4	1

row update with indirect access — (c) Before

```
for $r \leftarrow 0$ to $rows$ do
    $dest[indirect[r]] = dest[indirect[r]] - src[r]$;
end
```

↓

block update with direct access — (c) After

```
for $b \in blocks$ do
    $dest\_offset = dest + b.offset$;
    for $r \leftarrow b.start$ to $b.end$ do
        $dest\_offset[r] = dest\_offset[r] - src[r]$;
    end
end
```

🟥 U panel

🟦 L and diagonal panel

...... Pointers to L and U panels are directly reference

— Copy and pad L|U panels before gemm / Scatter V according to the index table

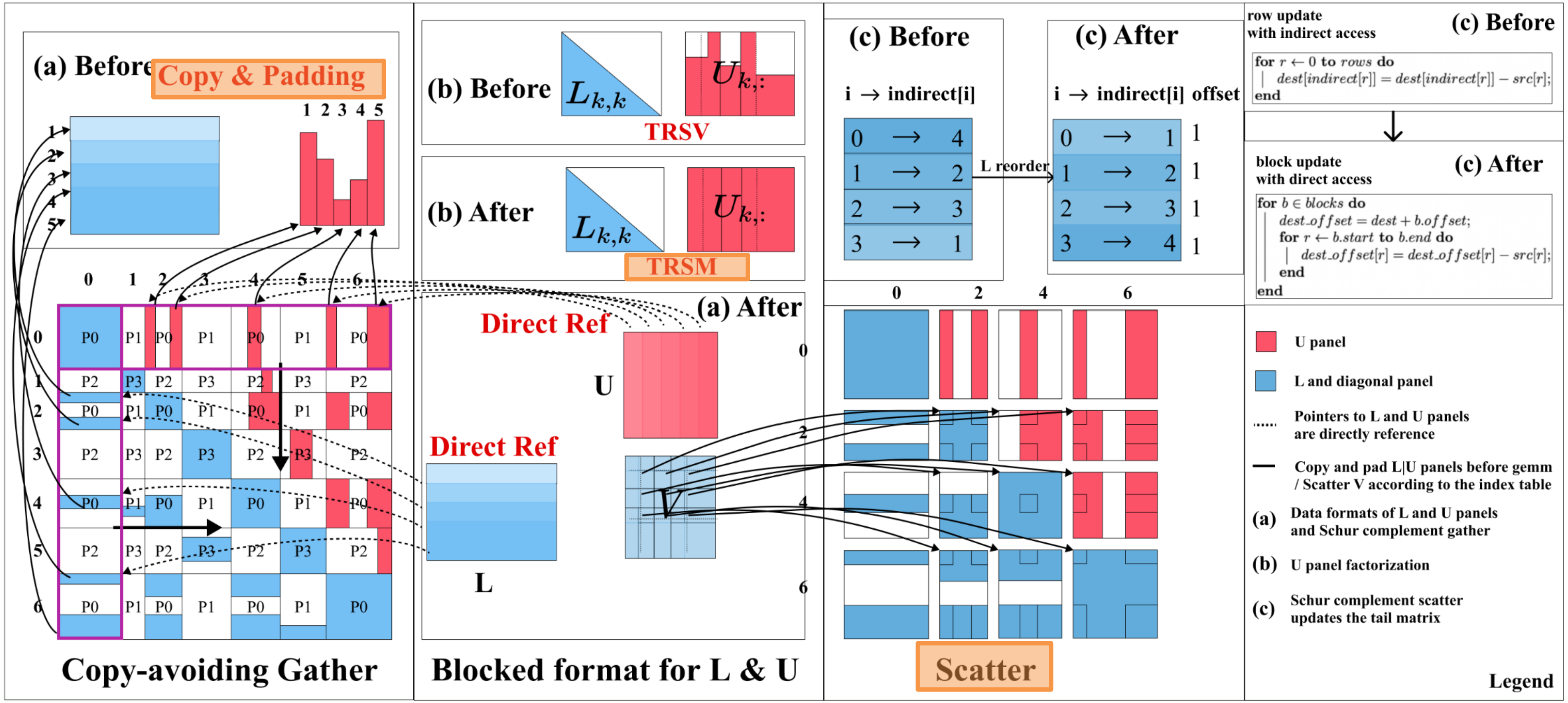
(a) Data formats of L and U panels and Schur complement gather

(b) U panel factorization

(c) Schur complement scatter updates the tail matrix

Design: Optimized Operations

- Redesign critical operations based on our blocked format



(c) Before

i	→	indirect[i]
0	→	4
1	→	2
2	→	3
3	→	1

(c) After

i	→	indirect[i]	offset
0	→	1	1
1	→	2	1
2	→	3	1
3	→	4	1

row update
with indirect access

(c) Before

```
for $r \leftarrow 0$ to $rows$ do
    $dest[indirect[r]] = dest[indirect[r]] - src[r];$
end
```

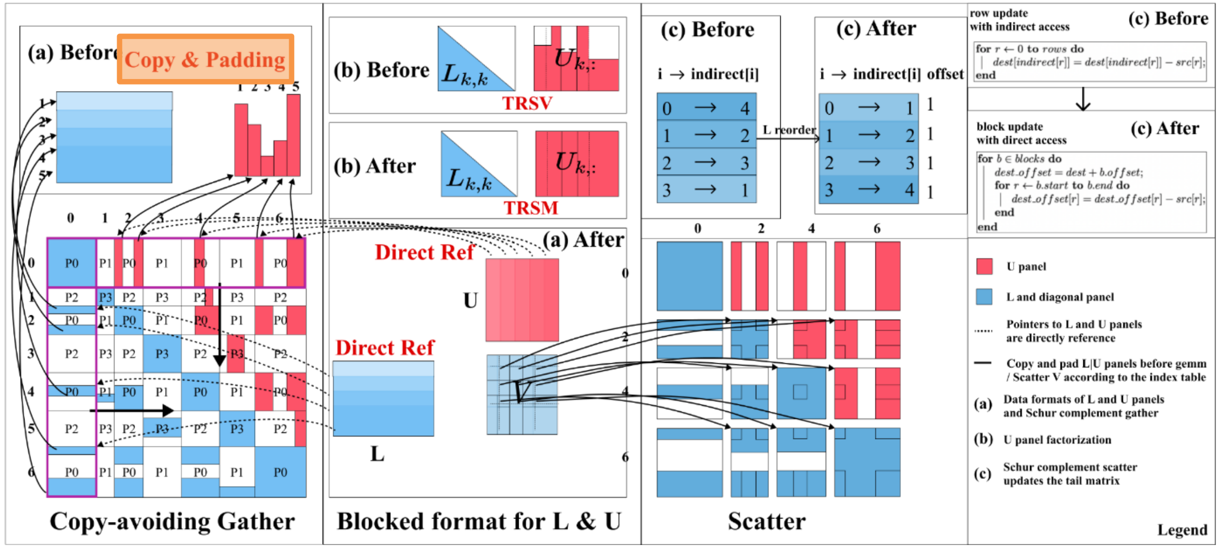
block update
with direct access

(c) After

```
for $b \in blocks$ do
    $dest\_offset = dest + b.offset;$
    for $r \leftarrow b.start$ to $b.end$ do
        $dest\_offset[r] = dest\_offset[r] - src[r];$
    end
end
```

Design: Optimized Operations

- **Redesign critical operations based on our blocked format**
  - Gather:
    - we record the beginning and the offset of the L and U matrices in our block format to access the data by the pointer dereferences instead of copy and padding.
  - Scatter
  - Panel Factorization



Copy-avoiding Gather

Blocked format for L & U

Scatter

Legend

row update with indirect access (c) Before

```
for $r \leftarrow 0$ to $rows$ do
    $dest[indirect[r]] = dest[indirect[r]] - src[r];$
end
```

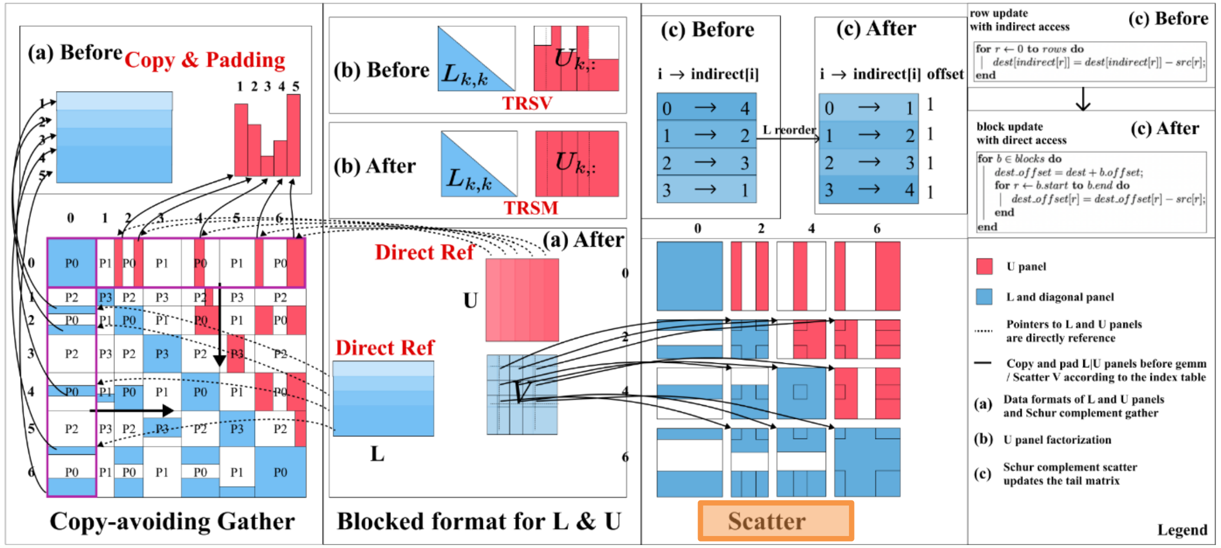
block update with direct access (c) After

```
for $b \in blocks$ do
    $dest\_offset = dest + b.offset;$
    for $r \leftarrow b.start$ to $b.end$ do
        $dest\_offset[r] = dest\_offset[r] - src[r];$
    end
end
```

(a) Before
Copy & Padding
1 2 3 4 5
0 1 2 3 4 5 6
P0 P1 P2 P3
(b) Before
$L_{k,k}$
$U_{k,:}$
TRSV
(b) After
TRSM
(a) After
Direct Ref
U
L
V
(c) Before
i → indirect[i]
(c) After
i → indirect[i] offset
L reorder
0 → 4
1 → 2
2 → 3
3 → 1
0 → 1 1
1 → 2 1
2 → 3 1
3 → 4 1
🟥 U panel
🟦 L and diagonal panel
...... Pointers to L and U panels are directly reference
— Copy and pad L|U panels before gemm / Scatter V according to the index table
(a) Data formats of L and U panels and Schur complement gather
(b) U panel factorization
(c) Schur complement scatter updates the tail matrix

Design: Optimized Operations

- **Redesign critical operations based on our blocked format**
  - Scatter:
    - We construct a continuous index table and it can be used in the traverse of V block columns.
    - We keep the matched location utilizing the ordering.
    - We record the dense block information (address, number, offset, etc.).
  - Panel Factorization



row update
with indirect access

(c) Before

```
for $r \leftarrow 0$ to $rows$ do
    $dest[indirect[r]] = dest[indirect[r]] - src[r]$;
end
```

block update
with direct access

(c) After

```
for $b \in blocks$ do
    $dest\_offset = dest + b.offset$;
    for $r \leftarrow b.start$ to $b.end$ do
        $dest\_offset[r] = dest\_offset[r] - src[r]$;
    end
end
```

🟥 U panel

🟦 L and diagonal panel

...... Pointers to L and U panels are directly reference

— Copy and pad L|U panels before gemm / Scatter V according to the index table

(a) Data formats of L and U panels and Schur complement gather

(b) U panel factorization

(c) Schur complement scatter updates the tail matrix

Legend

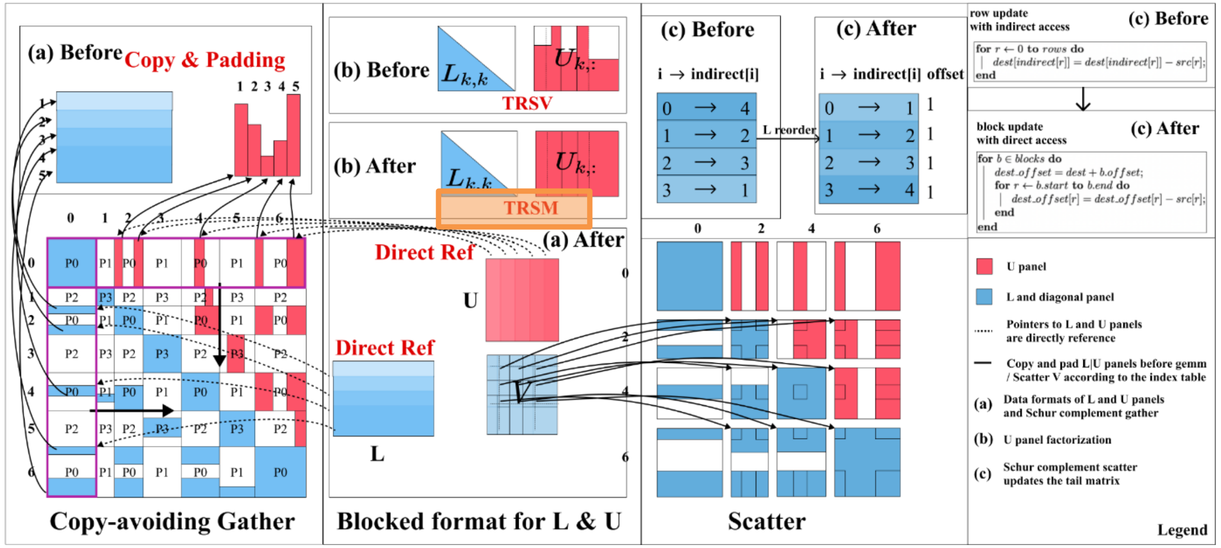
Design: Optimized Operations

- **Redesign critical operations based on our blocked format**
  - Panel Factorization:
    - Replacing a series of TRSV kernels with single TRSM kernel, as U factors stored in blocks
    - Data reuse of the input L blocks, more computation intensity, and more parallelism in data level.

Algorithm 2 `TRSM` kernel (when $m = 1$, `TRSV` kernel).

Input: l: dense lower triangular matrix of column major
Input: b: multiple right-hand vectors (when $m = 1$, a dense vector)

```
for $k = 0$ to $n - 1$ do
    for $i = 0$ to $m - 1$ do
        $b[k \times ldb + i] \leftarrow b[k \times ldb + i] / l[k \times lda + k]$
    for $j = k + 1$ to $n - 1$ do
        for $i = 0$ to $m - 1$ do
            $b[j \times ldb + i] \leftarrow b[j \times ldb + i] - l[k \times lda + j] \times b[k \times ldb + i]$
```



row update with indirect access

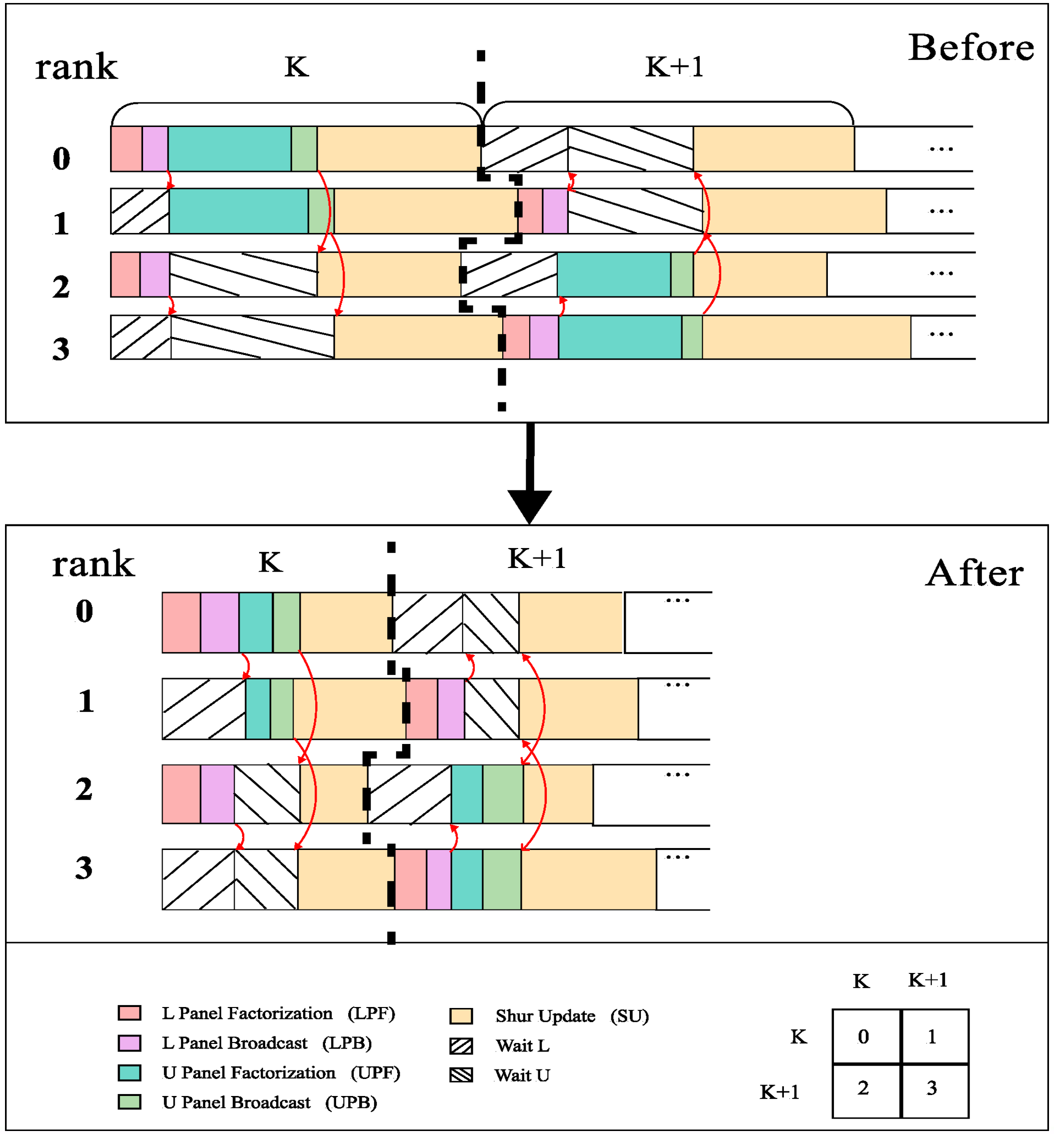
```
for $r \leftarrow 0$ to $rows$ do
    $dest[indirect[r]] = dest[indirect[r]] - src[r]$;
end
```

block update with direct access

```
for $b \in blocks$ do
    $dest\_offset = dest + b.offset$;
    for $r \leftarrow b.start$ to $b.end$ do
        $dest\_offset[r] = dest\_offset[r] - src[r]$;
    end
end
```

Design: Better Pipeline

- **Mitigate data dependency stall**
  - We optimize local computation
    - Data Structure
    - Operations (Gather, Scatter, TRSM)
  - Decrease inter-process waiting
    - P1 waits for the L factors of P0
    - P2 waits for the U factors of P0
    - P3 waits for the U factors of P2 and L factors of P1



OUTLINE

- Background
- Motivation
- Design
- Evaluation
- Conclusion

Evaluation: Setup

- **Platform:**

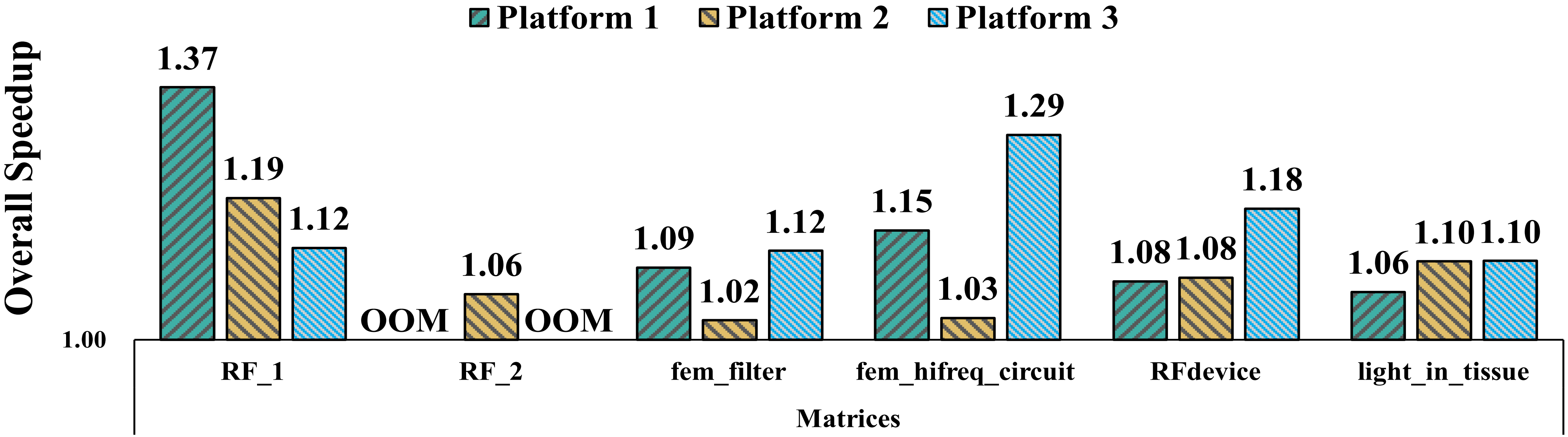
Platform	CPU	GPU (NVIDIA)
1	AMD E5-2640v4 (20 cores, 2.40GHz, 25MiB LLC)	V100×4 (16GB HBM2, Volta)
2	Intel Platinum 8375C (32 cores, 2.90GHz, 54MiB LLC)	A100×4 (80GB HBM2e, Ampere)
3	Intel Gold 6326 (16 cores, 2.90GHz, 24MiB LLC)	3090×4 (24GB GDDR6X, Ampere)

- **Dataset:**

Matrix	row	nnz
RF_1	5810K	194M
RF_2	419K	922M
fem_filter	74K	2M
fem_hifreq_circuit	491K	20M
Rfdevice	74K	0.4M
light_in_tissue	30K	0.4M

Evaluation: Performance

- **Overall Performance**



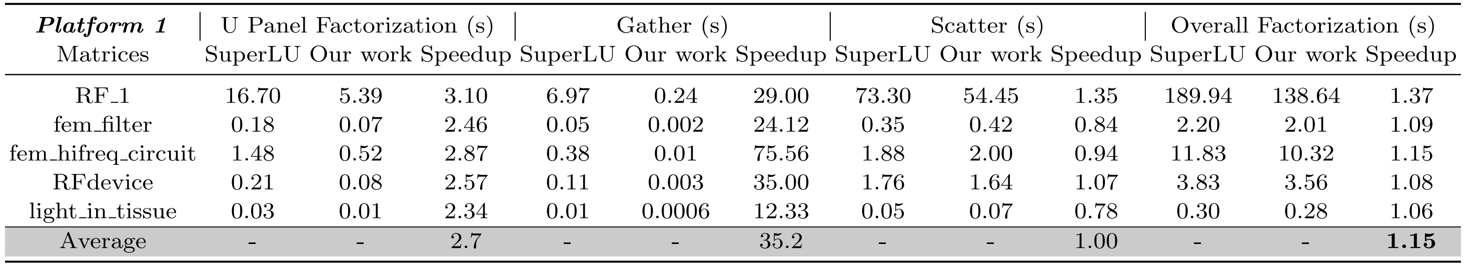
Achieve up to 1.37× (avg 1.15×), 1.19× (avg 1.08×), and 1.29× (avg 1.16×) overall speedup on 3 platforms

Evaluation: Breakdown

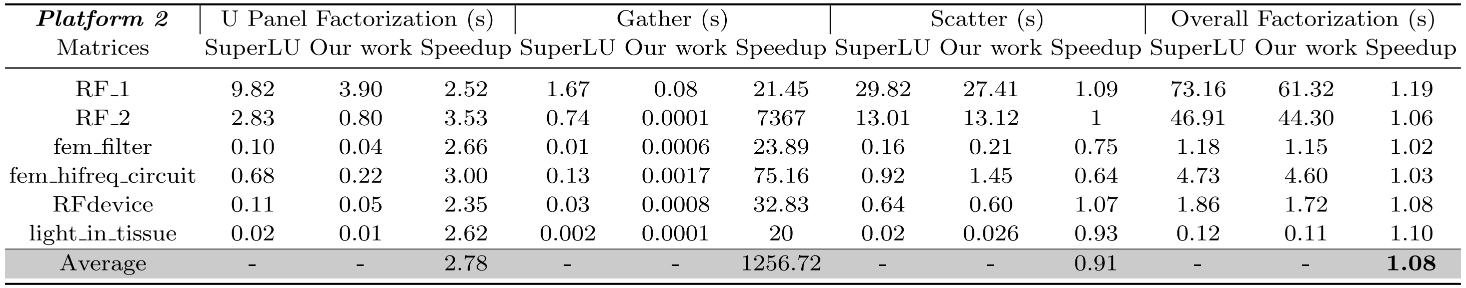
- **Detailed performance in each phase of numeric factorization**

Compared with SuperLU_DIST, We achieve <span>orders of magnitude</span> improvement in these critical data movement component.

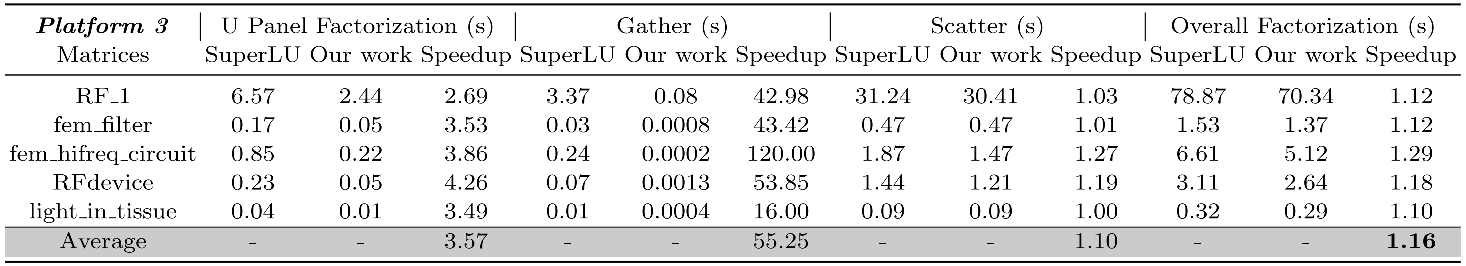
Overall runtime reduction can exceed the sum of the reductions of each optimized memory-bound step.



Platform 1	U Panel Factorization (s)			Gather (s)			Scatter (s)			Overall Factorization (s)		
Matrices	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup
RF_1	16.70	5.39	3.10	6.97	0.24	29.00	73.30	54.45	1.35	189.94	138.64	1.37
fem_filter	0.18	0.07	2.46	0.05	0.002	24.12	0.35	0.42	0.84	2.20	2.01	1.09
fem_hifreq_circuit	1.48	0.52	2.87	0.38	0.01	75.56	1.88	2.00	0.94	11.83	10.32	1.15
RFdevice	0.21	0.08	2.57	0.11	0.003	35.00	1.76	1.64	1.07	3.83	3.56	1.08
light_in_tissue	0.03	0.01	2.34	0.01	0.0006	12.33	0.05	0.07	0.78	0.30	0.28	1.06
Average	-	-	2.7	-	-	35.2	-	-	1.00	-	-	**1.15**



Platform 2	U Panel Factorization (s)			Gather (s)			Scatter (s)			Overall Factorization (s)		
Matrices	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup
RF_1	9.82	3.90	2.52	1.67	0.08	21.45	29.82	27.41	1.09	73.16	61.32	1.19
RF_2	2.83	0.80	3.53	0.74	0.0001	7367	13.01	13.12	1	46.91	44.30	1.06
fem_filter	0.10	0.04	2.66	0.01	0.0006	23.89	0.16	0.21	0.75	1.18	1.15	1.02
fem_hifreq_circuit	0.68	0.22	3.00	0.13	0.0017	75.16	0.92	1.45	0.64	4.73	4.60	1.03
RFdevice	0.11	0.05	2.35	0.03	0.0008	32.83	0.64	0.60	1.07	1.86	1.72	1.08
light_in_tissue	0.02	0.01	2.62	0.002	0.0001	20	0.02	0.026	0.93	0.12	0.11	1.10
Average	-	-	2.78	-	-	1256.72	-	-	0.91	-	-	**1.08**



Platform 3	U Panel Factorization (s)			Gather (s)			Scatter (s)			Overall Factorization (s)		
Matrices	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup	SuperLU	Our work	Speedup
RF_1	6.57	2.44	2.69	3.37	0.08	42.98	31.24	30.41	1.03	78.87	70.34	1.12
fem_filter	0.17	0.05	3.53	0.03	0.0008	43.42	0.47	0.47	1.01	1.53	1.37	1.12
fem_hifreq_circuit	0.85	0.22	3.86	0.24	0.0002	120.00	1.87	1.47	1.27	6.61	5.12	1.29
RFdevice	0.23	0.05	4.26	0.07	0.0013	53.85	1.44	1.21	1.19	3.11	2.64	1.18
light_in_tissue	0.04	0.01	3.49	0.01	0.0004	16.00	0.09	0.09	1.00	0.32	0.29	1.10
Average	-	-	3.57	-	-	55.25	-	-	1.10	-	-	**1.16**

OUTLINE

- Background
- Motivation
- Design
- Evaluation
- Conclusion

Conclusion

- We identify the performance **bottleneck** and application **features** of RF circuit simulation
- We introduce a **blocked** sparse matrix format for efficiently handling large-scale simulation matrices
- We comprehensively optimize the **critical memory operations** based on our new format
- Our strategies improve both the **data movement and computation** performance based on SuperLU_DIST, and should work for others

Thanks for your attention!

Any Questions?

Welcome to contact us:

Email: gff.7@outlook.com, limingzhen@ict.ac.cn

Wechat: MaxF07