

Matrix Factorizations for Parallel Integer Transforms

Yiyuan She^{1,2,3}, Pengwei Hao^{1,2}, Yakup Paker²



¹Center for Information Science, Peking University



²Queen Mary, University of London



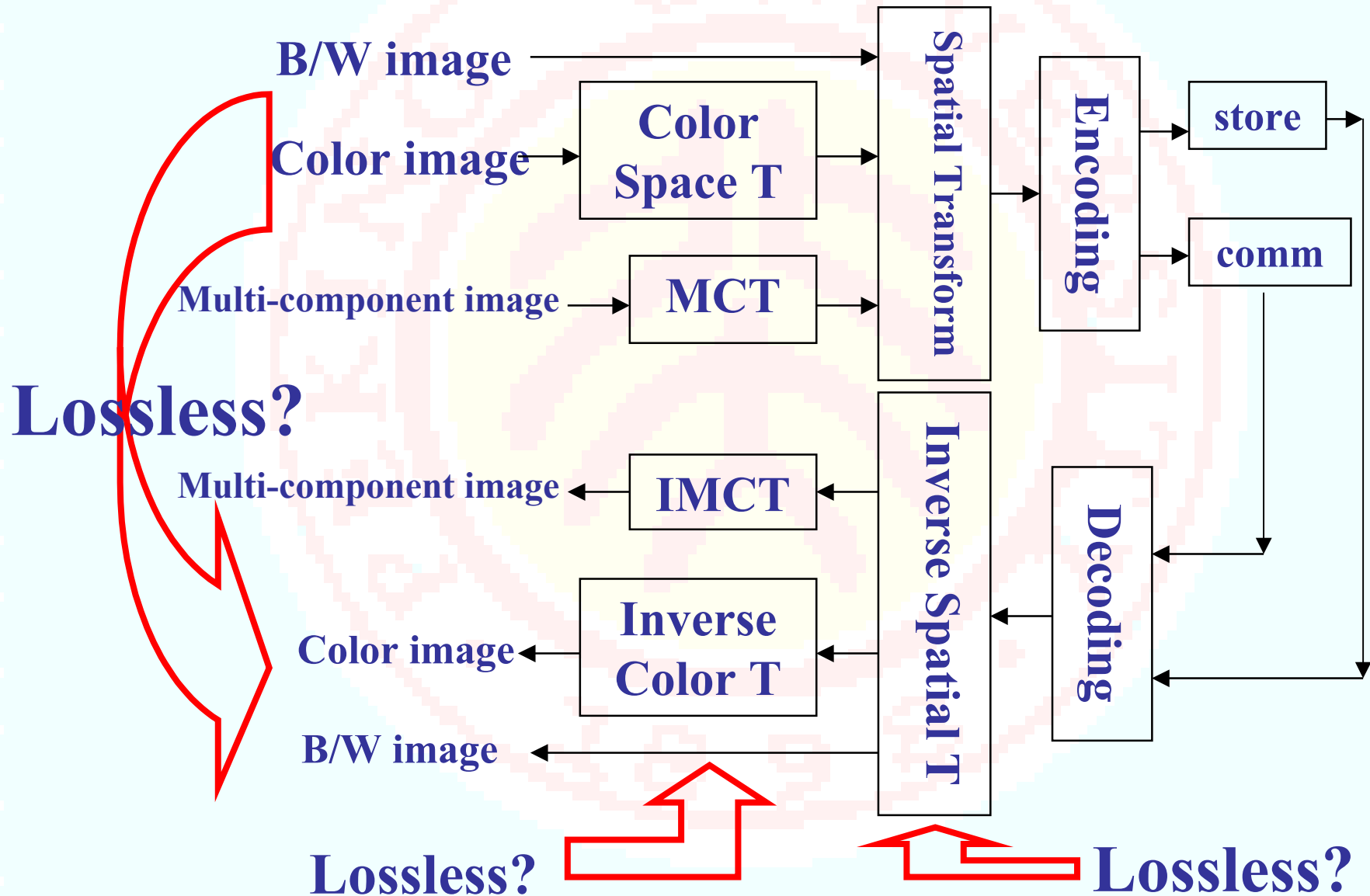
³Department of Statistics, Stanford University



Contents

- 1. Introduction**
- 2. Point & block factorizations**
- 3. Parallel ERM factorization (PERM)**
- 4. Parallel computational complexity**
- 5. Matrix blocking strategy**
- 6. Conclusions**

Why integer transform reversible?





How to implement?

- **Wavelet construction**
 - **S transform (Blume & Fand, 1989)**
 - **TS transform (Zandi et al, 1995)**
 - **S+P transform (Said & Pearlman, 1996)**
- **Ladder structure (Bruekers & van den Enden, 1992)**
- **Lifting scheme (2D, Sweldens, 1996)**
- **Approximated color transform (Gormish et al, 1997)**
- **General wavelet transform (2D, Daubechies et al, 1998)**



Matrix factorizations

- P. Hao and Q. Shi, Invertible linear transforms implemented by integer mapping, *Science in China, Series E (in Chinese)*, 2000, 30, pp. 132-141.
- P. Hao and Q. Shi, Matrix factorizations for reversible integer mapping, *IEEE Trans. Signal Processing*, 2001, 49 pp. 2314-2324.
- P. Hao and Q. Shi, Proposal of reversible integer implementation for multiple component transforms, *ISO/IEC JTC1/SC29/WG1N1720*, Arles, France, 2000.
- Y. She and P. Hao, A block TERM factorization of nonsingular uniform block matrices, *Science in China, Series E (in Chinese)*, 2004, 34(2).

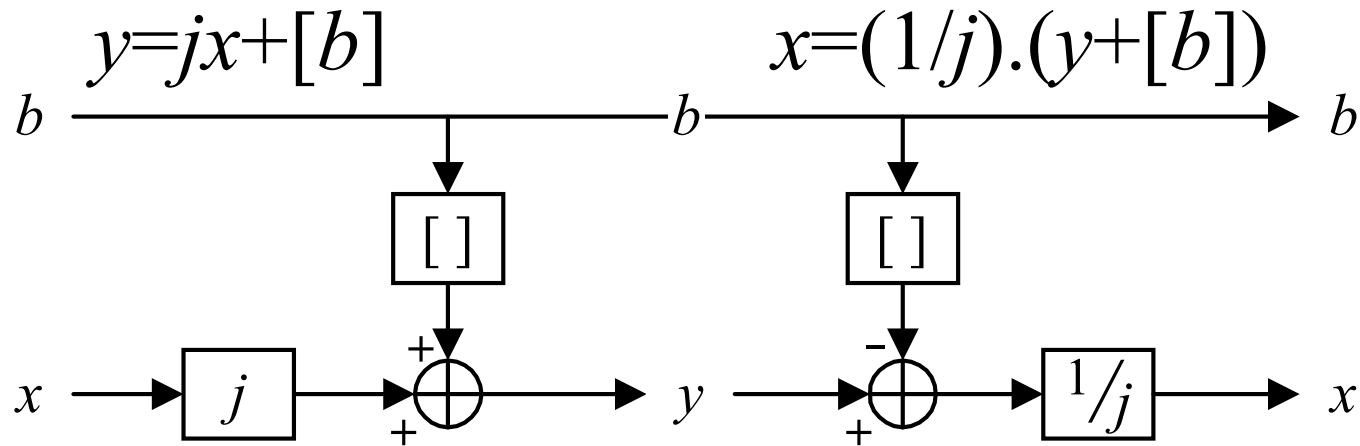


Can we make it more efficient?

- **Less factor matrices**
- **Less rounding error**
- **Integer computation**
- **Parallel computing**

How to increase the degree of parallelism?

Elementary reversible structure



- **Integer factor:** j
- Flexible rounding: round(), floor(), ceil(), ...
- Generalized lifting scheme: for $j = 1$, it is the same as ladder structure and the lifting scheme
- Implementation: $y = jx + [b]$ and $x = (1/j) \cdot (y + [b])$

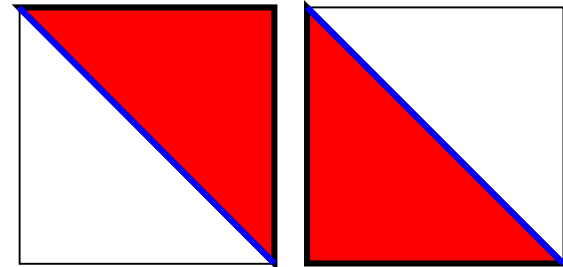


Elementary reversible matrix (ERM)

- Diagonal elements: **Integer factors**

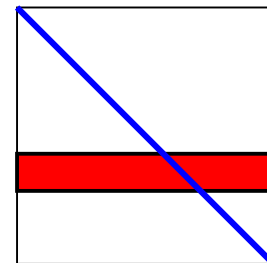
- Triangular ERM (TERM)

- Upper TERM
- Lower TERM



- Single-row ERM (SERM)

- $\mathbf{S}_m = \mathbf{J} + \mathbf{e}_m \mathbf{s}_m^T$
- Only one row off-diagonal nonzeros





Point factorizations (PLUS)

$$A = PLUS_0 D_R$$

$$LU = S_N S_{N-1} \cdots S_1$$

$$\text{if } \det P^T A = \det D_R \neq 0$$

$$D_R = \text{Diag}(1, 1, \dots, 1, \det(P^T A))$$

$$S_m = I + e_m s_m^T$$

$$S_0 = I + e_N s_0^T = I + e_N \cdot [s_1, s_2, \dots, s_{N-1}, 0]$$



Block factorizations (BLUS)

$$A = PLUS_0 D_R$$

$$LU = S_N S_{N-1} \cdots S_1$$

if $DET(P^T A) = DET(D_R)$ exists

$$D_R = \text{Diag}(I, I, \dots, I, DET(P^T A))$$

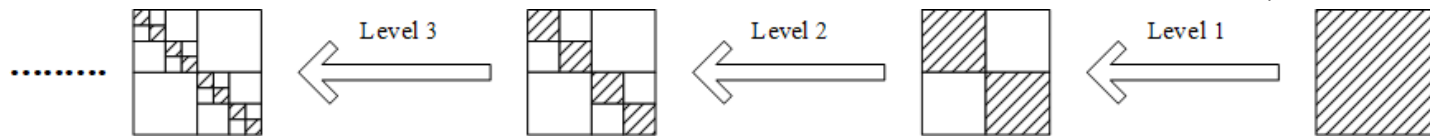
$$S_m = I + e_m s_m^T$$

$$S_0 = I + e_N s_0^T = I + e_N \cdot [s_1, s_2, \dots, s_{N-1}, 0]$$

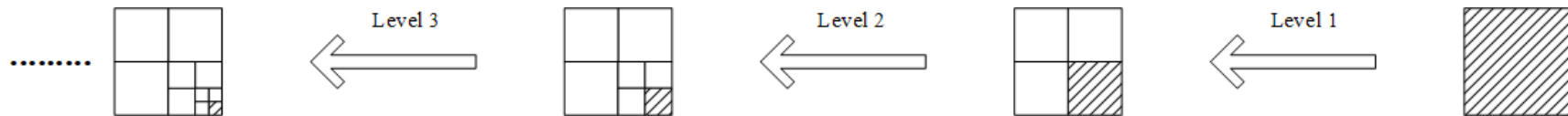
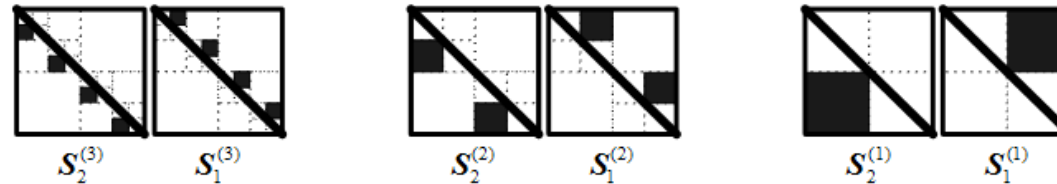
Parallel factorizations (PERM)

$$N_1 = m^{(0)} \xrightarrow{n^{(1)}} m^{(1)} \xrightarrow{n^{(2)}} m^{(2)} \cdots m^{(K-1)} \xrightarrow{n^{(K)}} m^{(K)} = N_2$$

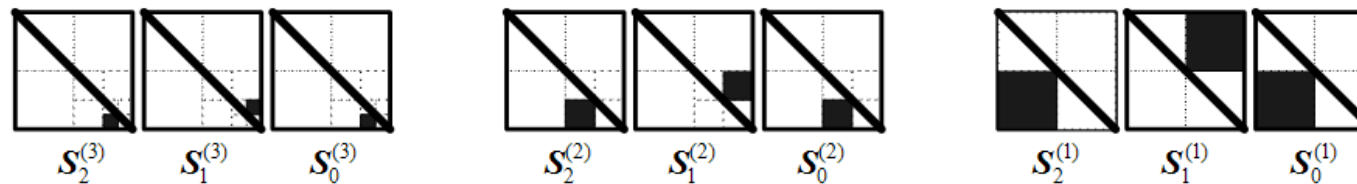
$$A = P^{(1)} P^{(2)} \cdots P^{(K)} D^{(K)} (L^{(K)} U^{(K)}) \cdots (L^{(1)} U^{(1)}) = PD \prod_{k=K}^1 S_{n^{(k)}}^{(k)} \cdots S_1^{(k)}$$



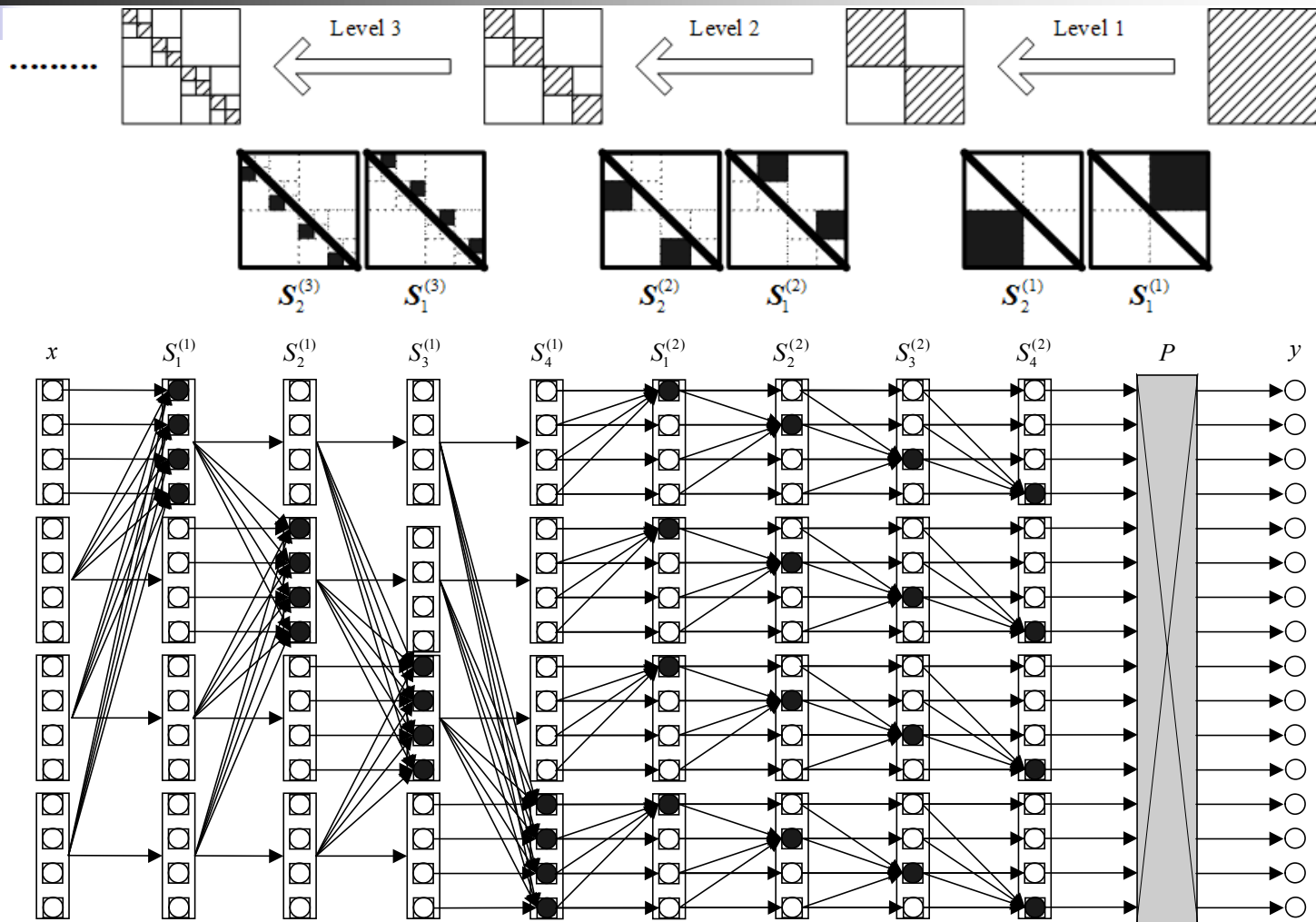
PERM⁽⁰⁾



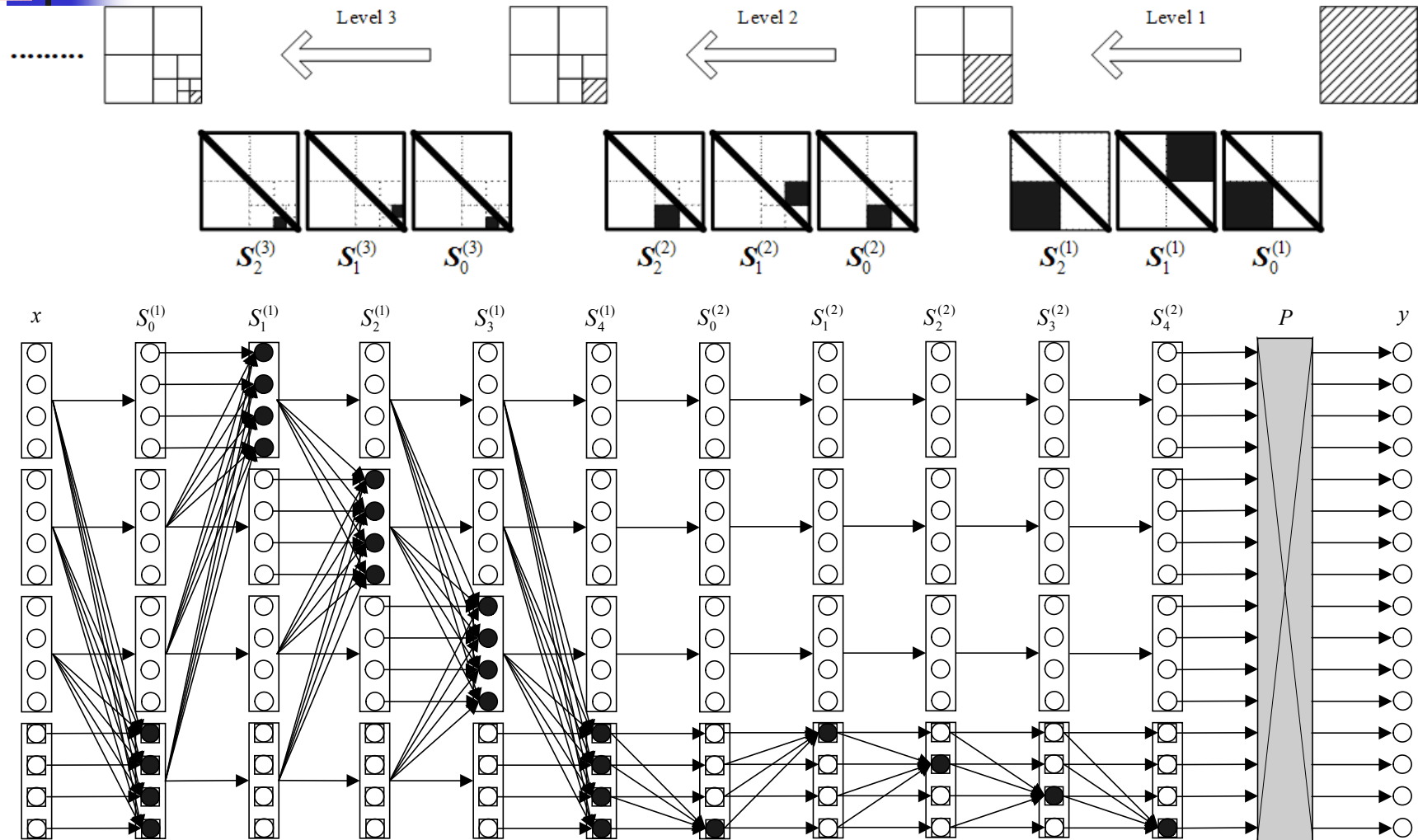
PERM⁽¹⁾



Parallel computing PERM⁽⁰⁾



Parallel computing PERM⁽¹⁾



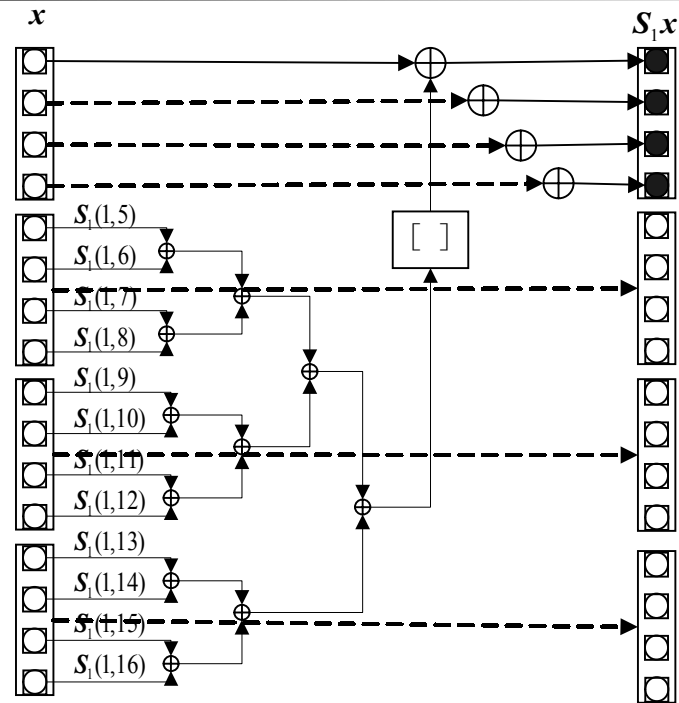


Parallel multiplication

For p processors to implement multiplications of n pairs of numbers the computational time is:

$$T^* = \left[\frac{n}{p} \right]$$

Parallel addition



$$T^+ = \begin{cases} \lceil \log_2 n \rceil & \text{if } n < 2p \\ \lceil n/p + C \log_2 p \rceil & \text{if } n \geq 2p \end{cases}$$



Computational complexity *

$$N_1 = m^{(0)} \xrightarrow{n^{(1)}} m^{(1)} \xrightarrow{n^{(2)}} m^{(2)} \cdots m^{(K-1)} \xrightarrow{n^{(K)}} m^{(K)} = N_2$$

For $n^{(k)} m^{(k)} = m^{(k-1)}$, $m^{(0)} = N_1$, $m^{(K)} = N_2$,
the parallel multiplication time is:

$$\begin{aligned} T_{PERM^{(0)}}^* &= \sum_{k=1}^K n^{(k)} \left[m^{(k)} (m^{(k-1)} - m^{(k)}) / p \right] \frac{N_1}{m^{(k-1)}} \\ &\approx \frac{N_1}{p} \sum_{k=1}^K (m^{(k-1)} - m^{(k)}) = \frac{N_1(N_1 - N_2)}{p} \end{aligned}$$

$$\begin{aligned} T_{PERM^{(1)}}^* &= \sum_{k=1}^K (n^{(k)} + 1) \left[m^{(k)} (m^{(k-1)} - m^{(k)}) / p \right] \\ &\approx \frac{1}{p} \sum_{k=1}^K ((m^{(k-1)})^2 - (m^{(k)})^2) = \frac{N_1^2 - N_2^2}{p} \end{aligned}$$

It's independent of the blocking manners.



Computational complexity +

$$N_1 = m^{(0)} \xrightarrow{n^{(1)}} m^{(1)} \xrightarrow{n^{(2)}} m^{(2)} \cdots m^{(K-1)} \xrightarrow{n^{(K)}} m^{(K)} = N_2$$

For $n^{(k)} m^{(k)} = m^{(k-1)}$, $m^{(0)} = N_1$, $m^{(K)} = N_2$, the parallel addition time:

$$T_{PERM^{(0)}}^+ = \sum_{k=1}^{K_p} n^{(k)} \left[\left(m^{(k)} (m^{(k-1)} - m^{(k)}) - p \right) / p + C \left(\log_2 p - \log_2 m^{(k)} \right) \right] \frac{N_1}{m^{(k-1)}} \\ + \sum_{k=K_p}^K n^{(k)} \left[\log_2 (m^{(k-1)} - m^{(k)}) \right] \frac{N_1}{m^{(k-1)}}$$

$$T_{PERM^{(1)}}^+ = \sum_{k=1}^{K_p} (n^{(k)} + 1) \left[\left(m^{(k)} (m^{(k-1)} - m^{(k)}) - p \right) / p + C \left(\log_2 p - \log_2 m^{(k)} \right) \right] \\ + \sum_{k=K_p}^K (n^{(k)} + 1) \left[\log_2 (m^{(k-1)} - m^{(k)}) \right]$$

There is a turning point K_p , where $m^{(K_p)} (m^{(K_p-1)} - m^{(K_p)})$
is close to but less than $2p$.



Blocking strategy

$$N_1 = m^{(0)} \xrightarrow{n^{(1)}} m^{(1)} \xrightarrow{n^{(2)}} m^{(2)} \cdots m^{(K-1)} \xrightarrow{n^{(K)}} m^{(K)} = N_2$$

Since the parallel computational time has a turning point (ignoring the factors like communication time)

We propose a three-phase blocking strategy

$$\text{if } N \geq 2p : N \rightarrow \cdots \rightarrow p$$

$$\text{if } 2\sqrt{p} \leq N < 2p : N \rightarrow \cdots \rightarrow \sqrt{p}$$

$$\text{if } N \leq 2\sqrt{p} : N \rightarrow \cdots \rightarrow 1$$



Computational complexity

$$N_1 = m^{(0)} \xrightarrow{n^{(1)}} m^{(1)} \xrightarrow{n^{(2)}} m^{(2)} \dots m^{(K-1)} \xrightarrow{n^{(K)}} m^{(K)} = N_2$$

$$T_{\text{PERM}^{(1)}}^*(N, p) = \begin{cases} f_1^*(N, p) = \left(\frac{N}{p} + 1\right) \cdot \left(\frac{N}{p} - 1\right) \cdot p + f_2^*(p, p) & p \leq \frac{N}{2} \\ f_2^*(N, p) = \left(\frac{N}{\sqrt{p}} + 1\right) \cdot \left(\frac{N}{\sqrt{p}} - 1\right) + f_3^*(\sqrt{p}, p) & \frac{N}{2} < p < \frac{N^2}{4} \\ f_3^*(N, p) = 5 \log_4 N & p \geq \frac{N^2}{4} \end{cases}$$

$$T_{\text{PERM}^{(1)}}^+(N, p) = \begin{cases} f_1^+(N, p) = \left(\frac{N}{p} + 1\right) \cdot \left(\frac{N}{p} - 1\right) \cdot p + f_2^+(p, p) & p \leq \frac{N}{2} \\ f_2^+(N, p) = \left(\frac{N}{\sqrt{p}} + 1\right) \cdot \left(\frac{N}{\sqrt{p}} - 1 + C \log_2 \sqrt{p}\right) + f_3^+(\sqrt{p}, p) & \frac{N}{2} < p < \frac{N^2}{4} \\ f_3^+(N, p) = 5 \log_4 N (\log_4 9N - 1) & p \geq \frac{N^2}{4} \end{cases}$$



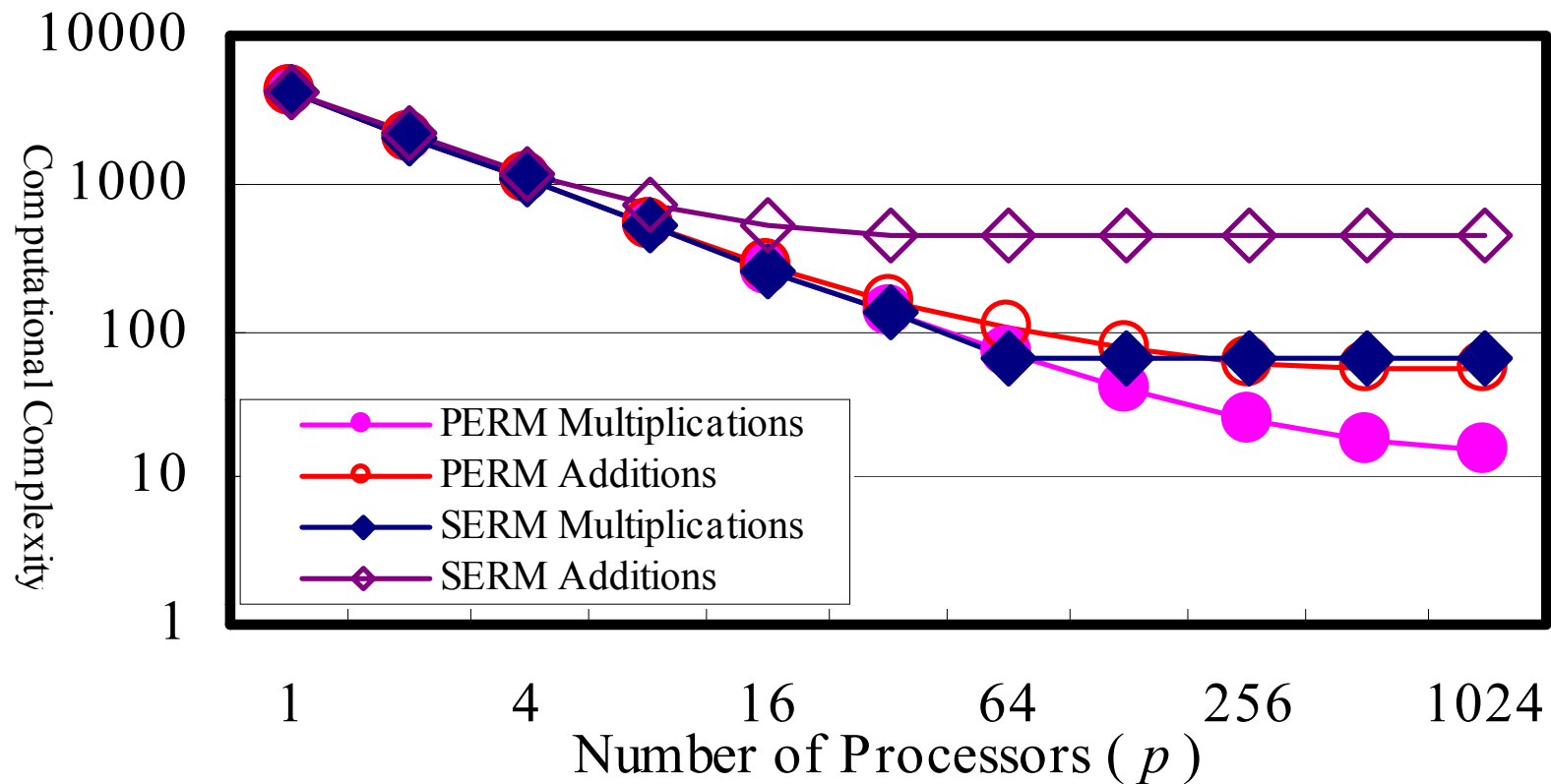
Complexity comparison

$$T_{p\text{SERM}^{(1)}}^*(N, p) = (N + 1) \frac{N - 1}{p}$$

$$T_{p\text{SERM}^{(1)}}^+(N, p) = (N + 1) \left(\frac{N - 1}{p} + C \log_2 p \right)$$

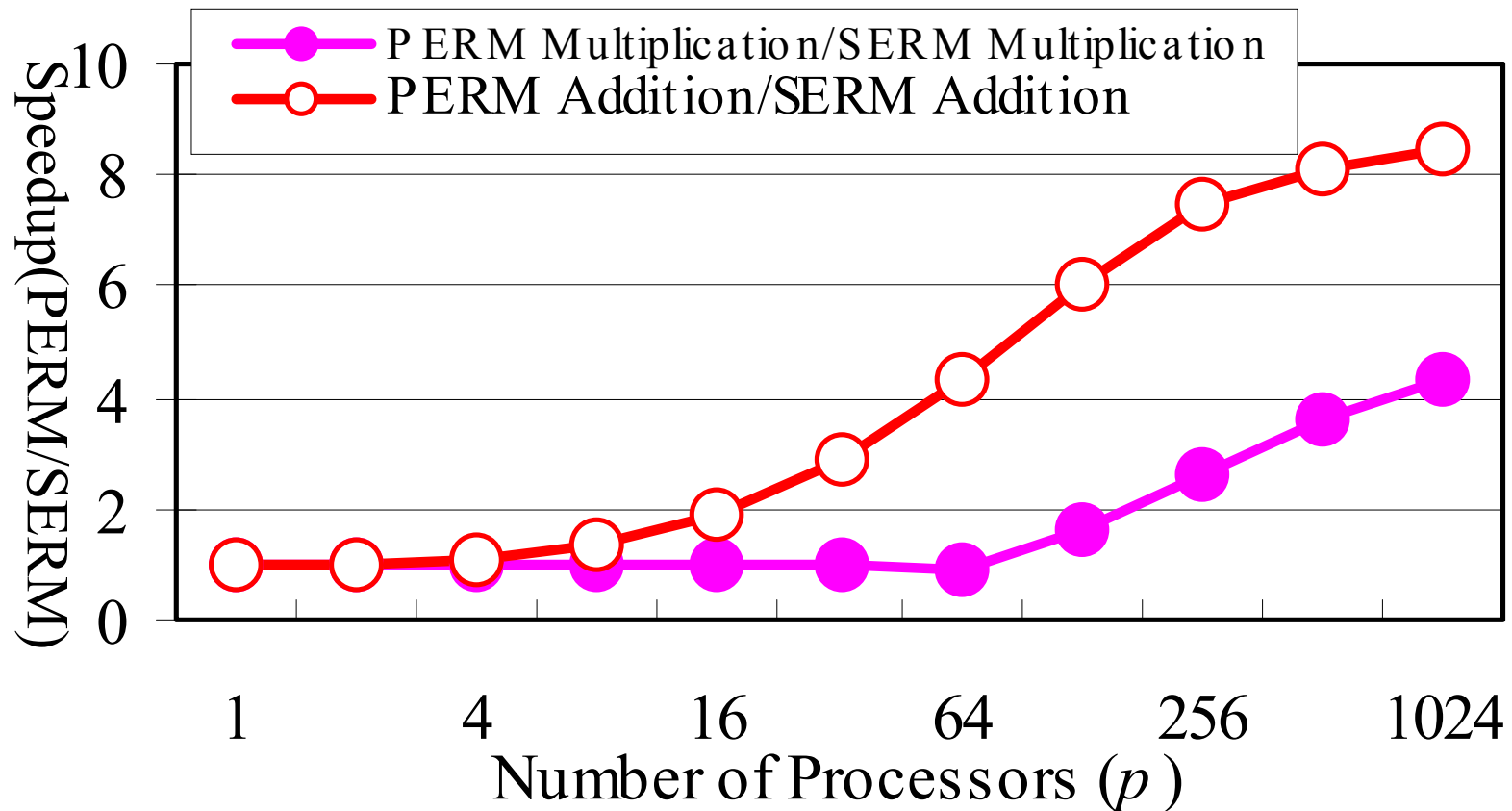
Operation		p	$O(N)$	$O(N^2)$
		Multiplications	SERM ⁽¹⁾	$O(N)$
PERM ⁽¹⁾	$O(N)$		$O(\log N)$	
Additions	SERM ⁽¹⁾	$O(M \log N)$	$O(M \log N)$	
	PERM ⁽¹⁾	$O(N)$	$O(\log^2 N)$	

PERM vs. parallel SERM



Computational complexity ($N = 64, C = 1$)

PERM vs. parallel SERM



Relative speedup($N = 64, C = 1$)



Conclusions

- **For parallel computing:**
 - **Increase the degree of parallelism**
 - **Accommodate more processors**
- **For sequential computing:**
 - **May be more efficient for sequential computing with special matrix computation software such as BLAS**
- **More factorization levels possibly result in greater rounding error**



Thank You

`phao@cis.pku.edu.cn`

`phao@dcscs.qmul.ac.uk`

`http://www.dcs.qmul.ac.uk/~phao`