Parallel FFT-algorithms

Shmeleva Yulia
Saint-Petersburg State University
Department of Computational Physics

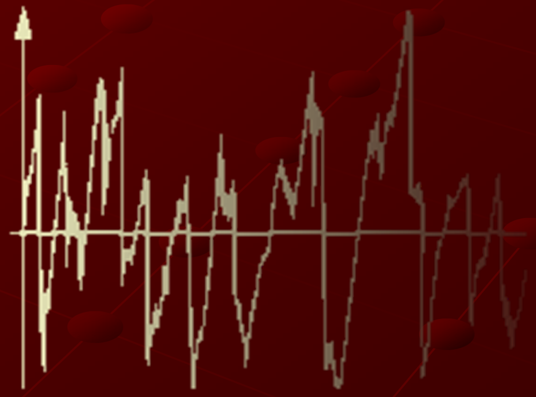
Joint Advanced Student School 2009

Outline

- Motivation
- Mathematical theory
- Serial FFT
- Parallel FFT
  - Binary-exchange algorithm
  - Transpose algorithm
- Conclusion

Motivation

- Linear partial differential equations
- Waveform analysis
- Convolution and correlation
- Digital signal processing
- Image filtering



Continuous Fourier Transform

- (Forward) Fourier Transform

$$H(f) = \int_{-\infty}^{+\infty} h(t) e^{2\pi i f t} dt, \text{ where } i = \sqrt{-1}$$

- (Inverse) Fourier Transform

$$h(t) = \int_{-\infty}^{+\infty} H(f) e^{-2\pi i f t} df, \text{ where } i = \sqrt{-1}$$

Discrete Fourier transform

- Finite time series , sampled at an interval Δ

$$h = < h[0], h[1], ..., h[N-1] >$$

where

$$h[k] = h[t_k] = h[k\Delta]$$

$$k = 0, 1, ..., \mathrm{N} - 1$$

Discrete Fourier transform

- The discrete (forward) Fourier transform

$$H = < H[0], H[1], ..., H[N-1] >$$

where

$$H[j] = \sum_{k=0}^{N-1} h[k] e^{2\pi i k j / N},$$

$$j = 0, 1, ..., \mathrm{N}-1$$

Discrete Fourier transform

- Let

$$W_N = e^{2\pi i / N}$$

- The discrete (forward) Fourier transform

$$H[j] = \sum_{k=0}^{N-1} h[k] W_N^{\;jk}$$

- The discrete (inverse) Fourier transform

$$h[k] = \frac{1}{N} \sum_{j=0}^{N-1} H[j] W_N^{\;-jk}$$

Discrete Fourier transform

Each *H[j]* requires *N* multiplications

$$\Downarrow$$

The entire sequence *H* needs an order of

N^2 operations!

DFT property of symmetry

Recall $W_N = e^{2\pi i/N} \Rightarrow \quad \mathrm{W_N}^N = 1, \quad W_N^{N/2} = -1$

If we associate $h[k] \Leftrightarrow H[j]$

then

$$h[-k] \Leftrightarrow H[-j]$$

$$h[k+l] \Leftrightarrow W^{-lk}\mathrm{H[j]}$$

$$\mathrm{W}^{\mathrm{lk}} h[k] \Leftrightarrow H[j+l]$$

Serial FFT (Cooley and Tukey, 1965)

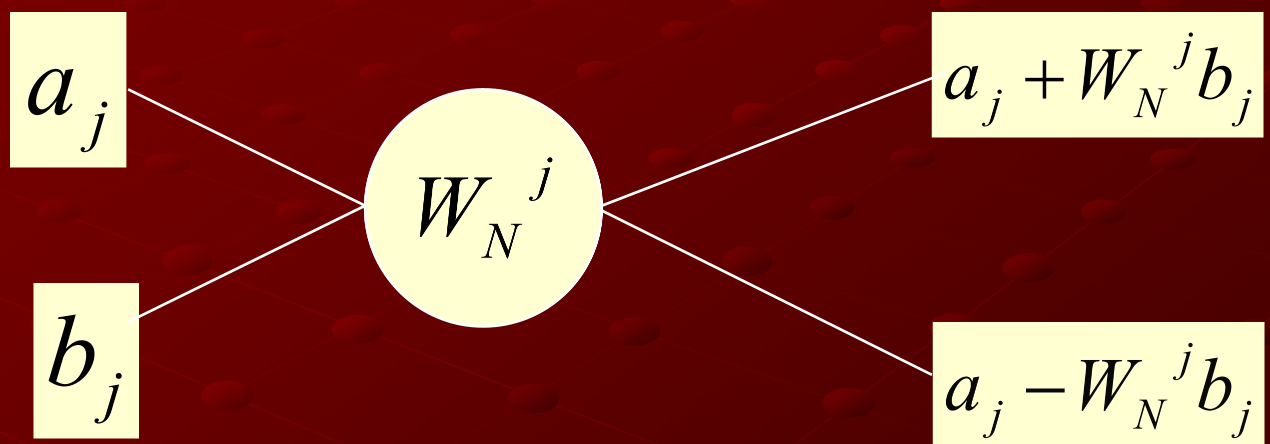
Assume that $N = 2^d$

$$H[j] = \sum_{k=0}^{N/2-1} h[2k] W_N^{2kj} + W_N^{j} \sum_{k=0}^{N/2-1} h[2k+1] W_N^{2kj}$$

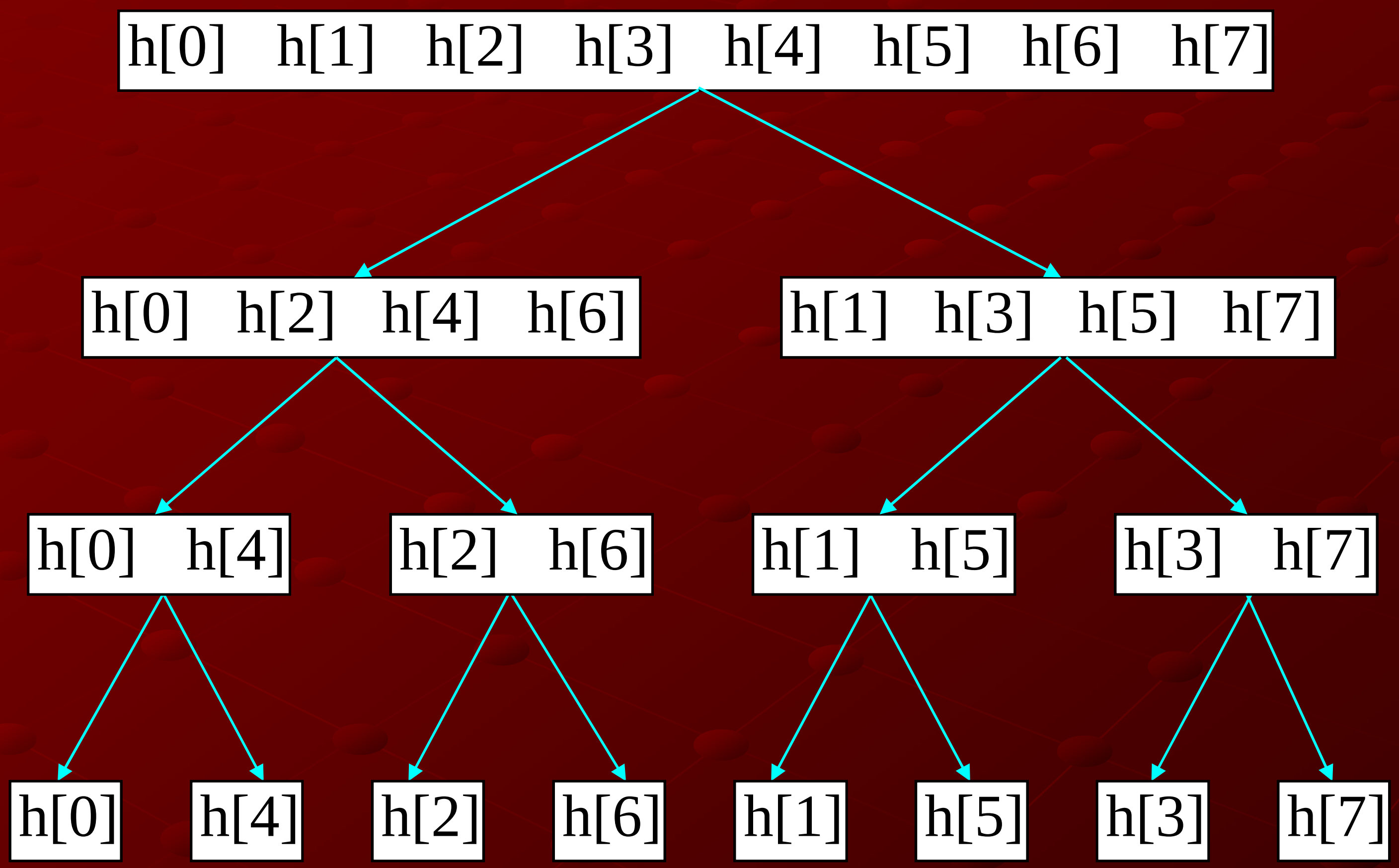
$$H[j+N/2] = \sum_{k=0}^{N/2-1} h[2k] W_N^{2kj} - W_N^{j} \sum_{k=0}^{N/2-1} h[2k+1] W_N^{2kj}$$

Serial FFT

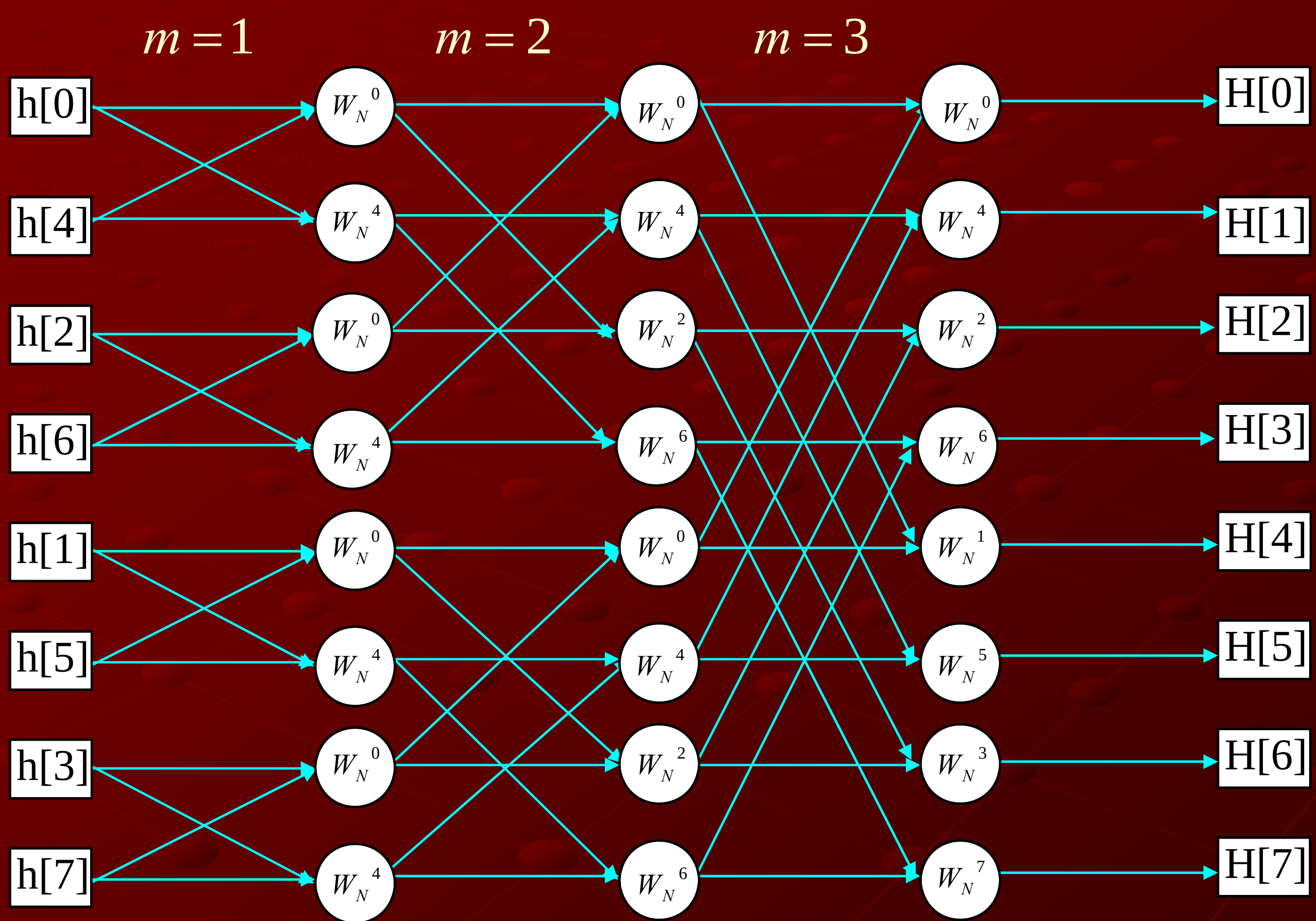
Butterfly



Serial FFT (decomposition)



Serial FFT (Cooley and Tukey, 1965)



Serial FFT (Cooley and Tukey, 1965)

- $\log_2 N$ iterations
- During each iteration – N operations

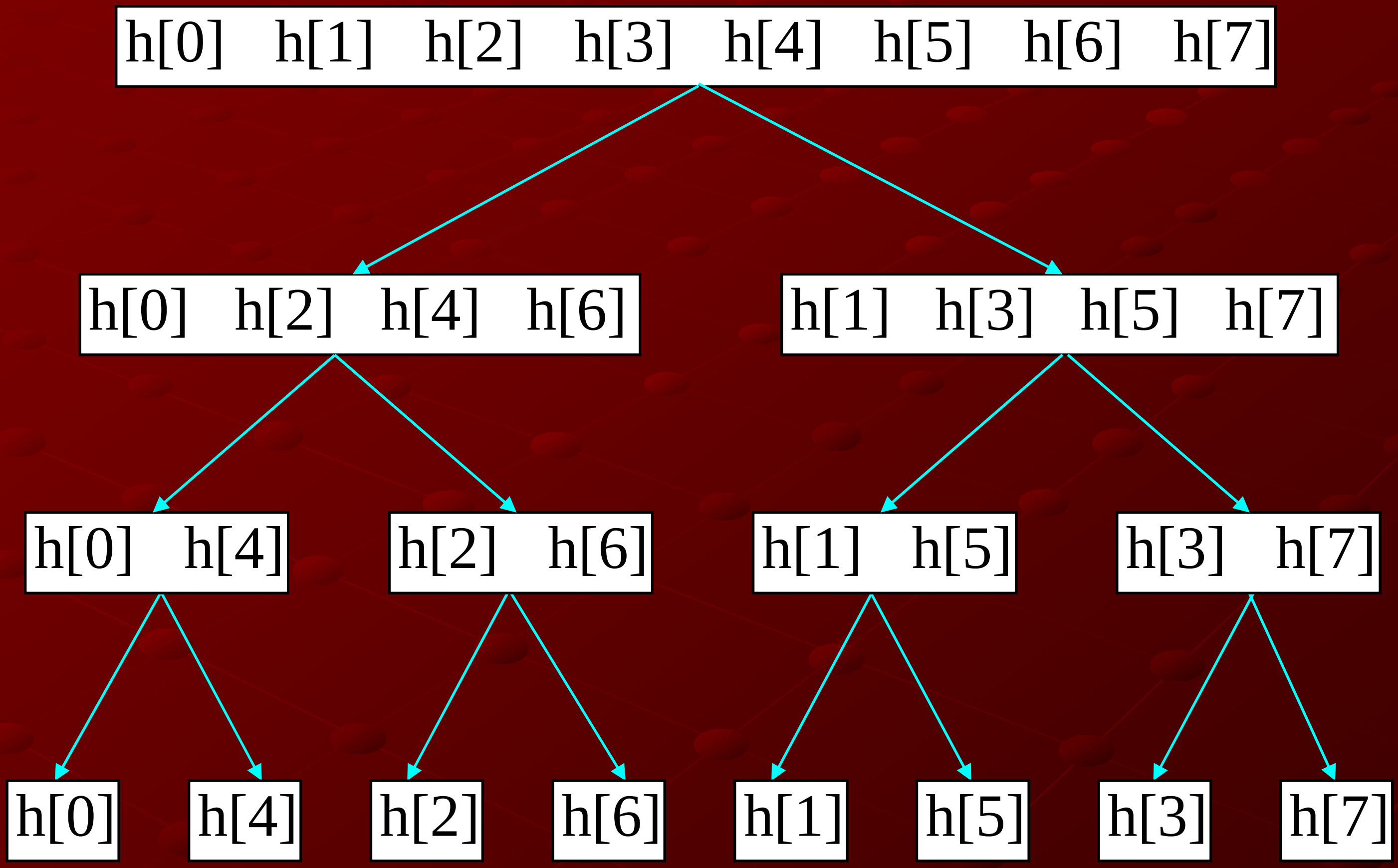
Serial FFT

Compare!

Original DFT: N^2 operations

FFT: $N \log_2 N$ operations

Serial FFT (decomposition)



Serial FFT

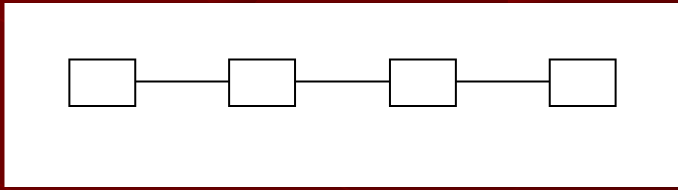
- Bit reversal sorting algorithm

Original sequence			Rearranged sequence	
decimal	binary		binary	decimal
0	000		000	0
1	001		100	4
2	010		010	2
3	011	$\Rightarrow$	110	6
4	100		001	1
5	101		101	5
6	110		011	3
7	111		111	7

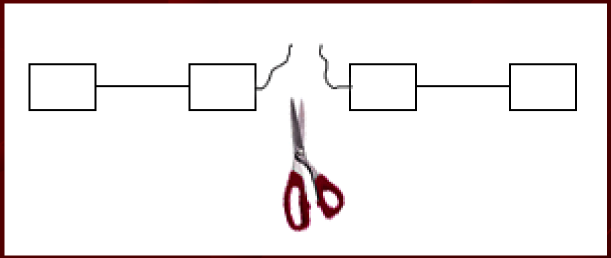
Definitions

- Bisection width –

  minimum number of communication links that can be removed to break a network into two equal sized disconnected networks



bisection width is 1

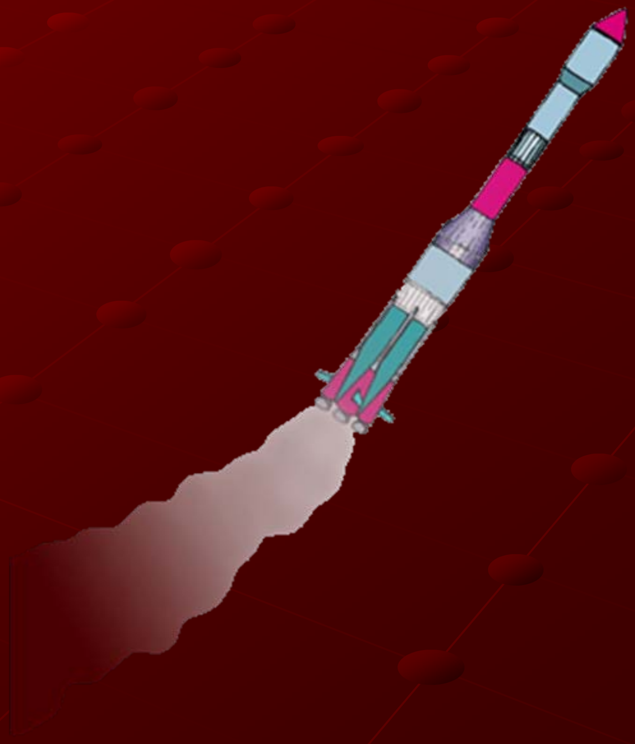


Definitions

- Speedup –

  measure that gives the relative benefit of solving a problem in parallel

$$S = \frac{T_s}{T_p}$$



Definitions

- Efficiency –

  measure of the fraction of time for which a processing element is usefully employed

$$E = \frac{S}{p}$$

Definitions

- Scalability –

  measure of capacity to increase speedup in proportion to the number of processing elements in order to maintain efficiency fixed.

Definitions

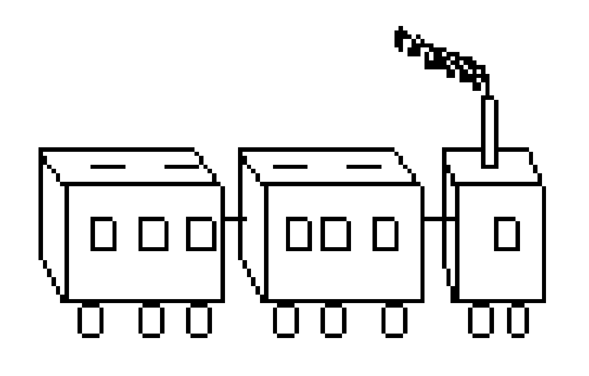
- Problem size –

  number of basic computation steps in the best sequential algorithm to solve the problem on a single processing element

Definitions

- Isoefficiency function –

  function which dictates the growth rate of problem size required to keep the efficiency fixed as a number of processors increases.



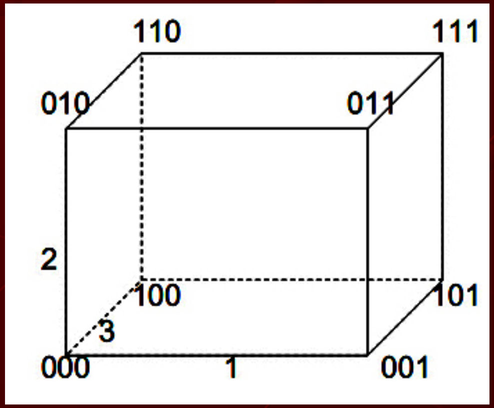
Parallel FFT

1. The Binary-Exchange algorithm
2. The transpose algorithm

Binary-Exchange algorithm

- Full bandwidth network
  - p parallel processes
  - bisection width is an order of p

Example: hypercube network



Binary-Exchange algorithm

- Simple mapping:

One task ↔ one process

p = N

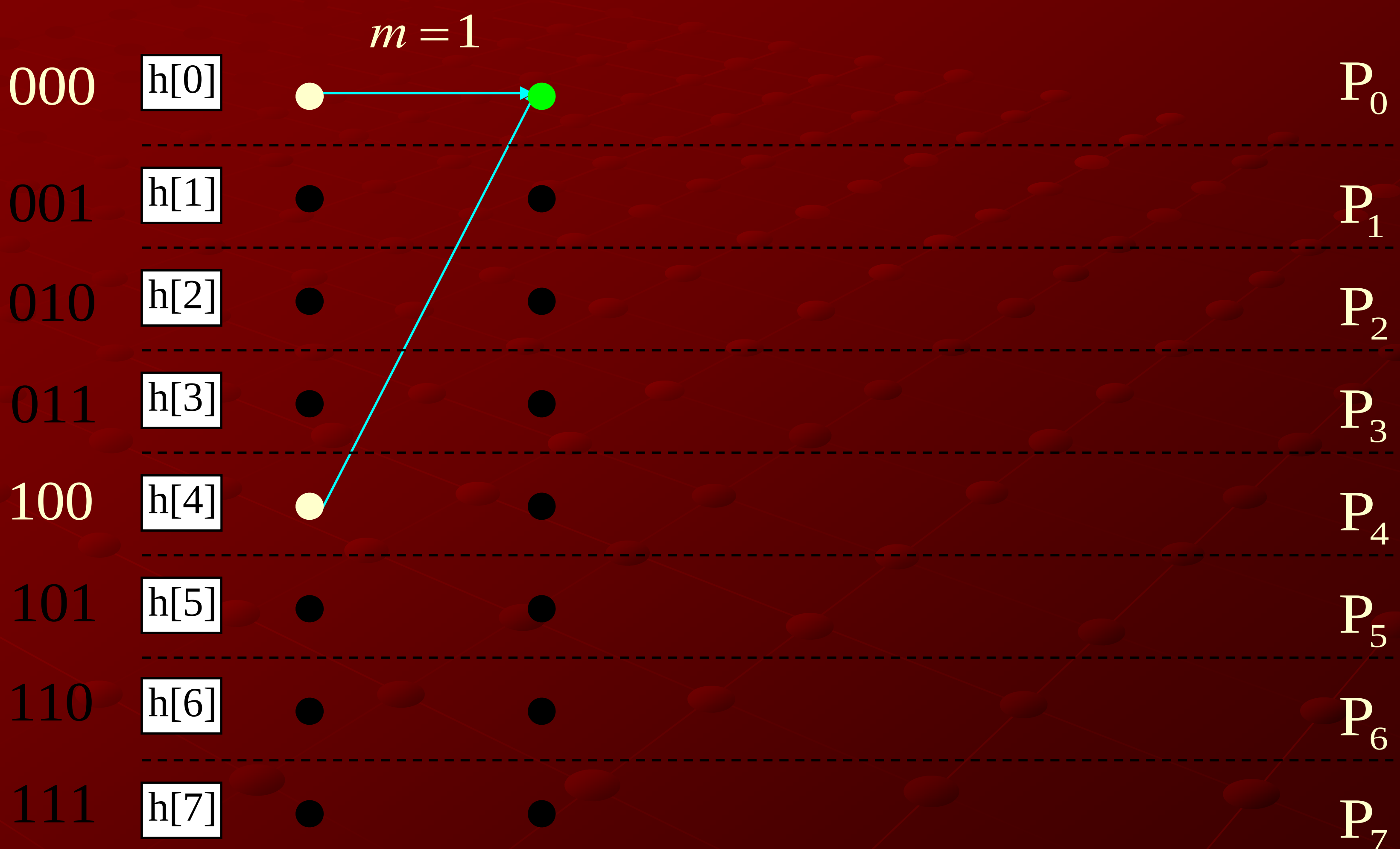
- Assume that

$$N = 2^r$$

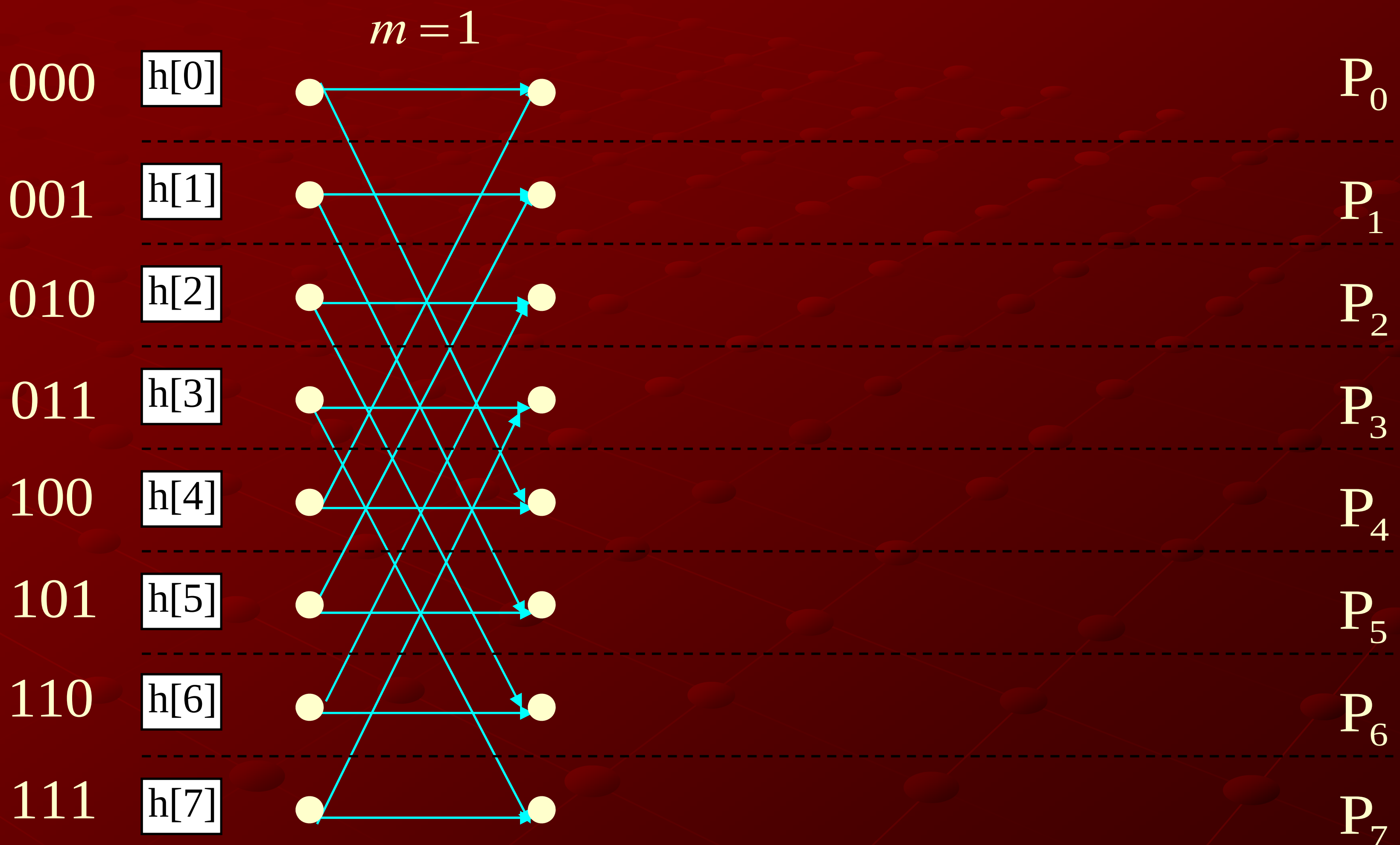
Binary-Exchange algorithm

Binary-Exchange algorithm

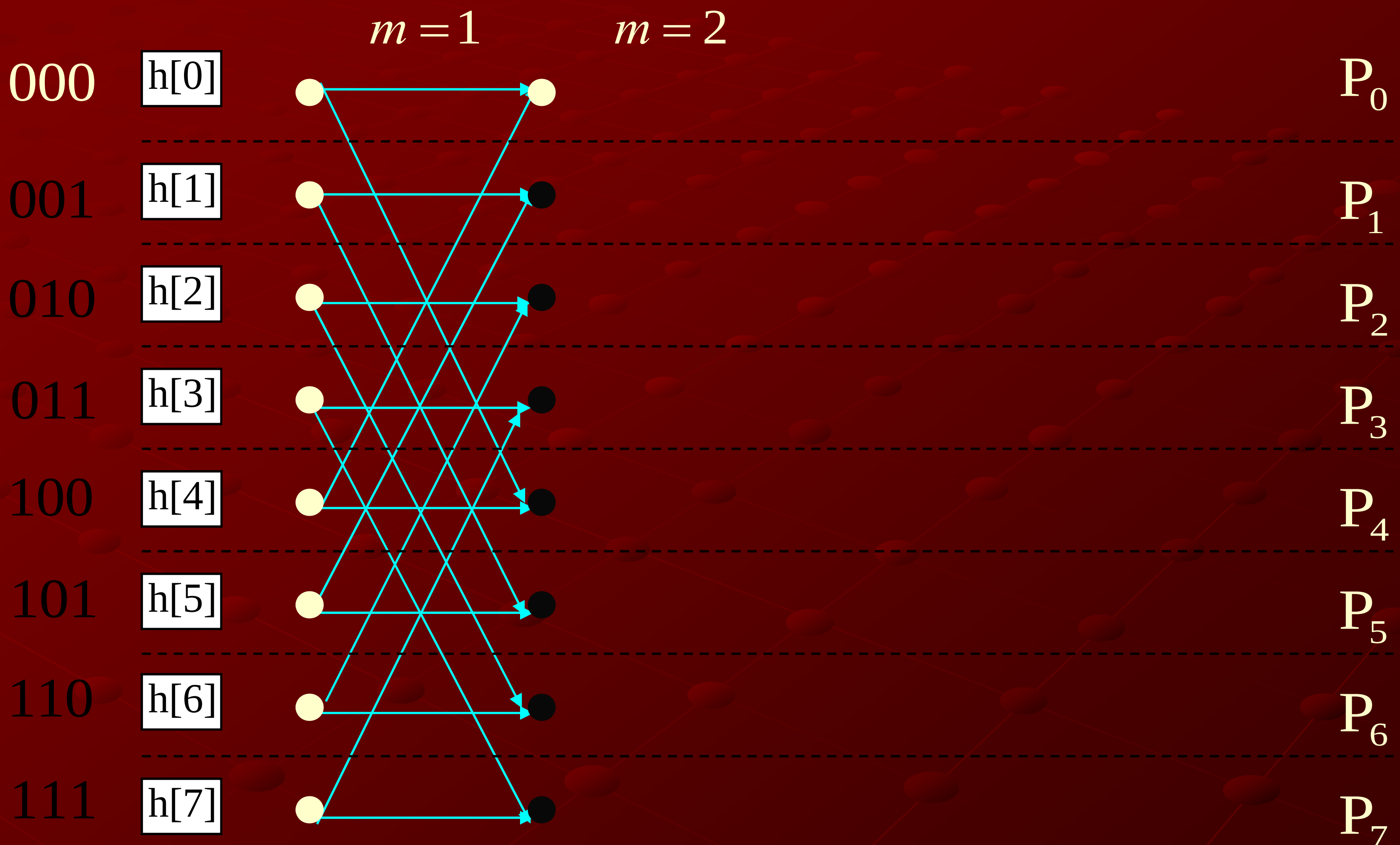
Binary-Exchange algorithm



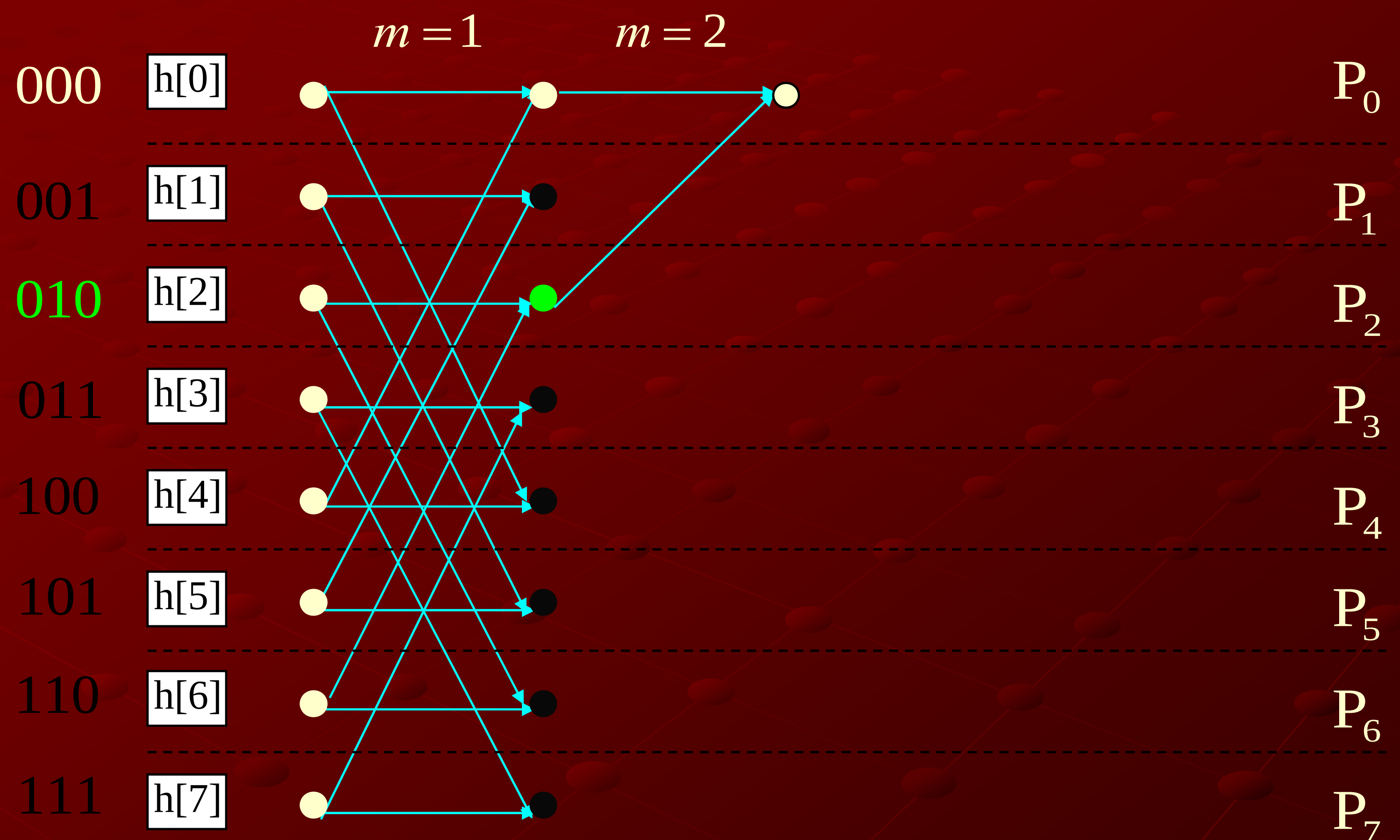
Binary-Exchange algorithm



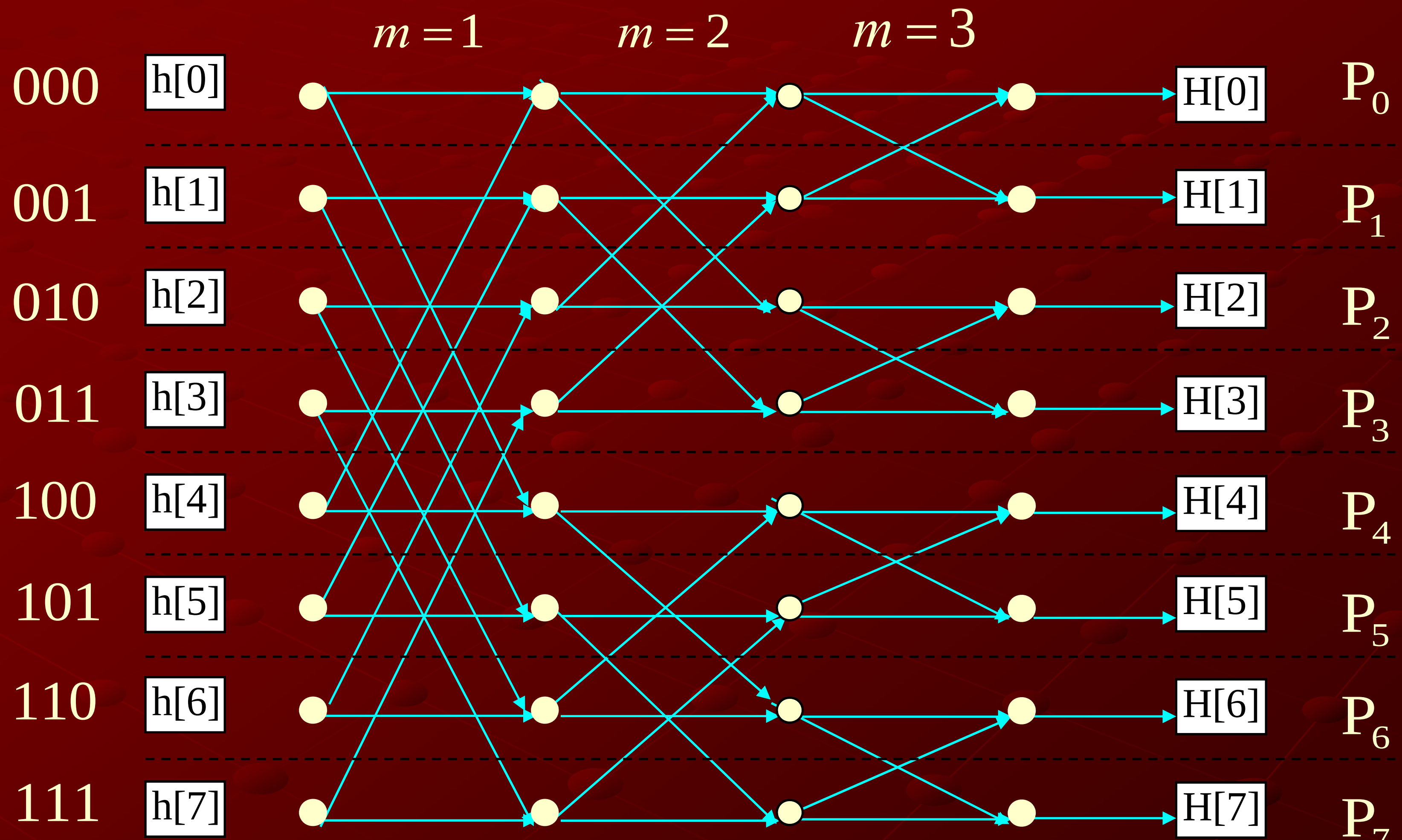
Binary-Exchange algorithm



Binary-Exchange algorithm



Binary-Exchange algorithm



Binary-Exchange algorithm

- Another mapping:

multiple tasks ↔ one process

$p < N$

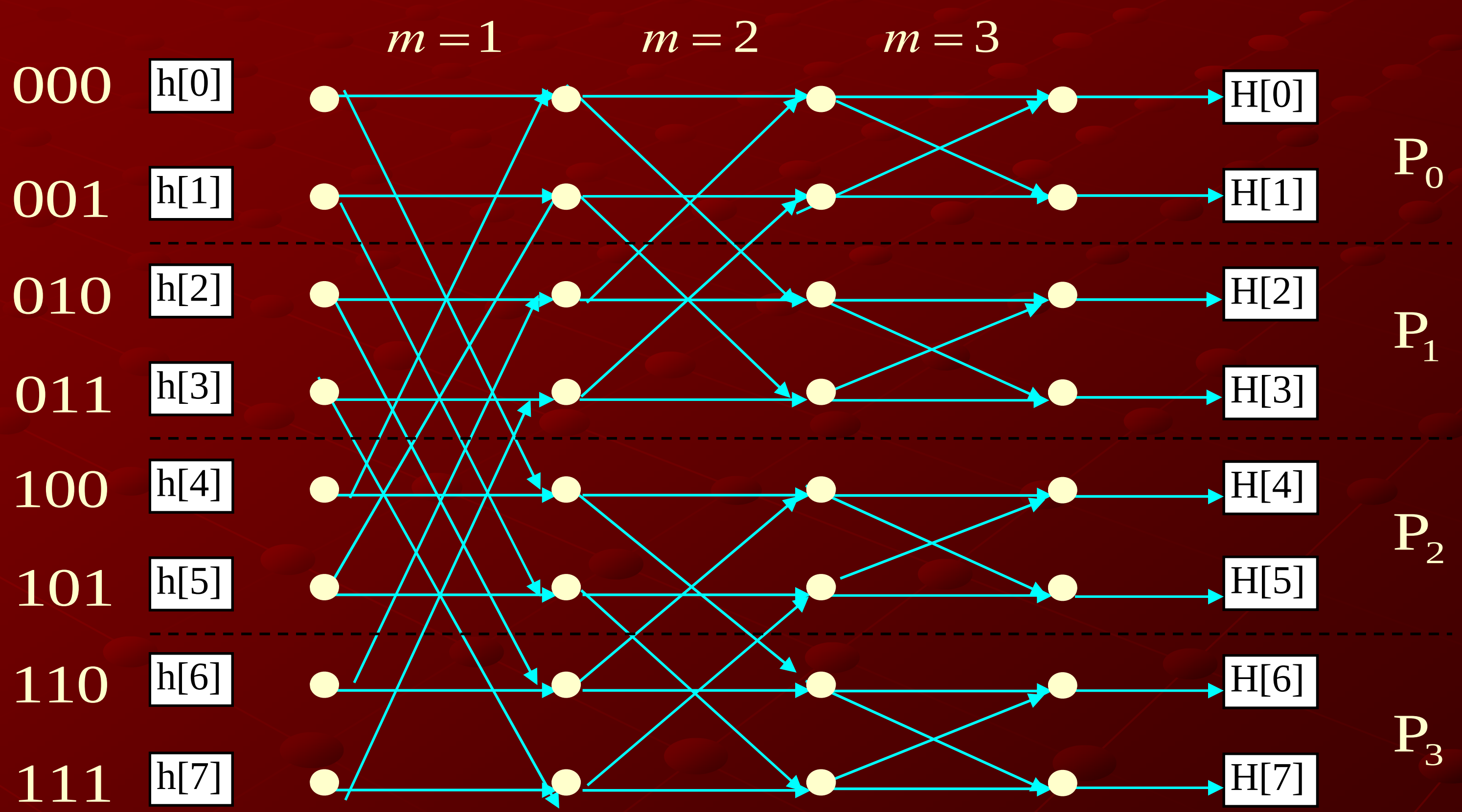
- Assume that $N = 2^r$, $p = 2^d$

- Partition the sequence into blocks of N/p elements

one block ↔ one process

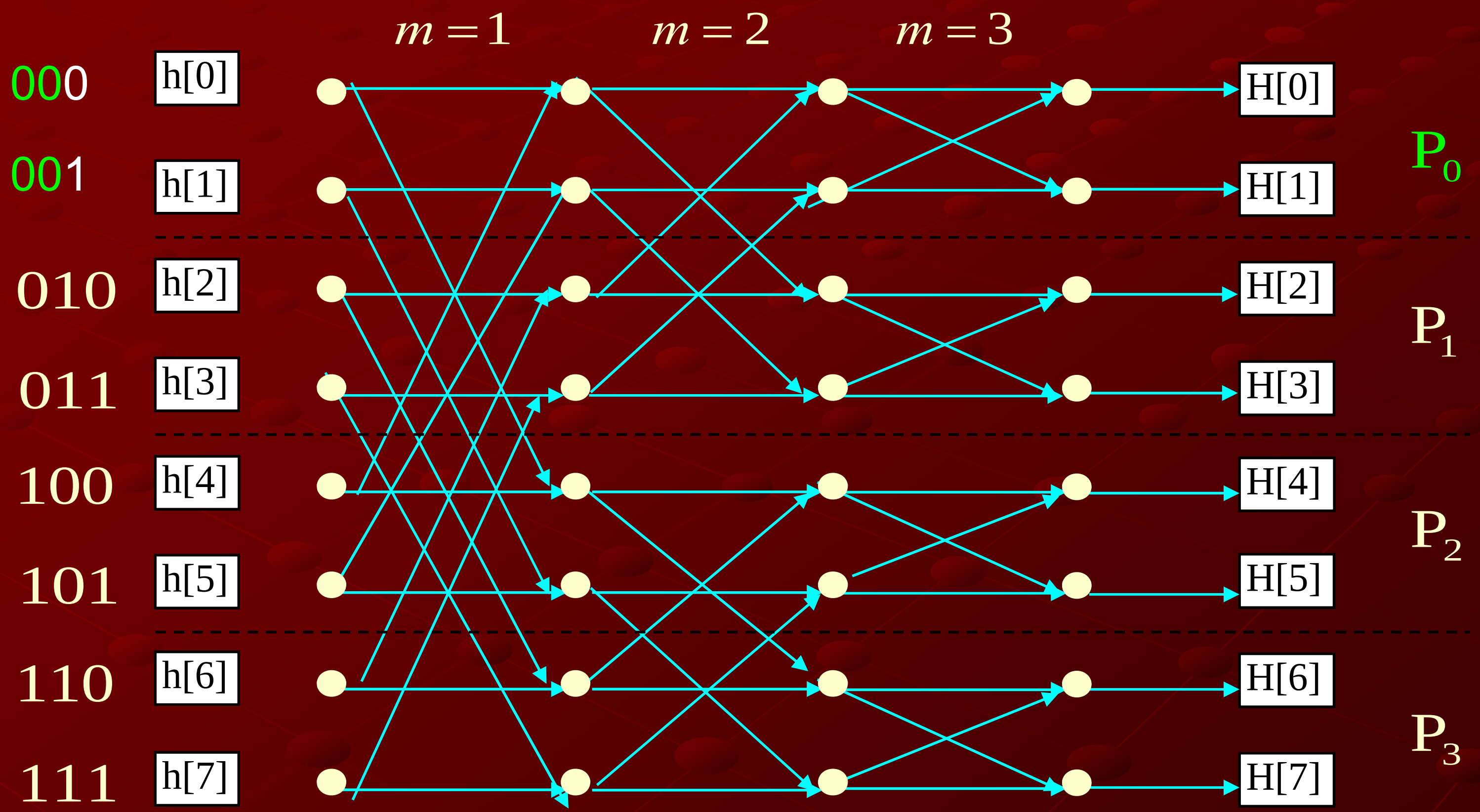
Binary-Exchange algorithm

- Consider $N = 8, \ p = 4 \ (d = 2, r = 3)$



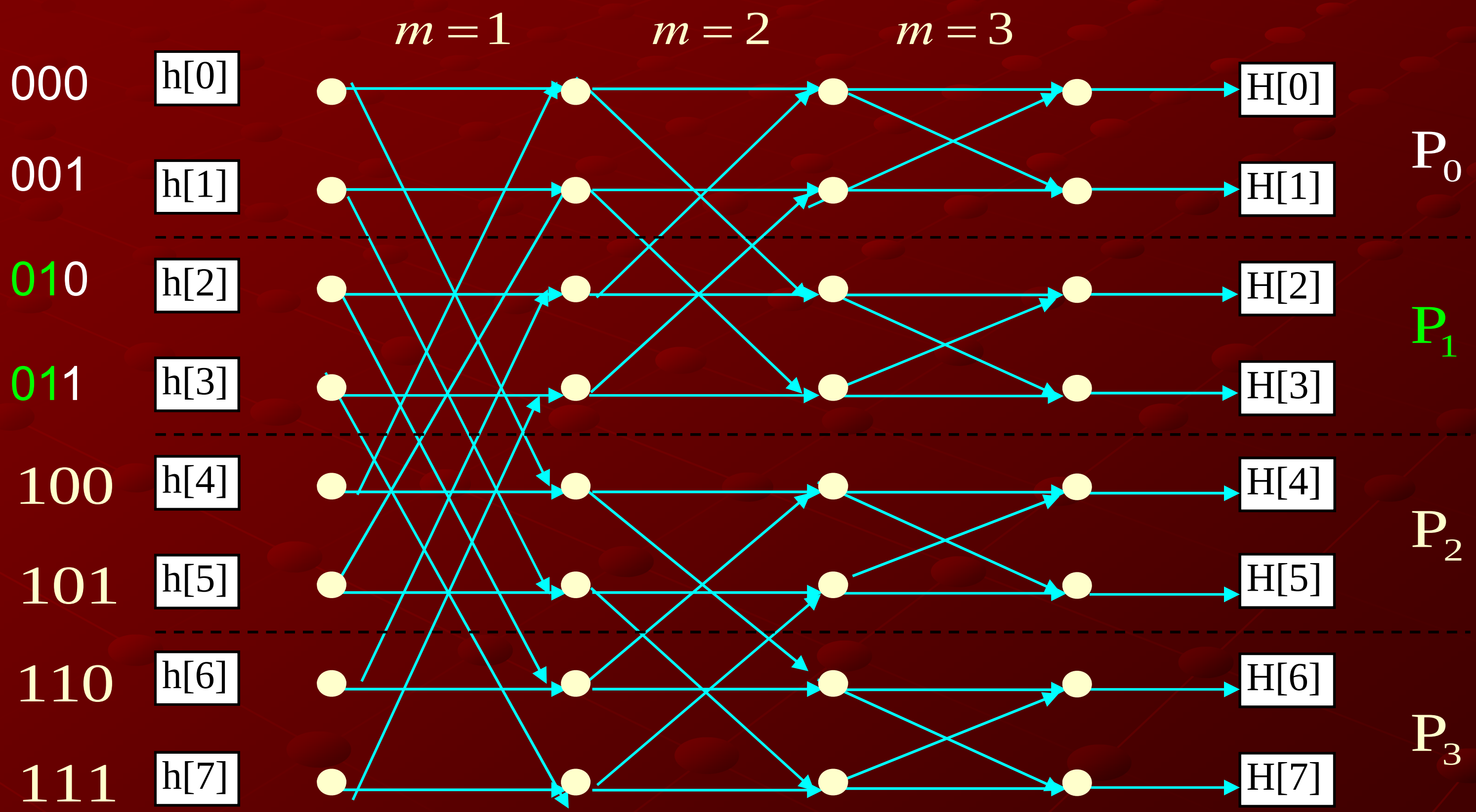
Binary-Exchange algorithm

- Consider $N = 8, p = 4\,(d = 2, r = 3)$



Binary-Exchange algorithm

- Consider $N = 8,\ p = 4\ (d = 2, r = 3)$



Binary-Exchange algorithm

First d iterations $\longrightarrow$ *N/p* words of data exchange

$$d = \log_2 p$$

Last *(r-d)* iterations $\longrightarrow$ No interprocess interaction

$$r = \log_2 N$$

Binary-Exchange algorithm

- Denote:

  t_s – the startup time for the data transfer

  t_w – the per-word transfer time

  t_c – computation time

t_c

Binary-Exchange algorithm

- The parallel run time

$$T_p = t_c \frac{N}{p} \log_2 N + t_s \log_2 p + t_w \frac{N}{p} \log_2 p$$

- Speedup

$$S = \frac{t_c N \log_2 N}{T_p} = \frac{pN \log_2 N}{N \log N + (t_s/t_c) p \log_2 p + (t_w/t_c) N \log_2 p}$$

Binary-Exchange algorithm

- Efficiency

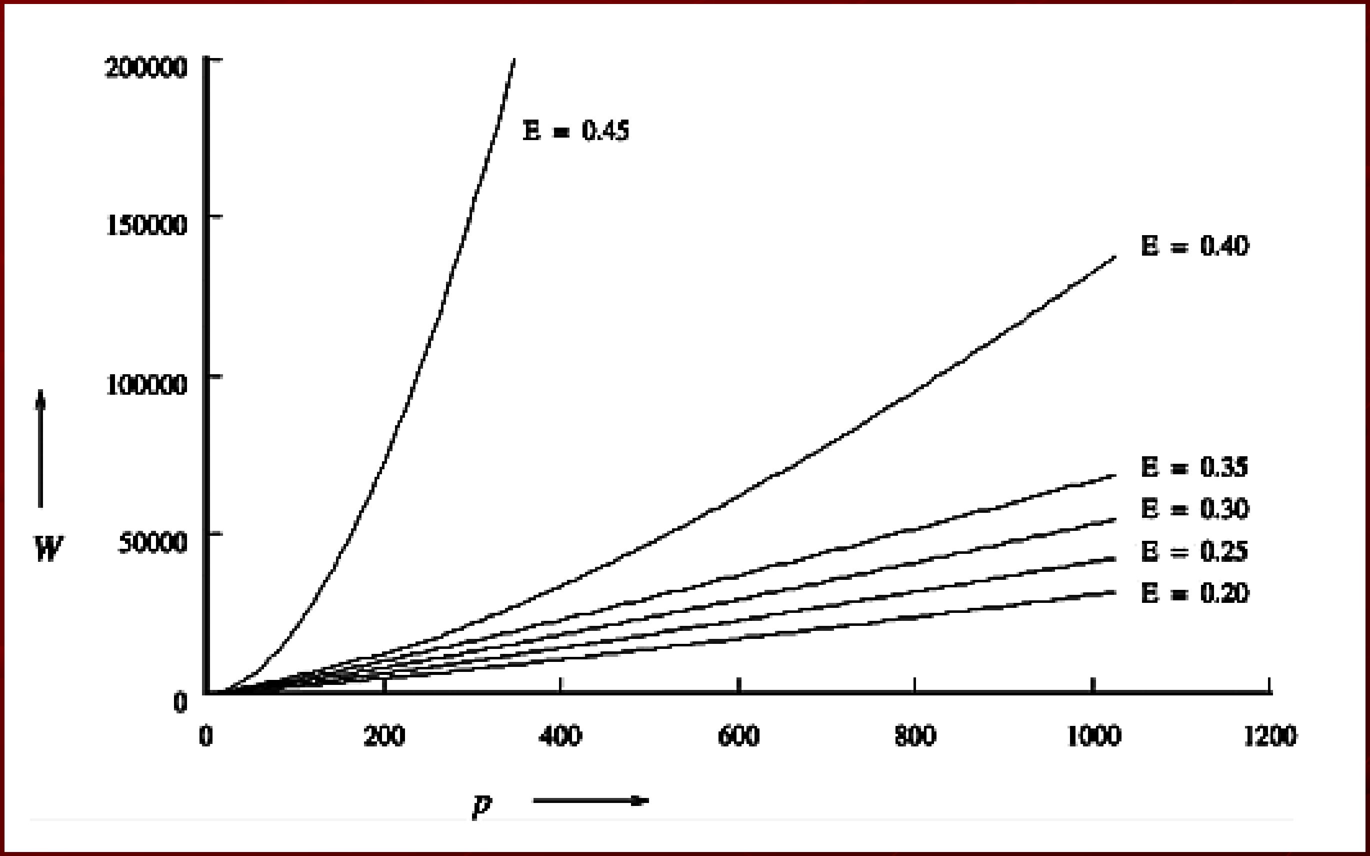
$$E = \frac{1}{1 + (t_s p \log_2 p)/(t_c N \log_2 N) + (t_w \log_2 p)/(t_c \log_2 N)}$$

- Scalability

$$W = \max\left\{ p \log_2 p, K \frac{t_s}{t_c} p \log_2 p, K \frac{t_w}{t_c} p^{Kt_w/t_c} \log_2 p \right\}, \quad K = \frac{E}{1-E}$$

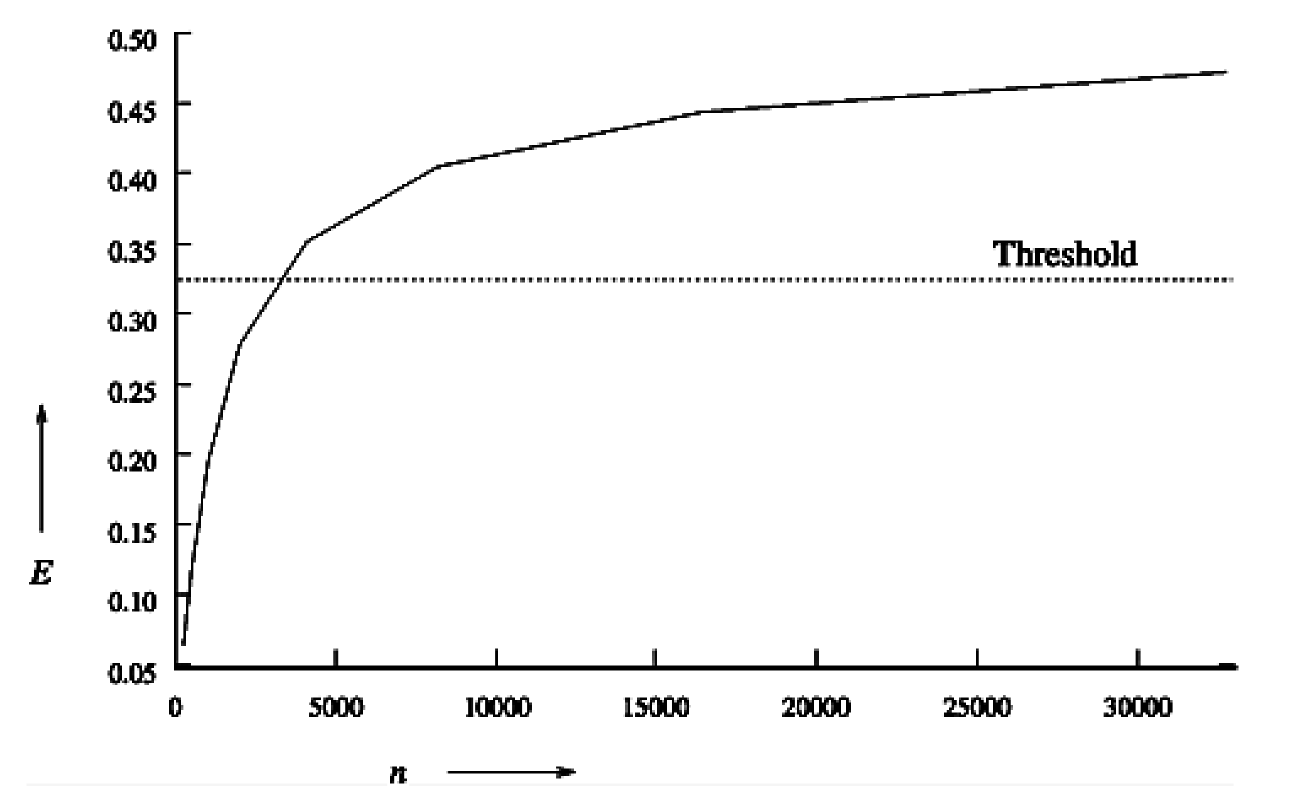
Binary-Exchange algorithm

- Assume that $t_c = 2, t_w = 4, t_s = 25$



Binary-Exchange algorithm

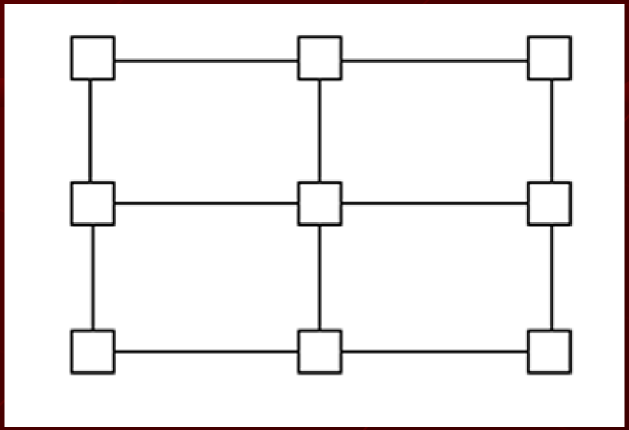
- Assume that $t_c = 2, t_w = 4, t_s = 25, p = 256$



Binary-Exchange algorithm

- Limited bandwidth network
  - p parallel processes
  - bisection width is less then $\Theta(p)$

Example: a mesh interconnection network



Binary-Exchange algorithm

- Mapping:

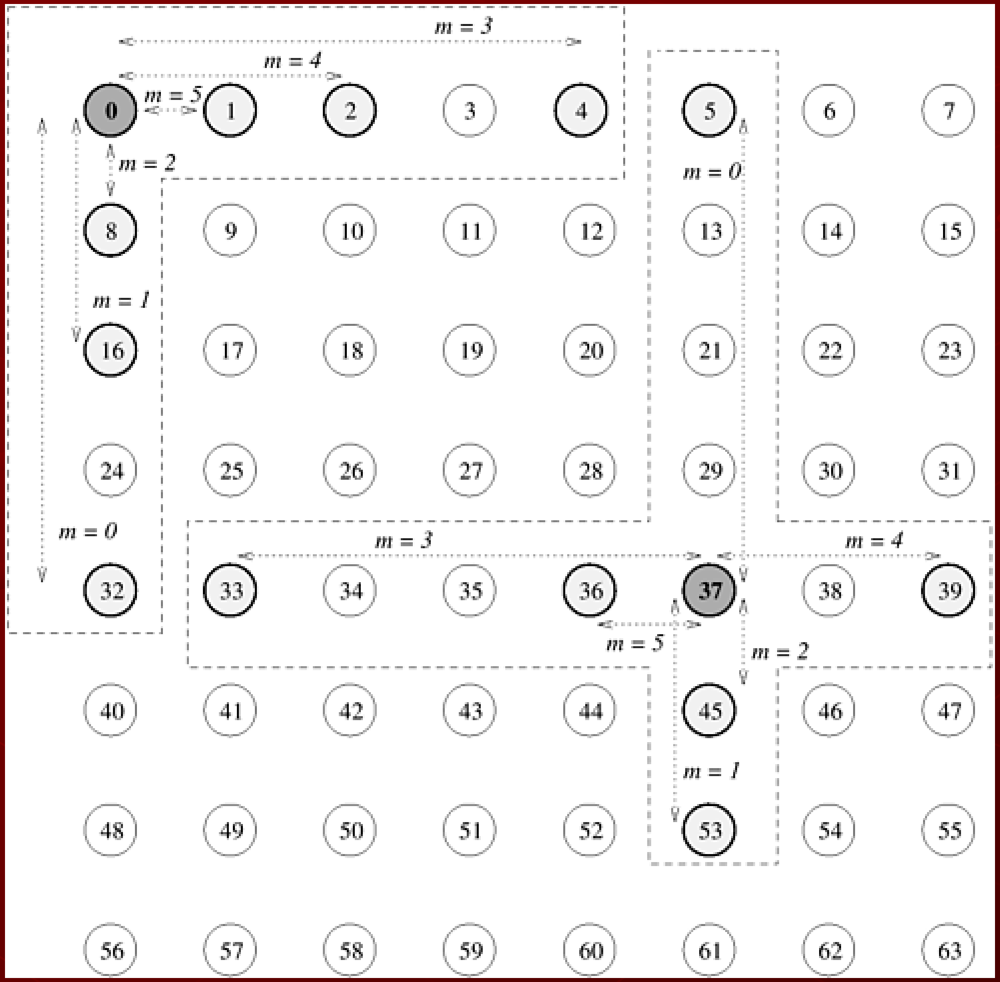
$$N \text{ tasks} \leftrightarrow \sqrt{p} \times \sqrt{p} \text{ processes}$$

- Assume that

$$N = 2^r$$

$$p = 2^d$$

Binary-Exchange algorithm



$\log_2 \sqrt{p}$ steps –

communicating processes are in the same row

$\log_2 \sqrt{p}$ steps –

communicating processes are in the same column

t_c

Binary-Exchange algorithm

- The parallel run time

$$T_p = t_c \frac{N}{p} \log_2 N + t_s \log_2 p + 2t_w \frac{N}{\sqrt{p}}$$

- Speedup

$$S = \frac{pN \log_2 N}{N \log N + (t_s/t_c) p \log_2 p + 2(t_w/t_c) N \sqrt{p}}$$

Binary-Exchange algorithm

- Efficiency

$$E = \frac{1}{1 + (t_s p \log_2 p)/(t_c N \log_2 N) + 2(t_w \sqrt{p})/(t_c \log_2 N)}$$

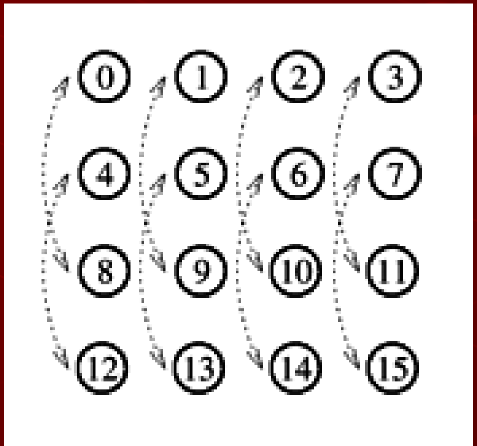
- Scalability

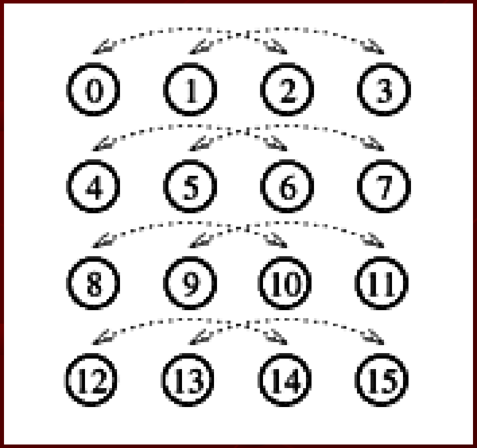
$$W = \max\left\{ p \log_2 p,\, K \frac{t_s}{t_c} p \log_2 p,\, 2K \frac{t_w}{t_c} p^{2(Kt_w/\mathrm{t_c})\sqrt{p}} \sqrt{p} \right\}, \quad K = \frac{E}{1-E}$$

Transpose algorithm

- Two-dimensional transpose algorithm

sequence of size $N \rightarrow \sqrt{N} \times \sqrt{N}$ array





Transpose algorithm

1. A $\sqrt{N}$-point FFT is computed for each column of the initial array

2. The array is transposed

3. A $\sqrt{N}$-point FFT for each column of the transposed array

Transpose algorithm

- Full bandwidth network
  - bisection width is an order of p
  - example: hypercube

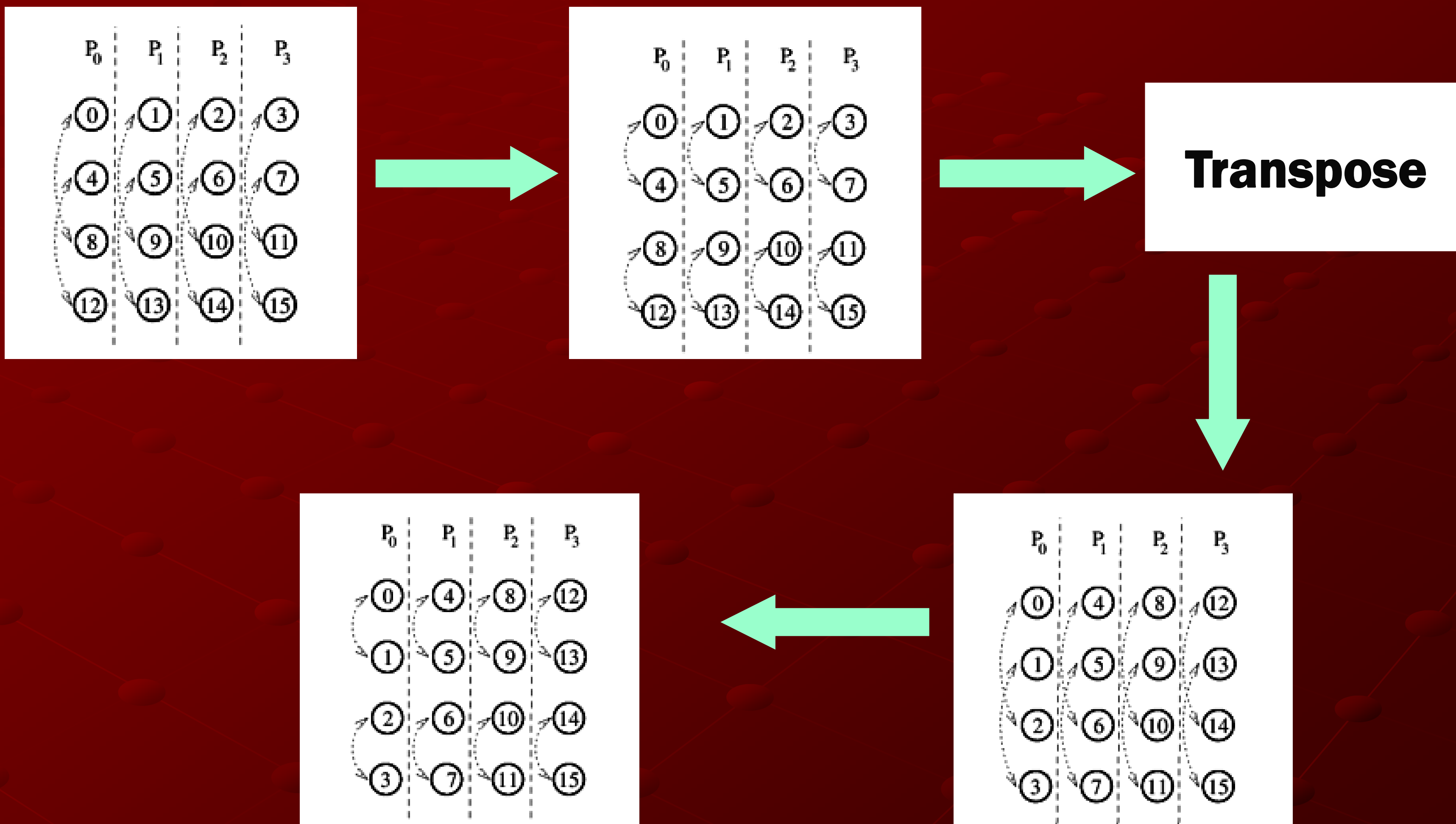
- Simple mapping

one column ↔ one process

$$p = \sqrt{N}$$

- Assume that $\sqrt{N} = 2^r$, $p = 2^d$

Transpose algorithm



Transpose algorithm

- Another mapping:

Several columns ↔ one process

$$p < \sqrt{N}$$

- Assume that $\sqrt{N} = 2^r$, $p = 2^d$

- Partition the array into blocks of $\sqrt{N} / p$ rows

one block ↔ one process

t_c

Transpose algorithm

- The parallel run time

$$T_p = t_c \frac{N}{p} \log_2 N + t_s (p-1) + t_w \frac{N}{p}$$

- Speedup

$$S \approx \frac{pN \log_2 N}{N \log_2 N + (t_s / t_c) p^2 + (t_w / t_c) N}$$

Transpose algorithm

- Efficiency

$$E \approx \frac{1}{1 + (t_s p^2)/(t_c N \log_2 N) + t_w/(t_c \log_2 N)}$$

- Scalability

$$W = \Theta(p^2 \log_2 p)$$

Transpose algorithm

Compare two algorithms!

Binary-exchange algorithm

$$T_p = t_c \frac{N}{p}\log_2 N + \underline{t_s \log_2 p} + t_w \frac{N}{p}\log_2 p$$

Transpose algorithm

$$T_p = t_c \frac{N}{p}\log_2 N + \underline{t_s (p-1)} + t_w \frac{N}{p}$$

Transpose algorithm

Compare two algorithms!

Binary-exchange algorithm

$$T_p = t_c \frac{N}{p} \log_2 N + t_s \log_2 p + t_w \frac{N}{p} \log_2 p$$

Transpose algorithm

$$T_p = t_c \frac{N}{p} \log_2 N + t_s (p-1) + t_w \frac{N}{p}$$

Transpose algorithm

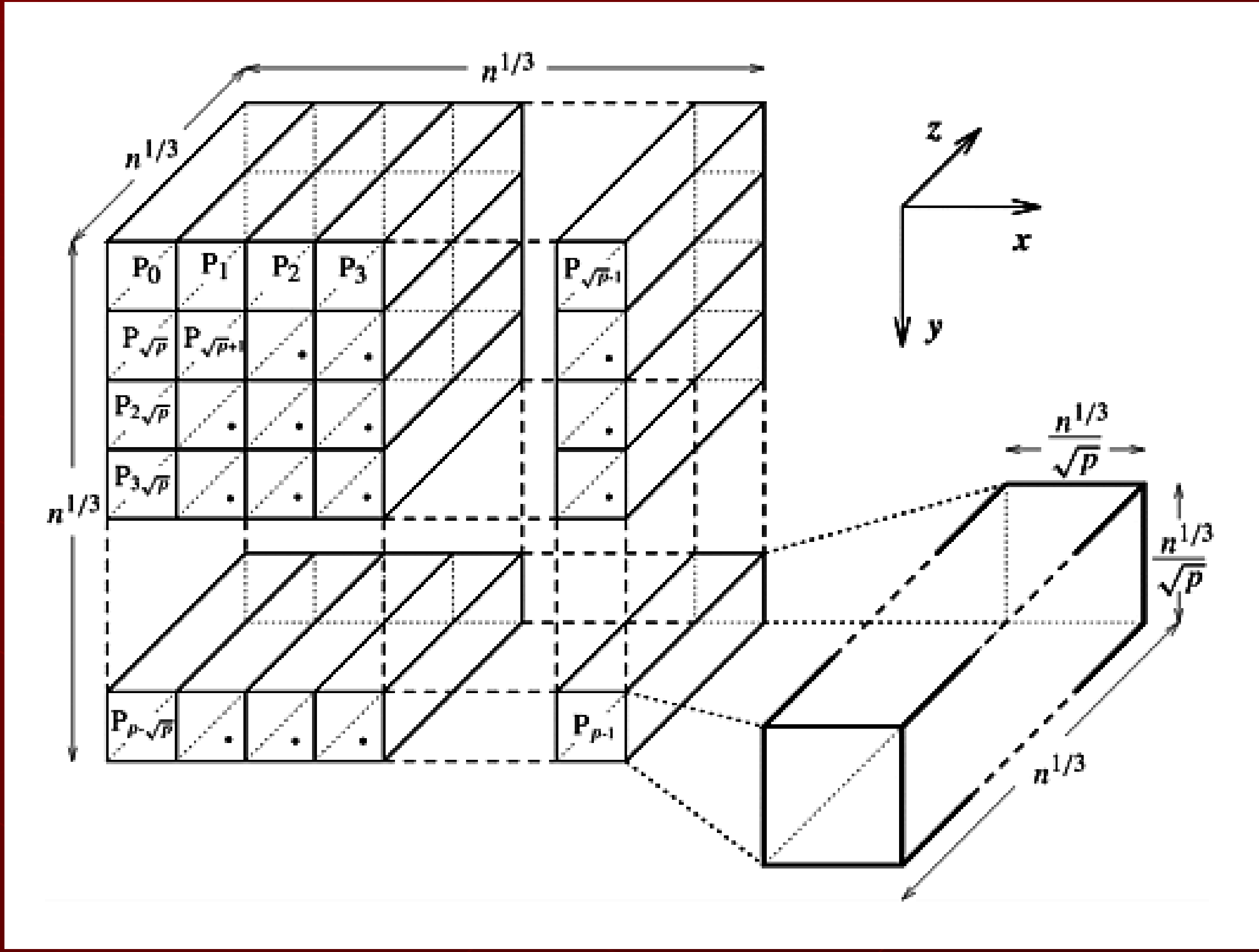
- Three-dimensional transpose algorithm

sequence of size *N* $\rightarrow \sqrt[3]{N} \times \sqrt[3]{N} \times \sqrt[3]{N}$ array

- Mapping

$\sqrt[3]{N} \times \sqrt[3]{N} \times \sqrt[3]{N}$ array $\leftrightarrow$ $\sqrt{p} \times \sqrt{p}$ mesh of processes

Transpose algorithm

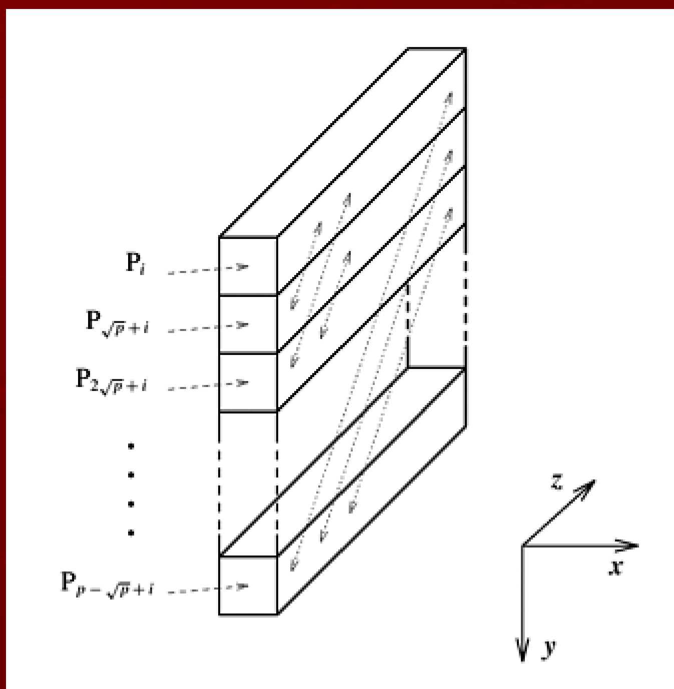


Transpose algorithm

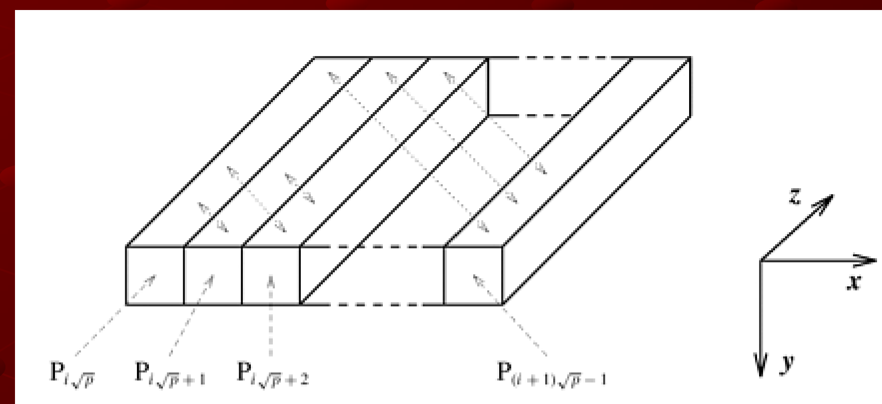
1. A $\sqrt[3]{N}$-point FFT along the z-axis
2. Each of the $\sqrt[3]{N} \times \sqrt[3]{N}$ cross-sections along the y-z plane is transposed
3. A $\sqrt[3]{N}$-point FFT along the z-axis
4. Each of the $\sqrt[3]{N} \times \sqrt[3]{N}$ cross-sections along the x-z plane is transposed
5. A $\sqrt[3]{N}$-point FFT along the z-axis

Transpose algorithm

- The transposition phases in the tree-dimensional transpose algorithm



(a) Transpose in the i^{th} column of processors during phase 2



(b) Transpose in the i^{th} row of processors during phase 4

t_c

Transpose algorithm

- The parallel run time

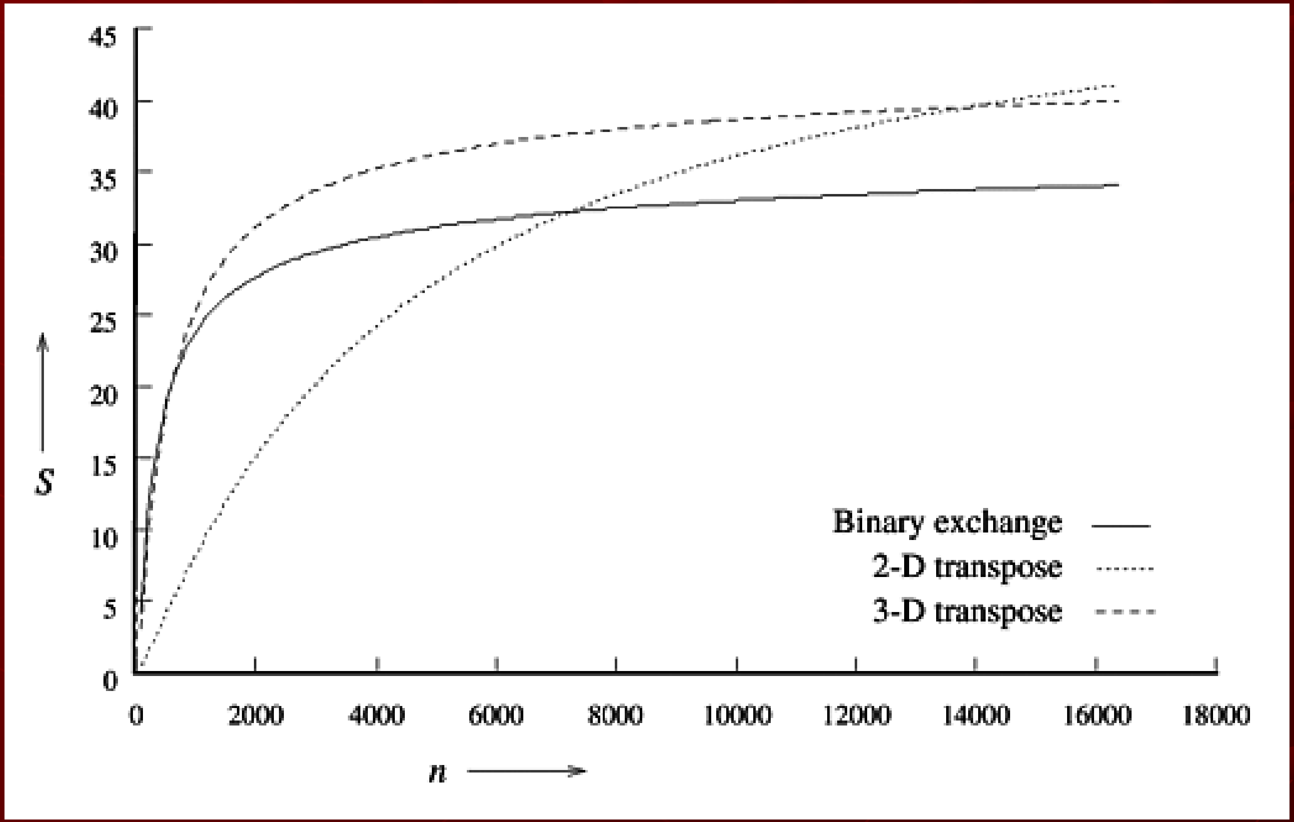
$$T_p = t_c \frac{N}{p} \log_2 N + 2t_s(\sqrt{p} - 1) + 2t_w \frac{N}{p}$$

- Speedup

$$S \approx \frac{pN \log_2 N}{N \log_2 N + 2(t_s / t_c) p(\sqrt{p} - 1) + 2(t_w / t_c) N}$$

Comparison of described algorithms

- Assume that $t_c = 2, t_w = 4, t_s = 25, p = 64$



Conclusion

- Binary-exchange algorithm
  - high communication bandwidth
  - shared memory (Open MP)

- Transpose algorithm
  - limited communication bandwidth
  - distributed memory (MPI)

Parallel FFT software

- **Parallel Engineering and Scientific Subroutine Library (PESSL)**
- **"Fastest Fourier Transform in the West." (FFTW)**
- **Intel® Math Kernel Library (Intel® MKL)**

Acknowledgments

Sergei Andreevitch Nemnyugin

Sergei Yurievitch Slavyanov

Roman Kuralev

Thank you for attention!