Parallel FFT

Sathish Vadhiyar

Sequential FFT – Quick Review

$$Y[i] = \sum_{k=0}^{n-1} X[k]\omega^{ki}, 0 \le i < n$$

- $\omega = e^{2\pi\sqrt{-1}/n}$

  Twiddle factor – primitive n^th root of unity in complex plane -

$$Y[i] = \sum_{k=0}^{(n/2)-1} X[2k]\omega^{2ki} + \sum_{k=0}^{(n/2)-1} X[2k+1]\omega^{(2k+1)i}$$

$$= \sum_{k=0}^{(n/2)-1} X[2k]e^{2(2\pi\sqrt{-1}/n)ki} + \sum_{k=0}^{(n/2)-1} X[2k+1]\omega^{i} e^{2(2\pi\sqrt{-1}/n)ki}$$

$$= \sum_{k=0}^{(n/2)-1} X[2k]e^{2\pi\sqrt{-1}ki/(n/2)} + \omega^{i} \sum_{k=0}^{(n/2)-1} X[2k+1]e^{2\pi\sqrt{-1}ki/(n/2)}$$

Sequential FFT – Quick Review

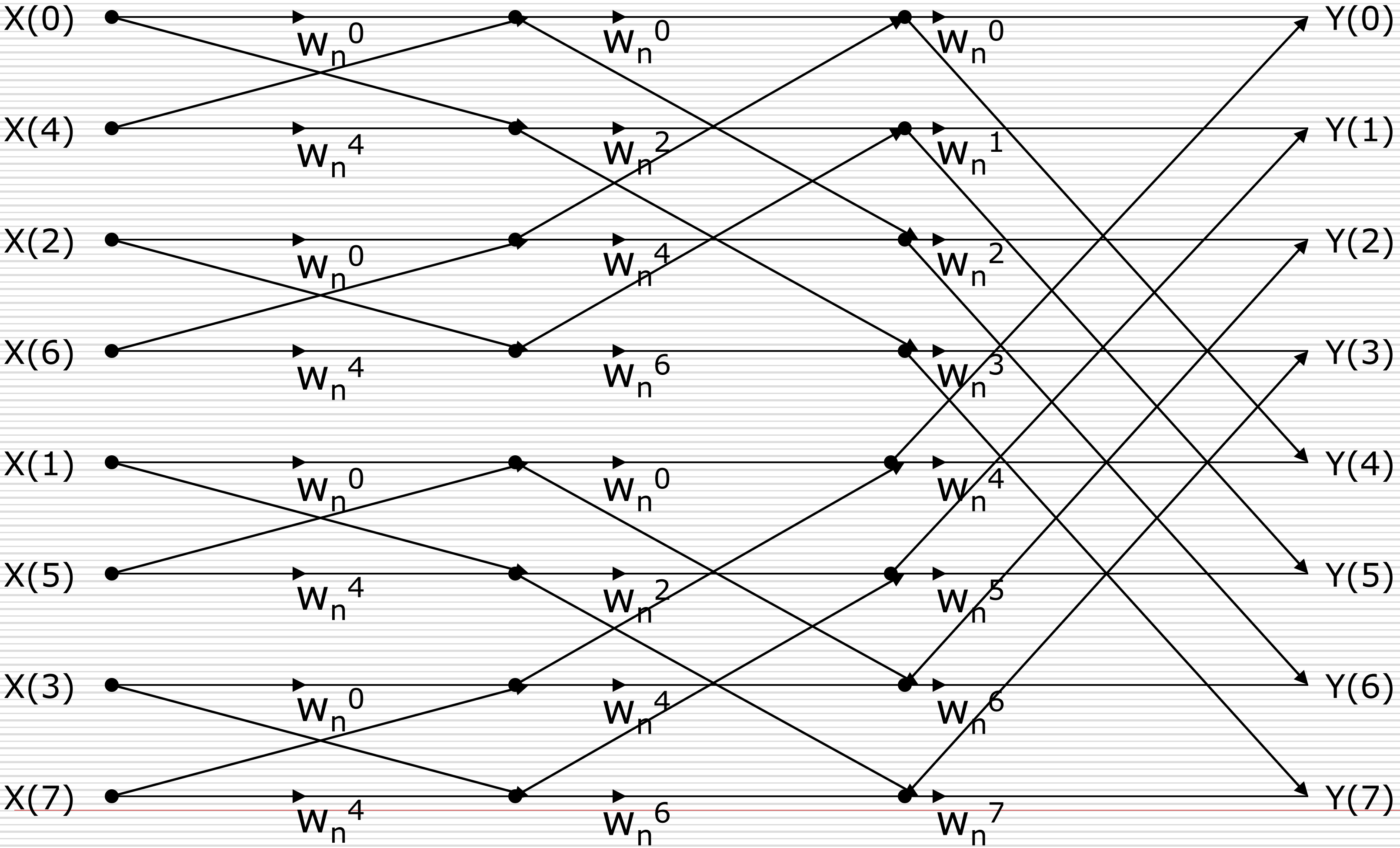
- (n/2)th root of unity

$$\tilde{\omega} = e^{2\pi\sqrt{-1}/(n/2)} = \omega^2$$

$$Y[i] = \sum_{k=0}^{(n/2)-1} X[2k]\tilde{\omega}^{ki} + \omega^i \sum_{k=0}^{(n/2)-1} X[2k+1]\tilde{\omega}^{ki}$$

- 2 (n/2)-point DFTs

Sequential FFT – quick review



Sequential FFT – recursive solution

```
procedure R_FFT(X, Y, n, w)
if (n=1) then Y[0] := X[0] else
begin
  R_FFT(<X(0), X(2), …, X[n-2]>,
            <Q[0], Q[1], …, Q[n/2]>, n/2, w^2)
  R_FFT(<X(1), X(3), …, X[n-1]>,
            <T[0], T[1], …, T[n/2]>, n/2, w^2)
  for i := 0 to n-1 do
     Y[i] := Q[i mod (n/2)] + w^i T(i mod (n/2)];
end R_FFT
```

Sequential FFT – iterative solution

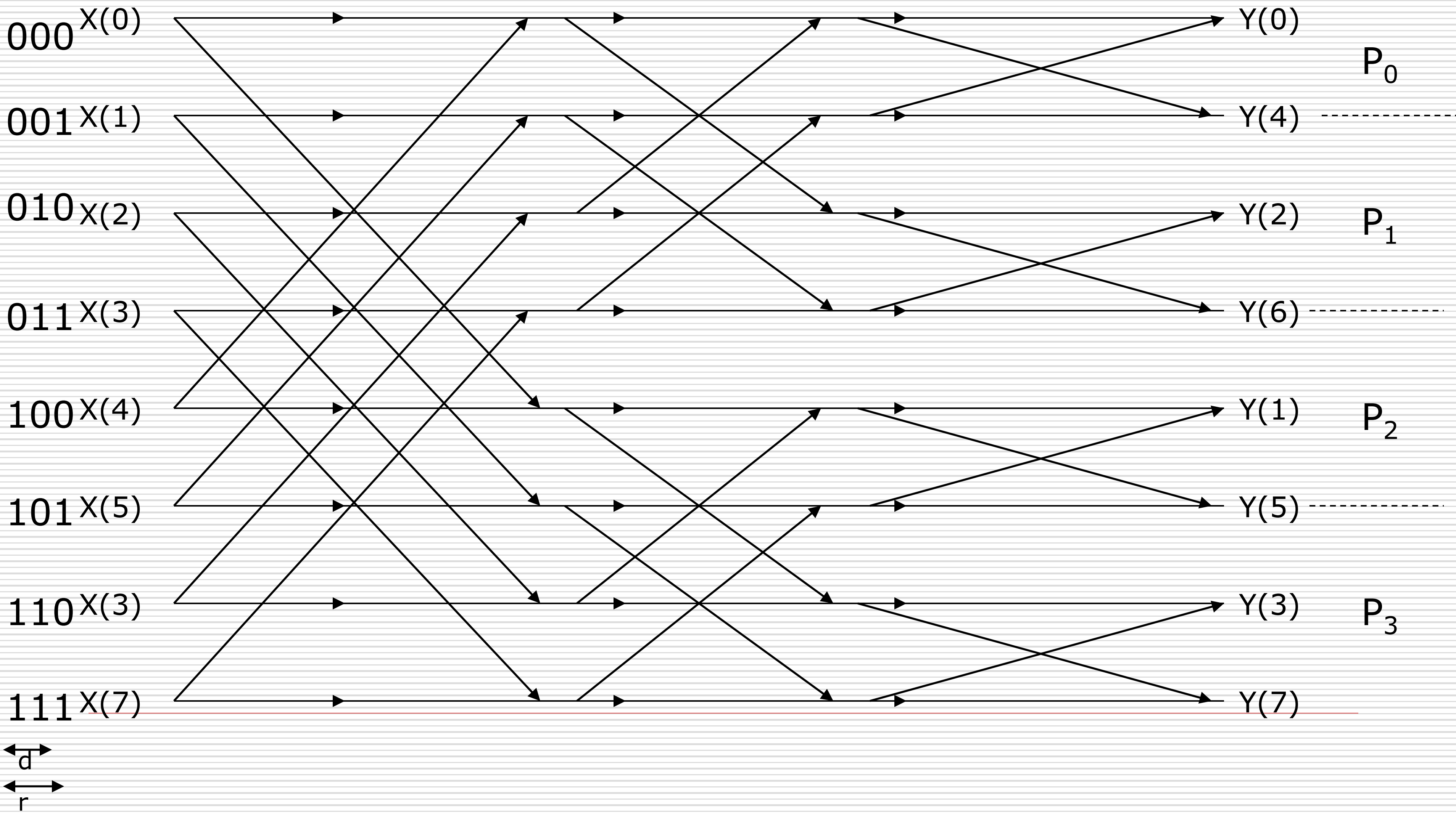
```
procedure ITERATIVE_FFT(X, Y, n)
begin
  r := log n;
  for i:= 0 to n-1 do R[i] := X[i];
  for m:= 0 to r-1 do
    begin
      for i:= 0 to n-1 do S[i] := R[i];
      for i:= 0 to n-1 do
        begin
     /* Let (b0, b1, b2, … br-1) be the binary representation of i */
          j := (b_0 … b_m-1 0 b_m+1 .. b_r-1);
          k := (b_0 … b_m-1 1 b_m+1 .. b_r-1);
          R[i] := S[j] + S[k] x w^(b_m b_m-1 ..b_0 0..0) ;
      endfor;
   endfor;
   for i:= 0 to n-1 do Y[i] := R[i];
 end ITERATIVE_FFT
```

Example of w calculation

m/ i	0	1	2	3	4	5	6	7
0	000	000	000	000	100	100	100	100
1	000	000	100	100	010	010	110	110
2	000	100	010	110	001	101	011	111

For a given m and i, the power of w is computed by reversing the order of the m+1 most significant bits of i and padding them by 0's to the right.

Parallel FFT – Binary exchange

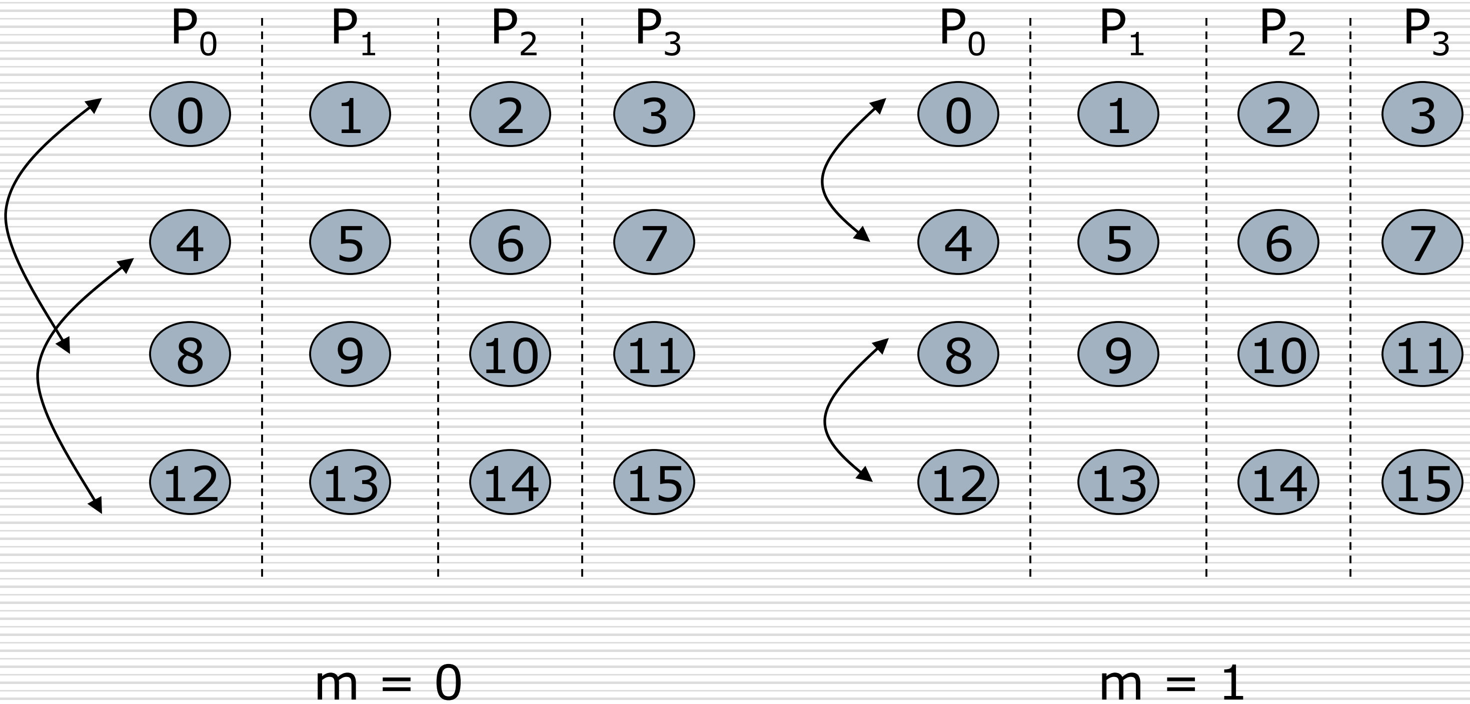


Binary Exchange

- d – number of bits for representing processes; r – number of bits representing the elements
- The d most significant bits of element i indicate the process that the element belongs to.
- Only the first d of the r iterations require communication
- In a given iteration, m, a process i communicates with only one other process obtained by flipping the (m+1)th MSB of i
- Total execution time - ?

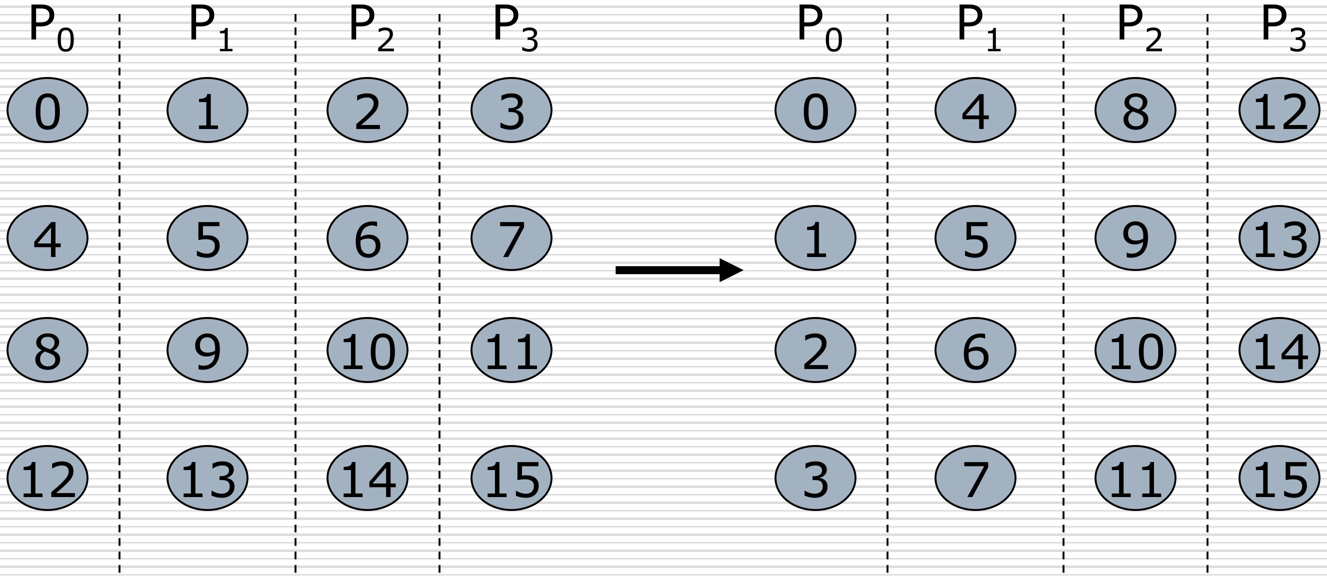
  (n/P)logN + logP(l) + (n/P)logP (b)

Parallel FFT – 2D Transpose



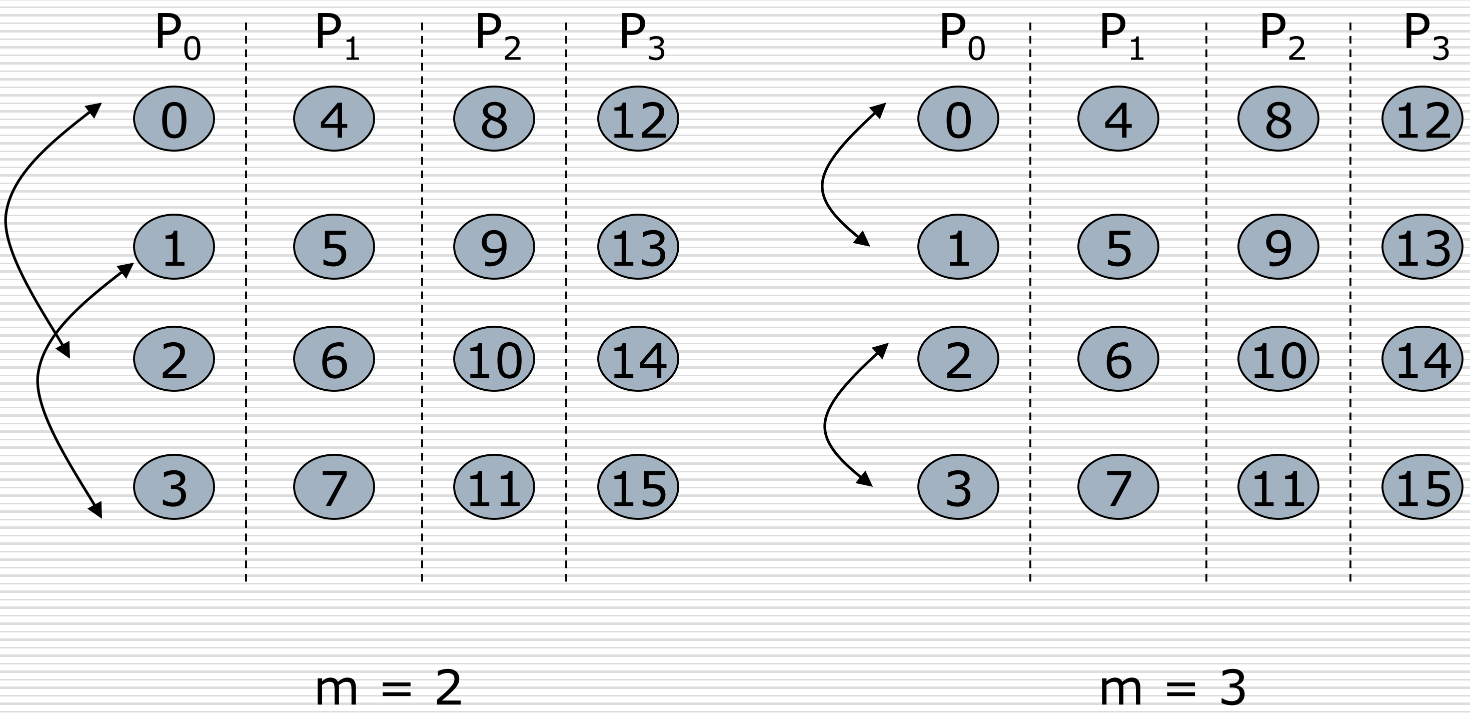
Phase 1 – FFTs along columns

Parallel FFT – 2D Transpose



Phase 2 – Transpose

Parallel FFT – 2D Transpose



Phase 3 – FFTs along columns

2D Transpose

- In general, n elements arranged as $\sqrt{n} \times \sqrt{n}$
- p processes arranged along columns. Each process owns $\sqrt{n}/p$ columns
- Each process does $\sqrt{n}/p$ FFTs of size $\sqrt{n}$ each
- Parallel runtime – $2(\sqrt{n}/p)\sqrt{n}\log\sqrt{n} + (p-1)(l) + n/p(b)$

3D Transpose

- $n^{1/3} \times n^{1/3} \times n^{1/3}$ elements
- $\sqrt{p} \times \sqrt{p}$ processes
- Steps ?
- Parallel runtime –

$$(n/p)\log n(c) + 2(\sqrt{p}-1)(l) + 2(n/p)(b)$$

In general

- For q dimensions:
- Parallel runtime –

$(n/p)\log n + (q-1)(p^{1/(q-1)} - 1)$ [l]+

$(q-1)(n/p)$ [b]

- Time due to latency decreases; due to bandwidth increases
- For implementation – only 2D and 3D transposes are feasible. Moreover, there are restrictions on n and p in terms of q.

Choice of algorithm

- Binary exchange – small latency, large bandwidth
- 2D transpose – large latency, small bandwidth
- Other transposes lie between binary exchange and 2D transpose
- For a given parallel computer, based on l and b, different algorithms can give different performances for different problem sizes