

# Fast Fourier Transform

Matt McCarthy

Christopher Newport University

CPSC 621

November 30, 2015

# Discrete Fourier Transform

## Definition 1 (Discrete Fourier Transform)

Let  $\mathbf{X} = (x_0, x_1, \dots, x_{n-1}) \in \mathbb{C}^n$ . Then the *Discrete Fourier Transform* of  $\mathbf{X}$  is defined as  $\mathbf{Y} = (y_0, y_1, \dots, y_{n-1})$  where

$$y_j := \sum_{k=0}^{n-1} x_k \omega^{jk}$$

with  $\omega = e^{2\pi i/n}$ . Furthermore, we denote  $\mathbf{Y} = \mathcal{F}(\mathbf{X})$ .

Complexity:  $\Theta(n^2)$ .

# Slightly Faster Fourier Transform

If we assume  $n$  is even, then by symbol pushing we get

$$\begin{aligned}y_j &= \sum_{k=0}^{n/2-1} x_{2k} \omega^{(2k)j} + \sum_{k=0}^{n/2-1} x_{2k+1} \omega^{(2k+1)j} \\&= \sum_{k=0}^{n/2-1} x_{2k} e^{2(2\pi i/n)jk} + \sum_{k=0}^{n/2-1} x_{2k+1} \omega^j e^{2(2\pi i/n)jk} \\&= \sum_{k=0}^{n/2-1} x_{2k} \left(e^{2\pi i/n}\right)^{2jk} + \omega^j \sum_{k=0}^{n/2-1} x_{2k+1} \left(e^{2\pi i/n}\right)^{2jk}\end{aligned}$$

Let  $\tilde{\omega} = \omega^2$  and we have

$$y_j = \sum_{k=0}^{n/2-1} x_{2k} \tilde{\omega}^{jk} + \omega^j \sum_{k=0}^{n/2-1} x_{2k+1} \tilde{\omega}^{jk}.$$

# Fast Fourier Transform

If  $n = 2^k$  for some  $k \in \mathbb{Z}^+$ , then we can iterate this process using the following algorithm.

The one-dimensional, unordered, radix 2, FFT algorithm.

```
1: function R-FFT(X,Y, $n,\omega$ )
2:   if  $n=1$  then
3:      $y_0 = x_0$ 
4:   else
5:     Let  $\mathbf{Q} = \mathbf{0}, \mathbf{T} = \mathbf{0} \in \mathbb{C}^n$ 
6:     R-FFT( $(x_0, x_2, \dots, x_{n-2}), (q_0, q_2, \dots, q_{n-2}), n/2, \omega^2$ )
7:     R-FFT( $(x_1, x_3, \dots, x_{n-1}), (t_1, t_3, \dots, t_{n-1}), n/2, \omega^2$ )
8:     for all  $j \in \{0, 1, \dots, n-1\}$  do
9:        $y_j = q_{j \bmod n/2} + \omega^j t_{j \bmod n/2}$ 
10:    end for
11:  end if
12: end function
```

# Fast Fourier Transform: Serial Analysis

- Since  $n = 2^k$ , we do  $\lg n = k$  steps
- At the  $m$ th level of recursion we do  $2^m$  FFTs of size  $n/2^m$ 
  - Each level is  $\Theta(n)$
- Thus, FFT is  $\Theta(n \lg n)$ .

# Iterative Formulation

```
1: function I-FFT(X,Y, $n$ )
2:    $t := \lg n$ 
3:   R = X
4:   for  $m = 0$  to  $t - 1$  do
5:     S = R
6:     for  $l = 0$  to  $n - 1$  do
7:       Let  $(b_0 b_1 \dots b_{t-1})$  be the binary expansion of  $l$ 
8:        $j := (b_0 \dots b_{m-1} 0 b_{m+1} \dots b_{t-1})$ 
9:        $k := (b_0 \dots b_{m-1} 1 b_{m+1} \dots b_{t-1})$ 
10:       $r_i := s_j + s_k \omega^{(b_m b_{m-1} \dots b_0 0 \dots 0)}$ 
11:    end for
12:  end for
13:  Y := R
14: end function
```

# Binary-Exchange Algorithm: Pseudocode

```
1: function I-FFT(X,Y, $n$ )
2:    $t := \lg n$ 
3:   R = X
4:   spawn process for  $l = 0$  to  $n - 1$  do
5:     for  $m = 0$  to  $t - 1$  do
6:       Let  $(b_0 b_1 \dots b_{t-1})$  be the binary expansion of  $l$ 
7:        $j := (b_0 \dots b_{m-1} 0 b_{m+1} \dots b_{t-1})$ 
8:        $k := (b_0 \dots b_{m-1} 1 b_{m+1} \dots b_{t-1})$ 
9:        $(s_j, s_k) \leftarrow \text{REQUEST}(j, k)$ 
10:       $r_i := s_j + s_k \omega^{(b_m b_{m-1} \dots b_0 0 \dots 0)}$ 
11:     end for
12:   end spawn
13:   join all
14:   Y := R
15: end function
```

# Binary-Exchange Algorithm: Assumptions

- We need a bisection width of  $\Theta(p)$  for  $p$  processors.
- We have  $p$  processors on a  $\lg p$  dimensional hypercube.
  - Analysis applicable to any network with  $O(p)$  bandwidth.



# Binary-Exchange Algorithm: One Task per Process

- Output based decomposition
  - Create  $n$  tasks, task  $l$  generates  $y_l$
  - Load  $x_l$  into task  $l$
  - Map each task to a unique process ( $n = p$ )
- Each process executes lines 7 to 10 of the iterative formulation
  - Each process does this  $\lg n$  times
- To execute line 10, each process needs an element of  $\mathbf{S}$  that differs from  $l$  only by one bit.
- At iteration  $m$ , each process communicates with the process whose label differs from it at  $m$ th bit.
- Each iteration has one addition, multiplication, and exchange
- Ergo  $T_p = \Theta(\lg n)$  and  $C = pT_p = \Theta(n \lg n)$

# Binary-Exchange Algorithm: Multiple Tasks per Process

```
1: function I-FFT(X,Y, $n$ )
2:    $t := \lg n$ ,  $BLK := n/p$ 
3:   R = X
4:   spawn process for  $l = 0$  to  $BLK - 1$  do
5:     for  $c = l \cdot BLK$ , to  $l \cdot (BLK + 1)$  do
6:       for  $m = 0$  to  $t - 1$  do
7:         Let  $(b_0 b_1 \dots b_{t-1})$  be the binary expansion of  $c$ 
8:          $j := (b_0 \dots b_{m-1} 0 b_{m+1} \dots b_{t-1})$ 
9:          $k := (b_0 \dots b_{m-1} 1 b_{m+1} \dots b_{t-1})$ 
10:         $(s_j, s_k) \leftarrow \text{REQUEST}(j, k)$ 
11:         $r_i := s_j + s_k \omega^{(b_m b_{m-1} \dots b_0 0 \dots 0)}$ 
12:      end for
13:    end for
14:  end spawn
15:  join all
16:  Y := R
17: end function
```

# Binary-Exchange Algorithm: Example

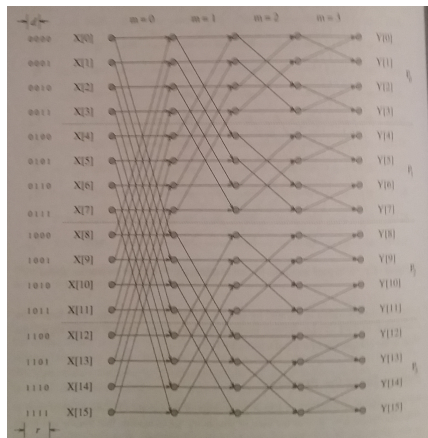


Figure: An example of FFT with  $n = 16$  and  $p = 4$

# Binary-Exchange Algorithm: Analysis 2

- Assume both  $n = 2^t$  and  $p = 2^d$
- Same procedure as previous analysis
- However, interprocessor communication happens on first  $d$  steps, and not at all on the remaining steps
- Communication cost:

$$t_s \lg p + t_w \frac{n}{p} \lg p$$

- Computation cost:

$$t_c \frac{n}{p} \lg n$$

# Binary-Exchange Algorithm: Analysis 2

- Total runtime:

$$T_p = t_c \frac{n}{p} \lg n + t_s \lg p + t_w \frac{n}{p} \lg p$$

- Cost:

$$C = pT_p = t_c n \lg n + t_s \lg p + t_w n \lg p$$

- Cost optimal for all  $p \leq n$  ( $C = O(n \lg n)$ )
- Speedup:

$$S = \frac{pn \lg n}{n \lg n + (t_s/t_c)p \lg p + (t_w/t_c)n \lg p}$$

- Efficiency:

$$E = \left( 1 + \frac{t_s p \lg p}{t_c n \lg n} + \frac{t_w \lg p}{t_c \lg n} \right)^{-1}$$

# Binary-Exchange Algorithm: Mesh Analysis

- Computation cost remains the same.

$$t_c \frac{n}{p} \lg n$$

- Row/Column Communication cost:

$$\sum_{m=0}^{d/2-1} t_s + t_w \frac{n}{p} 2^m$$

- Total runtime:

$$T_p \approx t_c \frac{n}{p} \lg n + t_s \lg p + 2t_w \frac{n}{\sqrt{p}}$$

# Binary-Exchange Algorithm: Mesh Analysis

- Speedup

$$S \approx \frac{pn \lg n}{n \lg n + (t_s/t_c)p \lg p + 2(t_w/t_c)n\sqrt{p}}$$

- Efficiency

$$E \approx \left(1 + \frac{t_s p \lg p}{t_c n \lg n} + \frac{2t_w \sqrt{p}}{t_c \lg n}\right)^{-1}$$

- Cost

$$C = pT_p \approx t_c n \lg n + t_s p \lg p + 2t_w n \sqrt{p}$$

- Cost optimal iff  $\sqrt{p} = O(\lg n)$ .

# Binary-Exchange Algorithm: Duplicated Work

```
1: function I-FFT(X,Y, $n$ )
2:    $t := \lg n$ ,  $BLK := n/p$ 
3:   R = X
4:   spawn process for  $l = 0$  to  $BLK - 1$  do
5:     for  $c = l \cdot BLK$ , to  $l \cdot (BLK + 1)$  do
6:       for  $m = 0$  to  $t - 1$  do
7:         Let  $(b_0 b_1 \dots b_{t-1})$  be the binary expansion of  $c$ 
8:          $j := (b_0 \dots b_{m-1} 0 b_{m+1} \dots b_{t-1})$ 
9:          $k := (b_0 \dots b_{m-1} 1 b_{m+1} \dots b_{t-1})$ 
10:         $(s_j, s_k) \leftarrow \text{REQUEST}(j, k)$ 
11:         $r_i := s_j + s_k \omega^{(b_m b_{m-1} \dots b_0 0 \dots 0)}$ 
12:       end for
13:     end for
14:   end spawn
15:   join all
16:   Y := R
17: end function
```



- Two dimensional transpose algorithm
  - Uses matrix transposition to do FFT
  - Total exchange
  - Works for low-bandwidth situations
  - Cost optimal iff  $n \lg n = \Omega(p^2 \lg p)$
  - Preferred over binary-exchange if  $t_s$  is small



Grama et. al.  
Introduction to Parallel Computing.  
Pearson Education.