Fast Fourier Transform

Matt McCarthy

Christopher Newport University

CPSC 621
November 30, 2015

Discrete Fourier Transform

Definition 1 (Discrete Fourier Transform)

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

function $\mathrm{R\text{-}FFT}(\mathbf{X},\mathbf{Y},n,\omega)$
    if $n=1$ **then**
        $y_0 = x_0$
    else
        Let $\mathbf{Q} = \mathbf{0}, \mathbf{T} = \mathbf{0} \in \mathbb{C}^n$
        $\mathrm{R\text{-}FFT}((x_0, x_2, \ldots, x_{n-2}),(q_0, q_2, \ldots, q_{n-2}),n/2,\omega^2)$
        $\mathrm{R\text{-}FFT}((x_1, x_3, \ldots, x_{n-1}),(t_1, t_3, \ldots, t_{n-1}),n/2,\omega^2)$
        for all $j \in \{0, 1, \ldots, n-1\}$ **do**
            $y_j = q_{j \bmod n/2} + \omega^i t_{j \bmod n/2}$
        end for
    end if
end function

Fast Fourier Transform: Serial Analysis

- Since $n = 2^k$, we do $\lg n = k$ steps
- At the mth 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

function I-FFT($\mathbf{X}$,$\mathbf{Y}$,n)
  $t := \lg n$
  $\mathbf{R} = \mathbf{X}$
  for $m = 0$ to $t-1$ **do**
    $\mathbf{S} = \mathbf{R}$
    for $l = 0$ to $n-1$ **do**
      Let $(b_0 b_1 \ldots b_{t-1})$ be the binary expansion of l
      $j := (b_0 \ldots b_{m-1} 0 b_{m+1} \ldots b_{t-1})$
      $k := (b_0 \ldots b_{m-1} 1 b_{m+1} \ldots b_{t-1})$
      $r_i := s_j + s_k \omega^{(b_m b_{m-1} \ldots b_0 0 \ldots 0)}$
    end for
  end for
  $\mathbf{Y} := \mathbf{R}$
end function

Binary-Exchange Algorithm: Pseudocode

function $\textsc{I-FFT}(\mathbf{X},\mathbf{Y},n)$
    $t := \lg n$
    $\mathbf{R} = \mathbf{X}$
    spawn process for $l = 0$ to $n-1$ **do**
        for $m = 0$ to $t-1$ **do**
            Let $(b_0 b_1 \ldots b_{t-1})$ be the binary expansion of l
            $j := (b_0 \ldots b_{m-1} 0 b_{m+1} \ldots b_{t-1})$
            $k := (b_0 \ldots b_{m-1} 1 b_{m+1} \ldots b_{t-1})$
            $(s_j, s_k) \leftarrow \textsc{Request}(j,k)$
            $r_i := s_j + s_k \omega^{(b_m b_{m-1} \ldots b_0 0 \ldots 0)}$
        end for
    end spawn
    join all
    $\mathbf{Y} := \mathbf{R}$
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 **S** that differs from l only by one bit.
- At iteration m, each process communicates with the process whose label differs from it at mth 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

function I-FFT($\mathbf{X}$,$\mathbf{Y}$,n)
  $t := \lg n$, $BLK := n/p$
  $\mathbf{R} = \mathbf{X}$
  spawn process for $l = 0$ to $BLK - 1$ **do**
    for $c = l \cdot BLK$, to $l \cdot (BLK + 1)$ **do**
      for $m = 0$ to $t - 1$ **do**
        Let $(b_0 b_1 \ldots b_{t-1})$ be the binary expansion of c
        $j := (b_0 \ldots b_{m-1} 0 b_{m+1} \ldots b_{t-1})$
        $k := (b_0 \ldots b_{m-1} 1 b_{m+1} \ldots b_{t-1})$
        $(s_j, s_k) \leftarrow \text{REQUEST}(j,k)$
        $r_i := s_j + s_k \omega^{(b_m b_{m-1} \ldots b_0 0 \ldots 0)}$
      end for
    end for
  end spawn
  join all
  $\mathbf{Y} := \mathbf{R}$
end function

Binary-Exchange Algorithm: Analysis 2

- Assume both $n = 2^t$ and $p = 2^d$
- Same procedure as previous analysis
- However, interprocessor communication happpens 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:

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

Author
Title
where

Thank you!