Parallel FFT-algorithms

Shmeleva Yulia
Saint-Petersburg State University
Department of Computational Physics
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

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

• (Inverse) Fourier Transform

\[ h(t) = \int_{-\infty}^{+\infty} H(f)e^{-2\pi ift} df, \text{ where } i = \sqrt{-1} \]
Discrete Fourier transform

- Finite time series, sampled at an interval $\Delta$

\[ h = \langle h[0], h[1], ..., h[N-1] \rangle \]

where

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

\[ k = 0, 1, ..., N - 1 \]
Discrete Fourier transform

- The discrete (forward) Fourier transform

\[ H = \langle H[0], H[1], \ldots, H[N - 1] \rangle \]

where

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

\[ j = 0, 1, \ldots, N - 1 \]
Discrete Fourier transform

- Let

\[ W_N = e^{ \frac{2\pi i}{N} } \]

- 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

⇓

The entire sequence $H$ needs an order of $N^2$ operations!
DFT property of symmetry

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

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

then

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

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

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

\[ a_j \quad b_j \quad W_N^j \quad a_j + W_N^j b_j \quad a_j - W_N^j b_j \]
Serial FFT (decomposition)
Serial FFT (Cooley and Tukey, 1965)

$m = 1$

$h[0] \rightarrow W_N^0$

$m = 2$

$h[4] \rightarrow W_N^4$

$m = 3$

$h[2] \rightarrow W_N^0$

$h[6] \rightarrow W_N^4$

$h[1] \rightarrow W_N^0$

$h[5] \rightarrow W_N^4$

$h[3] \rightarrow W_N^0$

$h[7] \rightarrow W_N^4$

$H[0]$

$H[1]$

$H[2]$

$H[3]$

$H[4]$

$H[5]$

$H[6]$

$H[7]$

Jass 2009, Shmeleva Yulia, SPbSU
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**

<table>
<thead>
<tr>
<th>Original sequence</th>
<th>Rearranged sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>binary</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
</tbody>
</table>
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

\[ m = 1 \]

\[
\begin{array}{cccc}
000 & h[0] & \bullet & P_0 \\
001 & h[1] & \bullet & P_1 \\
010 & h[2] & \bullet & P_2 \\
011 & h[3] & \bullet & P_3 \\
100 & h[4] & \bullet & P_4 \\
101 & h[5] & \bullet & P_5 \\
110 & h[6] & \bullet & P_6 \\
111 & h[7] & \bullet & P_7 \\
\end{array}
\]
# Binary-Exchange algorithm

$m = 1$

<table>
<thead>
<tr>
<th>000</th>
<th>h[0]</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>h[1]</td>
<td>001</td>
</tr>
<tr>
<td>010</td>
<td>h[2]</td>
<td>010</td>
</tr>
<tr>
<td>011</td>
<td>h[3]</td>
<td>011</td>
</tr>
<tr>
<td>100</td>
<td>h[4]</td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>h[5]</td>
<td>101</td>
</tr>
<tr>
<td>110</td>
<td>h[6]</td>
<td>110</td>
</tr>
<tr>
<td>111</td>
<td>h[7]</td>
<td>111</td>
</tr>
</tbody>
</table>

$P_0$

$P_1$

$P_2$

$P_3$

$P_4$

$P_5$

$P_6$

$P_7$
Binary-Exchange algorithm

\[ m = 1 \]

<table>
<thead>
<tr>
<th>000</th>
<th>h[0]</th>
<th>P_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>h[1]</td>
<td>P_1</td>
</tr>
<tr>
<td>010</td>
<td>h[2]</td>
<td>P_2</td>
</tr>
<tr>
<td>011</td>
<td>h[3]</td>
<td>P_3</td>
</tr>
<tr>
<td>100</td>
<td>h[4]</td>
<td>P_4</td>
</tr>
<tr>
<td>101</td>
<td>h[5]</td>
<td>P_5</td>
</tr>
<tr>
<td>110</td>
<td>h[6]</td>
<td>P_6</td>
</tr>
<tr>
<td>111</td>
<td>h[7]</td>
<td>P_7</td>
</tr>
</tbody>
</table>
Binary-Exchange algorithm

\[ m = 1 \]

\begin{align*}
000 & \quad h[0] \\
001 & \quad h[1] \\
010 & \quad h[2] \\
011 & \quad h[3] \\
100 & \quad h[4] \\
101 & \quad h[5] \\
110 & \quad h[6] \\
111 & \quad h[7]
\end{align*}

\begin{align*}
P_0 \\
P_1 \\
P_2 \\
P_3 \\
P_4 \\
P_5 \\
P_6 \\
P_7
\end{align*}

Jass 2009, Shmeleva Yulia, SPbSU
Binary-Exchange algorithm

$m = 1$

$m = 2$

000 $h[0]$  
001 $h[1]$  
010 $h[2]$  
011 $h[3]$  
100 $h[4]$  
101 $h[5]$  
110 $h[6]$  
111 $h[7]$  

$P_0$  
$P_1$  
$P_2$  
$P_3$  
$P_4$  
$P_5$  
$P_6$  
$P_7$
**Binary-Exchange algorithm**

$m = 1$

$m = 2$

000 \(h[0]\)

001 \(h[1]\)

010 \(h[2]\)

011 \(h[3]\)

100 \(h[4]\)

101 \(h[5]\)

110 \(h[6]\)

111 \(h[7]\)

\(P_0\)

\(P_1\)

\(P_2\)

\(P_3\)

\(P_4\)

\(P_5\)

\(P_6\)

\(P_7\)

Jass 2009, Shmeleva Yulia, SPbSU
Binary-Exchange algorithm

\[ m = 1 \quad m = 2 \quad m = 3 \]

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Value</th>
<th>( h )</th>
<th>( H )</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>010</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>011</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>111</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>
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 \( N = 8, \ p = 4 \ (d = 2, \ r = 3) \)

\[
\begin{align*}
000 & \rightarrow h[0] \\
001 & \rightarrow h[1] \\
010 & \rightarrow h[2] \\
011 & \rightarrow h[3] \\
100 & \rightarrow h[4] \\
101 & \rightarrow h[5] \\
110 & \rightarrow h[6] \\
111 & \rightarrow h[7]
\end{align*}
\]

\[
\begin{align*}
m = 1 & \quad m = 2 & \quad m = 3 \\
\end{align*}
\]

\[
\begin{align*}
P_0 & \rightarrow H[0] \\
P_1 & \rightarrow H[1] \\
P_2 & \rightarrow H[2] \\
P_3 & \rightarrow H[3] \\
\end{align*}
\]
Binary-Exchange algorithm

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

\[
\begin{align*}
000 & \rightarrow h[0] \rightarrow H[0] \\
001 & \rightarrow h[1] \rightarrow H[1] \\
100 & \rightarrow h[4] \rightarrow H[4] \\
111 & \rightarrow h[7] \rightarrow H[7]
\end{align*}
\]
Binary-Exchange algorithm

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

\[
\begin{align*}
1 &= m_3 \quad & 2 &= m_2 \quad & 3 &= m_1 \\
&\quad & & &
\end{align*}
\]

\[
\begin{align*}
000 &\quad h[0] & & & H[0] \\
\end{align*}
\]

Jass 2009, Shmeleva Yulia, SPbSU
Binary-Exchange algorithm

First $d$ iterations

\[ d = \log_2 p \]

$N/p$ words of data exchange

Last $(r-d)$ iterations

\[ r = \log_2 N \]

No interprocess interaction
Binary-Exchange algorithm

- Denote:

  \( t_s \) — the startup time for the data transfer

  \( t_w \) — the per-word transfer time

  \( t_c \) — computation time
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

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

• 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 than $\Theta(p)$

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

\[ \log_2 \sqrt{p} \text{ steps} \]
- Communicating processes are in the same row

\[ \log_2 \sqrt{p} \text{ steps} \]
- Communicating processes are in the same column

Jass 2009, Shmeleva Yulia, SPbSU
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

\[ S = \frac{pN \log_2 N}{N \log N + \left(\frac{t_s}{t_c}\right)p \log_2 p + 2\left(\frac{t_w}{t_c}\right)N \sqrt{p}} \]
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**

\[
W = \max \left\{ \begin{array}{l}
p \log_2 p, \quad K \frac{t_s}{t_c} p \log_2 p, \\
2K \frac{t_w}{t_c} p^{2(Kt_w/t_c)} \sqrt{p} \end{array} \right\}, \quad K = \frac{E}{1-E}
\]

Jass 2009, Shmeleva Yulia, SPbSU
Transposte 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 $\leftrightarrow$ one process
    
  \[ p = \sqrt{N} \]

- Assume that
  \[ \sqrt{N} = 2^r, \quad p = 2^d \]
Transpose algorithm
Transpse algorithm

• Another mapping:

Several columns ↔ one process

\[ p < \sqrt{N} \]

• Assume that

\[ \sqrt{N} = 2^r, \quad p = 2^d \]

• Partition the array into blocks of \( \frac{\sqrt{N}}{p} \) rows

one block ↔ one process
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**
  
  \[ E \approx \frac{1}{1 + \left( t_s p^2 \right) / \left( t_c N \log_2 N \right) + t_w / \left( t_c \log_2 N \right)} \]

- **Scalability**
  
  \[ W = \Theta(p^2 \log_2 p) \]
Transpose algorithm

Compare two algorithms!

Binary-exchange algorithm

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

Transpose algorithm

\[ T_p = t_c \frac{N}{p} \log_2 N + t_s (p - 1) + t_w \frac{N}{p} \]
Transpose algorithm

Compare two algorithms!

Binary-exchange algorithm

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

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

- The transposition phases in the tree-dimensional transpose algorithm
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!