Polynomials and Coding

Jimmy Dobler
Definitions

• Zero Norm
  – For a polynomial \( p \), \( \|p\|_0 := \) the number of non-zero coefficients in \( p \)
  – E.g., if \( p = 1 + 3x^2 + 4x^7 \), \( \|p\|_0 = 3 \).
Definitions

• Problem: Given a maximal degree $n$, find a polynomial $p$ of degree less than or equal to $n$ such that for all polynomials $q$ of degree less than or equal to $n$, $\|p \ast q\|_0$ or $\|q\|_0$ is large.
Definitions

• Problem: Given a maximal degree $n$, find a polynomial $p$ of degree less than or equal to $n$ such that for all polynomials $q$ of degree less than or equal to $n$, $\|p \ast q\|_0$ or $\|q\|_0$ is large.

• Large?
  - Let $m = \min_q (\max(\|p \ast q\|_0, \|q\|_0))$.
  - Ideally, $m = \Theta(n)$.
  - However, $m = \Theta\left(\frac{n}{\log(n)}\right)$ or even $m = \Theta(n^{0.95})$ would be acceptable.
Simplifications

• Restrict to coefficients of 0 and 1 only, and perform all math mod 2
  – XOR

• Can encode as binary strings
  – Multiplication becomes the XOR of binary shifts of the original polynomial
Example

- Let $p = 1 + x + x^3 + x^4 + x^6 + x^7 + \cdots + x^{21} + x^{22}$
- In binary, $p = 11011011011011011011011$
- Suppose $q = 1 + x^3 = 1001$
Example

• Let $p = 1 + x + x^3 + x^4 + x^6 + x^7 + \cdots + x^{21} + x^{22}$

• In binary, $p = 11011011011011011011011$

• Suppose $q = 1 + x^3 = 1001$

• $p \ast q = 11011011011011011011011$

  $XOR$  
  $11011011011011011011011$

  $= 110000000000000000000000011$
Example

- Let \( p = 1 + x + x^3 + x^4 + x^6 + x^7 + \cdots + x^{21} + x^{22} \)
- In binary, \( p = 11011011011011011011011 \)
- Suppose \( q = 1 + x^3 = 1001 \)
- \( p \ast q = 110110110110110110110111011011011\)
  \[ XOR \quad 11011011011011011011011 \]
  \[ = 1100000000000000000000000000000011 \]
  \( \|-\|p \ast q\|_0 = 4\ldots \text{not very good} \)
Why useful?

• Coding theory
  – Equivalent to having large distance between codewords
  – Allows for easy decoding, error correction
  – Existing methods all sacrifice something
Why do in parallel?

• Too many values of $p$ to start trying randomly in serial

• Even given a promising $p$:
  – Number of possible values of $q$ is proportional to $2^n$
  – Takes too long to calculate
Implementation

- Method 1: Given $n$, check all possible $p$
- Almost 100% in parallel
- Using MPI, C/C++
Implementation

• Method 2: Given a promising p, find m for increasing values of n

• We need to calculate the shifts on each processor and spend some time merging data, but this takes very little time.
Algorithm

for (int i=1; i < chosen.size()+1; i++)
    shiftSet <<= 1;
    if(chosen[i]==1) {
        sumSet^=shiftSet;
    }
}
Algorithm

for (int i=1; i < chosen.size()+1; i++)
    shiftSet <<= 1;
    if (chosen[i] == 1) {
        sumSet ^= shiftSet;
    }

• But how do we generate all the different sets of shifts to use?
Algorithm
Technical Details

• Experiments performed on 12-core nodes using MPI
• Average runtime of 10 randomly chosen bitstrings (generated using random.org)
• For testing lengths <32, use substring of above strings
## Results (Full)

<table>
<thead>
<tr>
<th>Core Count</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>281.13</td>
<td>144.03</td>
<td>77.56</td>
<td>39.10</td>
<td>20.39</td>
<td>10.40</td>
<td>5.40</td>
<td>2.80</td>
</tr>
<tr>
<td>29</td>
<td>592.65</td>
<td>306.25</td>
<td>163.49</td>
<td>82.00</td>
<td>42.83</td>
<td>21.90</td>
<td>11.41</td>
<td>5.84</td>
</tr>
<tr>
<td>30</td>
<td>1303.51</td>
<td>668.63</td>
<td>518.77</td>
<td>356.37</td>
<td>94.54</td>
<td>48.35</td>
<td>24.95</td>
<td>12.77</td>
</tr>
<tr>
<td>31</td>
<td>2566.96</td>
<td>1322.49</td>
<td>705.58</td>
<td>354.04</td>
<td>184.72</td>
<td>94.77</td>
<td>83.62</td>
<td>25.17</td>
</tr>
<tr>
<td>32</td>
<td>5032.04</td>
<td>2594.25</td>
<td>2290.09</td>
<td>705.24</td>
<td>718.00</td>
<td>187.75</td>
<td>176.32</td>
<td>50.62</td>
</tr>
</tbody>
</table>
## Results (Full)

<table>
<thead>
<tr>
<th>Number of Cores</th>
<th>Data Size 28</th>
<th>Data Size 29</th>
<th>Data Size 30</th>
<th>Data Size 31</th>
<th>Data Size 32</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8192</td>
<td>4096</td>
<td>2048</td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td>2</td>
<td>4096</td>
<td>2048</td>
<td>1024</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>4</td>
<td>2048</td>
<td>1024</td>
<td>512</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>1024</td>
<td>512</td>
<td>256</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>16</td>
<td>512</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>64</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>256</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>512</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8192</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average Time (seconds)

- **Number of Cores**: 1, 2, 4, 8, 16, 32, 64, 128
- **Data Size**:
  - 28
  - 29
  - 30
  - 31
  - 32
Results -- Efficiency
## Results

```
10011011000000010101000001100111
```

<table>
<thead>
<tr>
<th>Data Size</th>
<th>Processor Count</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>5.85</td>
<td>3.23</td>
<td>1.84</td>
<td>1.35</td>
<td>0.91</td>
<td>0.63</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>7.61</td>
<td>4.16</td>
<td>2.41</td>
<td>1.83</td>
<td>1.25</td>
<td>0.87</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>26.97</td>
<td>14.78</td>
<td>8.27</td>
<td>6.04</td>
<td>4.04</td>
<td>2.72</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>88.81</td>
<td>47.28</td>
<td>25.95</td>
<td>18.40</td>
<td>12.05</td>
<td>7.88</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>259.03</td>
<td>136.44</td>
<td>73.98</td>
<td>50.60</td>
<td>32.36</td>
<td>20.60</td>
<td></td>
</tr>
</tbody>
</table>
Results -- 10011011000000010101000001100111

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>512</td>
<td>512</td>
</tr>
</tbody>
</table>

Data Size:
- 28
- 29
- 30
- 31
- 32

Number of Processors vs Time (Seconds) graph with data points for different data sizes.
Other Causes

• Load imbalance
Other Causes

• Load imbalance

• Highly varying times for same length input
Other Causes

• Load imbalance

• Highly varying times for same length input

• Word Size (?)
However...

<table>
<thead>
<tr>
<th>Processor Count</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>281.13</td>
<td>144.03</td>
<td>77.56</td>
<td>39.10</td>
<td>20.39</td>
<td>10.40</td>
<td>5.40</td>
<td>2.80</td>
</tr>
<tr>
<td>29</td>
<td>592.65</td>
<td>306.25</td>
<td>163.49</td>
<td>82.00</td>
<td>42.83</td>
<td>21.90</td>
<td>11.41</td>
<td>5.84</td>
</tr>
<tr>
<td>30</td>
<td>1303.51</td>
<td>668.63</td>
<td>518.77</td>
<td>356.37</td>
<td>94.54</td>
<td>48.35</td>
<td>24.95</td>
<td>12.77</td>
</tr>
<tr>
<td>31</td>
<td>2566.96</td>
<td>1322.49</td>
<td>705.58</td>
<td>354.04</td>
<td>184.72</td>
<td>94.77</td>
<td>83.62</td>
<td>25.17</td>
</tr>
<tr>
<td>32</td>
<td>5032.03</td>
<td>2594.25</td>
<td>2290.09</td>
<td>705.24</td>
<td>718.00</td>
<td>187.75</td>
<td>176.32</td>
<td>50.62</td>
</tr>
</tbody>
</table>

^what?
Future Goals

• Improve load balance without heavily increasing communication

• Optimization of math

• Implement in OpenMP
Questions/Comments?