PARALLEL EVENT DETECTION IN SENSOR DATA

Marc P. Greenbaum
CSE 633
Spring 2017
AGENDA

Event Detection with Sensor Data

Step Detection Algorithm

Parallel Implementation

Analysis
Radio Frequency Signals

RF Environment

Amplitude

Time
In-phase and Quadrature Sampling

\[ I = A \cos(\omega_{RF} t + \varphi_{RF}) \times \cos(\omega_{LO} t + \varphi_{LO}) = \frac{A}{2} \left[ \cos(\omega_{RF} t - \omega_{LO} t + \varphi_{RF} - \varphi_{LO}) + \cos(\omega_{RF} t + \omega_{LO} t + \varphi_{RF} + \varphi_{LO}) \right] \]

\[ V_i = \frac{A}{2} \left[ \cos(\varphi_{RF} - \varphi_{LO}) \right] \]

Let \( \omega_{RF} = \omega_{LO} \)

\[ Q = A \cos(\omega_{RF} t + \varphi_{RF}) \times \sin(\omega_{LO} t + \varphi_{LO}) = \frac{A}{2} \left[ \sin(\omega_{RF} t - \omega_{LO} t + \varphi_{RF} - \varphi_{LO}) + \sin(\omega_{RF} t + \omega_{LO} t + \varphi_{RF} + \varphi_{LO}) \right] \]

\[ V_q = \frac{A}{2} \left[ \sin(\varphi_{RF} - \varphi_{LO}) \right] \]

Let \( \omega_{RF} = \omega_{LO} \)

**Magnitude:** \( \sqrt{V_i^2 + V_q^2} \)

**Phase:** \( \arctan\left(\frac{V_q}{V_i}\right) \)
Digitized Environmental Data
In-phase and Quadrature Analysis
Serial Step Detection Algorithm

Input: I_Data<NUM>, Q_Data<NMU>, Threshold
Output: Start_Points<INDEX>, End_Points<INDEX>

Boolean above = false

For index[start:stop]
    amplitude_sq = I_DATA[index]^2 + Q_DATA[INDEX]^2
    if (amplitude_sq > threshold^2 && !above) {
        Start_Points.pushback(index)
        above = true
    }
    else if (amplitude_sq < threshold^2 && above) {
        Stop_Points.pushback(index)
        above = false
    }
Parallel Algorithm

Divide and Conquer

• Easy split
  – Divide data amongst N processors
• Stage 1 in Parallel
  – Determine if amplitude is above threshold
• Stage 2 in Parallel
  – Identify edges
• Combine
  – Merge vectors
  – Handle pulse overlap
Parallel Algorithm

Data Input \rightarrow Characterize \rightarrow Stage 1 Threshold Compare \rightarrow Stage 2 Pulse Detection \rightarrow Edge Stitch \rightarrow Output

Thread 1

Thread 2

Thread N

5 25 80 210 290
60 180 250 305 320
120 305 380 395
Parallel Implementation

C++ Application
- Data generation utility
- Framework for stage based signal processing
- Processing algorithms in discrete kernels

OpenMP
- omp parallel for
- omp_set_num_threads()

Large Memory Cluster
- 32 Core Machine
- 256+ GB RAM
Runtime Analysis

100 Million Samples

Time (s)

1 2 4 8 16 32

Compute Nodes

1 2 4 8 16 32

1 Billion Samples

Time (s)

1 2 4 8 16 32

Compute Nodes

Processing Rate

Sample Rate (MSPS)

1 2 4 8 16 32

Compute Nodes

100 Million Samples

1 Billion Samples
Runtime Analysis

\[ S(n) = 0.693 \log_2(n) + 1.0034 \]
Conclusions

Successes

• Linear Speed Up achieved
  – 4x speed up on 8 core machine
• Reduce user facing processing time by utilizing idle cores
• Creation of parallelizable Digital Environment Processing Framework

Room for improvement

• Optimize processing at stage level

Future Work

• Add more DSP stages to processing framework
• Implement GPGPU kernels
Questions?
References

IQ Receiver Diagram

IQ Sampling

OpenMP and Slurm Support
## Runtime Data

<table>
<thead>
<tr>
<th>Samples/Cores</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00E+07</td>
<td>2.956138</td>
<td>1.839053</td>
<td>1.245545</td>
<td>0.953884</td>
<td>0.825309</td>
<td>0.841648</td>
</tr>
<tr>
<td>1.00E+08</td>
<td>31.872928</td>
<td>19.02572</td>
<td>12.410948</td>
<td>9.170127</td>
<td>7.524059</td>
<td>7.004921</td>
</tr>
<tr>
<td>1.00E+09</td>
<td>320.359305</td>
<td>189.516166</td>
<td>122.697726</td>
<td>90.522017</td>
<td>74.30246</td>
<td>66.487053</td>
</tr>
</tbody>
</table>