LabVIEW Real Time for High Performance Control Applications

Aljosa Vrancic, Lothar Wenzel, LabVIEW R&D, National Instruments

Case studies: Matrix-vector Multiplication & PDE

Most of the basic computational algorithms for high-performance computing are designed to have the best average performance for the most general case as they are used in off-line calculations: the goal is for calculation to end in as little time as possible for an arbitrary set of inputs but each step in general does not have a strict time deadline. As a result, they do not scale well into hard real-time embedded environments that have much stricter per-iteration timing constraints, often in a 1 ms range. We found the following approach to work much better for real-time environments:

- Divide algorithm into two steps:
  1. Off-line calculation – it is acceptable for this step to be expensive, as it is done up front
  2. On-line calculation – this step uses input and off-line calculation data to compute outputs deterministically

We applied this approach to matrix-vector multiplication and PDE solver problems that are at the heart of many control applications. PDEs are used to describe the system and matrix-vector multiplication is used to apply the model to sensor data in order to generate actuator data.

### Matrix-Vector Multiplication in Real-time

**E-ELT Telescope**

- ESO (European Southern Observatory)
- Five mirrors (M1 through M5)
- National Instruments involvement:
  - Data Acquisition
  - M1 – 46m primary segmented mirror
  - M4 – adaptive mirror

**Layer 1**

- **Development time:** 2 weeks
- **Throughput/Latency for 80MB/s network setup**
  - Data size: 8MB
  - Throughput: 80MB/s
  - Latency: 2.5ms

**Layer 2**

- **Problem:**
  - For a 1000x1000 grid, M dimensions are $1,000,000 \times 1,000,000$
  - Calculating $M^{-1}$ directly may be impossible

**Solution:**

1. Cast PDE into a set of equations: $Mx - b = f(x)$
2. Use iterative algorithm: $x(n+1) = M^{-1}(f[x(n)] + b)$

### Custom FPGA-based Communication Protocol

**Throughput/Latency for 80MB/s network setup**

<table>
<thead>
<tr>
<th>Throughput</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>80MB/s</td>
<td>2.5ms</td>
</tr>
</tbody>
</table>

**FPGA Block Diagram**

- **PC**
- **FPGA**
- **DMA**
- **Matrix fits in cache**
- **Assembly code**
- **L2 Cache Threshold:** Matrix fits
- **L1 Cache Threshold:** Matrix too big

**E-ELT Telescope:**

- **M1 Mirror**
  - 984 hexagonal segments
  - 6 sensors/3 actuators per segment
  - 1 ms control cycle
  - Math tricks to reduce problem to $36 \times 36$ symmetric matrix by 16x vector multiplication

### Matrix-Vector Multiplication

**Layer 3**

- **Problem:**
  - For a 1000x1000 grid, M dimensions are $1,000,000 \times 1,000,000$
  - Calculating $M^{-1}$ directly may be impossible

**Solution:**

1. Solve the problem for the unit boundary vector using any algorithm
2. Iterate over smaller subsets of the problem for which $M^{-1}$ can be calculated directly.
3. Values for the solution corresponding to the grid point represent elements in the $M^{-1}$ row corresponding to the unit boundary vector. Repeat the process for all unit vectors.

**Minimum Grid Size**

- $11 \times 12^2$ grid (6993 non-linear equations with 6993 unknowns)
- One iteration: < 250 \( \mu \)s
- Total time for solution: <1 ms

**Layer 4**

- **Custom communication protocol**
- CPU off-load for data recombination
- Fully hand-shaked, with retries
- Sensors' reads on unit grid follow data delivery to CPU, eliminating access delay from memory