A Simple Multithreaded C++ Framework for High-Performance Data Acquisition Systems

Rolando Inglés  Piotr Perek  Mariusz Orlikowski  Andrzej Napieralski
{roling,pperek,mariuszo,napier}@dmcs.pl

MIXDES 2015
A Simple Multithreaded C++ Framework for High-Performance Data Acquisition Systems

OUTLINE

• Introduction
• Related Work
• The Simple C++ Framework

• Experimental Evaluation

• Results
• Conclusions
• Questions
Introduction

The Project

Monitoring Data Workflow

CONTROL SYSTEM

Data Archiving

Database

Off-line analysis

CRITICAL SYSTEMS

Real-Time Monitoring

Power Generator Control

Synchronization System

Data Acquisition System

Power Generator

Monitoring Camera

Data Acquisition System

EoSens® 3CL Full CL MC3010 high-speed camera

Monitoring Data Production Rate

Resolution in Pixel

- 1,696 (H) x 1,710 (V) 217 frames/s
- 1,280 (H) x 1,024 (V) 452 frames/s
- 680 (H) x 480 (V) 1,495 frames/s

Data Width

- 630 Mbyte/s
- 593 Mbyte/s
- 460 Mbyte/s

Source: EoSens® 3CL Camera Manual, Rev. 1.01, Copyright © 2010 Mikrotron GmbH.
Introduction

A Simple Multithreaded C++ Framework for High-Performance Data Acquisition Systems

Introduction

The Project

The System

Related Work

CxxDisruptor & FastFlow

The Simple C++ Framework

Experimental Evaluation

The Environment

Timing

Process

Results

Average

Latency

Frequency

Latency

Conclusions

Questions

Monitoring Data Workflow

The System

Data Acquisition System

CONTROL SYSTEM

Data Archiving

Database

Off-line analysis

CRITICAL SYSTEMS

Real-Time Monitoring

Synchronization System

Power Generator Control

Data Acquisition Control
Related Work

**CxxDisruptor**

LMAX Disruptor pattern C++ Implementation.
Source Code: http://sourceforge.net/projects/cxxdisruptor/
Author: Henrik Baastrup

High Performance Inter-Thread Messaging Library, based on Java.

**FastFlow**

C++ parallel programming framework.
Source Code: http://sourceforge.net/projects/mc-fastflow/

Parallel programming: Alpha group
Parallel Computing research cluster,
Department of Computer Science
University of Torino

**libcds**

CDS C++ library - Concurrent Data Structures
Source Code: http://sourceforge.net/projects/libcds/files/
Author: Max Khizhinsky

**NORTE**

a library of non-blocking synchronization protocols
Source: http://www.noble-library.org/index.html
Author: Håkan Sundell

**FastFlow**

C++ parallel programming framework.
Source Code: http://sourceforge.net/projects/mc-fastflow/

Parallel programming: Alpha group
Parallel Computing research cluster,
Department of Computer Science
University of Torino

**CxxDisruptor**

LMAX Disruptor pattern C++ Implementation.
Source Code: http://sourceforge.net/projects/cxxdisruptor/
Author: Henrik Baastrup

High Performance Inter-Thread Messaging Library, based on Java.

**FastFlow**

C++ parallel programming framework.
Source Code: http://sourceforge.net/projects/mc-fastflow/

Parallel programming: Alpha group
Parallel Computing research cluster,
Department of Computer Science
University of Torino

**libcds**

CDS C++ library - Concurrent Data Structures
Source Code: http://sourceforge.net/projects/libcds/files/
Author: Max Khizhinsky

**NORTE**

a library of non-blocking synchronization protocols
Source: http://www.noble-library.org/index.html
Author: Håkan Sundell
The Simple C++ Framework

PRODUCER PUBLISHING LOOP

```
do{
    next_slot = monitor.get_next_free_slot();
    event_pointer = monitor.get_slot_pointer( next_slot );
    event_pointer.fill_data( data );
    monitor.publish( next_slot );
}
while( not EOF);
```

CONSUMER GATHERING LOOP

```
eof=false;
do{
    if ( monitor->next_event( consumer_id ) )
        event_ptr = monitor->gather( consumer_id );
    else if ( monitor->eof( consumer_id ) )
        eof = true;
    else
        yield();
}while( not eof);
```
Experimental Evaluation

**The Environment**

**Red Hat® Enterprise Linux® 7 64bits**

**Intel® Core™ i5-520M Processor (3M Cache, 2.40 GHz)**

**4 GB of RAM Memory**

**Linux Kernel 3.10.0-123.el7.x86_64**

**GNU/Linux**

**g++ (GCC) 4.8.2 20140120 (Red Hat 4.8.2-16)**

**Copyright (C) 2013 Free Software Foundation, Inc.**

**Input File containing frames with 680x480 pixels resolution**

*Hint: The camera produces 1,495 fps with 680x480 pixels resolution configuration.*

**CxxDisruptor & FastFlow**

**The Simple C++ Framework**

*Hint: STL & Boost libraries have been used*
Experimental Evaluation

Timing Process

Related Work
CxxDisruptor & FastFlow
The Simple C++ Framework
Experimental Evaluation

The Environment
Timing Process

Results
Average Latency Frequency Latency

Conclusions
Questions

\[
\text{message latency} = \text{greatest}(C_1 T - PT, C_2 T - PT)
\]

NUMBER OF SLOTS IN RING-BUFFER
256 – 512 – 1024 – 2048 – 4096 – 8192
Results

Average Latency (ns)

- C++ Boost = Lowest latency
- More slots ≠ Lower latency

<table>
<thead>
<tr>
<th>#SLOTS</th>
<th>C++ boost</th>
<th>CxxDisruptor</th>
<th>FastFlow</th>
<th>C++ STL</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>956</td>
<td>2968</td>
<td>4836</td>
<td>1290</td>
</tr>
<tr>
<td>512</td>
<td>663</td>
<td>4323</td>
<td>4189</td>
<td>2093</td>
</tr>
<tr>
<td>1024</td>
<td>880</td>
<td>5602</td>
<td>4245</td>
<td>2817</td>
</tr>
<tr>
<td>2048</td>
<td>1045</td>
<td>5354</td>
<td>3702</td>
<td>2670</td>
</tr>
<tr>
<td>4096</td>
<td>1240</td>
<td>21972</td>
<td>3894</td>
<td>2786</td>
</tr>
<tr>
<td>8192</td>
<td>3137</td>
<td>29964</td>
<td>3631</td>
<td>2754</td>
</tr>
</tbody>
</table>
Results

Latency Message Frequency (ns)

- CxxDisruptor = 97%
- C++ STL = 98%
Conclusions

- By using C++ standard libraries is possible to develop a C++ framework obtaining satisfactory performance.

- Some optimization should be incorporated to reduce the amount of messages with variable latency.

- The four frameworks work with 100% of accuracy.
...appreciate your kind attention!

Questions...