Improving Packet Processing Performance of a Memory-Bounded Application

Jörn Schumacher
CERN / University of Paderborn, Germany
jorn.schumacher@cern.ch

On behalf of the ATLAS FELIX Developer Team
CMS

ALICE

LHCb

ATLAS

LHC ring: 27 km circumference
ATLAS: General-purpose particle detector, designed to observe phenomena involving high-energy particle collisions

# Channels: $100 \cdot 10^6$
Weight: 7000t
Collaborators: >3000
Raw data produced: \(~60\ \text{TB/s}\)
Data recorded to disk: \(1-2\ \text{GB/s}\) (after filtering)

Data collection, selection, processing, monitoring, etc. requires tens of thousands of distributed applications, processing in quasi-realtime

ATLAS: General-purpose particle detector, designed to observe phenomena involving high-energy particle collisions

# Channels: \(100 \cdot 10^6\)
Weight: 7000t
Collaborators: >3000
In this talk

1. Integration of a new Distributed Event-Based System in the ATLAS experiment

2. Experience in analysis and optimization of a packet-based software
ATLAS DAQ:

Detector Cavern

Service Cavern

Datacenter

Detector-specific optical links

Common optical links

Ethernet

High-Level Trigger Farm

Frontend

ReadOut Driver

ReadOut System (Data Buffer)

PCs (COTS)

Custom electronic components

40 MHz

100 kHz / ~200 GB/s

~1800 links

~100 servers

~1500 servers

Today

Run 2 2015

Run 3 2017

Run 4 2019

Run 5 2021

Run 6 2023

Run 7 2025

Event Proc.

Event Proc.

Event Proc.

Event Proc.

Event Proc.

Event Proc.
Custom Electronics

Challenging maintenance and operation:
Advantages in using COTS components (PC Technology)
Event Processing in Software

“In an event-based mode of interaction components communicate by generating and receiving event notifications [...] An event notification service [...] mediates between the components of an event-based system (EBS) and conveys notifications from producers [...] to consumers [...]”

[...] The notification service decouples the components so that producers unaware of any consumers and consumers rely only on the information published, but not on where or by whom it is published. [...] The event-based style carries the potential for easy integration of autonomous, heterogeneous components into complex systems that are easy to evolve and scale.”
“In an event-based mode of interaction components communicate by generating and receiving event notifications [...] An event notification service [...] mediates between the components of an event-based system (EBS) and conveys notifications from producers [...] to consumers [...]”

[...] The notification service decouples the components so that producers unaware of any consumers and consumers rely only on the information published, not on where or by whom it is published. The event-based style carries the potential for easy integration of autonomous, heterogeneous systems that are easy to evolve and scale.”
“In an event-based mode of interaction components communicate by generating and receiving event notifications [...]. An event notification service [...] mediates between an event-based system (EBS) and conveys notifications from producers [...] to consumers [...].

 [...] The notification service decouples the components so that producers unaware of any consumers and consumers rely only on the event-based style carries the potential for easy integration of autonomous, heterogeneous components into complex systems that are easy to evolve and scale.”
“In an event-based mode of interaction components communicate by generating and receiving event notifications [...] An event notification service [...] mediates between an event-based system (EBS) and conveys notifications from producers [...] to consumers [...]

The notification service decouples the components such that producers unaware of any consumers and consumers rely only on the information published, not on where or by whom it is published. The event-based style carries the potential for easy integration of autonomous, heterogeneous components into complex systems that are easy to evolve and scale.”
Event Processing in Software

“In an event-based mode of interaction components communicate by generating and receiving event notifications [...] An event notification service [...] mediates between the components of an event-based system (EBS) and conveys notifications from producers [...] to consumers [...]

The notification service decouples the components such that producers unaware of any consumers and consumers rely only on the published information about where or by whom the information was published. The event-based style carries the promise of easy integration of autonomous, heterogeneous systems that are easy to evolve and scale.

Still missing today?
An Event Distribution System for ATLAS

Requirements:

• Simple operation and maintenance (PC technology over custom designed electronics)

• Scalability (Switched networks over point-to-point links)

• Interface to radiation-hard detector links

• Heterogeneous workloads
ATLAS DAQ:

Detector Cavern

Service Cavern

Datacenter

~1500 servers

High-Level Trigger Farm


Ethernet

Custom electronic components

P Cs (COTS)

New Systems, Common Link Technology

FE FE FE

ROD ROD ROD

ROS ROS ROS

40 MHz

2018

Run 2 2015 2017 Run 3 Run 4 2019 2021 2023 2025

ATLAS DAQ (COTS)
ATLAS DAQ:

Detector Cavern
- New Systems, Common Link Technology
- FELIX
- HPC Network
- ROS
- Event Proc.
- High-Level Trigger Farm
- ~1500 servers
- PCs (COTS)
- Custom electronic components

Service Cavern
- 40 GbE, Infiniband

Datacenter
- 40 MHz
- 2018

Timeline:
- Run 2
- Run 3
- Run 4
ATLAS DAQ:

Detector Cavern

Service Cavern

Datacenter

Custom electronic components

PCs (COTS)

2018

Run 2 2015  Run 3 2017 Run 4 2019 2021 Run 5 2023 2025

New Systems, Common Link Technology

HPC Network

40 GbE, Infiniband

~1500 servers


High-Level Trigger Farm

Ethernet

40 MHz

FE

ROD

ROS

Readout

Custom electronic components

PCs (COTS)

2018

Run 2 2015  Run 3 2017 Run 4 2019 2021 Run 5 2023 2025
ATLAS DAQ:

Detector Cavern

Service Cavern

Datacenter

Common optical links

HPC Network

Ethernet

High-Level Trigger Farm

2023

2015 2017 2019 2021 2023 2025

ATLAS DAQ:

Detector Cavern

Service Cavern

Datacenter

Common optical links

HPC Network

Ethernet

High-Level Trigger Farm

2023

2015 2017 2019 2021 2023 2025
Automatic failover and load balancing

Scalable architecture

Routing of multiple traffic types: physics events, detector control, configuration, calibration, monitoring

Industry standard links: data processors/handlers can be SW in PCs - Less custom electronics, more COTS components

Reconfigurable data path, multi-cast, cloning, QoS

Automatic failover and load balancing
CERN GBT Link Technology
(GigaBit Transceiver)

Point-to-point link technology developed at CERN, progressively replaces detector-specific links in ATLAS

Designed for common High-Energy Physics environments (high radiation, magnetic fields, ...)

Typical raw bandwidth 4.8 Gbps or 9.6 Gbps

Supports variable-width virtual links ("elinks") for mixed traffic types

(Optional) Forward Error Correction
Development Platform

HiTech Global PCIe development
Xilinx Virtex-7
PCIe Gen-2/3 x8
24 bi-directional links
http://hitechglobal.com/Boards/PCIE-CXP.htm
With custom TTCfx FMC

SuperMicro X10DRG-Q
2x Haswell CPU, up to 10 cores
6x PCIe Gen-3 slots
64 GB DDR4 Memory

Mellanox ConnectX-3 VPI
FDR/QDR Infiniband
2x10/40 GbE
FELIX Architectural Overview

- FPGA Card
- Memory
- CPU
  - DMA
  - MSI-X
  - Custom Device Driver
  - FELIX Application
- PCIe Gen-3
- DMA, IRQ
- Config
- GBT
- Elink router
- TTC FMC
- Optical Links
- 24 – 48 links
- Large Buffers per group of elinks
- 2 – 4 40-Gb/s ports
- NIC
  - Optical Links
- 64 Gb/s
- 64 Gb/s
- Optical Links
- PCIe Gen-3
- PCIe Gen-3
FELIX Architectural Overview

- FPGA Card
  - DMA, IRQ
  - Config
  - GBT
  - Elink router
  - TTC FMC
  - Optical Links
  - 24 – 48 links

- Memory
  - Large Buffers per group of elinks
  - PCIe Gen-3

- CPU
  - DMA
  - MSI-X
  - Custom Device Driver
  - FELIX Application

- NIC
  - Optical Links
  - 2 – 4 40-Gb/s ports

- PCIe Gen-3
  - 64 Gb/s

- Optical Links
  - 64 Gb/s

Elink router
CPU Data Processing Pipeline

1. Read Blocks from File Descriptors
2. Block Decode
3. Statistics
4. Extract Meta Information
5. Route
6. Load Balancing, Error Handling
7. Send To Network
Program DMA transfers and read fixed blocks of data that have been encoded for the transfer over PCIe

Decode into variable sized chunks for transmission over network

e-link streams share the same pipeline

Load Balancing, Error Handling

Send To Network

Read Blocks from File Descriptors

Block Decode

Statistics

Extract Meta Information

Route

elink 0

elink 1

elink 2

...
Fixed Block Decoding

Stream of Blocks transmitted over PCIe:

Stream of reconstructed chunks with meta information for further processing:

Packets are analyzed, routed and transmitted to network destinations
Optimizations (single-threaded)

Isolated benchmarks of the packet processing benchmarks with in-memory input data.

Target: 0.125us per block to process PCIe Gen-3 x8 data at full rate.

Average Processing Time per Block [us]

Chunk size [byte]

Measurements done on Intel Core i7-3770 CPU
4 cores @ 3.40 GHz
Optimizations (single-threaded)

- `emplace` instead of `push_back` (Construction of objects in-place in containers)
- Moving support data structures to class-scope to avoid initializations on each algorithm call
- Pre-reserve memory for `std::vector` on initialization
- NUMA-aware memory allocations (using libnuma)
- Data prefetching using GCC’s `__builtin_prefetch`
- Compiler option tuning

Measurements done on Intel Core i7-3770 CPU 4 cores @ 3.40 GHz
Optimizations (single-threaded)

- New data structure is used only for chunks that fit into a single block
- Avoids keeping track of past blocks altogether
- Especially useful for small chunk sizes (much more likely that a chunk fits in a block)
- Downside: 2x more code

Measurements done on Intel Core i7-3770 CPU
4 cores @ 3.40 GHz
Optimizations (single-threaded)

- New data structure is used only for chunks that fit into a single block
- Avoids keeping track of past blocks altogether
- Especially useful for small chunk sizes (much more likely that a chunk fits in a block)
- Downside: 2x more code

Measurements done on Intel Core i7-3770 CPU
4 cores @ 3.40 GHz
Multi Threading

Measurements done on
Intel Xeon E5645
2x6 cores @ 2.40 GHz
Multi Threading

Measurements done on Intel Xeon E5645 2x6 cores @ 2.40 GHz
Multi Threading

Measurements done on Intel Xeon E5645
2x6 cores @ 2.40 GHz
Multi Threading

Measurements done on Intel Xeon E5645
2x6 cores @ 2.40 GHz
Memory Performance

Each line represents a different memory access pattern:

- 16, 32, 64, 128, 256 Byte accesses
- Read vs. Write
- SSE (different versions), AVX
- Scanning vs. Random Access
Memory Performance

Each line represents a different memory access pattern.
Memory Performance

RAM scanning (16 bit - 128 bit reads)

Memory Access Pattern:
Sequential reads (Scanning), but with gaps
(only trailers read, not actual data)
Memory Performance

RAM scanning (16 bit - 128 bit reads)

Memory Access Pattern:
Sequential reads (Scanning), but with gaps (only trailers read, not actual data)

random access
Roofline Analysis

Operational Intensity = Number of useful instructions performed per byte read

Performance in CPU Operations/Cycle

Measures the ‘value’ of data transfers
Roofline Analysis

Measurements done on
Intel Core i7-3770 CPU
4 cores @ 3.40 GHz
Roofline Analysis

Performance limited by memory speed

Measurements done on Intel Core i7-3770 CPU
4 cores @ 3.40 GHz
Roofline Analysis

Measurements done on Intel Core i7-3770 CPU
4 cores @ 3.40 GHz
Roofline Analysis

Measurements done on Intel Core i7-3770 CPU
4 cores @ 3.40 GHz
Roofline Model: Critic

- Hard to obtain accurate data, lots of guessing
- Result can only be seen as a first-order approximation

But: Good way to visualize the results that can be confirmed by a performance analysis tool like Intel VTune
Summary

1. FELIX: A new central event distribution layer for the ATLAS experiment

2. Optimizations and analysis of the application bottleneck (packet processing)