Accelerating the performance of distributed stream processing systems with in-network computing

Boris Koldehofe

@DEBS 2013
Short Introduction
Boris Koldehoefe
Distributed and Operating Systems Group
Technical University of Ilmenau

Research
- Distributed data analytics
- Computer system principles
- Reliability and security

Specific Focus
- Distributed Event-based systems (DEBS)
- In-Network Computing
Data Driven Applications

Nowadays everywhere!
- Autonomous driving, smart factories, smart cities, telemedicine, and many more

MAPE loop of IoT services:
- Monitor and Analyze “Things”
- Plan and Execute Processes

Insights into data key to adapt applications
- Billions of things
- Exabytes of context knowledge

But Performance and Low Latency is not straight forward!
Outline

Why low latency response?

The Bottleneck in Data Movements

In-Network Computing Technologies accelerating performance

Examples in the context of Distributed Event-Based Systems

Conclusion
Low Latency responses

Often relates to highly accurate time stamps of events

Manufacturing process
  ▪ Understand correct position over time
  ▪ Low Jitter in Communications

Telemedicine
  ▪ Understand situations with very low reaction time

Financial applications
  ▪ Algorithmic trading
  ▪ Very low responses in detecting and analyzing packets
  ▪ See DEBS 2020 Grand Challenge
Improving Timestamp Accuracy
Technological developments

5G and even 6G Campus networks
- Goal interconnect processors fast
- 100µs - 1ms delays, high mobility

TSN
- Real-time guarantees for industrial applications

Edge Computing
- Offload Computations

Accelerators
- Computation
- I/O
- Protocols / Architectures

<table>
<thead>
<tr>
<th>Timestamp inaccuracy</th>
<th>Location Inaccuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>10m</td>
</tr>
<tr>
<td>1ms</td>
<td>10cm</td>
</tr>
<tr>
<td>1µs</td>
<td>0.1mm</td>
</tr>
</tbody>
</table>

Moving Object of 36km/h
DEBS / Real-Time Analytics

Correlations on data stream
- With low end to end delay
- High accuracy detection

Paradigm:
- Operators identify pattern on partial data stream: window
- E.g. CEP operator, Filter, Neural Network, Deep Learning Model
Distributed Real-Time Analytics

Execute operator network on a distributed infrastructure
Increase Scalability and Performance

Optimization subject to potentially conflicting goals
- Decoupling producers and consumers
- Low end-to-end delay
- High accuracy
- ...

Traffic Event

Producers
Consumers

Correlation
Adaptive Distributed Application

- Producer (P)
- Consumer (C)
- Correlation (ω)

Publications → Notification → Event Pattern

Real-Time Analytics
Scalable and Network Centric Adaptation

- Pub/Sub
- CEP

Subject to
- Low Latency
- Bandwidth
- Mobility
- Reliability
- Security
- Energy

Mechanisms
- Event Distribution
- Operator Execution
- Operator Migration
- Operator Recovery
- Access Control

Building on Virtual Sensors

Virtual Compute

SDN

Compute Centric Adaptation

Subject to Low Latency, Bandwidth, Mobility, Reliability, Security, Energy

Building on Virtual Sensors

Virtual Compute

SDN

Meeting Performance of Time Sensitive Distributed Applications

Cyberphysical application
- Low latency?
- Predictable performance?

Bottlenecks in data movement and processing

Requires much more flexibility in using mechanisms of the distributed infrastructure!
Lack for Flexibility: Communication Protocols and Operating Systems

Hardcoded in network appliances

Time for data to bypass the kernel
Ingredients for Increased Flexibility

Programable hardware
- P4 Switches
- NetFPGA

New networking paradigms
- Software-Defined Networking
- Network Function Virtualization

Significant changes in the infrastructure
- Edge Data Center
- Technologies & Concepts
  - DPDK, P4, OpenFlow, RDMA

Enabler for in-network computing!
In-Network Computing

Idea enable computations on the data path

Traditionally,
- Packet header processing,
  - e.g., routing, firewall, packet classification, load balancing, deep packet inspection

Often
- Match/action pipeline model of networking hardware
- Management interface, specific programming interfaces, …
Evolution: In-Network Computing resources

Towards flexible, high performance, and energy-efficient in-network computing

- “Blackbox”
- Vendor-lock
- “Blackbox” → “Whitebox”
- Software-defined Networking
- Controller Interface
- Still “fixed” ASIC
- “Why hardware?”
- Just do it in C!
- Standard Server
- DPDK
- Performance?
- Efficiency?
- Programmable ASIC for packet processing
- Very high bandwidth
- Limited flexibility
- Energy Efficient Switching?
- Computational Intelligence inside the network?

ASIC = “fixed silicon chip for special purpose, e.g. packet switching”
Performance Acceleration via INP

INP resources can reduce the time to move data, e.g.

- DPDK: circumvent OS
- OS Kernel: Enhance Communication Protocol
- NIC: process ahead of OS
- Switch: closer to producer/consumer
Performance Acceleration via INP

INP resources can reduce the time to move data, e.g.
- DPDK: circumvent OS
- OS Kernel: Enhance Communication Protocol
- NIC: process ahead of OS
- Switch: closer to producer/consumer

INP resources can accelerate the processing time
- Efficient Matching: TCAM
- Transformation and routing

INP enables dynamic exchange of functionality
Is INP = Low Latency?  
High Performance Packet Analytics in P4STA

<table>
<thead>
<tr>
<th>actual fiber length</th>
<th>avg. latency</th>
<th>std. dev.</th>
<th>loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 m</td>
<td>1.06 m</td>
<td>107.830 ns</td>
<td>1.46 ns</td>
</tr>
<tr>
<td>2 m</td>
<td>2.08 m</td>
<td>112.850 ns</td>
<td>1.61 ns</td>
</tr>
<tr>
<td>3 m</td>
<td>3.18 m</td>
<td>118.336 ns</td>
<td>1.52 ns</td>
</tr>
<tr>
<td>10 m</td>
<td>10.12 m</td>
<td>152.883 ns</td>
<td>1.61 ns</td>
</tr>
</tbody>
</table>

P4 timestamping:

Challenges in using them for Real-time analytics

Specific domain specific programming models
- OpenFlow, P4, Verilog

Breaking distribution transparency
- E.g., applications does not work on byte streams, but packets!

Increased heterogeneity

Headers may leak information on the packet content
Outline

Why low latency response?

The Bottleneck in Data Movements

In-Network Computing Technologies accelerating performance

Examples in the context of Distributed Event-Based Systems

Conclusion
Publish/Subscribe and Performance

Efficient distribution by means of overlays

Bandwidth efficient overlays

BUT big performance gap

- Overlay
- Underlay
High Performance Publish/Subscribe: Basic Idea

Reduce the overhead:
- Message duplications
- Matching subscriptions at the hardware
High Performance Publish/Subscribe: Basic Idea

Reduce the overhead:
- Message duplications
- Matching subscriptions at the hardware

Publisher: Notification (position: xpos = 50, ypos = 30)

Subscriber: position
- 40 > xpos > 60
- 25 > ypos > 35

Subscriber: position
- 35 > xpos > 55
- 22 > ypos > 32

No Duplicates!

Hardware Filtering!

B

C1

C2

Underlay-based

SDN-based Publish/Subscribe Middleware

Publisher: Will send positions
Subscriber: position
40 > xpos > 60
25 > ypos > 35
Configuration Based on OpenFlow

Forwards packets from in ports to out ports by means of flow table, e.g.,

Controller can add, change and remove flow entries using OpenFlow

RQ: How to represent and match content-based subscriptions, e.g. in OpenFlow?
Subscription and event matching in flow table

1. Generate binary representation based on spatial indexing
2. Map binary representation to IPv6 Multicast address
   - Coexistence with other services

Approach overview

Subscription/Advertisement
- Sent to controller with predefined IP address

Controller optimizes topology
- Establish paths between publishers and subscribers
- Paths are established along a tree

Events
- Directly sent to the network
- IP Prefix o bit string
Result: Forwarding performance

Hierarchical fat-tree topology
10 Open vSwitches and 8 end-hosts
10,000 events
Properties

OpenFlow-based Management enables expressive subscription management

But requires from every publisher/ subscriber
- Understand the encoding

Relied on specific Header Fields!

But would work in general using a big field or mask

More Complex, state-full not considered!
Extending to P4 based INP

P4 supports Programming Reconfigurable Match Action Pipeline
Define own Protocol Headers to be used by DEBS
Define Matching Operations for specific Header fields

define bit<32> timestamp_t;
define bit<16> type_t;
define bit<8> attribute_t;
define bit<8> value_t;

header event_h {
    type_t type; /* example: weather. */
    timestamp_t timestamp; /* event occurrence time. */
    attribute_t attribute; /* example: humidity, temperature. */
    value_t value; /* example: 45% humidity and 23 degrees celsius temperature. */
}
P4: Enhancing Stateful Operations

Limited Support for Stateful Operations

Many pitfalls:
- No sharing of registers between different stages of the pipeline
- Exclusive read or write operations
- Packet cannot iterate over all registers

However, can be used to model for specific platforms stateful CEP operators!

Supporting parallel operator execution with P4

Operator Parallelization is a common method in DEBS

**Splitter:**
- Partition streams in independent processable windows
- Operator instances return results to the merger
- Merger coordinates streams

Processing rate of the splitter is the bottleneck in scaling operators

Can be done already on the path between producers and consumers
P4 Splitter: Window Operators

Idea: perform stream partitioning via INP

Problems:
- Dynamically expressing multiple distinct window semantics for operators
  - Time-based, Count based, …
- Needs to be performed in line-rate with
  - Match Action Logic
  - Registers state

Basic Idea / Procedure

1. Each stream identified by an id → Matching events
2. Window specifications can be dynamically added/removed
   ▪ with respect to a unique stream id (Dynamically Matching rules)
3. Window state is captured via registers
4. Incoming events trigger updates to window state (dependent on window)
5. Will be added to a multicast group that sends an event packet to all destinations
Example

Specifying a window semantics:
- e.g. Count-based Sliding Window
  - stream ID $idx$
  - parallelism degree $N$
  - window size $n$
  - window shift $\delta$, $\delta \leq n$ (counting)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x23</td>
<td>$\omega_0$</td>
<td>0</td>
<td>$\omega_0$</td>
</tr>
<tr>
<td>0x23</td>
<td>$\omega_1$</td>
<td>1</td>
<td>$\omega_0, \omega_1$</td>
</tr>
<tr>
<td>0x23</td>
<td>$\omega_2$</td>
<td>2</td>
<td>$\omega_0, \omega_1, \omega_2$</td>
</tr>
<tr>
<td>0x23</td>
<td>$\omega_3$</td>
<td>3</td>
<td>$\omega_0, \omega_1, \omega_2, \omega_3$</td>
</tr>
<tr>
<td>0x23</td>
<td>$\omega_4$</td>
<td>3</td>
<td>$\omega_1, \omega_2, \omega_3, \omega_4$</td>
</tr>
<tr>
<td>0x23</td>
<td>$\omega_0$</td>
<td>3</td>
<td>$\omega_2, \omega_3, \omega_4, \omega_0$</td>
</tr>
<tr>
<td>0x23</td>
<td>$\omega_1$</td>
<td>3</td>
<td>$\omega_3, \omega_4, \omega_0, \omega_1$</td>
</tr>
<tr>
<td>0x23</td>
<td>$\omega_2$</td>
<td>3</td>
<td>$\omega_4, \omega_0, \omega_1, \omega_2$</td>
</tr>
</tbody>
</table>

Round-robin load-balancing over 5 operator instances with $n = 4$, $\delta = 1$. 
Challenge: How to evaluate such a system?

Although P4 facilitates programming INP hardware, it is time consuming

One device costs ~5000-10000€

Huge Development effort

Don’t expect large scale comparison or a baseline comparison with Apache Flink

In general baseline comparisons
  ▪ Being faster neither straight forward nor very insightful
What did we evaluate:

P4STA for Packet generation and validation

What is the latency introduced by a INP

Measure feasible throughput

Measure resources
- How many streams, operators, and windows can be supported
Some Findings
Throughput and Low Latency

High Throughput low and stable latency

Throughput depends on window semantics
- Load generator is a bottleneck
- Higher parallelization degree and overlap increases bandwidth

Latency
- Independent of count-base vs time-based

<table>
<thead>
<tr>
<th>Measurement</th>
<th>CBTW</th>
<th>CBSW</th>
<th>TBTW</th>
<th>TBSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Latency</td>
<td>1.76 μs</td>
<td>1.86 μs</td>
<td>1.75 μs</td>
<td>1.87 μs</td>
</tr>
<tr>
<td>Minimum Latency</td>
<td>1.72 μs</td>
<td>1.8 μs</td>
<td>1.72 μs</td>
<td>1.83 μs</td>
</tr>
<tr>
<td>Maximum Latency</td>
<td>1.8 μs</td>
<td>1.92 μs</td>
<td>1.8 μs</td>
<td>1.93 μs</td>
</tr>
</tbody>
</table>

Some findings

Resource Usage

Resource Usage Determines Scalability

Comprises

- Stages, Tables, and Register Arrays
- Tofino1 has max 12 stages

<table>
<thead>
<tr>
<th>Resource</th>
<th>CBTW</th>
<th>CBSW</th>
<th>TBTW</th>
<th>TBSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stages</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Match Tables</td>
<td>12</td>
<td>12</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>Registers Arrays</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

- Could deploy with line rate-performance
  - Count-Based windows: $457k$ operators, $286k$ concurrent streams
  - Time-based windows: $362k$ operators and up to $65k$ streams
Interesting Approaches in INP for Data Driven Applications

Networking Community is working on many abstractions for Stateful INP
Challenge: understand practicality and applicability in Middleware services

But also very interesting work in distributed computing!
Everything on Performance?

Not really!

Data movements are the cause for high energy efficiency!

Moving to sustainable computing components!

Recent example
- **TCAmM**\textsuperscript{CogniGron}: Energy Efficient Memristor-Based TCAM for Match-Action Processing

ASIC = „fixed silicon chip for special purpose, e.g. packet switching“

---

Conclusion

Distributed Real-Time Analytics is a fundamental and challenging paradigm in the Internet of Things

Accelerators based on In-Network Computing
- Reduce performance bottlenecks
- Utilize the Distributed Infrastructure more efficient

Distributed systems mechanisms
- Flexible usage of heterogeneous resources
- No single mechanism fits them all

Future Research:
- Better understanding of Distributed Computing + In-Network computing
- Energy-efficiency of In-Network Computing
Questions


Prof. Dr. Boris Koldehofe
Technische Universität Ilmenau
Department of Computer Science and Automation
Distributed Systems and Operating Systems Group