Information-Centric Dataflow

Re-Imagining Reactive Distributed Computing

Dirk Kutscher, Laura Al Wardani, T M Rayhan Gias

IRTF COINRG Meeting at IETF-112
2021-11-11

Andrew Moore Retweeted

Programming Wisdom @CodeWisdom · 18h

The eight fallacies of distributed computing:
1. The network is reliable;
2. Latency is zero;
3. Bandwidth is infinite;
4. The network is secure;
5. Topology doesn’t change;
6. There is one administrator;
7. Transport cost is zero;
8. The network is homogeneous.
— L Peter Deutsch

💬 21  🔴 589  💘 2.3K
My Perspective: Two strands

Dataplane Programmability

Enhancing IETF Protocols to Support Connecting Computations

Distributed Computing

Re-imagine relationship of networking and computing

This work

Dataplane Programmability

Enhancing IETF Protocols to Support Connecting Computations

Distributed Computing

Re-imagine relationship of networking and computing

This work
Distributed Computing

Many Different Types of Interactions

- Message passing
- Remote Method Invocation
- Dataset synchronization
- Key-value store
Dataflow

Structured Distributed Data Processing
Dataflow
Structured Distributed Data Processing

Received asynchronously at $f_1$
Dataflow

Structured Distributed Data Processing

Triggering Computation at f1, Consumed by f1

input source

f1

f2

f3

accumulated result
Dataflow
Structured Distributed Data Processing

input source → f1 → f2 → f3

Newly produced result object

accumulated result
Dataflow
Structured Distributed Data Processing

Triggering computation at f2, consumed at f2

input source

f1

f2

f3

accumulated result
Dataflow

Structured Distributed Data Processing

input source → f1 → f2 → f3

Newly produced result object
Dataflow

Structured Distributed Data Processing

input source

f1

f2

f3

Newly produced result object

accumulated result
Dataflow

Structured Distributed Data Processing

input

accumulated
result
Dataflow

Poster Child Example: word-count

text-to-lines -> lines-to-words #1 -> words -> count-words #1 -> word-occurences

some lines

lines-to-words #2 -> words -> count-words #2 -> word-occurences

some lines

text file

collect-results

a:     42
the:   39
tree:  27	house: 13
dog:   4

word-occurences
Dataflow Concepts

Batch & Stream Processing

- Data objects as asynchronous events

- Stream processing: each data object processed independently (unbounded)

- Batch processing: grouping of data objects (bounded)
Dataflow Concepts

Windowing

- Slicing data sets for processing as a group (aggregation)
- One data item can be assign to more than one group
- Directing data to specific consumers
Dataflow Concepts

Timing

• Elastic data processing
• Asynchronous sourcing
• Unpredictable transport and processing delays
• Ideally: processing matches production rate
• Task of a Dataflow system: adjust processing graph to production rate and "real-time requirements"

Actual watermark:
Ideal watermark:
Event Time Skew:
Dataflow
Mainstream Implementations

• **Apache BEAM**
  • Unified programming model for data processing pipelines

• **Dataflow runners**
  • Execution environments for Dataflow applications
  • Apache Flink, Samza, Spark
  • Google Cloud Dataflow
Recent Additions to Flink
Announced at Flink Forward 2021

Buffer Debloating
Minimizing the in-flight data while keeping the network connections saturated

- Network memory buffers records to keep network connections fully utilized
- All buffered records need to be processed before a checkpoint can complete
- The more buffered records, the longer the checkpoint takes
- Ideally, Flink adjusts memory to minimize in-flight records and keep connections fully utilized

Buffer debloating (FLIP-183)
- Dynamically adjust memory wrt to consumers throughput
- Keep as many bytes as can be processed in X ms
- Stable and predictable checkpoint times under backpressure

Elastic Jobs
How to react to changing workloads?

- Long running streaming applications will eventually face changing workloads
- Risk to over/under provision
- Ideally Flink would adjust resources based on workload
Dataflow

Transport and Back Pressure

- Example: Apache Flink
- Connections connect task managers, not tasks
- Need to regulate upstream processing rates
Problem Statement

Overlays, Pipes, Address Mappings, Orchestration

- Overlays do not match the inherent logic of processing immutable data objects
  - Data is locked into connections
  - Connections are virtual channels between IP hosts
  - Orchestrator required to track resources, maintain mappings of task relationships to connections between hosts

- Elastic Dataflow requires agile function instantiation, flow graph updates etc.

- Performance is a function of upstream data rates, network throughput, processing speed
  - Limited visibility into root causes of performance problems at orchestrator
IceFlow
Information-Centric Dataflow

/infrastr1/a
/infrastr1/b
/infrastr1/c
/infrastr1/d
/infrastr1/e
/infrastr1/f
infrastr1/g
/infrastr1/h
IceFlow
Information-Centric Dataflow

/infrastr1/a
/infrastr1/b
/infrastr1/c
/infrastr1/d
/infrastr1/e
/infrastr1/f
/infrastr1/g
/infrastr1/h

/word-count/text-to-lines/
IceFlow

Information-Centric Dataflow
IceFlow

Information-Centric Dataflow

/infrastr1/a
/infrastr1/b
/infrastr1/c
/infrastr1/d
/infrastr1/e
/infrastr1/f
/infrastr1/g
/infrastr1/h

/word-count/count-words/1
/word-count/lines-to-words/1
/word-count/text-to-lines/
IceFlow
Information-Centric Dataflow

/network-count/text-to-lines/

/network-count/lines-to-words/1
/network-count/lines-to-words/2

/network-count/count-words/1
/network-count/count-words/2

/network-count/count-words/1

/network-count/count-words/2
IceFlow

Concepts

• Just Names
  • For infrastructure
  • And for actors

• Computation results as Named Data Objects
  • Usual ICN properties...

• Asynchronous data production
  • Consumer has to know when data is available

• Flow control
  • Some coupling between consumers and producers

• Garbage collection
  • Producers may be resource-constrained
  • Cannot keep data forever

/app/[actor]/[instance]/data/[partition]/[object]

<table>
<thead>
<tr>
<th>app</th>
<th>the name of the application</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor</td>
<td>the name of a Dataflow actor</td>
</tr>
<tr>
<td>instance</td>
<td>actor instance number</td>
</tr>
<tr>
<td>partition</td>
<td>monotonically increasing partition number to structure data objects on the producer's side</td>
</tr>
<tr>
<td>object</td>
<td>monotonically increasing sequence number</td>
</tr>
</tbody>
</table>

/word-count/text-to-lines/1/data/1/1
/word-count/lines-to-words/2/data/3/27

Diagram:
- Circular flow with arrows indicating data flow:
  - Nodes likely represent producers and consumers.
  - Arrows indicate data transmission.
  - "acc" likely denotes the accumulation or aggregation point.

Diagram not labeled, assuming standard data flow representation.
IceFlow Operation

Dataset Synchronization

- Producers produce data under a known prefix
  - Consumers subscribe to prefix
  - And learn update new input data
- Ideally: one prefix for whole application ("word-count")
  - Everyone could learn about all data in the app context
  - For practical reasons: need indirection
  - One prefix per consumer group
IceFlow
Windows and Result Sharing

• Need more flexibility to re-use computation results in different contexts
  • Group data objects in windows
  • Group windows under per-consumer name prefixes
IceFlow

Dataflow data and configuration

• Need additional shared information
  • Static application flowgraph
  • Actual current dynamic flowgraph
• Also: loose coupling between consumers and producers
  • Consumers reports: what windows have been processed
  • So that producer can advance
• Result: share namespace with Dataflow data and configuration info
  • Some config info represented in CRDTs (like in CFN)
IceFlow
Resource Management

- IceFlow can be smarter than receiver-driven AIMD
  - No need to fetch data that cannot be processed at throughput speed
  - "Receive Window"
- Producers should not overrun consumers
  - Output queue occupancy...
  - When consistently full: trigger scale-out

See Demo at ACM ICN-21
IceFlow

Insights So Far

• **Todays Dataflow systems are powering many data science applications**

• Overlay approach
  • Usual address mapping and virtual circuit issues
  • Limited data sharing
  • Centralized orchestration

• **Real opportunity for redesigning distributed data processing with ICN**
  • Elegant name-based approach: no mappings, no resolution – just data
  • Direct sharing of computation results
  • Potentially better visibility into network performance

• **Dataset synchronization in principle the right approach**
  • NDN Psync performance not great in experiments (NFD)
  • Also requires multicast forwarding strategy

• **Additional mechanisms needed**
  • Name-based routing (NLSR should be fine)
  • Failure recovery

• **Take-aways for COIN**
  • IceFlow an example for new protocol work
  • Breaking up overlays
  • Here: Dataflow – other interaction classes next?
This project receives funding from the German Federal Ministry for Economic Affairs and Energy (BMWi) within the "Development of Digital Technologies" framework programme and is managed by the "Digital Technologies and Applications" project agency of the German Aerospace Center (DLR) in Bonn, Germany.