Geo-Distributed Stream Processing

WPE II Presentation
Caleb Stanford
Stream Processing Applications

Internet of Things

medical devices

video streams

High-rate, distributed data streams

Required real-time decisions
Stream Processing Applications

Internet of Things

High-rate, distributed data streams

Required real-time decisions

Process large quantities of data

with low latency
(response time)

medical devices

video streams

drones
Existing Programming Solution: Distributed Stream Processing Systems (DSPS)

Process large quantities of data with low latency

- **Spark** (Apache Spark Streaming) 4.7k
- **Apache Flink** 3.7k
- **Apache Storm** 2.5k
- **Samza**
- **Apache Heron**
Limitations for Geo-Distributed Data

High Bandwidth Use

High Latency (response time)
Limitations for Geo-Distributed Data

Bandwidth between Ireland and California

Bandwidth Congestion ⇒ Periods of High Latency
Next Generation Stream Processing Systems?

- Embrace the **geo-distributed** nature of data
- Process data **near the source** when possible

“Edge Computing” Paradigm
Outline

● Background
  ○ Distributed Stream Processing Systems -- Programming Model
  ○ The Geo-Distributed Streaming Problem

● Recent Approaches
  ○ Optimal Operator Placement (2016)
  ○ SpanEdge (2016)
  ○ AWStream (2018)

● Comparisons and Summary

DSPS: The Dataflow Programming Model

“Total pedestrian activity, by geographic area, in the last 10 minutes”
DSPS: The Dataflow Programming Model

“Total pedestrian activity, by geographic area, in the last 10 minutes”
DSPS: The Dataflow Programming Model

“Total pedestrian activity, by geographic area, in the last 10 minutes”
The Geo-Distributed Stream Processing Problem

input (video data) → classify → sliding window (10 min) → output

High Bandwidth Use
The Geo-Distributed Stream Processing Problem

Dataflow Graph

System

- Schedule program operations at physical nodes
- Minimize Latency
- Minimize Bandwidth Use

Geo-Distributed
- Source Streams
- Output Consumers
- Physical Nodes / Clusters
Overview of Approaches

- Optimize Scheduling of Operators
  - Optimal Operator Placement
- Extend the Programming Model
  - SpanEdge
- Approximate Expensive Data Streams
  - AWStream

(see the report for other papers in these categories)
Optimal Operator Placement

Operator Placement Problem

- Minimize Latency
- Minimize Bandwidth Use
Optimal Operator Placement

Formalized problem: MILP

Latency
= max along a path
  [sum of delays on the path]

Network Load
= sum over DSP links
  [traffic between corresponding DSP nodes
   (if different)]

\[ N = \sum_{i,j,i',j'} \lambda_{i,j} y((i,j),(i',j')) [i' \neq j'] \]

Other metrics...
Optimal Operator Placement: Implementation

Profiling:
- Network delays
- Network traffic
- Processing time per element
- Node availability
- ...

Distributed sources and nodes

Formal Optimal Operator Placement

MILP solver

Placement (Scheduler)
Optimal Operator Placement: Evaluation

Matches existing hand-tuned solutions (Bandwidth and Latency)
Optimal Operator Placement: Scalability

NP complete ⇒ Heuristics?
Optimal Operator Placement: Summary

Formalizes Operator Placement as MILP

- Matches performance of metric-specific solutions
- Extensible
- Optimal solver not scalable (exponential)

Research directions:

- Scalability: better heuristics
- Formalize throughput (not just bandwidth and latency)
SpanEdge: Programming Extensions Approach

Optimal Operator Placement

SpanEdge: lightweight extension to DSPS for geo-distributed stream processing

Simple programming abstraction?

Sweet Spot

Solver + Profiling Overhead

Programmer Effort (Manual Optimizations)
**SpanEdge: Example**

Local Task 1

- video data
- classify
- High Bandwidth Use

Global Task 1

- sliding window (10 min)
- output

**Local Task:** Replicate + process data at the source

**Global Task:** Process data in central cluster
SpanEdge: Implementation

Local Task: Replicate + process data at the source

Global Task: Process data in central cluster
SpanEdge: Evaluation

Fig. 8: The overall bandwidth consumption.

Fig. 9: The average tuple processing time (the local result).

Fig. 10: The average tuple processing time (the global result).
SpanEdge: Summary

Local Tasks and Global Tasks for DSPS

- “Sweet spot” programming abstraction to hint at optimization
- Lightweight (low overhead)

Questions and Directions:

- Evaluation leaves room for improvement
- Learn local / global tasks automatically? (Compare also: Geelytics)
- Higher level abstractions
AWStream: Approximation and Degradation

Relax the requirements: answer can be approximate

Bandwidth congestion or constraints $\Rightarrow$ Degrade Data Streams (before sending over the network)
- Low bandwidth
- No latency spikes
AWStream: Example

Video data → classify → sliding window (10 min) → output

High Bandwidth Use ⇒ Degrade Stream

Video degradation functions:

- Reduce Frame Rate
- Reduce Resolution

Programming extension for degradation: “knobs”
AWStream: Implementation

Accuracy vs. Bandwidth

- Reduce Frame Rate
- Reduce Resolution

Depends on application

Profile + adjust dynamically to network and accuracy changes
AWStream: Evaluation

![Graph showing evaluation of AWStream and other methods on the axes of fidelity (accuracy in %) and freshness/latency reversed (seconds).]
AWStream: Evaluation

Response to bandwidth congestion

(t=200 to 400)

Not shown:
Profiling Overhead
AWStream: Summary

Degrade data streams to adapt to bandwidth variations

- Performs well with thorough evaluation
- High profiling overhead (exponential in # knobs)

Questions and Directions:

- Reducing profiling overheads
- Programming models for degradation
## Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal Operator Placement [1]</td>
<td>N/A</td>
<td>Low</td>
<td>Exponential (in MILP)</td>
<td>Emulated</td>
<td>100-150ms</td>
<td>Matches [33, 36, 44]</td>
</tr>
<tr>
<td>SpanEdge [2]</td>
<td>Local/global tasks</td>
<td>N/A</td>
<td>≈ Linear (in program)</td>
<td>Emulated</td>
<td>&lt; 1ms local; ≈ 25ms global</td>
<td>≈ ( \frac{1}{4} \times ) Storm</td>
</tr>
<tr>
<td>AWStream [3]</td>
<td>Maybe knobs for degradation</td>
<td>High</td>
<td>Exponential (in # knobs)</td>
<td>Real</td>
<td>&lt; 50ms; ≈ 100ms during low bandwidth</td>
<td>Adapts to available</td>
</tr>
</tbody>
</table>
Research Directions

- No experimental comparison or benchmarks exist
- Data privacy
- Iterative and machine learning computations
- Higher level abstractions
Summary

1. **Optimal Operator Placement:**
   - Formal, extensible optimization problem

2. **SpanEdge:**
   - Lightweight programming extension

3. **AWStream:**
   - Adaptive degradation of data streams

**Geo-Distributed Stream Processing Systems**