THIS COURSE

• Seminar, 2 pts

• Lectures: Mondays 10-12, CHN G22

• Staff:

  • Vasiliki (Vasia) Kalavri (kalavriv@inf.ethz.ch) CAB E 63 (En)
  • Desislava Dimitrova (dimitrova@inf.ethz.ch) CAB E61.2 (En, De)

• Lecturers: us, you, guests from academia & industry

https://www.systems.ethz.ch/courses/spring2018/dsp
ABOUT ME

• Postdoctoral Fellow at ETH Zürich
  • Systems Group: https://www.systems.ethz.ch/

• Previously at KTH, Sweden & UCLouvain, Belgium

• PMC member of Apache Flink

• Research interests
  • Large-scale graph processing
  • Streaming dataflow engines

www.systems.ethz.ch/people/vasiliki-kalavri
ABOUT DESISLAVA

• Postdoctoral Researcher at ETH Zürich
  • Systems Group: https://www.systems.ethz.ch/

• Previously at University of Bern, University of Twente (NL)

• Networking person turned systems researcher

• Research interests
  • Software-Defined Networks
  • Streaming engines
  • Understanding datacenter behavior

https://www.systems.ethz.ch/people/desislava-cvetanova-dimitrova
OUR PROJECT: STRYMON

Datacenter

event streams

traces, configuration, topology updates, ...

policy enforcement, what-if scenarios, ...

queries, complex analytics, simulations, ...

Strymon

Datacenter state

strymon.systems.ethz.ch
WHAT YOU WILL LEARN (1)

• Design and architecture of distributed stream processing systems

• Research topics and open issues

• Stream Processing Systems
• Fault-tolerance
• Dynamic Scaling
• State Management
• Handling Time
• Windowing
• Query languages and Complex Event Processing
WHAT YOU WILL LEARN (II)

• How to
  • critically read and understand a research paper
  • briefly and effectively present a technical paper
  • review a research paper
<table>
<thead>
<tr>
<th>Date</th>
<th>Lecture</th>
<th>Presenters</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.02</td>
<td>Introduction to Stream Processing [1, 2]</td>
<td>V. Kalavri</td>
</tr>
<tr>
<td>26.02</td>
<td>Stream Processing Systems I [3, 4]</td>
<td></td>
</tr>
<tr>
<td>05.03</td>
<td>Stream Processing Systems II [5, 6]</td>
<td></td>
</tr>
</tbody>
</table>
| 12.03  | **Guest Lecture:** Apache Kafka  
| 19.03  | **Guest Lecture:** Timely & Differential Dataflow  
| 26.03  | Fault Tolerance I [9, 10]                        |                  |
| 16.04  | **Self-study**                                   |                  |
| 23.04  | Dynamic Scaling & State Management I [12, 13]    |                  |
| 30.04  | Dynamic Scaling & State Management II [14, 15]   |                  |
| 07.05  | Handling Time [16, 17]                           |                  |
| 14.05  | Windowing [18, 19]                               |                  |
| 28.05  | Libraries and Languages [20, 21]                 |                  |
GRADING SCHEME

- No exam
- Grade will be based on participation in class, presentation, and reports.

Workload:

- Read one paper per week and for that
  - prepare a set of questions for discussion during the lecture (20%)
  - list the paper's strong, weak, and interesting points (50%)
- One 20' presentation (20%)
- One complete review (10%)
WEEKLY HOMEWORK

• Questions for discussion in class
  • 3 questions about the paper, e.g.:
    • how does the system guarantee correct results in the face of failures?
    • how would you extend the system to support dynamic workloads?

• Short report
  • a list of strong, weak, and interesting points
PRESENTATION

20’ presentation of a paper of your choosing

• motivation and problem statement
• description of techniques and proposed solutions
• implementation details
• main results
• comparison with related work
• your opinion
What's a review?

• expert’s opinion that justifies accepting / rejecting the paper to a conference / journal

• feedback to the authors on how to improve the work and/or paper presentation

Writing reviews for system conferences:

REVIEW REPORT STRUCTURE

• **Summary:** one paragraph summarizing the paper
  • what is the problem this paper is addressing?
  • what is the proposed solution?
  • how is it different than existing approaches?
  • what are the main results?

• **Contributions:** why is this paper important
  • what is novel in this paper?
  • what will someone learn by reading it?

• **Your opinion:** what you liked and didn’t like about the paper
  • what did you find cool about the paper? Any “aha!” moments when reading?
  • how would you improve it?
INTRODUCTION TO DISTRIBUTED STREAM PROCESSING
WHAT IS A STREAM?

• In traditional data processing applications, we know the entire dataset in advance
  • e.g. tables stored in a database

• Data streams are high-volume, real-time data that might be unbounded
  • we cannot store the entire stream in an accessible way
  • we have to process stream elements on-the-fly using limited memory
EXAMPLE STREAMS AND APPLICATIONS

- Sensor measurements
  - anomaly detection, incident risk calculation
- Financial transactions
  - fraud detection, stock trading
- Vehicle location data and traffic data
  - report train system status, find optimal routes
- Web logs
  - online recommendations, personalization
- Network packets
  - intrusion detection, load balancing
- Online social interactions
  - trending topics, sentiment analysis
Stream processor

Input streams

Output streams

in-memory storage

external stable storage

ad-hoc queries

Standing queries
DATA STREAM BASICS

• Events/Tuples: elements of computation - respect a schema
• Data Streams: unbounded sequences of events
• Stream Operators/Tasks: consume and produce data streams
• Events are consumed once - no backtracking!
• Queries are evaluated on all events seen so far or on most recent event

where are computations stored?
We cannot infinitely store all events seen

- **Synopsis**: A stream summary

  - It is in principle any streaming operator state

- Examples: samples, histograms, sketches, state machines…

1. process t, s
2. update s
3. produce t’
DATAFLOW COMPUTATIONS

Twitter source -> Extract hashtags -> Count topics -> Trends sink

source -> map -> topK -> print
DISTRIBUTED EXECUTION

- scale to high-volume streams
- exploit task and data parallelism
- issues: computation progress, fault-tolerance and result guarantees, automatic scaling and state migration, out-of-order processing
The 8 Requirements of Real-Time Stream Processing

Michael Stonebraker
Computer Science and Artificial Intelligence Laboratory, M.I.T., and StreamBase Systems, Inc.
stonebraker@csail.mit.edu

Uğur Çetintemel
Department of Computer Science, Brown University, and StreamBase Systems, Inc.
ugur@cs.brown.edu

Stan Zdonik
Department of Computer Science, Brown University, and StreamBase Systems, Inc.
sbz@cs.brown.edu

Models and Issues in Data Stream Systems

Brian Babcock  Shivnath Babu  Mayur Datar  Rajeev Motwani  Jennifer Widom

Department of Computer Science
Stanford University
Stanford, CA 94305

{babcock, shivnath, datar, rajeev, widom}@cs.stanford.edu
HOW TO READ A PAPER


- First pass (10'-15’)
  - Read title, abstract, introduction
  - Read section and subsection headings
  - Read the conclusions

- Second pass (up to 1h)
  - Read the whole paper but ignore details (e.g. proofs)
  - Carefully look at diagrams, figures, results

- Third pass (up to 4 hours)
  - Carefully read every detail
  - Think about how you would have done it
1. What type of paper is this? (measurements, analysis of existing system, vision, research prototype, etc.)

2. Which other papers is it related to?

3. What are the assumptions made and are they valid?

4. What are the paper’s main contributions?

5. Is it well-written?
REQUIREMENTS FOR STREAMING SYSTEMS (I)

1. Process events *online* without storing them

2. Support a high-level language (e.g. StreamSQL)

3. Handle missing, out-of-order, delayed data

4. Guarantee deterministic (on replay) and correct results (on recovery)
5. Combine batch (historical) and stream processing

6. Ensure availability despite failures

7. Support distribution and automatic elasticity

8. Offer low-latency
<table>
<thead>
<tr>
<th>Feature</th>
<th>DBMS</th>
<th>Rule engine</th>
<th>SPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep the data moving</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL on streams</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Handle stream imperfections</td>
<td>Difficult</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Predictable outcome</td>
<td>Difficult</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>High availability</td>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Stored and streamed data</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Distribution and scalability</td>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Instantaneous response</td>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
</tbody>
</table>

Table 1: The capabilities of various systems.
NEXT WEEK’S PAPERS

• The Design of the Borealis Stream Processing System

• Aurora: a new model and architecture for data stream management