CAS CS 551 Streaming and Event-driven Systems

Course Syllabus Spring 2025

Instructor Name: Vasiliki Kalavri
Instructor Office Hours: Tue 5-7pm (CCDS 713, 7th floor).

Lectures: Tue-Thu 3:30pm - 4:45pm, CAS 201
Discussions: Wed 12:20pm - 1:20pm, CAS B36

Teaching Fellow: Muhammad Faisal
TF Office Hours: Thu 1pm - 3pm (Open space near CCDS 713, 7th floor).

IMPORTANT: Refrain from using email to reach the course staff. To contact the instructor or TF, send **a private Piazza post**.

1. Overview

1.1 Course Description

Modern data-driven applications increasingly require continuous, low-latency processing of large-scale, rapid data events such as clicks, search queries, online interactions, financial transactions, traffic records, and sensor measurements. Distributed stream processing has become highly relevant to industry and academia due to its capabilities to both improve established data processing tasks and to facilitate novel applications with real-time requirements. In this course, you will learn how to design, implement, and evaluate scalable and reliable stream processing and event-driven applications.

Specifically, we will cover the following topics:

- Publish/Subscribe systems
- Architecture of distributed stream processing systems
- Dataflow programming
- Fault-tolerance and processing guarantees
- Streaming state management
- Windowing semantics and optimizations
- Complex event processing
- Microservice architectures
- Serverless functions and their relationship to stream processing

1.2 Course Objectives

At the end of the course, successful students will have gained skills and hands-on experience on the following methods and technology:

- Design and implementation of dataflow stream processing applications

- Message queues, log-based message brokers, and publish/subscribe systems
- Ability to comprehensively compare the features, architecture, and processing guarantees of modern streaming systems
- Implementing, deploying, and evaluating event-based applications with Apache Flink and Apache Kafka.
- Operations for scalable and reliable stream processing, including logging, monitoring, debugging, and upgrading streaming applications.
- A solid understanding of the challenges and trade-offs one needs to consider when designing and deploying streaming applications.

Further, students will be exposed to recent developments in stream processing research through paper assignments and presentations. The collaborative semester-long project will prepare them for the practical aspects of their future careers and expose them to project management tools and software engineering best practices.

1.3 Prerequisites

CAS CS 112 and CAS CS 210; CAS CS 351 and CAS CS 460 or consent of instructor.
To be successful in this course, students will need to have strong programming skills, a solid understanding of Computer Systems fundamentals (CS 210) and some prior experience with object-oriented programming / Java (CS 211). Familiarity with Distributed Systems (CS 351) and Database Systems (CS 460) is highly recommended.

2. Instructional Format, Course Pedagogy, and Approach to Learning

2.1 Courseware

• We will use the **course website** to maintain an up-to-date class schedule:
https://vasia.github.io/cs551/index.html
• We will use **Piazza** for announcements, questions, discussions, and all other communication:
https://piazza.com/bu/spring2025/cascs551/home
• We will use **Github Classroom** for the discussions and the semester projects:
https://classroom.github.com/classrooms/193693607-cs-551-sp25
• We will use **Gradescope** for assignment submissions:
https://www.gradescope.com/courses/951192

2.2 Lectures

Lectures will be held during the assigned time slots. Section 6 of the syllabus provides the topic and assigned readings for each lecture. You are expected to *complete the readings before the day of the lecture and actively participate in class discussions*. Lecture slides will be made available on the class website prior to the lectures or shortly after.

2.3 Discussions

Discussions are a core component of the course and *discussion attendance is mandatory*. The Teaching Fellow will lead the discussion sessions and present material on the required tools

such as Apache Flink and Apache Kafka, that reinforce the concepts covered in the lectures, and answer questions (or provide clarifications) regarding the assignments and projects. The Teaching Fellow will post information to Piazza as necessary. In addition to the discussions, the Teaching Fellow will hold weekly Office Hours.

2.3.1 Software requirements

During the discussion sessions, you will solve a set of programming exercises using Apache Flink and Apache Kafka. Use your own laptop computer and make sure to set up your environment correctly as described below.

You can develop and execute Flink applications on Linux, macOS, and Windows. However, UNIX-based setups have complete tooling support and are generally preferred by Flink developers. **All assignments assume a UNIX-based setup**. If you are a Windows user, you are advised to use Windows subsystem for Linux (WSL), Cygwin, or a Linux virtual machine to run Flink in a UNIX environment.

To setup and run Flink, you additionally need:

- A **Java 11** installation. To develop Flink applications and use its DataStream API in Java or Scala you will need a Java JDK. A Java JRE is not sufficient!
- Apache **Maven 3.x or above**. Flink provides Maven archetypes to bootstrap new projects.
- An IDE for Java development. Common choices are IntelliJ IDEA, Eclipse, or Netbeans with appropriate plugins installed. **We recommend IntelliJ IDEA**.

Even though Apache Flink is a distributed data processing system, you will typically develop and run initial tests on your local machine. This makes development easier and simplifies cluster deployment, as you can run the exact same code in a cluster environment without making any changes.

2.5 Course Materials

There is no required textbook for this class. Slides, lecture notes, and other publicly available resources will be published on the course website and on Piazza. A list of readings is provided at the end of this document. You should be able to access all readings when connected to the campus network. Please contact the instructor if any of the listed readings is unavailable or inaccessible.

3. Assignments and Grading Criteria

3.1 Semester Project

This class is highly collaborative and research-oriented. During the second week, you will be provided with a list of semester projects and you will be asked to select your top-3 preferences. You will then be assigned to a project team with 3-5 students. During the semester, the team will be working together to deliver the following assignments:

- A **design document** outlining (1) the project goals, (2) an implementation and evaluation plan, (3) the task distribution among team members.
- A **midterm project demo**. Demos will be presented during class time.
- A **final demo** to be presented during the last week of class.
- The project's **github repository**, including code, tests, automation and plotting scripts, and documentation.

3.2 Paper assignments
During the semester, we will read and discuss various technical papers. For each paper, you will be asked to submit:

1. a short quiz assessing your understanding of the paper's core ideas

2. a list of questions to be discussed during lecture time. **These deliverables are individual.**

Paper assignments are always due at 11:59pm on the day before the corresponding lecture. See Section 6 for the list of papers and the tentative dates for each paper.

3.3 Discussion quizzes
At the end of each discussion session, you will be asked to complete a short quiz in class. These quizzes are designed to assess your understanding of the discussion material and identify confusions and areas for improvement. Details about the format and content of the quizzes will be posted on Piazza.

Discussion quizzes are always due at the end of the discussion session. You can miss up to 2 discussion quizzes without penalty.

3.4 Grading Scheme
Your final grade will be determined as follows:

1. Participation & effort (**10%**):
- In-class participation.
- Discussion participation.
- Piazza contributions.
- Git activity (project + discussions).
- Group meeting minutes.
- Office Hours participation.

2. Paper assignments (**25%**):
- Quizzes (15%)
- Questions (5%)
- Paper discussion participation (5%)

3. Discussion quizzes (**15%**)

4. Semester project (**50%**) (in teams of 3-5 students):
- Design document (maximum 3 pages) 5%.
- Midterm demo 15%.
- Final demo and deliverables: 30%.

The final deliverables include (1) the full code implementing the project tasks as defined in the project design document, (2) auxiliary code for data pre-processing, deployment, and testing, (3) complete supporting documentation.

Individual contributions to collaborative assignments will be assessed by taking into account the following:
- The quality of individual task deliverables outlined in the project design document.
- The individual's ability to answer questions about the project during demo presentations and office hours.
- The individual's performance during the paper and demo presentations.
- The individual's contribution to the project's gitlab repository (git history).

There is no final exam at the end of the course.

4. Class and University Policies

4.1 Homework submission

All assignments and the project deliverables will be submitted via the course Gitlab. All deliverables are **due by 11:59pm on the day of the respective deadline**.

4.2 Attendance

Students are expected to attend each class session unless they have a valid reason for being absent. Acceptable excused absences include observing religious holidays and illness. In such cases, students are advised to contact the instructor as soon as possible, so that reasonable accommodations can be provided. Please review the **BU attendance policy** and the **BU Policy on Religious Observance** for more information.

4.3 Late work policy

Students who submit homework late will only be eligible for up to **50% of the original score**.

4.4 Academic conduct

Academic standards and the code of academic conduct are taken very seriously at our university. Please take the time to review the CAS Academic Conduct Code: http://www.bu.edu/academics/resources/academic-conduct-code/ and the GRS Academic Conduct Code: http://www.bu.edu/cas/students/graduate/grs-forms-policies-procedures/academic-discipline-procedures/. Please review the sections regarding plagiarism and cheating carefully. Copies of the CAS Academic Conduct Code are also available in room CAS 105. A

student suspected to violate this code will be reported to the Academic Conduct Committee, and if found culpable, the student will receive a grade of "F" for the course

All assignments must be completed individually, unless instructed otherwise. Discussions with fellow students via Piazza or in-person are encouraged, but presenting the work of another person as your own is expressly forbidden. This includes “borrowing”, “stealing”, copying programs/solutions or parts of them from others. Note that we may use an automated plagiarism checker. *Cheating will not be tolerated under any circumstances*.

Any resources, including material from other students (current or past), that are used, beyond the text or that provided by the TF or professor must be clearly acknowledged and attributed. Using such material may result in a lower grade. However, if such material is used and not acknowledged and attributed, it will automatically be considered as possible academic misconduct.

4.5 On the use of GenAI tools

AI tools, such as chatGPT, should not be used to generate written text or code for your assignments in this course. *Uploading any course content (assignments, papers, slides) to a Generative AI tool will be treated as a case of possible academic misconduct.* However, you are allowed to use AI tools for enhancing your understanding of core concepts, brainstorming, and identifying external knowledge sources. Beware that using AI-generated content in your assignments and projects could negatively affect your grade, as it may contain subtle errors, omissions, bugs, and misinterpretations. *Any use of AI software must be explicitly disclosed in your submissions, including the prompts you used.*

5. Accommodations

If you are a student with a disability or believe you might have a disability that requires accommodations, please contact the Office for Disability Services (ODS) at (617) 353-3658 or access@bu.edu to coordinate any reasonable accommodation requests. ODS is located at 25 Buick Street on the 3rd floor.

6. Detailed Schedule

The rest of the syllabus is tentative and might be updated during the semester. We will be keeping you informed of any changes made to the readings or assignment deadline via Piazza. Make sure to become familiar with the **Official Semester Dates**.

Date	Topic	Readings	Assignment
1/21	Introduction to stream processing	[1]	
1/22	Disc #1: Introduction to Flink		
1/23	Publish/Subscribe systems	[2]	
1/28	**Paper 1**: Storing data streams (Quiz and questions due on 1/27 11:59pm)	[3]	Projects announced
1/29	Disc #2: Intro to Kafka		
30/1	Dataflow stream processing systems	[4]	
2/4	**Paper 2**: Apache Kafka Streams (Quiz and questions due on 2/3 11:59pm)	[5]	Project selection due
2/5	Disc #3: Writing Flink programs & DataStream API		
2/6	Notions of time	[6]	
2/11	**Paper 3:** Managing disorder (Quiz and questions due on 2/10 11:59pm)	[7]	
2/12	Disc #4: Windows & event-time I		
2/13	Windows	[8]	
2/19	Disc #5: NERC		
2/20	Windows cont.	[8]	Design document due
2/25	State management I	[9]	
2/26	Disc #6: Windows & event-time II		
2/27	**Paper 4**: Window aggregation (Quiz and questions due on 2/26 11:59pm)	[10]	
3/4	State management II	[9]	
3/5	Disc #7: State Management		

3/6	**NO CLASS**		
3/18	**GUEST LECTURE**		
3/19	Disc #8: Checkpoints		Midterm presentation due
3/20	Midterm project presentations		
3/25	Midterm project presentations		
3/26	Disc #9: Metrics & monitoring		
4/1	Streaming operator placement	[11]	
4/2	Disc #10: Project hacking		
4/3	**Paper 5**: Reactive scaling (Quiz and questions due on 4/2 11:59pm)	[12]	
4/8	Distributed snapshots	[13]	
4/9	Disc #11: Project hacking		
4/10	**Paper 6**: Fault-tolerant stream processing (Quiz and questions due on 4/9 11:59pm)	[14]	
4/15	Flow control, backpressure, elasticity	[15-16]	
4/16	Disc #12: Project hacking		
4/17	Stream query optimization	[17]	
4/22	**Paper 7**: Streaming at the edge (Quiz and questions due on 4/21 11:59pm)	[18]	
4/23	Substitute Monday (no discussion)		Final presentation due
4/24	Demo presentations		
4/29	Demo presentations		
5/1	Last day of classes: An overview of Flink 2.0		Final repositories due

Readings

[1] Streaming 101: https://www.oreilly.com/radar/the-world-beyond-batch-streaming-101/

[2] The many faces of publish/subscribe: https://dl.acm.org/doi/pdf/10.1145/857076.857078

[3] Pravega: A Tiered Storage System for Data Streams
https://dl.acm.org/doi/pdf/10.1145/3590140.3629113

[4] Streaming 102: https://www.oreilly.com/radar/the-world-beyond-batch-streaming-102/

[5] Consistency and Completeness: Rethinking Distributed Stream

Processing in Apache Kafka https://dl.acm.org/doi/pdf/10.1145/3448016.3457556
[6] Flexible time management in data stream systems:
https://dl.acm.org/doi/pdf/10.1145/1055558.1055596
[7] Impatience is a Virtue: Revisiting Disorder in High-Performance Log Analytics
https://www.microsoft.com/en-us/research/uploads/prod/2018/04/impatience-icde18.pdf
[8] Survey of window types for aggregation in stream processing systems
https://link.springer.com/content/pdf/10.1007/s00778-022-00778-6.pdf
[9] State Management in Apache Flink: https://dl.acm.org/doi/10.14778/3137765.3137777
[10] LightSaber: Efficient Window Aggregation on Multi-core Processors
https://lsds.doc.ic.ac.uk/sites/default/files/lightsaber-sigmod20.pdf
[11] CAPSys: Contention-aware task placement for data stream processing
https://hdl.handle.net/2144/49285
[12] Sponge: Fast Reactive Scaling for Stream Processing with Serverless Frameworks
https://www.usenix.org/system/files/atc23-song.pdf
[13] Lightweight Asynchronous Snapshots for Distributed Dataflows:
https://arxiv.org/pdf/1506.08603.pdf
[14] MillWheel: fault-tolerant stream processing at internet scale:
https://research.google/pubs/pub41378/
[15] Three steps is all you need: fast, accurate, automatic scaling decisions for distributed streaming dataflows: https://www.usenix.org/system/files/osdi18-kalavri.pdf
[16] A Survey on the Evolution of Stream Processing Systems:
https://arxiv.org/pdf/2008.00842.pdf
[17] A catalog of stream processing optimizations: https://dl.acm.org/doi/10.1145/2528412
[18] WASP: Wide-area Adaptive Stream Processing:
https://dl.acm.org/doi/pdf/10.1145/3423211.3425668