CECS 327 – Networks and Distributed Computing

Semester Project: Distributed Systems and Applications

Project Goals

- Apply distributed systems and networking concepts to an open-ended, real-world-inspired project.
- Develop experience in software design, performance testing, and research-based experimentation.
- Improve your ability to work with uncertain, open-ended problems.
- Learn how to analyze and communicate performance results effectively.
- Explore how cutting-edge tools like large language models (LLMs), distributed ML frameworks, and edge/cloud platforms are transforming modern computing.

You should submit the required deliverable materials on Canvas by **11:55pm, August 10th (Sunday), 2025.**

Project Overview

This semester project is designed to help you bridge the gap between classroom concepts and real-world distributed computing. You'll undertake a substantial technical and research-oriented group project involving the design, implementation, testing, and evaluation of a distributed system, possibly enhanced by AI/LLM-based methods. You will simulate the research workflow by picking a topic, designing an architecture, experimenting with performance trade-offs, and documenting your findings. Topics may range from traditional distributed systems to modern applications integrating AI and large-scale models.

Use this opportunity to explore how distributed computing intersects with AI—including large language models (LLMs), multi-agent systems, data stream processing, function-as-a-service, and cloud/edge intelligence.

Creating a comprehensive grading rubric for these projects is challenging due to their unique nature. Not every group needs to target an 'A' grade to achieve success. However, here are general benchmarks for success at each grade level:

- **A Level (Novelty)** - This project showcases a notable and well-executed new implementation and/or analysis of a distributed system. It might involve a fresh system (***selected platform was released by 2021 or later***) or a creative variation of an existing one, integrating numerous concepts covered in the course. The project's documentation should encompass an in-depth analysis and discourse on performance, scaling tendencies, and pertinent subjects. Such projects often qualify for potential publication.
- **B Level (Insight)** - This project reveals significant insights into a distributed system. While it might or might not involve novel implementations (***selected platform was released between 2015 to 2020***), it should adeptly apply a range of class-discussed concepts. The project's documentation should feature an exhaustive analysis and dialogue regarding performance, scaling characteristics, and other relevant aspects.

- **C Level (Distribution)** - This project exemplifies a distributed system that utilizes at least two or three concepts covered in class. The project's documentation should encompass a discussion about performance, scaling tendencies, and other pertinent subjects.
- **Note**: The choice made at each of the three levels does not singularly determine your final level grade. The level's decision contributes to approximately 20% of the overall final grade, indicating that the primary emphasis remains on the performance and design of the entire project.

However, remember this: **stay calm**! This project has been known to induce unnecessary anxiety because of its open-ended nature. However, mastering the skill of excelling in projects with no predefined boundaries is a valuable skill that will likely prove essential in your career journey. Think of this as an opportunity to cultivate that skill in a relatively low-pressure environment. Whenever possible, try to view this project as a chance for growth rather than a hurdle.

Group Work

• Up to 2 students per group.
• All group members should contribute significantly and clearly indicate their roles in the final report.

Example Topics & AI-Enhanced Ideas

- Distributed Systems & Parallel Architectures
    - Parallelize and optimize existing code using MPI, Spark, or Ray.
    - Analyze strong vs. weak scaling behavior of a distributed algorithm.
    - Design a microservice-based system using Docker/Kubernetes.
- AI and LLM Integration
    - Deploy and compare distributed LLM inference using APIs (OpenAI, HuggingFace) across multiple nodes.
    - Build a chatbot that runs on a distributed backend, where services like intent recognition, response generation, and data retrieval are decoupled across nodes.
    - Implement a distributed vector database using FAISS or Weaviate to support semantic search over documents.
    - Create an edge-based AI application using a lightweight LLM (e.g., Mistral, TinyLlama) with inference split across devices.
    - Investigate distributed fine-tuning or inference of LLMs using HuggingFace Accelerate or DeepSpeed.
- Cloud/Edge/IoT Systems
    - Develop a function-as-a-service app using AWS Lambda, OpenFaaS, or Google Cloud Functions.
    - Build a distributed monitoring system for smart IoT devices.
    - Train and deploy a federated learning model across Raspberry Pi or Android devices.

- Performance & Analysis
    - Compare latency and throughput of a distributed queue (e.g., Kafka vs. Redis Streams).
    - Run experiments on cloud VMs or clusters to measure real-time performance.
    - Analyze scaling behavior of a distributed LLM serving platform.

Deliverables (Due: 11:55pm, August 10, 2025)

Deliverable	Details
README	How to compile and run your system
Source Code	Include GitHub link or zipped files with comments
Project Report (PDF/IEEE Style)	Include design, implementation, performance analysis, and contribution
Demo Video	Show the system running and explain the output and performance results

Note: Use professional formatting and provide citation for any external sources. Projects using LLMs or AI must clearly describe the role of these tools (but you can not copy code directly from tools like ChatGPT).

Grading Rubric

Criteria	Points
README with compile/run instructions	5 pts
Well-commented source code	15 pts
Demo video showcasing system execution	10 pts
Complete report with clear explanation & analysis	30 pts
Individual contributions clearly stated in report	5 pts
Working, compilable code with valid output	35 pts
Total	100

Tips for Success

- Start early! Plan weekly meetings and checkpoints.
- Schedule a meeting with the instructor if you're unsure of your topic.
- Stay calm and treat this as a low-risk opportunity to grow.
- Build something you'd be proud to show off to a future employer.