SW 401- Course Project Description (Fall 2025)

Overall Project Goal

Design, implement, and analyze a scalable parallel and distributed system that evolves step-by-step through the three phases.

You will begin from a sequential baseline and gradually integrate shared-memory, distributed, and big-data components, demonstrating measurable speedup, scalability, and fault tolerance.

Project Phases Overview

Phase	Focus	Deliverables	Deadline
Phase 1 – Parallel Foundations (25%)	Develop sequential and shared-memory parallel versions of your application (OpenMP). Measure performance.	Report + code + plots + demo video	**Week 6**
Phase 2 – Scaling Across Nodes (35%)	Extend to distributed memory using MPI and CUDA; evaluate scaling on multi-node or GPU cluster.	Working MPI/CUDA code + report + benchmark graphs+ demo video	**Week 10**
Phase 3 – Resilience & Big Data Integration (40%)	Wrap the system into a service or data pipeline using gRPC, Spark, or Flink; demonstrate fault tolerance.	Final report + presentation + code repo	**Week 14**

Suggested Project Topics (choose one)

1. Parallel Image/Video Processing – edge detection, convolution, motion detection.
2. Scientific Simulation – Finite Elements, N-body, heat diffusion.
3. Graphics and Visualization: physics-based particle simulation, ray tracing renderer, fractal rendering (Mandelbrot/Julia Set)
4. Search / Optimization – parallel pathfinding, genetic algorithm, Monte Carlo simulation, Parallel Traffic Flow Simulator, Distributed Graph PageRank
5. Data Analytics / Stream Processing – log aggregation, parallel word count, stream filtering, real-time anomaly detection in system logs

Each project (across the 3 phases) must show measurable speedup, explain overheads, and apply at least four programming models (OpenMP, MPI, CUDA, Spark, Flink, etc.).

Team Composition for the project

- Teams of **3–4 students**.
- Each team must assign roles:
    - Parallelization Lead
    - Performance Analyst
    - Code Integrator / Tester
    - Documentation Coordinator

Detailed Requirements per Phase

Phase 1 – Parallel Foundations (Week 6)

Objective: Move from a working serial program to a shared-memory parallel version and evaluate its performance.

Tasks

1. **Baseline Implementation**

- Write a sequential version of your chosen problem in C/C++ (or Python + NumPy).
- Verify correctness and record its runtime on test datasets.

2. **OpenMP Parallelization**
   - Use #pragma omp parallel for & task constructs to parallelize major loops & tasks.
   - Avoid data races; use proper reduction clauses or critical sections.
3. **Performance Profiling**
   - Measure T_S and T_P for 1, 2, 4, 8 threads.
   - Compute Speedup (S = T_S / T_P) and Efficiency (E = S / p).
   - Interpret results using Amdahl's Law and discuss scalability limits.
4. **Memory Hierarchy**
   - Run cache-sensitive test (e.g., stride or blocking).
   - Comment on cache misses / locality using perf stat or gprof.

Deliverables

- Source code repository (GitHub link)
- Short report (4–6 pages PDF) containing:
  1. Problem statement and baseline design
  2. Parallelization approach and OpenMP directives
  3. Performance table + Speedup/Efficiency plots
  4. Discussion of Amdahl's and Gustafson's analysis
  5. Reflection: what bottlenecks remain?
- 2-min demo video (showing problem definition, execution and speedup graph)

Phase 1 Deadline and Structure

- **Due Date: Sat. 25 October 2025, 11:59 PM**
  - /src (folder with source code)
  - /data (test inputs if any)
  - /report.pdf
  - /plots (Speedup vs Threads, Efficiency vs Threads)

Late Policy (applies to the 3 phases): –10% per day (up to 2 days), no submissions after two days.

Expected Learning Outcomes

By completing Phase 1, you will be able to:

- Parallelize a sequential program using OpenMP directives.
- Measure speedup and efficiency and apply Amdahl and Gustafson laws.
- Analyze cache and memory hierarchy effects on performance.
- Produce professional performance reports with data visualization.

Evaluation Rubric

Criterion	Weight (Phase 1)	Performance Indicators
Problem definition & baseline correctness	15%	Clear description + verified serial output
Parallel design (OpenMP implementation)	25%	Proper parallel regions and thread safety
Performance analysis (speedup, efficiency)	25%	Accurate timing + insightful discussion

Use of profiling tools (cache, CPU)	10%	Evidence of measurements with perf/gprof
Report clarity and presentation	15%	Organized writing + well-labeled graphs (Plots of Speedup vs Threads, Efficiency vs Threads)
Demo Clarity	10%	Clear outline and takeaways

Phase 2 – Scaling Across Nodes (Weeks 7–10)
Objective: Transform the shared-memory version into a distributed one.
Tasks

- Implement communication using MPI (point-to-point and collectives).
- For compute-heavy tasks, implement a CUDA kernel or hybrid CPU–GPU version.
- Measure latency vs bandwidth for data exchange.
- Produce scaling plots (Weak & Strong Scaling).

Deliverables

- MPI/CUDA code + runtime scripts
- Report (4–6 pages) including topology diagram and analysis of communication overheads.
- 2-minute video for the main findings & challenges

Phase 3 – Resilience and Big-Data Integration (Weeks 11–14)
Objective: Integrate fault tolerance and data-parallel frameworks.
Tasks

- Wrap core algorithm in a **gRPC service** or a **Spark/Flink (or any other big data tools) pipeline**.
- Simulate node failures and show system recovery.
- Evaluate throughput and latency under streaming loads.
- Present final demo + report (4–6 pages).