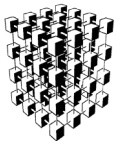

CS451: **Introduction to Parallel and Distributed Computing**
Spring 2026

This document describes what your course project must fulfill and provides you with choices for your project topic. You're also welcome to explore variations of these project ideas, and to propose project ideas that aren't on this list. Please discuss variations or ideas with the instructor before starting work on a project.

Project Design: Before starting to work on the project, you must submit a *Project Design* by the deadline that's shown on the course calendar. The design consists of a PDF file that describes: (a) What you will build and how you will build it, (b) What programming language(s) and dependencies (e.g., tools, libraries) you plan to use, (c) How your implementation will be structured (as modules), (d) At least 3 examples of inputs that your implementation should be able to handle, and the output that it's expected to produce.

Project Outcomes: Your project submission (at the *Progress Update* or *final submission* stages) should consist of the following:

1. **Documentation** (up to 5 pages, as PDF)—This should describe: (a) What the project does, (b) How to use your implementation, (c) Limitations of your implementation, (d) What you'd add if you had 3 more months to work on this project.
2. **Working code**—Unless the project requires a specific programming language, you're free to choose any language you like, but your implementation must run on FABRIC and make use of JupyterHub. Bonus marks (5%) will be given for the use (or creation of) notebook widgets that demonstrate your work.
3. **Recorded demo** (as mp4, lasting at least 2 minutes, and showing your project being invoked on at least 3 test inputs).

Project ideas—pick one of the items below:

a) Build an **automated trading platform** that reads market data (consisting of a stream of asset prices) and produces market actions (consisting of buy/sell actions). You can choose the input and output formats for your system. The platform can trade whatever you choose—stocks, bonds, cryptocurrency, etc—but it must make use of distributed resources on FABRIC, and demonstrate how performance is affected by scaling (to use more resources). For inspiration, you could read about Cisco's trading infrastructure study and the FIX protocol.

b) Build a **cluster job manager** from scratch. This manager maps workload descriptions to cluster resources (such as VMs on FABRIC). For ideas, see HT-Condor and Slurm. Jobs and cluster resources can be described in a format of your choosing—such as YAML or JSON files. You need to devise a scheduling policy (that determines resource quotas for jobs and resource sharing—which can include load balancing), and decide how to handle (and communicate to the user) the occurrence

of errors (such as resource depletion, crashed or unresponsive nodes, and exceeded deadlines).

c) Build a **large-scale Pi calculator** that produces a high-precision evaluation of Pi by using FABRIC resources. For an example that used a commercial cloud, see GCP's description of calculating 100 trillion digits of Pi. (As of this writing, the current record consists of 300 trillion digits of Pi.)

d) *(Open-ended, theory-friendly project)* **Extend hashtray** with a new concurrent data structure and demonstrate its use in an application of your choosing. This project requires a good working background in data structure theory and practice. *Your implementation must be carried out in C,* and the project fluency in the C language and its tool-chain.

e) *(Open-ended, systems-oriented project)* **Build on or extend libcompart** to demonstrate the partitioning of open-source software, or to add new capabilities to libcompart. Such capabilities could include: supporting additional containment back-ends (e.g., Docker, K8s, jails, Capsicum, pledge, etc), supporting more flexible application topologies, or adding synchronization primitives (since currently libcompart only supports blocking semantics). *Your implementation must be carried out in C,* and the project requires a good working background in systems and fluency in the C language and its tool-chain.

f) *(Research-oriented)* **Extend Patchwork** to carry out distributed data storage and analysis (to complement its distributed data gathering, and accelerate its analysis time). This project requires strong systems skills.

g) *(Research-oriented)* **Visualize detector data** using distributed and parallel techniques to process hundreds-of-gigabits-per-second and produce a visualization in near-real-time. This project requires strong systems skills.

h) *(Research-oriented)* **Analyze detector data** using distributed and parallel techniques to process hundreds-of-gigabits-per-second and infer particle collisions in near-real-time. This project requires strong systems skills.