Scalable AI: Bridging Theory, Understanding, and Practice

EE 290 / 194 · Spring 2026

Instructors	Prof. Anant Sahai; Prof. Jiantao Jiao)
Teaching staff	Haocheng Xi (Teaching); Paul Zhou (Teaching); Venkat Srinivasan (NVIDIA infrastructure and technology)
Lecture	Tuesdays & Thursdays, 9:30am–11:00am (521 Cory Hall)
Office hours	Tuesdays, 11:00am–12:00pm (immediately after lecture; location announced on Ed)
Assignment checkoffs	Checkoff slots by sign-up
Resources	Website: `https://scalable-ai.eecs.berkeley.edu/` Forum (Ed): `https://edstem.org/us/courses/93630` Gradescope: `4DDRWY`
Questions	**Post on Ed.** We do *not* use Slack or other channels for course Q&A. Please only use Ed for all discussions.
Compute	Groups of 4–6. 1 H100 node (8×H100) per group (8 total nodes on GCP). Each group is assigned a single static external IP address for SSH'ing in and working.

Course Description

The central inquiry of this course is: *How do we build, train, and deploy large-scale AI systems by treating them as full-stack engineered artifacts, where hardware constraints, software stacks, and optimization dynamics jointly determine model behavior and performance?*

This course examines the principles required to build, train, and deploy large-scale AI models. We treat large-scale AI as an **end-to-end engineering discipline**, where a model is a computational graph that must be trained, specialized, evaluated, deployed, monitored, and iterated on—all under hard constraints from hardware, data, and serving economics.

At modern scales, large language models are constrained as much by hardware and systems realities as by algorithms. Matrix multiplication, memory bandwidth, interconnect topology, numerical precision, and optimizer stability define the feasible design space and shape both training and inference behavior. Architectural choices (dense vs. sparse, MoE, long-context mechanisms, parallelism strategies) and optimizer choices (including emerging optimizers like Muon and SOAP) directly determine convergence, efficiency, and deployability.

This course follows the lifecycle end to end:

Architecture → Pre-training → Post-training → Efficient inference → Applications → Research greenfields.

Throughout, we will work directly with the NVIDIA ecosystem and the software stack that underpins modern AI infrastructure. We will learn how the world built its AI infrastructure on NVIDIA—and what we can do to make it better.

Course questionnaire (Week 1–2). We will release a short onboarding questionnaire in the **first lecture (Jan 20)**. We will finalize enrollment/compute allocations and related decisions by the **end of the second week (Jan 29)**.

Communication policy (Ed only). All course questions **must** be posted on **Ed** (https://edstem.org/us/courses/93630). We answer, announce, and archive on Ed only.

Commitment and grading basis. This course requires sustained weekly engagement (group work, compute usage, and in-person checkoffs). **Letter grade only:** you may not take this course Satisfactory/Non-Satisfactory. Additionally, because the work is continuous and group-dependent, you should not enroll if you anticipate major recurring time commitments that would interfere with course participation (e.g., intensive interviewing or other major obligations).

Grading and Policies

This course has **no exams**. Evaluation is based on assignments, a semester-long research project, scribing, and participation.

Component	Weight
Research project	50%
Assignments	35%
Scribing (2 lectures per student)	5%
Attendance & participation	10%

Required components. Failing any one of them fails the class. You must do every component satisfactorily or else you will fail the class. This includes: the research project, all assignments, scribing, and participation.

The assignments consist of a code and report submission, as well as a presentation component where you will explain and defend your results.

The research report consists of a technical evaluation component, where the teaching staff and your peers will evaluate the quality of your work. In addition, there will be a community impact component.

Submission platform (Gradescope). All written submissions and code submissions will be submitted through **Gradescope** (4DDRWY), unless explicitly stated otherwise. Oral presentations require sign-up and in-person attendance to deliver the presentation.

Deadlines

Assignments

Assign.	Topic	Release	Due
A1	Parallelism Assignment	Jan 27	Feb 10
A2	Pre-Training Assignment	Feb 10	Feb 24
A3	Post-Training Assignment	Feb 24	Mar 10
A4	Inference and Serving Assignment	Mar 17	Apr 7
A5	Applications Assignment	Apr 7	Apr 21

Research Project

Milestone	Due	Notes
Research directions released	Jan 22	Posted on course site; brainstorming begins.
Group formation	Jan 29	Groups of 4–6; node assigned.
Hypothesis statement v1	Feb 12	One paragraph hypothesis + success metric.
Project proposal v2	Feb 26	2–3 pages: method, evaluation plan, risks.
Midterm check-in	Mar 19	Milestone report; check-in meetings scheduled around this date.
Draft report	Apr 26	Draft report submitted for peer review (Gradescope).
Final presentations	Apr 28/30	Short talks in the week **prior** to RRR week.
Peer review due	May 1	Peer reviews submitted (Gradescope).
Poster session	May 5	RRR week poster session.
Final deliverables	May 10	Code + final report.
Impact update close	May 15	Proof of merge/acceptance or submission to ArXiv, etc.

Infrastructure

This course uses shared GPU infrastructure. If you have questions about access, quotas, networking, or failures, **post on Ed** (include timestamps, job IDs, logs, and a short repro description).

- **Cluster and allocation:** We will use **8 total nodes on GCP**. Each group receives **one** H100 node (8×H100) for the semester. You will be given a static IP address that you can use to log into the node.
- **NVIDIA compute support:** Please route requests through **Ed** so issues are tracked and shared.
- **Multi-node experiments:** If your project requires more than one node, you must coordinate with another team to **share nodes** for a bounded window of time. Since each node has its own external IP, multi-node work may require coordinating host files and access rules across *multiple* team IPs.
  - Plan early and post on Ed to request staff help if infrastructure help is needed.
  - Agree on a schedule with the other team(s) to avoid interference.

- Keep runs reproducible (versioned code, pinned configs, logged seeds, and saved artifacts) so you can hop around if the need arises.