CS140: Parallel Scientific Computing

2026 Winter Class Introduction

Tao Yang, UCSB

CS 140 Course Information

- **Instructor: Tao Yang (tyang at cs.ucsb).**
  - Office Hours: MW after class -11:45pm
- **TA:** TBD
- **References (no required textbooks)**
  - Online references.
- **Class slides/online references:**
  - http://www.cs.ucsb.edu/~tyang_class/140w26

Course introduction

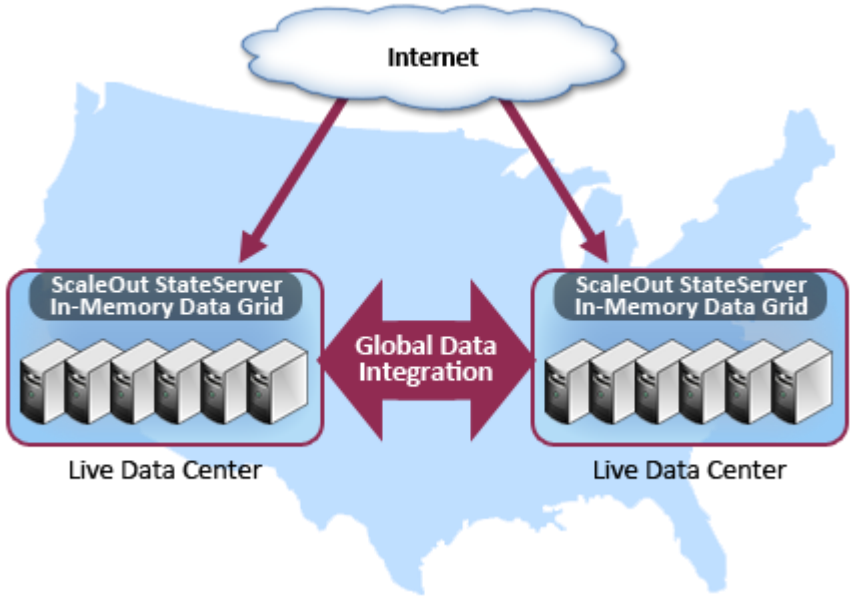
- Why parallel processing?
- Why writing (fast) parallel programs is hard
- Class Information

All computers use parallel computing

- Web+cloud+cluster computing

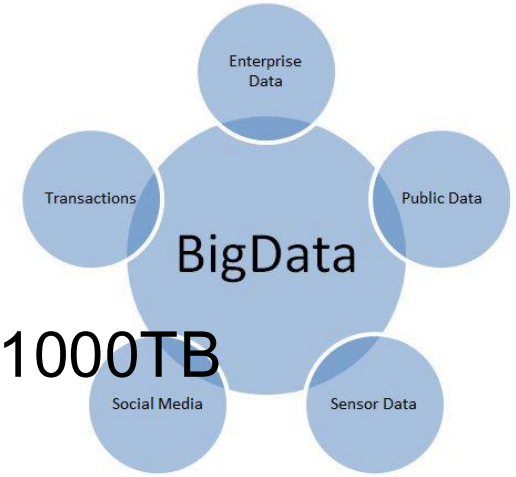
Big corporate computing

- Enterprise computing
- Personal computing with many cores: Desktops, laptops, phones



- Current fastest machines in the world
  - Up-to-date list at www.top500.org
  - Top one has 2,746 Pflop/s using 11 millions of CPU/GPU cores

Big Data Drives Computing Need



- **Web search/ads** (Google, Bing)
  - 10B-100B pages crawled -> indexing 500-1000TB /day
  - 10B+ queries+pageviews /day → 100+ TB log
- **Social media**
  - Facebook: Billions of content items shared. 500TB+ data ingested/day
  - Youtube: A few billion views/day. Millions of TB.
- **AI applications**
  - Self-driving cars with neural machine learning
  - Large language models for question answering
  - Training requires weeks/months of computing time

Performance numbers used in scalable computing with big data

FLOPS = flop/s = floating point operations per second

Mega	Mflop/s = 10^6 flop/sec	**Mbyte** = 2^{20} ~ 10^6 bytes
Giga	Gflop/s = 10^9 flop/sec	**Gbyte** = 2^{30} ~ 10^9 bytes
Tera	Tflop/s = 10^{12} flop/sec	**Tbyte** = 2^{40} ~ 10^{12} bytes
Peta	Pflop/s = 10^{15} flop/sec	Pbyte = 2^{50} ~ 10^{15} bytes
Exa	Eflop/s = 10^{18} flop/sec	Ebyte = 2^{60} ~ 10^{18} bytes
Zetta	Zflop/s = 10^{21} flop/sec	Zbyte = 2^{70} ~ 10^{21} bytes
Yotta	Yflop/s = 10^{24} flop/sec	Ybyte = 2^{80} ~ 10^{24} bytes

How fast can Intel single-core do in FLOPS?

How long does it take per each instruction?

1-100 Gflop/s. Nanosecond

How much parallel computing resource is needed for ChatGPT?

- **$10s - 200 million project to build a flagship AI model**
  - OpenAI's pre-training for GPT-4 with 300B parameters
    - a dataset of 13 trillion tokens
    - 2.15 x 10^{25} FLOPS
    - **25,000 NVIDIA A100s for 90+ days**
    - $63 million on A100s and ~ $22 million on H100s

- **Cost of online inference**
  - $18,000 purchase or $2 hourly rental for NVIDIA H100
  - 20 requests per second @500 tokens/request
  - A single ChatGPT call uses one plastic bottle of water for cooling

Architecture Trend

Can a single high speed core be used instead of many cores?

- Chip density is continuing increase ~2x every 2 years
- Clock speed is not

Use one machine with many cores and big shared memory?

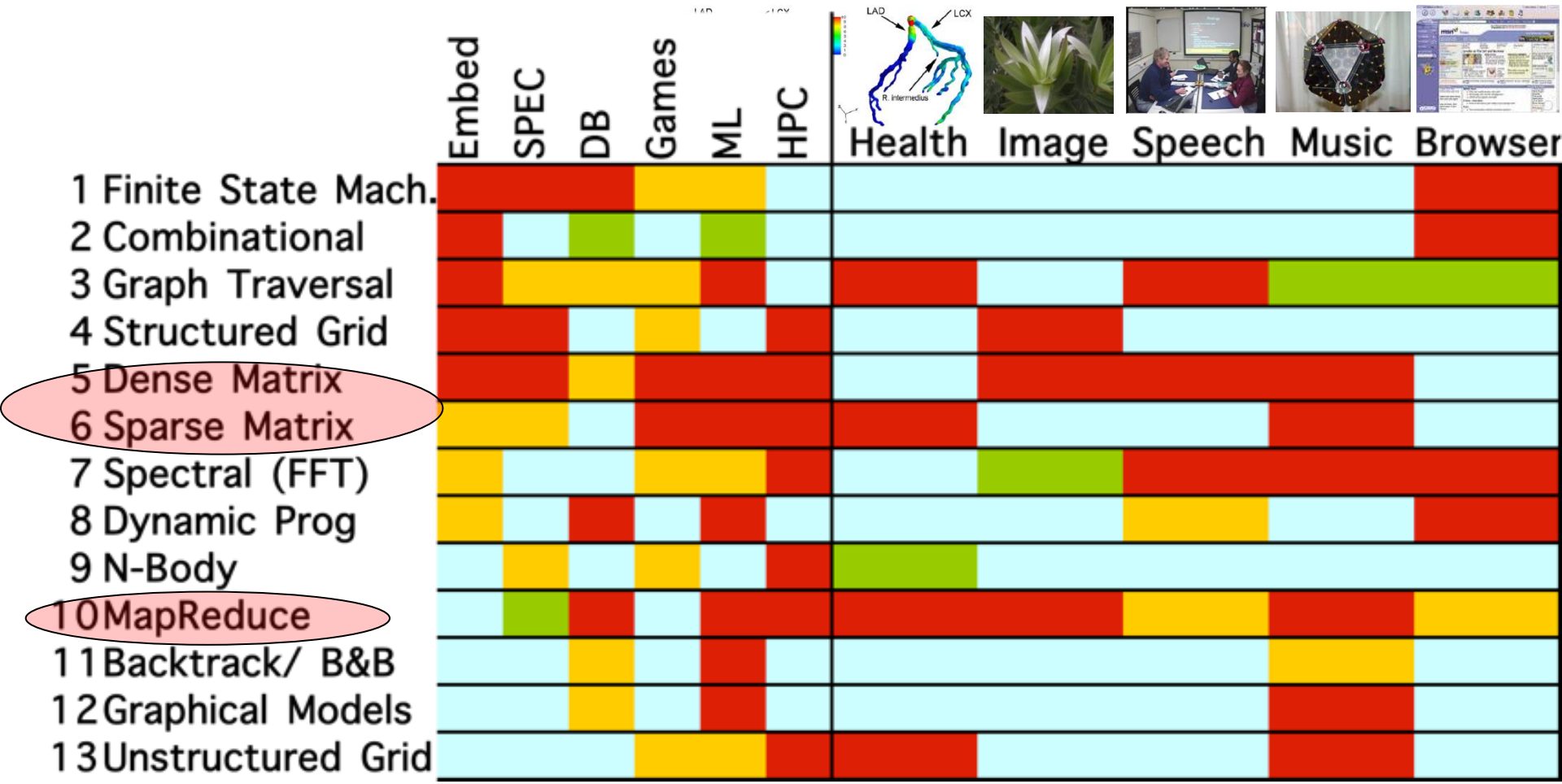
- Technology trends against increasing memory per core. Memory performance is not keeping pace
- # of cores increases steadily to 144 Intel Xeon. 192 AMD EPYC
- *Use a distributed architecture for many high-end computing*

- **Will all programmers have to be parallel programmers?**
  - Many applications require parallel/distributed programming
  - Parallelized libraries and compilers provide core support with limitations

What do compute-intensive applications have in common?

Common Computational Methods

(Red Hot → Blue Cool)



Machine learning applications require extensive matrix multiplication

Basic Scientific Computing Algorithms

- Matrix-vector multiplication
- Matrix-matrix multiplication
- Direct/iterative methods for solving linear equations
  - Gaussian Elimination. Jacobi, Gauss-Seidel.
  - Google web search ranking algorithm
- SGD iterative method for model optimization in machine learning applications
  - CNN &transformer computation for text, image, &audio
  - Time is dominated by Matrix/vector multiplication operations

Focus of CS140

Why writing (fast) parallel programs is hard

- **Finding enough parallelism (Amdahl's Law)**
- **Granularity**
  - Algorithm needs sufficiently large units of work to run fast in parallel (i.e. large granularity), but not so large that there is not enough parallel work
- **Overheads in parallelism with coordination**
  - cost of starting a thread or process
  - cost of accessing data, communicating shared data
  - cost of synchronizing
- **Locality with memory hierarchies (cache, memory, disk)**
- **Load balance wit static and dynamic scheduling**
- **Performance modeling for cost prediction since it is expensive to scale**

Course Objective

In depth understanding of:

- When is parallel computing useful?
- Understanding of parallel computing hardware options
- Overview of programming models (software) and tools and performance analysis
- Parallel algorithms for core computing methods

Course Topics

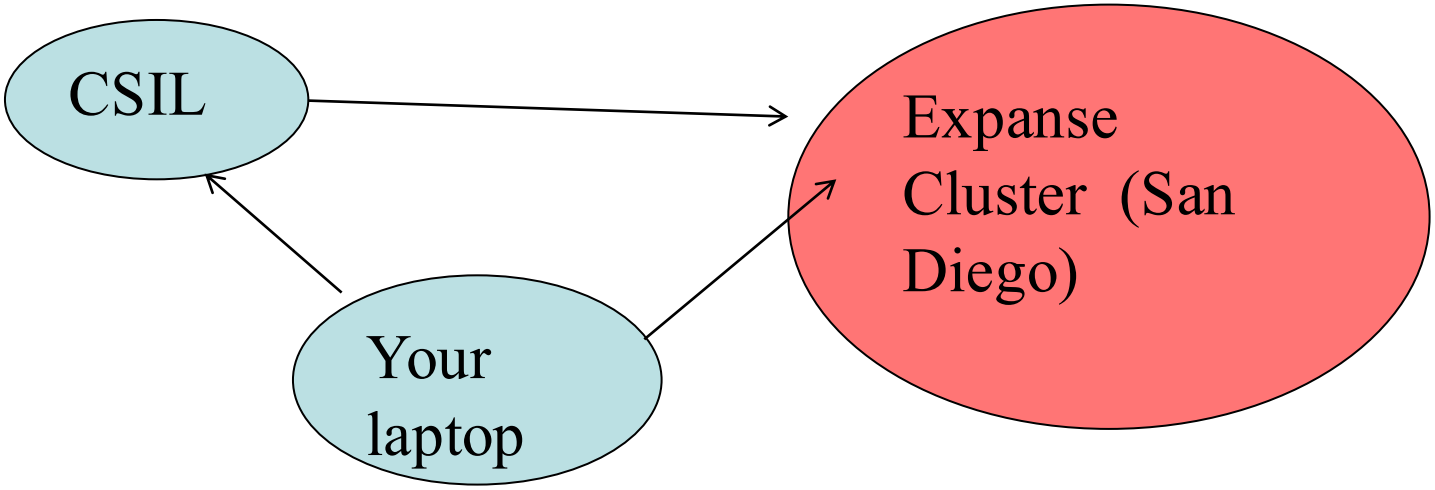
- **High performance computing**
  - Basics of computer architecture, clusters&cloud systems. Storage.
- **Parallel programming models, software/libraries**
  - Task graph computation. SPMD
  - Partitioning and mapping of program/data for shared memory and distributed memory machines
  - MPI, OpenMP, GPU, & Pthreads
  - MapReduce/Spark for data-intensive computing
  - Patterns of parallelism. Optimization techniques for parallelization and performance
- **Core computing algorithms**
  - Multiplication of matrices and vectors
  - Solving linear equations directly and iteratively
  - SGD optimization used in machine learning applications (e.g. ChatGPT)
- **SIMD/cache-aware programming for serial code optimization**

Class Resource: Expanse Cluster

Expanse: 728 standard CPU nodes, 54 GPU nodes and 4 large-memory nodes.

Each standard node: two 64-core AMD EPYC 7742 processors with 256 GB of DDR4 memory

Each GPU node contains four NVIDIA V100s (32 GB SMX2) hosted by dual 20-core Intel Xeon 6248 CPUs



Course Prerequisites and Challenges

- **Data structure**
  - Array/lists. Concept of algorithm complexity
- **Math**
  - Matrices/vectors. Partial derivates
- **Basic computer architecture**
  - CPUs, cache, memory
- **C programming with Linux**
  - Python concepts are used in some illustration
- **Challenges**
  - Difficult for the textbooks to capture the latest technology
    - Parallel computing technology is complex and evolves fast.
  - Reading with self-searching of web material is needed.

Course Workload and Grading

- **Tentative workload and weights**
  - 15% : 5 exercises
  - 32%: 3-4 parallel programming assignments
    - Parallel programming assignments are done by a group of two.
  - 17% midterm exam, 34% final exam.
  - 2% : class participation
  - 1% bonus: answering Piazza questions

If you decide to take this course

- **Account at CSIL**
  - C programming with Pthreads, OpenMP, & MPI
- **Account at Piazza**
  - Discussion group
  - We will send an invitation to your class email
- **Account at GradeScope.com**
  - Submit your homework
  - We will give you a class account join code in Exercise 1
- **Account at Expanse cluster**
  - You open an ACCESS account in https://access-ci.org/
  - We will post a google sheet link at Piazza that you fill your ACCESS account name.
  - We add a shared CPU hour allocation to your account then you can log in login.expanse.sdsc.edu