Parallel Machine Learning and Artificial Intelligence

Dr. Handan Liu

h.liu@northeastern.edu

Northeastern University

Course Introduction

Who am I?

- Dr. Liu, Handan
  - Associate Teaching Professor in Northeastern University
  - Expert on High Performance Computing, Parallel Computing, Distributed (cloud) Computing, Machine Learning and Artificial Intelligence, etc.
  - Have operated and managed the HPC supercomputing cluster of Northeastern for several years
  - Contact:
    - ✔ Email: h.liu@northeastern.edu or
    - ✔ Slack: https://handanliu.slack.com/ or
    - ✔ Linkedin: www.linkedin.com/in/handanliu

TA Introduction

- Chaoyi Yuan yuan.chao@northeastern.edu
- TA hours:
  - Each TA usually holds 2 or more hours every time and twice a week. Two TAs will be held at least 8 hours per week in this course.
  - TA hours is done on Zoom, scheduled by the TAs.
- TA's duty:
  - Quickly respond to your questions through Slack in addition to TA hours.
  - Review and grade your assignments and quizzes.
  - TA won't answer the questions of the assignments before submission. After submission, TAs can answer the questions of assignments and quizzes.
  - Questions should first go to the TAs. If TA can't answer them then TAs will forward the questions to the Professor.

Content

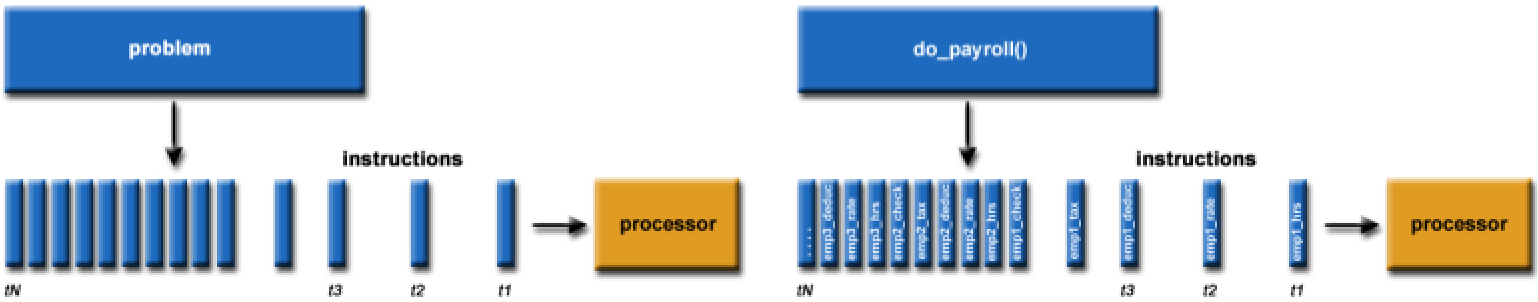
- High Performance Parallel Computing
  - Overview
  - Concepts and Terminology
  - Parallel Memory Architectures
  - Parallel Programming Model
  - Parallel Examples and Exercises

Overview

Serial Computing

Traditionally software has been written for ***serial*** computations:

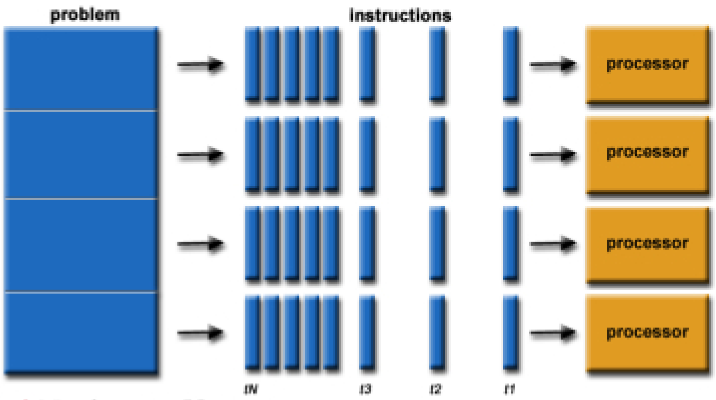
- A problem is broken into a discrete set of instructions
- Instructions are executed sequentially one after another
- Executed on a single processor
- Only one instruction can be executed at any moment in time

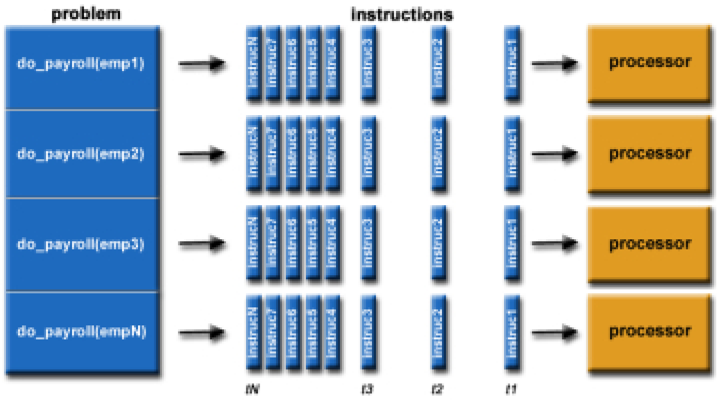


Parallel Computing

In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem:

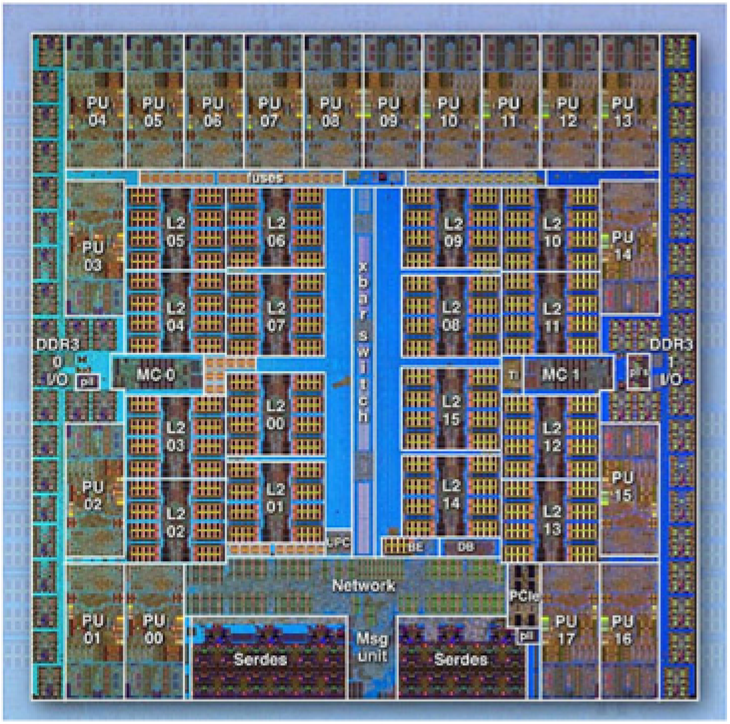
- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different CPUs





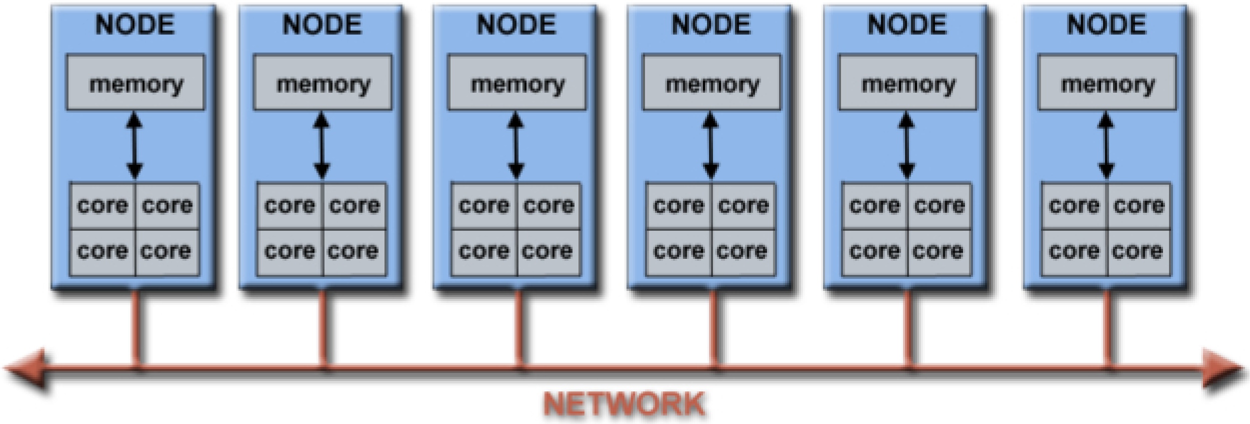
Serial to Parallel

- The computational problem should be able to:
  - Be broken apart into discrete pieces of work that can be solved simultaneously;
  - Execute multiple program instructions at any moment in time;
  - Be solved in less time with multiple compute resources than with a single compute resource.
- The compute resources are typically:
  - A single computer with multiple processors/cores
  - An arbitrary number of such computers connected by a network



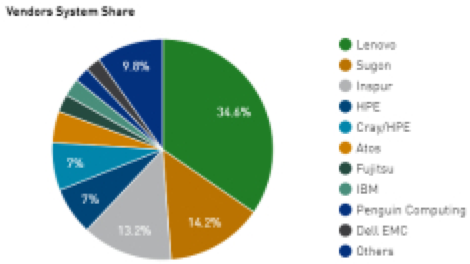
Parallel Computers

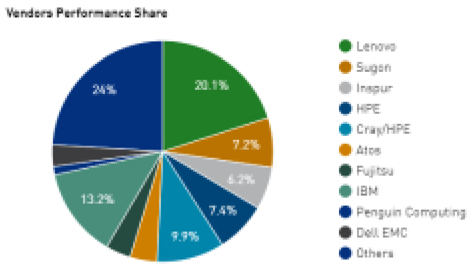
- Multiple functional units (L1 cache, L2 cache, branch, prefetch, decode, floating-point, graphics processing (GPU), integer, etc.)
- Multiple execution units/cores
- Multiple hardware threads



Parallel Computers

Networks connect multiple stand-alone computers (nodes) to make larger parallel computer clusters.





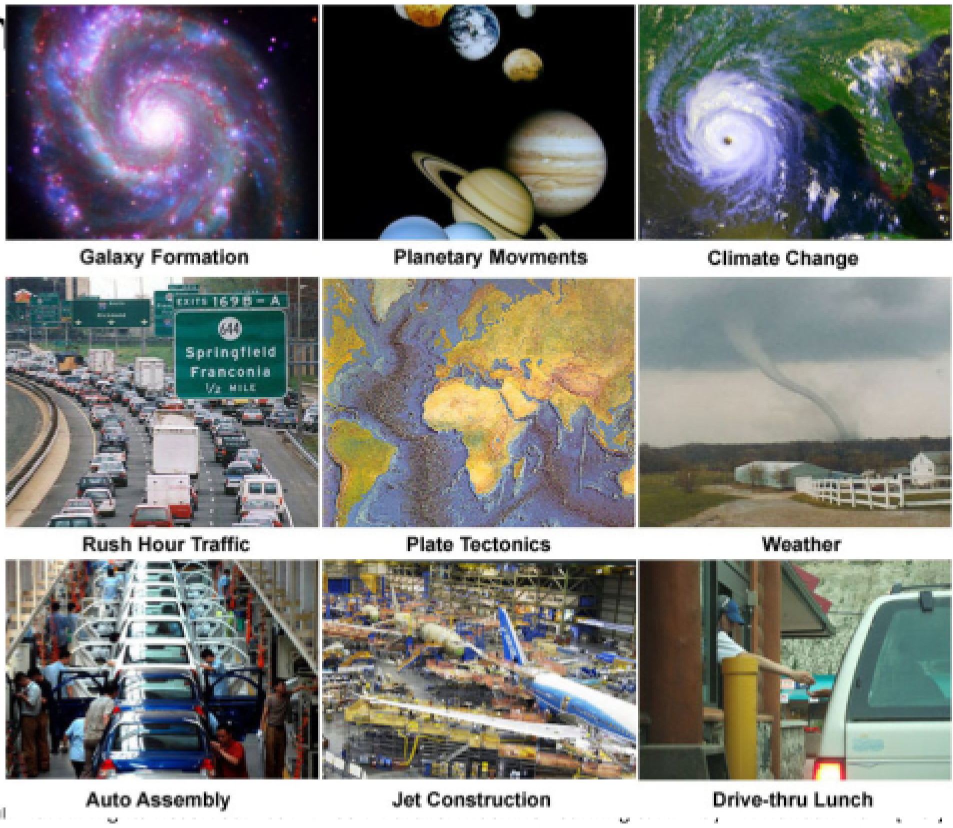
Vendors	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
Lenovo	173	34.6	330,546,526	610,258,957	12,339,312
Sugon	71	14.2	119,286,000	301,609,541	5,182,664
Inspur	66	13.2	102,120,870	219,011,562	2,666,744
HPE	35	7	122,353,488	179,809,284	3,174,784
Cray/HPE	35	7	162,476,312	249,470,362	6,029,800
Atos	23	4.6	64,827,910	103,336,972	2,117,592
Fujitsu	13	2.6	60,537,790	96,397,607	1,589,640
IBM	13	2.6	216,918,859	287,901,962	6,005,272
Penguin Computing	11	2.2	20,033,930	25,247,768	614,136
Dell EMC	11	2.2	52,309,160	89,403,596	1,276,988
Huawei	10	2	14,516,252	26,834,093	380,164
Nvidia	6	1.2	32,400,000	42,848,082	490,784
NEC	4	0.8	7,597,200	12,153,344	273,192
NUDT	3	0.6	66,081,890	108,454,198	5,342,848
IBM / NVIDIA / Mellanox	3	0.6	114,129,000	150,445,771	1,880,424
Intel	2	0.4	7,266,750	12,261,235	159,044
Cray Inc./Hitachi	2	0.4	11,461,000	18,250,444	271,584
Fujitsu / Lenovo / Xenon	1	0.2	1,676,220	3,801,424	87,224
Amazon Web Services	1	0.2	1,926,400	3,981,312	41,472

The majority of the world's large parallel computers (supercomputers) are clusters of hardware produced by a handful of (mostly) well-known vendors.

Courtesy of: Top500.org

Why Use Parallel Com

- In the natural world, many complex, interrelated events are happening at the same time, yet within a temporal sequence.
- Compared to serial computing, parallel computing is much better suited for modeling, simulating and understanding complex, real world phenomena.
- For example, imagine modeling these serially:



Why Use Parallel Computing?

- Computational requirements are ever increasing. There is continual demand for greater computational speed.

  ▯ Limits of single CPU computing: low performance and not enough memory available

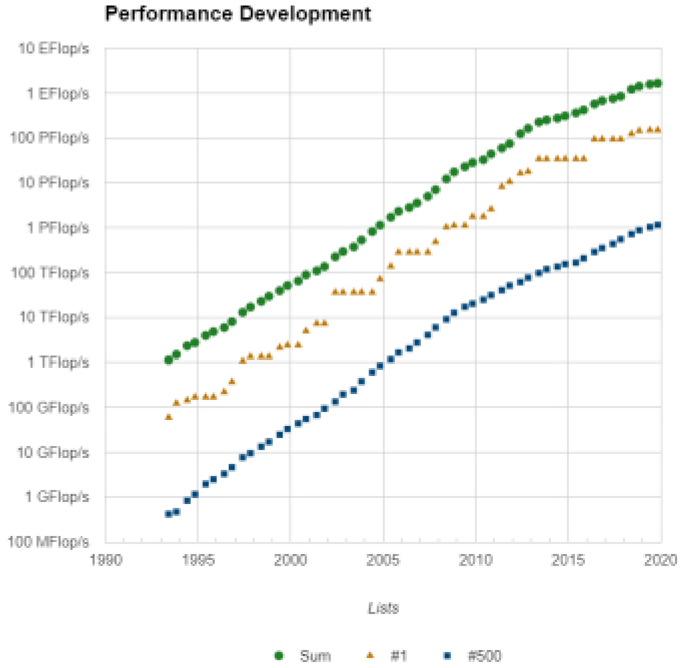
- Obviously, an execution time of 1 year is always **unreasonable**!
- Computations must be completed within a "***reasonable***" time period.

Why Use Parallel Computing?

- Save time / money
- Solve Larger / More Complex problems
  - Example: Web search engines/databases processing millions of transactions every second
- Provide Concurrency
  - Example: Collaborative Networks provide a global venue where people from around the world can meet and conduct work "virtually".
- Take Advantage of Non-local Resources
  - Using compute resources on a wide area network, or even the Internet when local compute resources are scarce or insufficient.
- Make Better Use of Underlying Parallel Hardware

The Future

- During the past 30 years, the trends indicated by ever faster networks, distributed systems, and multi-processor computer architectures (even at the desktop level) clearly show that parallelism is the future of computing.
- In this same time period, there has been a greater than **500,000x** increase in supercomputer performance, with no end currently in sight.
- ***The race is already on for Exascale Computing!***
    - Exaflop = 10^{18} calculations per second



Source: [Top500.org](Top500.org)

FLOPS

In computing, floating point operations per second (FLOPS, flops or flop/s) is a measure of computer performance, useful in fields of scientific computations that require floating-point calculations. For such cases it is a more accurate measure than measuring instructions per second.

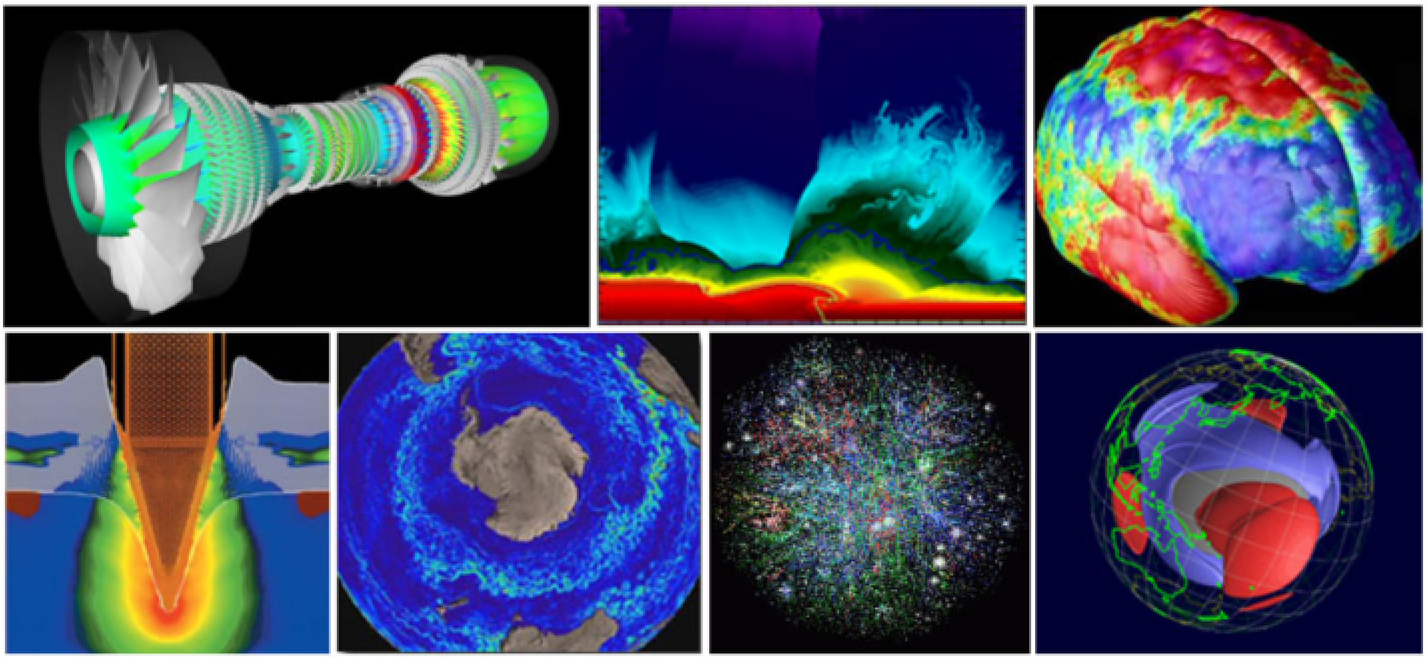
Computer performance		
Name	Unit	Value
kiloFLOPS	KFLOPS	10^3
megaFLOPS	MFLOPS	10^6
gigaFLOPS	GFLOPS	10^9
teraFLOPS	TFLOPS	10^{12}
petaFLOPS	PFLOPS	10^{15}
exaFLOPS	EFLOPS	10^{18}
zettaFLOPS	ZFLOPS	10^{21}
yottaFLOPS	YFLOPS	10^{24}

Who is Using Parallel Computing?

Science and Engineering:

- Historically, parallel computing has been considered to be "the high end of computing", and has been used to model difficult problems in many areas of science and engineering:

  - Atmosphere, Earth, Environment
  - Physics - applied, nuclear, particle, condensed matter, high pressure, fusion, photonics
  - Bioscience, Biotechnology, Genetics
  - Chemistry, Molecular Sciences
  - Geology, Seismology
  - Mechanical Engineering - from prosthetics to spacecraft
  - Electrical Engineering, Circuit Design, Microelectronics
  - Computer Science, Mathematics
  - Defense, Weapons

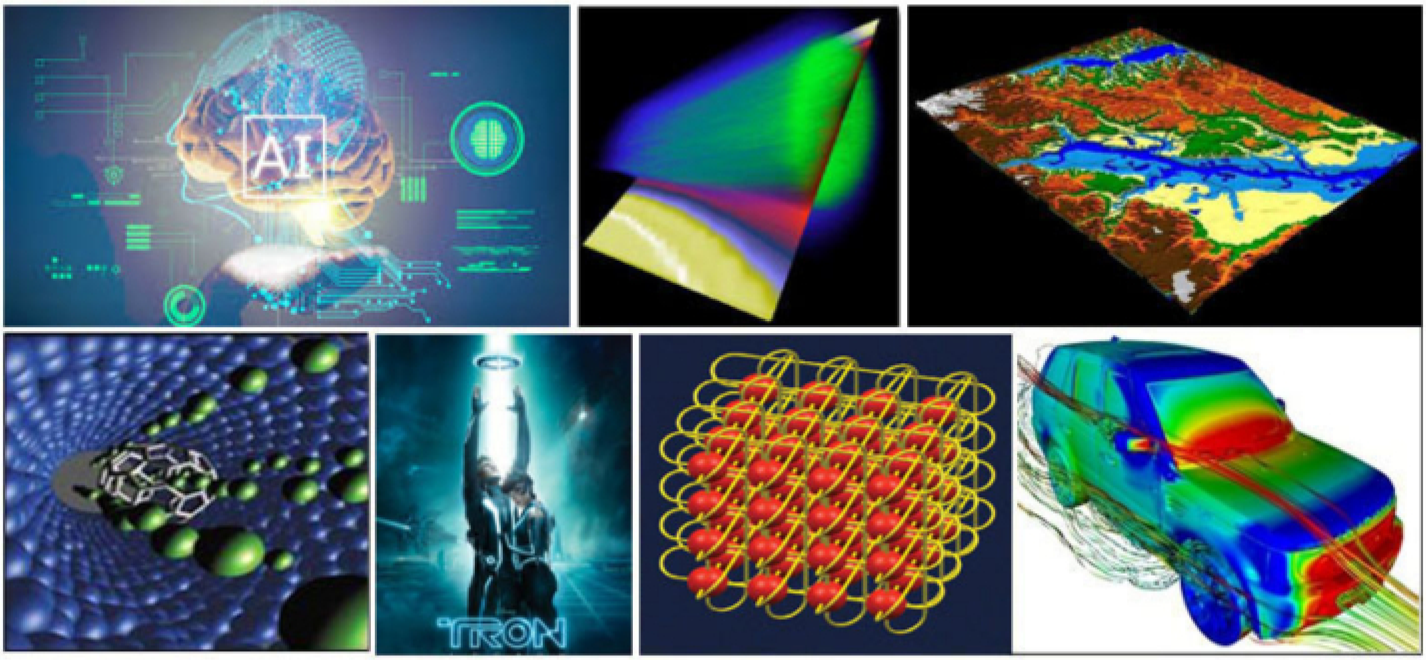


Who is Using Parallel Computing?

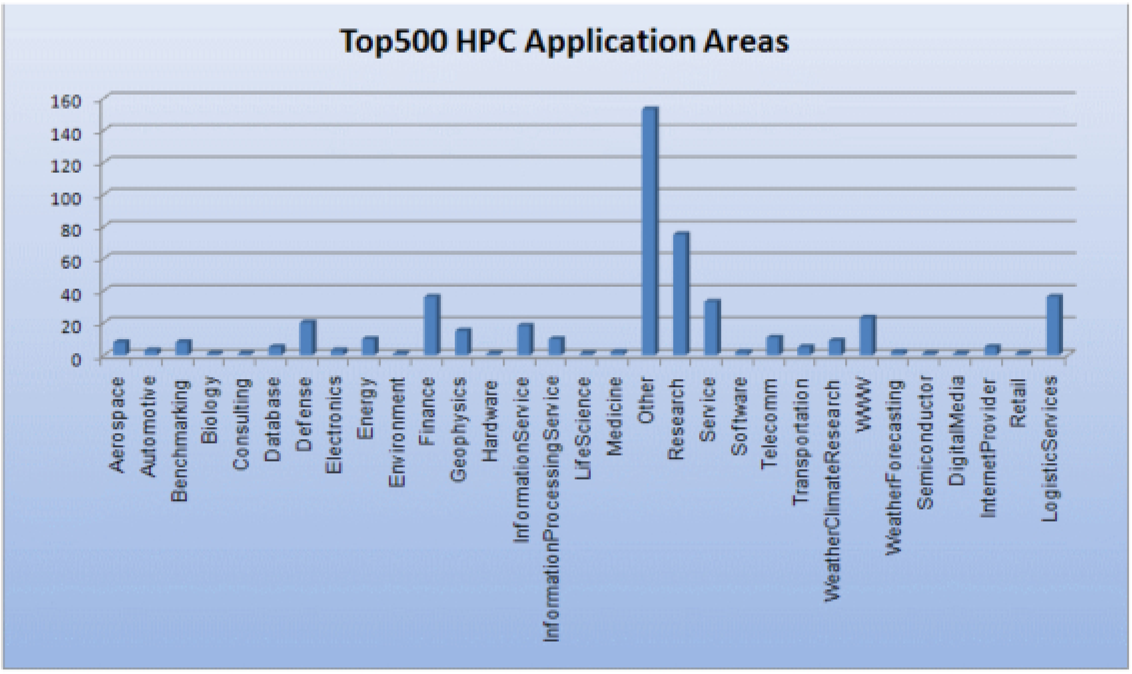
Industrial and Commercial:

- Today, commercial applications provide an equal or greater driving force in the development of faster computers. These applications require the processing of large amounts of data in sophisticated ways. For example:

  - "Big Data", databases, data mining
  - Artificial Intelligence (AI)
  - Web search engines, web-based business services
  - Medical imaging and diagnosis
  - Pharmaceutical design
  - Financial and economic modeling
  - Management of national and multi-national corporations
  - Advanced graphics and virtual reality, particularly in the entertainment industry
  - Networked video and multi-media technologies
  - Oil exploration


AI
TRON

Global Applications:



Source: [Top500.org]

•Stay safe!

•See you next class!

Next Lecture will Continue:

High Performance Parallel Computing

- Overview
- Concepts and Terminology
- Parallel Memory Architectures
- Parallel Programming Model
- Parallel Examples and Exercises