Prof. Thomas Sterling
Pervasive Technology Institute
School of Informatics & Computing
Indiana University

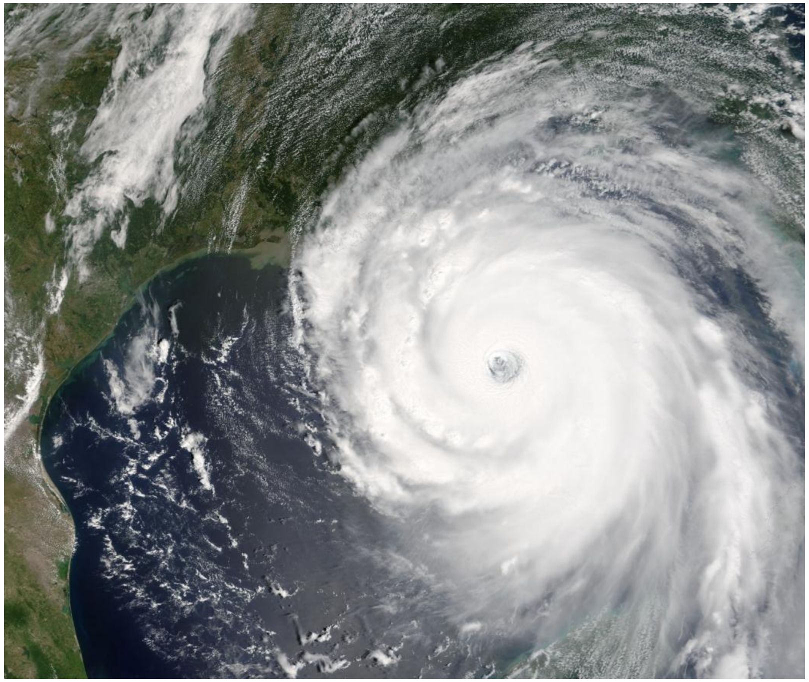
Dr. Steven R. Brandt
Center for Computation & Technology
Louisiana State University

Basics of Supercomputing

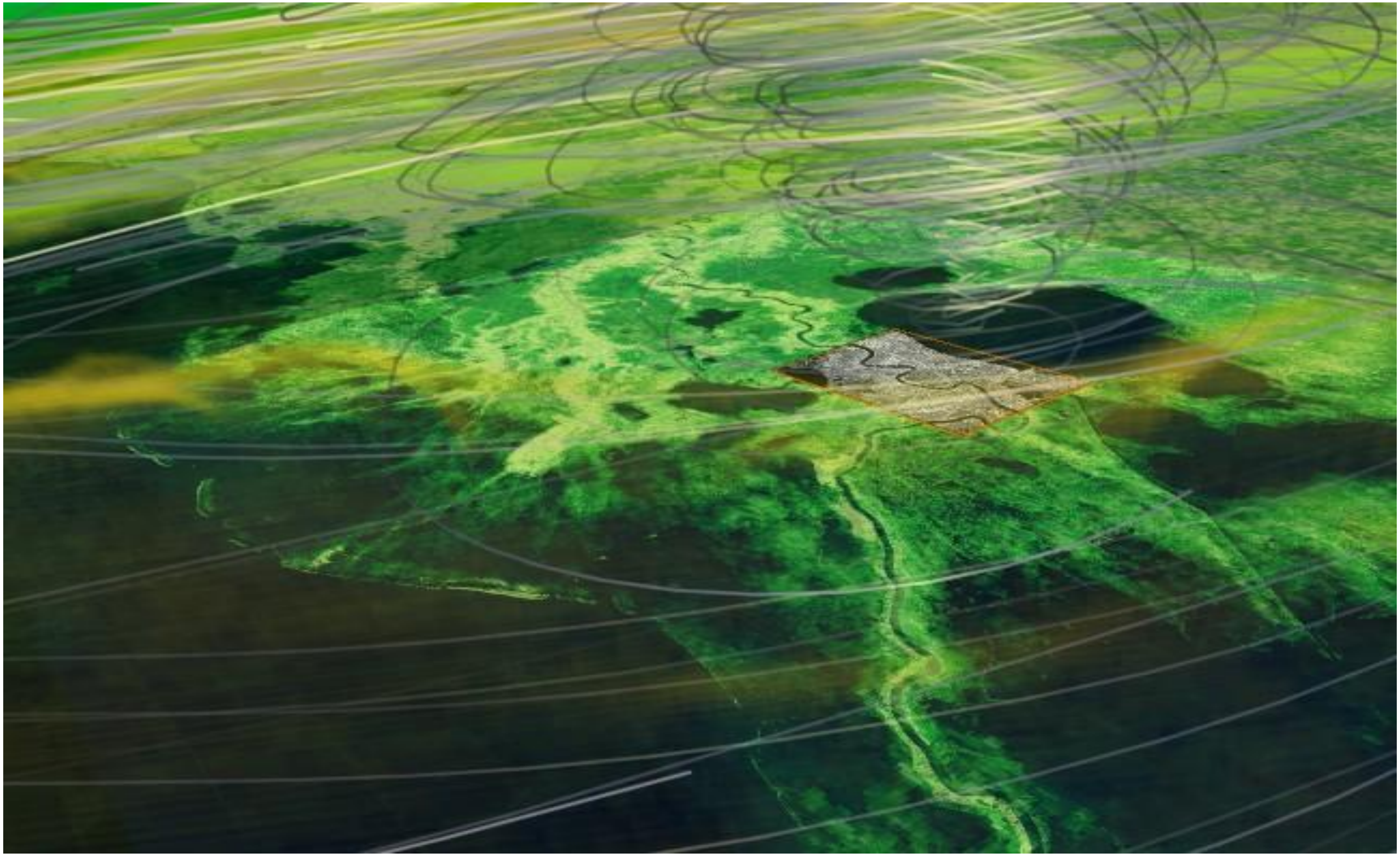
Introduction

Department of Computer Science
Louisiana State University

Devastation from Hurricane Katrina



Simulating Katrina



Goals of the Tutorial

- A first overview of the entire field of HPC
- Accessible by non-experts, first timers
- Basic concepts that govern the capability and effectiveness of supercomputers
- Techniques and methods for applying HPC systems
- Tools and environments that facilitate effective application of supercomputers
- Performance measurement methods, benchmarks, and metrics
- Practical real-world knowledge about the HPC community

New Fastest Computer in the World

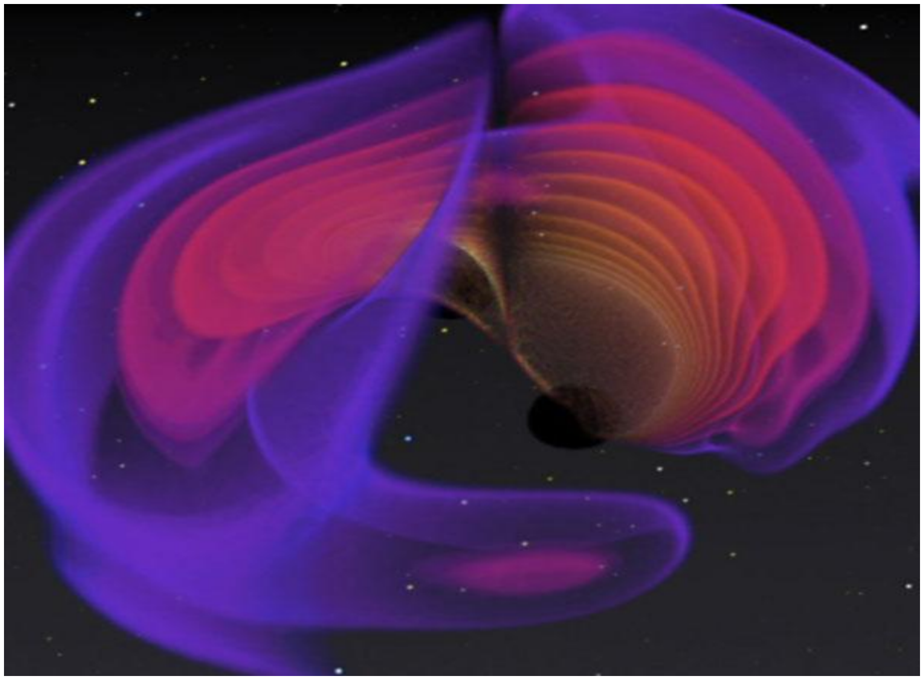
Definitions of "Supercomputer"

Supercomputer: A computing system exhibiting high-end performance capabilities and resource capacities within practical constraints of technology, cost, power, and reliability. *Thomas Sterling, 2007*

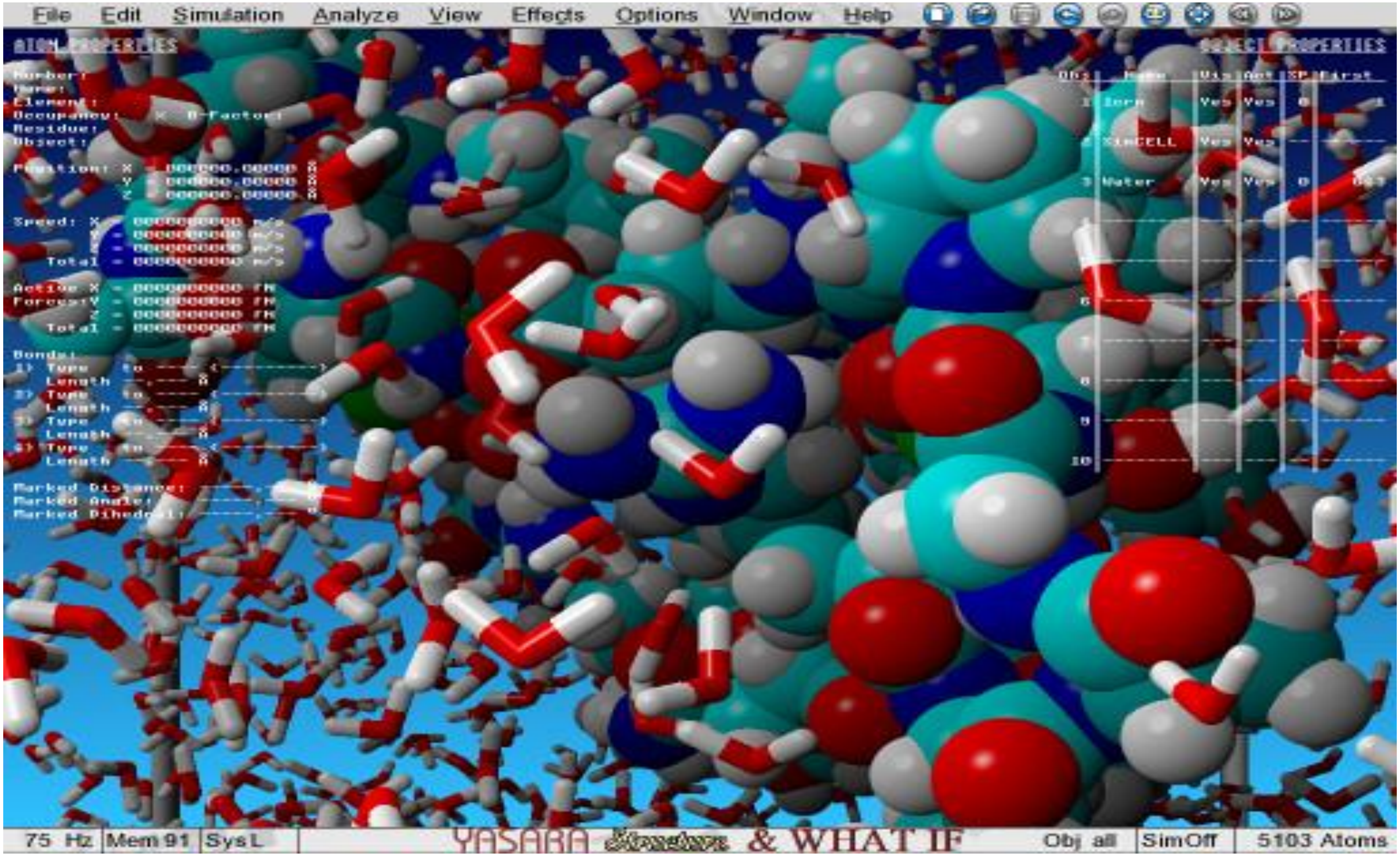
Supercomputer: a large very fast mainframe used especially for scientific computations. *Merriam-Webster Online*

Supercomputer: any of a class of extremely powerful computers. The term is commonly applied to the fastest high-performance systems available at any given time. Such computers are used primarily for scientific and engineering work requiring exceedingly high-speed computations. *Encyclopedia Britannica Online*

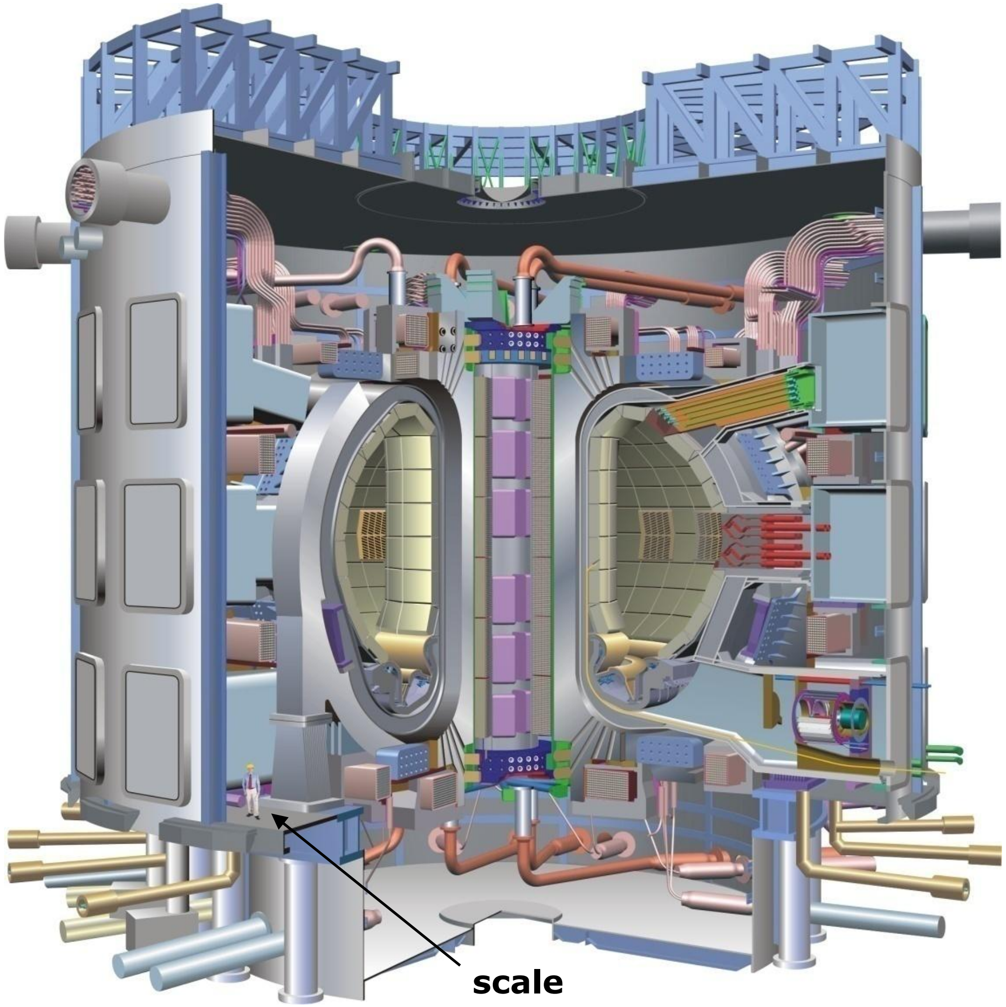
Cosmology



Molecular Dynamics



The U.S. is an official partner in ITER



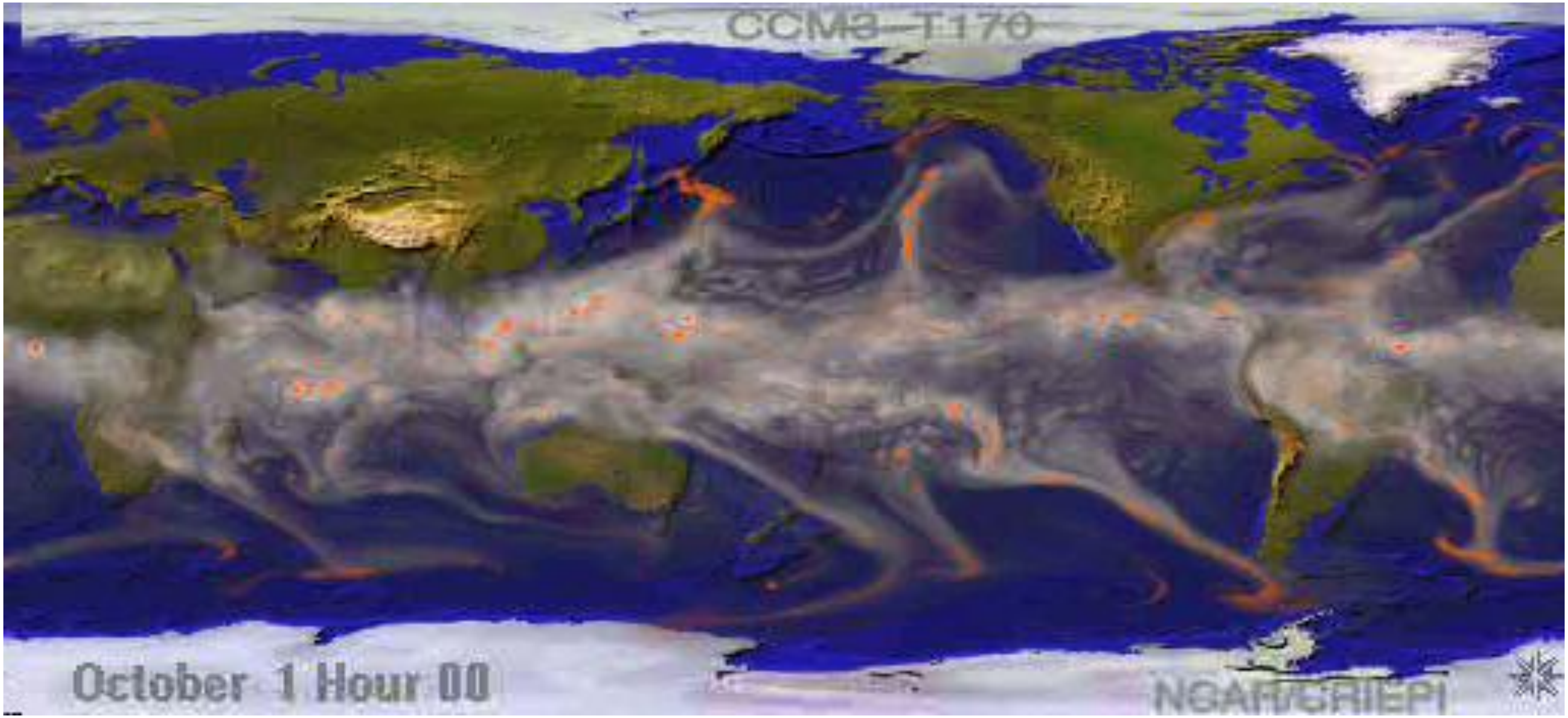
International
Thermonuclear
Experimental
Reactor

- European Union
- Japan
- United States
- Russia
- Korea
- China

- 500 MW fusion output
- Cost: $5-10 B
- To begin operation in 2015

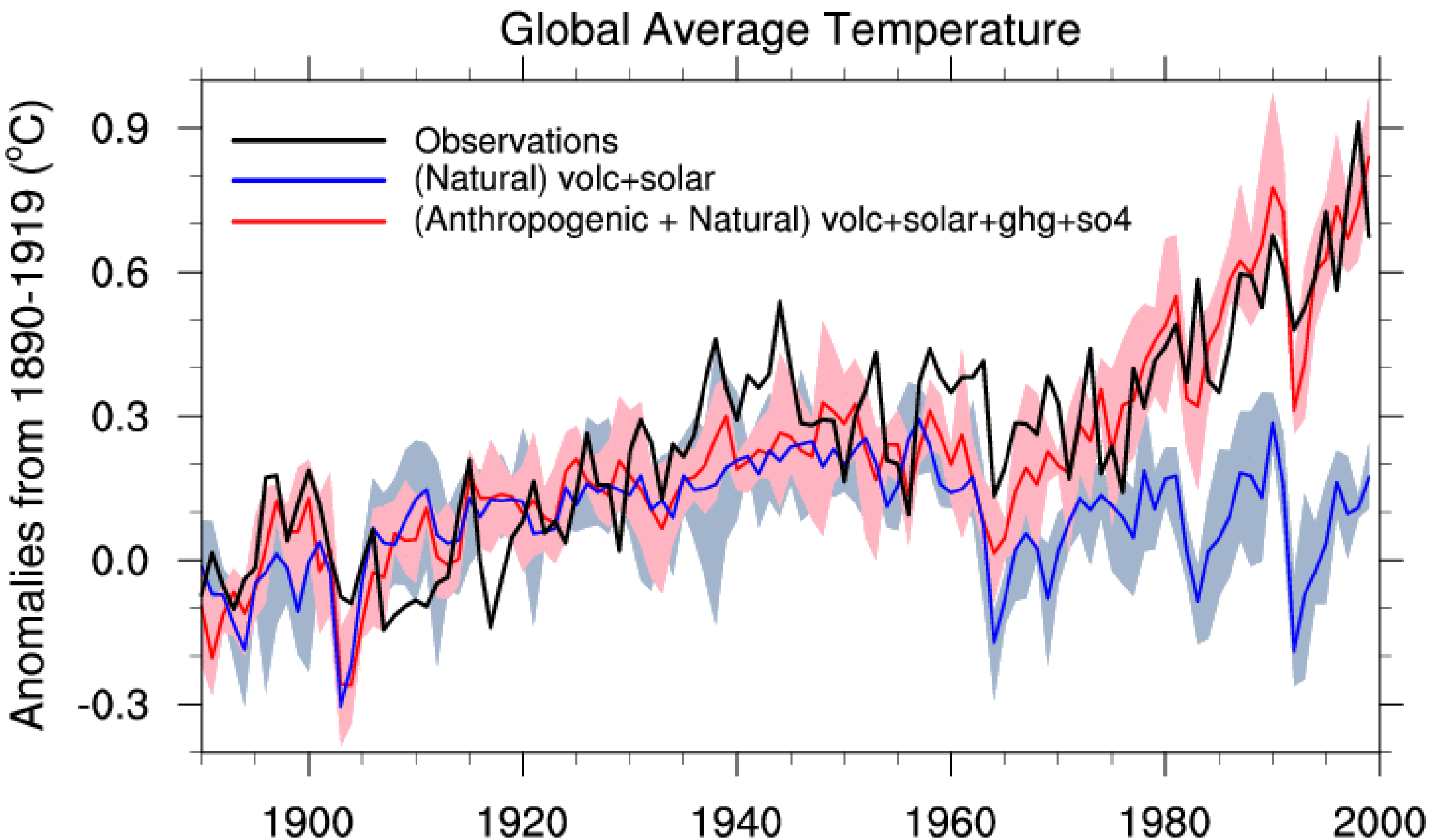
Example of Global Climate Model Simulation

Precipitable Water (gray scale) and Precipitation Rate (orange)

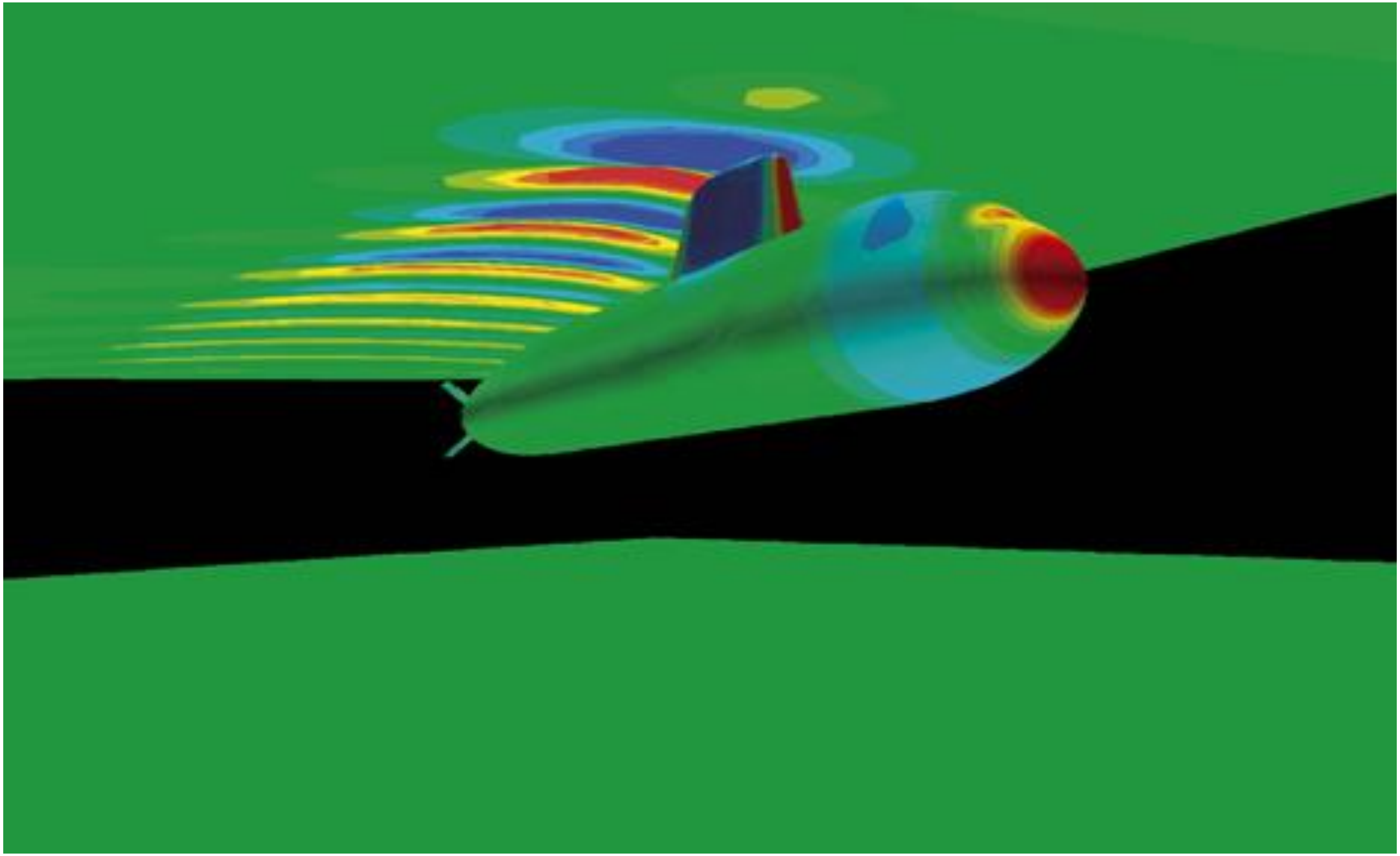


Animation courtesy of NCAR SCD Visualization and Enabling Technologies Section

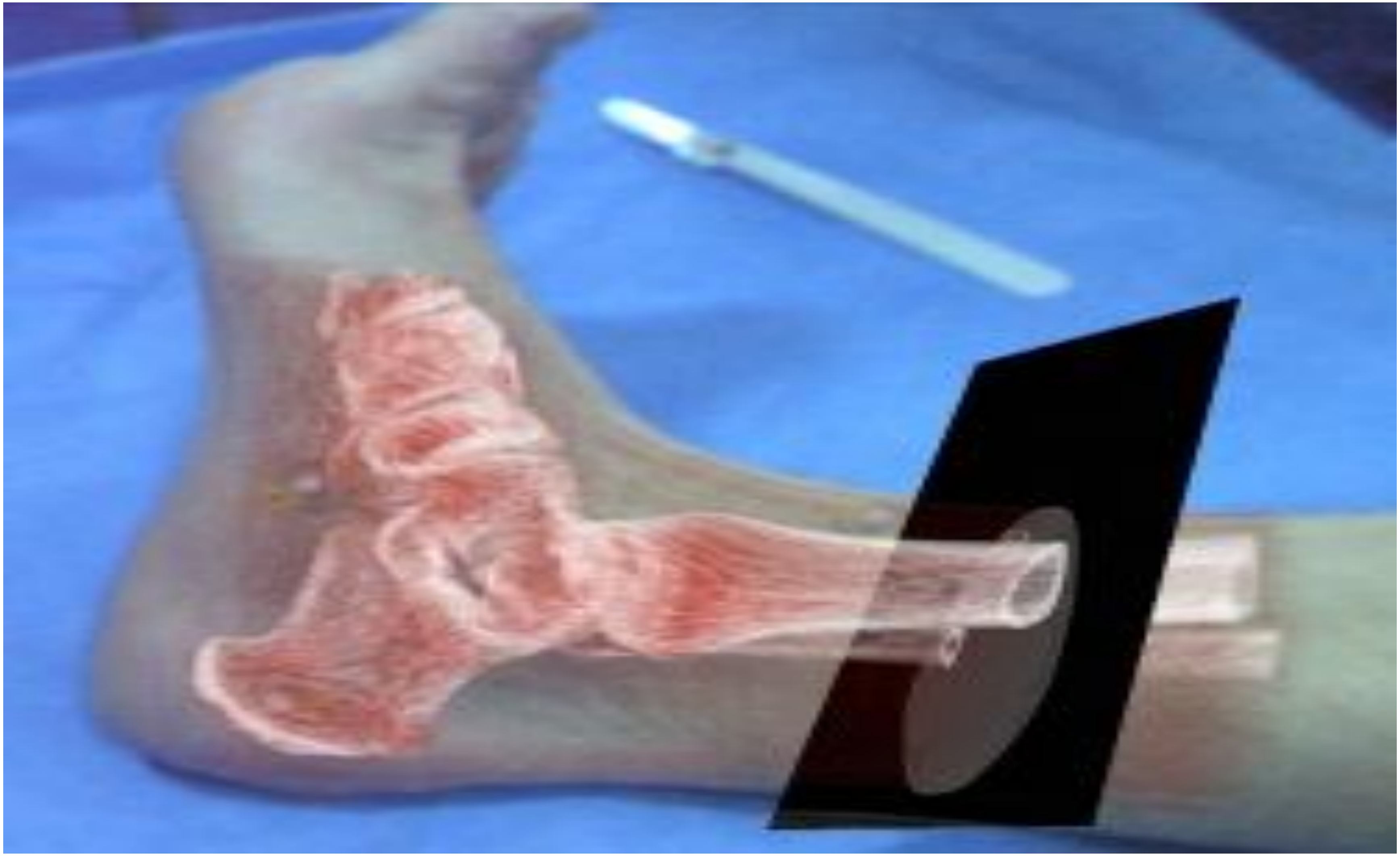
Observations: 20th Century Warming
Model Solutions with Human Forcing



Computational Fluid Dynamics



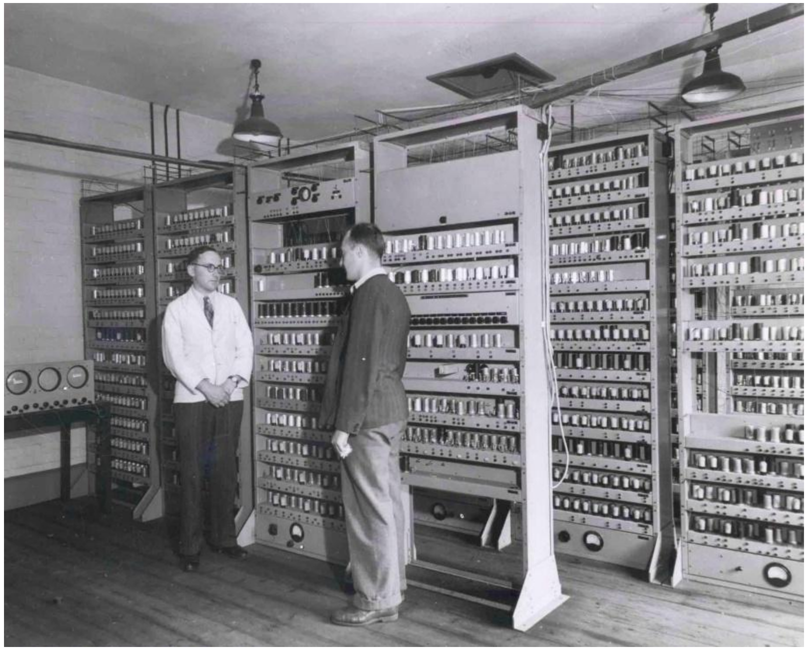
Healthcare



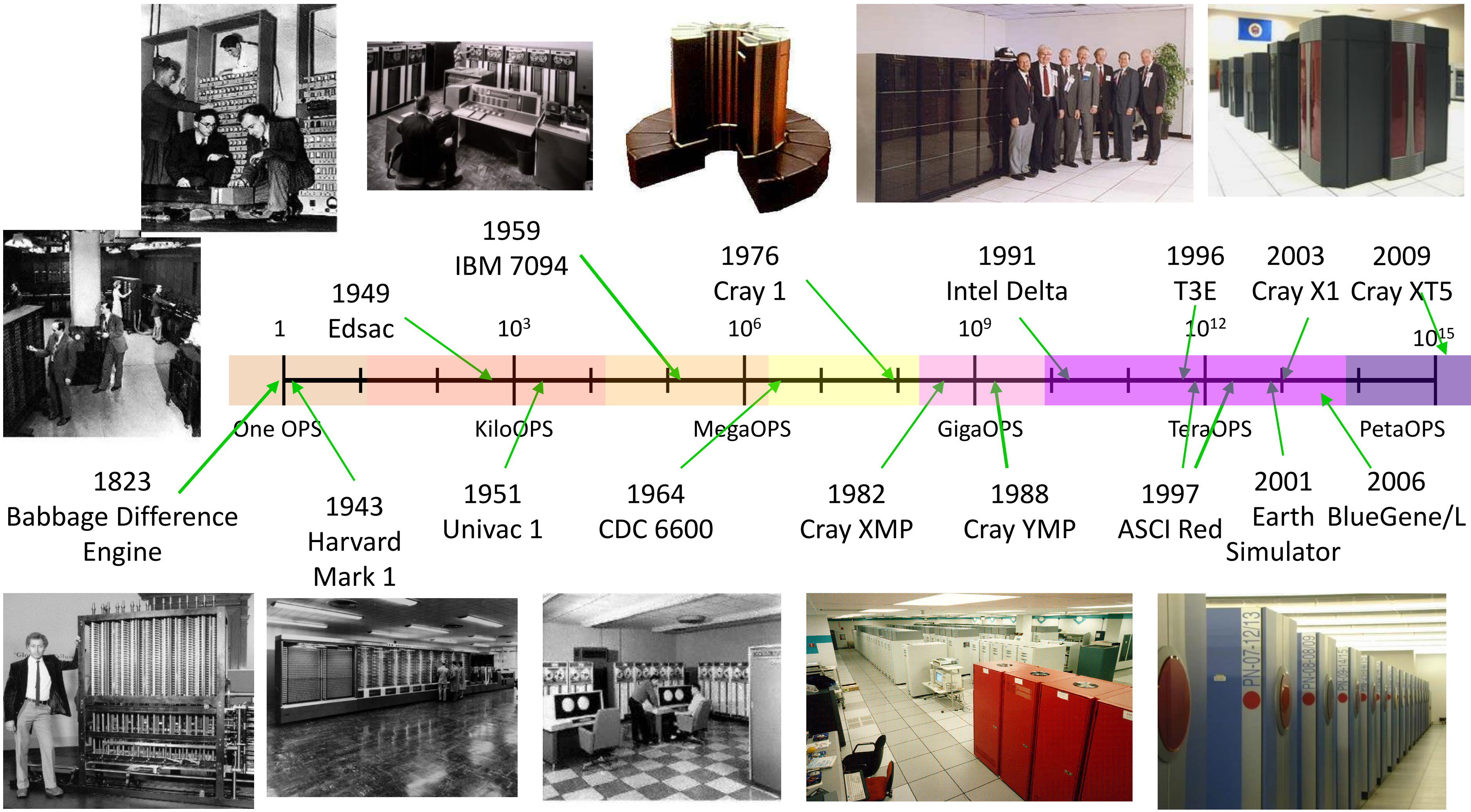
EDSAC

(Electronic Delay Storage Automatic Calculator)

- Maurice Wilkes, 1949.
- Mercury delay lines for memory and vacuum tubes for logic.
- Used one of the first assemblers called Initial Orders.
- Calculation of prime numbers, solutions of algebraic equations, etc.



Evolution of HPC



Fastest Computer in the US

Jaguar (Cray XT5-HE)

- Owned by Oak Ridge National Laboratory
- Breaks petaflops processing barrier(1.759e+15 flops)
- Contains 224,162 AMD x86_64 Opteron Six Core 2600 MHz chips

Tutorial Overview

- Introduction
  - Supercomputing and what it does
  - Performance and Parallelism
  - This tutorial and its objectives
- Architecture
  - Enabling technologies
  - Symmetric Multi-Processors
  - Commodity Clusters
- Throughput computing
  - Using Condor
  - Doing many jobs at one time
  - Parametric sweeps
- Shared Memory Computing
  - OpenMP programming
  - Multiple thread parallelism
- Distributed Memory
  - MPI programming
  - Scalable
  - Communicating processes
- User Environments
  - Operating system
  - Mass storage
  - Visualization

Who can benefit from this tutorial

- **Computational Scientists** who want to go beyond single computer capability
- **Computer science researchers** who want to branch out into HPC hardware and software
- **System Administrators** who want to expand their role to managing scalable systems
- **Design Engineers** who want to understand the challenges of parallel computer systems
- **Educators** who want to expand the frontiers of learning for their students and provide exciting opportunities
- **Managers** who need to bring this technology to their institutional requirements
- **Students** who want to get started

Top 500 List

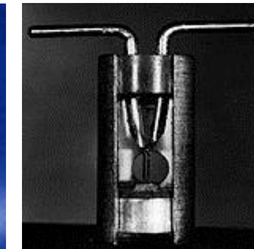
TOP500 SUPERCOMPUTER SITES
Performance Development
100PFlops
10 PFlops
1 PFlops
100 TFlops
10 TFlops
1 TFlops
100 GFlops
10 GFlops
1 GFlops
100 MFlops
Performance
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
43673.10 TF
2566.00 TF
31112 GF
1.12 TF
59.7 GF
0.42 GF
#1
#500
Sum
19/11/2010
http://www.top500.org/

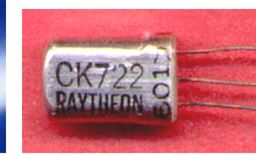
Performance

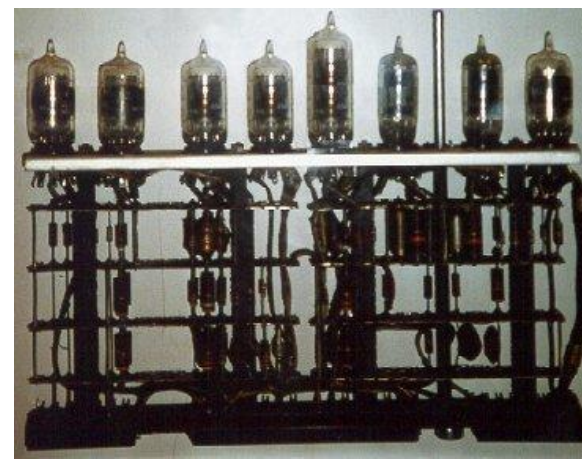
- *Performance*:
  - A quantifiable measure of the rate of doing (computational) work
  - Multiple such measures of performance
    - Delineated at the level of the basic operation
      - *ops* – operations per second
      - *ips* – instructions per second
      - *flops* – floating-point operations per second
    - Rate at which a *benchmark* program takes to execute
      - A carefully crafted and controlled code used to compare systems
      - *Linpack* Rmax (Linpack flops)
      - *gups* (billion updates per second)
- Two perspectives on performance
  - *Peak performance*
    - Maximum theoretical performance possible for a system
  - *Sustained performance*
    - Observed performance for a particular workload and run
    - Varies across workloads and possibly between runs

Where Does Performance Come From?

- Device Technology
  - Logic switching speed and device density
  - Memory capacity and access time
  - Communications bandwidth and latency
- Computer Architecture
  - Instruction issue rate
    - Execution pipelining
    - Reservation stations
    - Branch prediction
    - Cache management
  - Parallelism
    - Number of operations per cycle per processor
      - Instruction level parallelism (ILP)
      - Vector processing
    - Number of processors per node
    - Number of nodes in a system



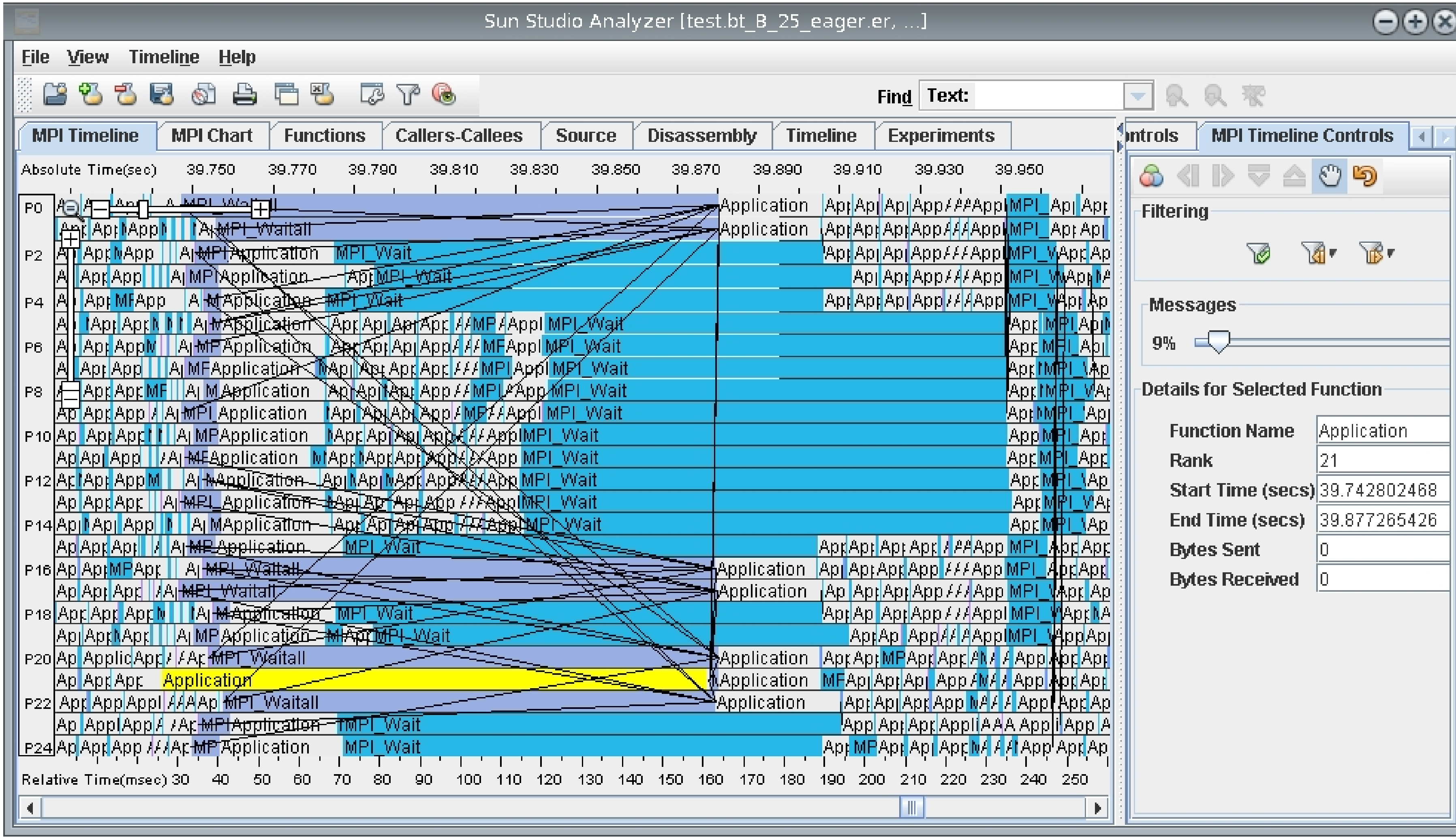




Measuring Performance

- Metrics
  - Wall clock time
  - Benchmarks
    - HPL (Linpack)
  - Processor efficiency factors
  - Scalability
  - System operations
    - Flops, Mflops, Gflops, Tflops, Pflops
- Tools
  - PAPI
  - Tau
  - Ganglia
  - Many others

MPI Performance Monitoring



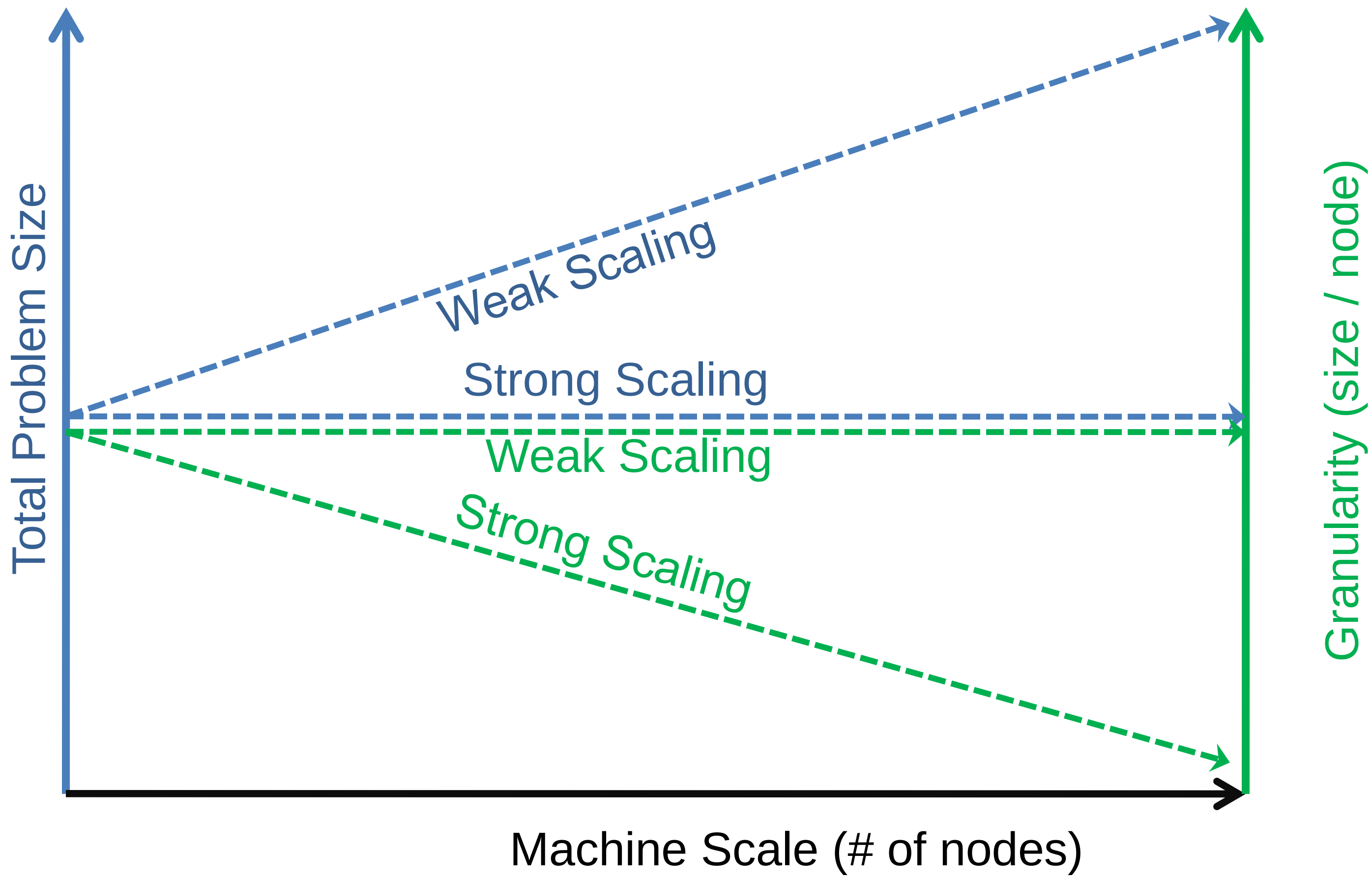
Machine Parameters affecting Performance

- Peak floating point performance
- Main memory capacity
- Bi-section bandwidth
- I/O bandwidth
- Secondary storage capacity
- Organization
  - Class of system
  - # nodes
  - # processors per node
  - Accelerators
  - Network topology
- Control strategy
  - MIMD
  - Vector, PVP
  - SIMD
  - SPMD

Scalability through Parallelism

- The ability to deliver proportionally greater sustained performance through increased system resources
- *Strong Scaling*
  - Fixed size application problem
  - Application size remains constant with increase in system size
- *Weak Scaling*
  - Variable size application problem
  - Application size scales proportionally with system size
- *Capability computing*
  - in most pure form: strict scaling
  - Marketing claims tend toward this class
- *Capacity computing*
  - Throughput computing
    - Includes job-stream workloads
  - In most simple form: weak scaling
- *Cooperative computing*
  - Interacting and coordinating concurrent processes
  - Not a widely used term
  - Also: *coordinated computing*

Strong Scaling, Weak Scaling

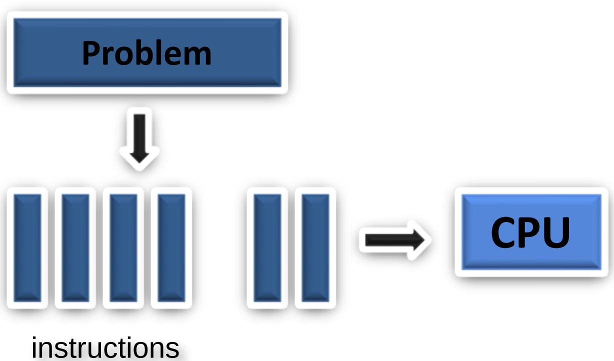


HPC Modalities

Modalities	Degree of Integration	Architectures	Execution Models	Programming Models
Capacity	Loosely Coupled	Clusters & Workstation farms	Workflow Throughput	Condor
Capability	Tightly Coupled	Vectors, SMP, SIMD	Shared Memory Multithreading	OpenMP
Cooperative	Medium	DM MPPs & Clusters	CSP	MPI

Key Terms and Concepts

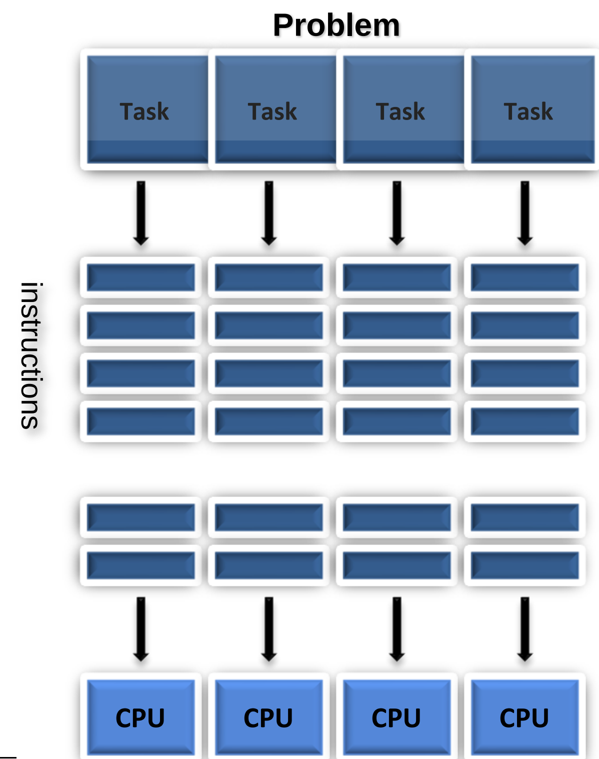
Conventional *serial execution* where the problem is represented as a series of instructions that are executed by the CPU



Parallel computing takes advantage of concurrency to :

- Solve larger problems within bounded time
- Save on wall clock time
- Overcome memory constraints
- Utilize non-local resources

Parallel execution of a problem involves partitioning of the problem into multiple executable parts that are mutually exclusive and collectively exhaustive represented as a partially ordered set exhibiting concurrency.



Key Terms and Concepts

- *Scalable Speedup:* Relative reduction of execution time of a fixed size workload through parallel execution

$$Speedup = \frac{execution\_time\_on\_one\_processor}{execution\_time\_on\_N\_processors}$$

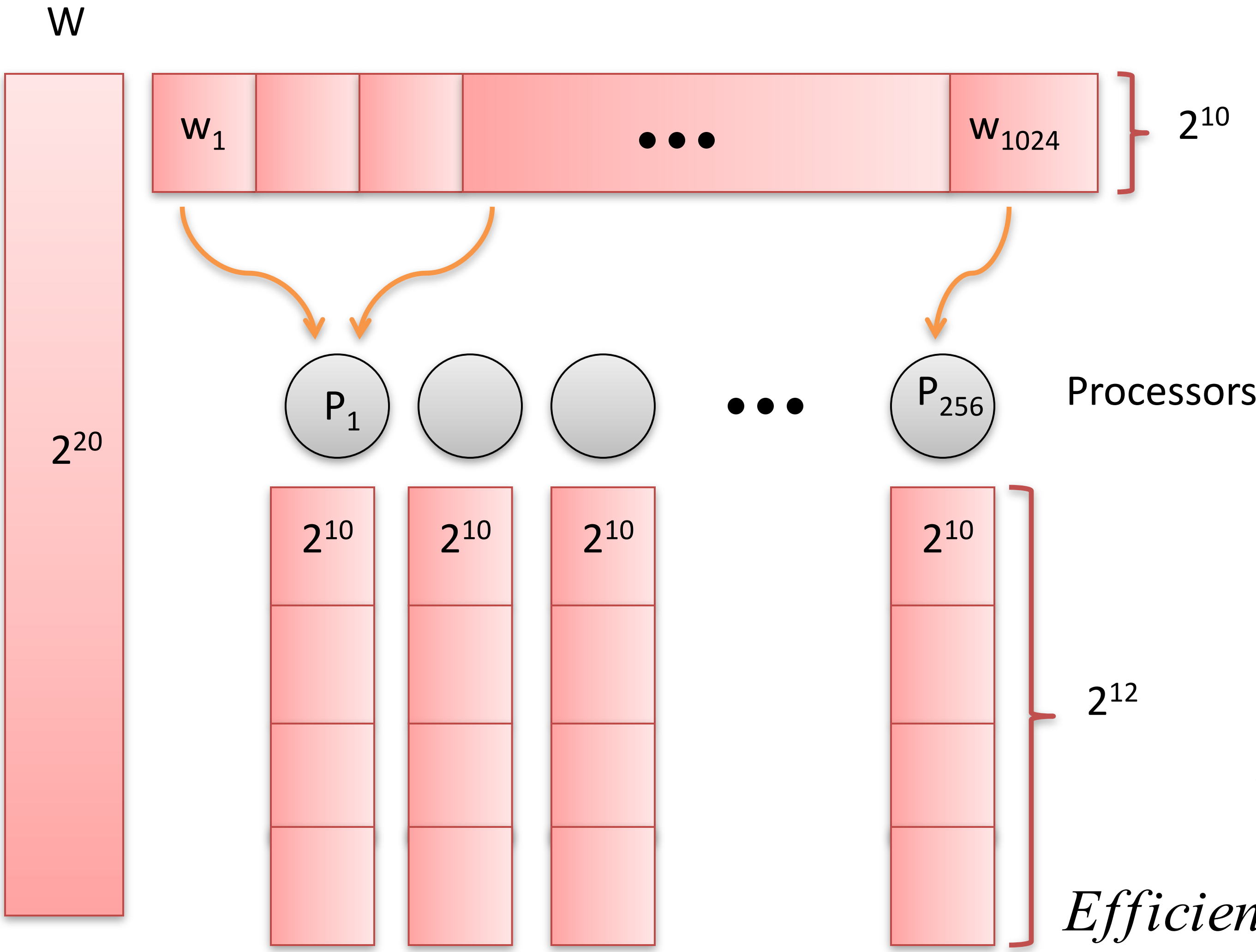
- *Scalable Efficiency :* Ratio of the actual performance to the best possible performance.

$$Efficiency = \frac{execution\_time\_on\_one\_processor}{(execution\_time\_on\_multiple\_processors \times number\_of\_processors)}$$

Ideal Speedup Issues

- ***W*** is total workload measured in elemental pieces of work (e.g. operations, instructions, etc.)
- ***T(p)*** is total execution time measured in elemental time steps (e.g. clock cycles) where ***p*** is # of execution sites (e.g. processors, threads)
- w_i is work for a given task ***i***
- Example: here we divide a million (really Mega) operation workload, ***W***, in to a thousand tasks, w_1 to w_{1024} , each of 1K operations
- Assume 256 processors performing workload in parallel
- ***T(***256***)*** = 4096 steps, speedup = 256, Eff = 1

Ideal Speedup Example



Units : steps

$$W = \sum_i w_i$$

$T(1)=2^{20}$

$T(2^8)=2^{12}$

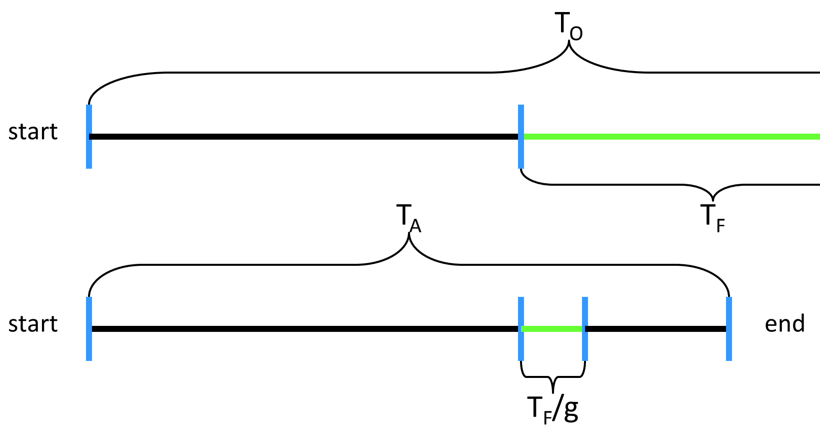
$$Speedup = \frac{2^{20}}{2^{12}} = 2^8$$

$$Efficiency = \frac{2^{20}}{2^{12} \times 2^8} = 2^0 = 1$$

Sources of Performance Degradation

- *Latency*
  - Waiting for access to memory or other parts of the system
- *Overhead*
  - Extra work that has to be done to manage program concurrency and parallel resources the real work you want to perform
- *Starvation*
  - Not enough work to do due to insufficient parallelism or poor load balancing among distributed resources
- *Contention*
  - Delays due to fighting over what task gets to use a shared resource next. Network bandwidth is a major constraint.

Amdahl's Law



$T_O \equiv$ time for non-accelerated computation

$T_A \equiv$ time for accelerated computation

$T_F \equiv$ time of portion of computation that can be accelerated

$g \equiv$ peak performance gain for accelerated portion of computation

$f \equiv$ fraction of non-accelerated computation to be accelerated

$S \equiv$ speed up of computation with acceleration applied

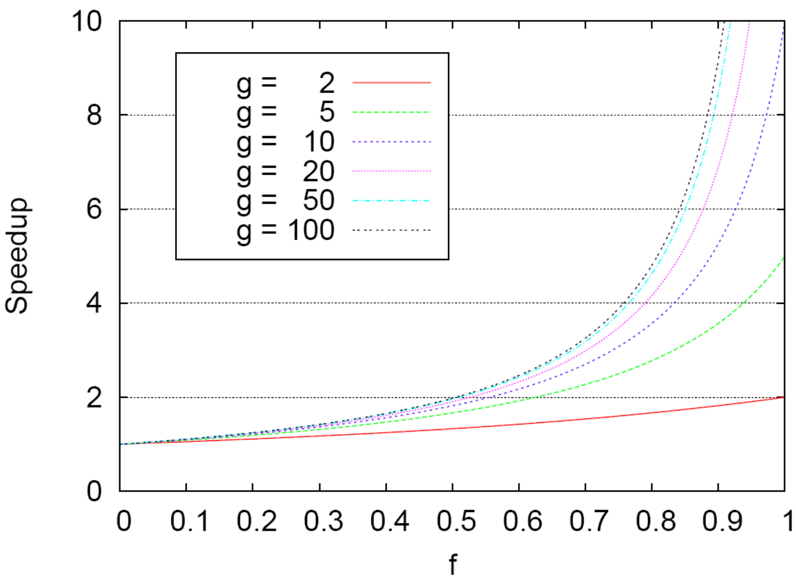
$S = T_O / T_A$

$f = T_F / T_O$

$$T_A = (1 - f) \times T_O + \left(\frac{f}{g}\right) \times T_O$$

$$S = \frac{T_O}{(1 - f) \times T_O + \left(\frac{f}{g}\right) \times T_O}$$

$$\boxed{S = \frac{1}{1 - f + \left(\frac{f}{g}\right)}}$$



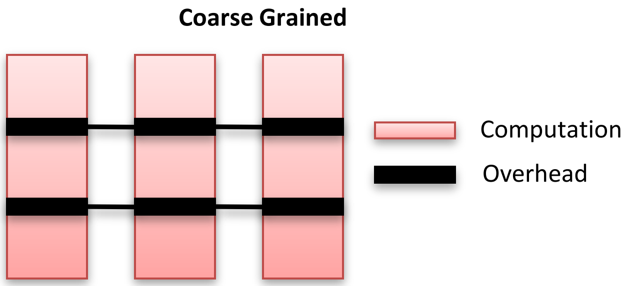
Granularities in Parallelism

Overhead

- The additional work that needs to be performed in order to manage the parallel resources and concurrent abstract tasks that is in the critical time path.

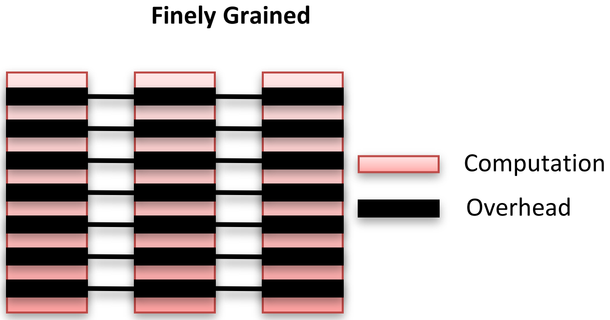
Coarse Grained

- Decompose problem into large **independent** tasks. Usually there is no communication between the tasks. Also defined as a class of parallelism where: “relatively large amounts of computational work is done between communication”



Fine Grained

- Decompose problem into smaller **inter-dependent** tasks. Usually these tasks are communication intensive. Also defined as a class of parallelism where: “relatively small amounts of computational work are done between communication events” –*www.llnl.gov/computing/tutorials/parallel_comp*



Images adapted from : http://www.mhpcc.edu/training/workshop/parallel_intro/

Supercomputing System Stack

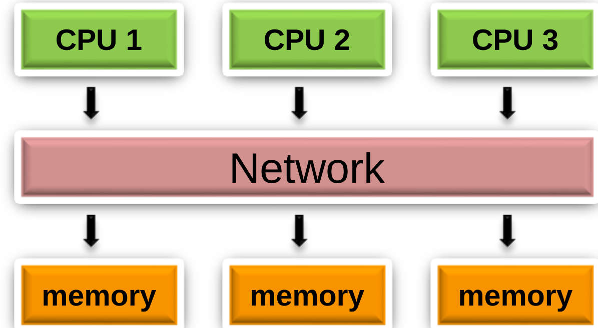
- Device technologies
  - Enabling technologies for logic, memory, & communication
  - Circuit design
- Computer architecture
  - semantics and structures
- Models of computation
  - governing principles
- Operating systems
  - Manages resources and provides virtual machine
- Compilers and runtime software
  - Maps application program to system resources, mechanisms, and semantics
- Programming
  - languages, tools, & environments
- Algorithms
  - Numerical techniques
  - Means of exposing parallelism
- Applications
  - End user problems, often in sciences and technology

Models of Parallel Processing

- Conventional models of parallel processing
  - Decoupled Work Queue (covered today)
  - Communicating Sequential Processing (CSP, message passing) (covered on day 2)
  - Shared memory multiple thread (covered on day 3)
- Some alternative models of parallel processing
  - SIMD
    - Single instruction stream multiple data stream processor array
  - Vector Machines
    - Hardware execution of value sequences to exploit pipelining
  - Systolic
    - An interconnection of basic arithmetic units to match algorithm
  - Data Flow
    - Data precedent constraint self-synchronizing fine grain execution units supporting functional (single assignment) execution

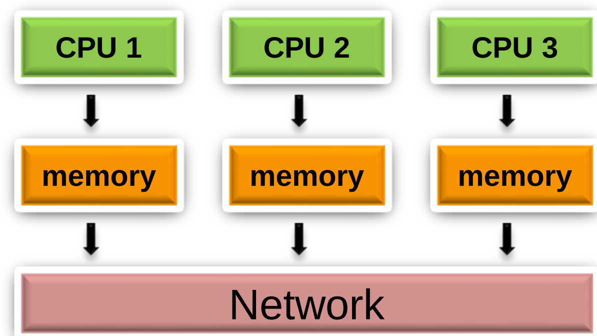
Shared Memory Multiple Thread

- Static or dynamic
- Fine grained
- OpenMP
- Distributed shared memory systems
- Covered on day 3



Symmetric Multi Processor
(SMP usually cache coherent)

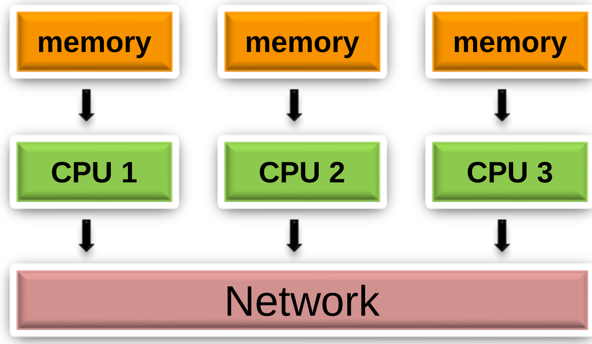
Orion JPL NASA



Distributed Shared Memory
(DSM often not cache coherent)

Communicating Sequential Processes

- One process is assigned to each processor
- Work done by the processor is performed on the local data
- Data values are exchanged by messages
- Synchronization constructs for inter-process coordination
- Distributed Memory
- Coarse Grained
- MPI
- Clusters and MPP
  - MPP is acronym for "Massively Parallel Processor"
- Covered on day 2



Distributed Memory
(DM often not cache coherent)

SuperMike LSU