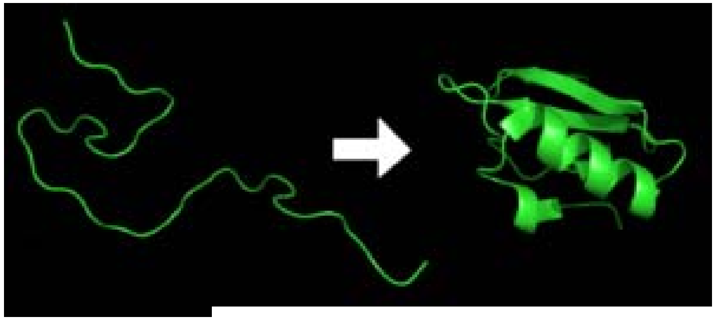
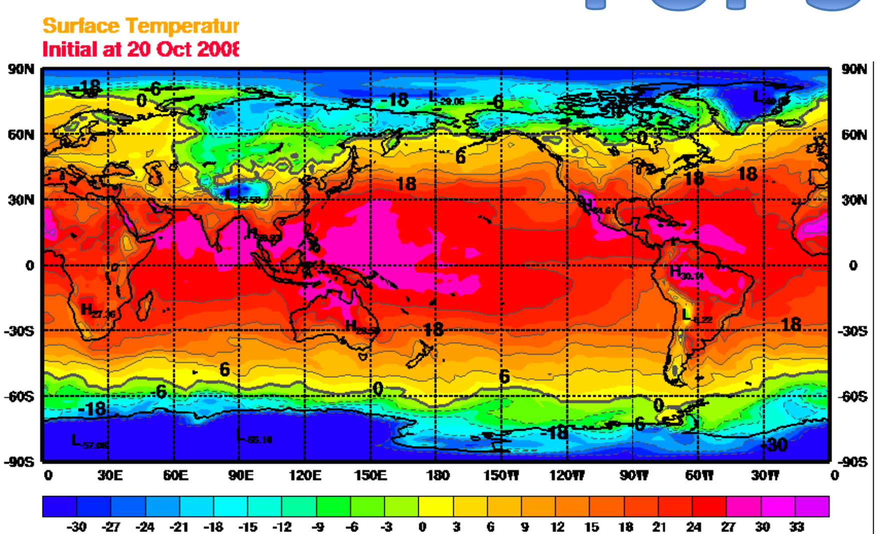


Parallel Computing: Power in Numbers (?) For Science



Let’s say I give you a homework assignment today with 100 problems.

Each problem takes 2 hours to solve.

The homework is due tomorrow.

Big problems and Very Big problems in Science

How do we live forever?

What happened in the past?

Molecular dynamics

Complex Computations Time - Cost

What will happen in the future ?

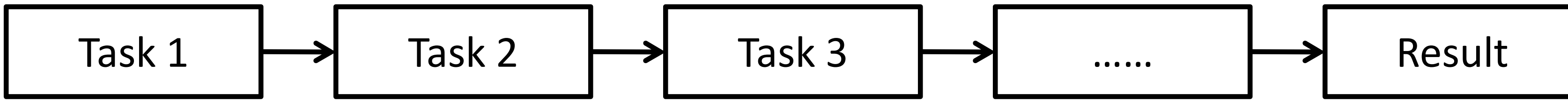
Earthquake prediction

Lottery ball prediction

- Is it possible to divide a VBP to BPs to Small Problems ?
  Cost to solve 1 VBP >> Cost to solve component SPs
- Are the SPs serial or parallel ? -- split up time and cost

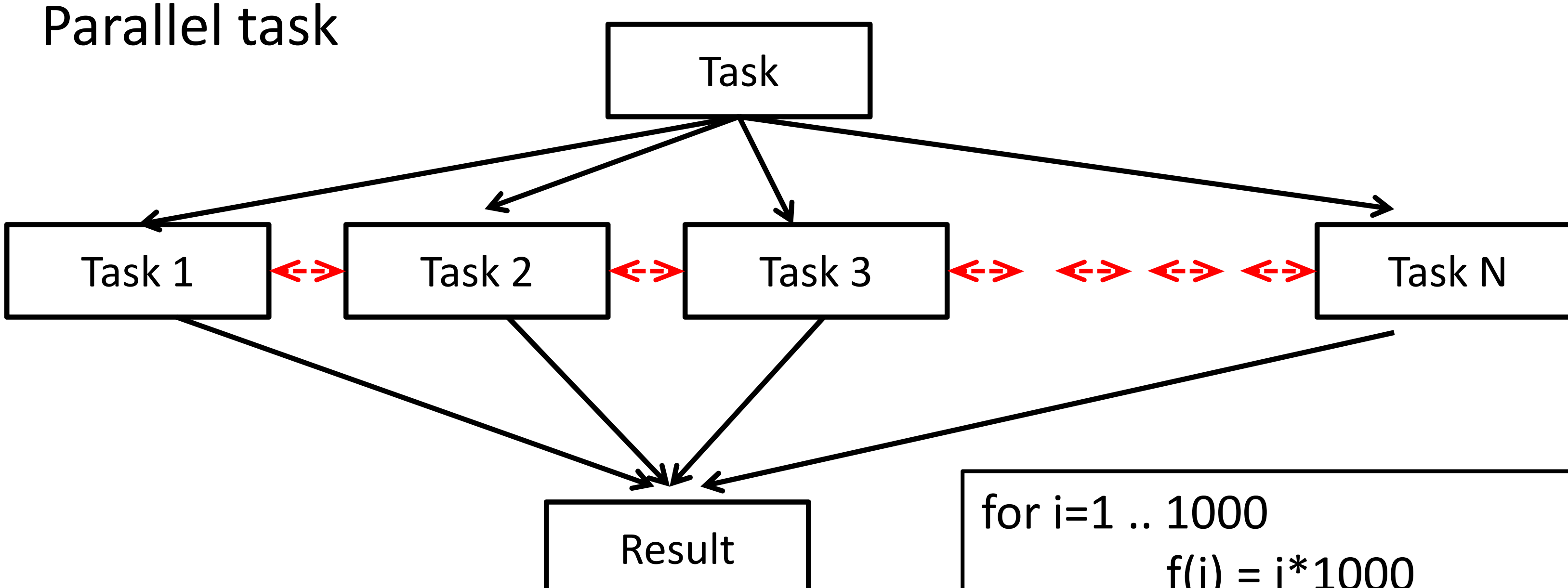
Sequential and Parallel Tasks

Sequential task




```
for i=1 .. 1000
        f(i) = f(i-1) + f(i-2)
```

Parallel task




```
for i=1 .. 1000
        f(i) = i*1000
```

Overview

Evolution of a computer

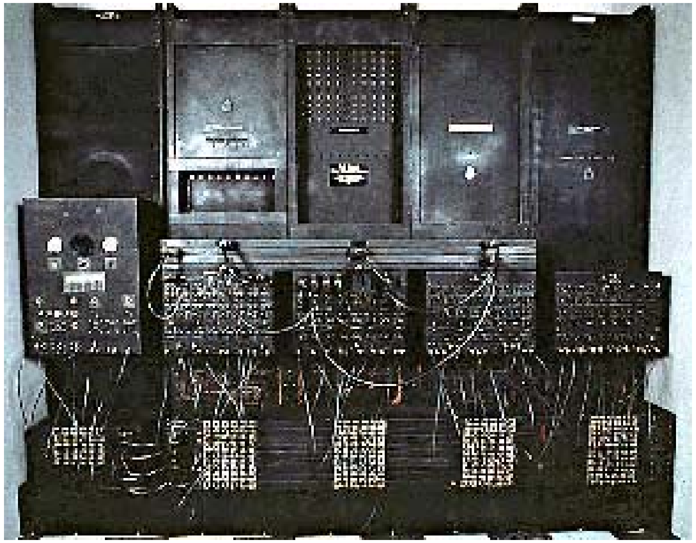
Parallel machines – Hardware and Software

Advantages and limits of Parallel computing

What are the applications in physics ?

Women Computers in World War 2

ENIAC (1946) 2.6m x 0.9mx28m
7000 transistors

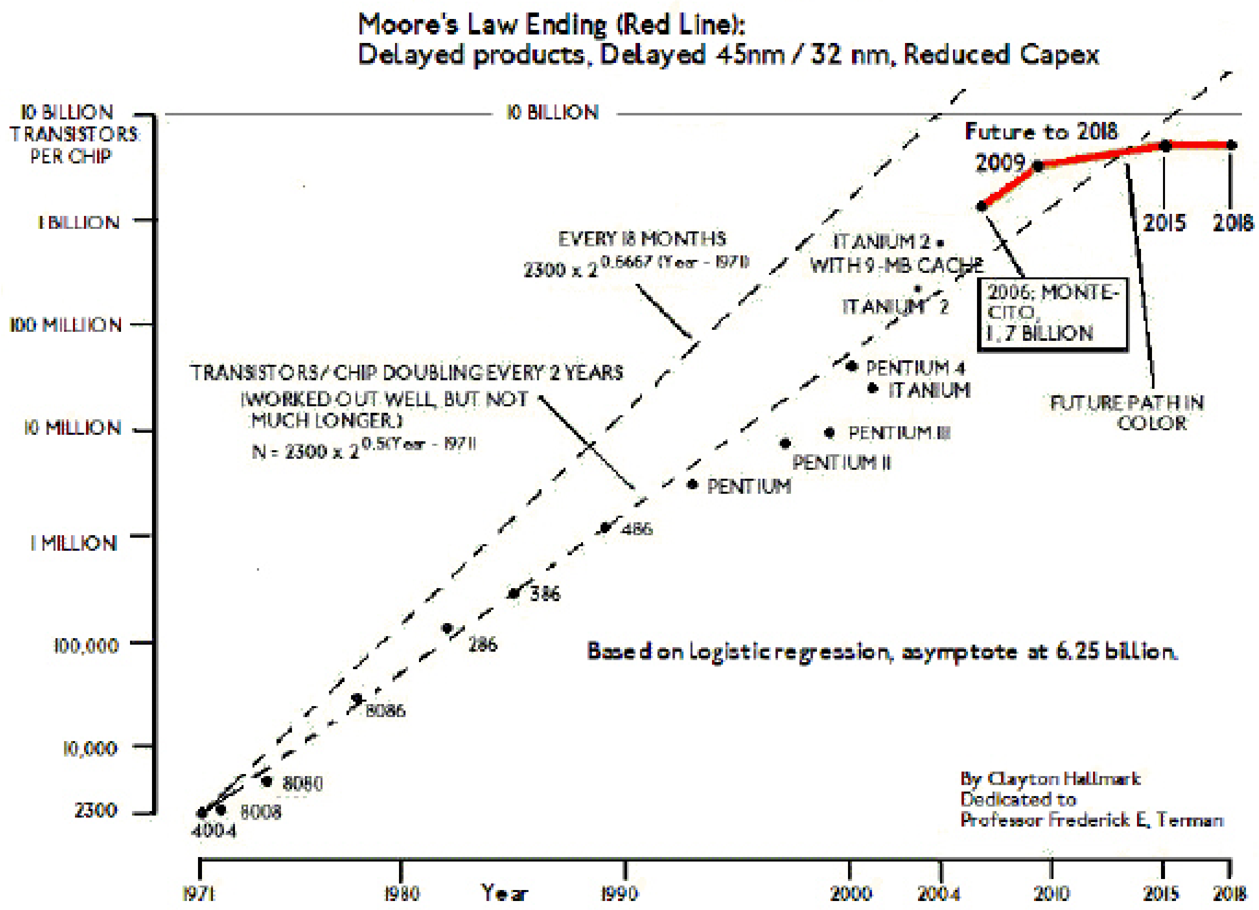


Intel Core 2 5cm x 5cm
2 x 10^9 transistors

IEEE Virtual Museum: http://www.ieee-virtual-museum.org/

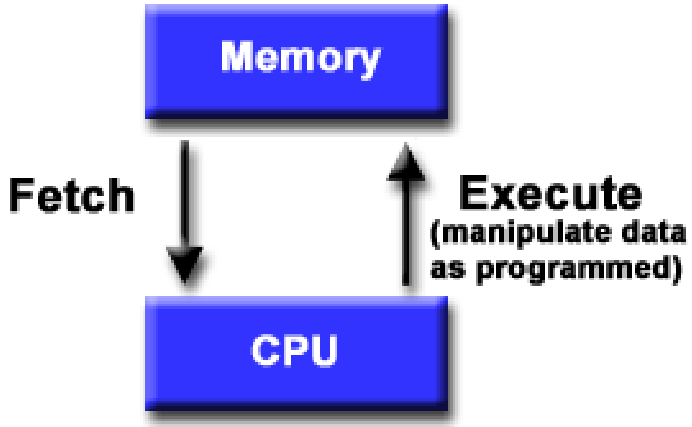
Moore's Law : The # of transistors that can be placed inexpensively on an integrated circuit increases exponentially, doubling ~ every 1.5 years

Limitless growth ?   Heat dissipation – Cost – Demand



A Computation Machine

Von Neumann Computer Model



1. Memory stores data and program

2. Program instructions tell the computer to do something. (Go to the market and buy groceries)

3. Data is information needed by the program to process. (Groceries = Milk, Eggs, Tofu, Bread …)

4. A CPU gets instructions and data from memory, and process the instruction and data.

https://computing.llnl.gov/tutorials/parallel_comp/

Flynn's Taxonomy

Go to the supermarket
Go to the library
Go to department store

Go to the supermarket

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

Milk

Milk
Books
Clothes

SISD – serial computation; older desktop PCs; Pentium

SIMD – each processor work on different data with same instruction; image proc.

MISD – multi. computations on a single data; multi. freq. filters for different freq.

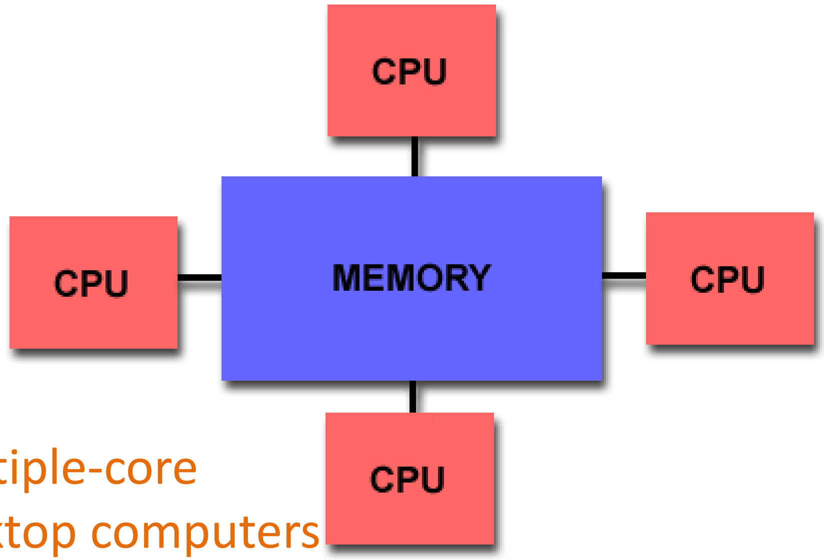
MIMD – each processor work on different data with different instruction; multicore

Parallel Machines

Shared Memory

Multiple processors share the same memory resources, but operate different instructions

Changes in a memory location effected by one processor => change for all processors



Multiple-core desktop computers

Advantage

User-friendly, easy to program with perspective to memory

Data sharing between CPUs is both fast and uniform

Disadvantages

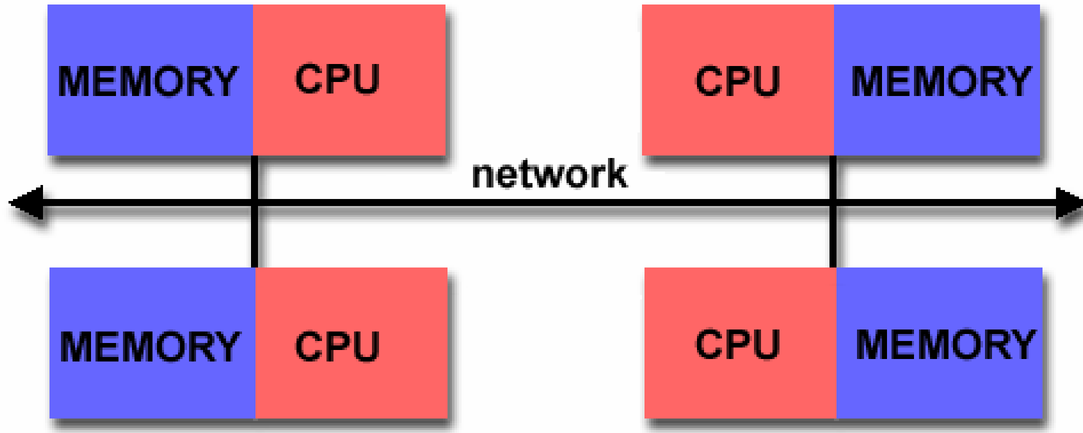
Expensive : Memory price is nonlinear

Lack of scalability between memory-CPU communication pathways.

Data change must be synchronized.

Distributed Memory

Pentium/Athlon computer clusters



Requires a communication network to connect inter-processor memory.

Local memory – not shared but can be communicated

Network is often the bottleneck

Advantage

Memory is scalable with # of processors.

Each processor can rapidly access its own memory without interference

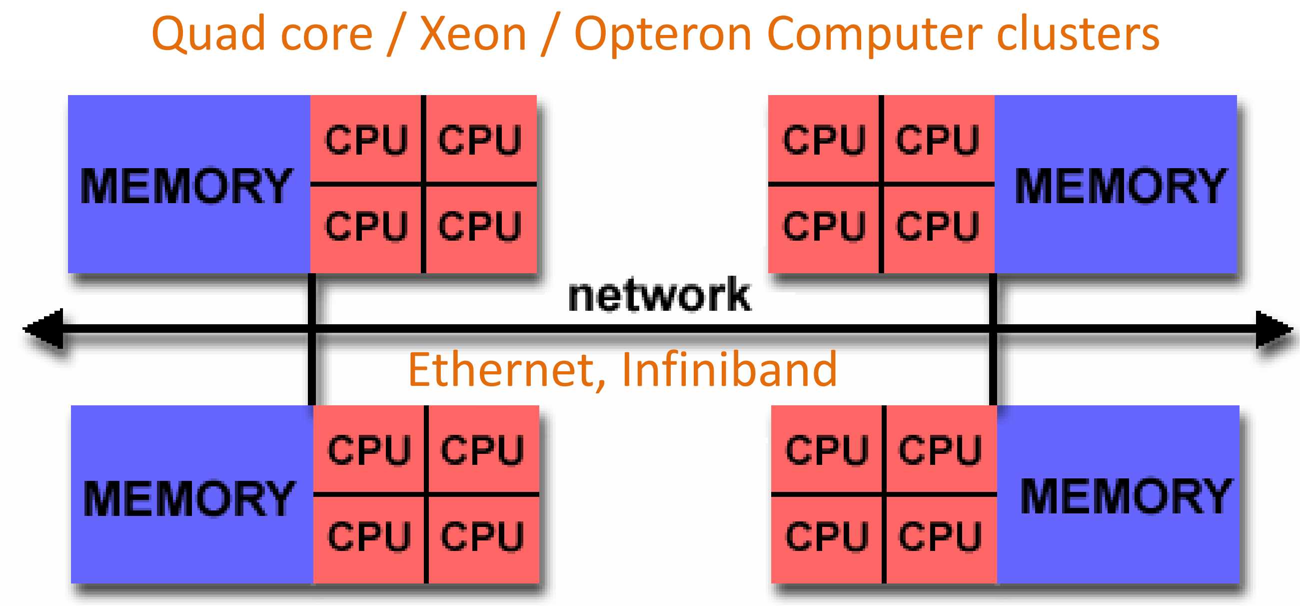
Less expensive : use many cheap CPUs to do the job of one very expensive one
-nonlinear pricing structure

Disadvantages

More difficult to program – need to account for communication across the network, performance is thus highly system dependent

Not always easy to separate the data structure into parts

Hybrids



- Processors on a given shared memory cache can address that machine's memory as global.

- SMPs know only about their own memory – network is needed to communicate data between SMPs.

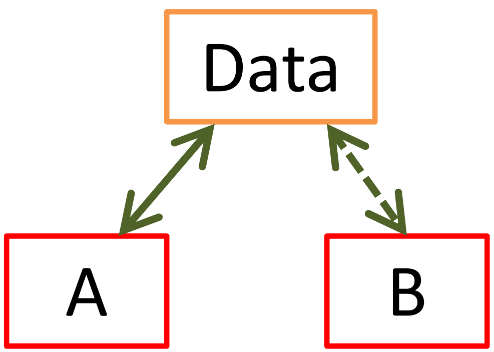
- The fastest supercomputers today are hybrids

Parallel Programming

Parallel programming models exist as an abstraction above hardware and memory architectures.

Shared Memory

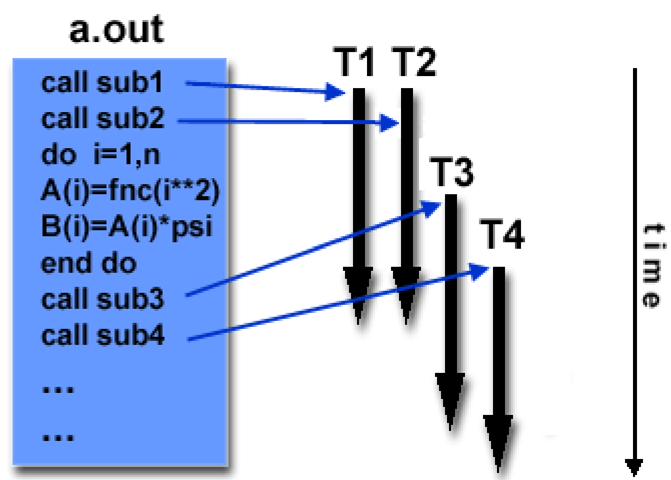
All processors work on the same data
- So programming mechanisms to enforce synchronized data.



Threads

Subroutines are sent to be executed concurrently on different processors

OpenMP

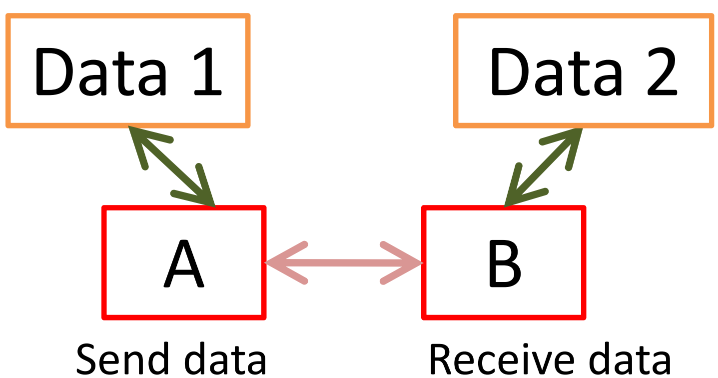


Message passing

Multiple processors execute on data in the local memory.

Then exchange data by coordinated network communication, i.e. one send signal has to correspond to one receive signal.

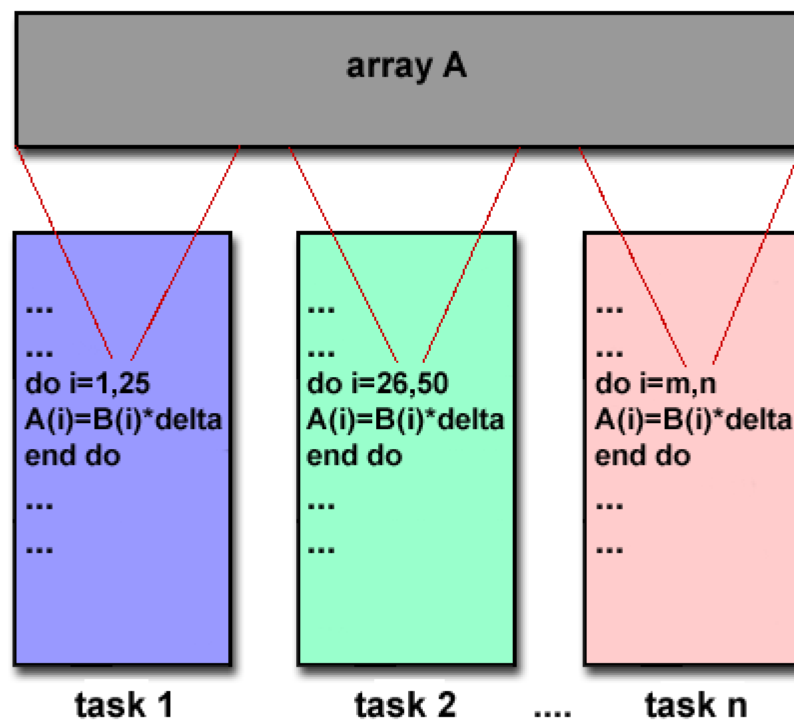
MPI



Data parallel

A set of tasks work collectively on the same data structure (like an array or matrix).

Each task works on a different partition of the same data structure. Such as "multiply every array by 2".



Designing a parallel program

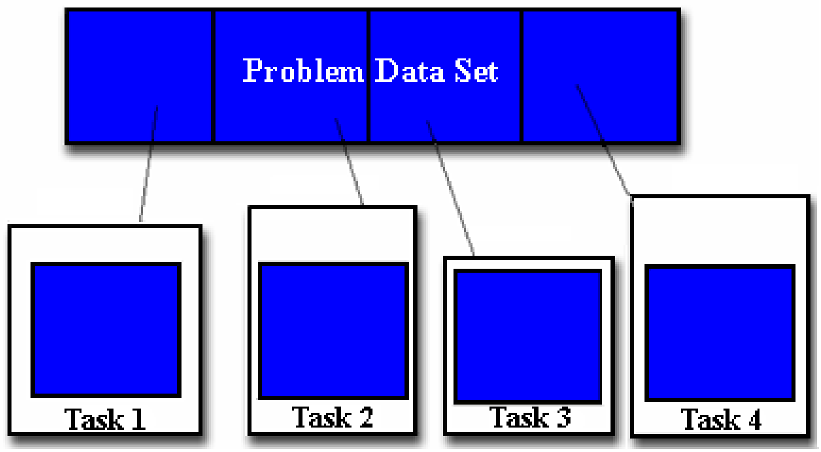
Not all problems can be solved using parallelism

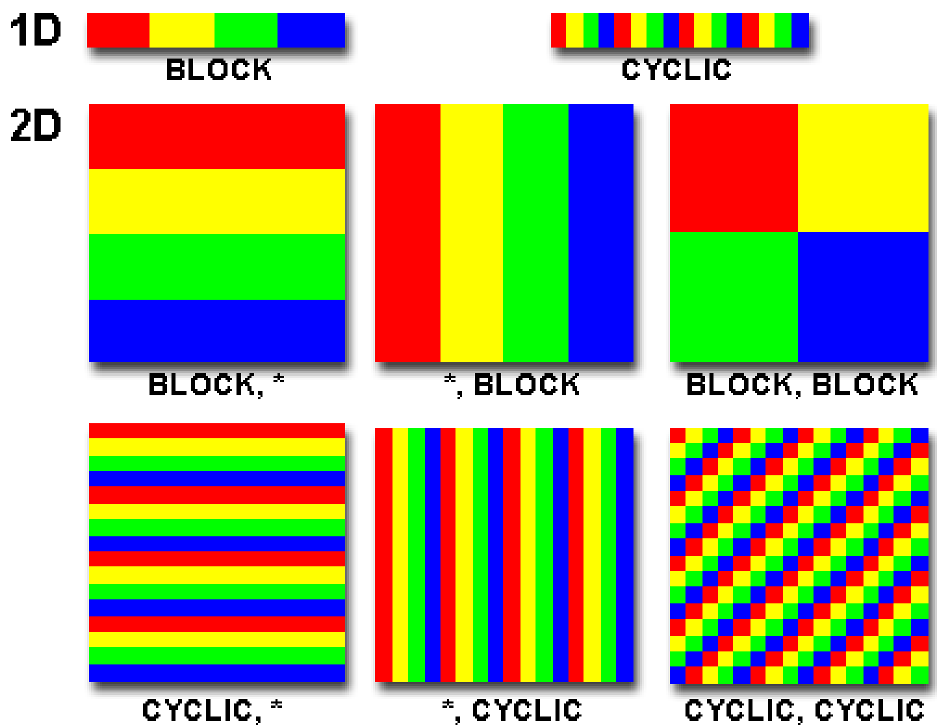
Calculation of the Fibonacci series $f(n) = f(n-1) + f(n-2)$ cannot be parallel

Calculation of the trajectories of 100 independent molecules can be parallel

- We need to know where computation is most intensive in a program.
  - do 1000 square roots
- We need to know where the bottlenecks are -- network communication ?
- Investigate different algorithms – different paths to parallelize
  Experience helps

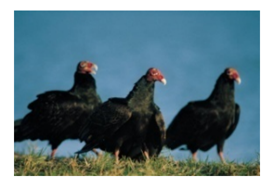
Data Partitioning





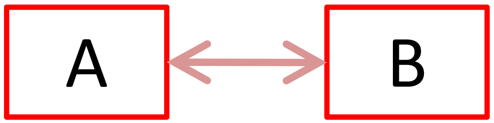
Functional Partitioning

Ecosystem



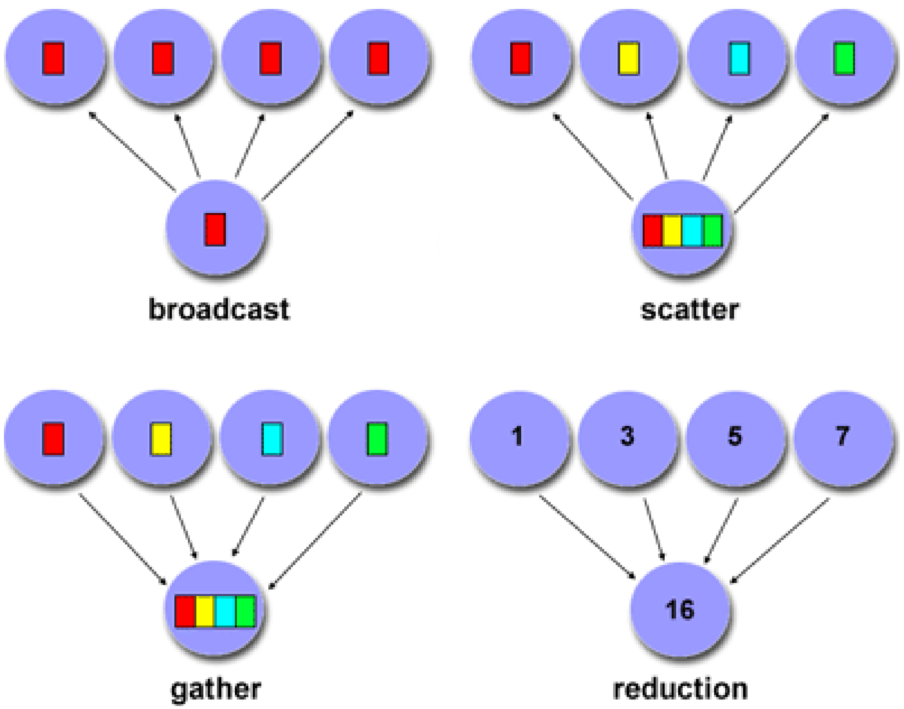
Communication

Point-to-point



Send data Receive data

Collective communication



Communication is usually expensive – highly dependent on the network – should be kept to minimum.

More work, less talking

Granularity defines the ratio to computation to communication.

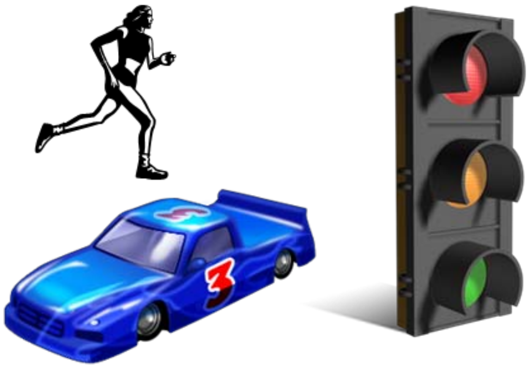
Coarse: many computational between communication events

Fine: small amounts of computational between communication events

Synchronization and Load Balance

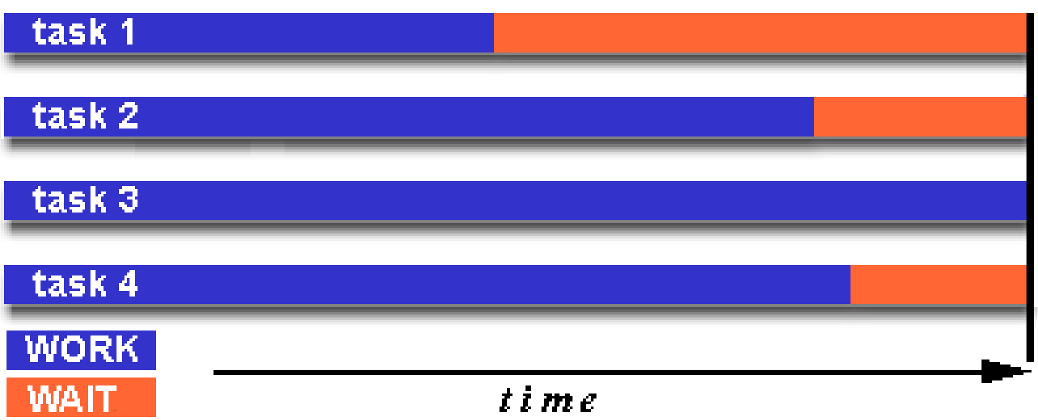
Processors often need to be synchronized

- exchange data
- execute serial routine
- Input / Output



When this happens, faster processes needs to wait for slower processes, thus waste time

=> Need to balance the load for each task to minimize wait time



Performance

CPU speed & CPU number

Memory speed & Memory size

CPU – Memory pathways

Network communication

For each problem and each system, the programming needs to be optimized

What’s the maximum speedup?

Amdahl’s Law

Seq = % of time a program is sequential

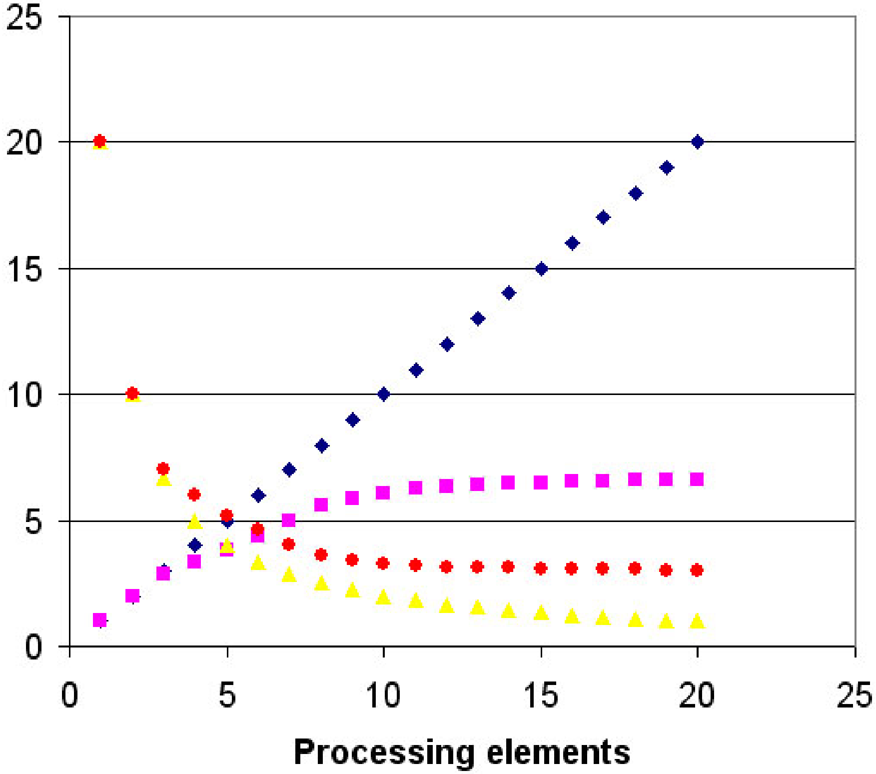
Par = % of time a program is parallel

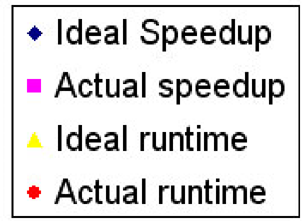
Execution = 1 = seq + par

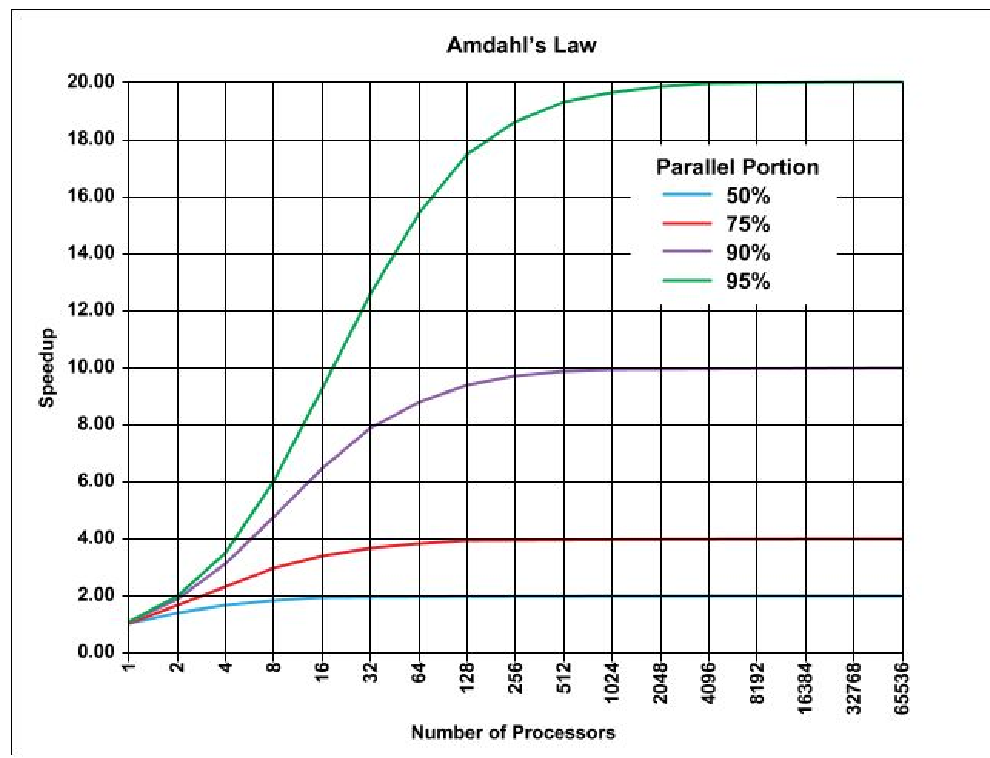
Number of processors =N

$$\text{speedup} = \frac{\text{seq'l}}{\text{para}} = \frac{(\text{seq} + \text{par})}{(\text{seq} + \text{par}/\text{N})}$$

$$\text{speedup} = \frac{\text{seq'l}}{\text{para}} = \frac{(\text{seq} + \text{par})}{(\text{seq} + \text{par/N})}$$





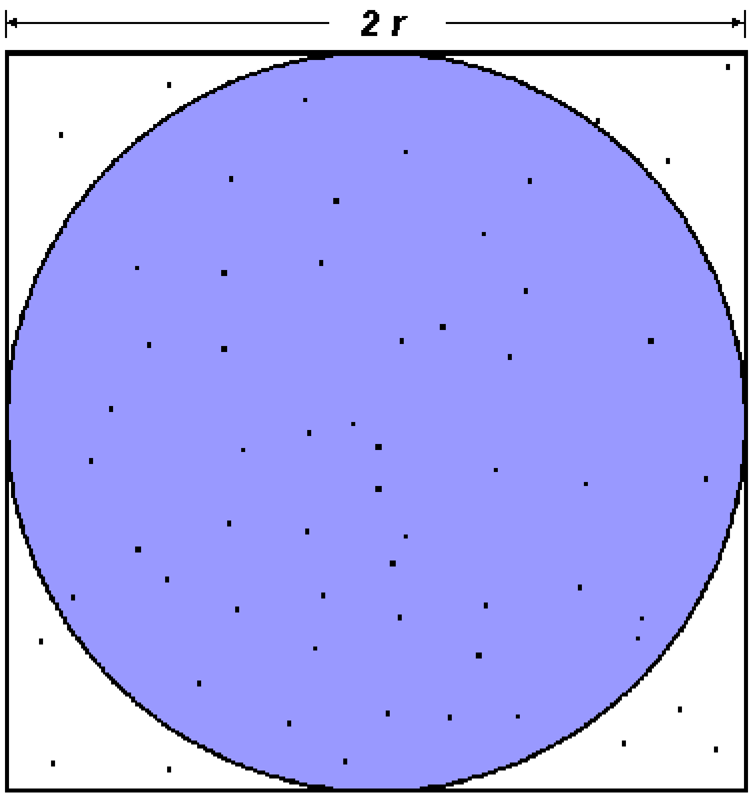


For small problems, the communication overhead could actually lead to lower performance.

Speedup could be super-linear if the sequential program is poorly designed

Avoid problems: Deadlock

Example: Calculate π



$$A_S = (2r)^2 = 4r^2$$

$$A_C = \pi r^2$$

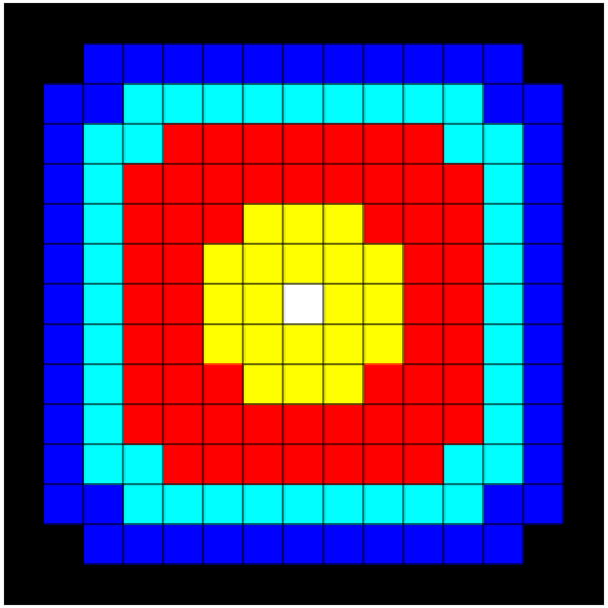
$$\pi = 4 \times \frac{A_C}{A_S}$$

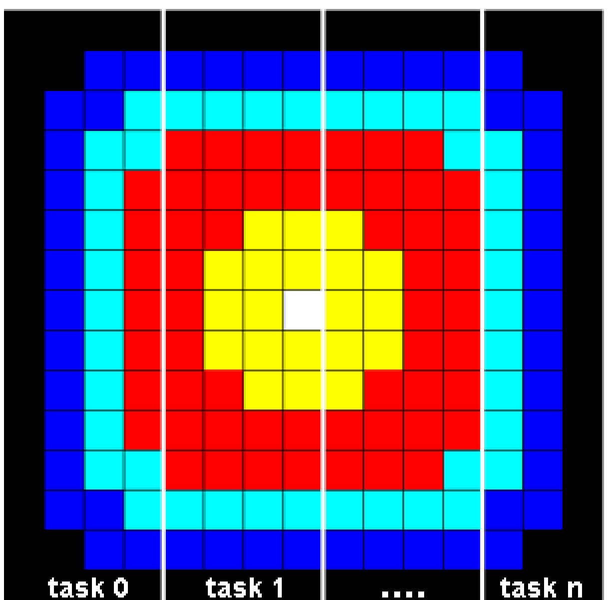
1. Draw a circle in a square
2. Randomly generate points in the square
3. Count the number of points in the square that are also in the circle
4. r =#of points in the circle divided by # of points in the square
5. π ~ 4 r

6, More points => More accuracy

Extremely Parallel

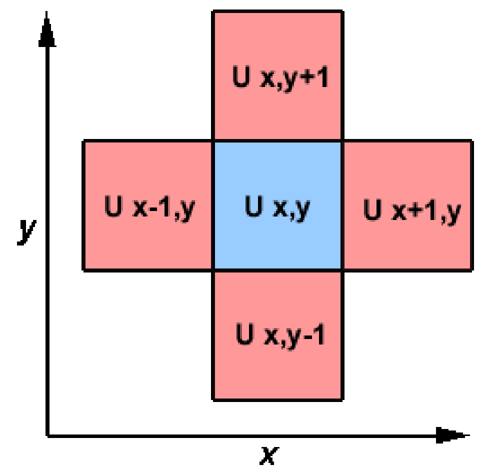
Example: Simple Heat Equation





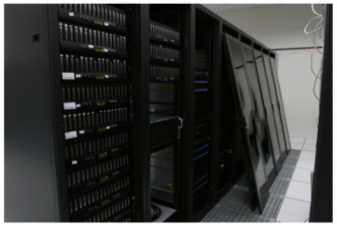
Temperature is 0 at the boundaries, high in the middle

$$
\begin{aligned}
U_{x,y}(t+\Delta t) &= U_{x,y}(t) \\
&+ C_x(U_{x+\Delta x,y}(t) + U_{x-\Delta x,y}(t) - 2U_{x,y}(t)) \\
&+ C_y(U_{x,y+\Delta y}(t) + U_{x,y-\Delta y}(t) - 2U_{x,y}(t))
\end{aligned}
$$



Communication is needed between neighbor tasks

Computing at Institute of Physics



TW - GRID

Over 2000 cores ~3 Teraflops

High speed serial / parallel computing

High energy physics

Protein folding

Genomic Science

Molecular dynamics

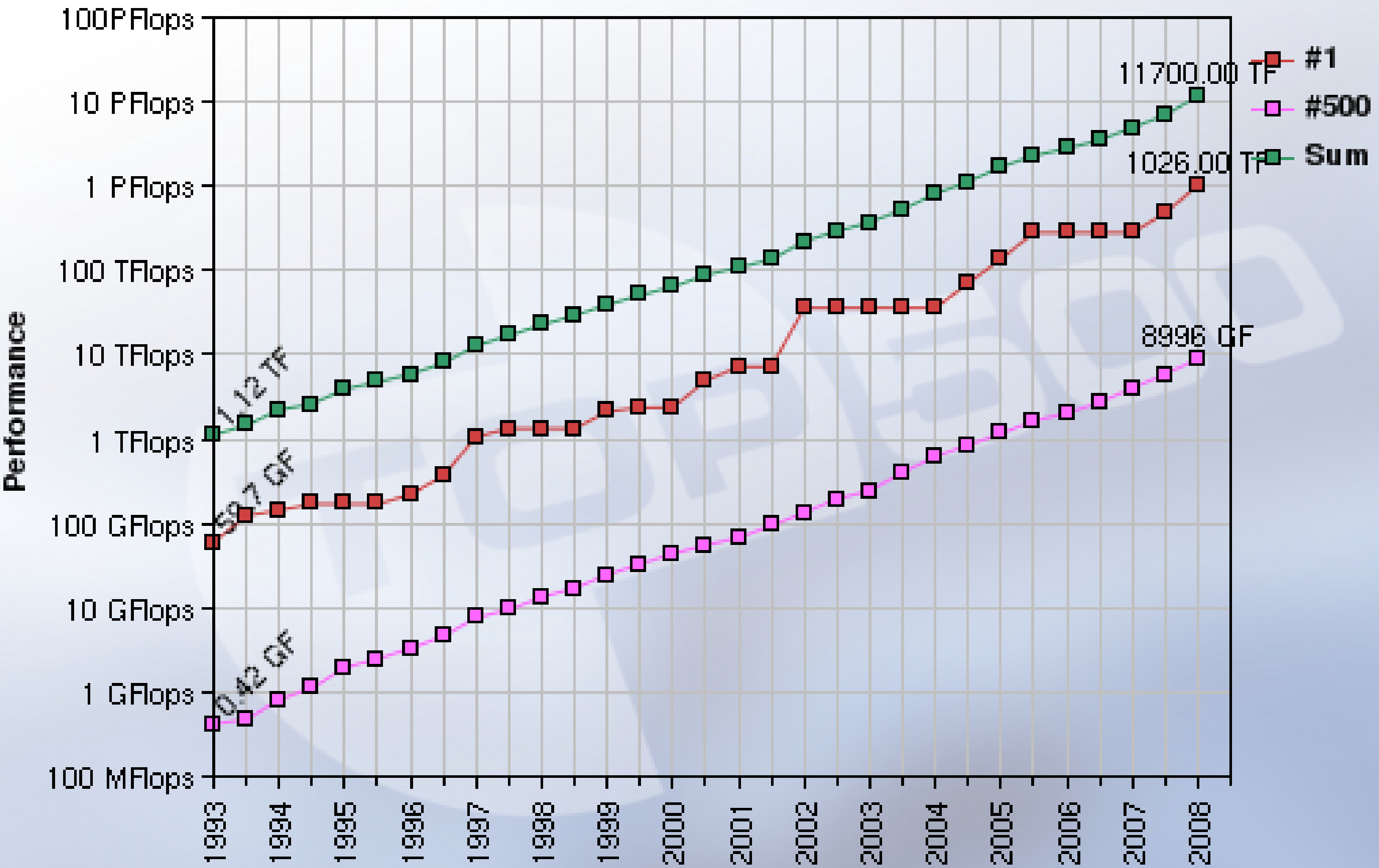
Biophysics

IBM Roadrunner – Breaking the Petaflop in May, 2008

2,960 IBM PowerXCell 8i CPUs and 6,480 AMD Opteron dual-core processors in specially designed server blades connected by Infiniband.

"Simulate how nuclear weapons age in order to predict whether the USA's aging arsenal of nuclear weapons is safe and reliable"

Performance Development



Distributed Computing

Parallel computing on a large scale

Grid computing : CPU scavenging and volunteer computing

Across the ethernet, software is run on your computer and the data is sent back to a collection server, creating a large, virtual supercomputer

BOINC : Berkeley Open Infrastructure for Network Computing

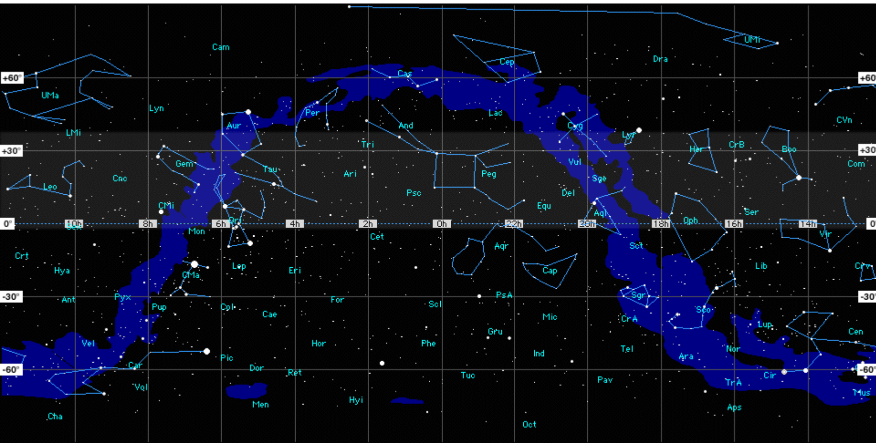
Download to your PC and recycle your unused PC time -- Screensaver

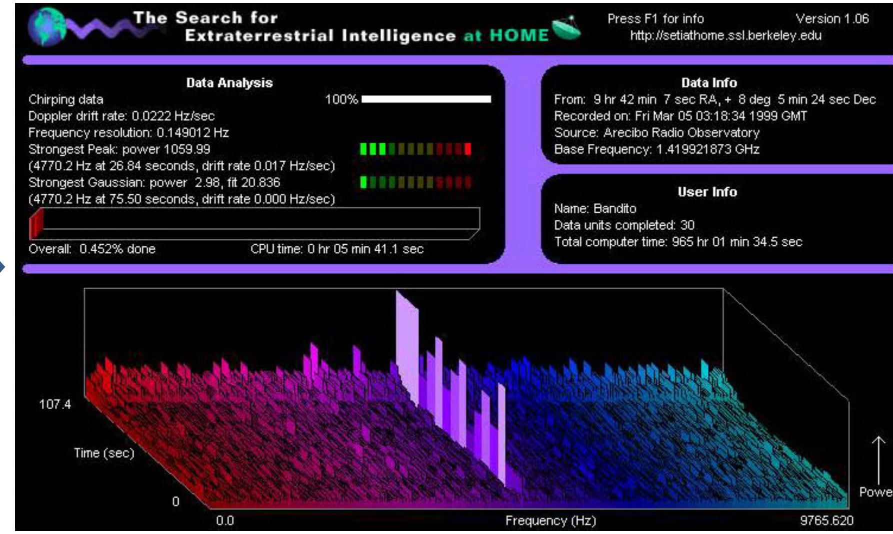
24-hour average: 918.22 TeraFLOPS

http://boinc.berkeley.edu/

Folding@home: How is protein folding linked to disease?

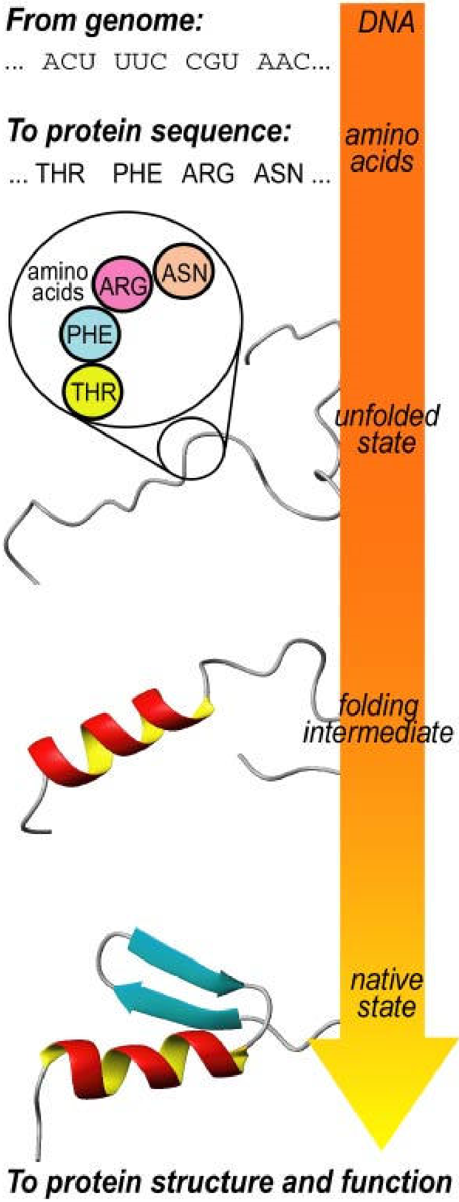
SETI @ home : Search for Extraterrestrial Intelligence

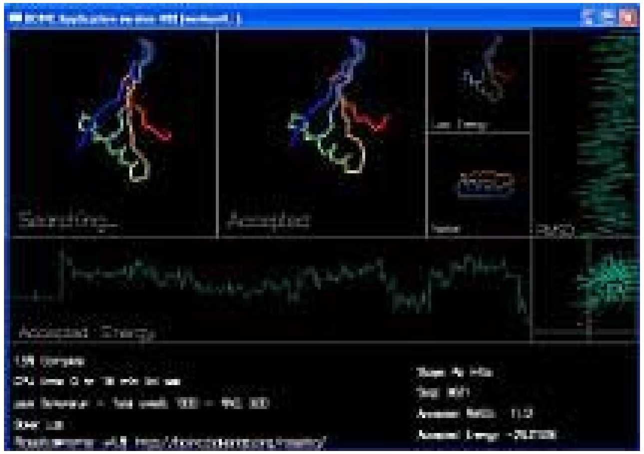




Arecibo Observatory – 305m

Folding @ home : Cure diseases and better health

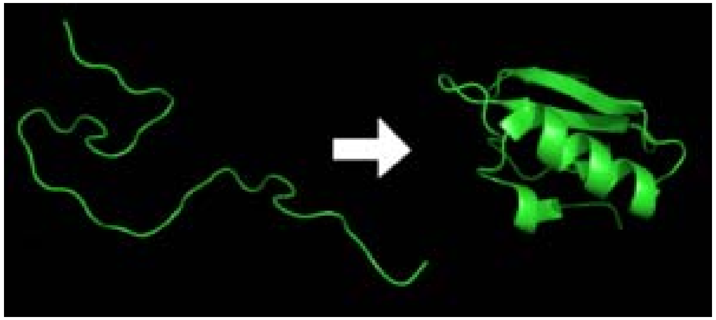


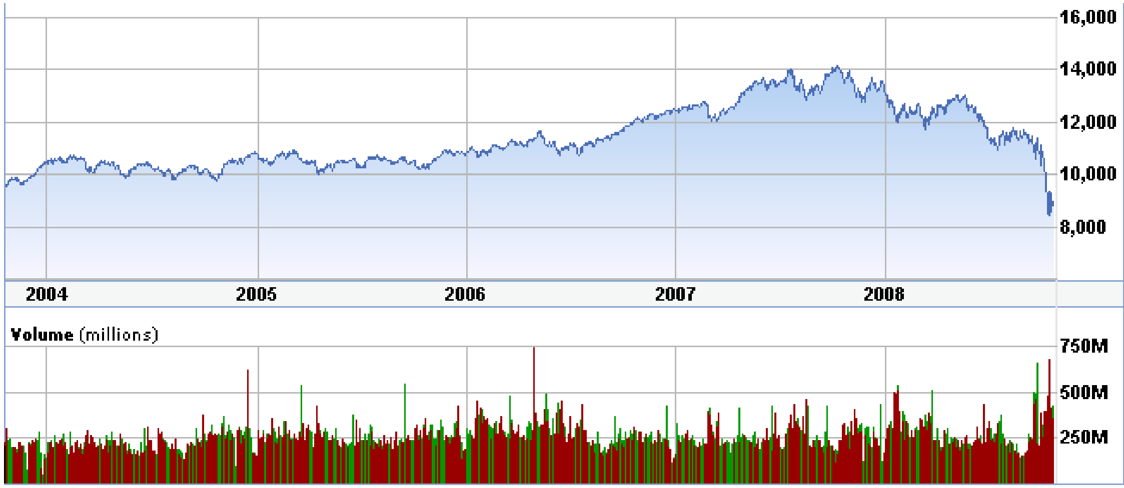


Even short polypeptides (<10) can fold into billions of different structures

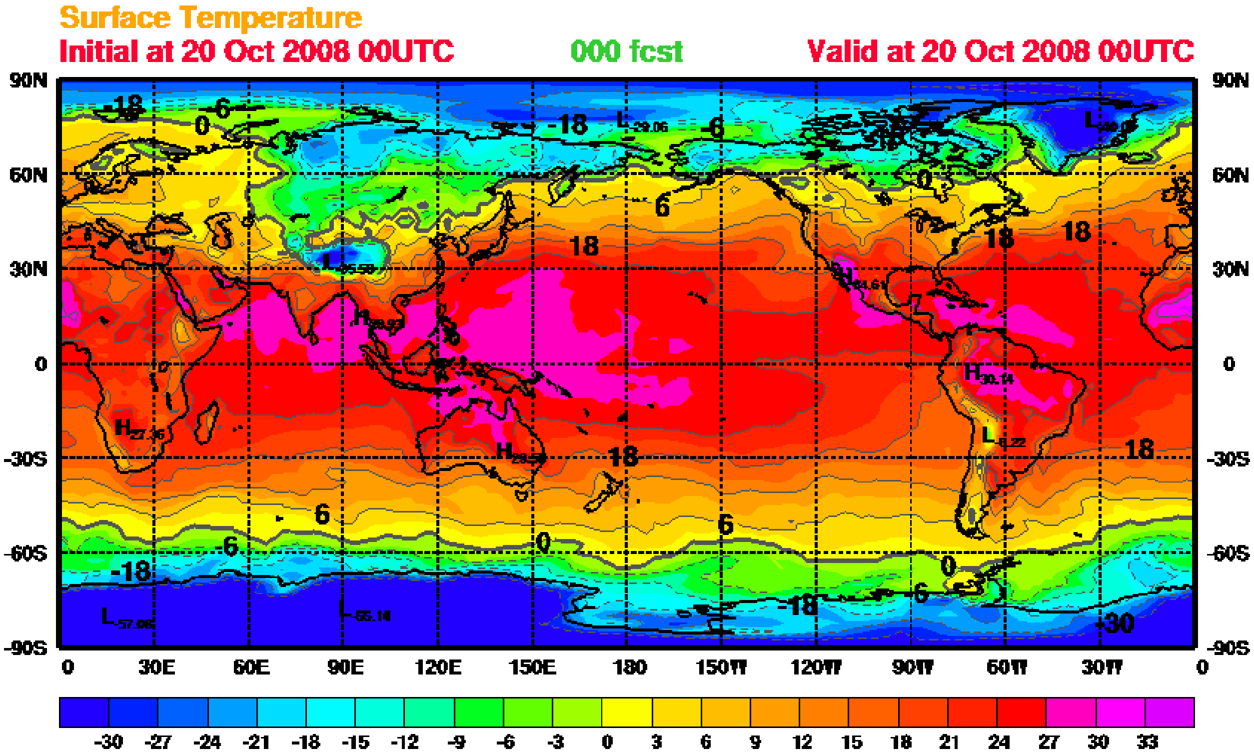
One strategy –
Divide and Conquer

⇒Optimize energy landscape in different regions
⇒ Parallelize structure folding





16,000
14,000
12,000
10,000
8,000
2004
2005
2006
2007
2008
Volume (millions)
750M
500M
250M



Surface Temperature
Initial at 20 Oct 2008 00UTC
000 fcst
Valid at 20 Oct 2008 00UTC
90N
60N
30N
0
-30S
-60S
-90S
0
30E
60E
90E
120E
150E
180
150W
120W
90W
60W
30W
0
-30
-27
-24
-21
-18
-15
-12
-9
-6
-3
0
3
6
9
12
15
18
21
24
27
30
33