Parallel Programming
in C with MPI and OpenMP

Michael J. Quinn

Chapter 1
Motivation and History

Outline

- Motivation
- Modern scientific method
- Evolution of supercomputing
- Modern parallel computers
- Seeking concurrency
- Data clustering case study
- Programming parallel computers

Why Faster Computers?

- Solve compute-intensive problems faster
  - Make infeasible problems feasible
  - Reduce design time
- Solve larger problems in same amount of time
  - Improve answer's precision
  - Reduce design time
- Gain competitive advantage

Definitions

- Parallel computing
  - Using parallel computer to solve single problems faster
- Parallel computer
  - Multiple-processor system supporting parallel programming
- Parallel programming
  - Programming in a language that supports concurrency explicitly

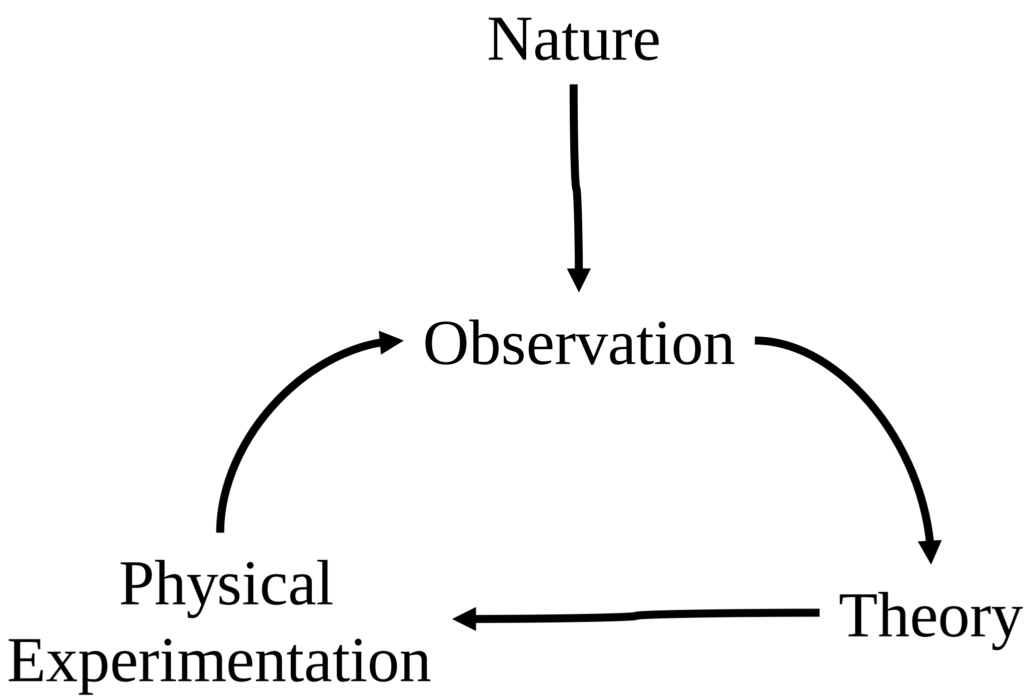
Why MPI?

- MPI = “Message Passing Interface”
- Standard specification for message-passing libraries
- Libraries available on virtually all parallel computers
- Free libraries also available for networks of workstations or commodity clusters

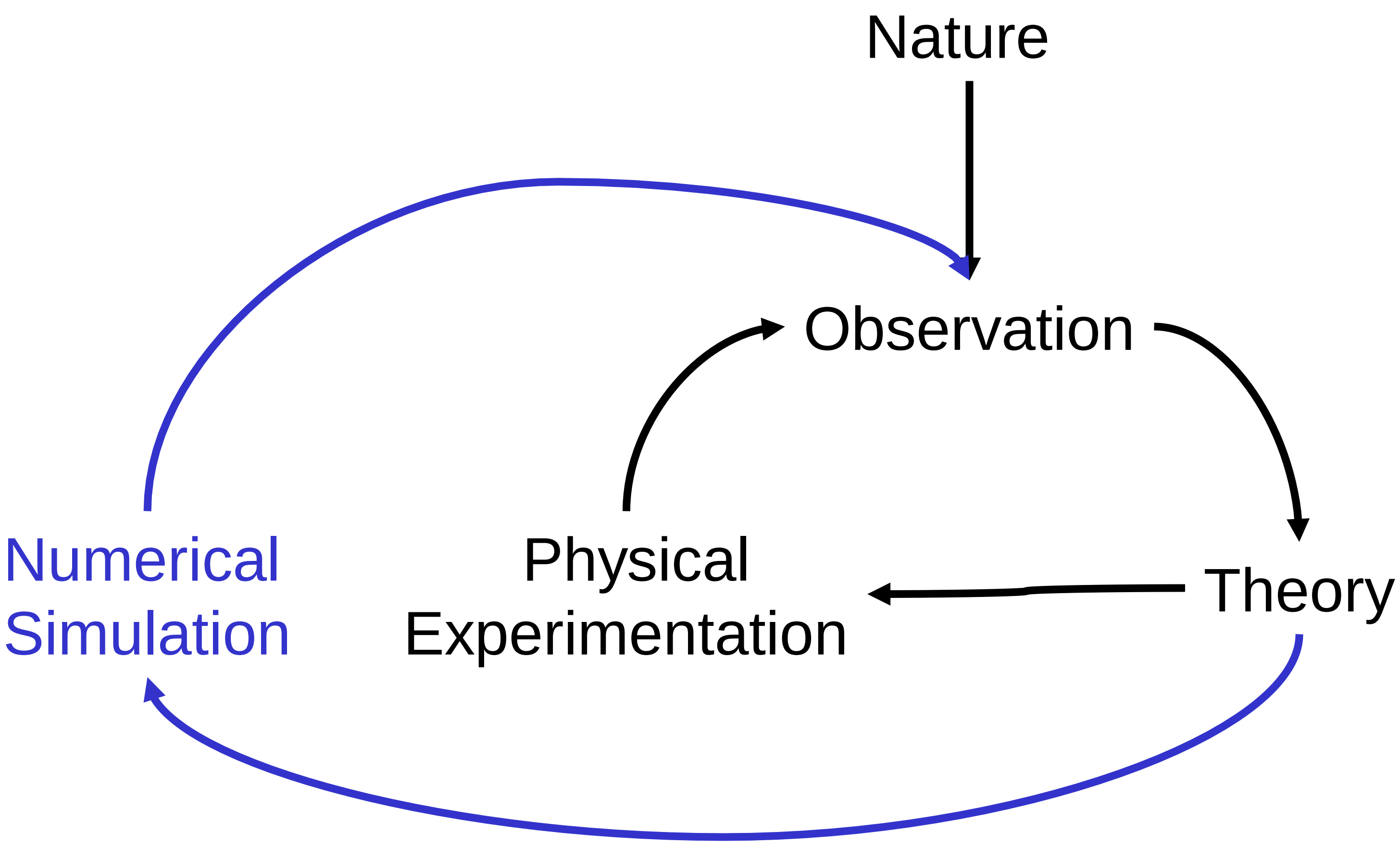
Why OpenMP?

- OpenMP an application programming interface (API) for shared-memory systems
- Supports higher performance parallel programming of symmetrical multiprocessors

Classical Science



Modern Scientific Method



Evolution of Supercomputing

- World War II
  - Hand-computed artillery tables
  - Need to speed computations
  - ENIAC
- Cold War
  - Nuclear weapon design
  - Intelligence gathering
  - Code-breaking

Supercomputer

- General-purpose computer
- Solves individual problems at high speeds, compared with contemporary systems
- Typically costs $10 million or more
- Traditionally found in government labs

Commercial Supercomputing

- Started in capital-intensive industries
  - Petroleum exploration
  - Automobile manufacturing
- Other companies followed suit
  - Pharmaceutical design
  - Consumer products

50 Years of Speed Increases

Today

> 1 trillion flops

One Billion Times Faster!

ENIAC

350 flops

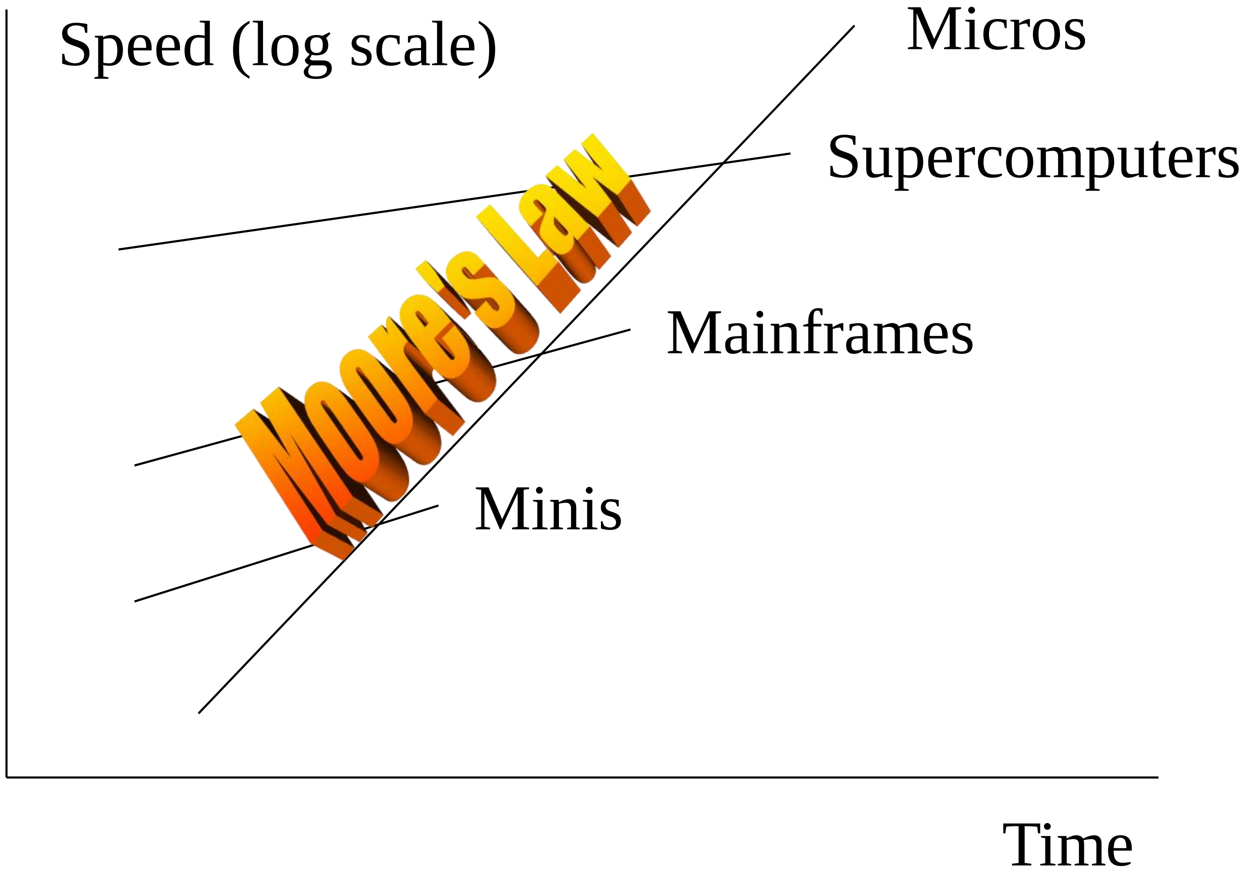
CPUs 1 Million Times Faster

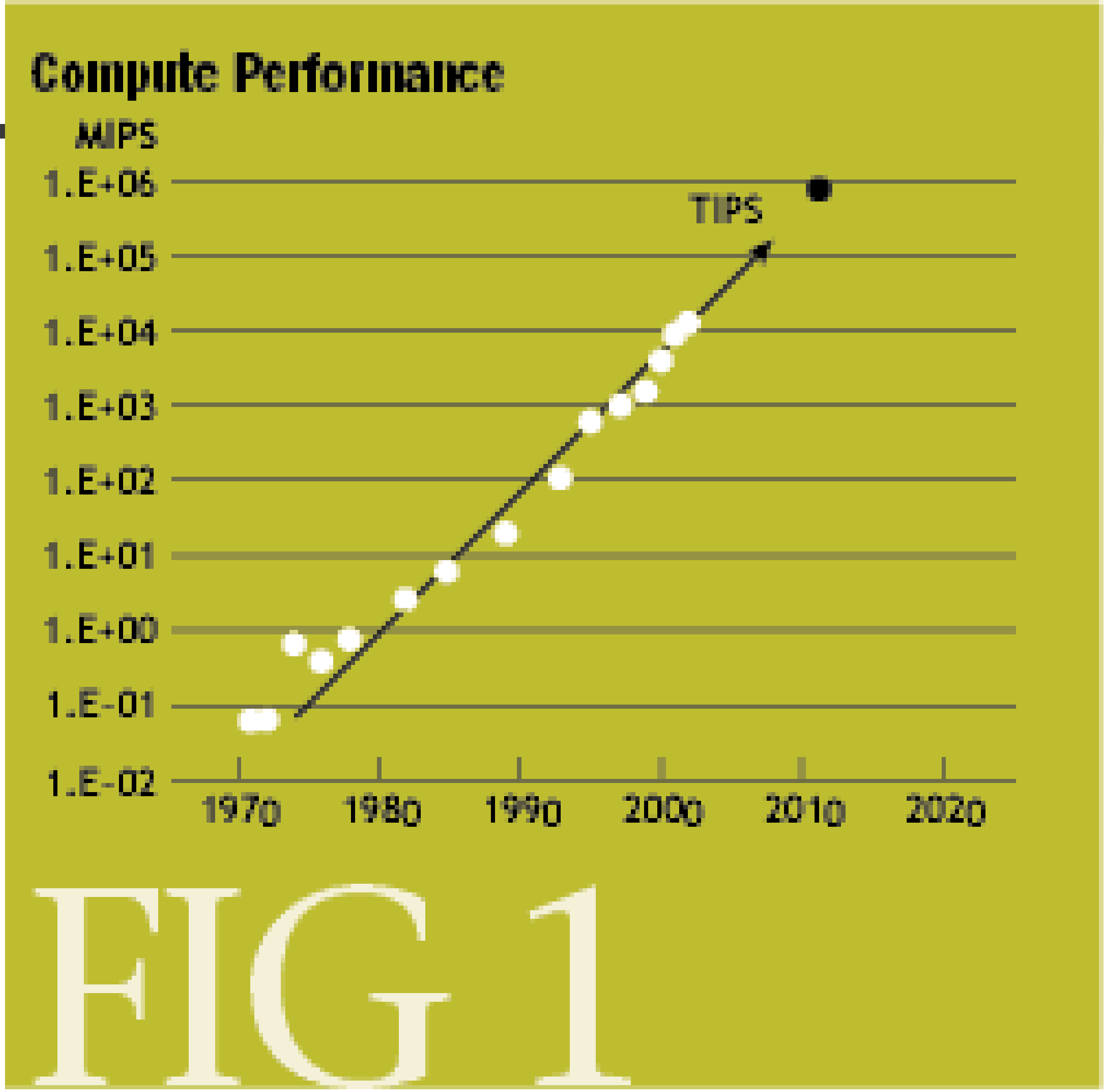
- Faster clock speeds
- Greater system concurrency
  - Multiple functional units
  - Concurrent instruction execution
  - Speculative instruction execution

Systems 1 Billion Times Faster

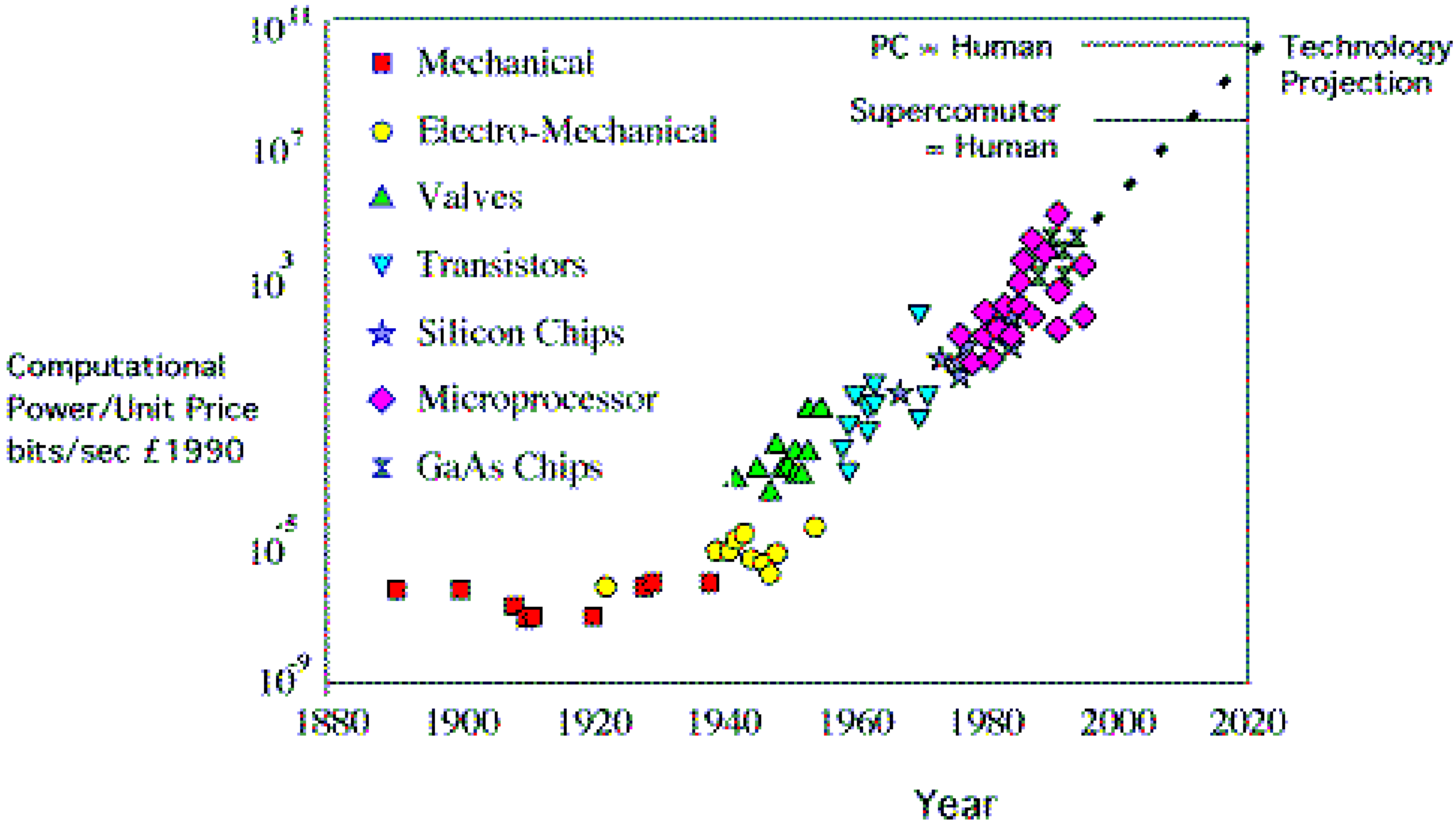
- Processors are 1 million times faster
- Combine thousands of processors
- Parallel computer
  - Multiple processors
  - Supports parallel programming
- Parallel computing = Using a parallel computer to execute a program faster

Microprocessor Revolution

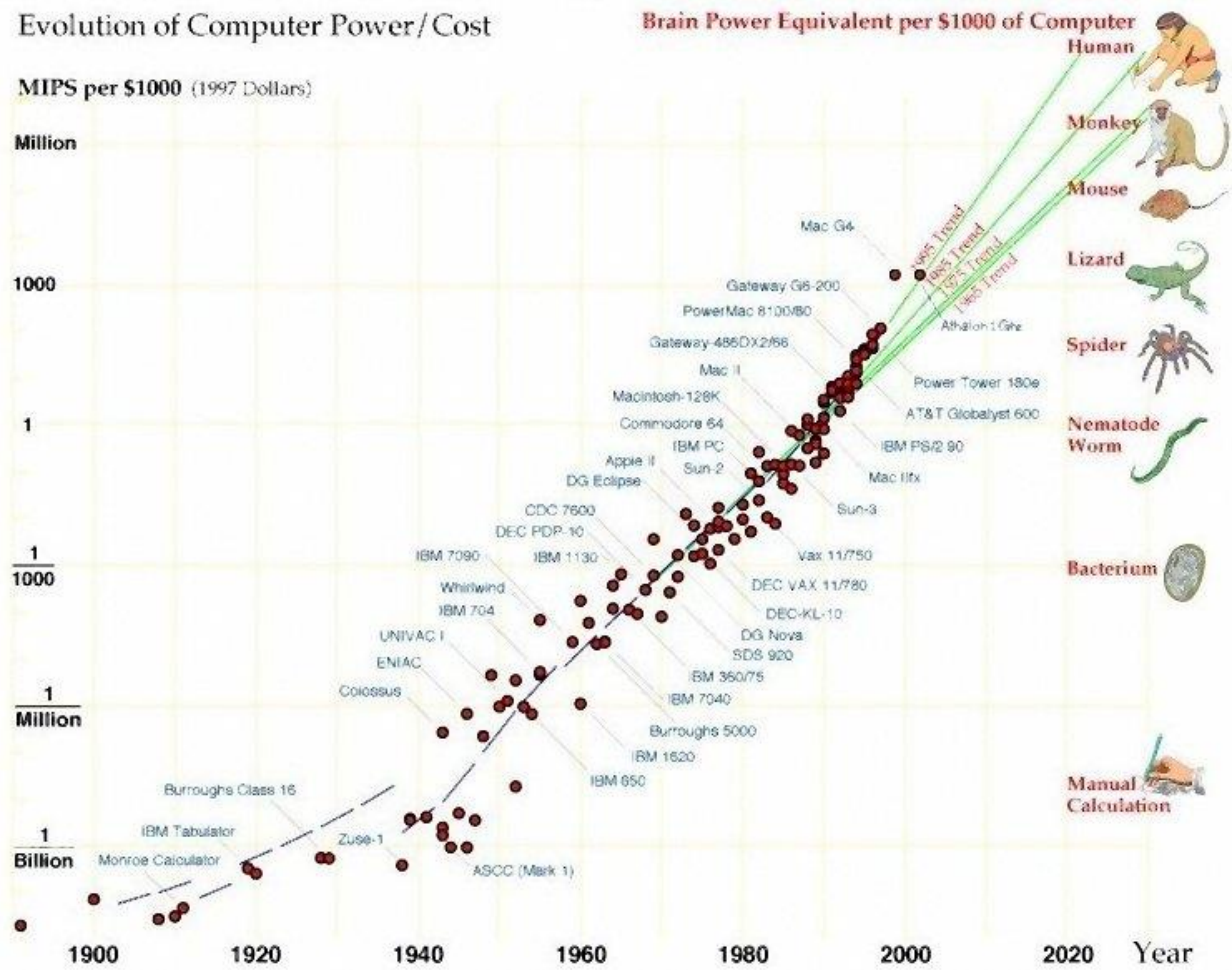




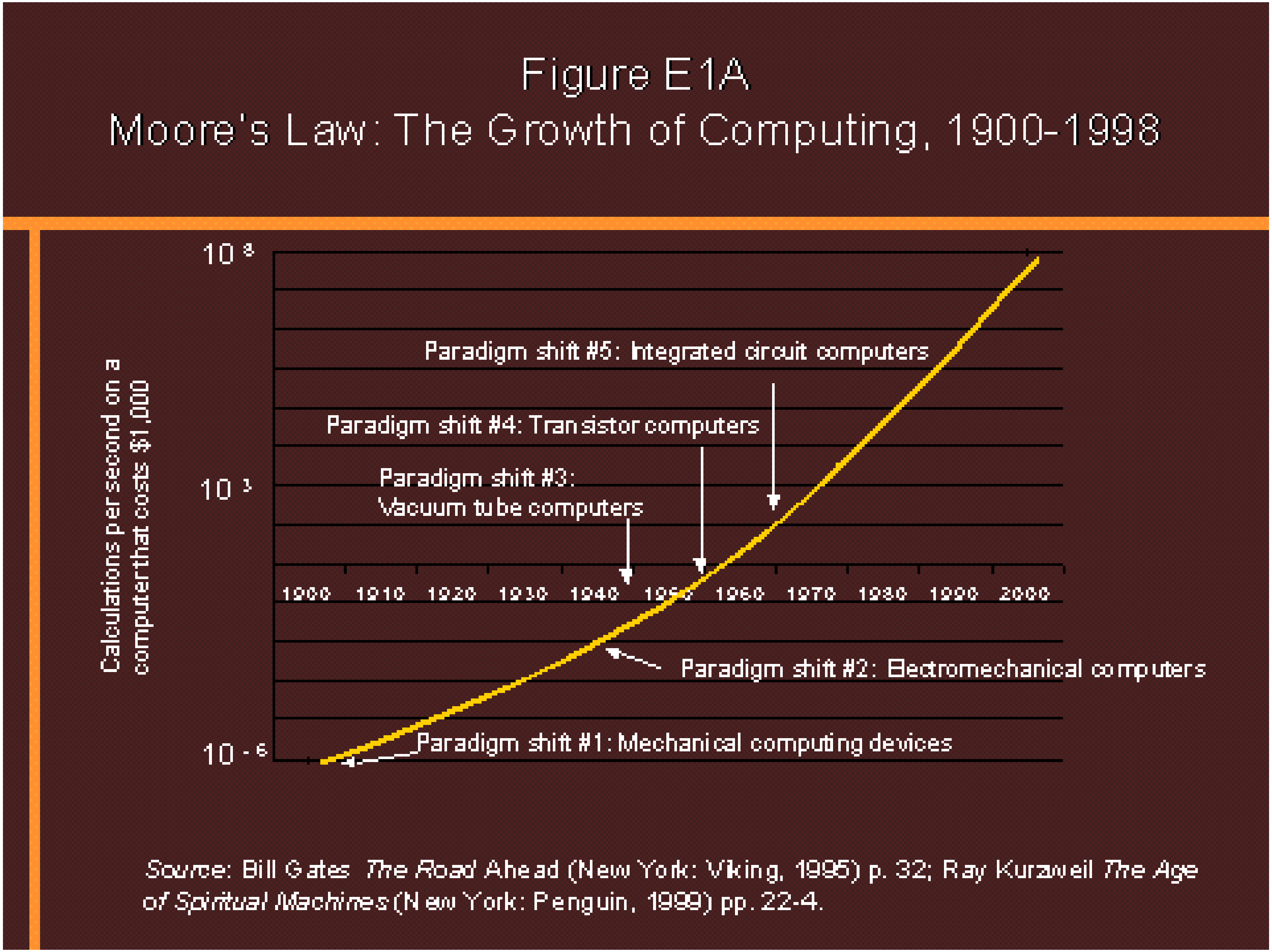
http://www.acmqueue.org/figures/issue007/borkarfig1.gif



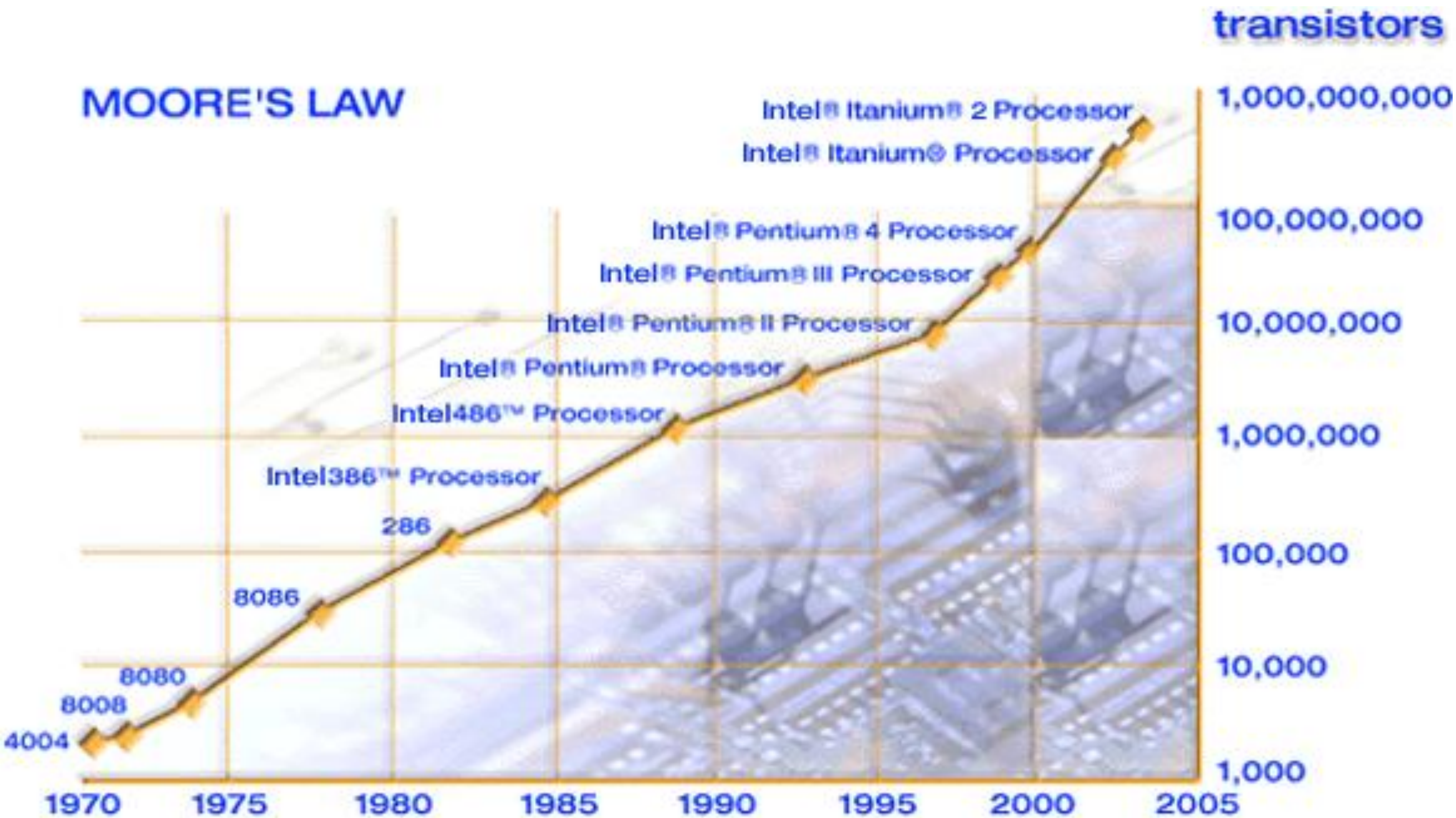
http://www.cochrane.org.uk/opinion/papers/images/exponentialeducation4.gif



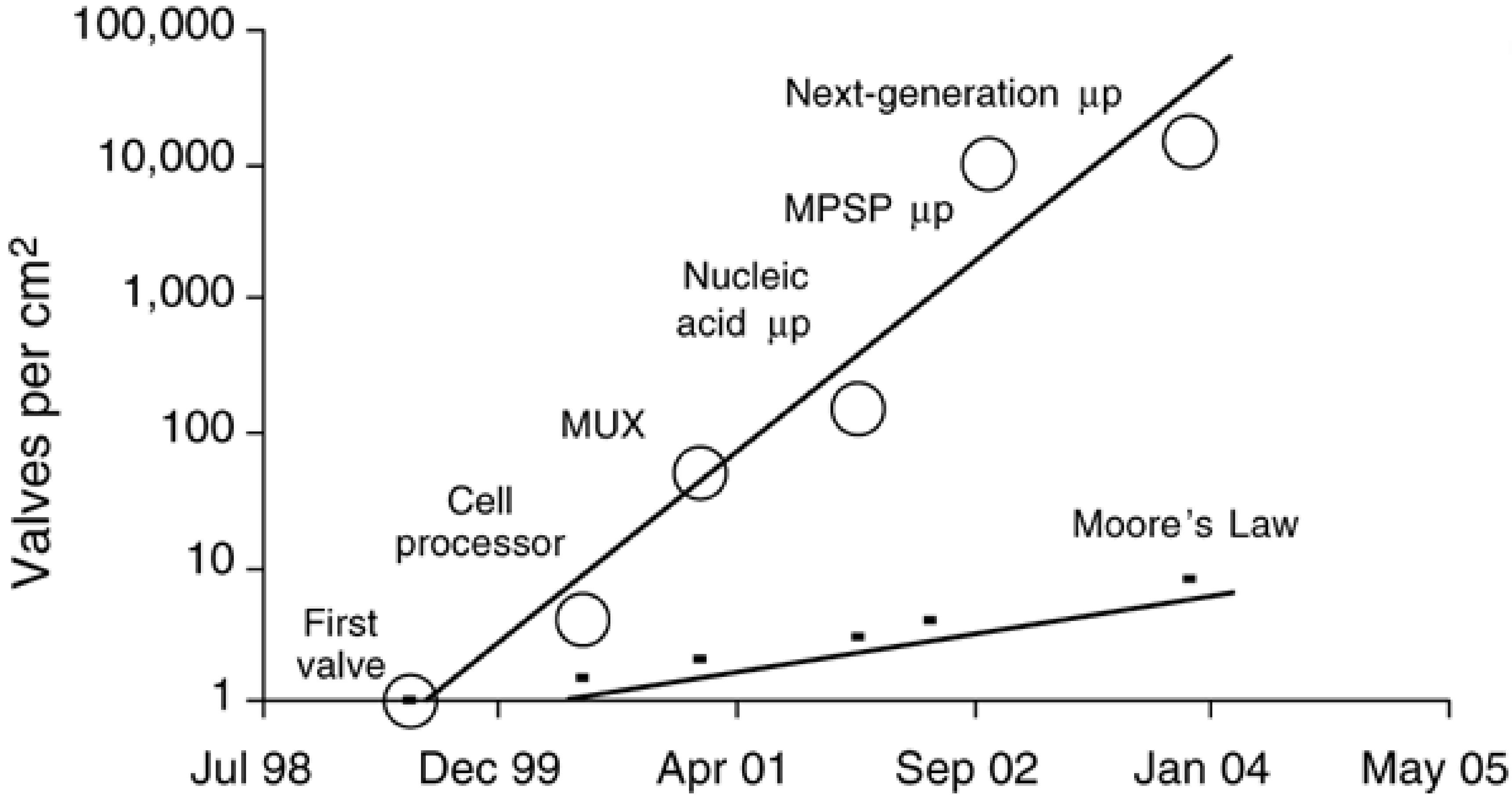
http://www.rubbertreeplant.co.uk/images/kmscreenshots/power_075.jpg



http://www.wadsworth.com/sociology_d/templates/student_resources/0155072129_brym/chapter01/FigureE1A.gif



http://www.intel.com/research/silicon/images/MooresLawgraph3.gif



http://www.nature.com/nbt/journal/v21/n10/images/nbt871-F4.gif

Modern Parallel Computers

- Caltech's Cosmic Cube (Seitz and Fox)
- Commercial copy-cats
  - nCUBE Corporation
  - Intel's Supercomputer Systems Division
  - Lots more
- Thinking Machines Corporation

Copy-cat Strategy

- Microprocessor
  - 1% speed of supercomputer
  - 0.1% cost of supercomputer
- Parallel computer = 1000 microprocessors
  - 10 x speed of supercomputer
  - Same cost as supercomputer

Why Didn't Everybody Buy One?

- Supercomputer ≠ Σ CPUs
  - Computation rate ≠ throughput
  - Inadequate I/O
- Software
  - Inadequate operating systems
  - Inadequate programming environments

After the “Shake Out”

- IBM
- Hewlett-Packard
- Silicon Graphics
- Sun Microsystems

Commercial Parallel Systems

- Relatively costly per processor
- Primitive programming environments
- Focus on commercial sales
- Scientists looked for alternative

Beowulf Concept

- NASA (Sterling and Becker)
- Commodity processors
- Commodity interconnect
- Linux operating system
- Message Passing Interface (MPI) library
- High performance/$ for certain applications

Advanced Strategic Computing Initiative

- U.S. nuclear policy changes
  - Moratorium on testing
  - Production of new weapons halted
- Numerical simulations needed to maintain existing stockpile
- Five supercomputers costing up to $100 million each

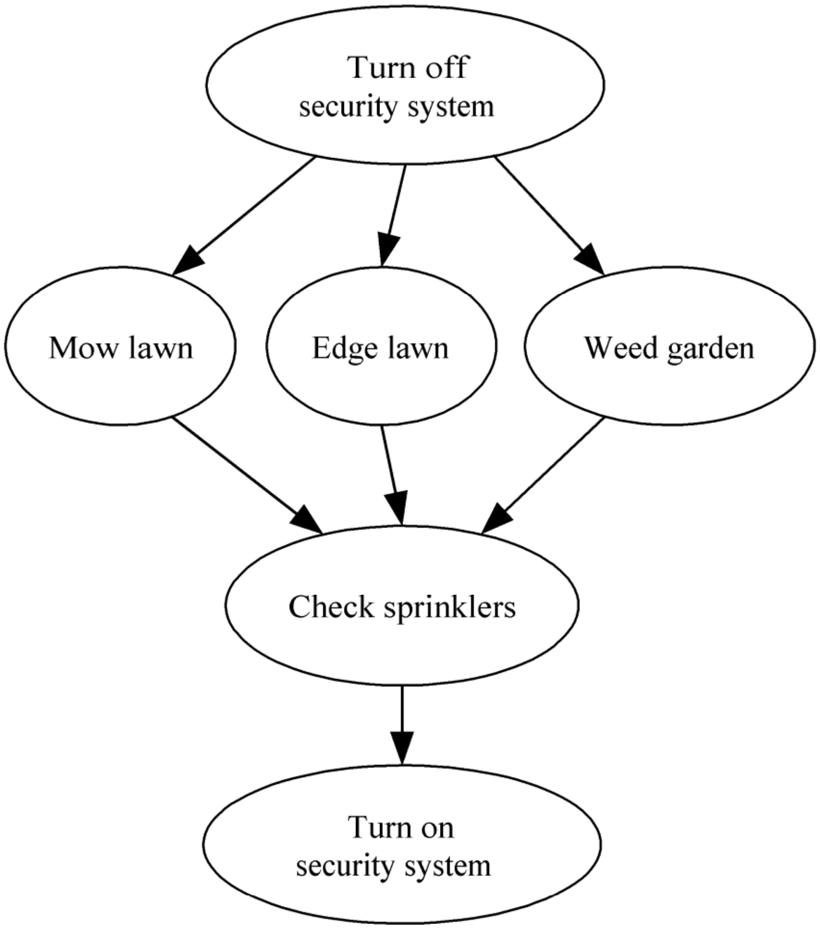
ASCI White (10 teraops/sec)

Seeking Concurrency

- Data dependence graphs
- Data parallelism
- Functional parallelism
- Pipelining

Data Dependence Graph

- Directed graph
- Vertices = tasks
- Edges = dependences



Data Parallelism

- Independent tasks apply same operation to different elements of a data set

```
for i ← 0 to 99 do
    a[i] ← b[i] + c[i]
endfor
```

- Okay to perform operations concurrently

Functional Parallelism

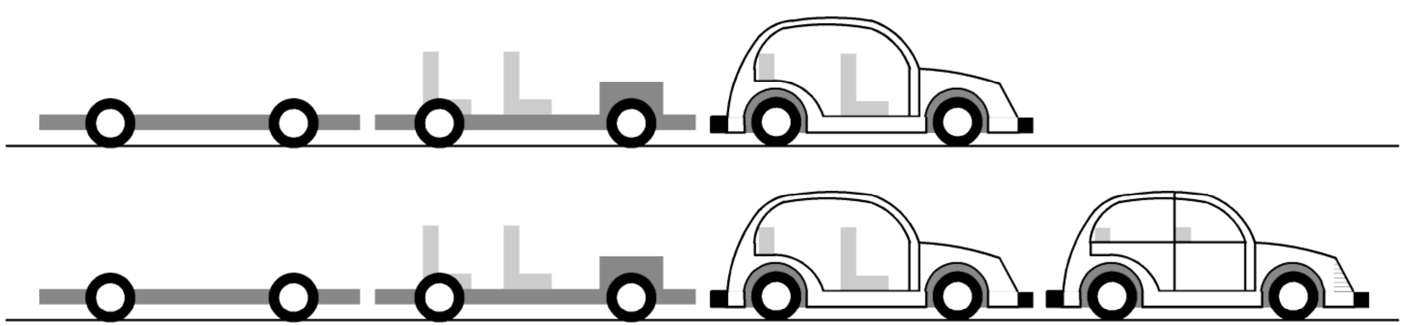
- Independent tasks apply different operations to different data elements

$$a \leftarrow 2$$
$$b \leftarrow 3$$
$$m \leftarrow (a + b) / 2$$
$$s \leftarrow (a^2 + b^2) / 2$$
$$v \leftarrow s - m^2$$

- First and second statements
- Third and fourth statements

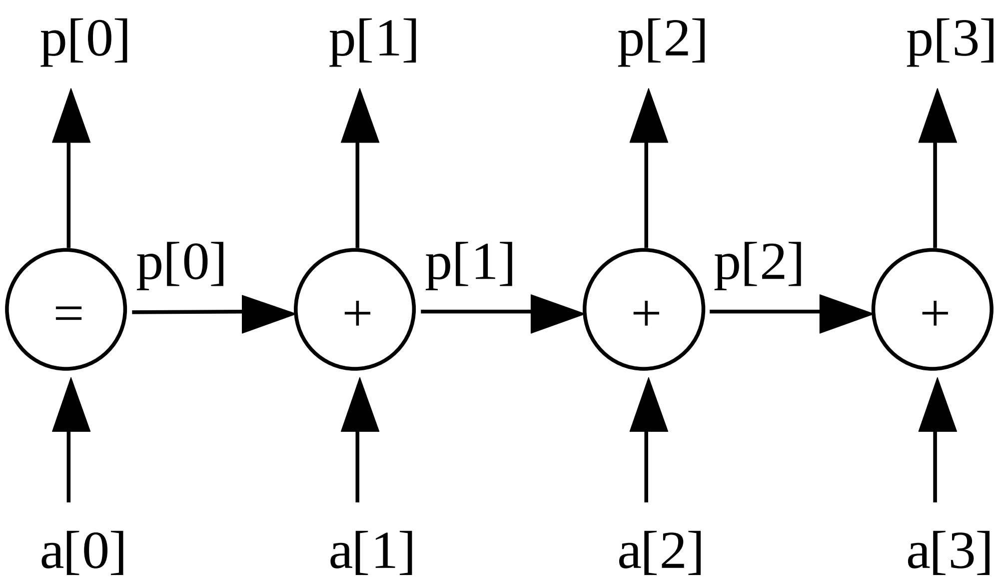
Pipelining

- Divide a process into stages
- Produce several items simultaneously



Partial Sums Pipeline

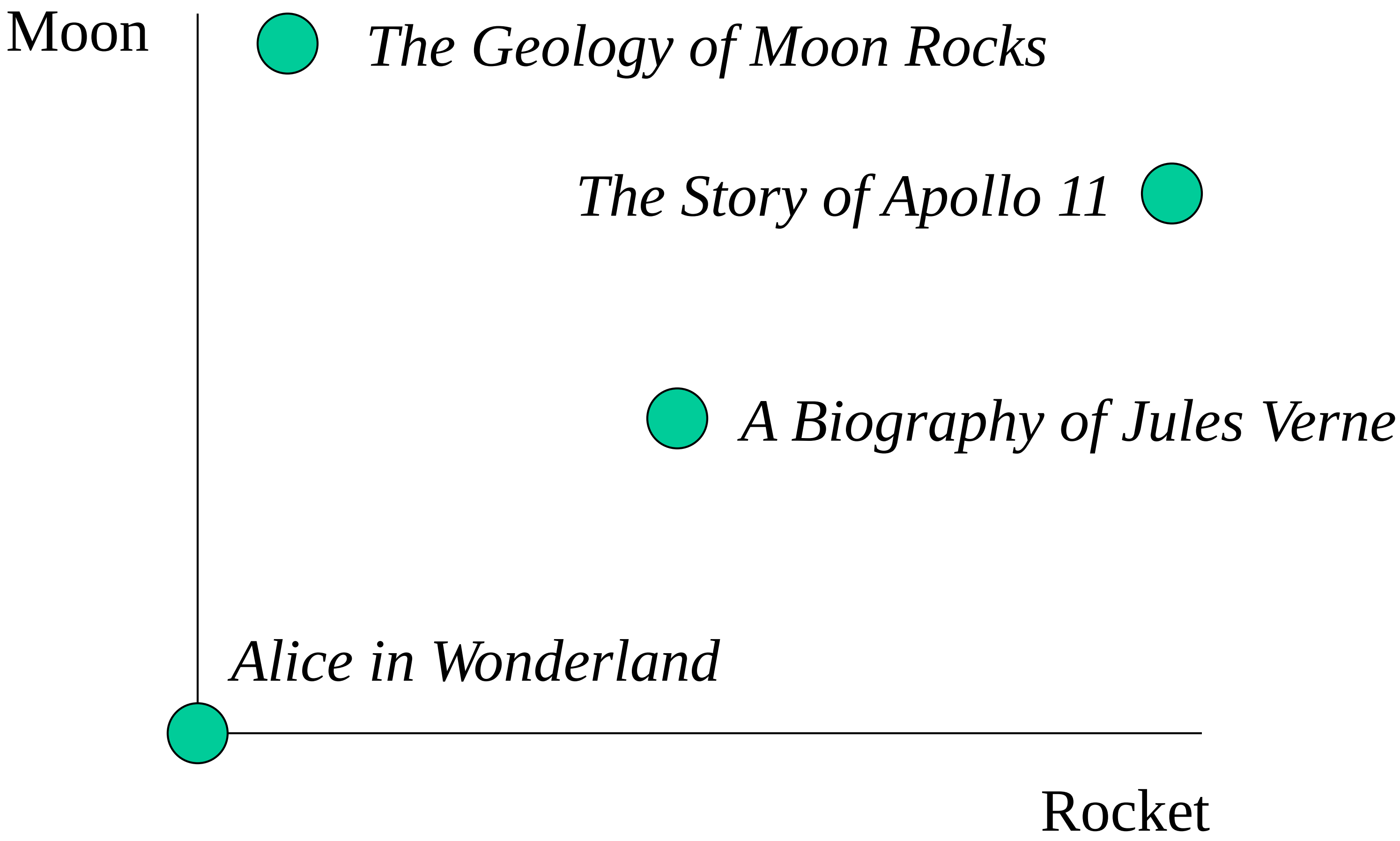
```
p[0] ← a[0]
for i ← 1 to 3 do
   p[i] ← p[i-1] + a[i]
endfor
```



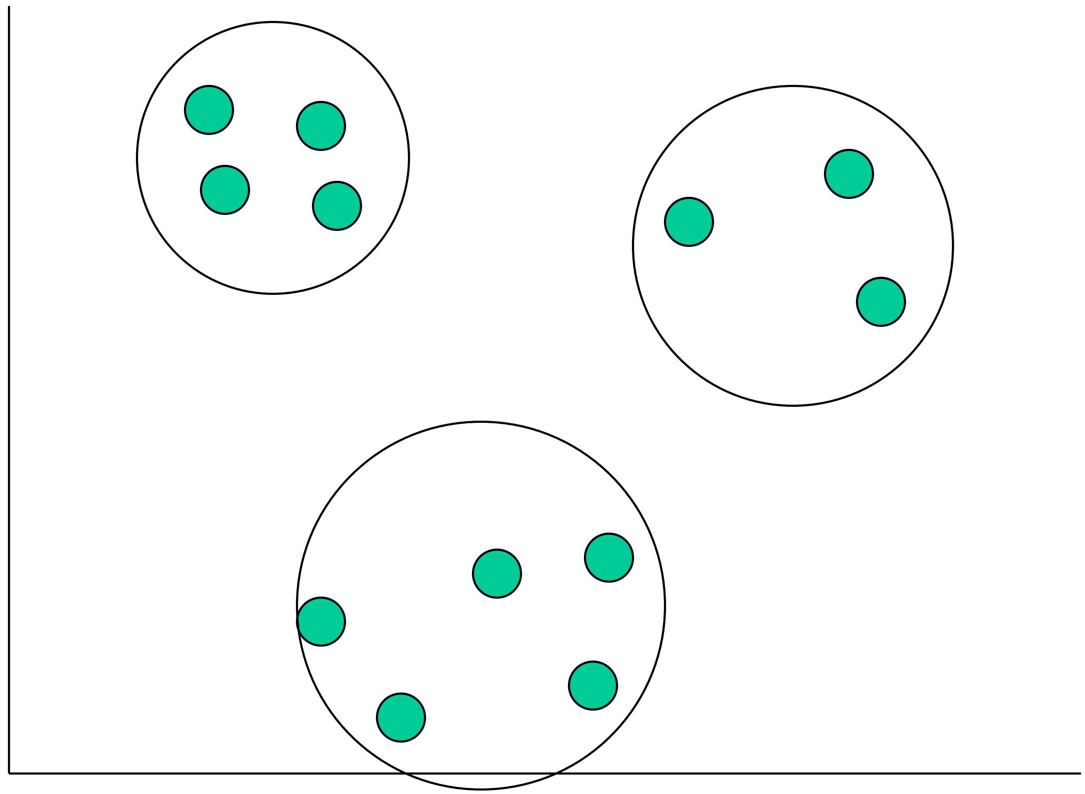
Data Clustering

- Data mining
  - looking for meaningful patterns in large data sets
- Data clustering
  - organizing a data set into clusters of “similar” items
- Data clustering can speed retrieval of related items

Document Vectors



Document Clustering



Clustering Algorithm

- Compute document vectors
- Choose initial cluster centers
- Repeat
  - Compute performance function
  - Adjust centers
- Until function value converges or max iterations have elapsed
- Output cluster centers

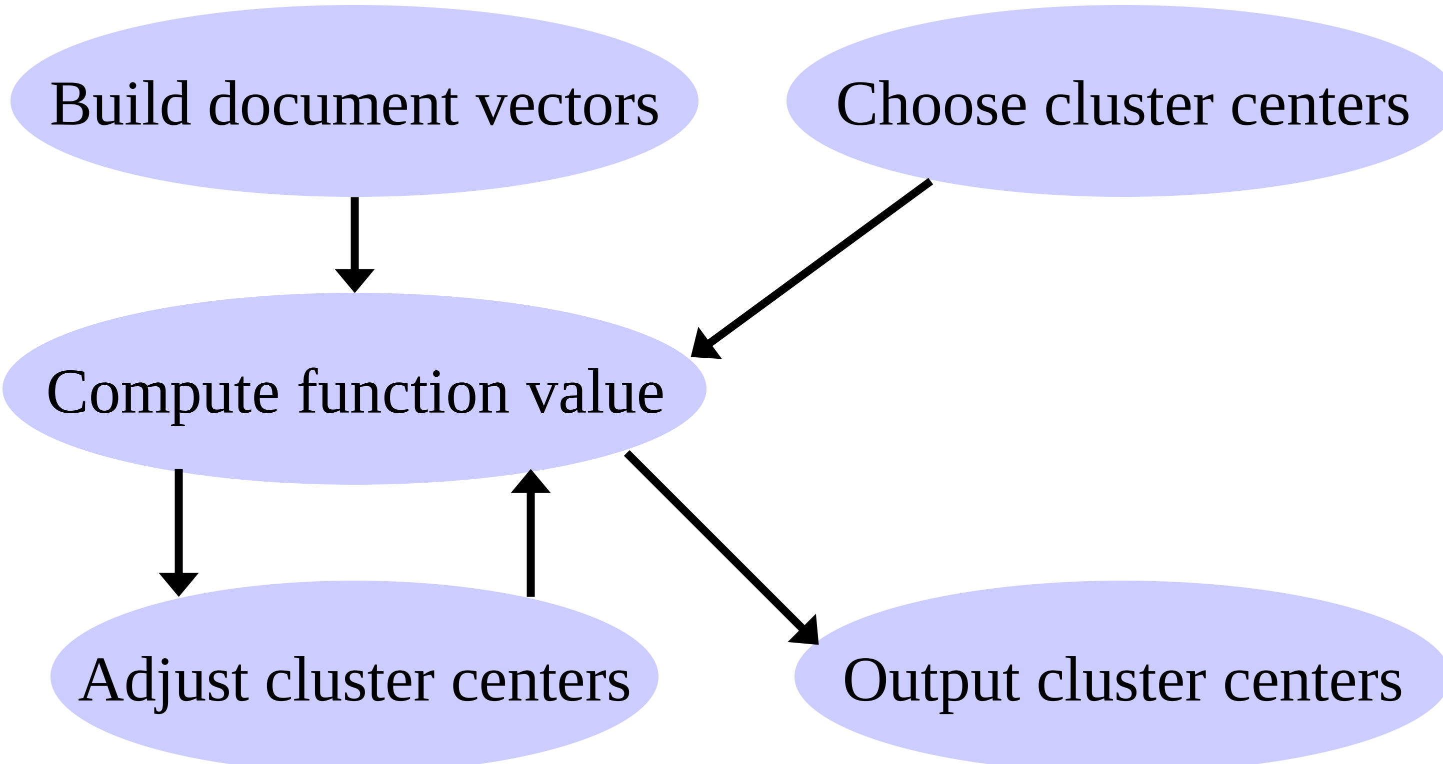
Data Parallelism Opportunities

- Operation being applied to a data set
- Examples
  - Generating document vectors
  - Finding closest center to each vector
  - Picking initial values of cluster centers

Functional Parallelism Opportunities

- Draw data dependence diagram
- Look for sets of nodes such that there are no paths from one node to another

Data Dependence Diagram



Programming Parallel Computers

- Extend compilers: translate sequential programs into parallel programs
- Extend languages: add parallel operations
- Add parallel language layer on top of sequential language
- Define totally new parallel language and compiler system

1. Extend Compilers

- Parallelizing compiler
  - Detect parallelism in sequential program
  - Produce parallel executable program
- Focus on making Fortran programs parallel

1. Extend Compilers (cont.)

- Advantages
  - Can leverage millions of lines of existing serial programs
  - Saves time and labor
  - Requires no retraining of programmers
  - Sequential programming easier than parallel programming
- Disadvantages
  - Parallelism may be irretrievably lost when programs written in sequential languages
  - Performance of parallelizing compilers on broad range of applications still up in air

2. Extend Language (Using API)

- Add functions to a sequential language
  - Create and terminate processes
  - Synchronize processes
  - Allow processes to communicate
- Example
  - PVM, MPI standard, OpenMP

2. Extend Language (cont.)

- Advantages
  - Easiest, quickest, and least expensive
  - Allows existing compiler technology to be leveraged
  - New libraries can be ready soon after new parallel computers are available
- Disadvantages
  - Lack of compiler support to catch errors
  - Easy to write programs that are difficult to debug

3. Add a Parallel Programming Layer

- Lower layer
  - Core of computation
  - Process manipulates its portion of data to produce its portion of result
- Upper layer
  - Creation and synchronization of processes
  - Partitioning of data among processes
- A few research prototypes have been built based on these principles – CODE, Hence

4. Create a Parallel Language

- Develop a parallel language “from scratch”
  - Erlang, occam is an example
- Add parallel constructs to an existing language
  - Fortran 90
  - High Performance Fortran
  - C*

4. New Parallel Languages (cont.)

- Advantages
  - Allows programmer to communicate parallelism to compiler
  - Improves probability that executable will achieve high performance
- Disadvantages
  - Requires development of new compilers
  - New languages may not become standards
  - Programmer resistance

Current Status

- Low-level approach is most popular
  - Augment existing language with low-level parallel constructs
  - MPI and OpenMP are examples
- Advantages of low-level approach
  - Efficiency
  - Portability
- Disadvantage: More difficult to program and debug

Summary (1/2)

- High performance computing
  - U.S. government
  - Capital-intensive industries
  - Many companies and research labs
- Parallel computers
  - Commercial systems
  - Commodity-based systems

Summary (2/2)

- Power of CPUs keeps growing exponentially
- Parallel programming environments changing very slowly
- Two standards have emerged
  - MPI library, for processes that do not share memory
  - OpenMP directives, for processes that do share memory

Exercise

- 1.10
- 1.13