Cornell University
Computing and Information Science

CS 5150 Software Engineering
15. Performance

William Y. Arms

Performance of Computer Systems

In most computer systems

The cost of people is much greater than the cost of hardware

Yet performance is important

A single bottleneck can slow down an entire system

Future loads may be much greater than predicted

When Performance Matters

- **Real time systems** when computation must be fast enough to support the service provided, e.g., fly-by wire control systems have tight response time requirements.
- **Very large computations** where elapsed time may be measured in days, e.g., calculation of weather forecasts must be fast enough for the forecasts to be useful.
- **User interfaces** where humans have high expectations, e.g., mouse tracking must appear instantaneous.
- **Transaction processing** where staff need to be productive and customers not annoyed by delays, e.g., airline check-in.

High-Performance Computing

High-performance computing:

- Large data collections (e.g., Amazon)
- Huge numbers of users (e.g., Google)
- Large computations (e.g., weather forecasting)

Must balance cost of hardware against cost of software development

- Some configurations are very difficult to program and debug
- Sometimes it is possible to isolate applications programmers from the system complexities

Performance Challenges for all Software Systems

Tasks

- Predict performance problems before a system is implemented.
- Design and build a system that is not vulnerable to performance problems.
- Identify causes and fix problems after a system is implemented.

Performance Challenges for all Software Systems

Basic techniques

- **Understand** how the underlying **hardware** and **networks** components interact with the **software** when executing the system.
- For each subsystem calculate the **capacity** and **load**. The capacity is a combination of the hardware and the software architecture.
- Identify subsystems that are near **peak capacity**.

Example

Calculations indicate that the capacity of a search system is 1,000 searches per second. What is the anticipated peak demand?

Interactions between Hardware and Software

Examples

- In a distributed system, what messages pass between nodes?
- How many times must the system read from disk for a single transaction?
- What buffering and caching is used?
- Are operations in parallel or sequential?
- Are other systems competing for a shared resource (e.g., a network or server farm)?
- How does the operating system schedule tasks?

Look for Bottlenecks

Usually, CPU performance is not the limiting factor.

Hardware bottlenecks

- Reading data from disk
- Shortage of memory (including paging)
- Moving data from memory to CPU
- Network capacity

Inefficient software

- Algorithms that do not scale well
- Parallel and sequential processing

Look for Bottlenecks

CPU performance is a limiting constraint in certain domains, e.g.:

- large data analysis (e.g., searching)
- mathematical computation (e.g., engineering)
- compression and encryption
- multimedia (e.g., video)
- perception (e.g., image processing)

Timescale of Different Components

	Operations per second
CPU instruction:	2,000,000,000
Disk latency:	200
Disk read:	100,000,000 bytes
Network LAN:	10,000,000 bytes

Actual performance may be considerably less than the theoretical peak

Look for Bottlenecks: Utilization

Utilization is the proportion of the capacity of a service that is used **on average**.

$$\textbf{utilization} = \text{proportion of capacity of service that is used}$$

$$= \frac{\text{mean service time for a transaction}}{\text{mean inter-arrival time of transactions}}$$

When the utilization of any hardware component exceeds 0.3, be prepared for congestion.

Peak loads and temporary increases in demand can be much greater than the average.

Predicting System Performance

- Direct measurement on subsystem (benchmark)
- Mathematical models (queueing theory)
- Simulation

All require detailed understanding of the interaction between software and hardware systems.

Mathematical Models

Queueing theory

Good estimates of congestion can be made for single-server queues with:

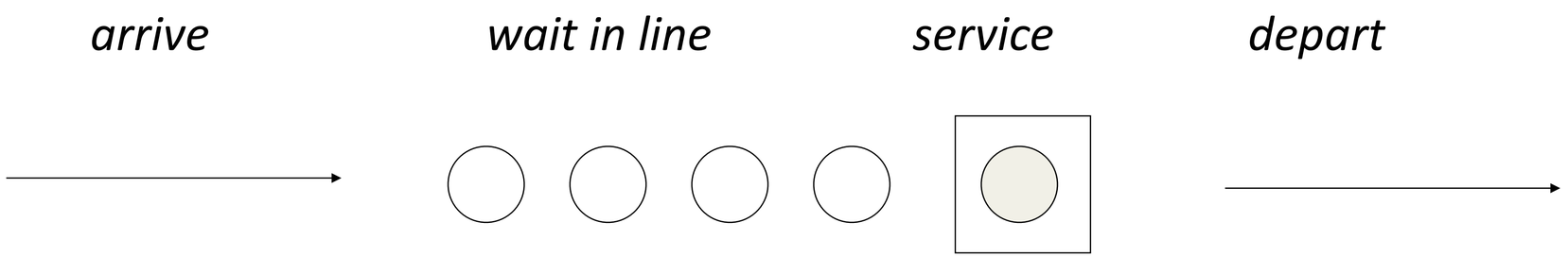
- arrivals that are independent, random events (Poisson process)
- service times that follow families of distributions (e.g., negative exponential, gamma)

Many of the results can be extended to multi-server queues.

Much of the early work in queueing theory by Erlang was to model congestion in telephone networks.

Mathematical Models: Queues

Single server queue

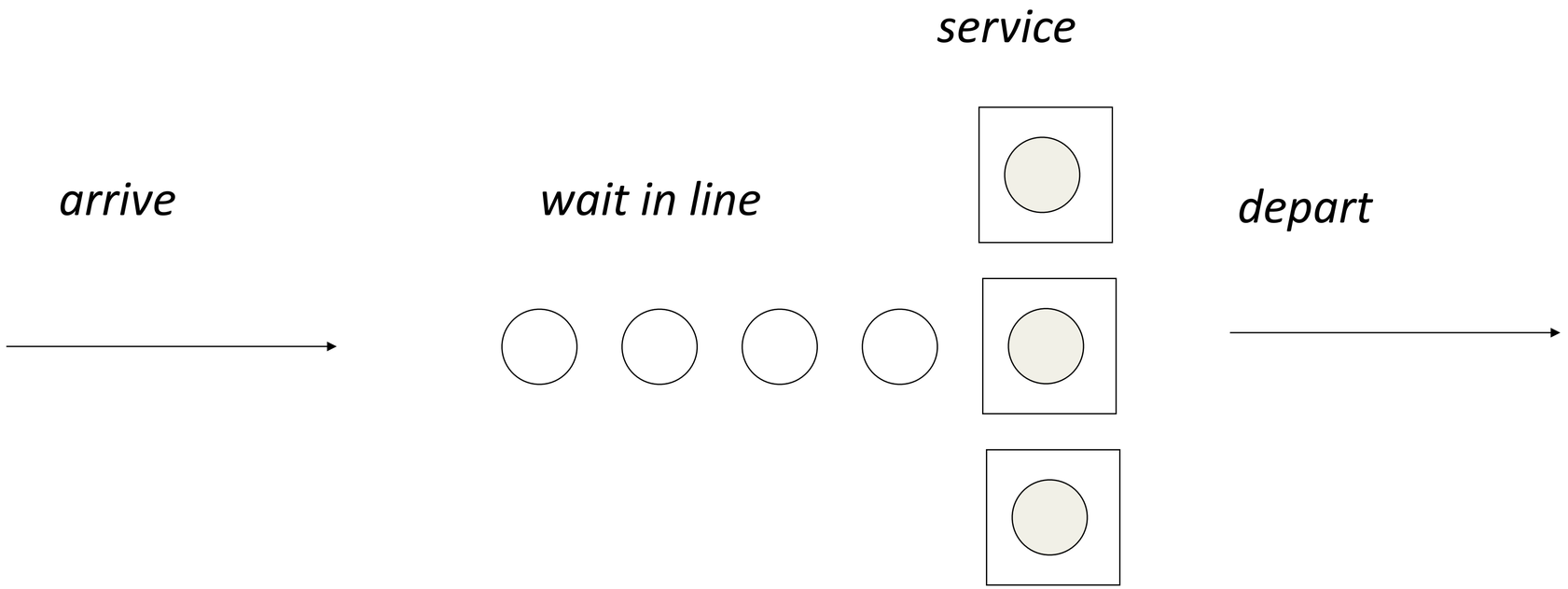


Examples

- Requests to read from a disk (with no buffering or other optimization)
- Customers waiting for check in at an airport, with a single check-in desk

Queues

Multi-server queue



Examples

- Tasks being processed on a computer with several processors
- Customers waiting for check in at an airport, with a several check-in desks

Techniques: Simulation

Build a computer program that models the system as set of states and events.

```
advance simulated time
    determine which events occurred
    update state and event list
repeat
```

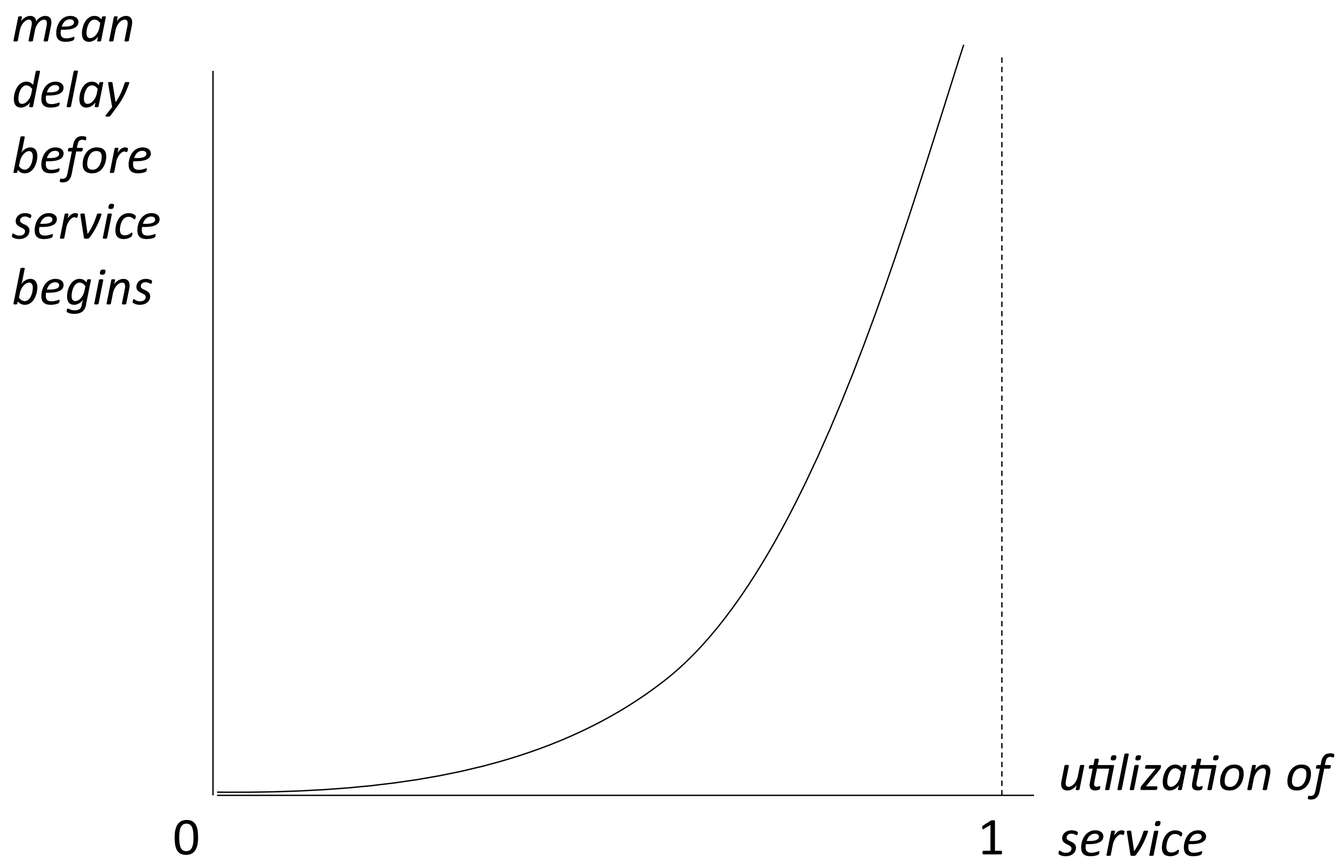
Discrete time simulation: Time is advanced in fixed steps (e.g., 1 millisecond)

Next event simulation: Time is advanced to next event

Events can be simulated by random variables (e.g., arrival of next customer, completion of disk latency), or by using data collected from an operational system.

Behavior of Queues: Utilization

The exact shape of the curve depends on the type of queue (e.g., single server) and the statistical distributions of arrival times and service times.



Measurements on Operational Systems

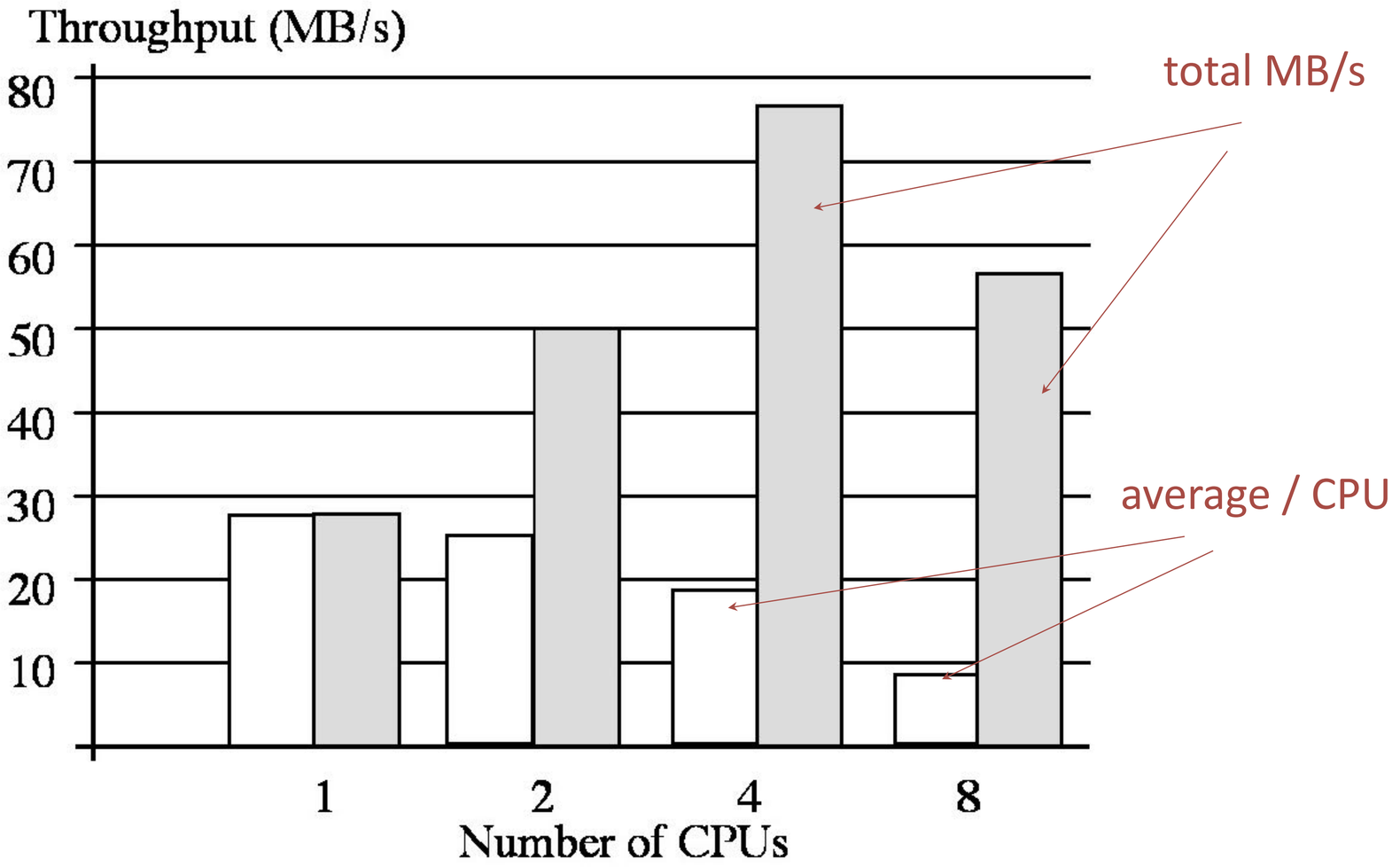
Measurements on operational systems

- Benchmarks: Run system on standard problem sets, sample inputs, or a simulated load on the system.
- Instrumentation: Clock specific events.

If you have any doubt about the performance of part of a system, experiment with a simulated load.

Example: Web Laboratory

Benchmark: throughput v. number of CPUs on a symmetric multiprocessor



Case Study: Performance of Disk Farm

When many transaction use a disk farm, each transaction must:

wait for specific disk
wait for I/O channel
send signal to move heads on disk
wait for I/O channel
pause for disk rotation (latency)
read data

Close agreement between: results from queuing theory, simulation, and direct measurement (within 15%).

Fixing Bad Performance

If a system performs badly, begin by identifying the cause:

Instrumentation. Add timers to the code. Often this will reveal that delays are centered in a specific part of the system.

Test loads. Run the system with varying loads, e.g., high transaction rates, large input files, many users, etc. This may reveal the characteristics of when the system runs badly.

Design and code reviews. Team review of system design, program design, and suspect sections of code. This may reveal an algorithm that is running very slowly, e.g., a sort, locking procedure, etc.

Find the underlying cause and fix it or the problem will return!

Predicting Performance Change: Moore's Law

Original version:

The density of transistors in an integrated circuit will double every year. (Gordon Moore, Intel, 1965)

Current version:

Cost/performance of silicon chips doubles every 18 months.

Moore's Law: Rules of Thumb

Planning assumptions

Silicon chips: cost/performance improves 30% / year
in 12 years = 20:1
in 24 years = 500:1

Magnetic media: cost/performance improves 40% / year
in 12 years = 50:1
in 24 years = 3,000:1

These assumptions are conservative. During some periods, the increases have been considerably faster.

Recently, the rate of performance increase in individual components, such as CPUs, has slowed down, but the overall rate of increase has been maintained by placing many CPU cores on a single chip.

Moore's Law and System Design

Feasibility study: *Production use:* *Withdrawn from production:*	2013	2016	2026
Processor speeds	1	2.2	30
Memory sizes:	1	2.2	30
Disk capacity:	1	2.7	80
System cost:	1	0.4	0.03

Moore's Law Example

Will this be a typical laptop?

	2017	2027
Processors	2 x 2.5 GHz	8 x 10 GHz
Memory	8 GB	200 GB
Disc	500 GB	15 TB
Network	1 Gb/s	25 Gb/s

or 100 processors?

Surely there will be some fundamental changes in how this this power is packaged and used.

Parkinson's Law

Original:

Work expands to fill the time available. (C. Northcote Parkinson)

Software development version:

(a) Demand will expand to use all the hardware available.

(b) Low prices will create new demands.

(c) Your software will be used on equipment that you have not envisioned.

False Assumptions from the Past

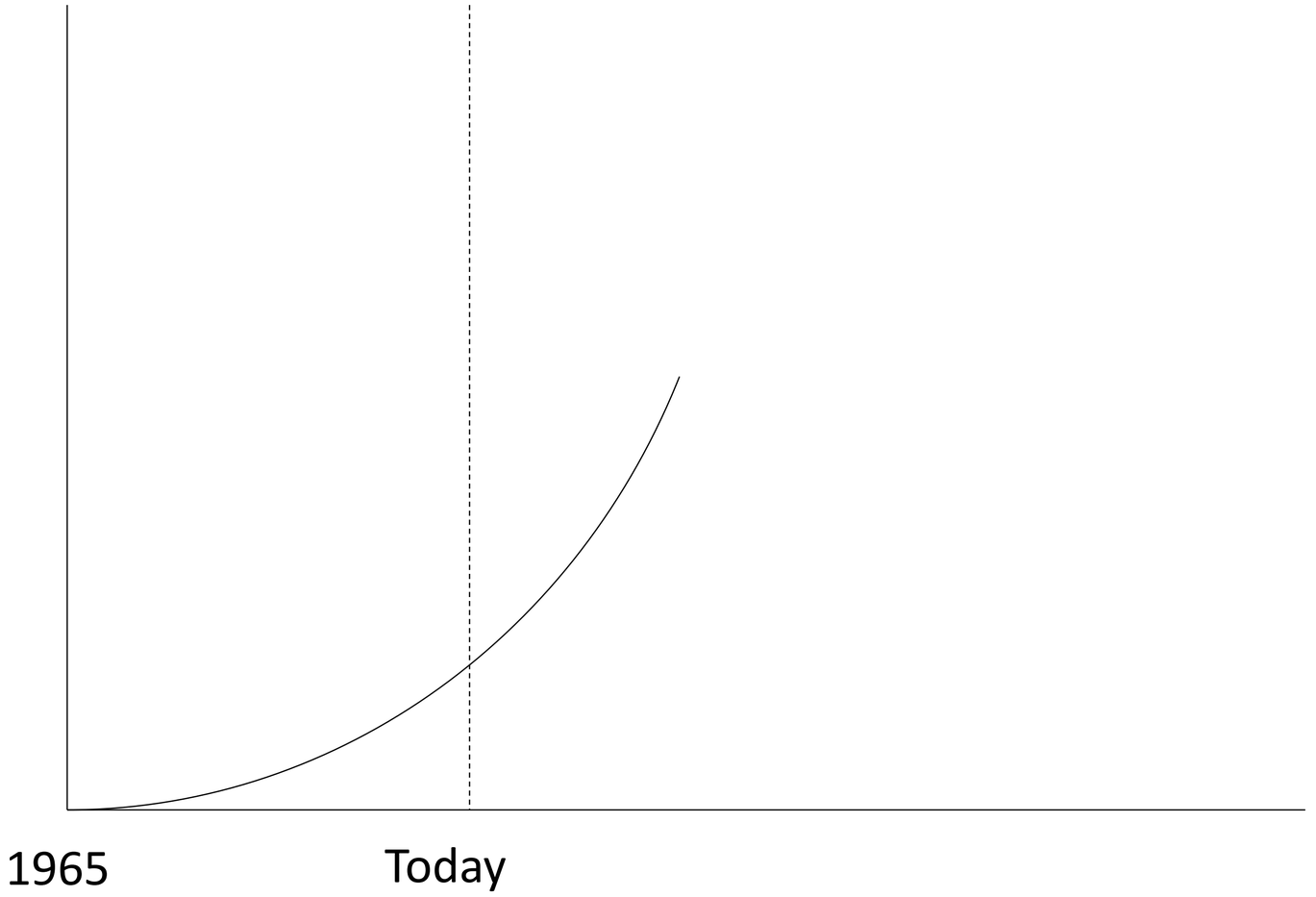
Be careful about the assumptions that you make

Here are some past assumptions that caused problems:

- Unix file system will never exceed 2 GBytes (2^{32} bytes).
- AppleTalk networks will never have more than 256 hosts (2^{8} bits).
- GPS software will not last more than 1024 weeks.
- Two bytes are sufficient to represent a year (Y2K bug).

etc., etc., ...

Moore's Law and the Long Term



Moore's Law and the Long Term