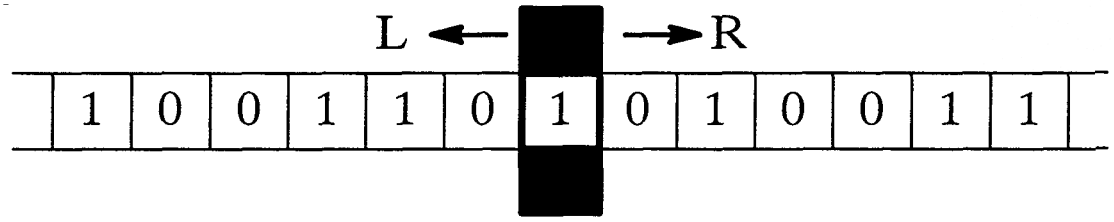
DEF: **Parallel Computer:** **A set of processors capable of working cooperatively to solve a computational problem.**

**BRIEF HISTORICAL OUTLINE:**

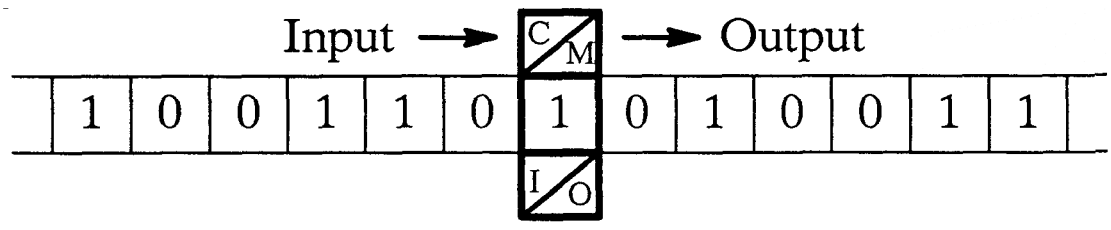
*Early contributors to the area of computing were the mathematicians:*

| | |
|---|---|
| *Late 1800's:* | *Frege creates the system of logic (truth tables, rules of inference, etc.)* |
| *Early 1900's:* | *Church and Turing independently ponder the issue of computability* |

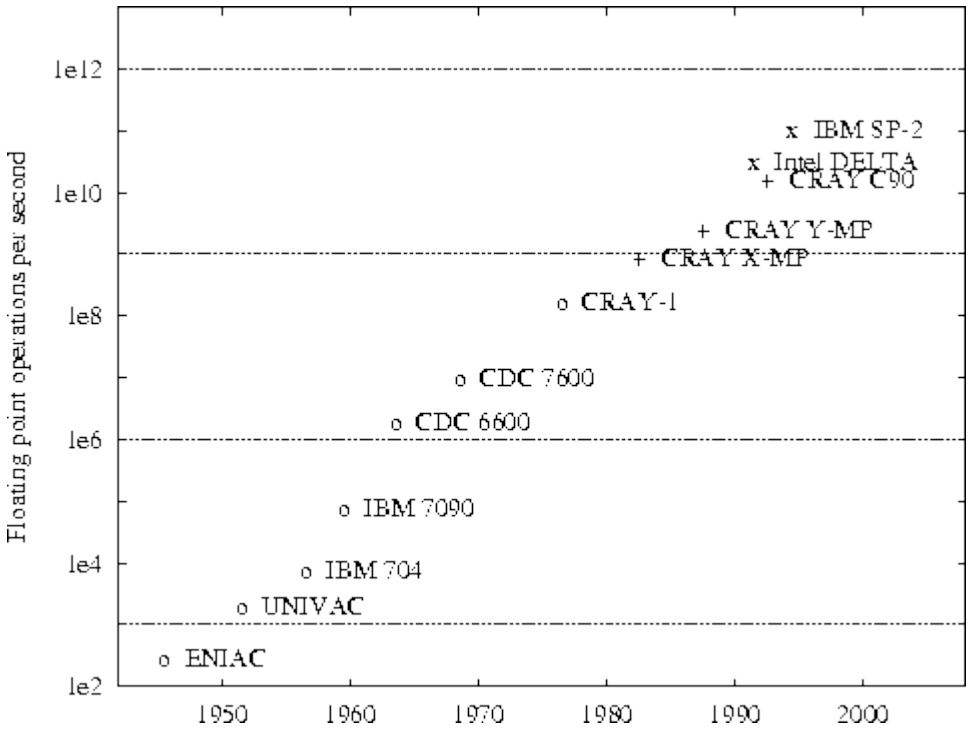
*The concept of the Turing machine:*




*Our present–day computers are due to von Neumann, and adhere pretty strictly to the "Turing architecture"*

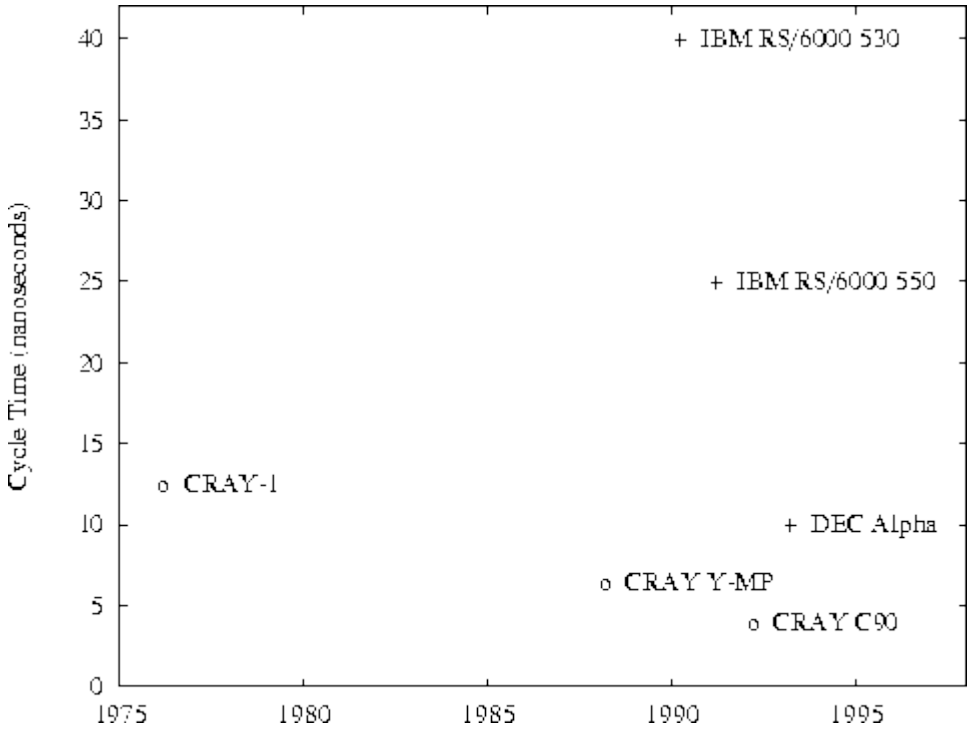


## TECHNOLOGICAL TRENDS (1)



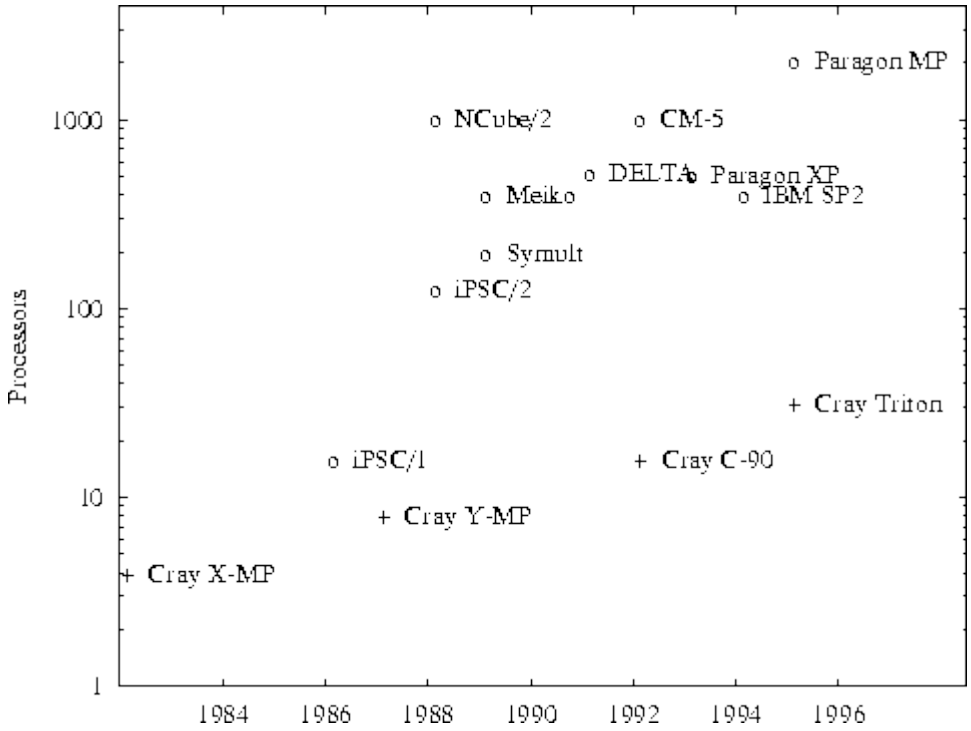

Peak performance of some of the fastest supercomputers, 1945--1995. The exponential growth flattened off somewhat in the 1980s but is accelerating again as massively parallel supercomputers become available. Here, ``o'' are uniprocessors, ``+'' denotes modestly parallel vector computers with 4--16 processors, and ``x'' denotes massively parallel computers with hundreds or thousands of processors. Typically, massively parallel computers achieve a lower proportion of their peak performance on realistic applications than do vector computers*. [source: Foster:3]*

## TECHNOLOGICAL TRENDS (2)




Trends in computer clock cycle times. Conventional vector supercomputer cycle times (denoted ``o'') have decreased only by a factor of 3 in sixteen years, from the CRAY-1 (12.5 nanoseconds) to the C90 (4.0). RISC microprocessors (denoted ``+'') are fast approaching the same performance. Both architectures appear to be approaching physical limits. *[source: Foster:3]*

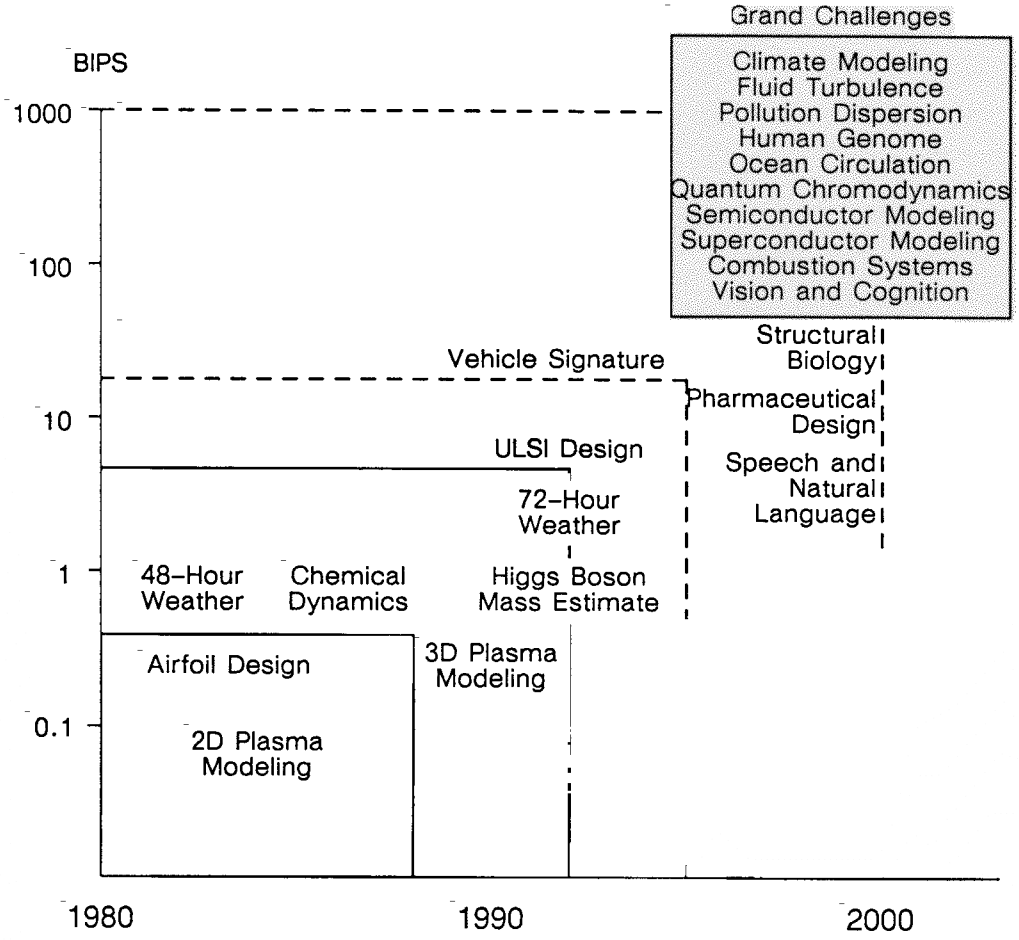
## TECHNOLOGICAL TRENDS (3)




Number of processors in massively parallel computers (``o'') and vector multiprocessors (``+''). In both cases, a steady increase in processor count is apparent. A similar trend is starting to occur in workstations, and personal computers can be expected to follow the same trend. *[source: Foster:3]*

## GRAND COMPUTATIONAL CHALLENGES

*Computational performance required to crack the "Grand Challenge" problems:*




*Source:* "Grand Challenges: High Performance Computing and Communications" *Report by the Committee on Physical, Mathematical and Engineering Sciences, Office of Science and Technology Policy, U.S. Government, 1991.*

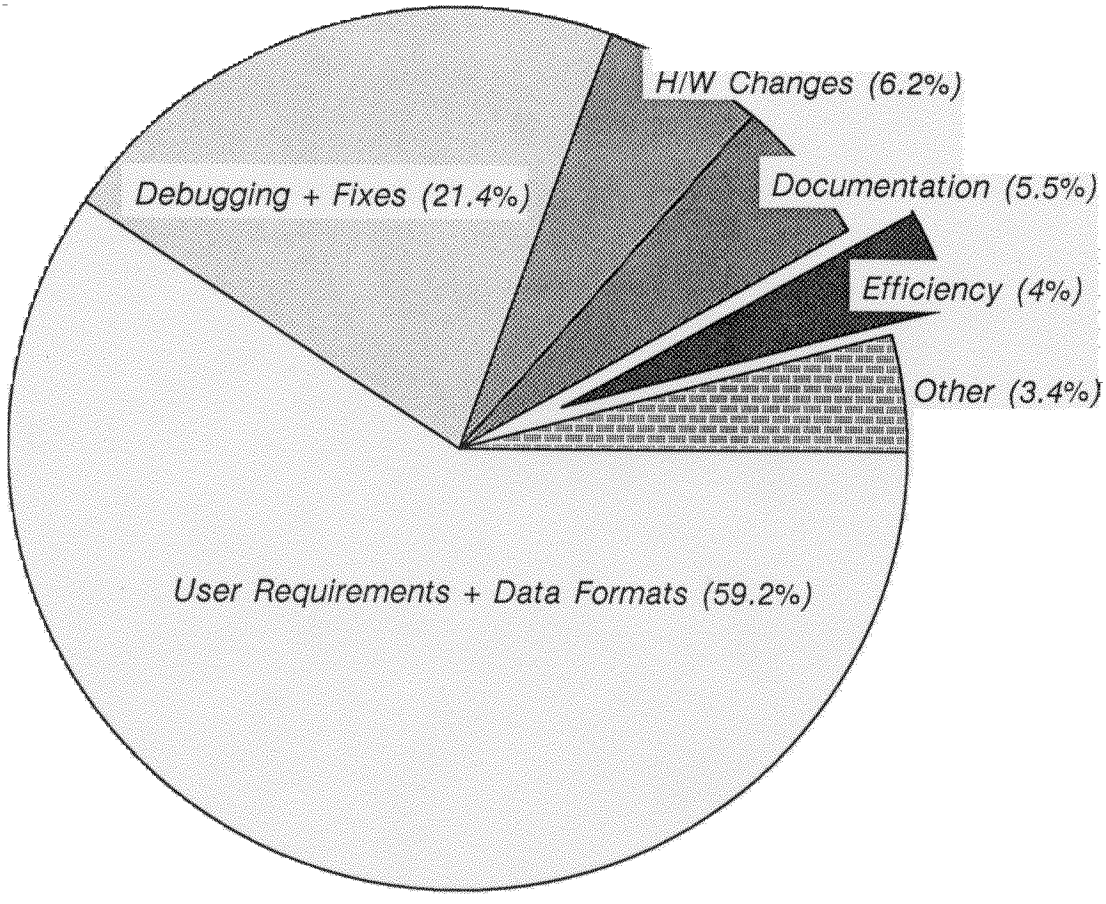
## COST OF CHALLENGES (1)

| Current State | Needed Capability | Cost |
|---|---|---|
| 100-km resolution | 10-km resolution | $10^2$–$10^3$ |
| Simple process representations | Improved process representations | 2–10 |
| Simple ocean | Fully coupled ocean | 2–5 |
| Simple atmospheric chemistry | Improved atmospheric chemistry | 2–5 |
| Limited biosphere | Comprehensive biosphere | about 2 |
| Tens of years | Hundreds of years | 10–$10^2$ |

Various refinements proposed to climate models,
and the increased computational requirements associated with these refinements.
Altogether, these refinements could increase computational requirements by
a factor of between $10^4$and $10^7$. *[source: Foster:3]*

## COST OF CHALLENGES (2)

### BREAKDOWN OF SOFTWARE MAINTENANCE COSTS




Source: After Lienz [1989]

## REALITY CHECK

*We have vastly improved the computer components:*

- *a new generation appears every 5 years*
  - *– chip density (i.e. memory capacity) * 4*
  - *– processing speed * 2*

*But we never dared to stray from the basic architecture, so:*
*the price of computing equipment stays at $200/lb,*
*we created languages for serial programming*
*thus painting ourselves into a corner (mental block)*

*Lately, we have got a feeling that all that is inadequate, because:*

- *we want to use computer as tools in the real world;*
- *in the real world things happen in parallel:*
  - *serializing parallel actions can be done easily*
  - *but you have to analyze in order to parallelize!*

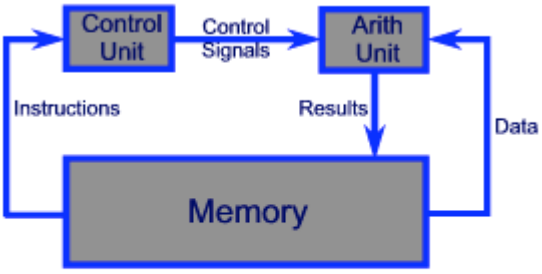
## CURRENT TECHNOLOGICAL LIMITATIONS:

*Partial list:*

1. *Most computers are of fixed architecture, forcing us to adapt problem solutions to the architectures available,*

2. *Most CASE tools preach software reusability, instead of allowing automatic software modification to fit the solution–driven computer architectures.*

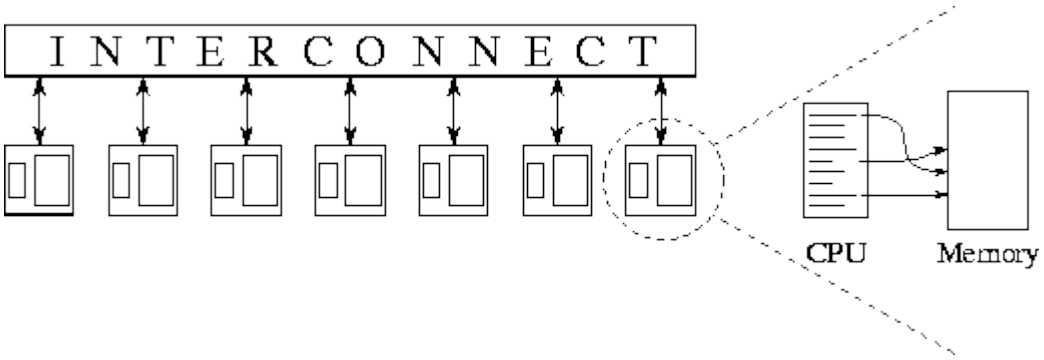
## MACHINE MODELS:
## The von Neumann Computer




**SISD (von Neumann) Architecture**

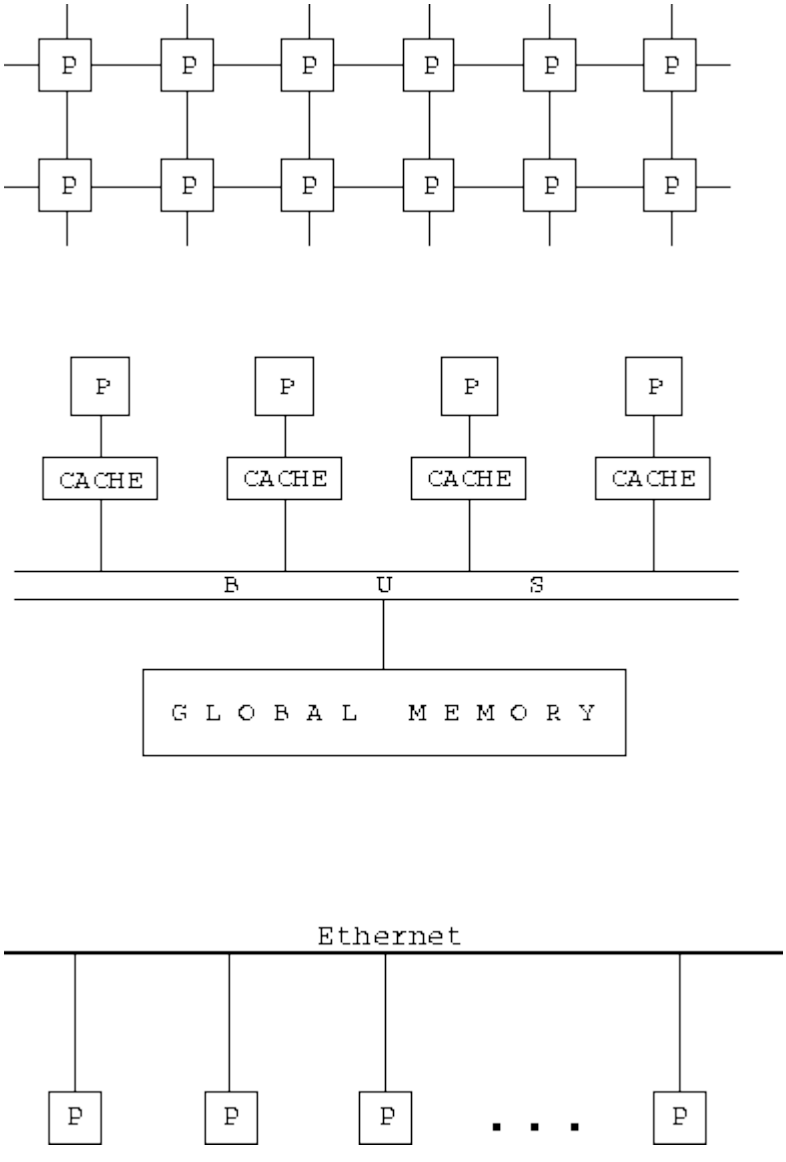
A central processing unit (CPU) executes a program that performs a sequence of read and write operations on an attached memory.

## MACHINE MODELS:
## The Multicomputer




An idealized parallel computer model. Each node consists of a von Neumann machine: a CPU and memory. A node can communicate with other nodes by sending and receiving messages over an interconnection network. *[source: Foster:3]*

## MACHINE MODELS:
## Sample Supercomputer Architectures




Classes of parallel computer architectures. From top to bottom: a distributed-memory MIMD computer with a mesh interconnect, a shared-memory multiprocessor, and a local area network (in this case, an Ethernet). In each case, P denotes an independent processor. *[source: Foster:3]*

## PARALLEL PROCESSING:

**Parallelism**: *Many operations performed concurrently.*

**Interaction**: *Cooperation between parallel streams of operations (processes).*

*This approach promises significant performance improvements, but first we must solve the problems of:*

- *process synchronization,*
- *interprocess communication*
- *deadlock,*
- *livelock*

## TYPES OF PARALLELISM

**Covert parallelism:**

*parallelism is NOT visible to the programmer,*
*compiler extracts parallelism from sequential,*
*algorithms and problem decompositions,*
*small speedups (typically *10 – *30).*

**Overt parallelism:**

*parallelism is visible to the programmer,*
*programmer has better understanding of*
*the problem at hand than any compiler,*
*usage of parallel algorithms is facilitated,*
*problem decompositions are facilitated,*
*large speedups (*1000 and greater) typical.*

## MEASURES OF PARALLELISM

*Speedup:*

$$Speedup = \frac{Speed\ of\ parallel\ algorithm}{Speed\ of\ BEST\ sequential\ algorithm}$$

*Node efficiency:*

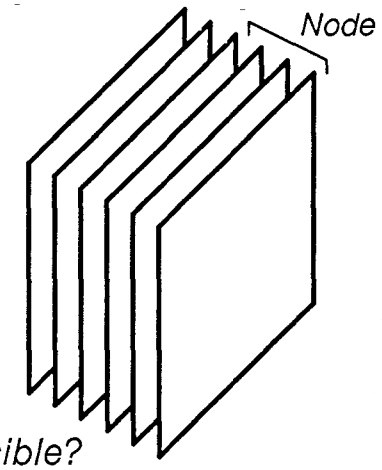
$$E_N = \frac{T_1}{NT_N}$$

*Area efficiency:*

$$E_A = \frac{T_1 A_1}{T_N A_N}$$

## NODE EFFICIENCY vs. AREA EFFICIENCY

*Node efficiency* $E_N = T_1/NT_N$ *leads to machines with a few large nodes:*




*NOTE: If S represents speedup, we have:*

$$E_N = \frac{S}{N}$$

*QUESTION: Is* $E_N > 1$ *possible?*

*Area efficiency* $E_A = T_1A_1/T_NA_N$ *leads to machines with many small nodes:*

*Most promising; allows:*

- *Further miniaturization*
- *Computing styles:*
  - *Eager*
  - *Just–in–time*