

# PERFORMANC EVALUATION

S.Lakshmivarahan

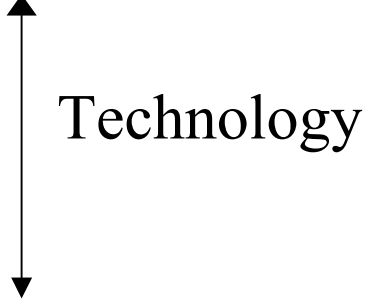
School of Computer Science

University of Oklahoma

# HYPE ABOUT SUPER/HYPER

- Super star/model
- Super market
- Super bowl
- Super sonic
- Super computers

# Every computer is a super computer of its time

- Drum to core memory
  - Vacuum tubes to integrated circuits
  - Pipelined/vector arithmetic unit
  - Multiple processors - Architecture
- 
- Technology

# Modern era began in mid 1970's

- CRAY-I – Los Alamos National Lab
- Intel Scientific computers in early 1980's
- NCUBE
- Alliant, Hitachi
- Denelcor
- Convex, SGI, IBM, Thinking Machine
- Kendall Square

# Today: super ~ parallel

- Lot of hype in the in 1980's
- Parallelism will be everywhere and without it you will be in back waters
- Analogy with helicopters

# This prediction did not materialize

- Silent revolution from behind
- Thanks to technology
- By mid 1990's workstations as powerful as CRAY-I was available for a fraction of the cost – from millions to a few ten thousands
- Desk top computing became a reality
- Many vendors went out of business
- When the dust settled access to very powerful (CRAY like) desk top became a model for computing

Envelop of problems needing large scale processors was pushed far beyond what was conceived in mid 1980's

- Lead to interesting debate in the 1990's
- **“Super computing ain't so super “– IEEE Computer 1994 by Ted Lewis**
- **“Parallel Computing:Glory and Collapse”- IEEE Computer, 1994 by Borko Furht**
- **“Parallel Computing is still not ready for mainstream” CACM, July 1997 by D. Talia**
- **“Parallel Goes Populist” Byte May 1997 by D. Pountain**

# Our view:

- Parallelism is here to stay, but not everyone needs it as was once thought.
- Computing will continue in mixed mode- serial and parallel will coexist and complement each other
- Used 128 processor Intel hypercube and Denelcor HEP-I at Los Alamos in 1984
- Alliant in 1986
- Cray J-90 and Hitachi in mid 1990's
- Several parallel clusters, the new class of machines



# A comparison:

- IEEE Computer Society president made a calculation that goes like this: If only automobile industry did what computer industry has done to computers we should be able to buy Mercedes Benz for a few hundred dollars

# Theme of this talk is Performance

- The question is: of what?
- Machine/Algorithm/Installation
- Raw power of the machine: Megaflop rating
- Multiprogramming was invented to increase machine/installation performance
- Analogy with dentist office, major airline hubs
- Parallel processing is more like open heart surgery or building a home

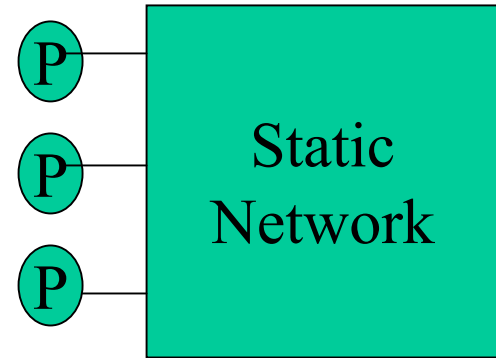
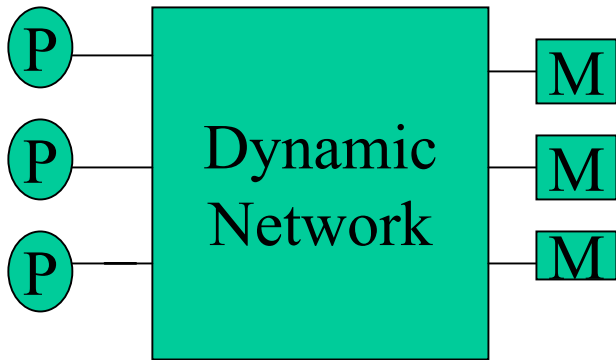
# Algorithm performance

- Total amount of work to be done measured in terms of the number of operations
- Interdependency among various parts of the algorithm
- Fraction of the work that is intrinsically serial
- This fraction is a determinant in deciding the overall reduction in parallel time

# Our Interest is solving problems

- Performance of machine – algorithm combination
- Every algorithm can be implemented serially but not all algorithms admits parallelism
- A best serial algorithm may be a bad parallel algorithm and a throw away serial algorithm may turn out to be a very good parallel algorithm

# A Classification of Parallel Architectures



- Shared Memory
- Indirect communication by read/write in shared memory
- Circuit switching- Telephone
- CARY-J90

- Distributed Memory
- Explicit Communication by send/receive
- Packet switching- Post-office
- Intel hypercube, Clusters
- Packet



An example of parallel performance analysis  
A simple Task done by one or two persons

Number of persons	Elapsed time	Total work done	Cost at @\$20/hr
1	10 hours	10	\$200
2	6 hours	12	\$240

- Speed up = Time by one person / Time by two persons  
=  $10/6 = 1.67 < 2$
- $1 < \text{speedup} < 2$
- Total work done is 12 man hours in parallel as opposed to 10 man hours serially
- Reduction in elapsed time at increased resources/cost

- Problem P of size N
- Parallel Processor with p processors
- Algorithm: a serial algorithm  $A_s$  and a parallel algorithm  $A_p$
- Let  $T_s(N)$  be the serial time and  $T_p(N)$  be the parallel time

$$\mathbf{Speedup} = S_p(N)$$

= Ratio of the elapsed time by the best known serial algorithm/ Time taken by the chosen parallel algorithm

$$= T_s(N)/T_p(N)$$

$1 \leq S_p(N) \leq p$ , the number of processors

In the above example speed is 1.67 and  $p=2$

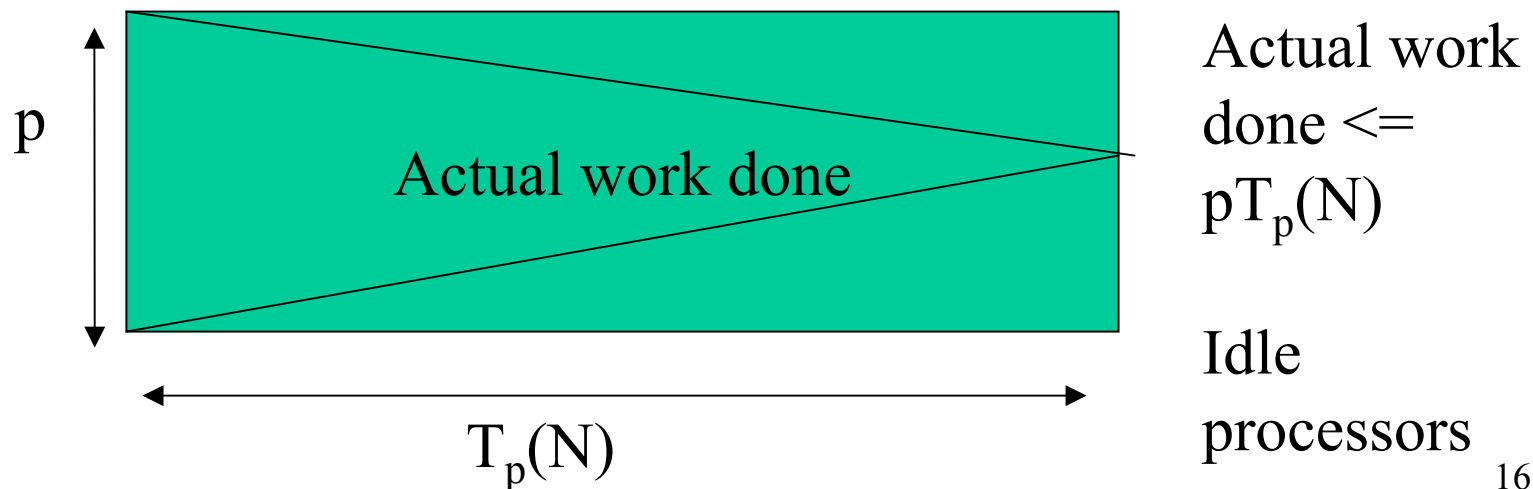
$$\text{Processor Efficiency} = E_p(N)$$

=Speedup per processor

$$= S_p(N)/p$$

$$0 < E_p(N) \leq 1$$

In the above example, efficiency =  $1.67/2 = 0.835$





$$\text{Redundancy} = R_p(N)$$

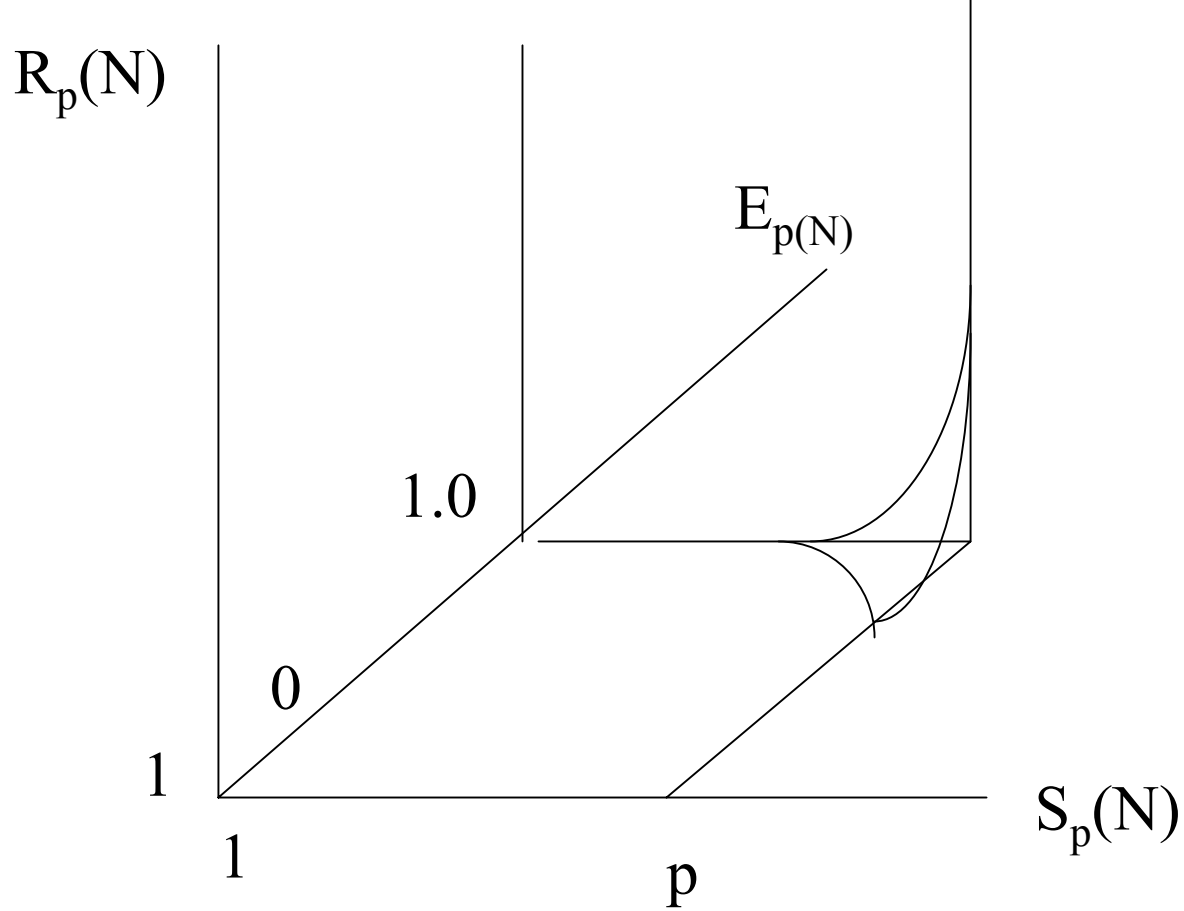
= Ratio of the work done by the parallel algorithm  
to the work done by the best serial algorithm

$$= \text{Work done in parallel} / T_s(N)$$

$$\geq 1$$

In the above example,  $R_p(N) = 12/10 = 1.2$

Desirable attributes: For a given  $p$ ,  $N$  must be large  
Speedup must be as close to  $p$   
Efficiency as close to 1  
Redundancy is close to 1.



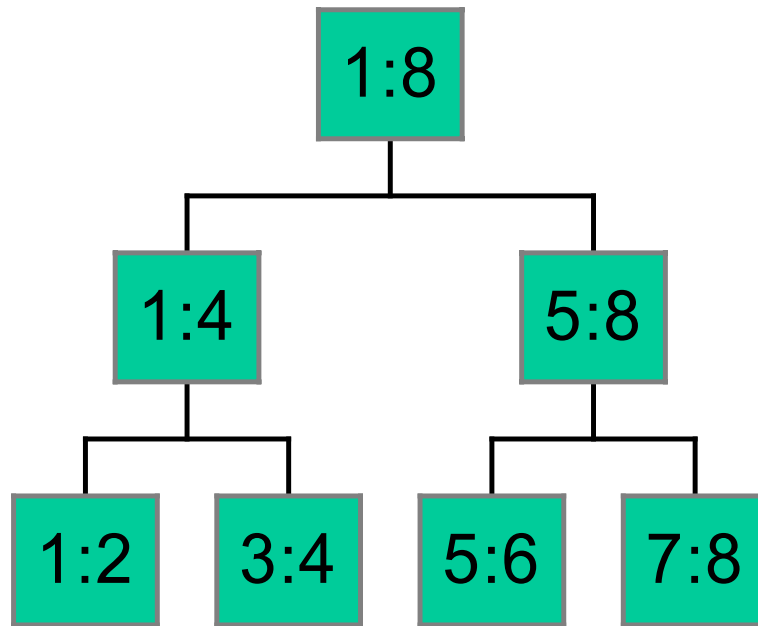
A View of the 3 Dimensional Performance Space

# CASE Study I

Find the sum of  $N$  numbers  $x_1, x_2, \dots, x_N$  using  $p$  processors

## Associative Fan-in Algorithm

$$1:4 = 1 + 2 + 3 + 4$$



$$N=8 \quad p=4$$

$$T_s(N) = 7$$

$$T_p(N) = 3$$

$$= \log 8$$

This is a complete binary tree with 3 levels

$$1:8 = 1:4 + 5:8 ; 1:4 = 1:2 + 3:4 ; 5:8 = 5:6 + 7:8$$

Number of operations performed decreases with time

Generalize: Problem size  $N$ , Processor size  $p = N/2$   
 $T_s(N) = N-1$ ,  $T_p(N) = \log N$

$S_p(N) = N-1/\log N \sim N/\log N$  - increases with  $N$ : **OK**

$E_p(N) = 2/\log N$  - decreases with with  $N$  : **Bad**

$R_p(N) = 1$  - **best value**

Why this? - **Too many processors & progressively idle**

Goal: Increase  $E_p(N)$  by decreasing  $p$

Strategy: **Keep  $p$  large but fixed and increase  $N$**

Fix the processor and scale the problem : **Think Big**

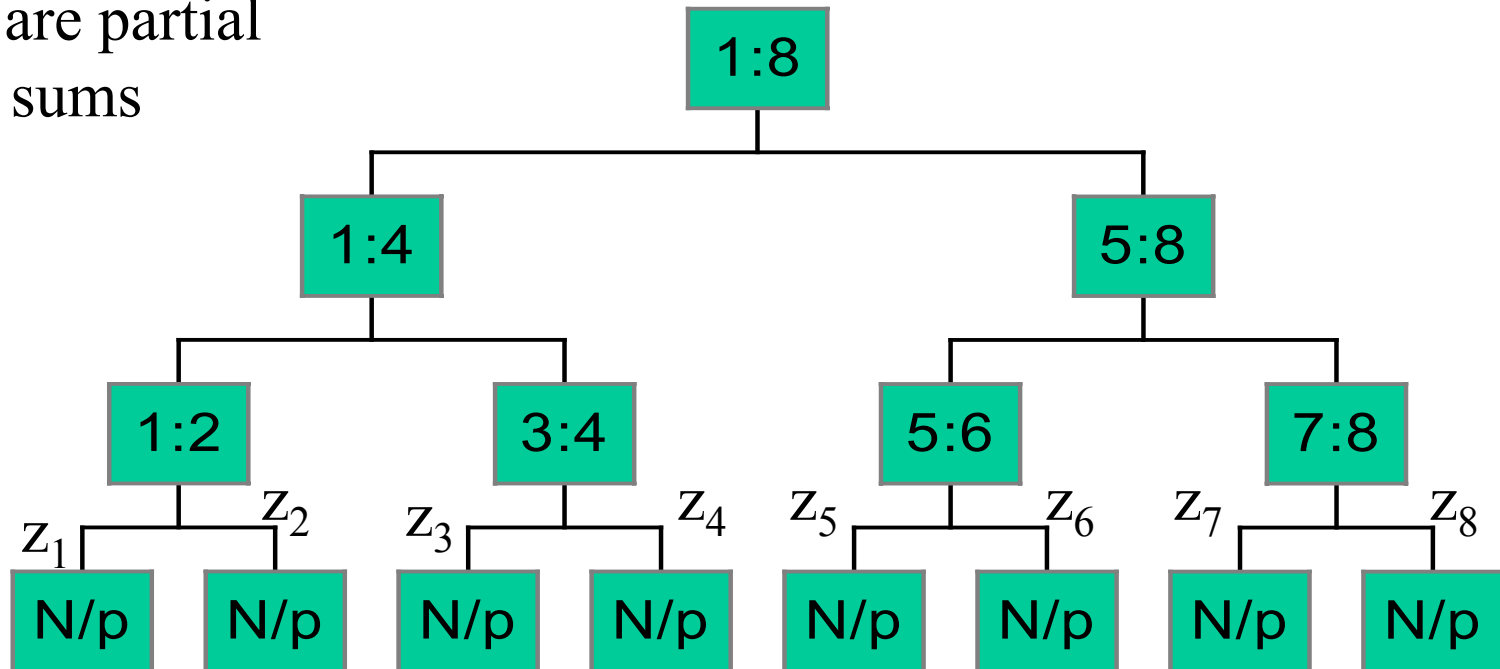
## Case Study II

$N=2^n$  and  $p=2^k$  with  $k < n$

Divide the problem into  $p$  parts of each of size  $N/p$

### Associative Fan-in Algorithm

$Z_i$ 's are partial sums



The first step takes  $N/p$  steps followed by  $\log p$  steps

Serial Time  $T_s(N) = N$

Parallel time  $T_p(N) = N/p + \log p$

Speed up =  $N / (N/p + \log p)$

Let  $N = Lp \log p$ . Since  $p$  is fixed, increase  $L$  to increase  $N$

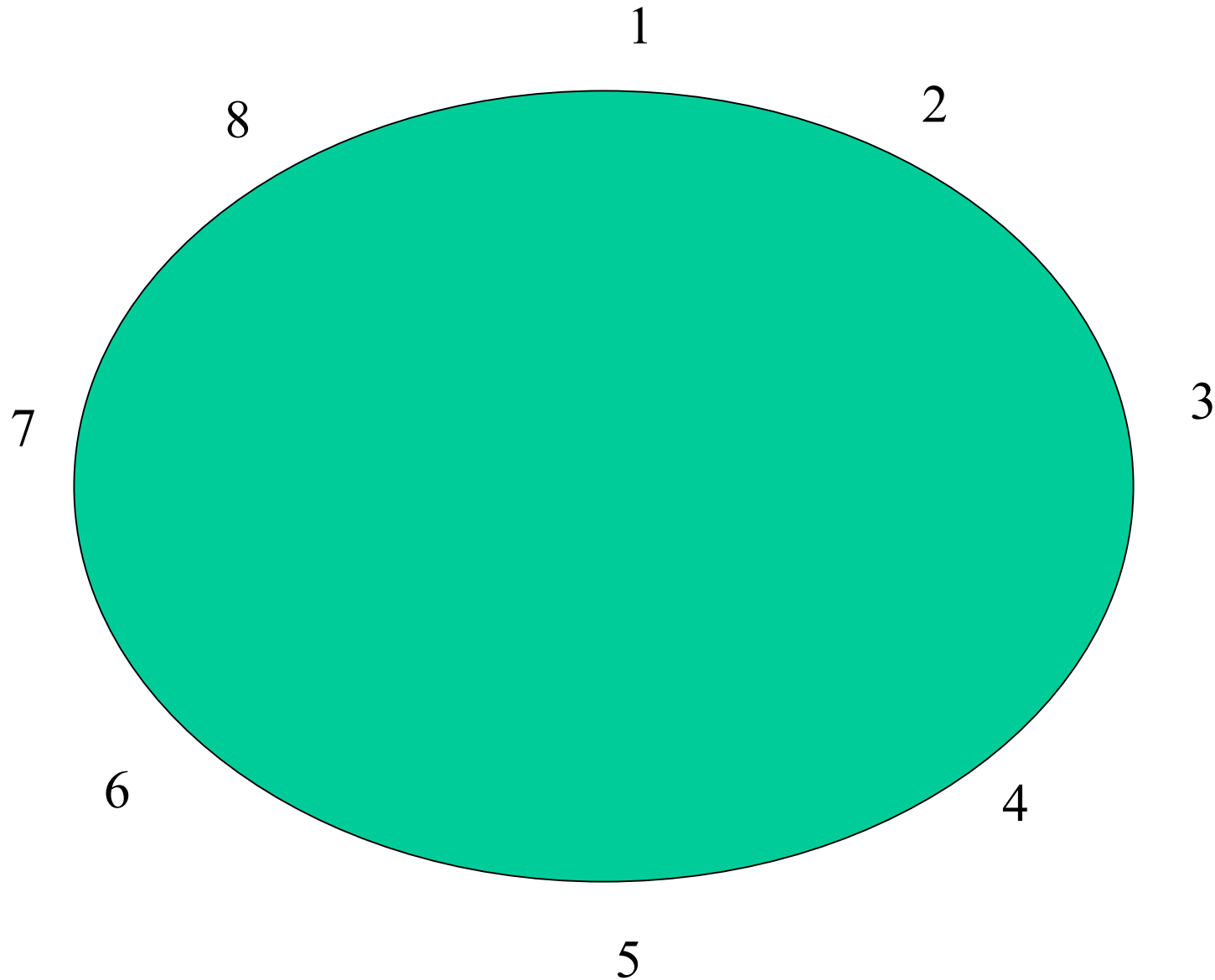
Speedup =  $Lp \log p / [(L+1) \log p]$   
=  $p [L / (L+1)]$   
=  $p$  the best possible when  $L$  is large

Efficiency =  $L / (L+1) \sim 1$  the best possible when  $L$  is large

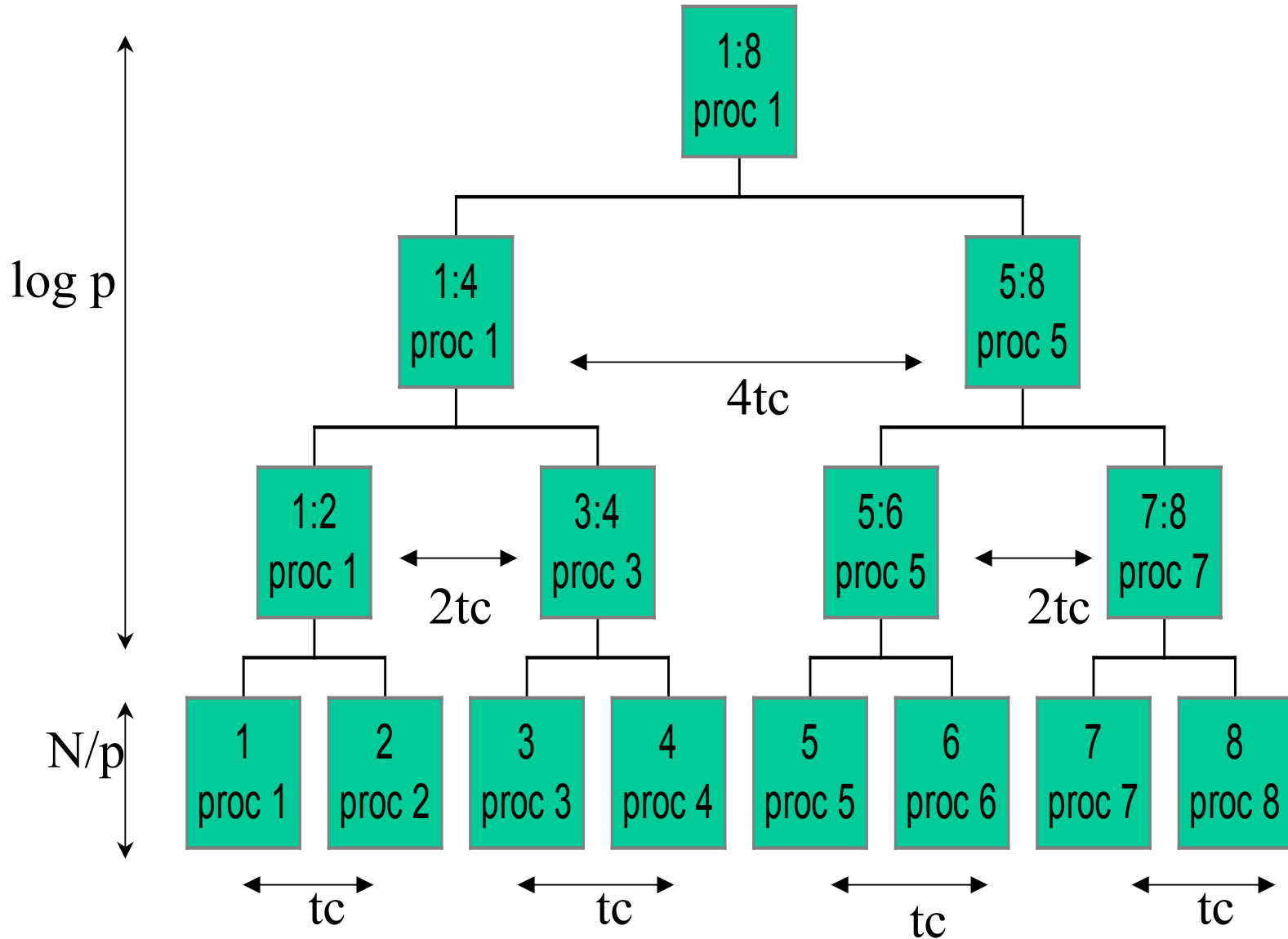
Redundancy = 1

The scaling does the trick – “Think big”

Case Study III: Impact of communication: N node ring  
Processor  $k$  has the number  $x_k$ . Find the sum



# A tree version of the implementation in a ring





$t_a$  - the computation time per operation (Unit cost model)

$t_c$  - the communication time between neighbors

$$T_s(8) = 7t_a \quad \text{and} \quad T_p(8) = 3t_a + (1+2+4)t_c \\ = 3t_a + 7t_c$$

$$\text{Speedup} = 7t_a / [3t_a + 7t_c] \\ = 7 / [3 + 7r] \quad \text{where } r = t_c / t_a$$

Ratio  $r$  depends on  $t_a$  - processor technology

$t_c$  - network technology

In the early days,  $r \sim 200-300$ . Check the value  $r$  first.

Writing a parallel program is like writing music for an orchestra

We now provide a summary of our findings:

# of procs =  $2^k = p < N = 2^n =$  problem size

Interconnection scheme: p node ring

$$\text{Speed up} = \frac{N t_a}{[N/p + \log p] t_a + (p-1) t_c}$$

$[N/p + \log p] t_a$  – depends on the parallel algorithm

$(p-1)t_c$  – depends on the match between the algorithm and the parallel architecture and on the technology of the implementation of the network.

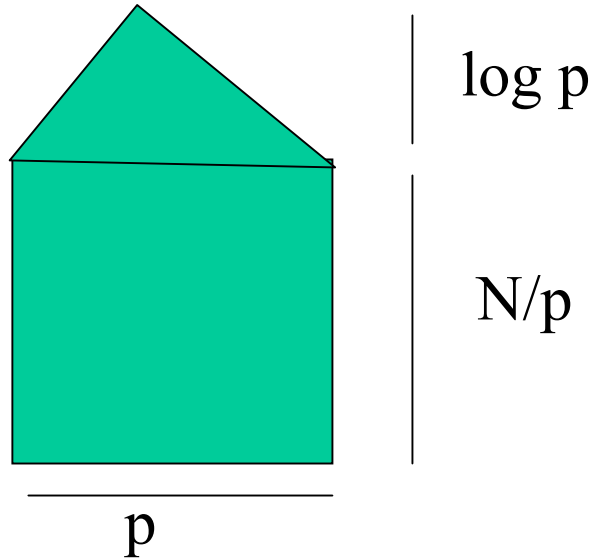
This expression for speedup will change if we change the following:

If we change the allocation of tasks to processors, it will change the communication pattern and hence the speedup

The network of interconnection is changed – hypercube, toroid  
Etc

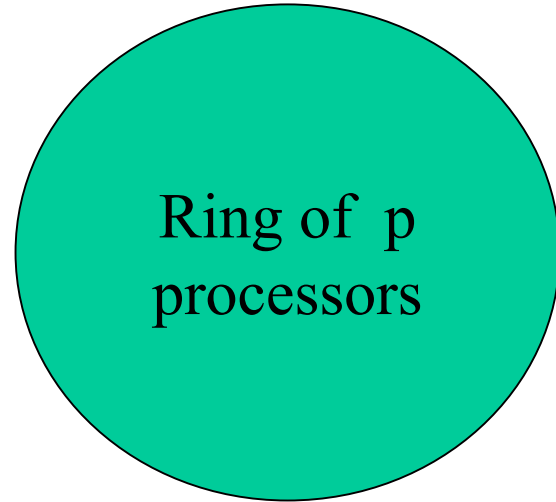
If we change the algorithm for the problem

# What is the import of these case studies?



Logical structure of the parallel algorithm

Host graph



Topology of the network of processors

Guest graph

Do these match?

The key question: Can we make every pair of processors who have to communicate be neighbors at all time?

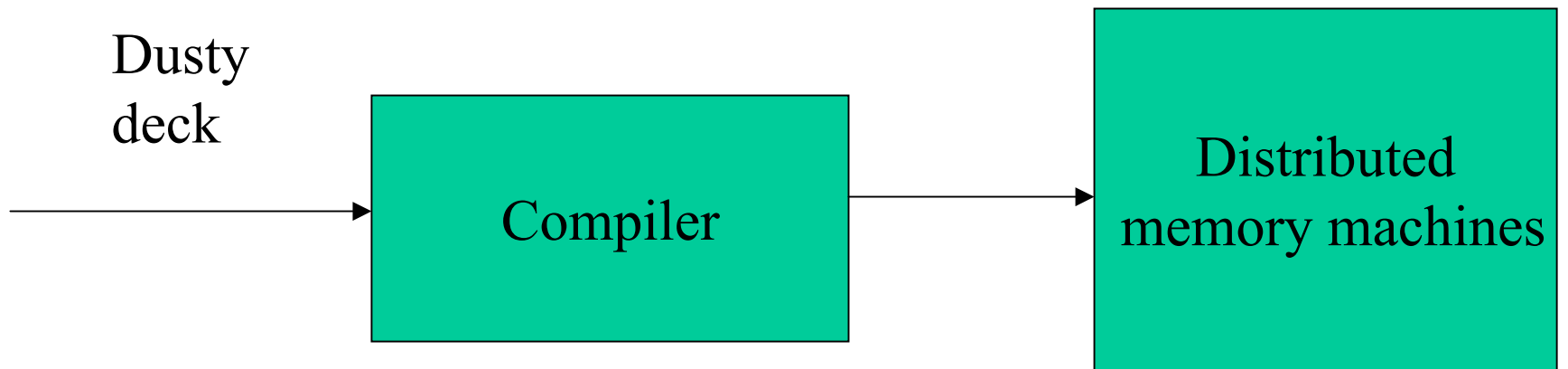
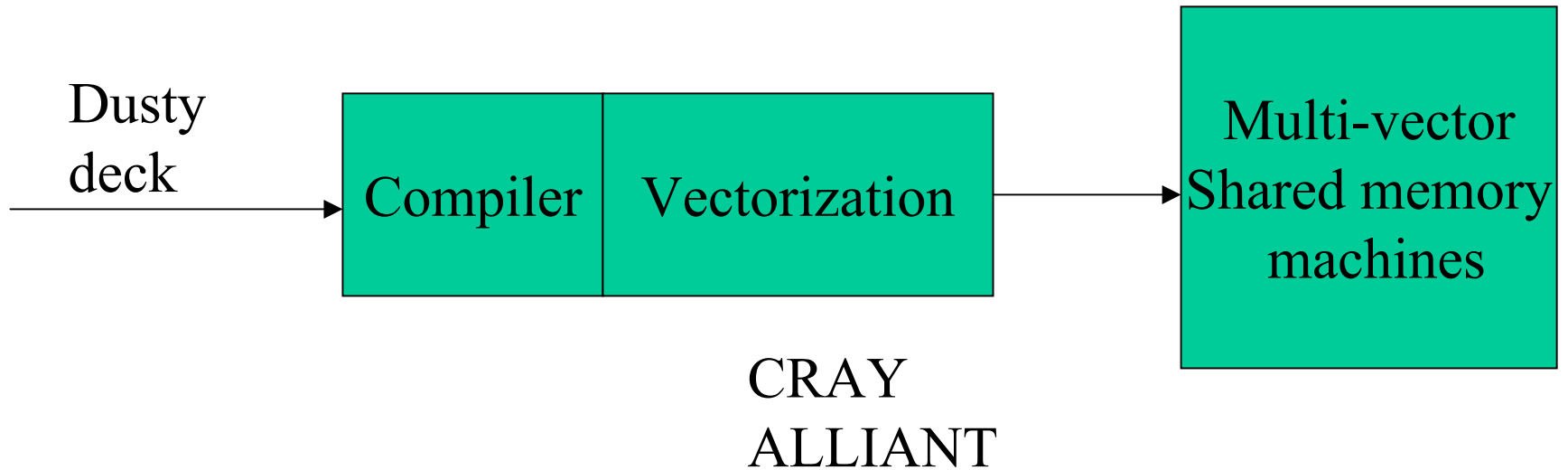
This is seldom possible. The new question is: can we minimize the communication demands of an algorithms?

The answer lies at the **match** between the chosen parallel algorithm and the topology of interconnection between processors of the available architecture.

The network of processors that is most suitable for your problem may not be available to you. This miss-match translates into more communication overhead

Your parallel program may give correct results but it may still a lot more time compared to when there is good match

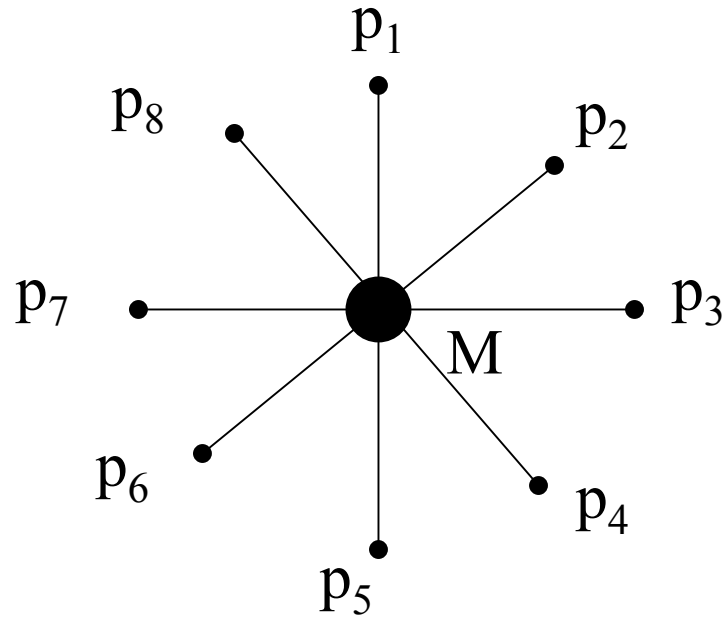
## Two modes of programming:



- To date there is no general purpose parallelizing compiler for distributed memory machines
- HPF provides a partial solution but not very efficient yet<sup>30</sup>

- The import is: **we have to do it all from ground up**
- PVM / MPI just provide the communication libraries  
-- FORTRAN and C versions
- Send, Receive, Broadcast, scatter, Gather etc.
- These are to be embedded in the program at the right places
- Thus, the program has to specify the following:
  - Who gets what part of the data set?
  - What type of computations to be done with this data?
  - who talks to whom at what time ?
  - How much they communicate?
  - Who has the final results?
- Cover rudiments of MPI in the next seminar

# The CS cluster – (16 +1) node Star interconnection



Master node is at the center  
Master handles the I/O



## References

S. Lakshmiarahan and S. K. Dhall [1990] **Analysis and Design of Parallel Algorithms**, McGraw Hill.

S.Lakshmiarahan and S. K. Dhall [1994] **Parallel Processing Using the Prefix Problem**, Oxford University Press.

Course Announcement: Spring 2003

**CS 5613 Computer Networks and Distributed Processing**

T-Th 5.00-615pm