INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

H4.SMR/854-12

College on Computational Physics

15 May - 9 June 1995

Performances of Parallel Applications

F. Massaioli

Università "la Sapienza"
Rome, Italy

CASPUR

FEDERICO MASSAIOLI

PERFORMANCES of PARALLEL APPLICATIONS

Parallel Computing Targets

- More speed
- Bigger problems
- Reduced costs

Parallel Computing Problems

- Efficient resources exploitation
- Greater program complexity
- New algorithms

Performance Analysis as a Development Tool

Preliminary Study

- Model of performances of the planned application
- Needed resources estimate
- Algorithms evaluation
- Is parallelization profitable?

Implementation

Test

- Measure of the resulting performances
- Comparison to the model
- Inefficiencies identification and resolution

Workstation Clusters

- Distributed Memory, MIMD
- n *multitasking* and *multiuser* processors
- Each one works on a fraction of the problem
- Inter-processors communications allow for the exchange of needed informations

Computing Time

- CPU Time T_{CPU} depends on the problem complexity, not on n
- Real Time T depends on n and enables speed measures

Speedup

$T(n)$ real time spent by n processors

T_s real time spent by the serial code on a single workstation

Speedup $S(n) = T_s \, / \, T(n)$

When T_s is not available, try the approximation: $S(n) \cong T(1) \, / \, T(n)$

Pay attention: $T(1) > T_s$!!!

Efficiency

Measure of the effectiveness of the use of n workstations in a single application

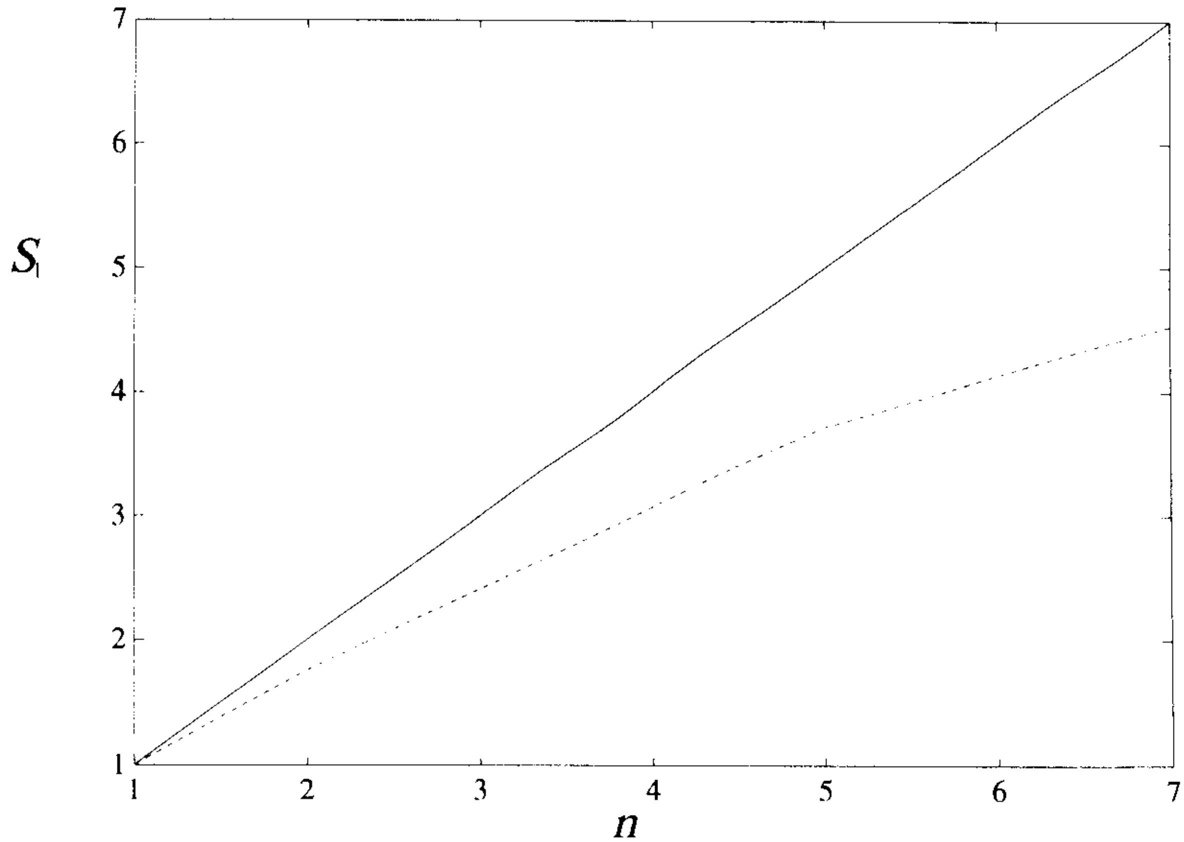
$$E(n) = S(n) \, / \, n$$

The Ideal Model

$$S_I(n) = n \qquad E_I(n) = 1$$

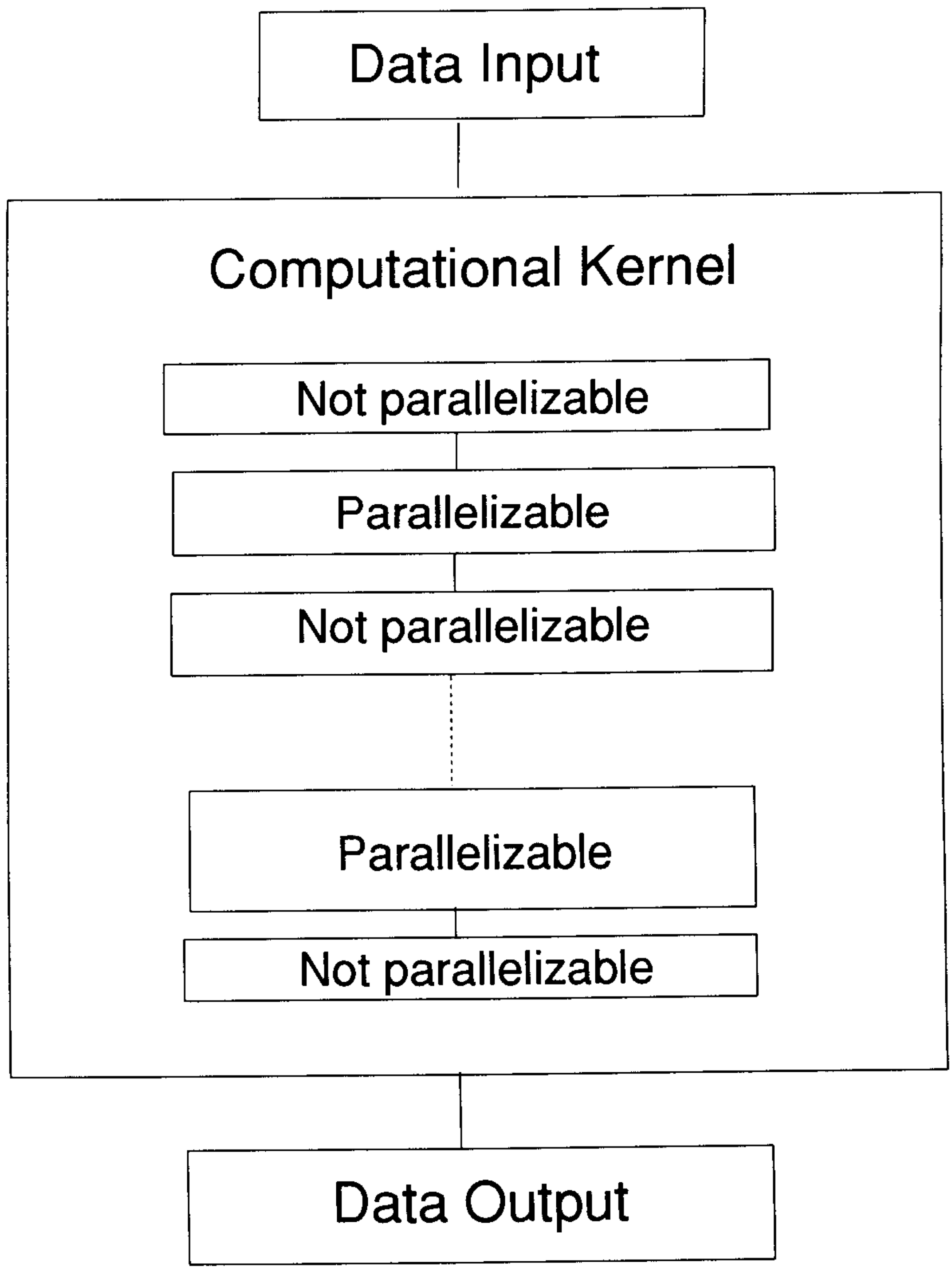
The Sad Truth

Baraglia, Laforenza, Perego, AICA '93
Sun Sparc2 Cluster, Ethernet, PVM 2.4



The ideal model isnaively *optimistic*

Application Structure



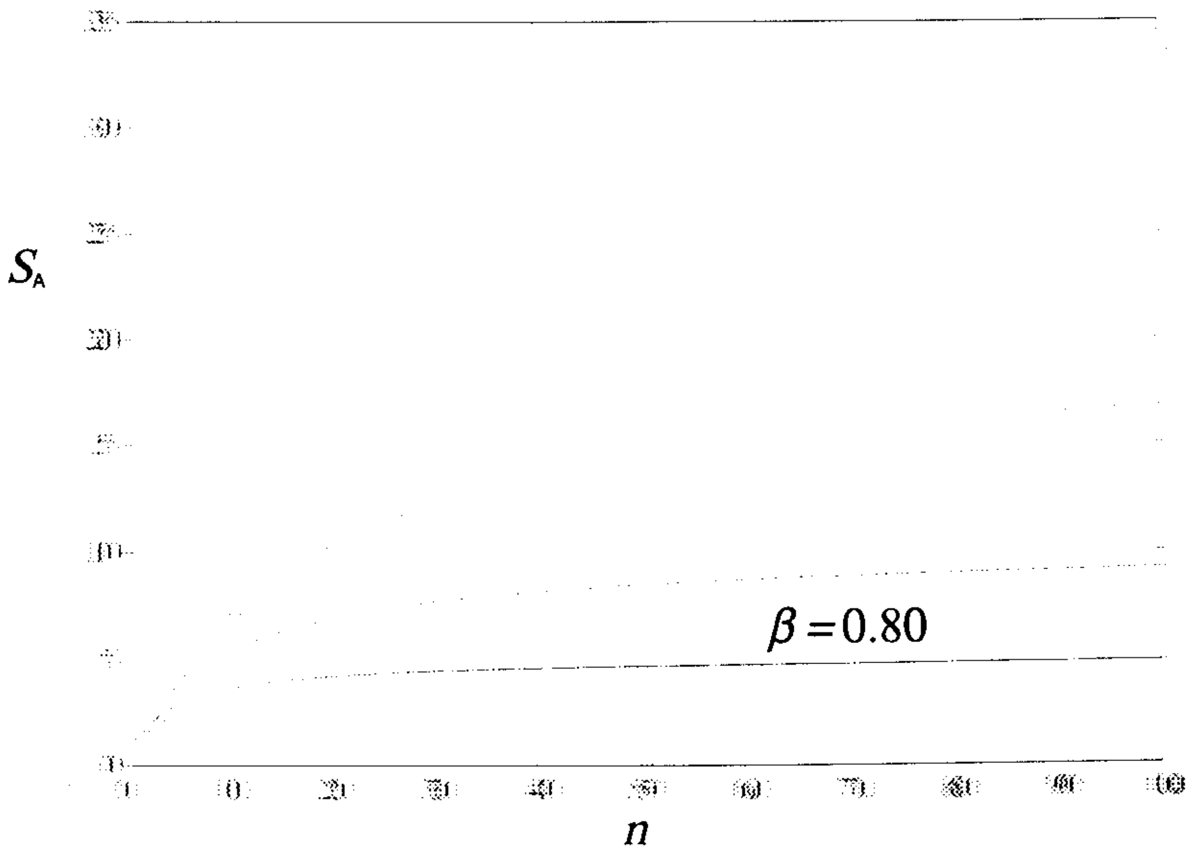
Factors affecting performances

- *Overhead* to launch n processes (negligibleble if every process is computation-intensive)
- Work load unbalance between processes (avoided with a careful implementation)
- Work load unbalance between processors (Parallel-minded queueing systems...)
- Communication dependent slow-downs (difficult to model)
- Computational Kernel sections not amenable to be parallelized

Amdahl's Law

Speedup $S_A(n) = 1 / (1 - \beta + \beta / n)$

where β is the fraction spent in parallelizable code of the total T_{CPU} needed by the whole computational kernel



For large n, $S_A \Rightarrow 1/(1-\beta)$, $E_A \Rightarrow 0$

Amdahl's Law is an upper limit

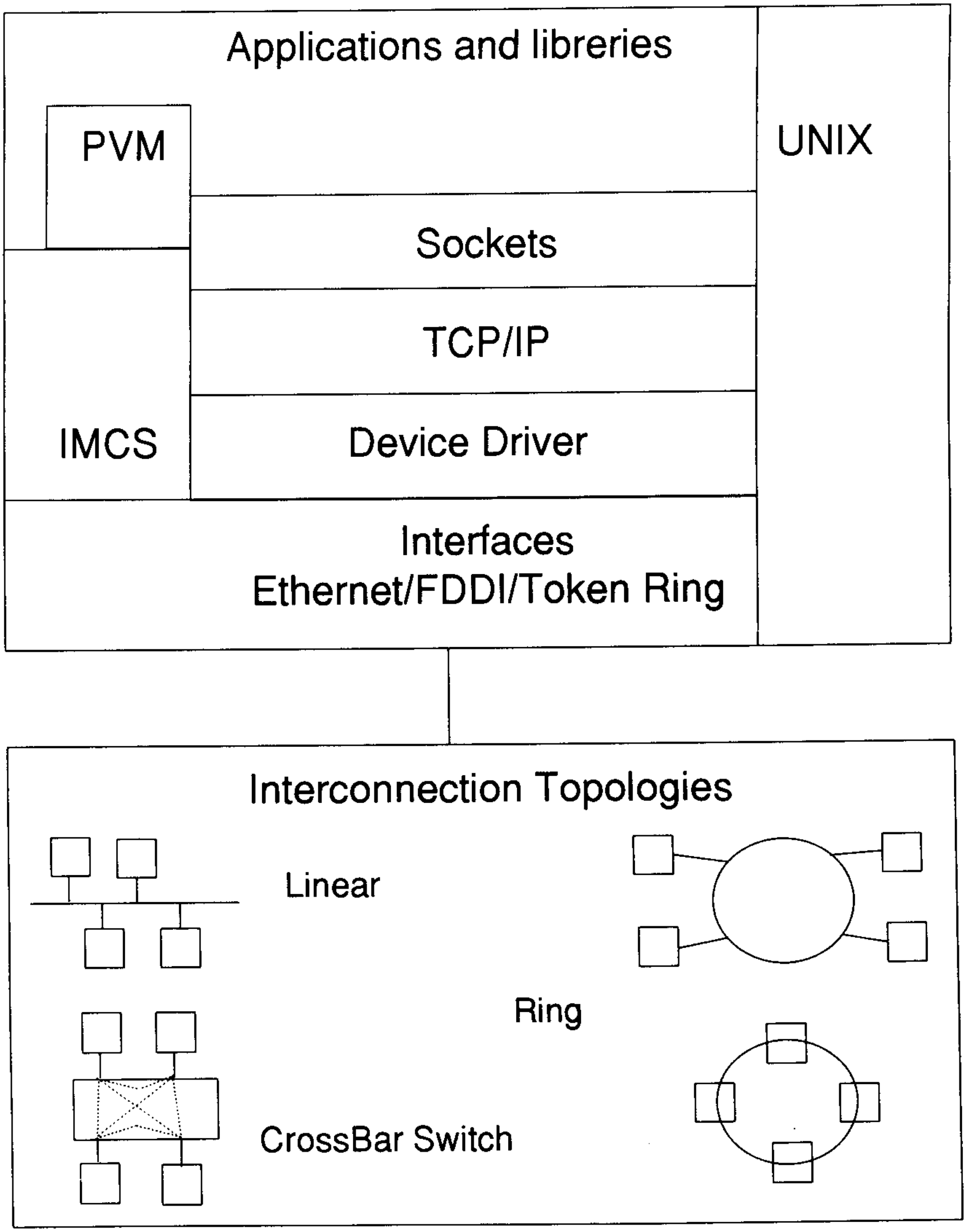
- Effects due to communications are overlooked

- Sections of codes not suitable for parallelism can imply heavy data communications, thus assuming a relevance greater than $1-\beta$

- β depends on the algorithm: a change of algorithm can yield far greater performances

- Careless implementations can always empair performances!!!

Communication model



Communication Throughput

- Depends on the *bandwidth* of the communications channel and on the ability of the processing hardware and operating system to sustain the I/O

- Typically around some megabits or megabytes per second

- Its relevance grows with the length of the messages

Communication latency

- Time needed to transmit a zero-length message
- Depend on both *hardware* and *software,*
- Typically around some hundred or thousand of microseconds, independent of the message length
- Has no relevance for long messages

Communication Time

- Ideally, if B is the bandwidth and L the latency, the transmission time of a message of length M is: $L + M / B$

Other Factors

- Time needed to pack and 'stamp' the message and to unpack it on arrival
- Communication paths can be reserved or shared with other applications
- The communication protocol can limit bandwith, latency and message lengths
- Interconnection topology

Connection Examples

Measures from:
IBM Technical Computing Solutions
Dallas

- Ethernet
  - *Bandwidth* : 1.25 MB/s
  - Measured rates: 0.38 - 1.00 MB/s
  - High latency

- Token ring
  - *Bandwidth*: 2.00 MB/s
  - Measured rates: 1.25 -1.50 MB/s
  - High latency

- FDDI (*Fiber Distributed Data Interface*)
  - *Bandwidth*: 12.5 MB/s
  - Measured rates: 3.00 - 6.00 MB/s
  - Very low latency

Messages in the application

- Process A send a message to process B

- Blocking (or Synchronous) Receive:
  B execution is suspended until message arrival

- Non-Blocking (or Asynchronous) Receive:
  B execution proceeds, but B must repeatedly check for the completion of the transmission

- Asynchronous receive is able to hide communications delays behind computations

Performances Measurements

- T_s (or $T(1)$) and $T(n)$ can be measured with `time`
- Measures must be taken under carefully controlled conditions
- Measures must be taken for different values of n
- To better identify problems, measures must be checked against the theoretical model of the application performances

Measurement Problems

- Problem size must be realistic!!!

  - A too small problem leads to overestimates of the role of communication

  - A too big problem leads to underestimates of the role of communications

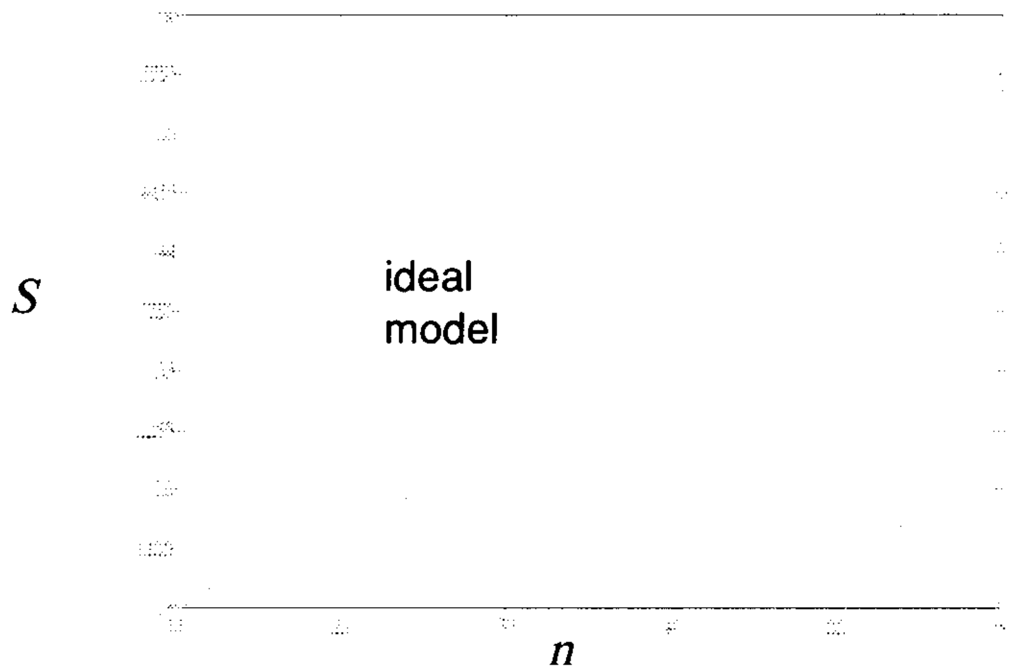
  - The test problem must be right-sized for the number of processors in use

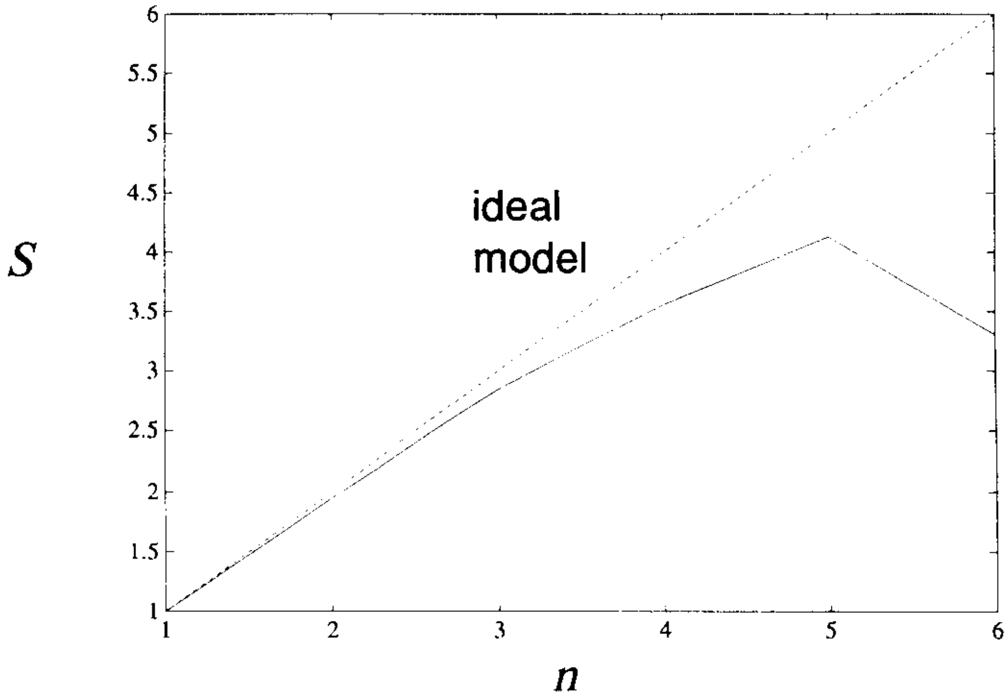
- Measures taken on dedicated processors are prone to hide a dependency of the application from strictly synvhronous interprocess communications

Dependence on the problem

Baraglia, Laforenza, Perego (Cnuce)

Sun Sparc2 Cluster, Ethernet, PVM 2.4

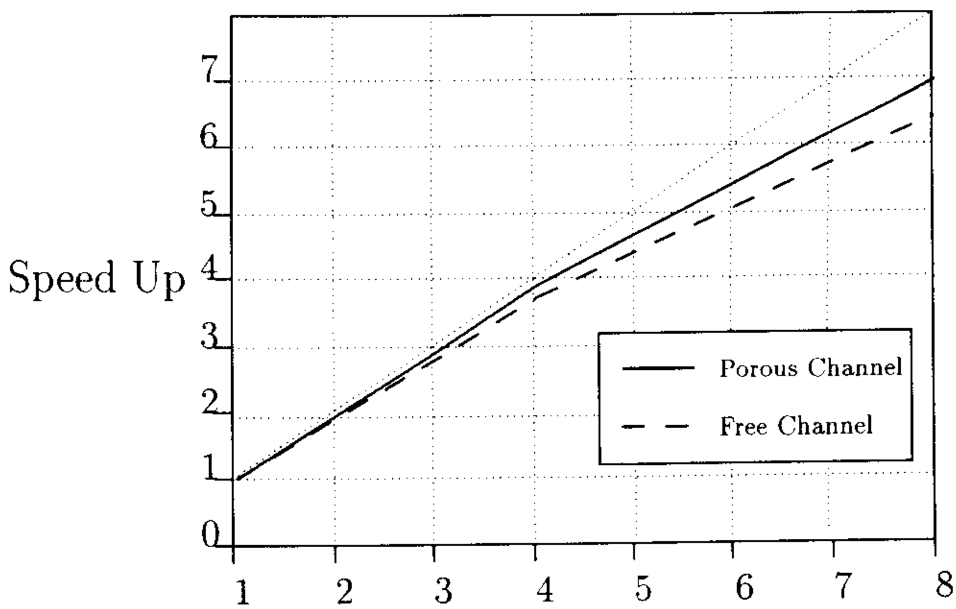




An intrinsically parallel algorithm

Succi, Betello, Massaioli, Righelli, Ruello
IBM ECSEC

Navier-Stokes Equations in 2D
Lattice Boltzmann Equation Method
IBM RS/6000-560 Cluster
PVMe su Token ring



In the Amdahl's model means
$\beta = 0.98$

Parallelism: a recipe...

Preliminary Phase

- Estimate β for the application to be parallelized and the upper speed-up limit from the Amdahl's law

- Estimate the relevance and impact of communications

- Estimate necessary resources (processors and interconnection system)

- Unsatisfied? Try a different algorithm for your app (maybe, **ask an expert**) and repeat the above steps

- No way to reach adequate performances? Give up, it happens.
  Otherwise...

...A recipe

Implementation

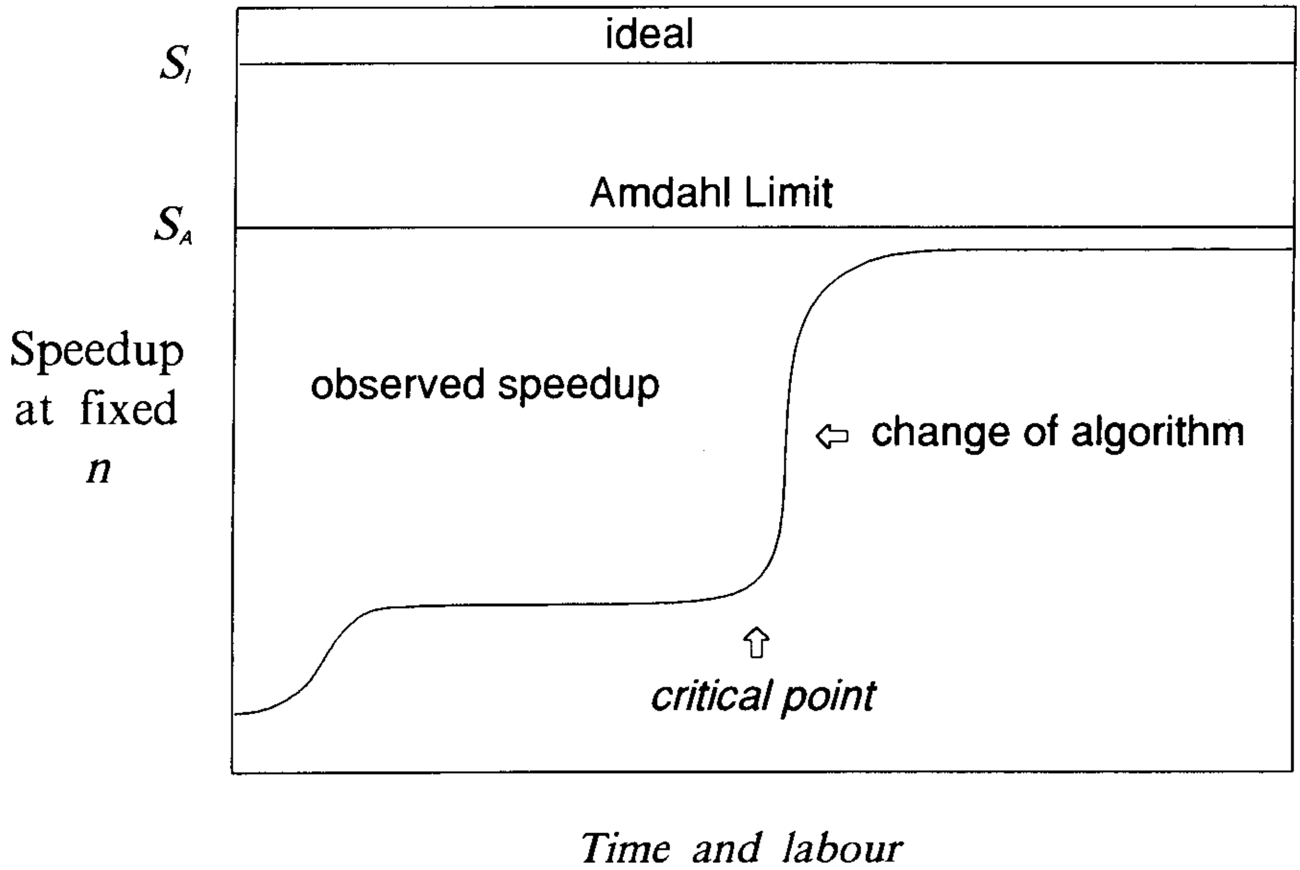
- Write the code, minimize communications effects (avoid critical sinchronizations!) and load unbalances

Test

- Get rid of errors in the code
- Measure obtained performances
- Identify problems causes and correct the implementation

If what you have done is interesting, share and publish it

The experience lesson



...an experienced people can help to rapidly reach and climb the critical point