ECE 669

Parallel Computer Architecture

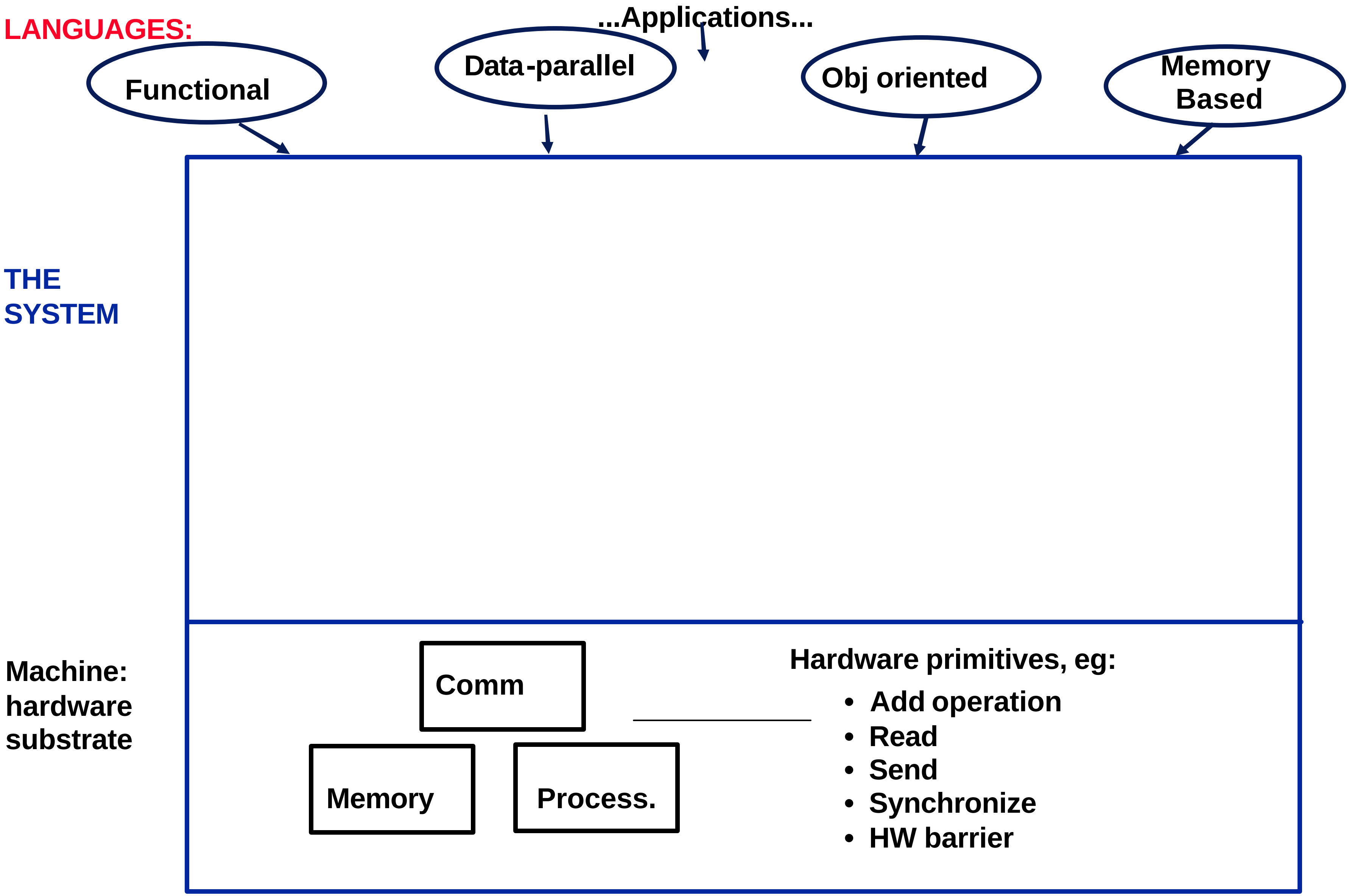
Lecture 7

Resource Balancing

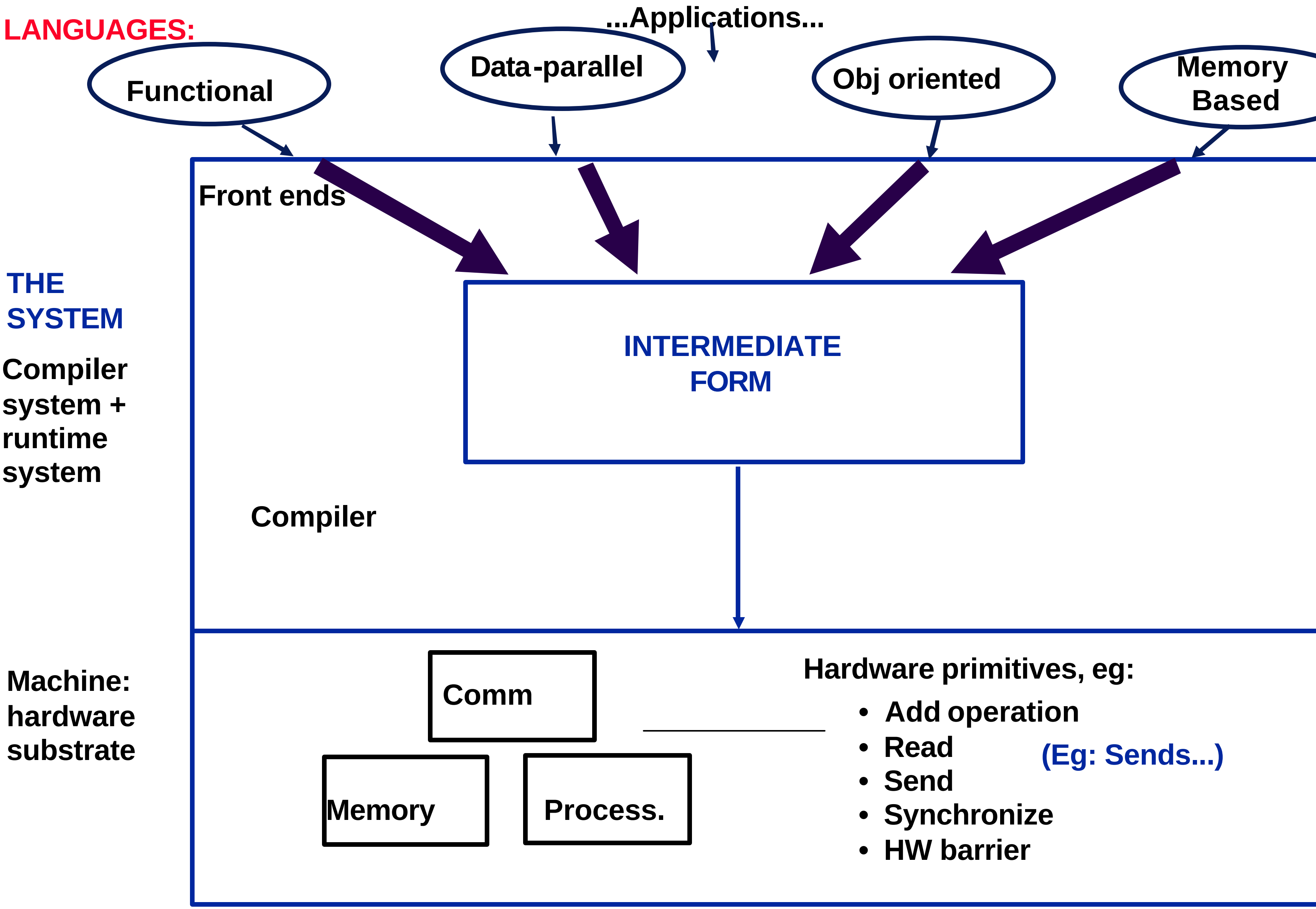
Outline

- Last time: qualitative discussion of balance
- Need for analytic approach
- Tradeoffs between computation, communication and memory
- Generalize from programming model for now
- Evaluate for grid computations
  - Jacobi
  - Ocean

Designing Parallel Computers: An Integrated Approach



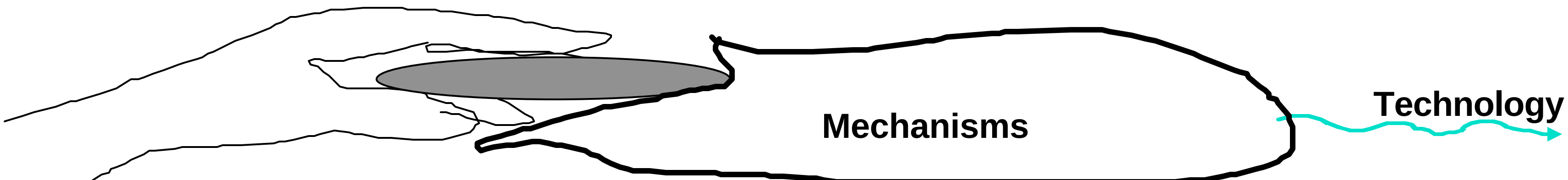
Designing Parallel Computers: An Integrated Approach



Hardware/Software Interaction

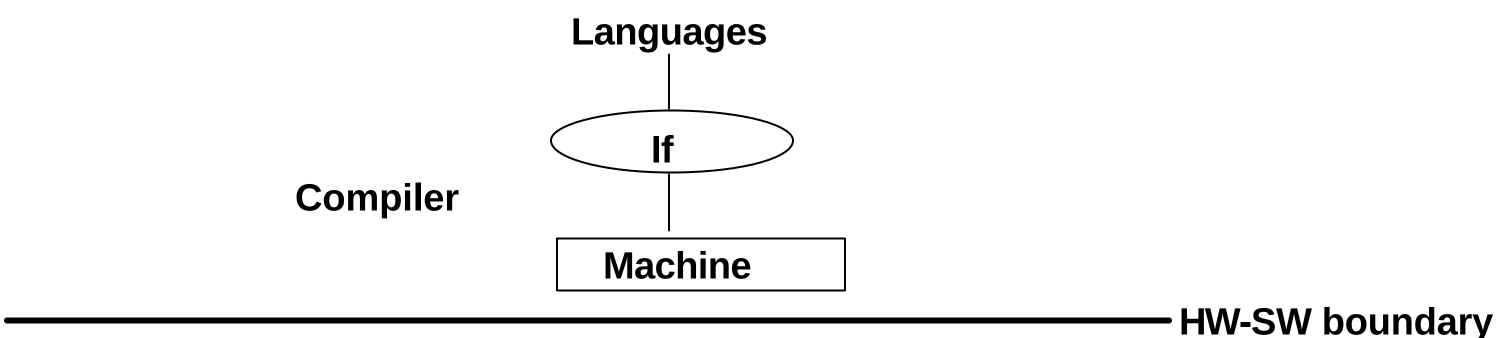
° Hardware architecture

- Principles in the design of processors, memory, communication
- Choosing primitive operations to be supported in HW.
  - Function of available technology.



° Computer + Runtime technology

- Hardware-software tradeoffs -- where to draw the line
- Compiler-runtime optimizations for parallel processing



Lesson from RISCs

High-level abstraction

IF

The ubiquitous POLYF

High-level Op

Simple hardware supported mechanisms

- **Add-Reg-Reg**
- **Reg-Reg Move**
- **Load-store**
- **Branch**
- **Jump**
- **Jump & link**
- **Compare branch**
- **Unaligned loads**
- **Reg-Reg shifts**

Binary Compatibility

- HW can change -- but must run existing binaries
- Mechanisms are important

vs.
Language-level compatibility
vs.
IF compatibility

- Pick best mechanisms given current technology -- fast!
- Have compiler backend tailored to current implementation
- Mechansims are not inviolate!

Key: “Compile & Run”

- Compute time must be small -- so have some IF

Choosing hardware mechanisms

° To some extent ... but it is becoming more scientific

- **Choosing mechanisms: Metrics**
  - **Performance**
  - **Simplicity**
  - **Scalability - match physical constraints**
  - **Universality**
  - **Cost-effectiveness (balance)**
  - **Disciplined use of mechanism**
  - **Because the Dept. of Defense said so!**
- **For inclusion, a mechanism must be:**
  - **Justified by frequency of use**
  - **Easy to implement**
  - **Implementable using off-the-shelf parts**

Hardware Design Issues

- Storage for state information
- Operations on the state
- Communication of information

- **Questions:**
  - How much storage --- comm bandwidth --- processing
  - What operations to support (on the state)
  - What mechanisms for communication
  - What mechanisms for synchronization
- **Let's look at:**
  - Apply metrics to make major decisions -- eg. memory --- comm BW --- processing
  - Quick look at previous designs

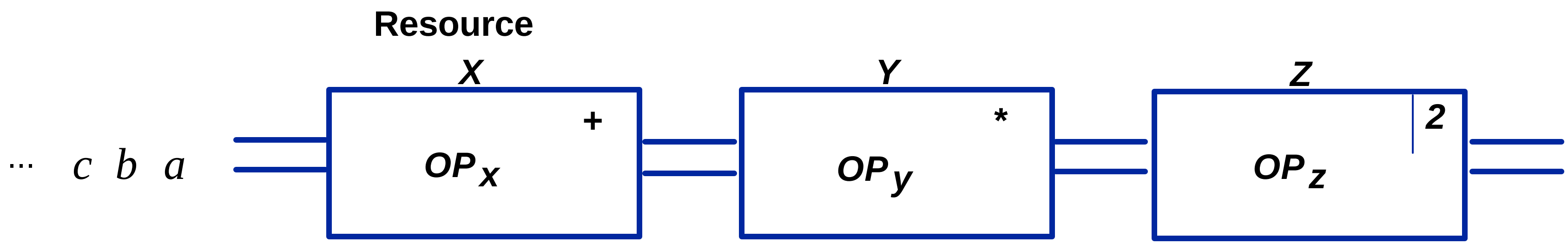
Example use of metrics

- **Performance**
  - **Support for floating point**
    - Frequency of use
  - **Caches speed mem ops.**
- **Simplicity**
  - **Multiple simple mechanisms -- SW synthesizes complex forms,**
  - **Eg. barrier from primitive F/E bits and send ops**
- **Universality**
  - **Must not preclude certain Ops. (preclude -- same as -- heavy performance hit) Eg. without past msg send, remote process invocation is very expensive**
- **Discipline**
  - **Avoid proliferation of mechanism. When multiple options are available, try to stick to one, Eg. software prefetch, write overlapping through weak ordering, rapid context switching, all allow latency tolerance.**
- **Cost effectiveness, scalability...**

Cost effectiveness and the notion of balance

- Balanced design -> every machine resource is utilized to fullest during computation
- Otherwise -> apportion $'s on underutilized resource to more heavily used resource
- Two fundamental issues:
  - Balance: Choosing size, speed, etc. of each resource so that no ideal time results due to mismatch!
  - Overlapping: implement so that resource can overlap its operation completely with the operation of other resources

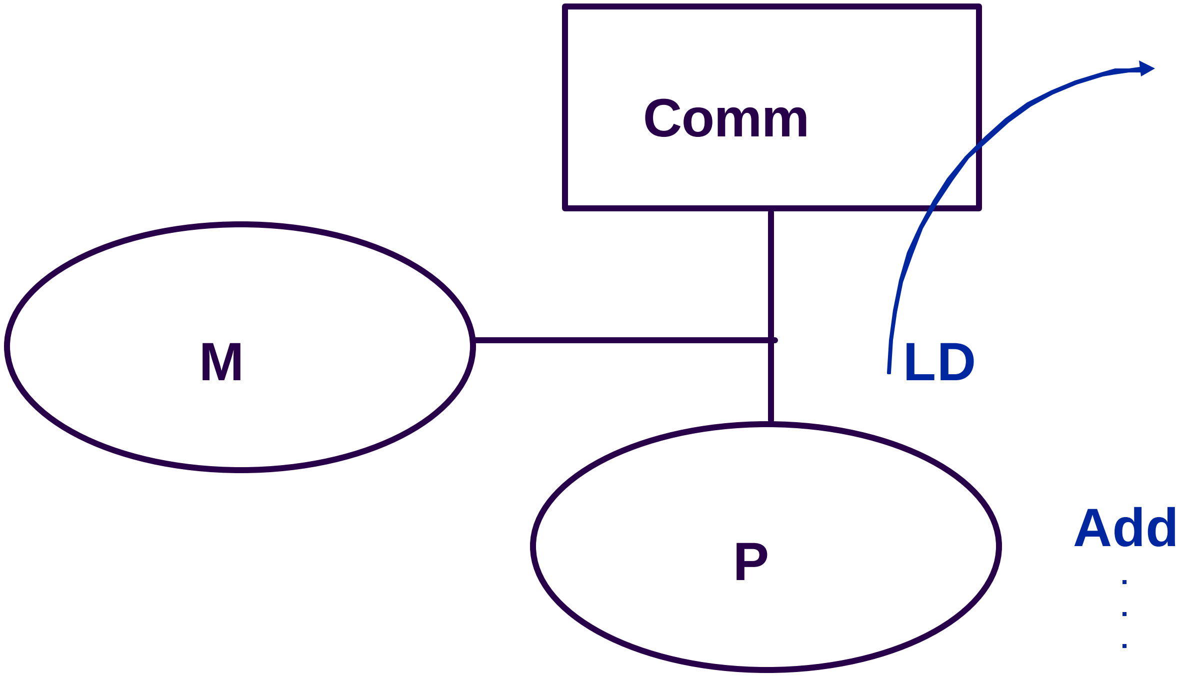
Consider the basic pipeline



- **Overlapping**
  - ***X*, *Y*, *Z* must be able to operate concurrently -- that is, when *X* is performing *OP_X* on *c*, *Y* must be able to perform *OP_y* on *b*, and *Z* must be able to perform *OP_z* on *a*.**
- **Balance**
  - **To avoid wastage or idle time in *X*, *Y*, or *Z*, design each so that:**

$$\mathrm{TIME}(OP_x) = \mathrm{TIME}(OP_y) = \mathrm{TIME}(OP_z)$$

Overlap in multiprocessors



- Processors are able to process other data while communication network is busy processing requests issued previously.

Balance in multiprocessors

- **A machine is balanced if each resource is used fully**
  - **For given problem size**
  - **For given algorithm**
- **Let's work out an example...**

N-body

$\forall$ *Body*

Compute force from

$N - 1$ others

Consider a single node

Application

R_p	**Number of processing OPs**
R_M	**Number of memory words**
R_C	**Number of comm. words**

All parameters are a function of

P: Number of nodes

N: Problem size

More parameters

p:	**Ops/sec**
m:	**Mem size**
c:	**Words/sec**

Function of available budget

Questions

- 1. Given ***N***, ***P*** find ***p***, ***m***, ***c*** for balance
- 2. Suppose $p = 10 \times 10^6$ $P = 10$
  $m = 0.1 \times 10^6$
  $c = 0.1 \times 10^6$
- For what ***N*** do we have balance?
- 3. Suppose $p = 2 \times 10 \times 10^6$, how do we rebalance by changing ***N***
- 4. Given fixed budget ***D*** and size ***N***, find optimal
  - Given: Per node cost of ***p***, ***m***, ***c***, ***P***

Memory: K_m
Proc: K_p
Comm: K_c

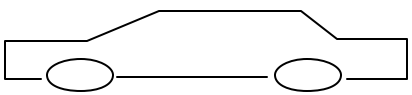
Issue of “size efficiency” - *SE*

- **A machine that requires a larger problem size to achieve balance has a lower *SE* grain size than a machine that achieves balance with a smaller problem size.**

Machine A

p = 10 x 10^6
c = 0.1 x 10^6
m = 100
P = 10
N - body naive

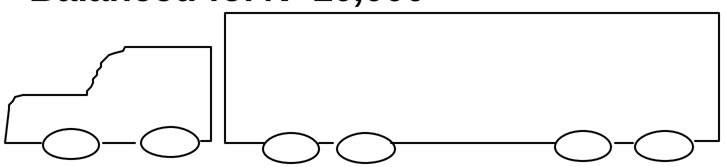
Balanced for *N=1,000*



Machine B

p = 10 x 10^6
c = 0.01 x 10^6
m = 1000
P = 10
N - body naive

Balanced for *N=10,000*



Intuition:

- **For typical problems**

$$\frac{\text{Comm. requirements per node}}{\text{Proc. requirements per node}}$$

Goes ↓ as *N* increases (*P* decreases)

Think of any counter examples?

- **So, machines that provide higher ratio of comm-- compute power tend to have higher *SE*.**
- **What about memory? Machines with small comm -- compute ratios tend to provide more mem. per node.**
  - **We now know why!**
- **However, the RIGHT *SE* is:**
  - **Problem dependent**
  - **Relative cost dependent as well.**

Scalability y

- **What does it mean to say a system is scalable.**
- **TRY: A scalable architecture enjoys speedup proportional to *P*, the number of nodes:**

$$y = \frac{T(1)}{T(P)} \propto P \text{ for scalable arch.}$$

- **If problem size is fixed at *N***
  - ***T(P)* will not decrease beyond some *P* [assuming some unit of computation**
  - **For example add, beyond which we do not attempt to parallelize algorithms].**

Scalability

$$\psi(N) = \frac{S_R(N)}{S_I(N)} = \frac{\text{Asymptotic speedup on machine}}{\text{Asymptotic speedup on EREW PRAM}}$$

- N is problem size

- **Asymptotic speedup for machine *R***

$$S_R(N) = \frac{\theta(\text{Serial running time})}{\theta(\text{Parallel running time})}$$

(Make problems very large)

Minimum

Computed using as many nodes as necessary to yield best running time

- **Intuitively, $\psi(N)$: Fraction of parallelism inherent in a given algorithm that can be exploited by any machine of that architecture, as a function of problem size *N*.**

- **Intuitively, $S_I(N)$: Maximum speedup achievable on any machine of the given architecture.**

Intuition

- Maximum speedup achievable on any sized machine of the given architecture

$$S_R(N)$$

- Fraction of parallelism inherent in a given algorithm that can be exploited by any machine of that architecture as a function of problem size ***N***.

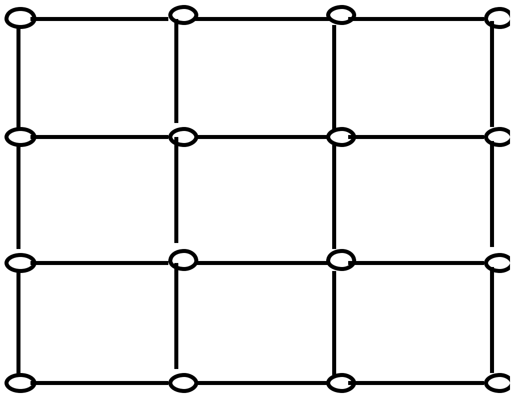
$$y\,(N)$$

Scalability

- **Example: The (by now) famous Jacobi**
- **<u>2D Mesh</u>**

$$S_I(N) = Q(N)$$
$$S_R(N) = Q(N)$$

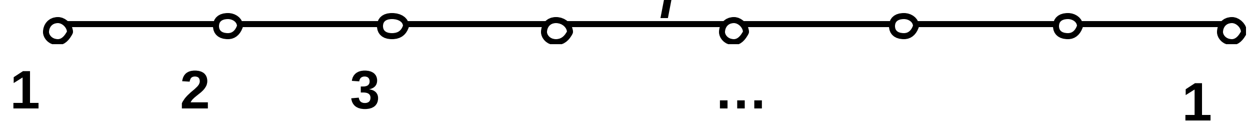
$$y\,(N) = 1$$

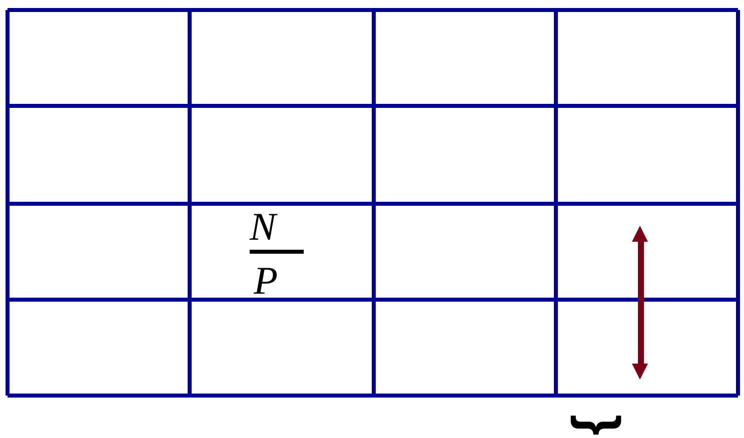


- **i.e. Mesh is 1-scalable for the Jacobi relaxation step?**

1-D Array of nodes for Jacobi

$$\frac{N}{P}\ ops$$





Comm

Comm $\sqrt{\frac{N}{P}}$

Model: 1 op, 1cycle
1comm/hop, 1cycle

$$T = \frac{N}{P} + \sqrt{P}$$

Scalability

- $$S_I(N) = N$$ $$S_R(N) = ?$$
- **Ideal speedup on any number of procs.**
- **<u>Find best *P*</u>**

$$T_{par} = \frac{N}{P} + \sqrt{P}$$
$$\frac{dT}{dP} = 0$$
$$P = N^{\frac{2}{3}} \ldots$$
$$T_{par} = q\left(N^{\frac{1}{3}}\right)$$
$$T_{seg} = N$$
$$S_R(N) = N^{\frac{2}{3}} = \frac{N}{N^{\frac{1}{3}}}$$

- **So,** $$y(N) = \frac{N^{\frac{2}{3}}}{N} = \frac{S_R(N)}{S_I(N)} = N^{-\frac{1}{3}}$$
- **So, 1-D array is** $\frac{1}{N^{\frac{1}{3}}}$ **scalable for Jacobi**

Solutions to Balance Questions

- **1.**

$$\boxed{N/P} \quad \begin{array}{l} N/P \\ N - N/P \end{array}$$

$$R_P = \frac{N^2}{P}, \quad R_M = \frac{N}{P}, \quad R_C = N$$

for balance:

$$T_{proc} = T_{comm} \text{ , memory full}$$

or:

$$\frac{R_P}{p} = \frac{R_C}{c}, \quad R_M = m$$

Yields *p/c* ratio:

$$\frac{N^2}{Pp} = \frac{N}{c} \quad \text{or} \quad \frac{p}{c} = \frac{N}{P}$$

$$T = \frac{R_P}{p} = \frac{N^2}{Pp}$$

Detailed Example

$$p = 10 \times 10^6$$
$$c = 0.1 \times 10^6$$
$$m = 0.1 \times 10^6$$
$$P = 10$$

$$\frac{p}{c} = \frac{N}{P}$$

or $$\frac{10 \times 10^6}{0.1 \times 10^6} = \frac{N}{10}$$

or $N = 1000$ for balance

also $$R_M = m$$
$$\frac{N}{P} = m$$
$$\frac{1000}{10} = 100 = m$$

Memory size of m = 100 yields a balanced machine.

Twice as fast processor

$$\frac{p}{c} = \frac{N}{P}$$

If $p \rightarrow 2p$, $N \rightarrow 2N$
$m \rightarrow 2m$

Double problem size.

Find Optimal Machine

a. **Optimize**

Subject to:

$$T = \frac{N^2}{Pp}$$

b. **Constraint (cost)**

$$D = [K_p + K_m + K_c] P$$

$$D = [pK_{ps} + m K_{ms} + c K_{cs}] P$$

c. **At opt, balance constraints satisfied, makes solution easier to find but not strictly needed.**

$$\frac{p}{c} = \frac{N^2}{Pp}, \quad m = \frac{N}{P}$$

- **(Opt process should discover c. if not supplied.)**

Eliminate unknowns

$$D = \left[pK_{ps} + mK_{ms} + cK_{cs} \right] P$$

$$D = \left[pK_{ps} + \frac{N}{P} K_{ms} + \frac{pP}{N} K_{cs} \right] P$$

or

$$p = \frac{D - NK_{ms}}{P\left[K_{ps} + \frac{P}{N} K_{cs}\right]}$$

substitute in

$$T = \frac{N^2}{P_P}$$

$$T = \frac{N^2\left(K_{Ps} + \frac{P}{N} K_{cs}\right)}{D - NK_{ms}}$$

***T* minimized when *P*=1!**

Summary

- Balance between communication, memory, and computation
- Different measures of scalability
- Problem often has specific optimal machine configuration
- Balance can be shown analytically for variety of machines
- Meshes, cubes, and linear arrays appropriate for various problems