

# Parallel Computer Architecture

## Part I

(Part 2 on March 18)

# Latency

- Defn: Latency is the time it takes one message to travel from source to destination. Includes various overheads.
- E.g. Taking the T (or the SMRT )
  - Time to walk out the door to the station
  - Time to wait for train
  - Time on the train
  - Time to walk to destination

# Latency

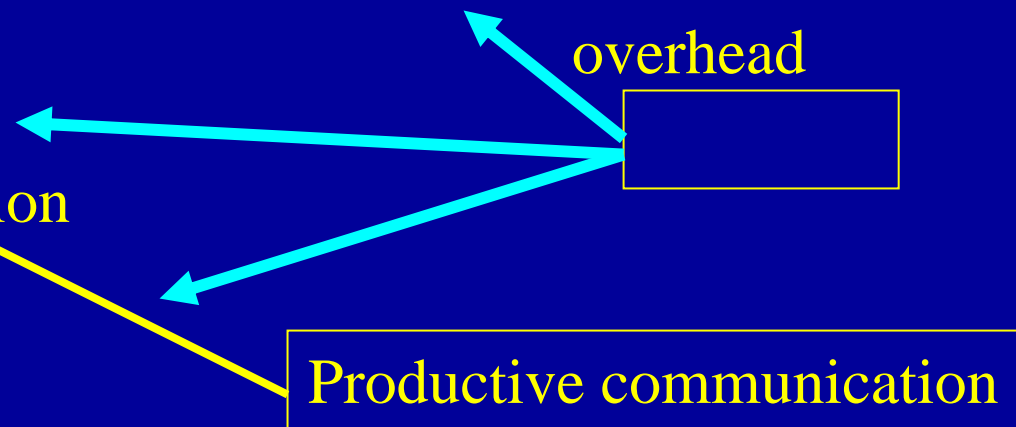
- Defn: Latency is the time it takes one message to travel from source to destination. Includes various overheads.
- E.g. Taking the T (or the SMRT )
  - Time to walk out the door to the station
  - Time to wait for train
  - Time on the train
  - Time to walk to destination



overhead

# Latency

- Defn: Latency is the time it takes one message to travel from source to destination. Includes various overheads.
- E.g. Taking the T (or the SMRT )
  - Time to walk out the door to the station
  - Time to wait for train
  - Time on the train
  - Time to walk to destination



# Latency

- Defn: Latency is the time it takes one message to travel from source to destination. Includes various overheads.
- E.g. Taking the T (or the SMRT )
  - Time to walk out the door to the station
  - Time to wait for train
  - Time on the train
  - Time to walk to destination

Productive communication



# Bandwidth

- Maximum Rate at which the network can propagate information
- E.g. (speed of train) \* (number of passengers)

# Important

1. Bandwidth is easy to measure and easy to hype. It is an idealized peak communications rate most likely to be achieved for long messages under conditions of no traffic contention.
  - (Long train ride, no delays.)
  - Ethernet = 100 Mbps
2. Latency varies with applications, conditions, etc.
  - (Train delays, rush hour, the weather, ...)
3. Latency = How long it takes you to get to work.  
Bandwidth = How many miles you can ideally move people / unit time.
4. Latency & Bandwidth are related but not the same. Latency includes more realities!

# Latency: the details

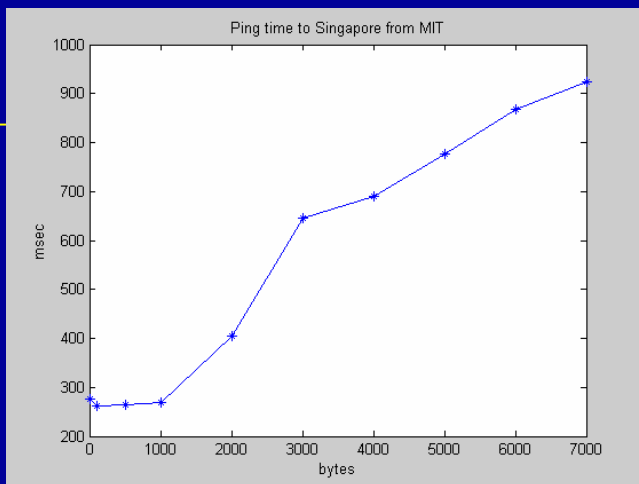
1.  $\text{Latency} = \text{Sender Overhead} + \text{Time of Flight} + \text{Transmission Time} + \text{Receiver Overhead}$
2. Sender Overhead: Time for processor to inject message into network. Time processor tied up from doing useful work because it is sending.
3. Time of Flight: Time for the start of the message to arrive at the receiver.
4. Transmission Time: Time for remainder of message to arrive at receiver =  $(\# \text{ bits}) / \text{bandwidth}$
5. Receiver Overhead: Time for receiver to pull message (usually larger than sender overhead)

# Latency vs Bandwidth on the internet

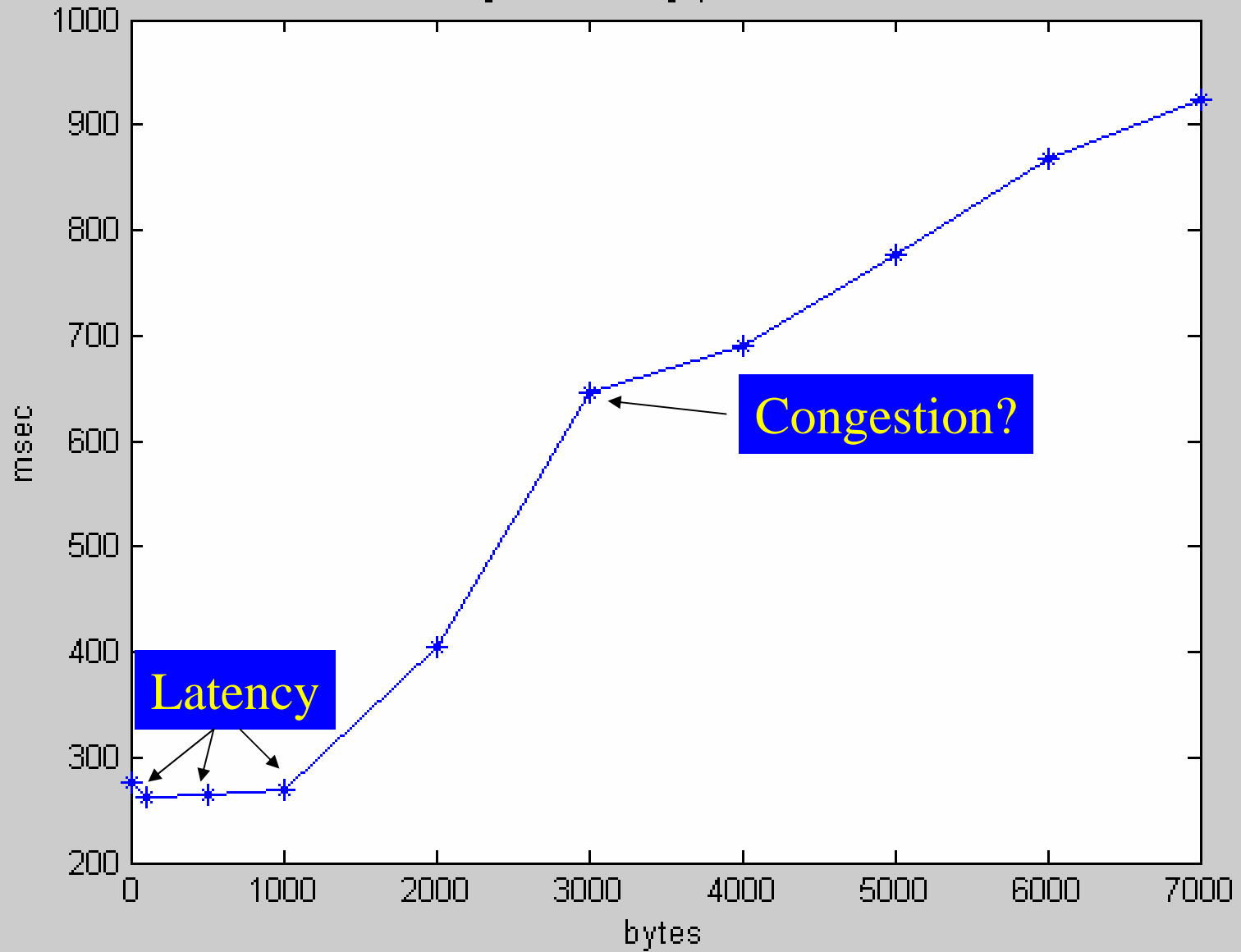
Experiment: Sending packets roundtrip from MIT to Singapore

- Specifications

- Time: 5am EST this morning (Feb 20) = 6pm Singapore
- Source: pythagoras.mit.edu (ip address = 18.87.0.29)
- Dest: sunsvr.comp.nus.edu.sg (ip address = 137.132.88.6)
- Method: ping -s 137.132.88.6 num\_bytes 5 (5 = num\_trials)
- Data: bytes: 8 100 500 1000 2000 3000 4000 5000 6000 7000
- msec: 276 263 264 270 404 645 690 777 868 923



Ping time to Singapore from MIT



# Congestion (Traffic!)

Latency and Bandwidth are not the whole story, congestion can also slow down communication!

# Node Architecture

1. CPU, e.g. Athlon MP1800

- Registers:  $O(400B)$  Speed:  $O(1nsec)$

2. Cache (on chip)

- L1: 64 KB Instruction + 64 KB Data
- L2: 256 KB Data  $O(10\ nsec)$

↓ Memory Bus ↓

3. Memory: e.g. 1 GB PC2100 DDR EEC

$O(100nsec)$  (Double Data Rate/Extended Error Correction)

↓ I/O Bus ↓

4. Disk: IBM 120GXP 40GB  $O(5msec)$

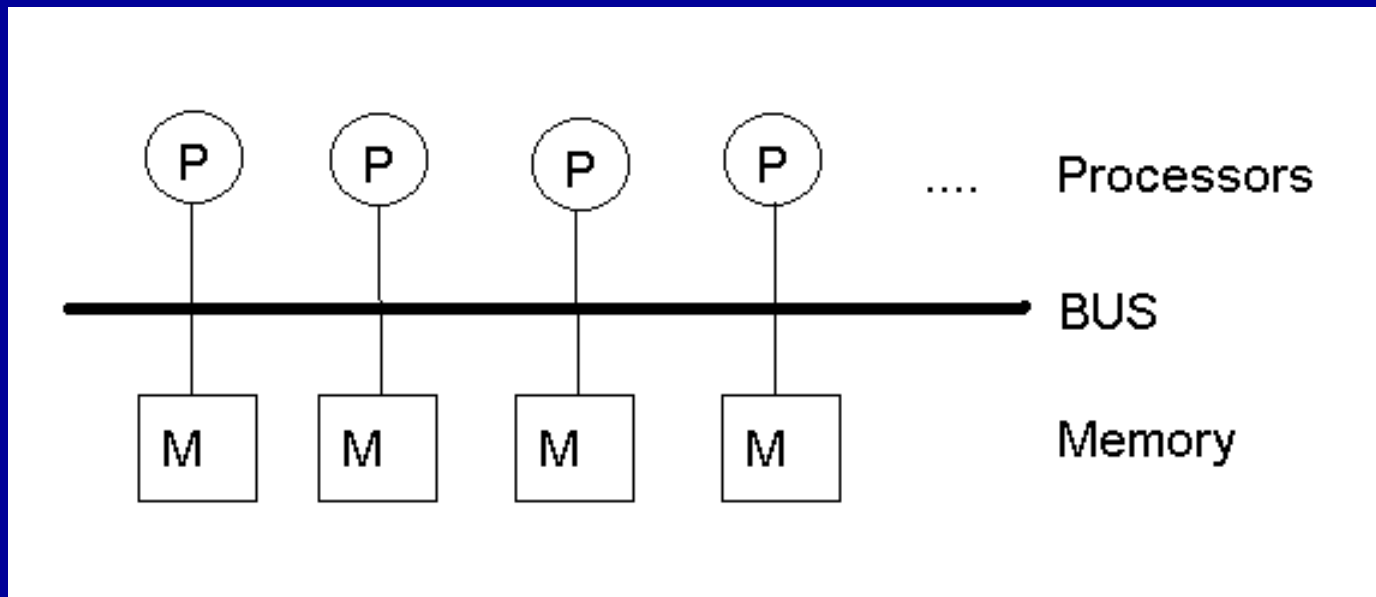
Warning: all O's are just guesses made in the early morning

# Node Architecture

|            |         |            |   |
|------------|---------|------------|---|
| 1. CPU     | O(400B) | O(1nsec)   |   |
| 2. Cache   |         |            |   |
| • L1:      | 128 KB  |            |   |
| • L2:      | 256 KB  | O(10 nsec) |   |
|            | ↓       | Memory Bus | ↓ |
| 3. Memory: | 1 GB    | O(100nsec) |   |
|            | ↓       | I/O Bus    | ↓ |
| 4. Disk:   | 40GB    | O(5msec)   |   |

Warning: all O's are just guesses made in the early morning


# Bus as an interconnect network



- One transaction at a time between source and destination
- The “P”’s and “M”’s could be cache and memory, disk and memory, etc.

# • Where does the network meet the processor?

Distance  
to Proc

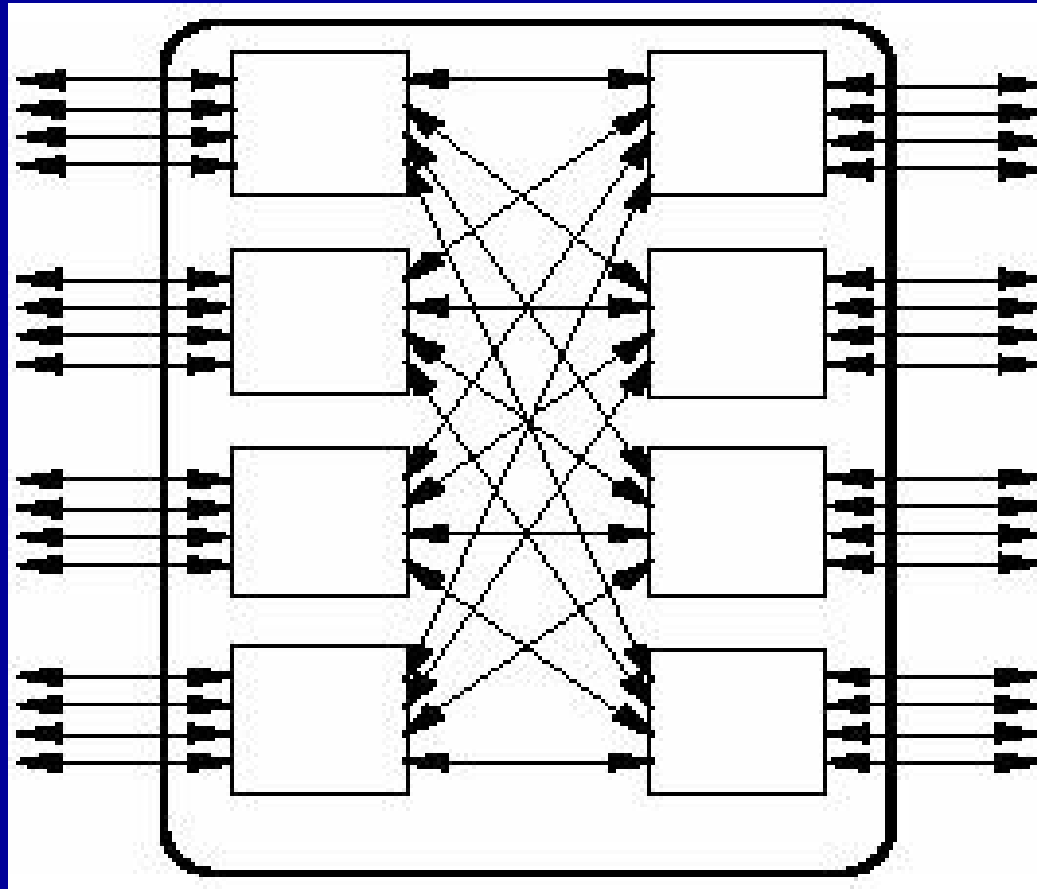


1. I/O Bus
  - Most Clusters
2. Memory Bus
  - Many MPPs
3. Processor Registers
  - RAW architecture at MIT

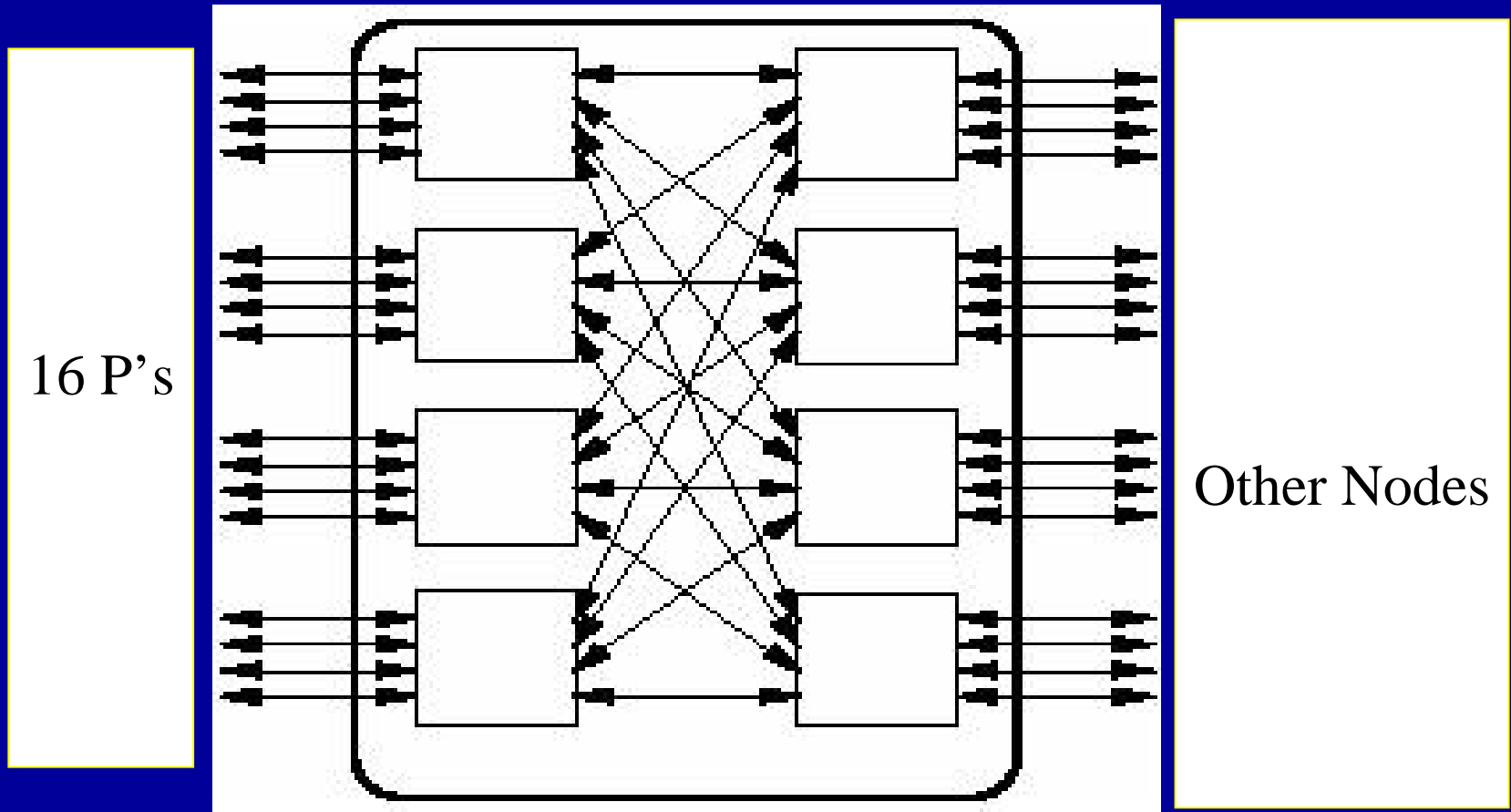
# Asci White at Lawrence Livermore (Currently #1)

- IBM SP System: SP = Scalable Powerparallel
- 512 “High Nodes” ({thin,wide,high}=physical space)
- 1 High Node = 16 Processor SMP Connected by 1 SP Switch
- Weight = 106 tons = roughly 100 Honda Civics!
- Processor = RS/6000 Power 3
  - RS=Risc System= “Reduced Instruction Set Chip/Comp”
- SP Switch connects to 16 nodes and up to 16 other switches

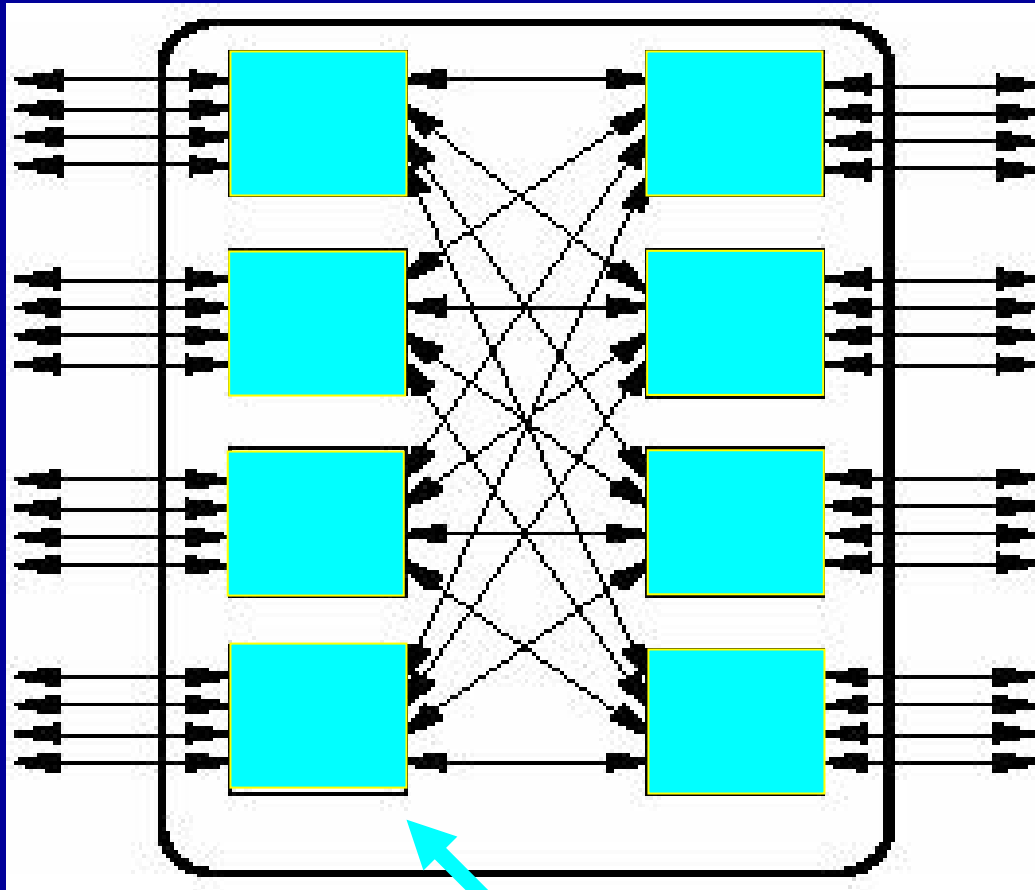
# IBM Sp Switch (Up to 16 P's)



# IBM Sp Switch (Up to 16 P's)

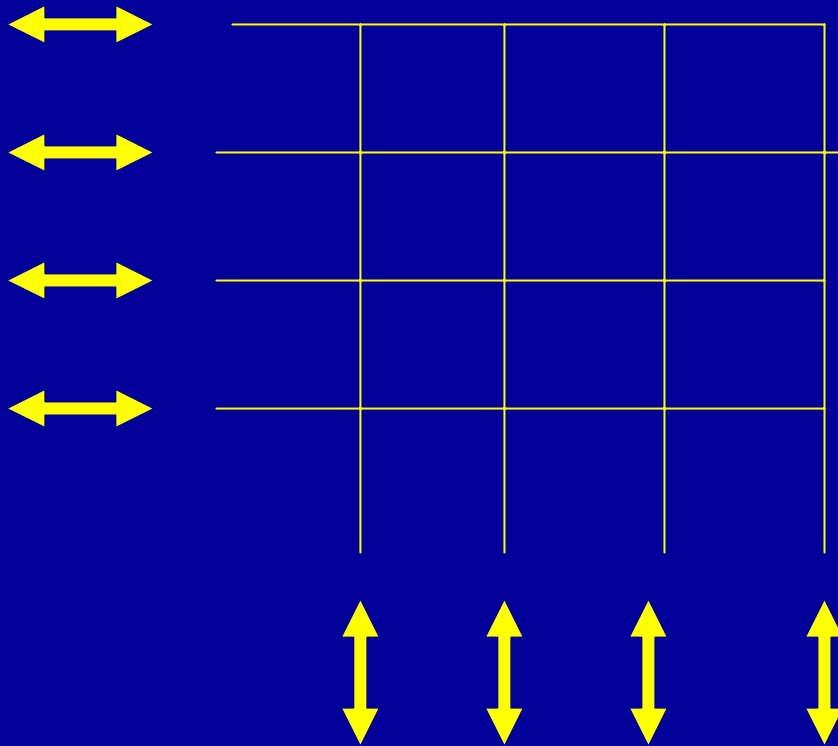


# IBM Sp Switch (Up to 16 P's)



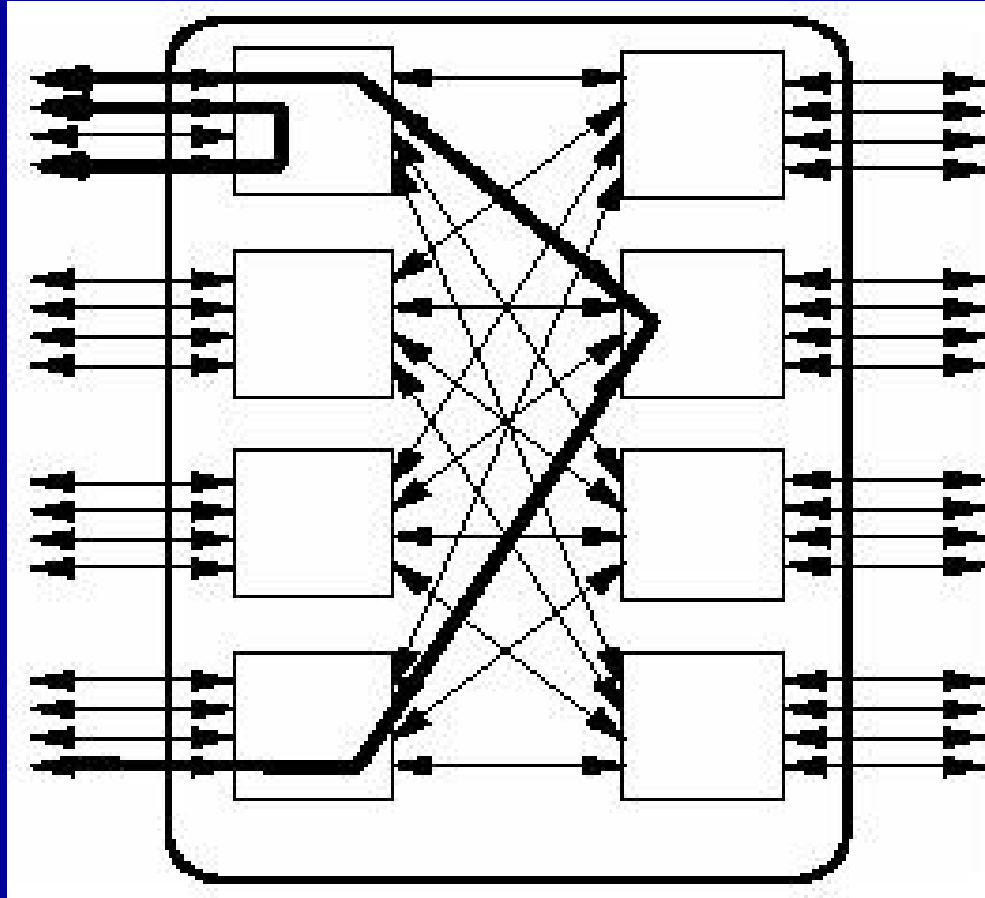
8 switch chips per switch

# Cross Bar



I'm guessing that every switch chip is a crossbar – someone check please -- maybe not???

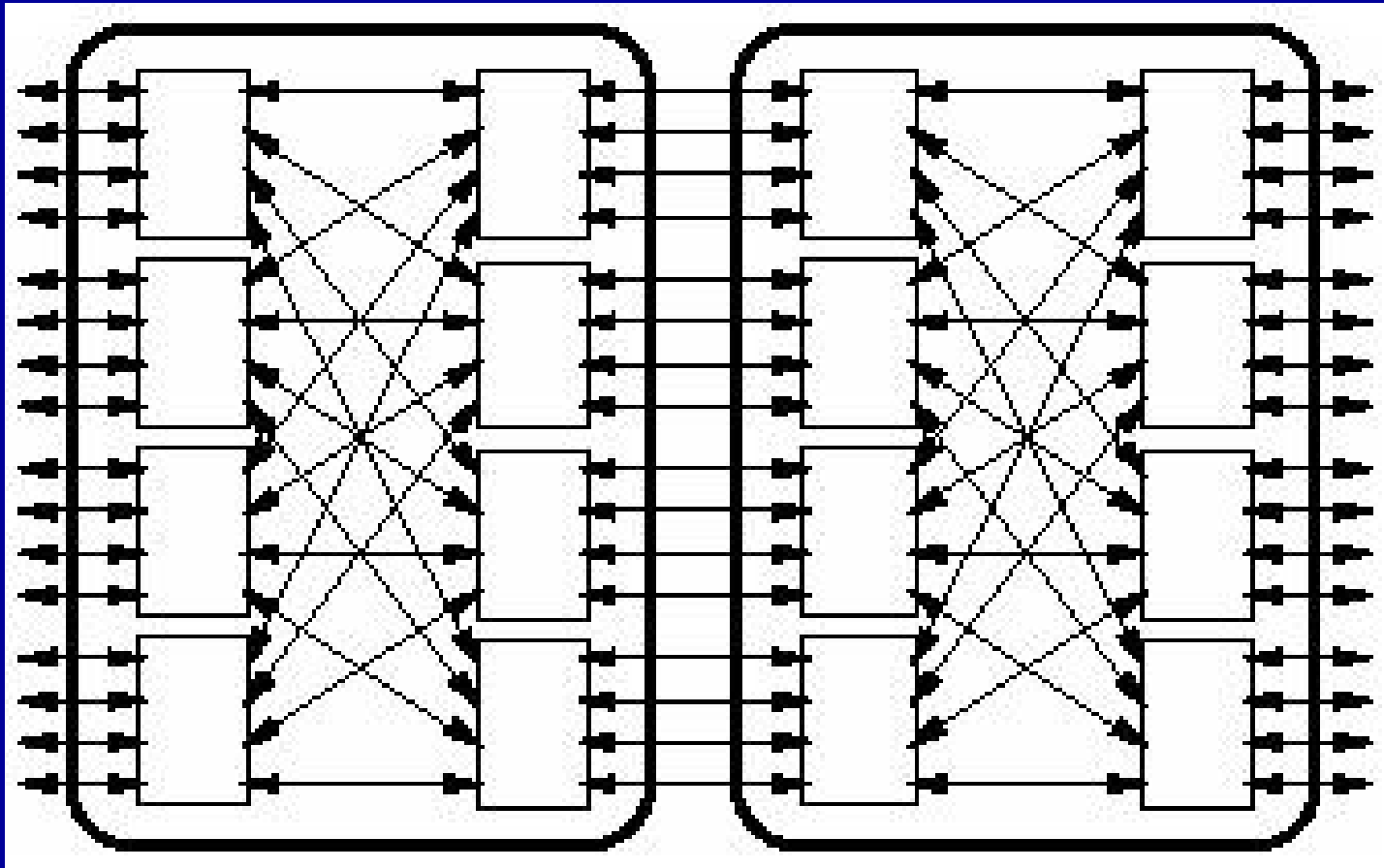
# Multiple Data Flows on an SP Switch



Four “on-chip” stay on chip, off-chip pairs take three hops but not typically much slower

# Connecting two 16-way SMPs =32 processors

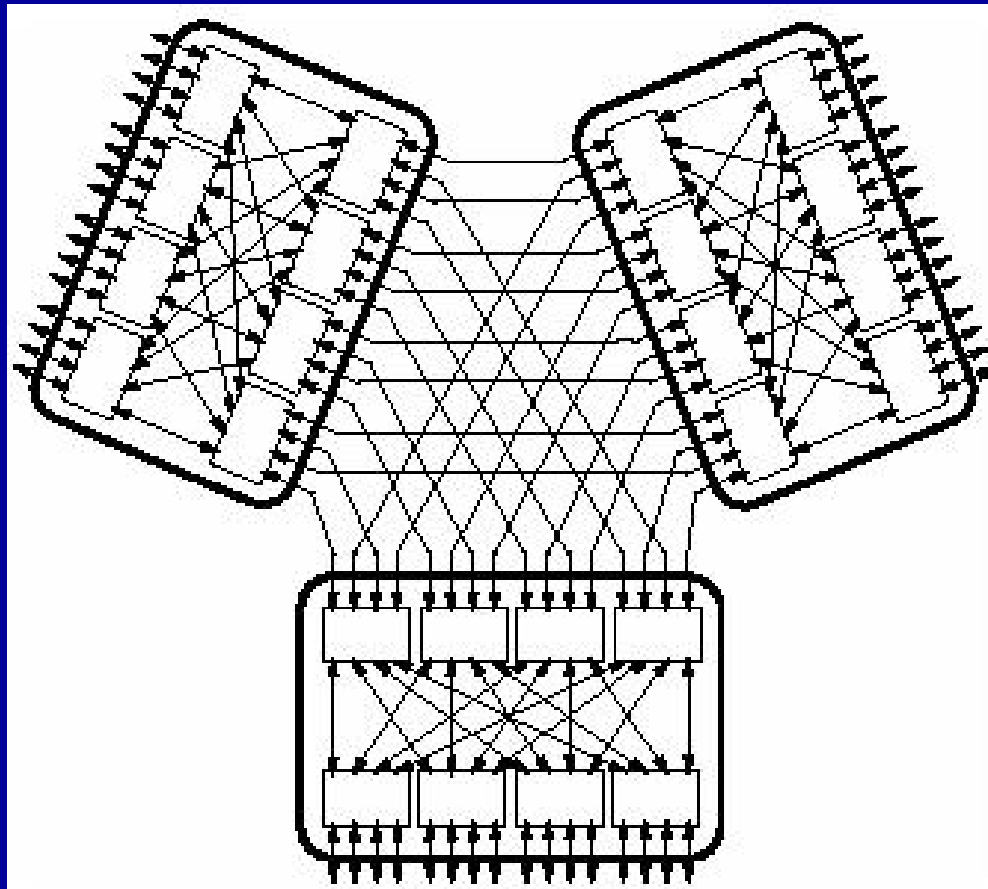
SMP



SMP

# Connecting three 16-way SMPs =48 processors

SMP



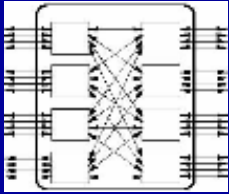
SMP

Note: every  
pair share 8  
wires

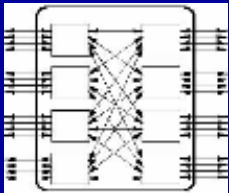
SMP

# Connecting five 16-way SMPs =80 processors

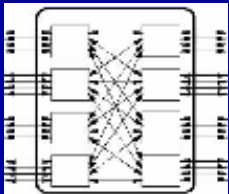
SMP



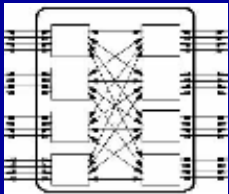
SMP



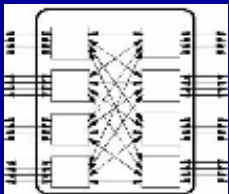
SMP



SMP



SMP



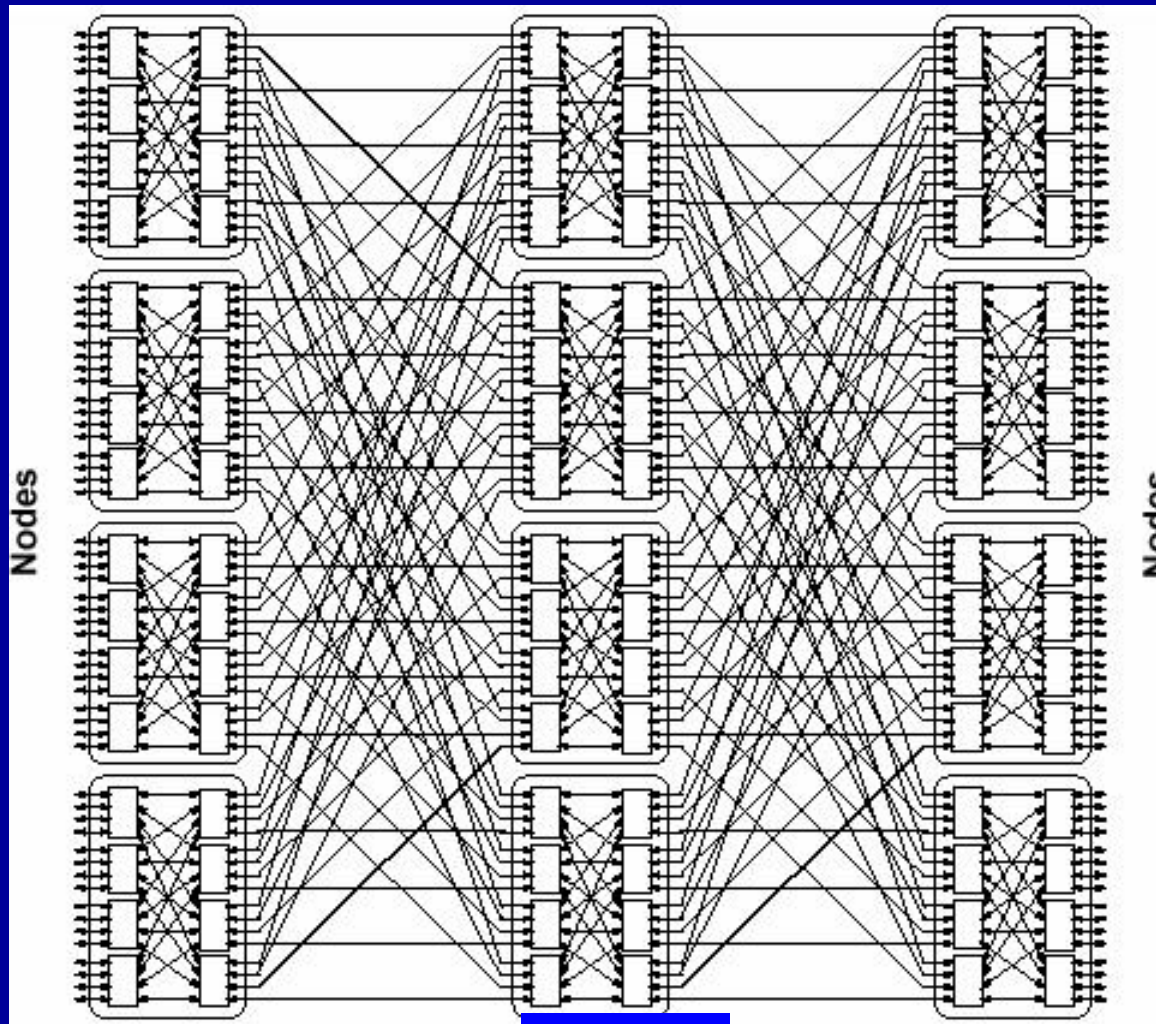
Every pair of SP chips not already on the same board are connected.

There are 16 off board SP chips. Total of 40 wires.

(This is a guess but the only logical one, I think)

Always 4 parallel paths between any two processors!

# Connecting eight 16-way SMPs =128 processors

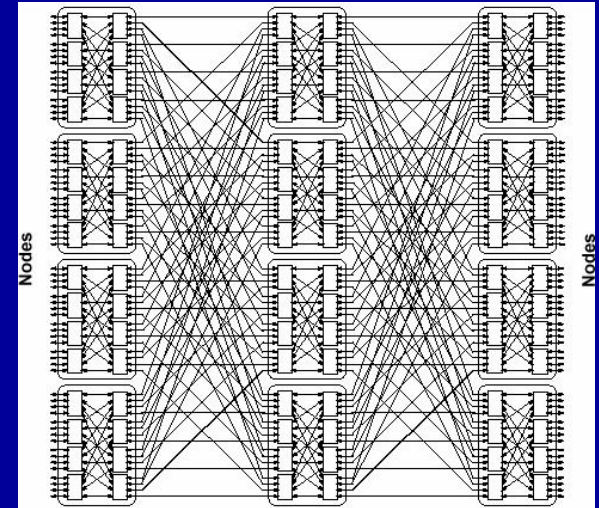
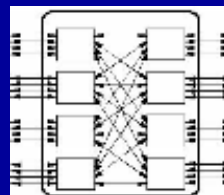
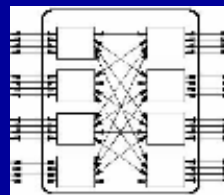
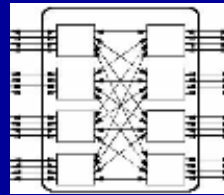
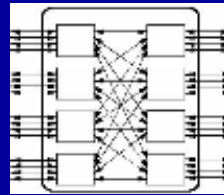
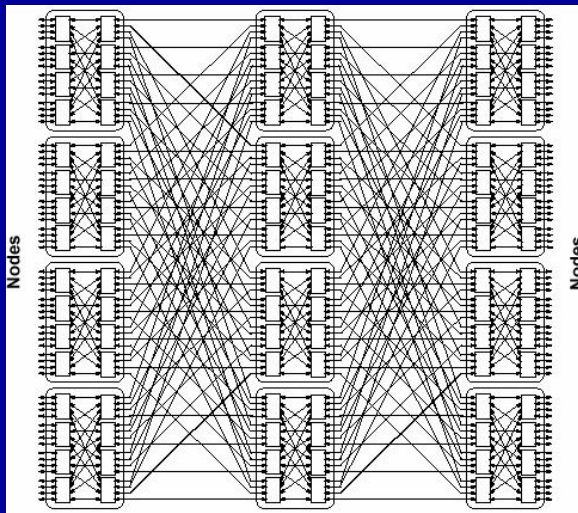


Intermediate  
switch  
boards  
needed for  
81-128  
nodes

Each chip  
connected to  
one ISB!

ISB's

# Guess at doubling



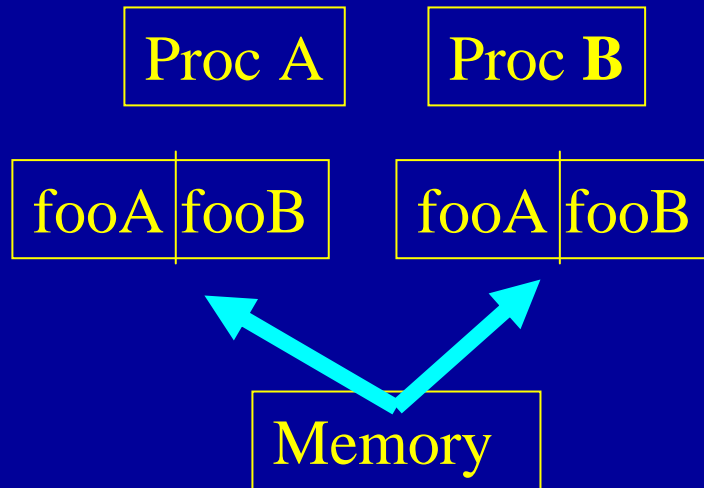
What must be done?

# Caches: A means of solving the memory to processor latency problem

- Memory just can't get out the door fast enough for the processor to handle
- Not all the data is needed all the time
- The data that is needed can often be predicted!
- Solution: faster (= more expensive) but smaller memory in between

# Cache Coherence

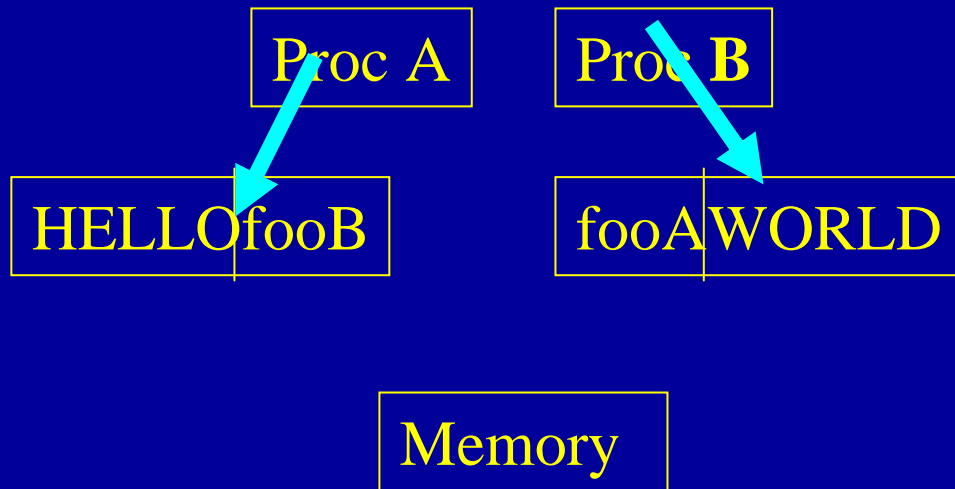
## Coherence Error Due to False Sharing



Both A and B get same line  
A only needs fooA,  
B only needs fooB

# Cache Coherence

## Coherence Error Due to False Sharing

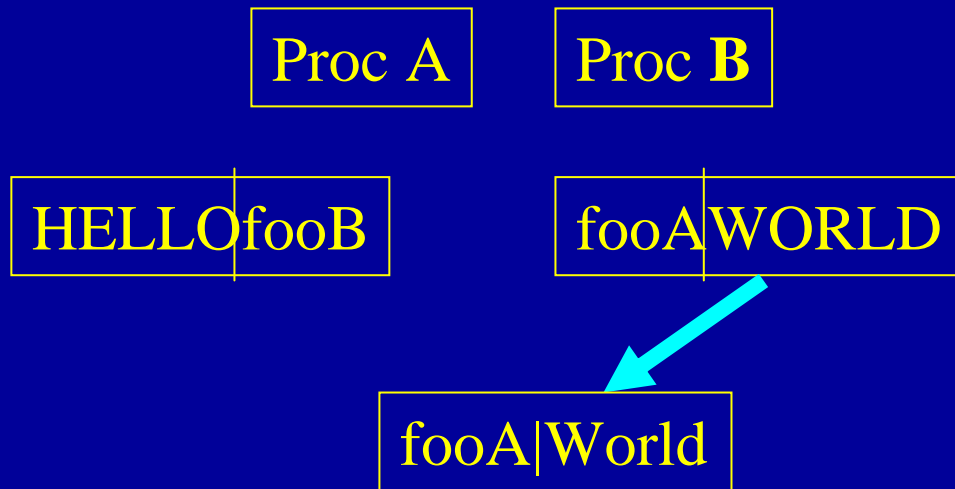


- Both A and B get same line  
A only needs fooA,  
B only needs fooB

- A writes HELLO  
B writes WORLD

# Cache Coherence

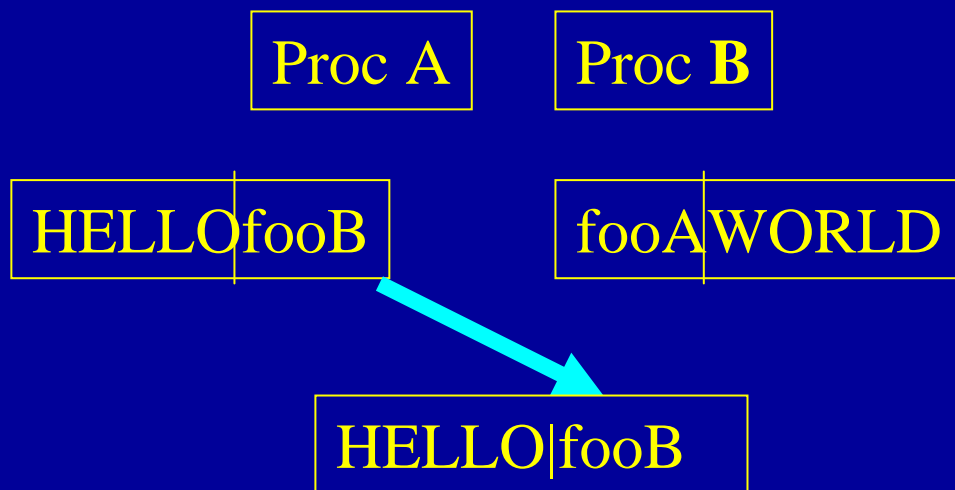
## Coherence Error Due to False Sharing



- Both A and B get same line  
A only needs fooA,  
B only needs fooB
- A writes HELLO  
B writes WORLD
- B writes cache to memory

# Cache Coherence

## Coherence Error Due to False Sharing



- Both A and B get same line  
A only needs fooA,  
B only needs fooB
- A writes HELLO  
B writes WORLD
- B writes cache to memory
- A writes cache to memory