# Lecture 1: Parallel Architecture Intro

- Course organization:
  - ~5 lectures based on Culler-Singh textbook
  - ~5 lectures based on Larus-Rajwar textbook
  - ~4 lectures based on Dally-Towles textbook
  - ~10 lectures on recent papers
  - ~4 lectures on parallel algorithms and multi-thread programming

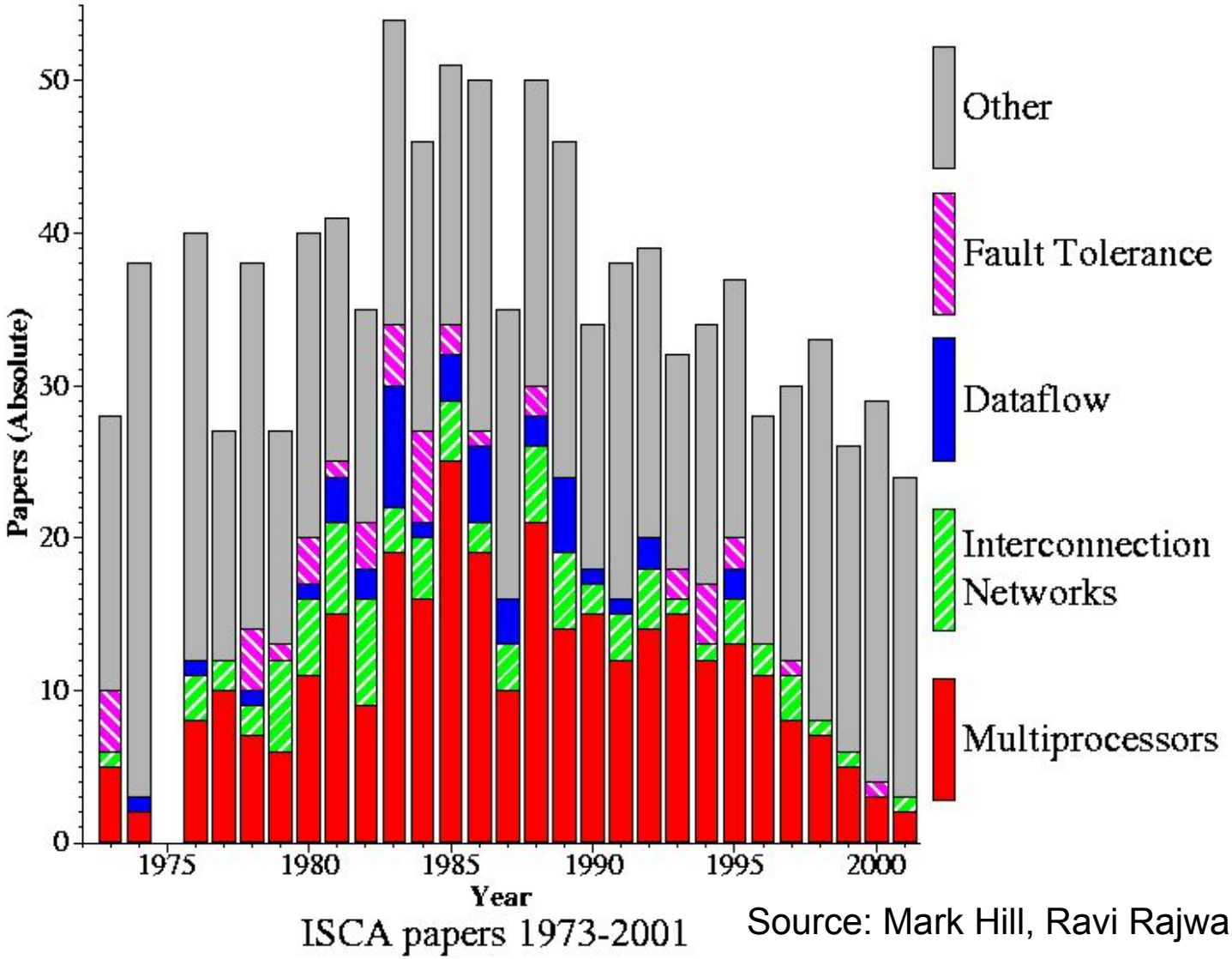
- Texts: Parallel Computer Architecture, Culler, Singh, Gupta
  Principles and Practices of Interconnection Networks, Dally & Towles
  Introduction to Parallel Algorithms and Architectures, Leighton
  Transactional Memory, Larus & Rajwar

# More Logistics

- Projects: simulation-based, creative, be prepared to spend time towards end of semester – more details on simulators in a few weeks

- Grading:
  - 50% project
  - 20% multi-thread programming assignments
  - 10% paper critiques
  - 20% take-home final

# Parallel Architecture Trends




ISCA papers 1973-2001

Source: Mark Hill, Ravi Rajwar

# CMP/SMT Papers

- CMP/SMT/Multiprocessor papers in recent conferences:

| | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 |
|---|---|---|---|---|---|---|---|
| ISCA: | 3 | 5 | 8 | 6 | 14 | 17 | 19 |
| HPCA: | 4 | 6 | 7 | 3 | 11 | 13 | 14 |

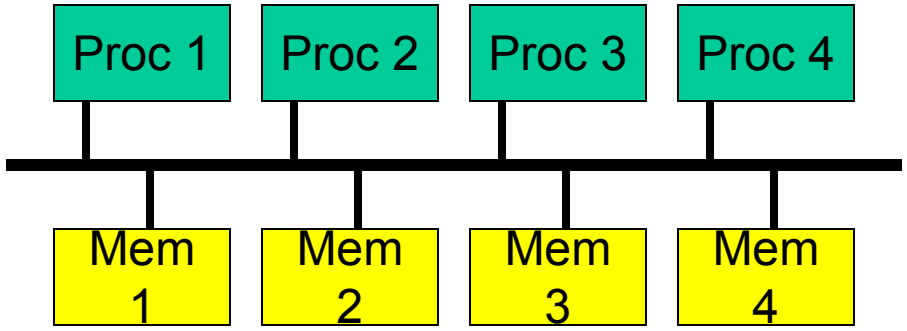
# Bottomline

- Can't escape multi-cores today: it is the baseline architecture
- Performance stagnates unless we learn to transform traditional applications into parallel threads
- It's all about the data!
  Data management: distribution, coherence, consistency
- It's also about the programming model: onus on application writer / compiler / hardware
- It's also about managing on-chip communication

# Symmetric Multiprocessors (SMP)

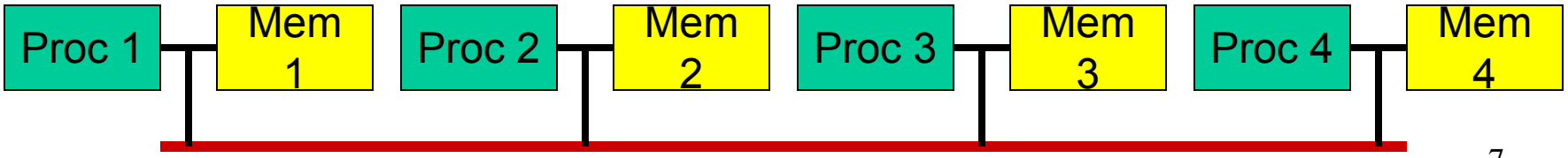
- A collection of processors, a collection of memory: both are connected through some interconnect (usually, the fastest possible)

- Symmetric because latency for any processor to access any memory is constant – uniform memory access (UMA)



# Distributed Memory Multiprocessors

- Each processor has local memory that is accessible through a fast interconnect
- The different nodes are connected as I/O devices with (potentially) slower interconnect
- Local memory access is a lot faster than remote memory – non-uniform memory access (NUMA)
- Advantage: can be built with commodity processors and many applications will perform well thanks to locality

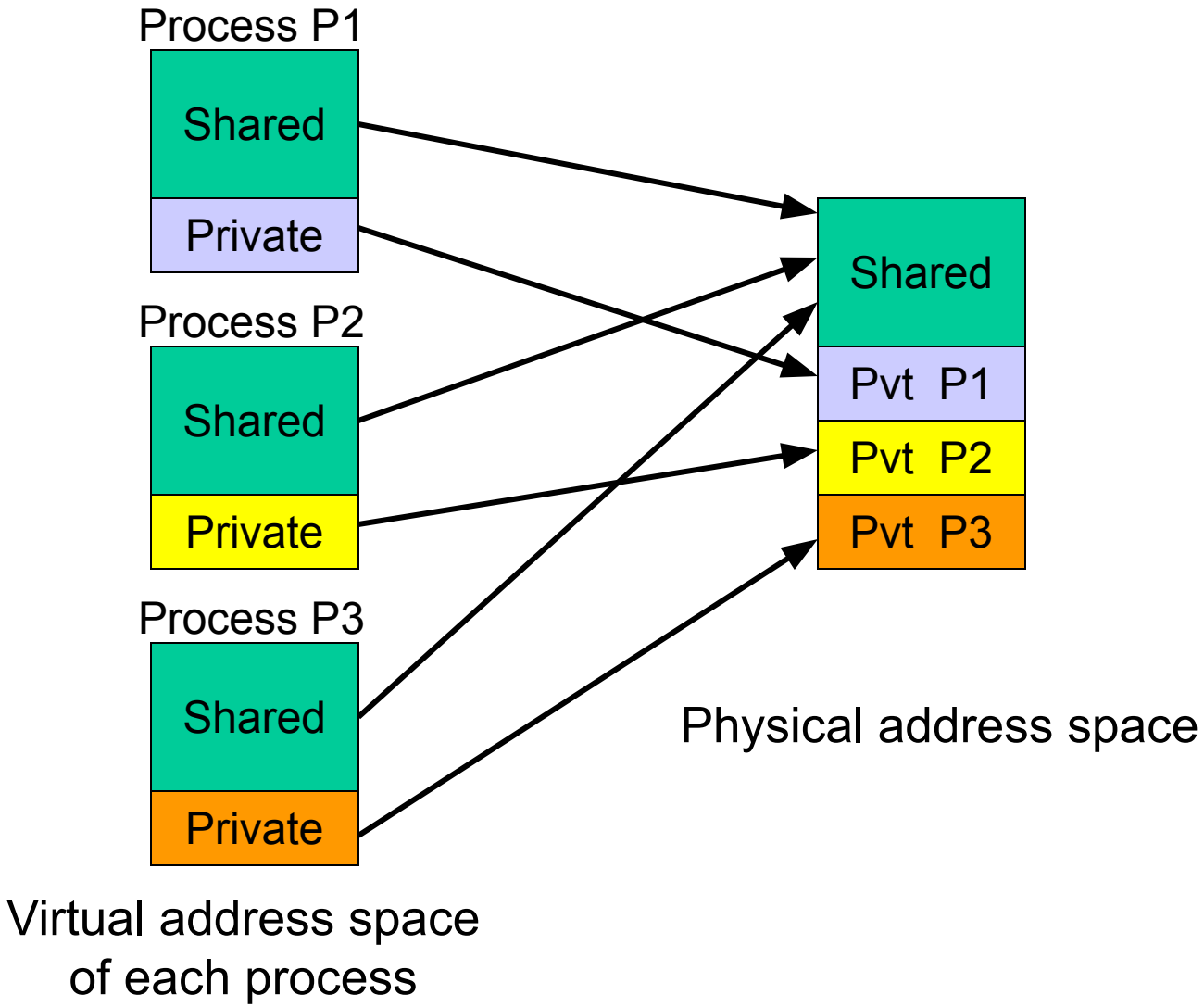


# Shared Memory Architectures

- Key differentiating feature: the address space is shared,
  i.e., any processor can directly address any memory location and access them with load/store instructions

- Cooperation is similar to a bulletin board – a processor writes to a location and that location is visible to reads by other threads

# Shared Address Space



# Message Passing

- Programming model that can apply to clusters of workstations, SMPs, and even a uniprocessor
- Sends and receives are used for effecting the data transfer – usually, each process ends up making a copy of data that is relevant to it
- Each process can only name local addresses, other processes, and a tag to help distinguish between multiple messages
- A send-receive match is a synchronization event – hence, we no longer need locks or barriers to co-ordinate

# Models for SEND and RECEIVE

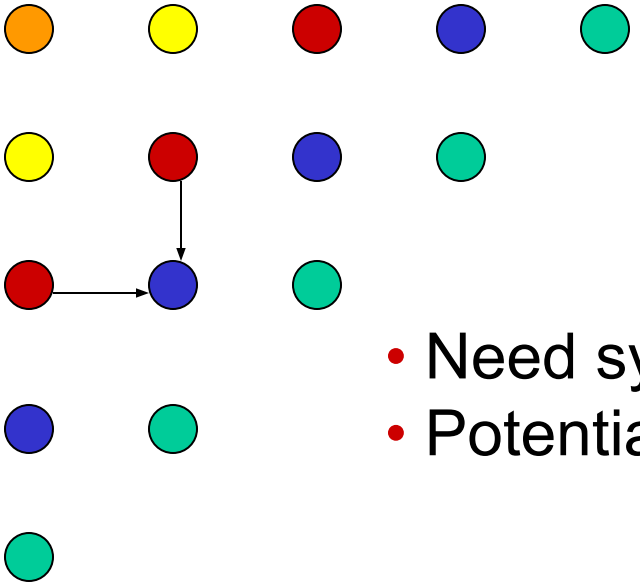
- Synchronous: SEND returns control back to the program only when the RECEIVE has completed

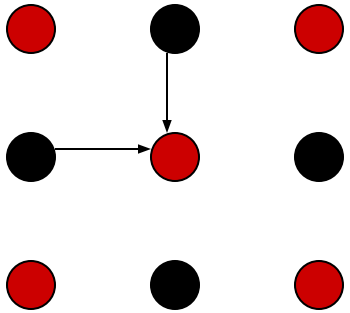
- Blocking Asynchronous: SEND returns control back to the program after the OS has copied the message into its space -- the program can now modify the sent data structure

- Nonblocking Asynchronous: SEND and RECEIVE return control immediately – the message will get copied at some point, so the process must overlap some other computation with the communication – other primitives are used to probe if the communication has finished or not

# Deterministic Execution

- Shared-memory vs. message passing
- Function of the model for SEND-RECEIVE
- Function of the algorithm: diagonal, red-black ordering





- Need synch after every anti-diagonal
- Potential load imbalance

# Cache Coherence

A multiprocessor system is cache coherent if

- a value written by a processor is eventually visible to reads by other processors – write propagation

- two writes to the same location by two processors are seen in the same order by all processors – write serialization

# Cache Coherence Protocols

- Directory-based: A single location (directory) keeps track of the sharing status of a block of memory

- Snooping: Every cache block is accompanied by the sharing status of that block – all cache controllers monitor the shared bus so they can update the sharing status of the block, if necessary

- Write-invalidate: a processor gains exclusive access of a block before writing by invalidating all other copies
- Write-update: when a processor writes, it updates other shared copies of that block

# Title

- Bullet