# Lecture 1: Parallel Architecture Intro

- Course organization:
  - ➢ ~18 parallel architecture lectures (based on text)
  - ➢ ~10 (recent) paper presentations
  - ➢ 2 lectures on parallel algorithms
  - ➢ student project presentations

- Textbook: Parallel Computer Architecture, Culler, Singh

- Projects: simulation-based, creative, be prepared to spend time towards end of semester

- Grading: 50% project, 10% paper presentation, 10% class participation, 30% take-home final

Parallel Architecture History

- A parallel computer is a collection of processing elements that communicate and cooperate to solve large problems fast

- Up to 80's: intended for large applications, costing millions of dollars

- 90's: every organization owns a "cluster" or an "SMP", costing an order of magnitude less

- 21st century: most desktops/workstations will have multiple cores and simultaneous threads on a single die – perhaps, motivated by designers' inability to scale single core performance

Parallel Architecture Trends
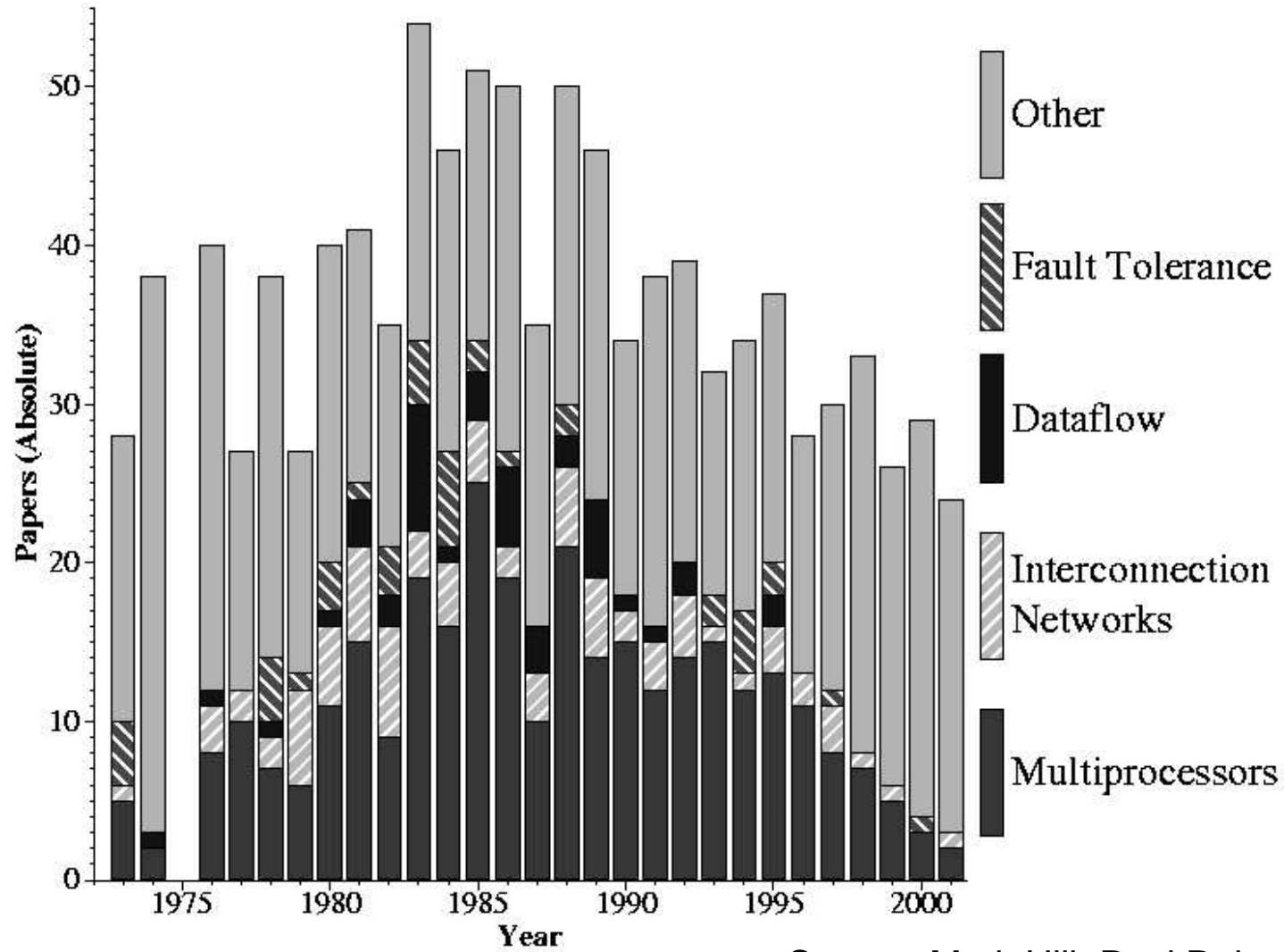
ISCA papers 1973-2001

Source: Mark Hill, Ravi Rajwar   3

CMP/SMT Papers
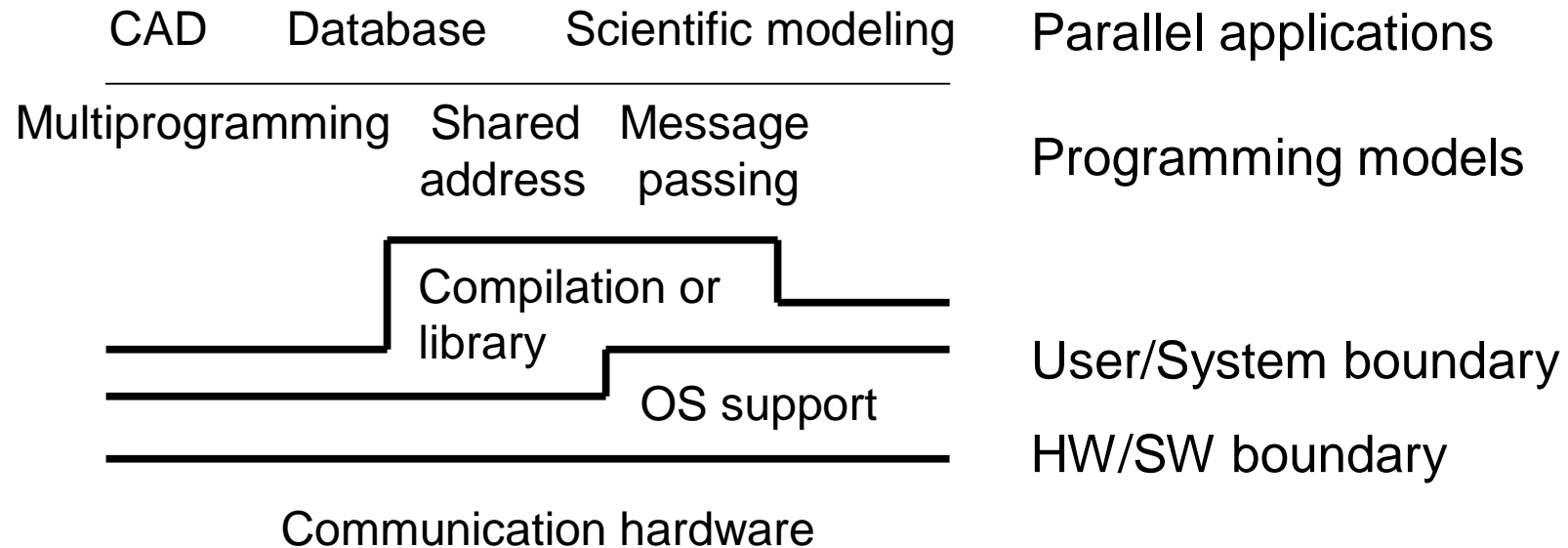
- CMP/SMT/Multiprocessor papers in recent conferences:

	2001	2002	2003	2004	2005
➢ ISCA:	3	5	8	6	
➢ HPCA:	4	6	7	3	11

Bottomline

- Expect an increase in multiprocessor papers

- 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
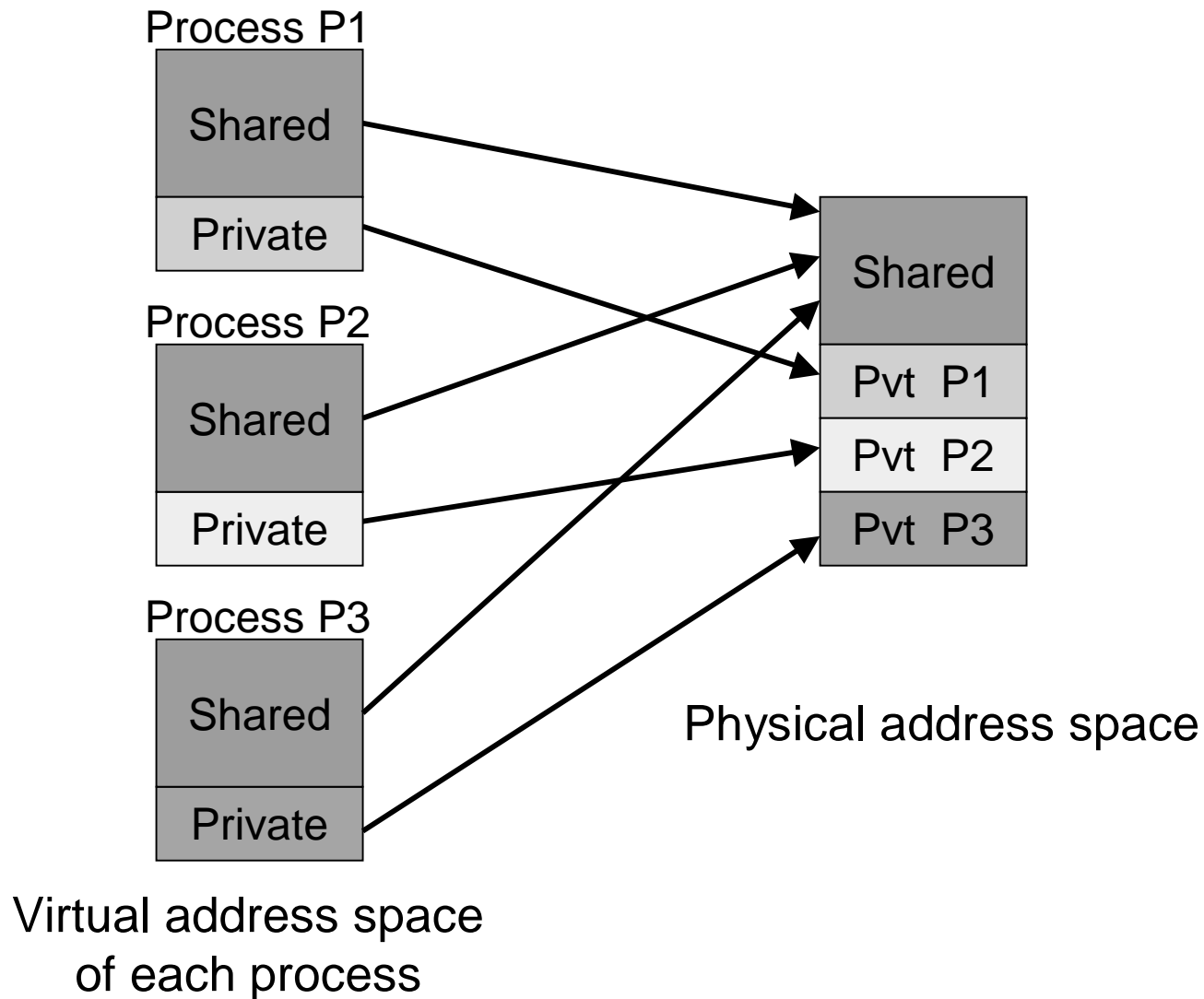
Communication Architecture

- At first, the hardware for shared memory and message passing were different – today, a single parallel machine can efficiently support different programming models

| CAD | Database | Scientific modeling | Parallel applications |

Multiprogramming   Shared   Message
                    address   passing   Programming models

Compilation or
library

OS support   User/System boundary

                    HW/SW boundary

Communication hardware
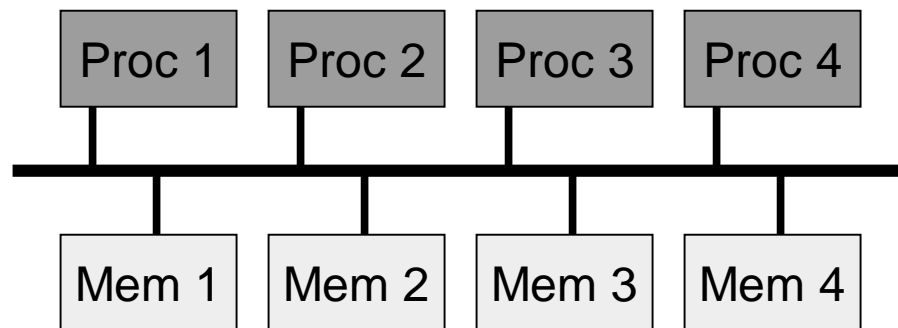
Shared Memory Architectures

- 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

Process P1

Shared
Private

Process P2

Shared
Private

Process P3

Shared
Private

Shared
Pvt P1
Pvt P2
Pvt P3

Physical address space

Virtual address space
of each process

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)

```
┌────────┐ ┌────────┐ ┌────────┐ ┌────────┐
│ Proc 1 │ │ Proc 2 │ │ Proc 3 │ │ Proc 4 │
└────────┘ └────────┘ └────────┘ └────────┘
    │          │          │          │
════╪══════════╪══════════╪══════════╪════
    │          │          │          │
┌───────┐ ┌───────┐ ┌───────┐ ┌───────┐
│ Mem 1 │ │ Mem 2 │ │ Mem 3 │ │ Mem 4 │
└───────┘ └───────┘ └───────┘ └───────┘
```

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

| Proc 1 | Mem 1 | Proc 2 | Mem 2 | Proc 3 | Mem 3 | Proc 4 | Mem 4 |
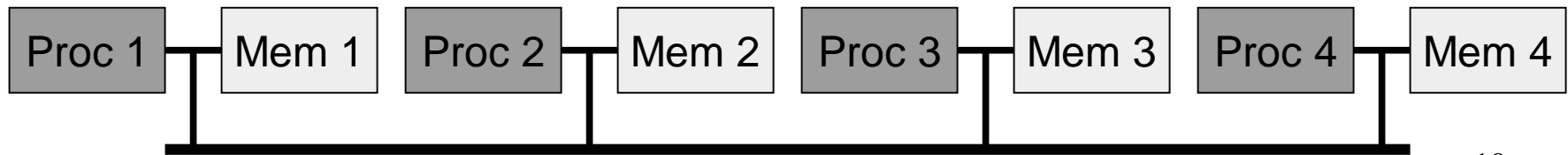
Message Passing

- 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

Title

- Bullet