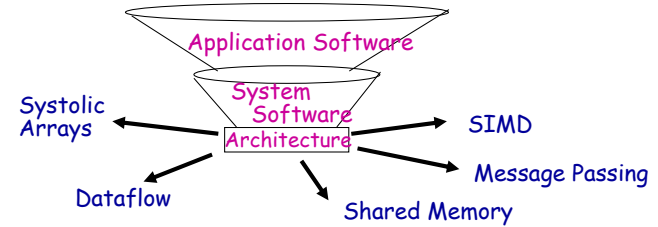


# Evolution and Convergence of Parallel Architectures

CS 418  
Lecture 2

## History

Historically, parallel architectures tied to programming models  
 • Divergent architectures, with no predictable pattern of growth.



*Uncertainty of direction paralyzed parallel software development!*

- 2 -

CS 418 S'04

## Today

Extension of "computer architecture" to support  
communication and cooperation

- OLD: **Instruction Set Architecture**
- NEW: **Communication Architecture**

### Defines

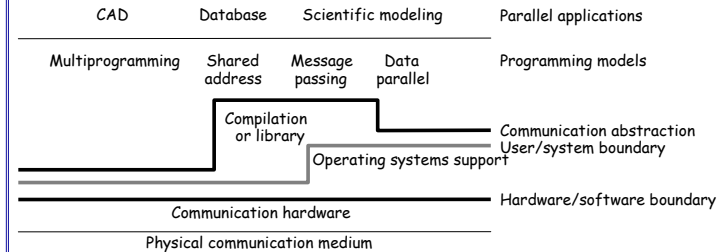
- Critical abstractions, boundaries, and primitives (interfaces)
- Organizational structures that implement interfaces (hw or sw)

Compilers, libraries and OS are important bridges  
today

- 3 -

CS 418 S'04

## Modern Layered Framework



- 4 -

CS 418 S'04

## Programming Model

What programmer uses in coding applications  
Specifies communication and synchronization

Examples:

- **Multiprogramming**: no communication or synch. at program level
- **Shared address space**: like bulletin board
- **Message passing**: like letters or phone calls, explicit point to point
- **Data parallel**: more regimented, global actions on data
  - Implemented with shared address space or message passing

- 5 -

CS 418 S'04

## Communication Abstraction

User level communication primitives provided

- Realizes the programming model
- Mapping exists between language primitives of programming model and these primitives

Supported directly by **hw**, or via **OS**, or via **user sw**

Lot of debate about what to support in sw and gap between layers

Today:

- Hw/sw interface tends to be flat, i.e. complexity roughly uniform
- Compilers and software play important roles as bridges today
- Technology trends exert strong influence

Result is convergence in organizational structure

- Relatively simple, general purpose communication primitives

- 6 -

CS 418 S'04

## Communication Architecture

= *User/System Interface* + *Implementation*

**User/System Interface:**

- Comm. primitives exposed to user-level by hw and system-level sw

**Implementation:**

- Organizational structures that implement the primitives: hw or OS
- How optimized are they? How integrated into processing node?
- Structure of network

**Goals:**

- Performance
- Broad applicability
- Programmability
- Scalability
- Low Cost

- 7 -

CS 418 S'04

## Evolution of Architectural Models

Historically, machines tailored to programming models

- Programming model, communication abstraction, and machine organization lumped together as the "architecture"

Evolution helps understand convergence

- Identify core concepts

**Most Common Models:**

- Shared Address Space, Message Passing, Data Parallel

**Other Models:**

- Dataflow, Systolic Arrays

Examine programming model, motivation, intended applications, and contributions to convergence

- 8 -

CS 418 S'04

## Shared Address Space Architectures

Any processor can **directly** reference any memory location

- Communication occurs implicitly as result of loads and stores

### Convenient:

- Location transparency
- Similar programming model to time-sharing on uniprocessors
  - Except processes run on different processors
  - Good throughput on multiprogrammed workloads

### Naturally provided on wide range of platforms

- History dates at least to precursors of mainframes in early 60s
- Wide range of scale: few to hundreds of processors

### Popularly known as *shared memory* machines or model

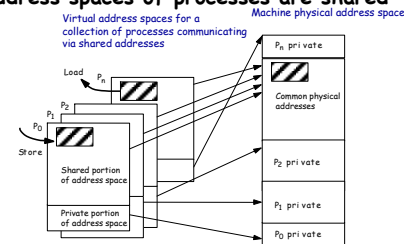
- Ambiguous: memory may be **physically distributed** among processors

- 9 -

CS 418 S'04

## Shared Address Space Model

Process: **virtual address space** plus one or more **threads of control**  
 Portions of address spaces of processes are shared



- Writes to shared address visible to other threads, processes
- **Natural extension of uniprocessor model:** **conventional memory operations for comm.;** **special atomic operations for synchronization**
- OS uses shared memory to coordinate processes

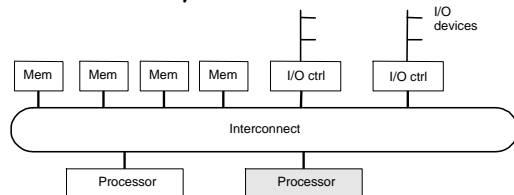
- 10 -

CS 418 S'04

## Communication Hardware

Also a natural extension of a uniprocessor

Already have processor, one or more memory modules and I/O controllers connected by hardware interconnect of some sort



Memory capacity increased by adding modules, I/O by controllers

- Add processors for processing!
- For higher-throughput multiprogramming, or parallel programs

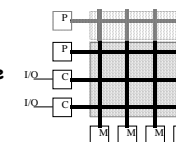
- 11 -

CS 418 S'04

## History

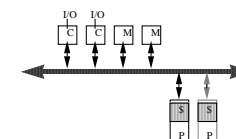
### "Mainframe" approach:

- Motivated by multiprogramming
- Extends crossbar used for mem bw and I/O
- Originally processor cost limited to small scale
  - later, cost of crossbar
- Bandwidth scales with  $p$
- **High incremental cost;** use multistage instead



### "Minicomputer" approach:

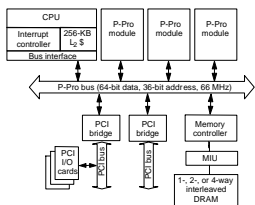
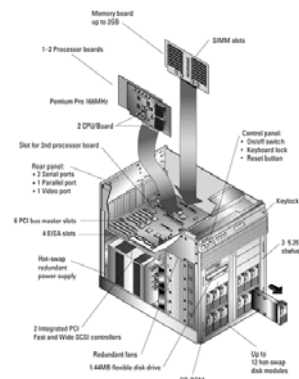
- Almost all microprocessor systems have bus
- Motivated by multiprogramming, TP
- Used heavily for parallel computing
- Called symmetric multiprocessor (SMP)
- Latency larger than for uniprocessor
- Bus is bandwidth bottleneck
  - **caching is key:** **coherence problem**
- Low incremental cost



- 12 -

CS 418 S'04

## Example: Intel Pentium Pro Quad

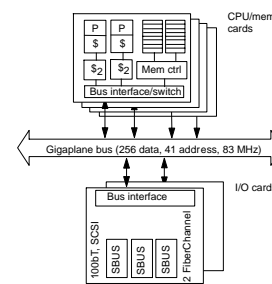


- All coherence and multiprocessing glue in processor module
- Highly integrated, targeted at high volume
- Low latency and bandwidth

- 13 -

CS 418 S'04

## Example: SUN Enterprise

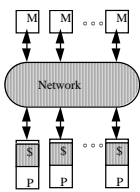


- 16 cards of either type: processors + memory, or I/O
- All memory accessed over bus, so symmetric
- Higher bandwidth, higher latency bus

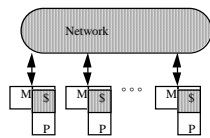
- 14 -

CS 418 S'04

## Scaling Up



"Dance hall"



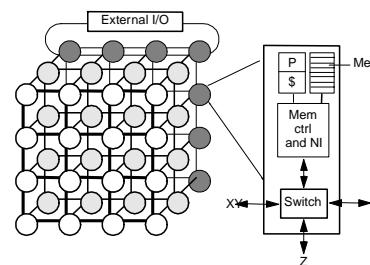
Distributed memory

- Problem is interconnect: cost (crossbar) or bandwidth (bus)
- Dance-hall: bandwidth still scalable, but lower cost than crossbar
  - latencies to memory uniform, but uniformly large
- Distributed memory or non-uniform memory access (NUMA)
  - Construct shared address space out of simple message transactions across a general-purpose network (e.g. read-request, read-response)
- Caching shared (particularly nonlocal) data?

- 15 -

CS 418 S'04

## Example: Cray T3E



- Scale up to 1024 processors, 480MB/s links
- Memory controller generates comm. request for nonlocal references
- No hardware mechanism for coherence (SGI Origin etc. provide this)

- 16 -

CS 418 S'04

## Message Passing Architectures

Complete computer as building block, including I/O

- Communication via explicit I/O operations

Programming model:

- directly access only **private address space** (local memory)
- communicate via explicit messages (**send/receive**)

High-level block diagram similar to distributed-mem SAS

- But comm. integrated at IO level, need not put into memory system
- Like networks of workstations (clusters), but tighter integration
- Easier to build than scalable SAS

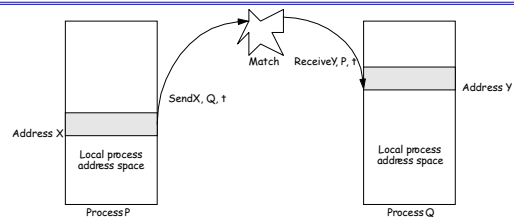
Programming model further from basic hardware ops

- Library or OS intervention

- 17 -

CS 418 S'04

## Message Passing Abstraction



- **Send** specifies buffer to be transmitted and receiving process
- **Recv** specifies sending process and application storage to receive into
- **Memory to memory copy**, but need to name processes
- **Optional tag** on send and matching rule on receive
- User process names local data and entities in process/tag space too
- In simplest form, the send/recv match achieves pairwise synch event
  - Other variants too
- **Many overheads: copying, buffer management, protection**

- 18 -

CS 418 S'04

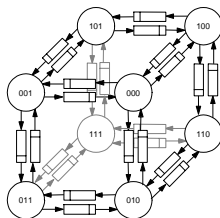
## Evolution of Message Passing

Early machines: FIFO on each link

- Hardware close to programming model
  - synchronous ops
- Replaced by DMA, enabling non-blocking ops
  - Buffered by system at destination until recv

Diminishing role of topology

- Store & forward routing: topology important
- Introduction of pipelined routing made it less so
- Cost is in node-network interface
- Simplifies programming



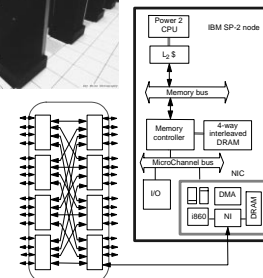
- 19 -

CS 418 S'04

## Example: IBM SP-2



General interconnection network formed from 8-port switches



- Made out of essentially complete RS6000 workstations
- Network interface integrated in I/O bus (bw limited by I/O bus)

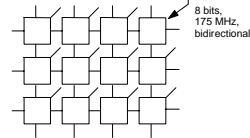
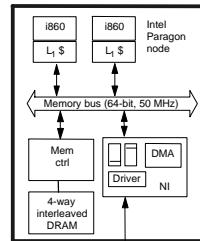
- 20 -

CS 418 S'04

## Example: Intel Paragon



Sandia's Intel Paragon XP/S-based Supercomputer



2D grid network with processing node attached to every switch

- 21 -

CS 418 S'04

## Toward Architectural Convergence

### Evolution and role of software have blurred boundary

- Send/recv supported on SAS machines via buffers
- Can construct global address space on MP using hashing
- Page-based (or finer-grained) shared virtual memory

### Hardware organization converging too

- Tighter NI integration even for MP (low-latency, high-bandwidth)
- At lower level, even hardware SAS passes hardware messages

### Even clusters of workstations/SMPs are parallel systems

- Emergence of fast system area networks (SAN)

### Programming models distinct, but organizations converging

- Nodes connected by general network and communication assists
- Implementations also converging, at least in high-end machines

- 22 -

CS 418 S'04

## Data Parallel Systems

### Programming model:

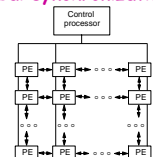
- Operations performed in parallel on each element of data structure
- Logically single thread of control, performs sequential or parallel steps
- Conceptually, a processor associated with each data element

### Architectural model:

- Array of many simple, cheap processors with little memory each
  - Processors don't sequence through instructions
- Attached to a control processor that issues instructions
- Specialized and general communication, cheap global synchronization

### Original motivation:

- Matches simple differential equation solvers
- Centralize high cost of instruction fetch & sequencing



- 23 -

CS 418 S'04

## Application of Data Parallelism

- Each PE contains an employee record with his/her salary
- ```

If salary > 100K then
    salary = salary * 1.05
else
    salary = salary * 1.10
    
```
- Logically, the whole operation is a single step
  - Some processors enabled for arithmetic operation, others disabled

### Other examples:

- Finite differences, linear algebra, ...
- Document searching, graphics, image processing, ...

### Some recent machines:

- Thinking Machines CM-1, CM-2 (and CM-5)
- Maspar MP-1 and MP-2,

- 24 -

CS 418 S'04

## Evolution and Convergence

### Rigid control structure (SIMD in Flynn taxonomy)

- **SISD** = uniprocessor, **MIMD** = multiprocessor

### Popular when cost savings of centralized sequencer high

- 60s when CPU was a cabinet; replaced by vectors in mid-70s
- Revived in mid-80s when 32-bit datapath slices just fit on chip
- No longer true with modern microprocessors

### Other reasons for demise

- Simple, regular applications have good locality, can do well anyway
- Loss of applicability due to hardwiring data parallelism
  - MIMD machines as effective for data parallelism and more general

### Programming model converges with SPMD (single program multiple data)

- Contributes need for fast global synchronization
- Structured global address space, implemented with either SAS or MP

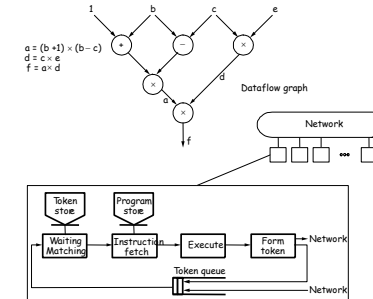
- 25 -

CS 418 S'04

## Dataflow Architectures

### Represent computation as a graph of essential dependences

- Logical processor at each node, activated by availability of operands
- Message (tokens) carrying tag of next instruction sent to next processor
- Tag compared with others in matching store; match fires execution



- 26 -

CS 418 S'04

## Evolution and Convergence

### Key characteristics:

- Ability to name operations, synchronization, dynamic scheduling

### Problems:

- Operations have locality across them, useful to group together
- Handling complex data structures like arrays
- Complexity of matching store and memory units
- Exposes too much parallelism (?)

### Converged to use conventional processors and memory

- Support for large, dynamic set of threads to map to processors
- Typically shared address space as well
- But separation of programming model from hardware (like data parallel)

### Lasting contributions:

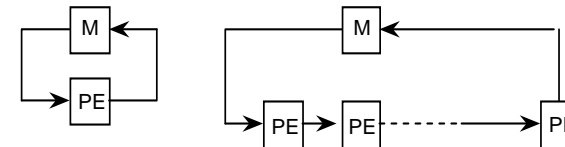
- Integration of communication with thread (handler) generation
- Tightly integrated communication and fine-grained synchronization
- Remained useful concept for software (compilers etc.)

- 27 -

CS 418 S'04

## Systolic Architectures

- Replace single processor with array of regular processing elements
- Orchestrate data flow for high throughput with less memory access



### Different from pipelining:

- Nonlinear array structure, multidirection data flow, each PE may have (small) local instruction and data memory

### Different from SIMD: each PE may do something different

### Initial motivation: VLSI enables inexpensive special-purpose chips

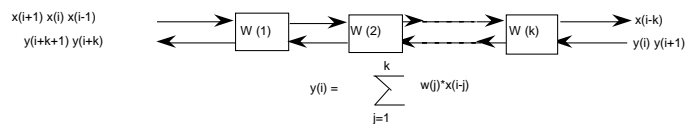
Represent algorithms directly by chips connected in regular pattern

- 28 -

CS 418 S'04

## Systolic Arrays (Cont)

### Example: Systolic array for 1-D convolution



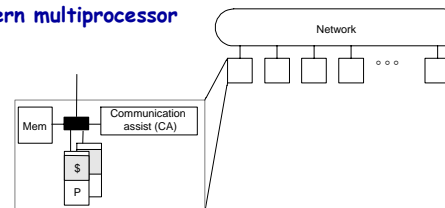
- **Practical realizations (e.g. iWARP) use quite general processors**
  - Enable variety of algorithms on same hardware
- **But dedicated interconnect channels**
  - Data transfer directly from register to register across channel
- **Specialized, and same problems as SIMD**
  - General purpose systems work well for same algorithms (locality etc.)

- 29 -

CS 418 S'04

## Convergence: General Parallel Architecture

### A generic modern multiprocessor



#### Node: processor(s), memory system, plus *communication assist*

- Network interface and communication controller
- Scalable network
- Convergence allows lots of innovation, now within framework
  - Integration of assist with node, what operations, how efficiently...

- 30 -

CS 418 S'04