Scalable Machine Designs

Many architects have decided not to provide direct support for shared memory, prefering instead to provide high performance basic operations, and leaving the programming facilities to the software

Scalability

Recall that the basic idea with parallel computing is that the more processors that can be applied to a program, the shorter should be it's execution time

- This idea ignores several critical points
  - The program may not have been written to take advantage of more parallelism
  - Because of overheads, more processors may result in worse performance
  - Other parts of the computer contribute to performance besides the processors

Nevertheless, bigger machines are seen to be better

Scaling Parallel Computers

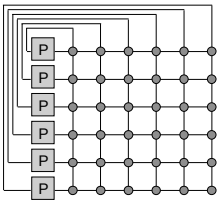
To produce a family of ever larger computers, it is necessary to increase various properties of the computer to maintain its balance ...

- Bandwidth -- the communication capacity between the processors
- Latency -- the time delay in communicating between processors
- Increment -- the amount by which the size of a parallel computer can be increased

Consider each property

Bandwidth Issues

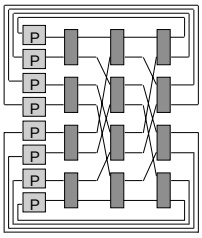
- Logically, as the next processor is added to a parallel computer the communication capacity should be increased to connect that processor to P other processors



Cross-bar switches scale as n^2, and so are applicable only for very small cases

Bandwidth Continued

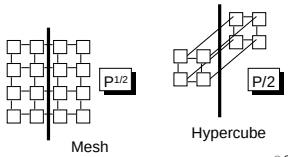
- Backing off from "complete connections" leads to approximations like the Omega network



The Omega network scales as *nlogn* but conflicts are possible since not all permutations are can be embedded in the network

Bisection Bandwidth

- The bisection bandwidth of a network is the capacity (b/s) across the wires of the "narrowest" bisector (separator dividing in 2)
- Motivation -- a random connection of processors must send about half the bits across the "middle" of the network



Bandwidth Across The Machine

- The network bandwidth is a significant component of communication capacity, but bandwidth at the Processor:Network interface, and elsewhere in the node are also important

Latency Issues

- Communication latency has several parts to it
  - Overhead of initiating and completing transfers
  - Number of network "hops"
  - Node latency, the time through a switch
  - Network contention
- Latency usually increases because of more hops and contention in larger machines
- Packets are pipelined, so a transfer of a msg of length m thru c-bit channel requires time
  $\alpha + \beta(m/c)$

Increment Issues

- For most scalable architectures it is not possible to add one more processor
  - The architecture does not grow in single units
  - Partially populated topologies can be hard to use
  - Systems components such as power supplies, racks, etc. grow by large units
- Typical growth rates are 2x since many architecture instances are powers of two

Message Passing

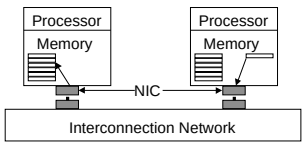
Message passing is a library-supported set of facilities allowing programmers to construct a parallel implementation

- Send/Receive messages between processors
- Spawn processes
- Synchronize

- Library routines are built on low level protocol
- Most machines come with native library, like Intel Paragon's NX Library
- Popular portable libraries: PVM, MPI

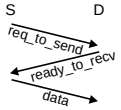
Interprocessor Communication

- The most basic communication facility is the transfer of a fixed length message
- It is essential that incoming messages be removed from the network to avoid "badness"



Message Size

- It is easy to provide space to store small messages, but what about 1MB messages?
- Use a handshaking protocol to set things up
  - Source sends destination: req_to_send(byteno)
  - Destination (al)locates the space
  - Destination replies: ready_to_receive
  - Source sends data



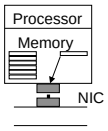
Details ... copying

A significant overhead can be repeated copying of messages as the NIC, operating system and library implementation manipulate process the message

- Copying serious only for large messages
- Past protocols have made as many as 3 copies
- For either the handshaking protocol or planned communication, arrange destination address in user space
- Have NIC drop data in the right place

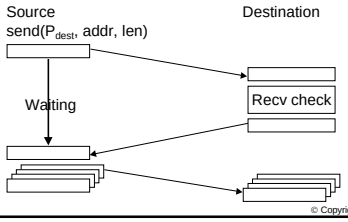
Details ... Asynchronous Interface

- In some machines the processor must implement the communication
- This is synchronous ... both source and destination processors must be processing the message at the same time -- inefficient
- Decouple using a NIC
  - Processor computes
  - Data managed by NIC
  - A NIC is potentially a very low latency interface



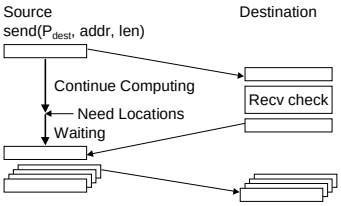
Higher Level Protocols

- When a processor sends, it must do so before it overwrites the data it is sending
- Since the library routines do not know when that is, they force the Source to wait til done



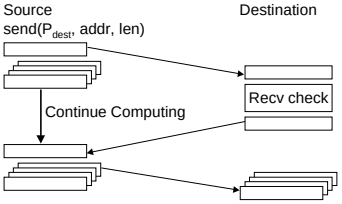
Higher Level Protocols, Continued

- Break coupling using an asynchonous scheme
  - Keep computing during handshake/Xmission
  - Need to have work, and enough memory



Higher Level Protocols, Continued

- Copy send solves the “need the location” problem, but at the cost of time/memory
- Though csend is simple, it is no longer popular



Active Messages

- Active message is a term for a restricted form of a remote procedure call
- Introduced in MIT’s J-machine
- Message structure

Dest Addr	Handler	Operand1, Operand2 ... Operand n

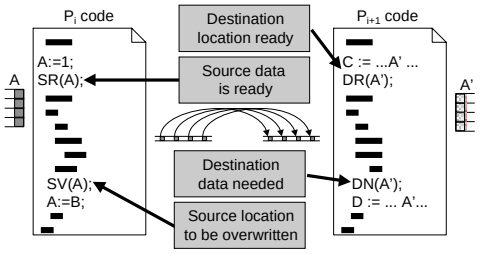
- When message arrives, processor interrupted, the handler is invoked to process the request
- Can be clumsy in bursts

Summary On Message Passing

- Message passing implements basic communication
- Though small messages can be sent without a handshaking protocol, larger messages need it
- Handshaking is a synchronization point between the communicating processors
- Asynchronous message passing is perhaps the most flexible, but it can be subtle

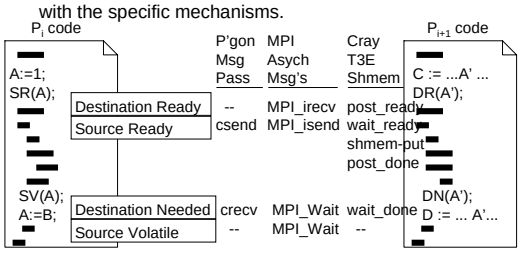
Ironman: Compiler Comm Interface

- Ironman says *what* is transferred and *when*, but not *how*
- Key idea: 4 calls demarcate the legal region of transfer



HW Customize: Binding Ironman Calls

With *what* and *when* specified by the 4 Ironman calls, the communication is implemented by linking in a library with the specific mechanisms.



	P'gon Msg Pass	MPI Asych Msg's	Cray T3E Shmem
Destination Ready	--	MPI_irecv	post_ready
Source Ready	csend	MPI_isend	wait_ready
			shmem-put
			post_done
Destination Needed	crecv	MPI_Wait	wait_done
Source Volatile	--	MPI_Wait	--

Ironman Summary ...

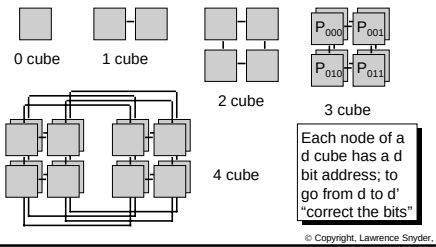
- Dumps message passing as compiler communication
- Replace w/ 4 calls saying what/when, but not how
  - `DR(), SR(), DN(), SV()`
- Strategy derives from CTA's abstract specification
  - No memory organization stated
- Bindings customize to hardware's mechanism
- Versatility covers commercial & prototype machines
  - message passing (all forms), shmem, shared, differential, ...
- Ironman concepts extended to other cases
  - Collective communication

Multiple Networks

- A single communication network is completely adequate, but the engineering effort needed for smooth operation can be substantial
- Dual network designs have been used often ...
  - The two networks may be only logical
  - One net for requests, one for responses assists in deadlock avoidance
  - One net for user comm and one net for system comm

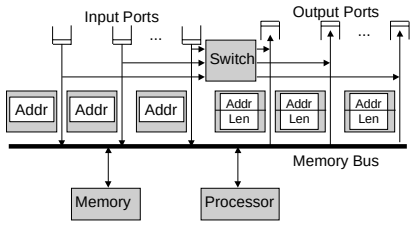
nCUBE/2: An Early MP Design

- The nCUBE/2 is a hypercube architecture
- Node communication channels grow as $\log_2 P$



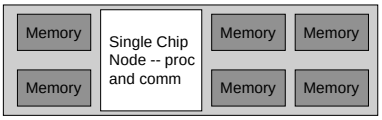
Schematic For nCUBE/2 Node

- Communication uses a simple DMA scheme



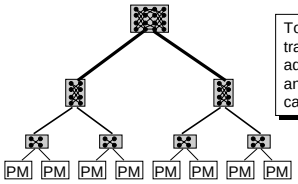
nCUBE Physical Implementation

A single chip performed of the operations allowing for a very economical node card



Connection Machine, CM-5

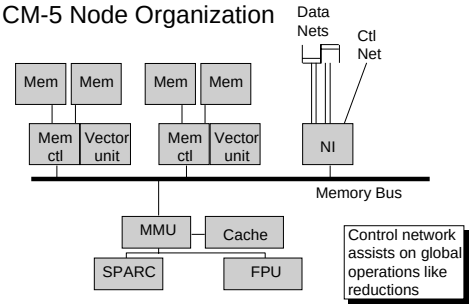
- A Fat Tree Interconnection Network



To handle increased traffic at higher levels add more channels, and greater switching capacity

CM-5 not a successor to CM-1 or CM-2

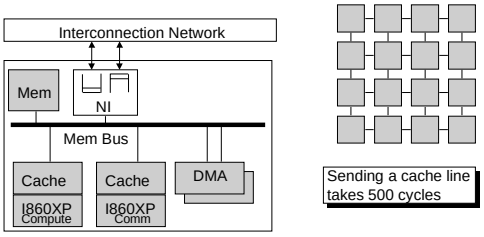
CM-5 Node Organization



Control network assists on global operations like reductions

Intel Paragon -- A 2D Mesh

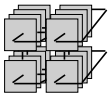
- The Paragon is based on a mesh
- Second processor serves comm co-processor



Sending a cache line takes 500 cycles

Cray T3D -- Shared Address Space

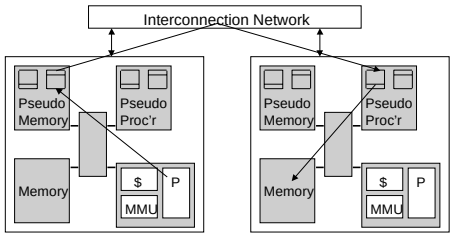
- True one sided communication is possible if a processor can get from or put to another processor's memory
- The Cray T3D implements these operations as "shmem get" and "shmem put"
- A (non coherent) shared address space



T3D is a 3 dimensional torus, i.e. mesh with wrap

Generic Shared Physical Address Space

- Conceptualize a pseudo-processor and a pseudo-memory



Cray T3D Shmem

- Shmem get/put eliminate a synchronization for the processor, though the pseudo-processor synchronizes
- Virtual-to-physical translation is performed at the "loading" end
- A transfer requires a short sequence of instructions, and then ~100 cycles for Xfer
- There is separate network support for global operations such as synchronization (eureka)

T3E

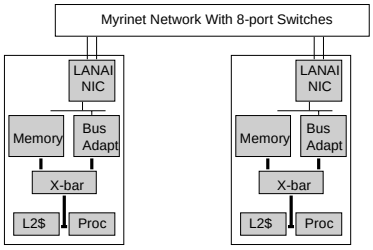
- Greater simplification over T3D through 512 64-bit E-registers used with load/store
  - Gets/Puts move data to/from global address to E-registers
  - Also, read/modify/write is possible through them
- Loading data
  - Put processor address portion if E-register
  - Issue get by mem-mapped store with addresses
  - Actual transfer is made from remote to E-register
  - Load from E-register gets data
- Twice the speed of T3D

Clusters -- Network of Workstations

- Processors (SMPs) and Memorys on an interconnect
  - Bus (E-net) or Ring (FDDI) and point-to-point (HPPI), Switched LAN (ATM), etc.
- Though cost effective in terms of hardware, the programming problem is at least as hard as any other solution
- Clusters are often used simply as servers rather than parallel machines

Standard NOW

- A Workstation + Myrinet is a standard design



Performance

- Difficult to generalize on performance
- Get/put achieves direct communication that can be superior to message passing and coherent shared memory
- The hardware is easily built ... "it's the software, stupid"