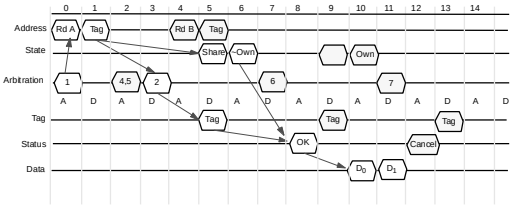
Scalability

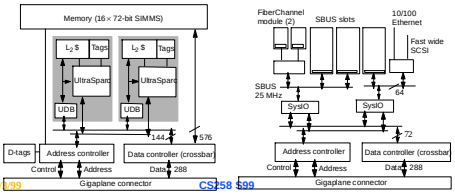
CS 258, Spring 99
David E. Culler
Computer Science Division
U.C. Berkeley

Recap: Gigaplane Bus Timing



Enterprise Processor and Memory System

- 2 procs per board, external L2 caches, 2 mem banks with x-bar
- Data lines buffered through UDB to drive internal 1.3 GB/s UPA bus
- Wide path to memory so full 64-byte line in 1 mem cycle (2 bus cyc)
- Addr controller adapts proc and bus protocols, does cache coherence
  - its tags keep a subset of states needed by bus (e.g. no M/E distinction)



Enterprise I/O System

- I/O board has same bus interface ASICs as processor boards
- But internal bus half as wide, and no memory path
- Only cache block sized transactions, like processing boards
  - Uniformity simplifies design
  - ASICs implement single-block cache, follows coherence protocol
- Two independent 64-bit, 25 MHz Sbuses
  - One for two dedicated FiberChannel modules connected to disk
  - One for Ethernet and fast wide SCSI
  - Can also support three SBUS interface cards for arbitrary peripherals
- Performance and cost of I/O scale with no. of I/O boards

Limited Scaling of a Bus

Characteristic	Bus
Physical Length	~ 1 ft
Number of Connections	fixed
Maximum Bandwidth	fixed
Interface to Comm. medium	memory inf
Global Order	arbitration
Protection	Virt -> physical
Trust	total
OS	single
comm. abstraction	HW

- Bus: each level of the system design is grounded in the scaling limits at the layers below and assumptions of close coupling between components

Workstations in a LAN?

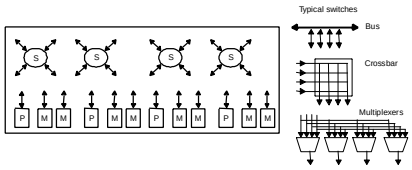
Characteristic	Bus	LAN
Physical Length	~ 1 ft	KM
Number of Connections	fixed	many
Maximum Bandwidth	fixed	???
Interface to Comm. medium	memory inf	peripheral
Global Order	arbitration	???
Protection	Virt -> physical	OS
Trust	total	none
OS	single	independent
comm. abstraction	HW	SW

- No clear limit to physical scaling, little trust, no global order, consensus difficult to achieve.
- Independent failure and restart

Scalable Machines

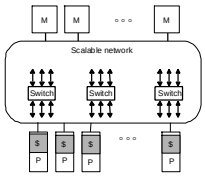
- What are the design trade-offs for the spectrum of machines between?
  - specialize or commodity nodes?
  - capability of node-to-network interface
  - supporting programming models?
- What does scalability mean?
  - avoids inherent design limits on resources
  - bandwidth increases with P
  - latency does not
  - cost increases slowly with P

Bandwidth Scalability



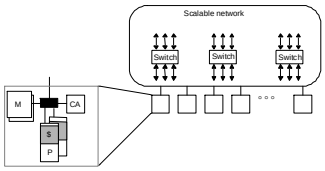
- What fundamentally limits bandwidth?
  - single set of wires
- Must have many independent wires
- Connect modules through switches
- Bus vs Network Switch?

Dancehall MP Organization



- Network bandwidth?
- Bandwidth demand?
  - independent processes?
  - communicating processes?
- Latency?

Generic Distributed Memory Org.



- Network bandwidth?
- Bandwidth demand?
  - independent processes?
  - communicating processes?
- Latency?

Key Property

- Large number of independent communication paths between nodes

=> allow a large number of concurrent transactions using different wires

- initiated independently
- no global arbitration
- effect of a transaction only visible to the nodes involved
  - effects propagated through additional transactions

Latency Scaling

- T(n) = Overhead + Channel Time + Routing Delay
- Overhead?
- Channel Time(n) = n/B --- BW at bottleneck
- RoutingDelay(h,n)

Typical example

- **max distance:** log n
- **number of switches:** α n log n
- overhead = 1 us, BW = 64 MB/s, 200 ns per hop
- Pipelined

$T_{64}(128)$ = 1.0 us + 2.0 us + 6 hops * 0.2 us/hop = 4.2 us
$T_{1024}(128)$ = 1.0 us + 2.0 us + 10 hops * 0.2 us/hop = 5.0 us

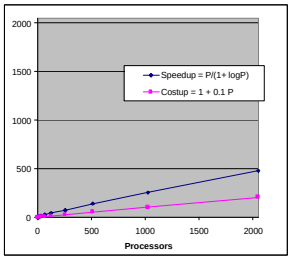
- Store and Forward

$T_{64}^{sf}(128)$ = 1.0 us + 6 hops * (2.0 + 0.2) us/hop = 14.2 us
$T_{64}^{sf}(1024)$ = 1.0 us + 10 hops * (2.0 + 0.2) us/hop = 23 us

Cost Scaling

- **cost(p,m) = fixed cost + incremental cost (p,m)**
- **Bus Based SMP?**
- **Ratio of processors : memory : network : I/O ?**
- **Parallel efficiency(p) = Speedup(P) / P**
- **Costup(p) = Cost(p) / Cost(1)**
- **Cost-effective: speedup(p) > costup(p)**
- **Is super-linear speedup**

Cost Effective?

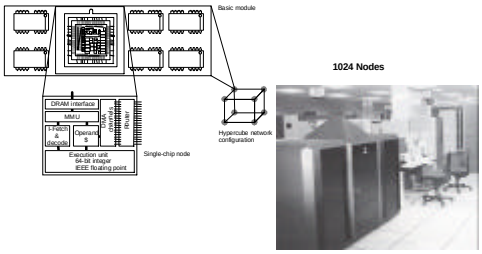


- **2048 processors: 475 fold speedup at 206x cost**

Physical Scaling

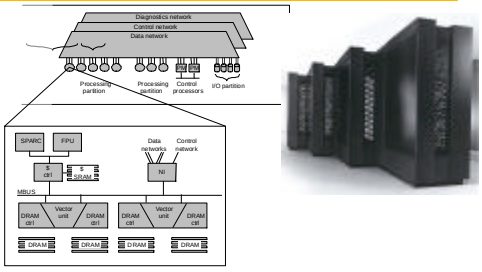
- **Chip-level integration**
- **Board-level**
- **System level**

nCUBE/2 Machine Organization

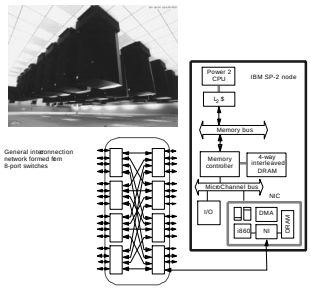


- **Entire machine synchronous at 40 MHz**

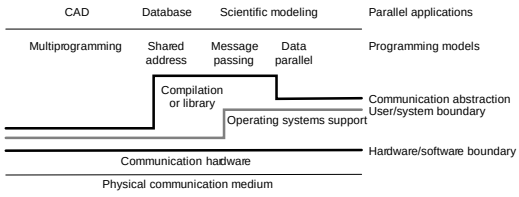
CM-5 Machine Organization



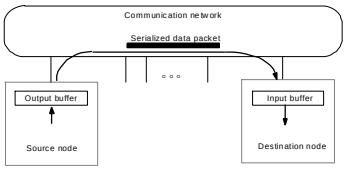
System Level Integration



Realizing Programming Models



Network Transaction Primitive



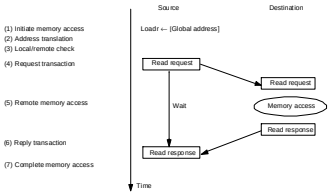
- **one-way transfer of information from a source output buffer to a dest. input buffer**
  - **causes some action at the destination**
  - **occurrence is not directly visible at source**
- **deposit data, state change, reply**

Bus Transactions vs Net Transactions

Issues:

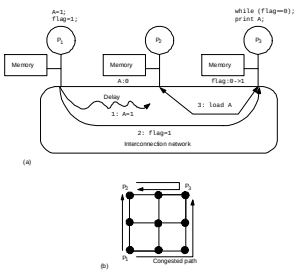
• **protection check**	V->P	??
• **format**	wires	flexible
• **output buffering**	reg, FIFO	??
• **media arbitration**	global	local
• **destination naming and routing**		
• **input buffering**	limited	many source
• **action**		
• **completion detection**		

Shared Address Space Abstraction

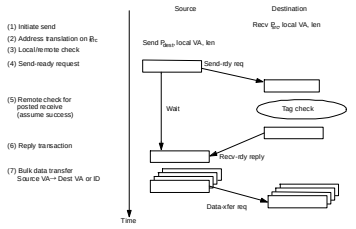


- **fixed format, request/response, simple action**

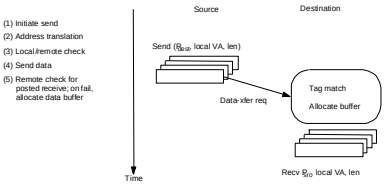
Consistency is challenging



Synchronous Message Passing

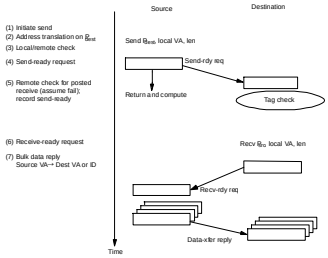


Asynch. Message Passing: Optimistic

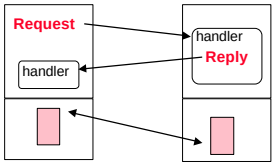


- **Storage???**

Asynch. Msg Passing: Conservative



Active Messages



- **User-level analog of network transaction**
- **Action is small user function**
- **Request/Reply**
- **May also perform memory-to-memory transfer**

Common Challenges

- **Input buffer overflow**
  - **N-1 queue over-commitment => must slow sources**
  - **reserve space per source (credit)**
    - » **when available for reuse?**
      - **Ack or Higher level**
  - **Refuse input when full**
    - » **backpressure in reliable network**
    - » **tree saturation**
    - » **deadlock free**
    - » **what happens to traffic not bound for congested dest?**
  - **Reserve ack back channel**
  - **drop packets**

Challenges (cont)

- **Fetch Deadlock**
  - **For network to remain deadlock free, nodes must continue accepting messages, even when cannot source them**
  - **what if incoming transaction is a request?**
    - » **Each may generate a response, which cannot be sent!**
    - » **What happens when internal buffering is full?**
- **logically independent request/reply networks**
  - **physical networks**
  - **virtual channels with separate input/output queues**
- **bound requests and reserve input buffer space**
  - **K(P-1) requests + K responses per node**
  - **service discipline to avoid fetch deadlock?**
- **NACK on input buffer full**
  - **NACK delivery?**

Summary

- **Scalability**
  - physical, bandwidth, latency and cost
  - level of integration
- **Realizing Programming Models**
  - network transactions
  - protocols
  - safety
    - » N-1
    - » fetch deadlock
- **Next: Communication Architecture Design Space**
  - how much hardware interpretation of the network transaction?