Scalable Distributed Memory Machines

Goal: ***Parallel machines that can be scaled to hundreds or thousands of processors.***

- **Design Choices:**
  - **Custom-designed or commodity nodes?**
  - **Network scalability.**
  - **Capability of node-to-network interface (*critical*).**
  - **Supporting programming models?**
- **What does hardware scalability mean?**
  - **Avoids inherent design limits on resources.**
  - **Bandwidth increases with machine size P.**
  - **Latency should not increase with machine size P.**
  - **Cost should increase slowly with P.**

MPPs Scalability Issues

- **Problems:**
  - **Memory-access latency.**
  - **Interprocess communication complexity or synchronization overhead.**
  - **Multi-cache inconsistency.**
  - **Message-passing and message processing overheads.**
- **Possible Solutions:**
  - **Fast dedicated, proprietary and scalable, networks and protocols.**
  - **Low-latency fast synchronization techniques possibly hardware-assisted .**
  - **Hardware-assisted message processing in communication assists (node-to-network interfaces).**
  - **Weaker memory consistency models.**
  - **Scalable directory-based cache coherence protocols.**
  - **Shared virtual memory.**
  - **Improved software portability; standard parallel and distributed operating system support.**
  - **Software latency-hiding techniques.**

One Extreme:

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

Poor Scalability

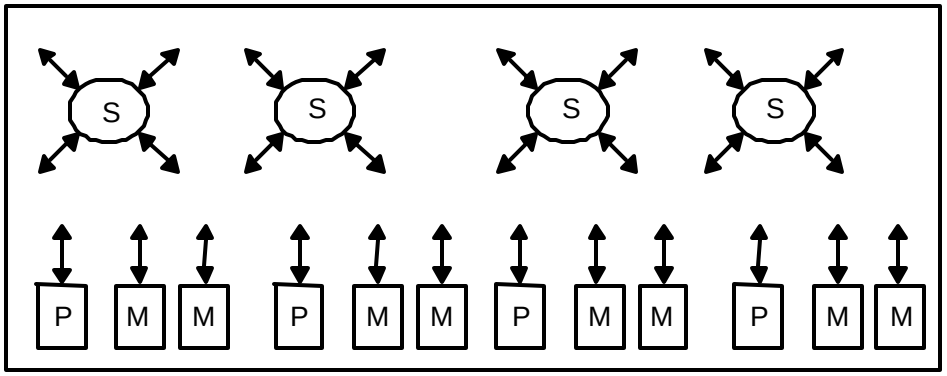
- **Bus: Each level of the system design is grounded in the scaling limits at the layers below and assumptions of close coupling between components.**

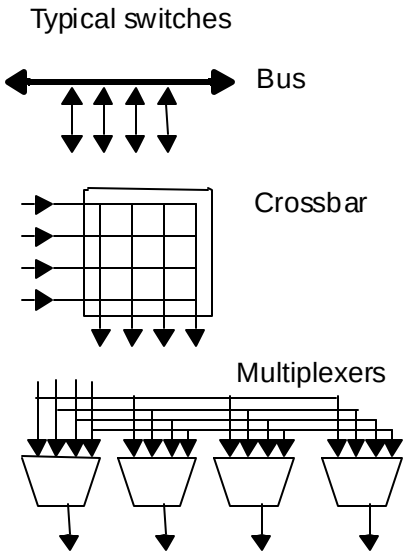
Another Extreme:
Scaling of Workstations in a LAN?

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, no global order, consensus difficult to achieve.**

Bandwidth Scalability

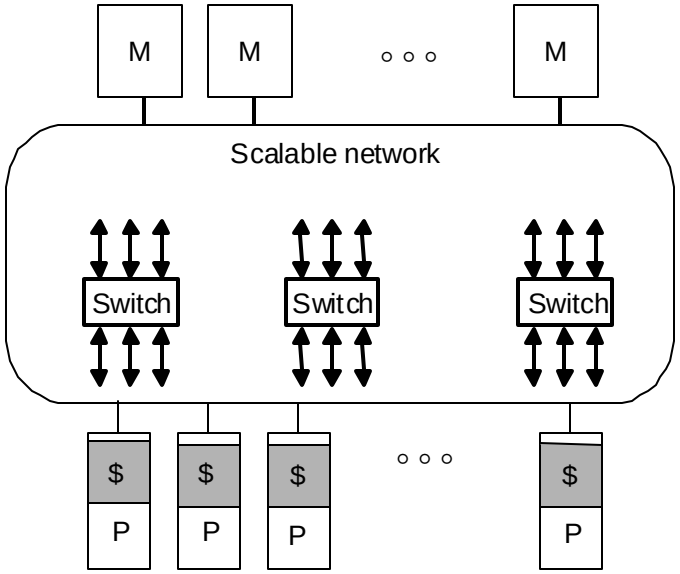




- **Depends largely on network characteristics:**
  - **Channel bandwidth.**
  - **Static: Topology: Node degree, Bisection width etc.**
  - **Multistage: Switch size and connection pattern properties.**
  - **Node-to-network interface capabilities.**

Dancehall MP Organization

- Network bandwidth?
- Bandwidth demand?
  - Independent processes?
  - Communicating processes?
- Latency?



Extremely high demands on network in terms of bandwidth, latency even for independent processes.

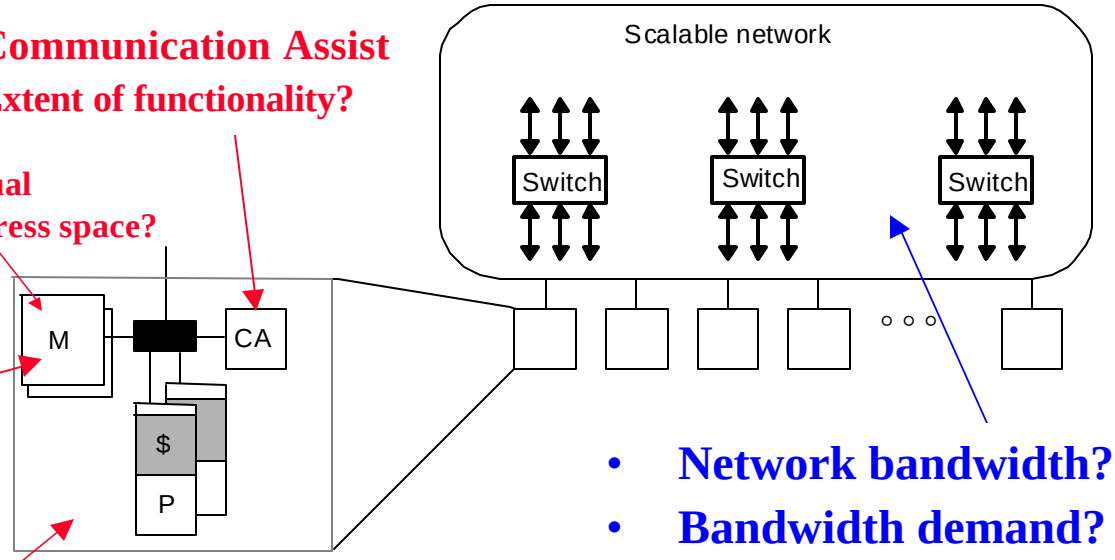
Generic Distributed Memory Organization

OS Supported?
Network protocols?

Communication Assist
Extent of functionality?

Global virtual Shared address space?

Message transaction DMA?



Node:
O(10) Bus-based SMP

Custom-designed CPU?
Node/System integration level?
How far? Cray-on-a-Chip?
SMP-on-a-Chip?

Multi-stage interconnection network (MIN)?
Custom-designed?

- **Network bandwidth?**
- **Bandwidth demand?**
  - **Independent processes?**
  - **Communicating processes?**
- **Latency? $O(\log_2 P)$ increase?**
- **Cost scalability of system?**

Key System Scaling Property

- **Large number of independent communication paths between nodes.**
  => Allow a large number of concurrent transactions using different channels.
- **Transactions are initiated independently.**
- **No global arbitration.**
- **Effect of a transaction only visible to the nodes involved**
  - **Effects propagated through additional transactions.**

Network Latency Scaling

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

Network Latency Scaling Example

$O(\log_2 n)$ Stage MIN using switches:

- **Max distance:** $\log_2 n$
- **Number of switches:** α n log n
- overhead = 1 us, BW = 64 MB/s, 200 ns per hop
- Using pipelined or cut-through routing:
- $T_{64}(128)$ = 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

Only 20% increase in latency for 16x size increase

- Store and Forward
- $T_{64}^{sf}(128)$ = 1.0 us + 6 hops * (2.0 + 0.2) us/hop = 14.2 us
- $T_{1024}^{sf}(128)$ = 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 ?

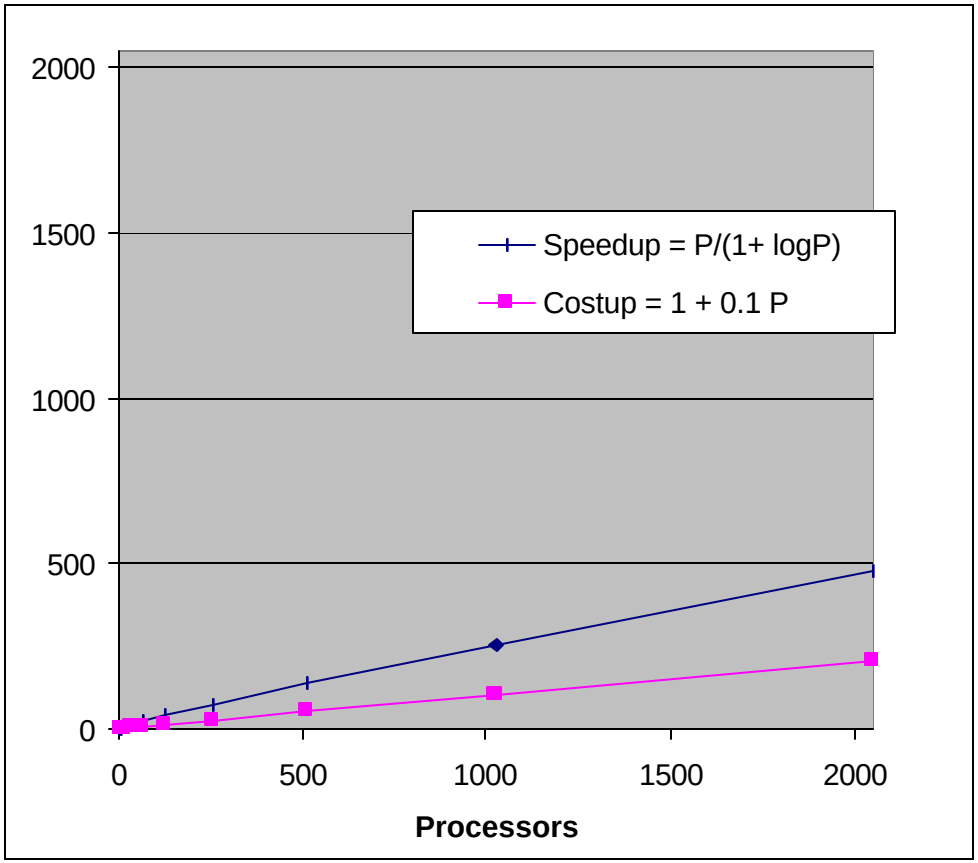
- Parallel efficiency(p) = Speedup(P) / P

- Similar to speedup, one can define:

  Costup(p) = Cost(p) / Cost(1)

- Cost-effective: speedup(p) > costup(p)

Cost Effective?



2048 processors: 475 fold speedup at 206x cost

Parallel Machine Network Examples

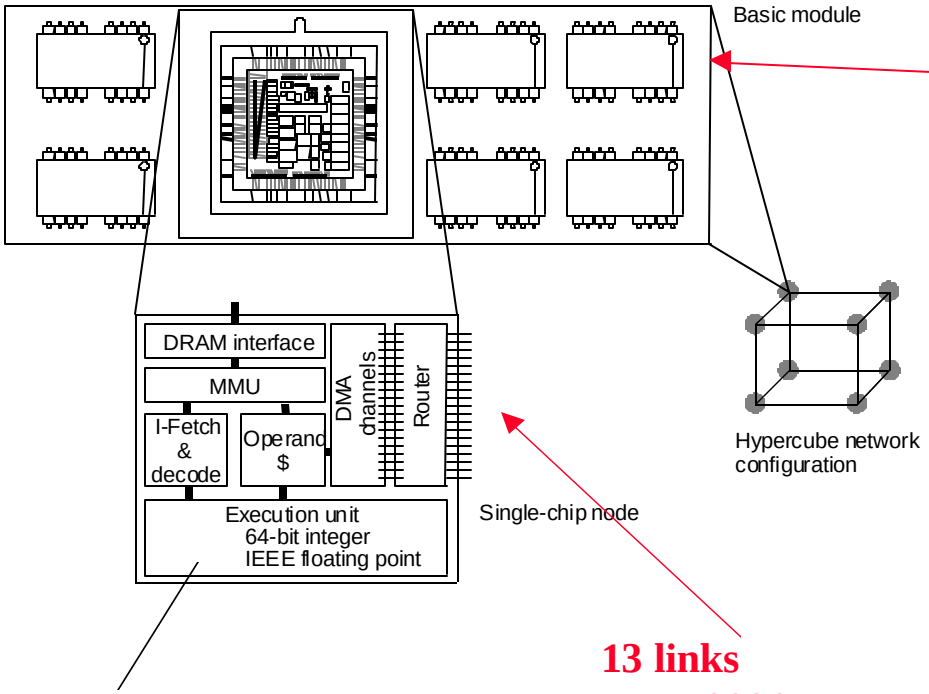
Machine	Topology	Cycle Time (ns)	Channel Width (bits)	Routing Delay (cycles)	Flit (data bits)
nCUBE/2	Hypercube	25	1	40	32
TMC CM-5	Fat-Tree	25	4	10	4
IBM SP-2	Banyan	25	8	5	16
Intel Paragon	2D Mesh	11.5	16	2	16
Meiko CS-2	Fat-Tree	20	8	7	8
CRAY T3D	3D Torus	6.67	16	2	16
DASH	Torus	30	16	2	16
J-Machine	3D Mesh	31	8	2	8
Monsoon	Butterfly	20	16	2	16
SGI Origin	Hypercube	2.5	20	16	160
Myricom	Arbitrary	6.25	16	50	16

Physical Scaling

- **Chip-level integration:**
  - **Integrate network interface, message router I/O links.**
    - **nCUBE/2, Alpha 21364, IBM Power 4**
  - **IRAM-style Cray-on-a-Chip: V-IRAM**
  - **Memory/Bus controller/chip set: Alpha 21364**
  - **SMP on a chip: Chip Multiprocessor (CMP): IBM Power 4**
- **Board-level:**
  - **Replicating using standard microprocessor cores.**
    - **CM-5 replicated the core of a Sun SparkStation 1 workstation.**
    - **Cray T3D and T3E replicated the core of a DEC Alpha workstation.**
- **System level:**
    - **IBM SP-2 uses 8-16 almost complete RS6000 workstations placed in racks.**

Chip-level integration Example:

nCUBE/2 Machine Organization



64 nodes socketed on a board

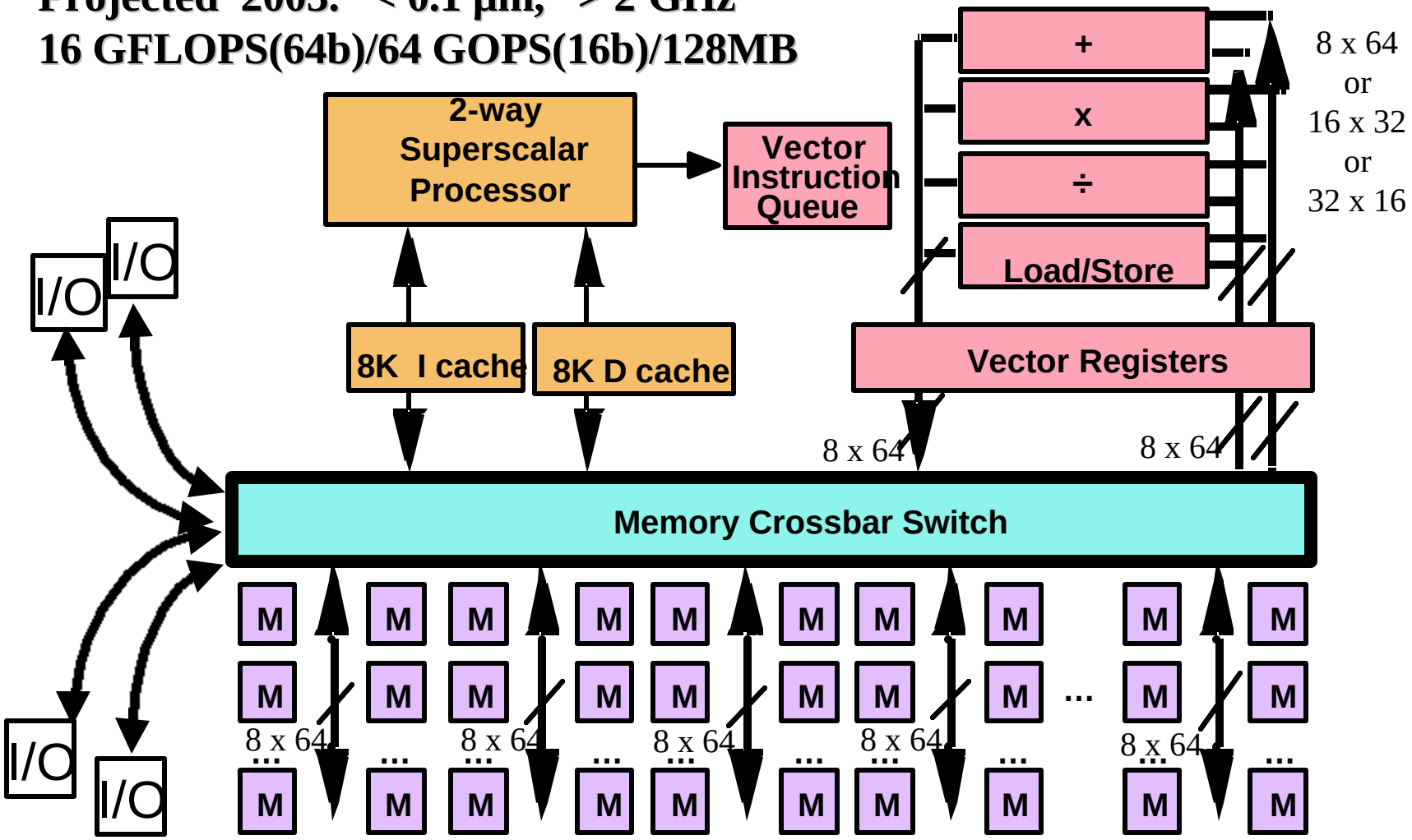
500, 000 transistors (considered large at the time)

13 links up to 8096 nodes possible

- Entire machine synchronous at 40 MHz

Chip-level integration Example:
Vector Intelligent RAM 2 (V-IRAM-2)

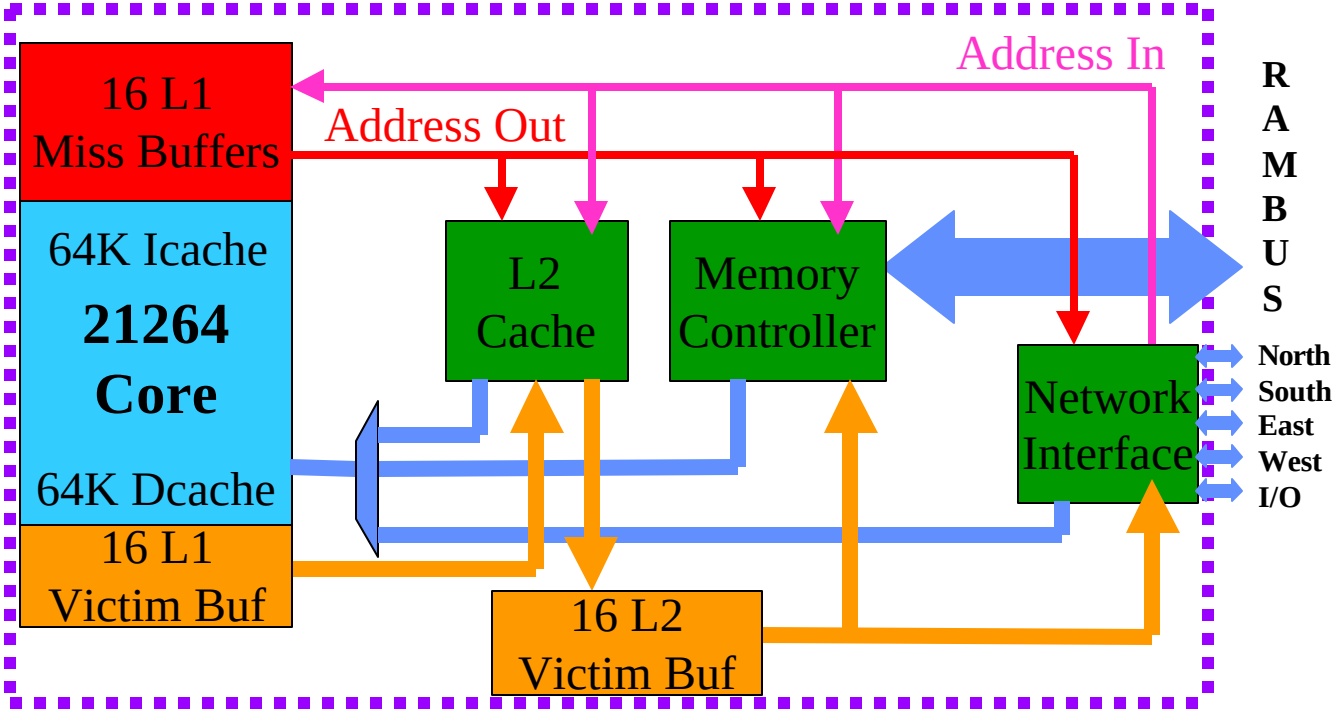
Projected 2003. < 0.1 µm, > 2 GHz
16 GFLOPS(64b)/64 GOPS(16b)/128MB



Chip-level integration Example:

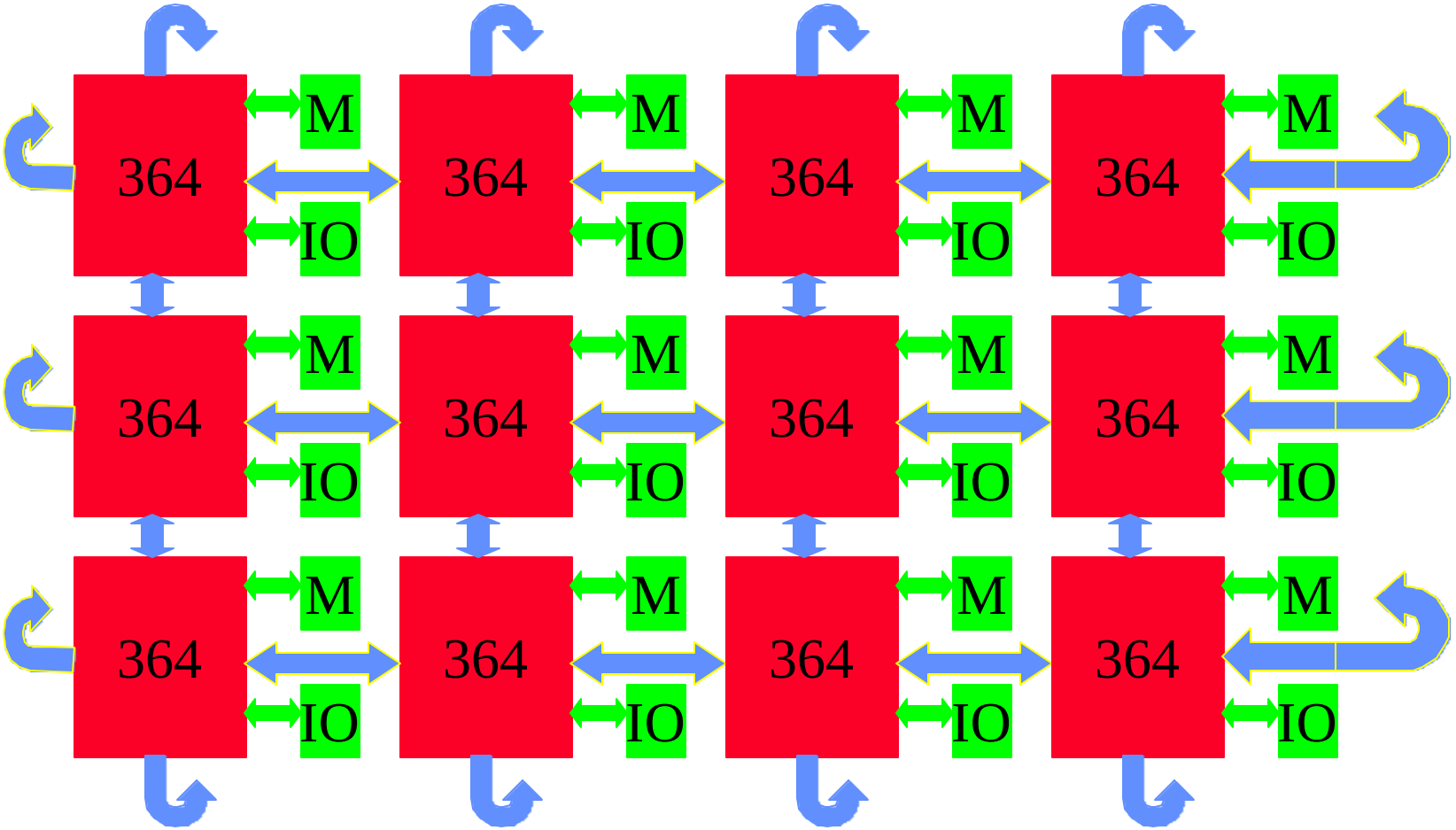
Alpha 21364

- Alpha 21264 core with enhancements
- Integrated Direct RAMbus memory controller:
  - 800 MHz operation, 30ns CAS latency pin to pin, 6 GB/sec read or write bandwidth
  - Directory based cache coherence
- Integrated network interface:
  - Direct processor-to-processor interconnect, 10 GB/second per processor
  - 15ns processor-to-processor latency, Out-of-order network with adaptive routing
  - Asynchronous clocking between processors, 3 GB/second I/O interface per processor



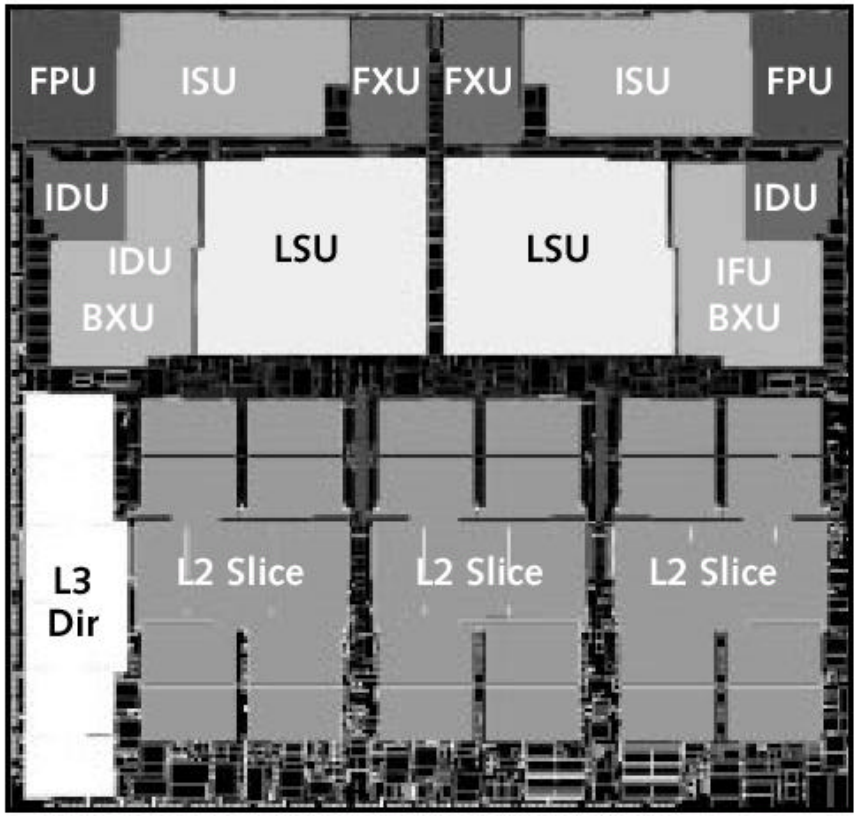
Chip-level integration Example:

A Possible Alpha 21364 System



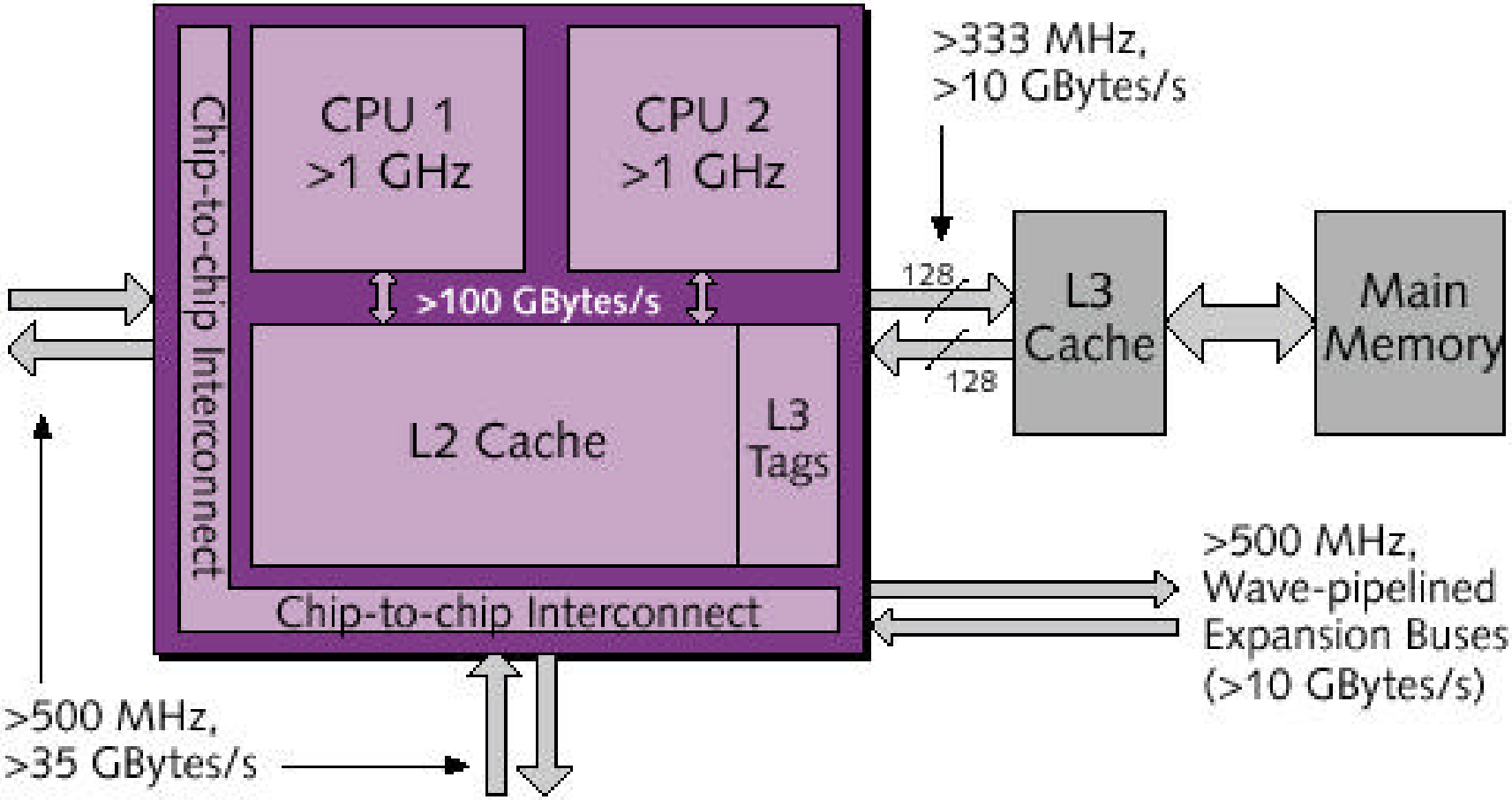
Chip-level integration Example:
IBM Power 4 CMP

- **Two tightly-integrated 1GHz CPU cores per 170 Million Transistor chip.**
- **128KB L_1 Cache per processor**
- **1.5 MB On-Chip Shared L_2 Cache**
- **External 32MB L_3 Cache: Tags kept on chip.**
- **35 Gbytes/s Chip-to-Chip interconnects.**



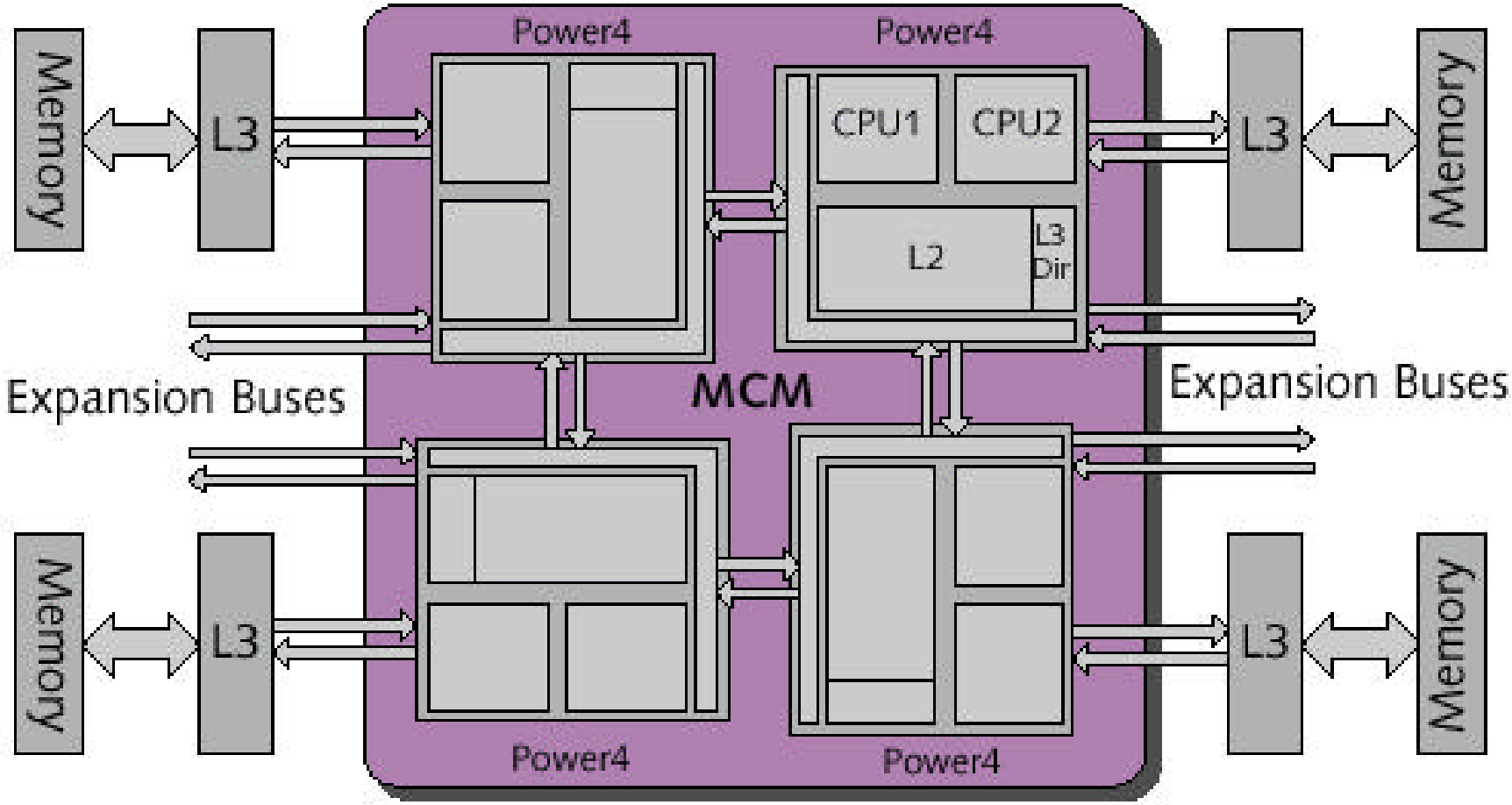
Chip-level integration Example:

IBM Power 4

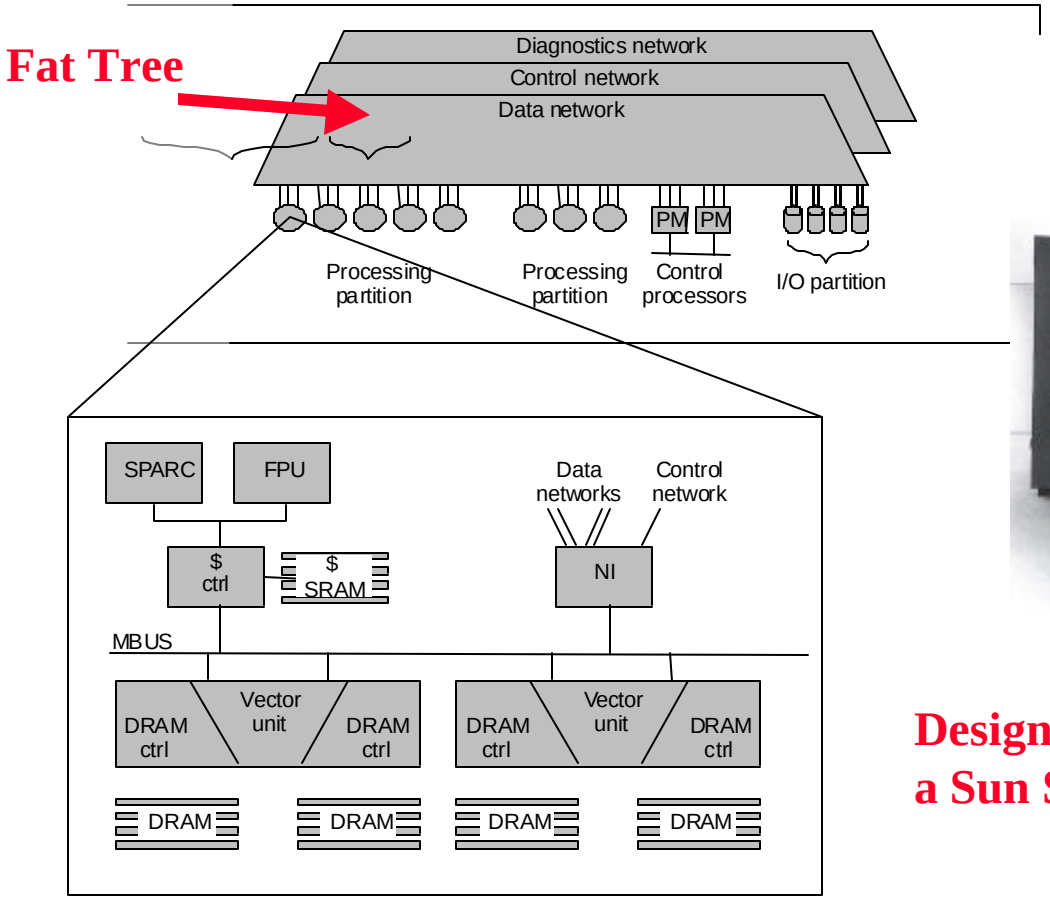


Chip-level integration Example:

IBM Power 4 MCM



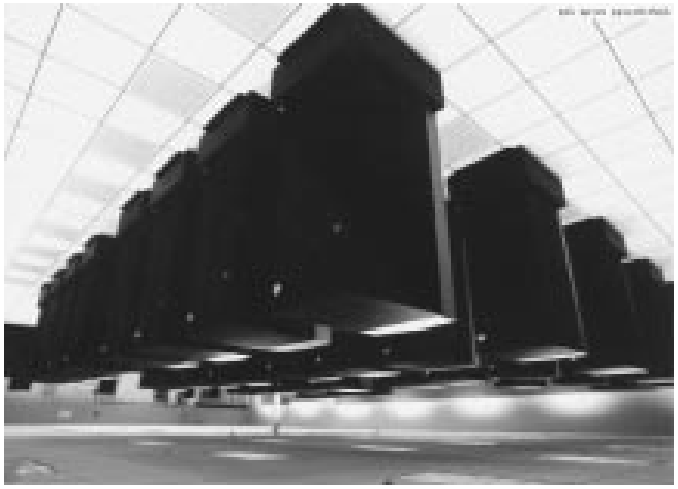
Board-level integration Example:
CM-5 Machine Organization



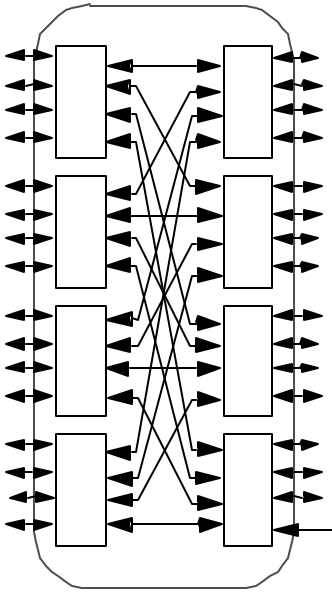
Design replicated the core of a Sun SparkStation 1 workstation

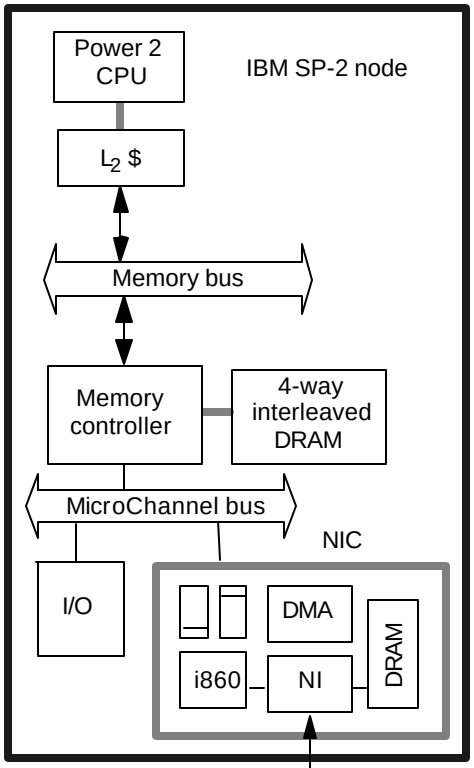
System Level Integration Example:

IBM SP-2



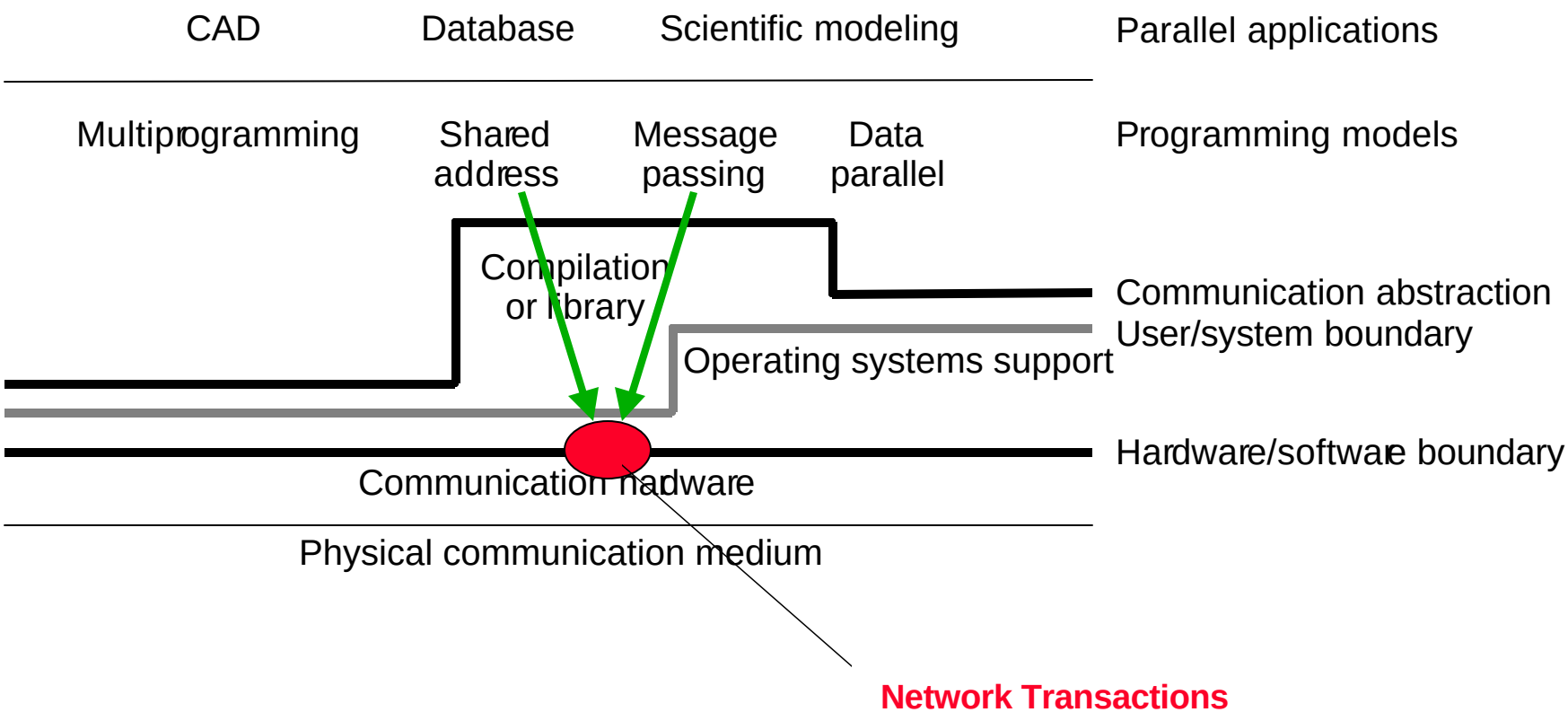
General interconnection network formed from 8-port switches





8-16 almost complete RS6000 workstations placed in racks.

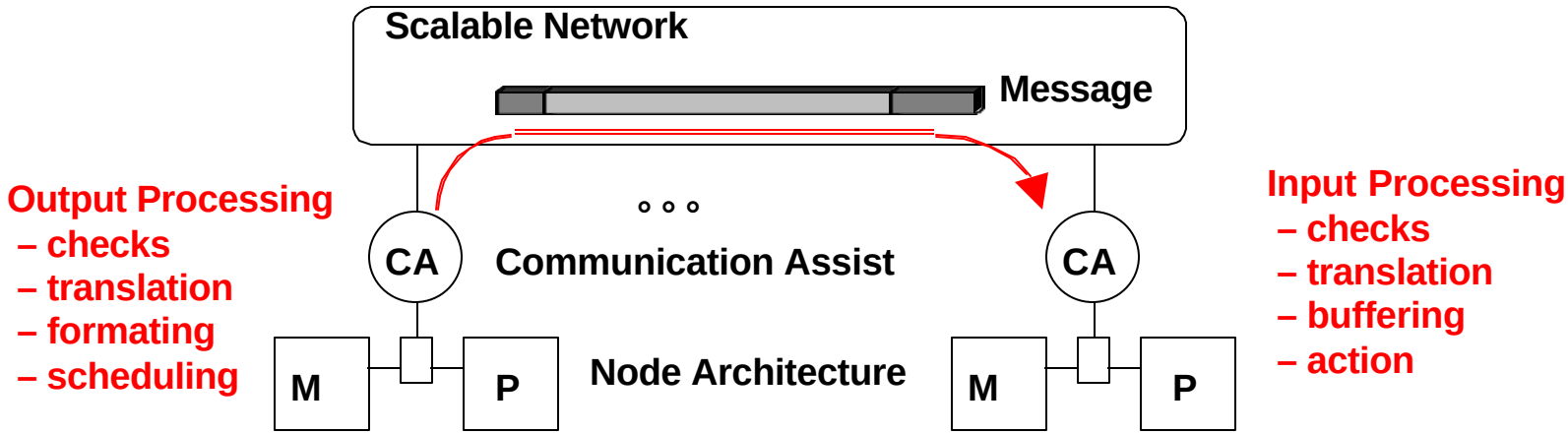
Realizing Programming Models:
Realized by Protocols



Challenges in Realizing Prog. Models in Large-Scale Machines

- No global knowledge, nor global control.
  - Barriers, scans, reduce, global-OR give fuzzy global state.
- Very large number of concurrent transactions.
- Management of input buffer resources:
  - Many sources can issue a request and over-commit destination before any see the effect.
- Latency is large enough that one is tempted to "take risks":
  - Optimistic protocols.
  - Large transfers.
  - Dynamic allocation.
- Many more degrees of freedom in design and engineering of these system.

Network Transaction Processing



CA = Communication Assist

Key Design Issue:

- **How much interpretation of the message by CA without involving the CPU?**
- **How much dedicated processing in the CA?**

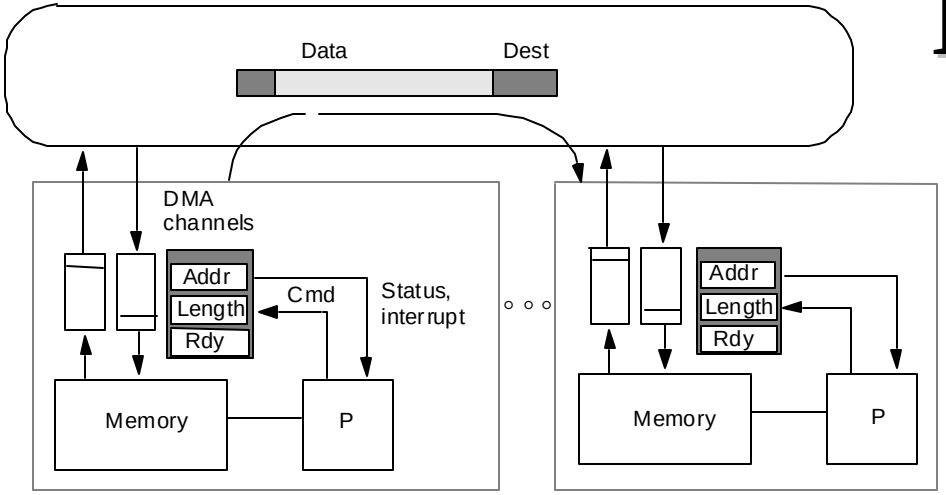
Spectrum of Designs

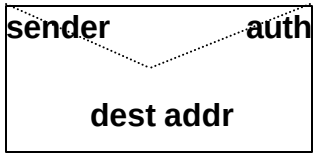
Increasing HW Support, Specialization, Intrusiveness, Performance (???)

None: Physical bit stream	
– blind, physical DMA	nCUBE, iPSC, . . .
User/System	
– User-level port	CM-5, *T
– User-level handler	J-Machine, Monsoon, . . .
Remote virtual address	
– Processing, translation	Paragon, Meiko CS-2
Global physical address	
– Proc + Memory controller	RP3, BBN, T3D
Cache-to-cache	
– Cache controller	Dash, KSR, Flash

No CA Net Transactions Interpretation:

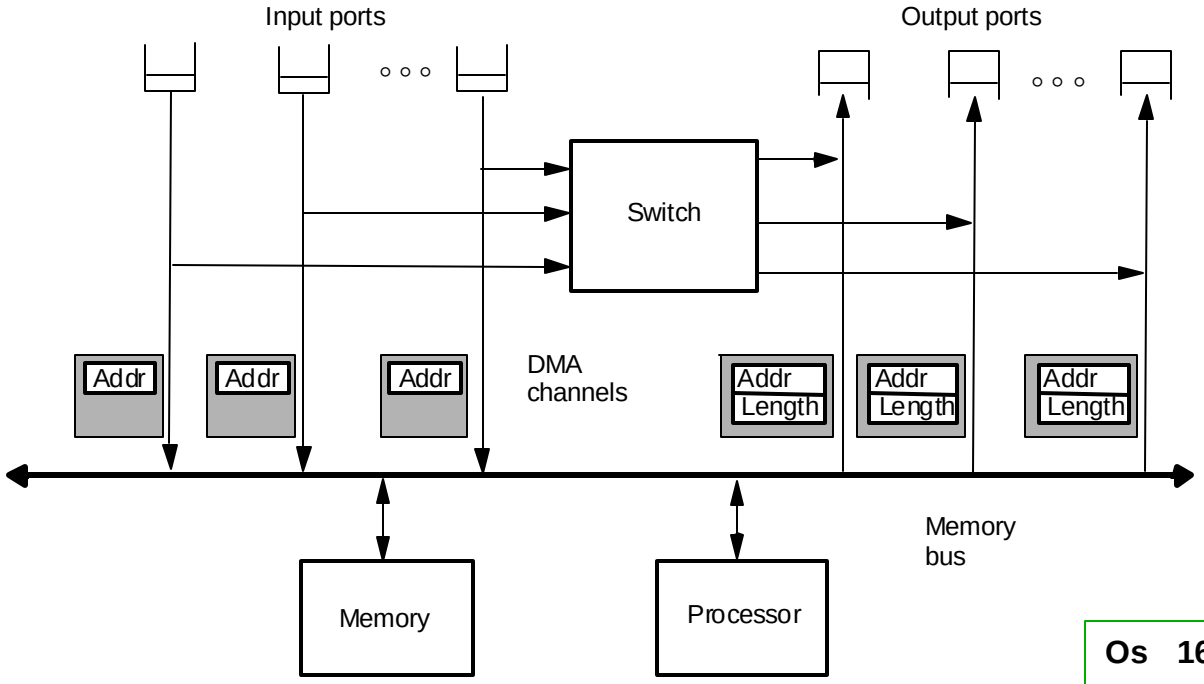
Physical DMA





- **DMA controlled by regs, generates interrupts.**
- **Physical => OS initiates transfers.**
- **Send-side:**
  - **Construct system "envelope" around user data in kernel area.**
- **Receive:**
  - **Must receive into system buffer, since no message interpretation in CA.**

nCUBE/2 Network Interface

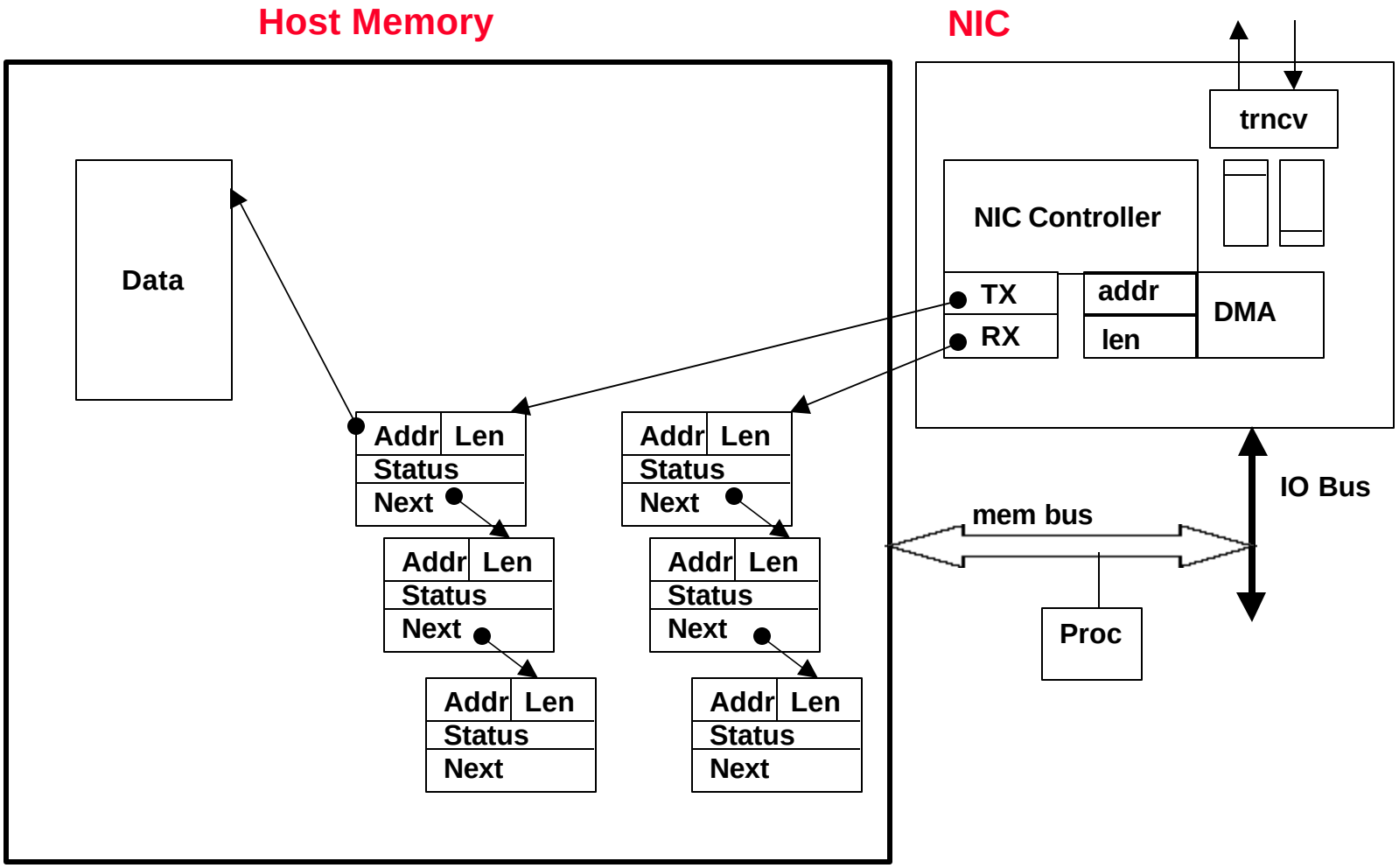


Os	16 ins	260 cy	13 us
Or	18	200 cy	15 us

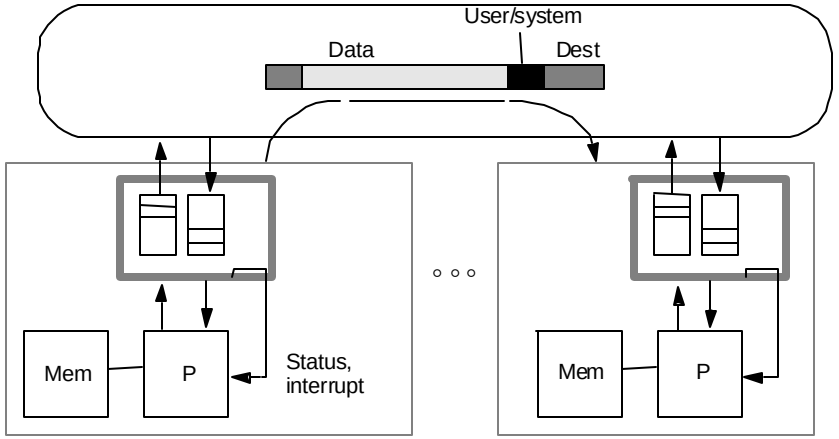
- includes interrupt

- **Independent DMA channel per link direction**
  - **Leave input buffers always open.**
  - **Segmented messages.**
- **Routing determines if message is intended for local or remote node**
  - **Dimension-order routing on hypercube.**
  - **Bit-serial with 36 bit cut-through.**

DMA In Conventional LAN Network Interfaces



User-Level Ports

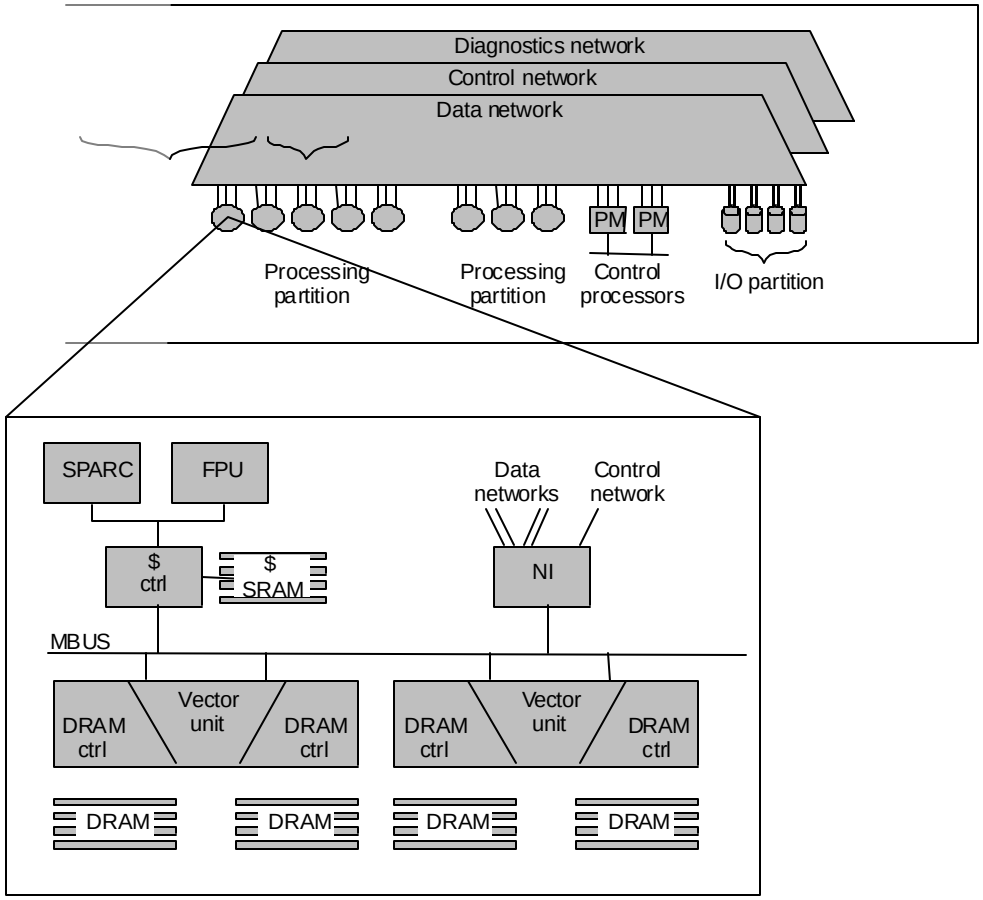


- **Initiate transaction at user level.**
- **CA interprets delivers message to user without OS intervention.**
- **Network port in user space.**
- **User/system flag in envelope.**
  - **Protection check, translation, routing, media access in source CA**
  - **User/sys check in destination CA, interrupt on system.**

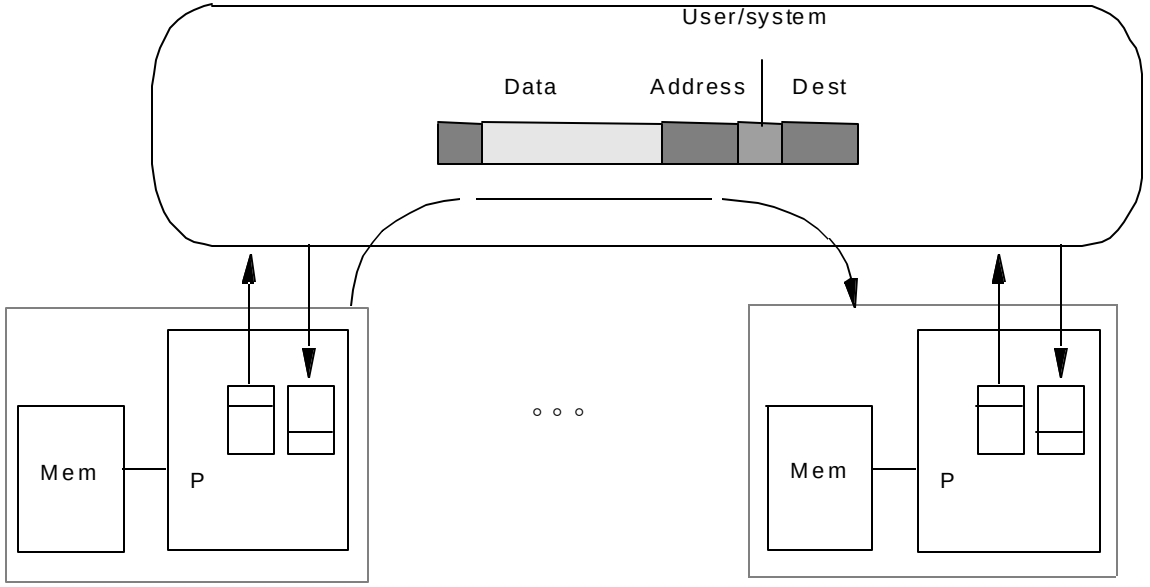
User-Level Network Example: CM-5

- Two data networks and one control network.
- Input and output FIFO for each network.
- Tag per message:
  - Index Network Inteface (NI) mapping table.
- *T integrated NI on chip.
- Also used in iWARP.

Os	50 cy	1.5 us
Or	53 cy	1.6 us
interrupt		10us

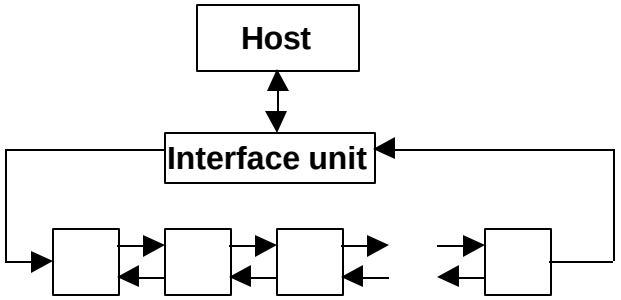


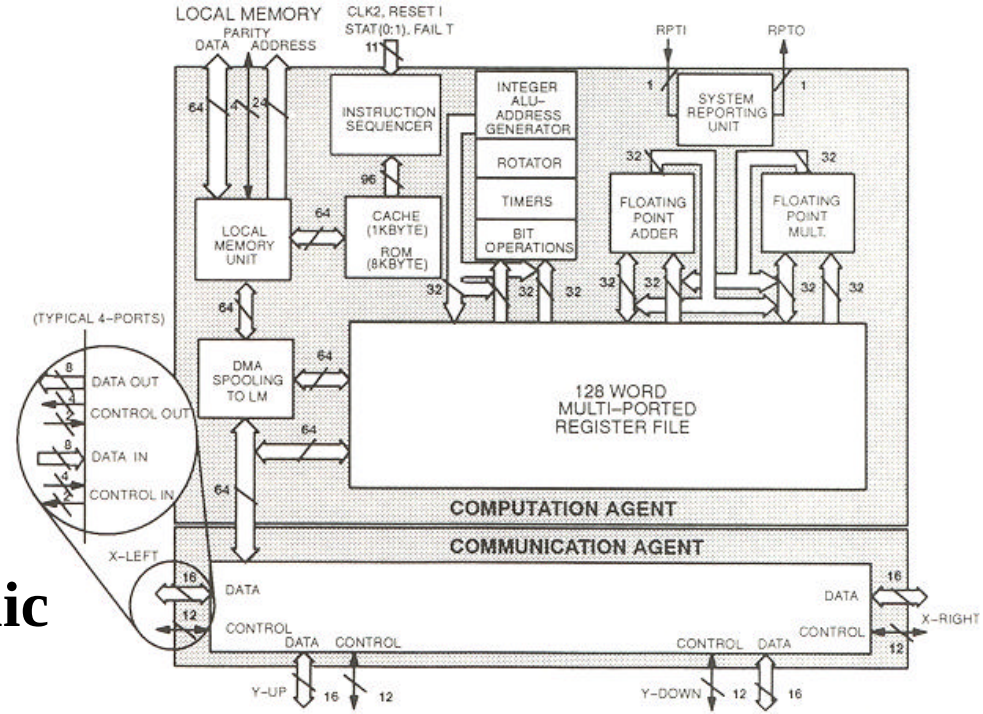
User-Level Handlers



- **Tighter integration of user-level network port with the processor at the register level.**
- **Hardware support to vector to address specified in message**
  - **message ports in registers.**

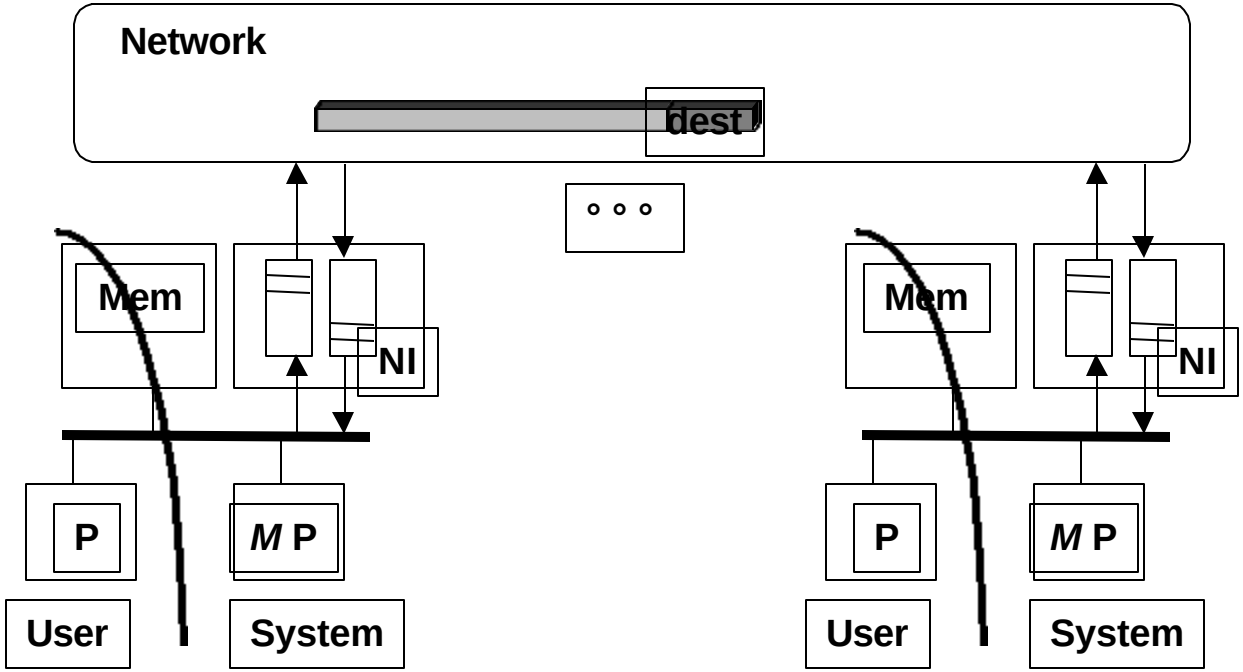
iWARP





- **Nodes integrate communication with computation on systolic basis.**
- **Message data direct to register.**
- **Stream into memory.**

Dedicated Message Processing Without Specialized Hardware Design



MP Message Processor

Node: Bus-based SMP

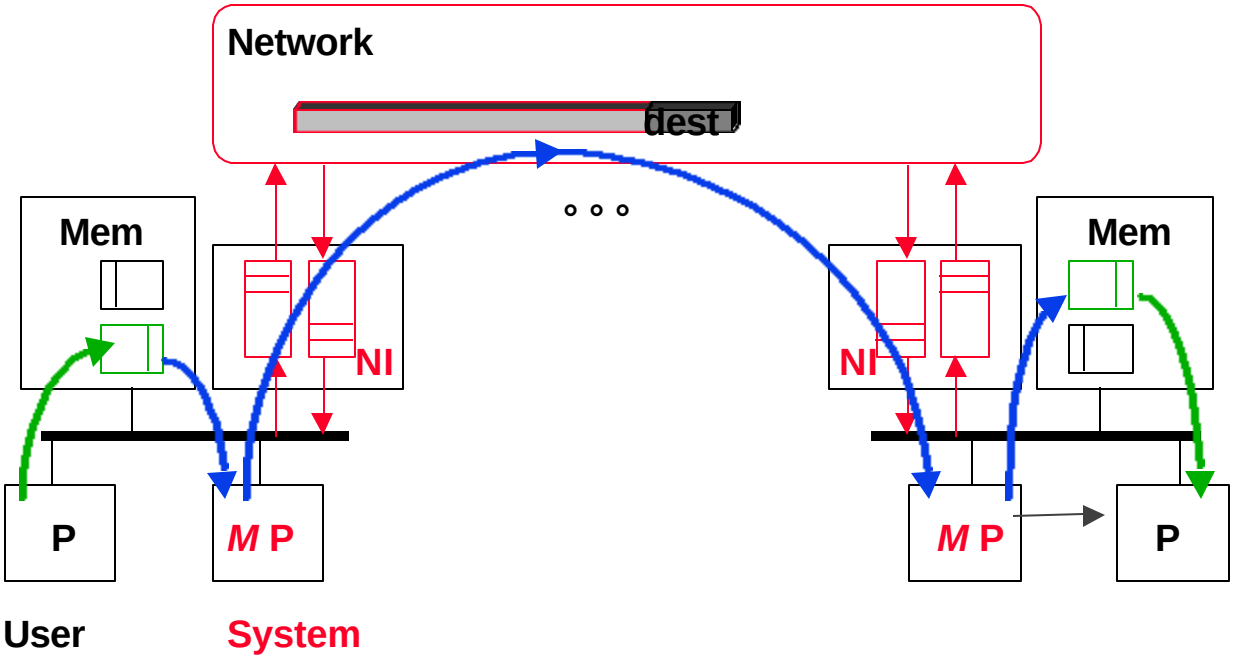
General Purpose processor performs arbitrary output processing (at system level)

General Purpose processor interprets incoming network transactions (at system level)

User Processor <–> Message Processor share memory

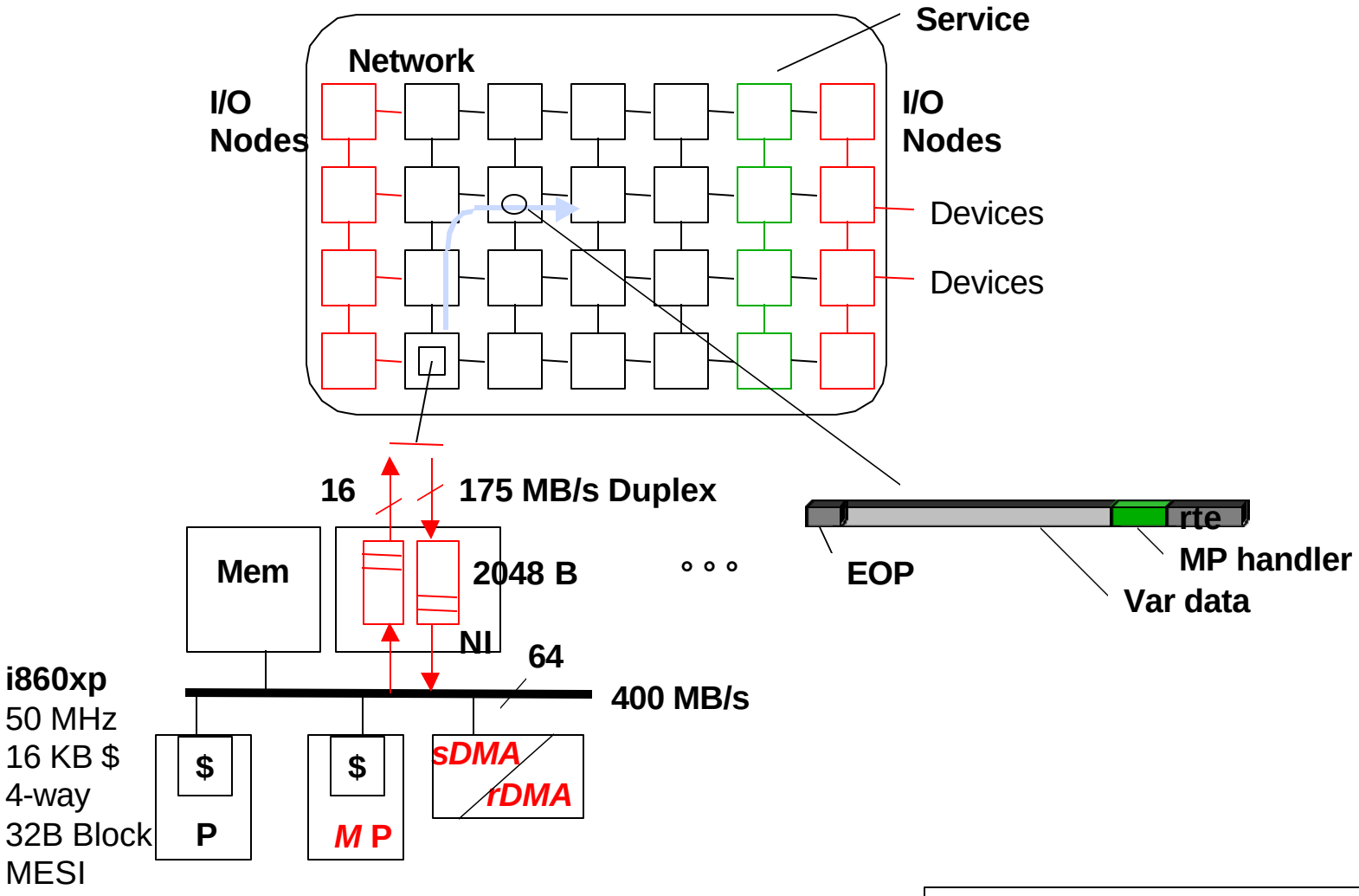
Message Processor <–> Message Processor via system network transaction

Levels of Network Transaction

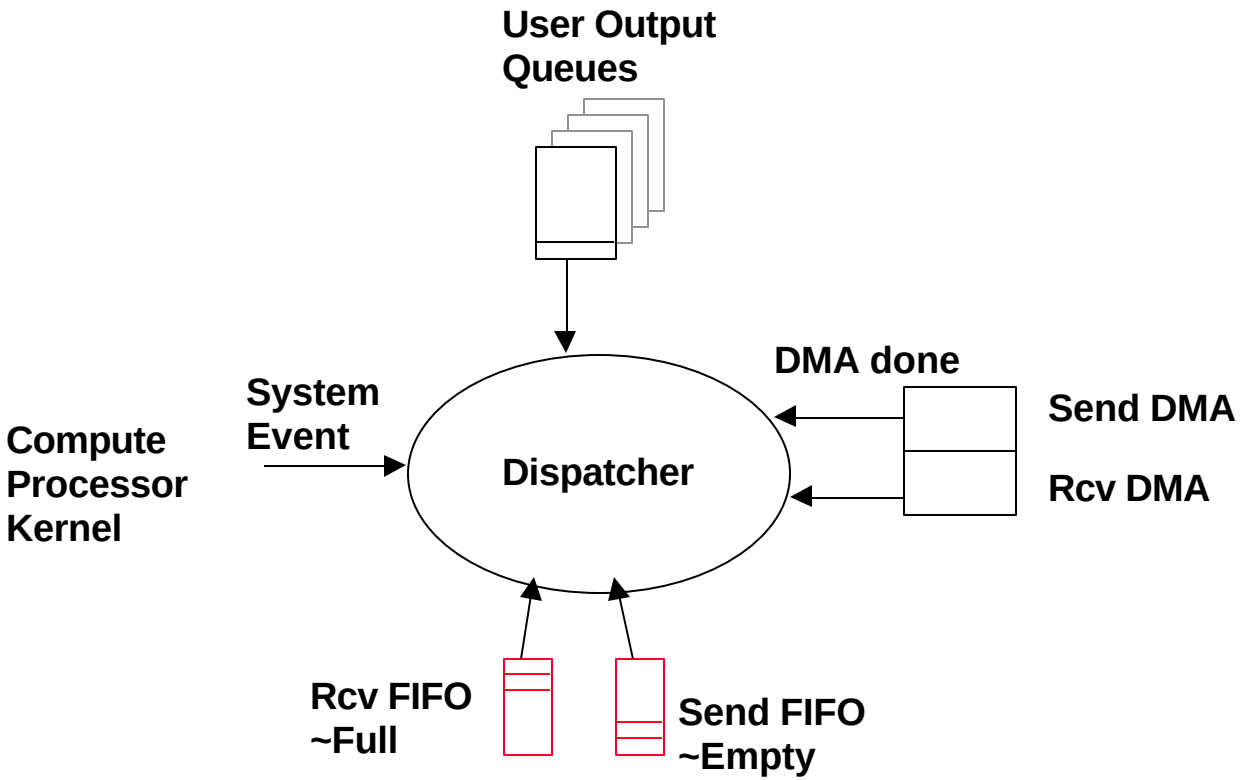


- User Processor stores cmd / msg / data into shared output queue.
  - Must still check for output queue full (or make elastic).
- Communication assists make transaction happen.
  - Checking, translation, scheduling, transport, interpretation.
- Effect observed on destination address space and/or events.
- Protocol divided between two layers.

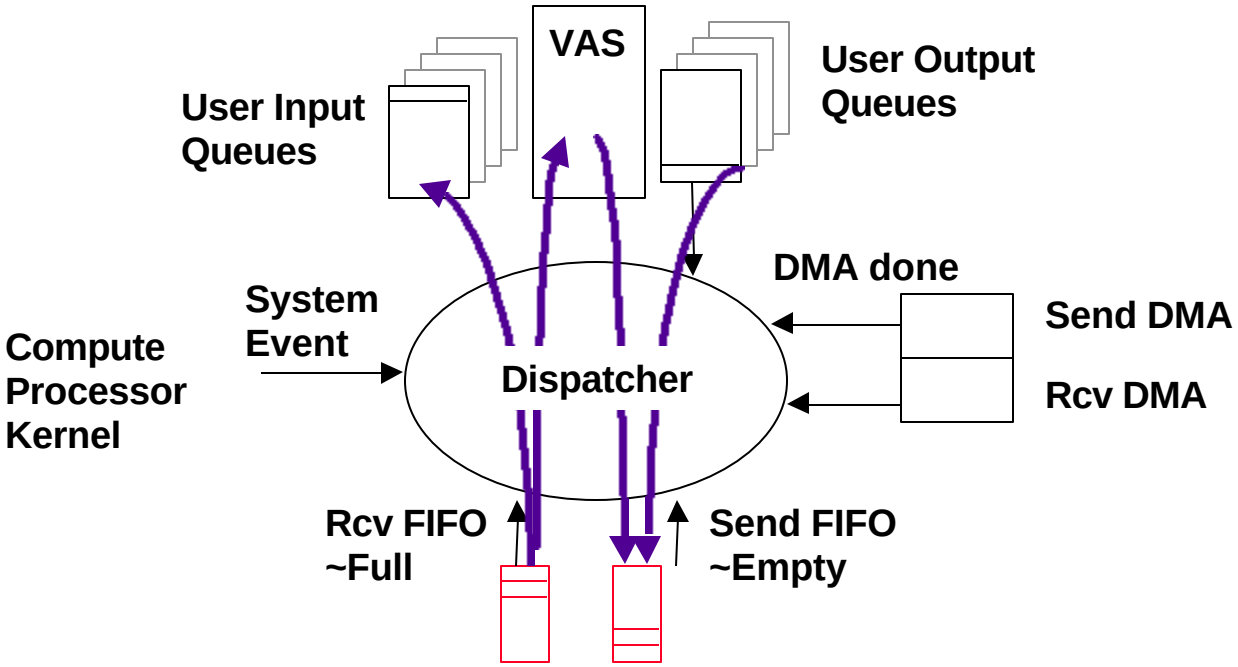
Example: Intel Paragon



Message Processor Events



Message Processor Assessment



- **Concurrency Intensive**
  - **Need to keep inbound flows moving while outbound flows stalled.**
  - **Large transfers segmented.**
- **Reduces overhead but adds latency.**