# Massively Parallel Processors

**Arvind**
**Laboratory for Computer Science**
**M.I.T.**



# Massively Parallel Processors

***Shared Memory***

***Cache-coherent***

- ~~KSR~~
- HP-Convex SPP 1200
- Sequent Symmetry 5000

circa 1995

***No global caches***

- Cray T-3D, T-3E

***Distributed Memory, Message passing***

- ~~TMC CM-5~~
- ~~Meiko CS-2~~
- ~~Intel Paragon~~
- IBM SP2
- nCube
- Pyramid
- Fujitsu VPP500, AP3000
- ~~C-DAC Param~~

**Application:*“Grand challenge” problems ?***

# Why MPPs have remained a fringe phenomenon ?

- driven by Grand Challenge problems and not by market forces
- too expensive in terms of both absolute cost and cost-performance
- MPP software is of little use in non-MPP environment

$\Rightarrow$ *few independent software developers*

MPPs = Massively Parallel Processors



# What is needed for Parallel Computing to become Ubiquitous

- Cost-effective Parallel Hardware
- Multiprocessing-capable standard Operating Systems
- Parallel programming models
- Scalability

**+**

*seamless transition from sequential to parallel computing*

# SMP's: Main Stream Parallel Computing

Small (< 4 processors) PC-class SMP's

- cheap (<< $10K)
- run NT, Solaris, Linux
- will track technology

⇒

- *sold in large numbers (100,000)*
- *incentive for ISVs to produce parallel software*
  compilers, debuggers, performance tools
  applications, libraries, ...



# Scalability Issue

Grand Challenge problems require "Jumbos" $

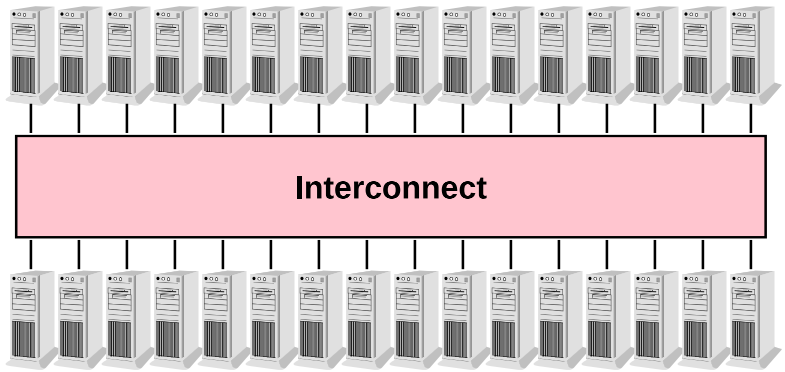
Enterprise computing requires scalable SMP's for growth $$$$

*Unfortunately, SMPs don't scale well and don't provide an incremental upgrade path*

*Large SMP's (>4 processors) have a superior memory system than small SMP's but*
64-Processor SMP >> 16 x 4-Processor SMP's

*Solution* ⇒ *Clusters of SMPs*

# Clusters: Scalable Parallel Machines




**Each SMP has an *adapter board* (with an embedded processor and network interface) to support *message passing* and *shared memory***

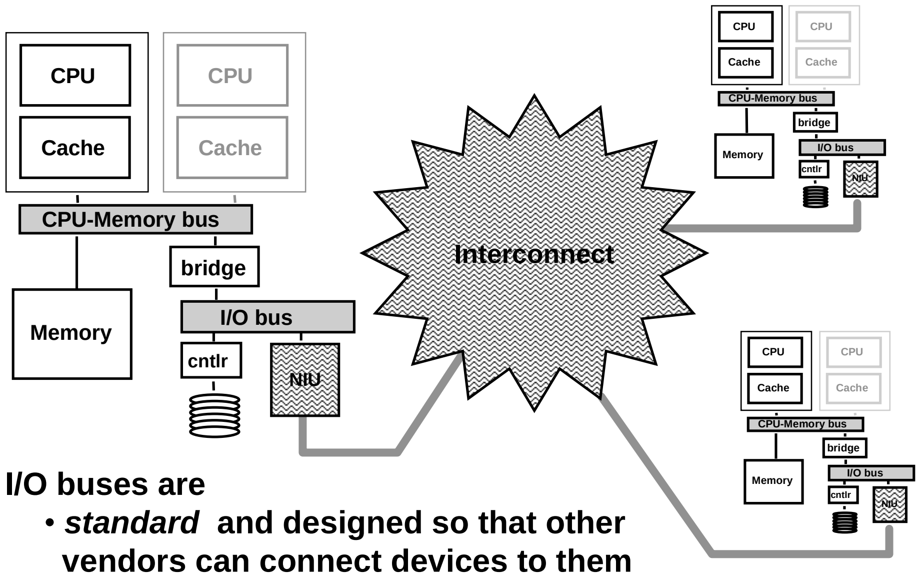
***+ I/O + Parallel OS layer + ...***



# Challenges

- **How to make a cluster look like an SMP**
  ***Shared Memory & Operating System issues***
- **Cost effective networks and Network Interface Units (NIUs)**
- **Fault tolerance**
- **High-level multithreaded *programming model***

# NIU on I/O Bus



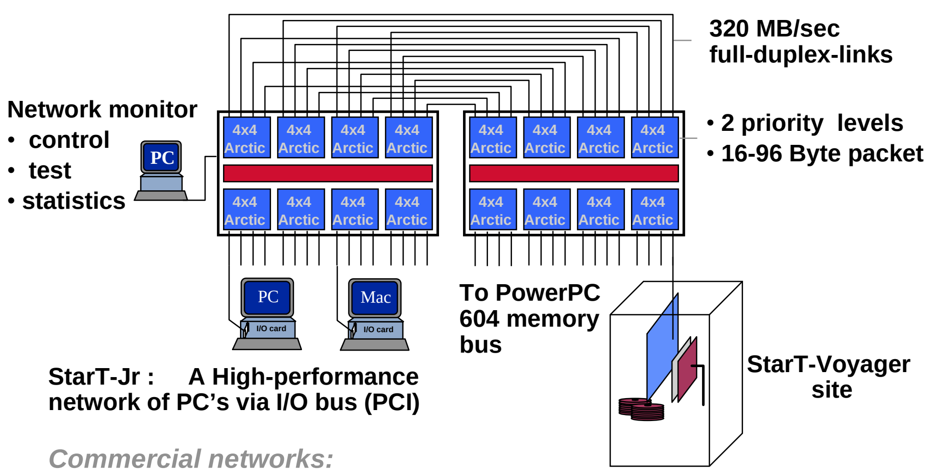

I/O buses are

- *standard* and designed so that other vendors can connect devices to them
- *significantly slower* than memory buses



# Arctic Cluster Interconnect *Andy Boughton*

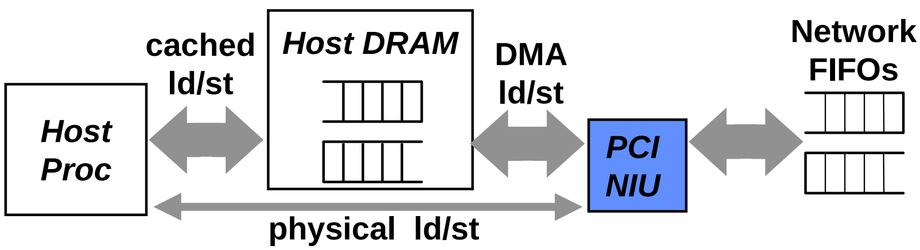
## Extensible Fat Tree




*Commercial networks:*

Myrinet, Dolphin SCI, DEC Memory Channel

# A PCI-based NIU *James Hoe*




PCI-based NIU's to connect SMP clusters via Arctic

| | |
|---|---|
| *SUN:* | 9 8-processor E5000 |
| *Digital:* | 7 4-processor Raw Hides |
| *Intel:* | 32 or more Quads |

*DMA performance determined by the host PCI bus*

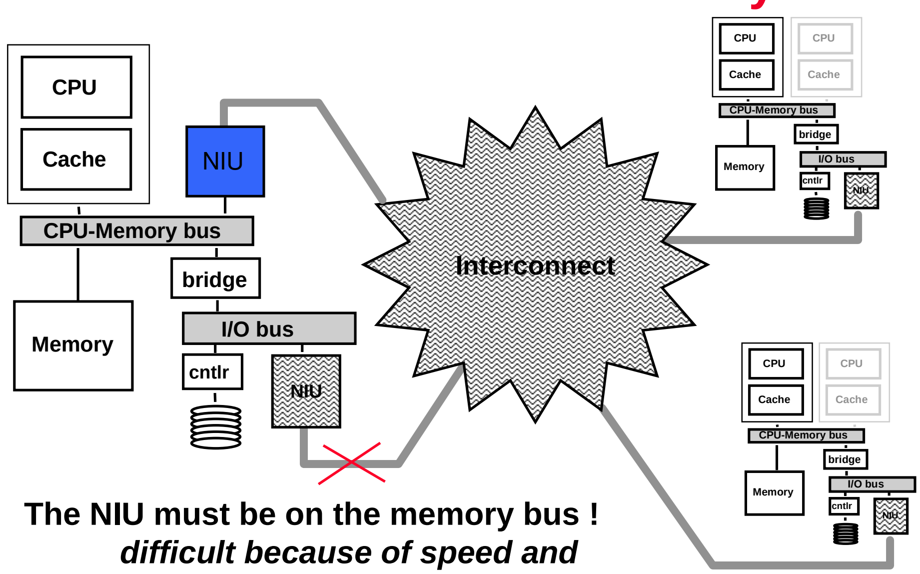
50 to 70 MB/s for sends, upto 90MB/s for receives

*Other projects:*

Shrimp@Priceton, NOW @ Berkeley & Illinois



# Cache-Coherent Distributed Shared Memory




The NIU must be on the memory bus !

*difficult because of speed and proprietary information*

# CCDSM Issues

- NIU must connect to the Processor-Memory bus
  - ⇒ few projects because of difficulty and expense
    - *MIT - Alewife*
    - *Stanford - Dash, Flash*
    - *MIT - Start Voyager*
    - *Commercial machines - KSR, Convex, SGI, ...*
- At the cluster level
  - *protocols are complex*
  - *lack of consensus about memory models*

  In research projects CC is usually done in software
- Directory representation
- NUMA (Non Uniform Memory access Architecture)

  *versus*

  S-COMA (Simple Cache-Only Memory Architecture)



# Outermost Directory Representation

| | | | |
|---|---|---|---|
| H | S | bit-vector C | Memory Block |

Bit-vector is a representation of which children caches have copies of this memory block

*At home*

(H=1, S= _) : no cached copy exists (R[ε])

*Read Only Copies*

(H=0, S=1) : for all $C_i$=1, $i^{th}$ child has a copy (R[Dir])

*Writable Copy at $C_i$*

(H=0, S=0) : for $C_i$=1, $i^{th}$ child has the Ex copy (W[id])

*size?*

# Directory Size

Directory size = (M / B) · [ s+N] / 8 Bytes

*where*

M = Memory size

B = Block size

N = number of children

s = no. of bits to represent the state

For M= $2^{32}$ Bytes, B=64 Bytes, s = 2 bits

Directory size = $(2^{(32-6)}) \cdot (2+N) / 8$ Bytes

= $2^{23} \cdot (2+N)$ Bytes

N=16 $\Rightarrow$ directory $\approx 2^{27}$ Bytes or ~ *4% overhead*

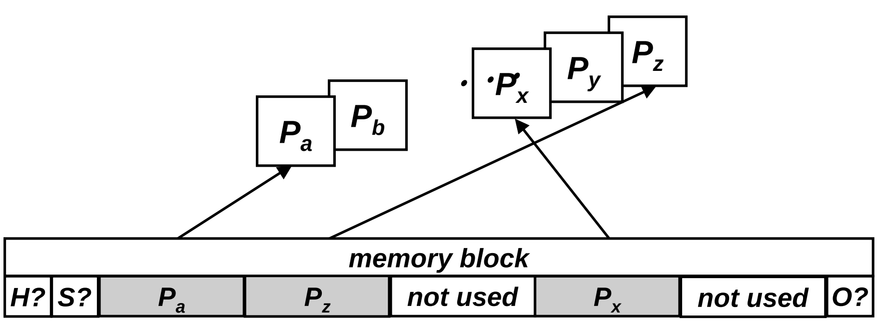
N=256 $\Rightarrow$ directory $\approx 2^{31}$ Bytes or ~ *50 % overhead*

This directory data structure is practical for small N but does not scale well !



# Reducing the Directory Size

*Limitless directories-* Alewife

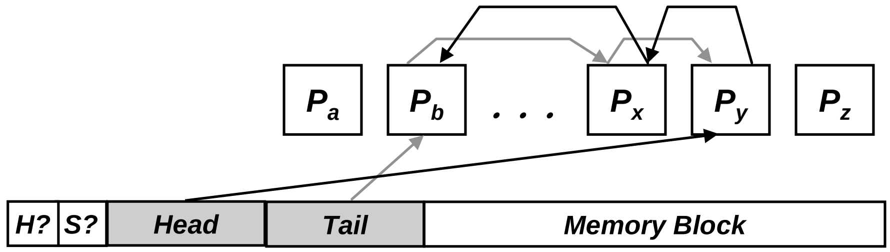



Instead of a N-bit-vector, keep n (*lg N-* bit) pointers; if more than n children request a copy, handle the overflow in software

*effective for large N and low degree of sharing*

# Reducing the Directory Size

## *linked-list* - SCI



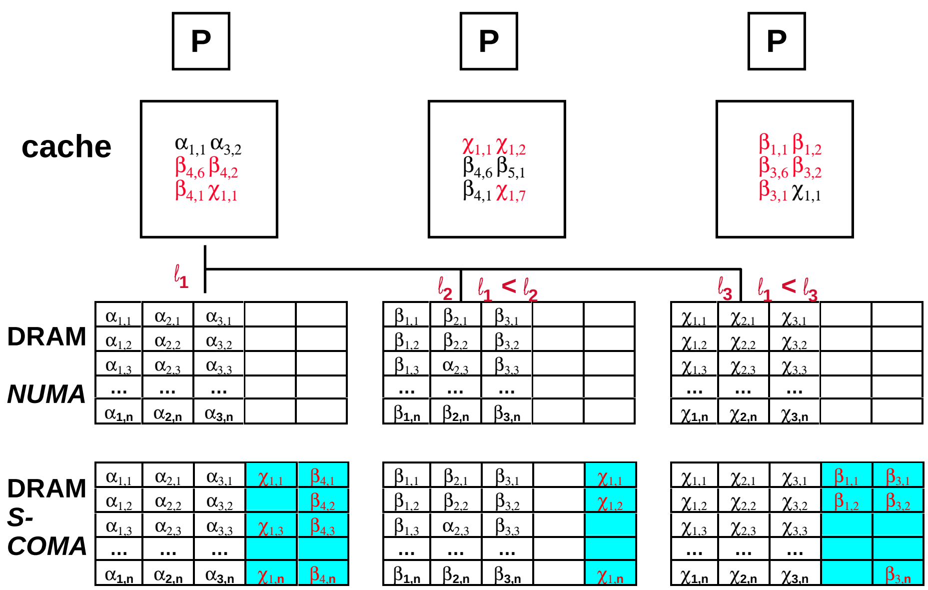

- Part of the directory is attached to each cache
- Home and each cache block keep two (*lg N)*-bit pointers per memory block
- A *doubly linked-list* of the cache blocks holding the same memory block is maintained, with *the root* at the home site

$\Rightarrow$ ***less storage but bad performance!!!***

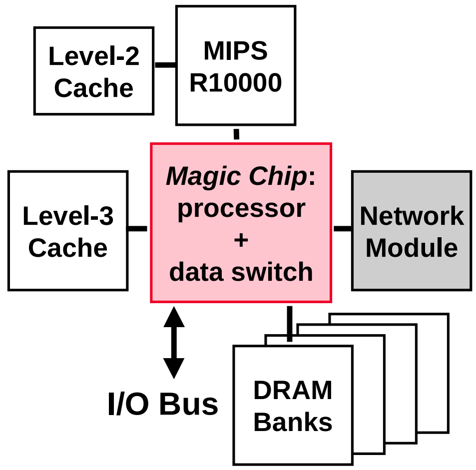
SCI = Scaleable Coherent Interface



# NUMA versus S-COMA



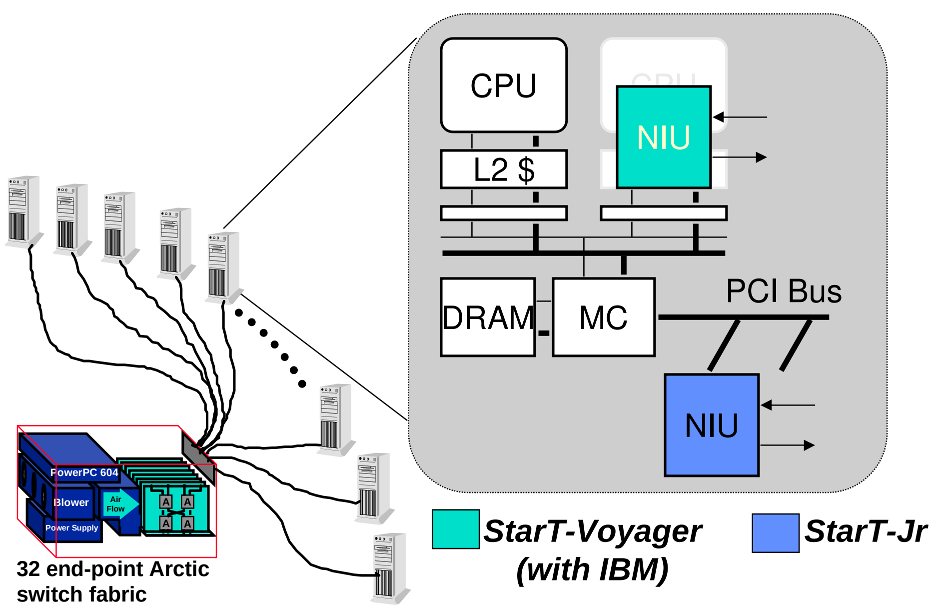
# Stanford's CCDSM: *Flash*




Magic is a static two-way superscalar processor.
Has to be fast because *all* memory traffic goes through it
$\Rightarrow$ ***lots of I/O pins !***

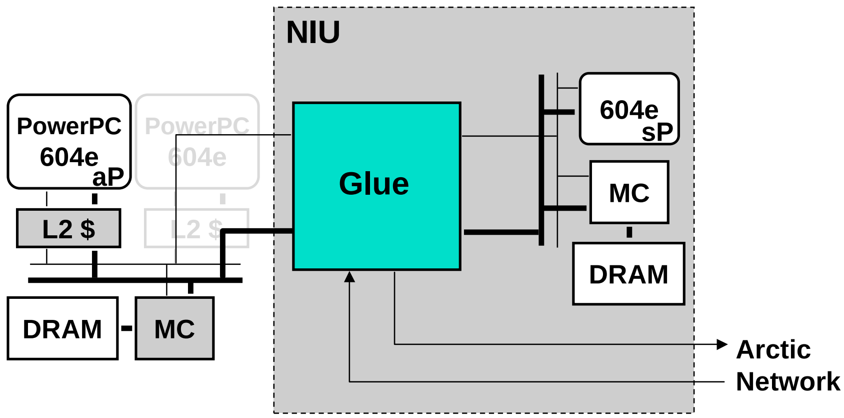


# The StarT Project



# StarT-Voyager

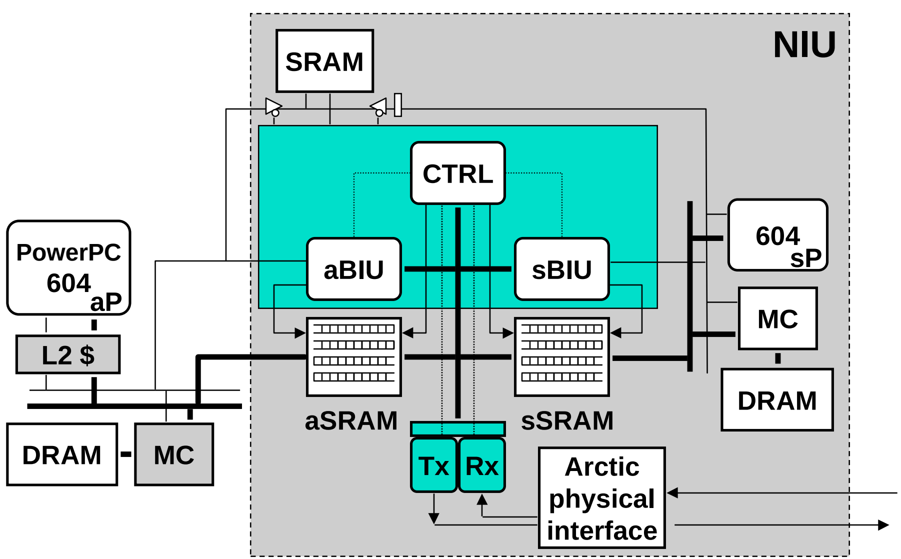
*Boon S. Ang & Derek Chiou*




*Full protection in multi-user time-sharing environment*



# Network Interface Unit



# StarT- Voyager Features

- Four message passing mechanisms optimized for different types of communication
  *basic, express, DMA, tag-on*

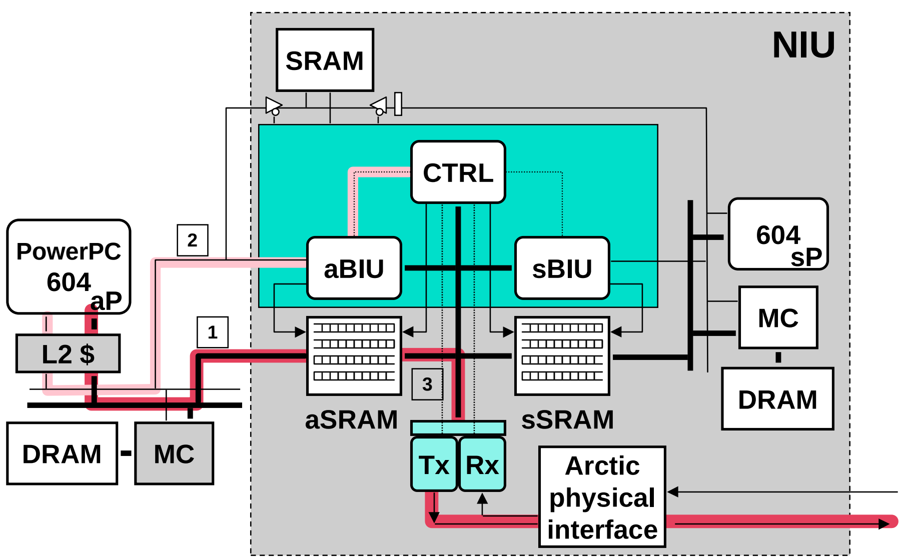
- Global shared memory
  - two mechanisms for inter-site memory accesses
    *S-COMA, NUMA*
  - various memory models & associated protocols

- Full protection in multi-user, time-sharing environment

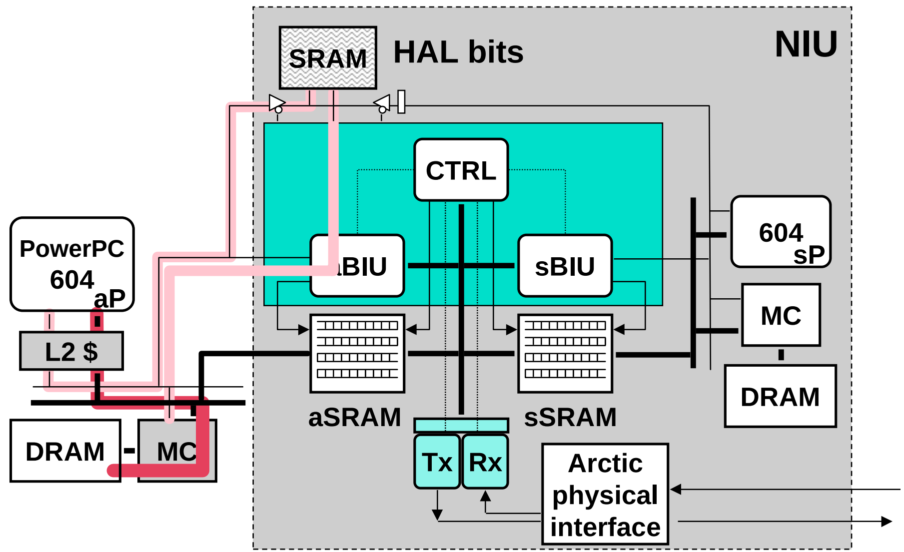
*A 32-Processor machine to be completed in 1998*



# Sending a Basic Message



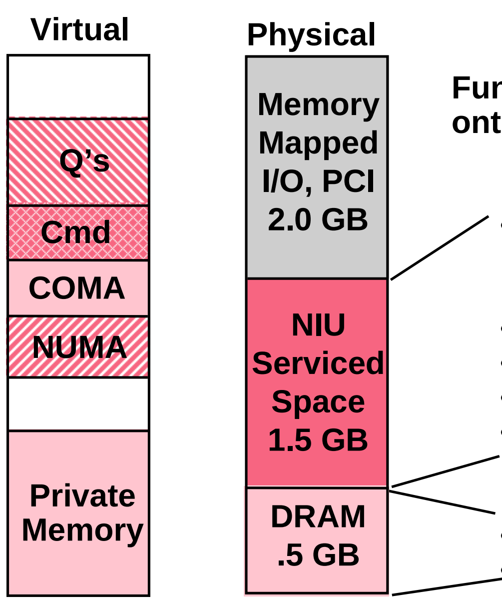
# S-COMA Style Memory Access




HAL: F(Bus action, Address) ⇒ {proceed, abort, sP, ignore}



# Functionality Map




Functionality is mapped onto Physical addresses

- 32 Resident queues
  - Express Tx, Rx
  - Basic Tx, Rx
- 512 Non-resident queues
- cmd & conf NIU
- DMA
- NUMA

- snoop space (COMA)
- local DRAM

# Protection and Multitasking

- Message queues and global address space are viewed as *resources* to be shared amongst processes
- Once a process has acquired a resource, virtual memory translation mechanisms protect access to it
- NIU provides protection in accessing remote resources
- Resident queues are dynamically allocated to the current process (soft context switch)



# What’s Next ?

We have covered a lot of ground -
now you should be able to read, appreciate, criticise the description of any modern microprocessor

We did not discuss
- farout ideas for processor architecture
  multi-threading, dataflow, speculative stores,...
- farout ideas for memory systems
  PIM, I-structures, ...
- special purpose architectures
  graphics, multimedia, ...
- Designing experiments to evaluate architectures
- Formal verification of implementations