MINI-Processors: Network Interface Cards (NICs) as First-Class Citizens

Wu-chun Feng*†
feng@lanl.gov
http://home.lanl.gov/feng

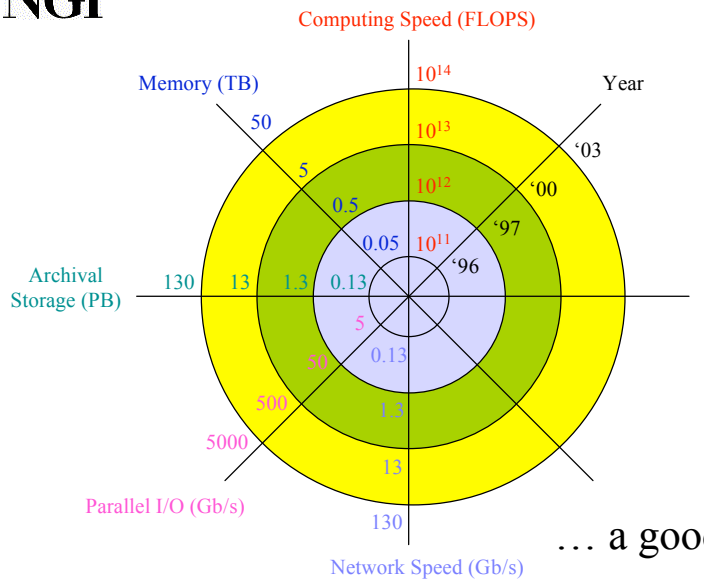
* Los Alamos National Laboratory
Los Alamos, NM 87545

Purdue University†
W. Lafayette, IN 47907

Funded in part by DOE Next-Generation Internet (NGI)

Presented at The Ohio State University, 11/18/99

The ASCI Target (or ASCI Curves)



… a good start but ...

Recent Solutions Between Processor & Network

- HiPPI-6400 NIC (beta prototype) ⟵ 6400 Mb/s (6.4 Gb/s)
  - NIC processor to free CPU from network operations.
  - Hardware capabilities
    - IP checksum
    - Error detection and re-transmission
    - Flow control
    - Low-level messaging operations for OS-bypass protocols.
- OS-Bypass Protocol
  - Orders-of-magnitude reduction in app-to-network latency.
- Problem
  - Application-to-network (vice versa) still a bottleneck!

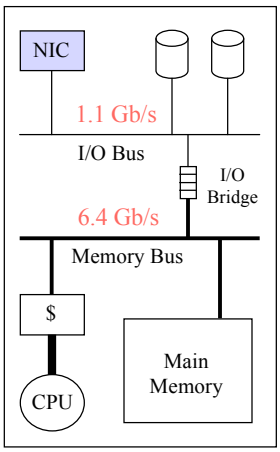
Current PC Technology

Goal: Alleviate application/network bottleneck.

(Example) Benefits

- Enable QoS in middleware.
- WWW ≠ World Wide Wait
- Remote Viz (FY01): 80 GB/s = 640 Gb/s.
- High-speed bulk data transfer.

Component	"Latency"	"Bandwidth"
CPU	1-2 *ns*	3.6 *Gi/s*
DRAM access time	60-100 *ns*	6.4 *Gb/s*
Network link	1 μ *s*	6.4 *Gb/s*
Memory bus	10 *ns*	6.4 *Gb/s*
I/O bus	15 *ns*	1.1 *Gb/s*
Appl-to-network (TCP/IP)	100-150 μ *s*	0.25-0.50 *Gb/s*
Appl-to-network (OS byp)	3 μ *s*	0.60-0.90 *Gb/s*



Trends

- CPU Speed: Doubling every 1.5 years.
- Memory Access Speed: 7% - 9% increase / year.
- Memory Capacity: Quadrupling every 3 years.
- Network Link BW: Doubling every year.
  - 10 Mb/s Ethernet (1988) to 6400 Mb/s HiPPI (1998)

PC technology

The future for I/O Bus and Memory Bus ...
- *PCI-X:* 4.3 Gb/s (1Q00)
- *RAMBUS:* 9.6 Gb/s, 28.8 Gb/s, 86.4 Gb/s ("now").

Supercomputer technology

SGI O2K (now): XIO BW = 6.4 Gb/s max, 0.8 Gb/s actual

Problems: Directory-based ccNUMA & 10:1 CPU:NIC ratio.

NICs as First-Class Citizens

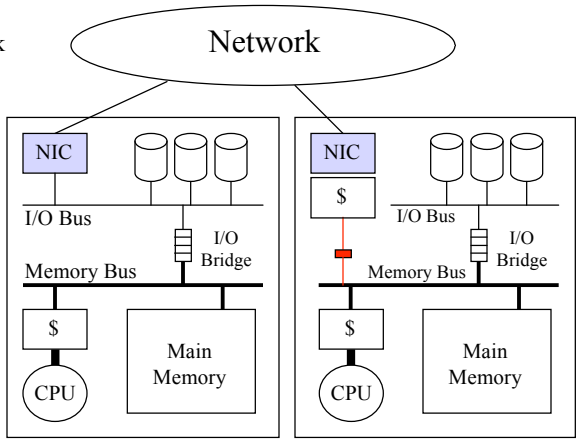
Goals
- Alleviate application/network bottleneck.

↓

- Move NIC to memory bus.
  - What's new?
- Integrate NIC into memory subsystem.
- Treat NIC as a peer CPU.

↓

That is, *m*emory-*i*ntegrated, *n*etwork-*i*nterface *processors* (MINI-Processors)



Note: Each node could contain multiple CPUs.

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & spec.

Move NIC from I/O Bus to Memory Bus

- I/O Bus
  - \+ Standard interface (e.g., PCI)
  - – High latency (e.g., PCI = 10-14 cycles = 300-425 *ns*)
  - – Low bandwidth (e.g., PCI = 1.1 Gb/s peak bandwidth)
- Memory Bus
  - – Non-standard interface but bridges possible (e.g., Intel AGP)
  - \+ Low latency (e.g., Intel DRAM = 60-100 *ns*)
  - \+ High bandwidth (e.g., Intel AGP = 4.2 *Gb/s* peak bandwidth)
  - \+ *Cache coherency*

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & spec.

Virtualize NIC & Bypass OS

- OS-Based Network Protocols
  - – High latency to access NIC
    - Packets go through OS via Unix sockets.
    - High DMA initiation overhead.
  - + Easy protection of address spaces
  - + Easy address translation for mbufs
- OS-Bypass Network Protocols (e.g., ST, PM, FM, etc.)
  - + Lower-latency and higher-bandwidth access to NIC
    - Use virtual memory HW to virtualize NIC, i.e., memory-map NIC.
    - Bypass OS.

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & spec.

Cache NIC Registers

- NIC Registers Currently Uncached
  - – High latency
  - – Low bandwidth
  - – CPU accesses to NIC may have side effects (unlike normal cache memory)
- Cache NIC Registers in CPU Cache(s)
  - + Complementary advantages of the above
  - + Exploit temporal locality
  - – False sharing

NICs as First-Class Citizens

Goals

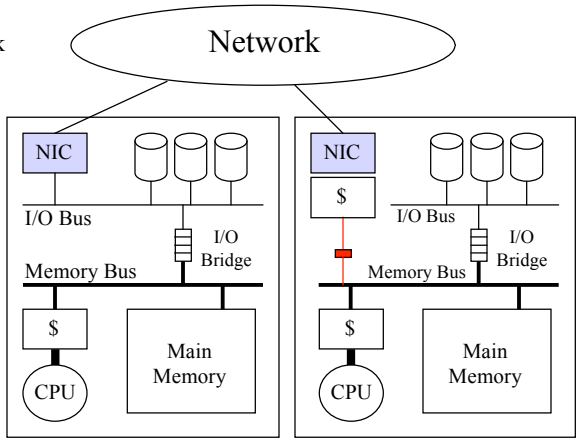
- Alleviate application/network bottleneck.

↓

- Move NIC to memory bus.
  - What's new?
- Integrate NIC into memory subsystem.
- Treat NIC as a peer CPU.

↓

That is, *m*emory-*i*ntegrated, *n*etwork-*i*nterface *processors* (MINI-Processors)



Note: Each node could contain multiple CPUs.

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & spec.

Transfer Packets via Cache Block Transfers

- I/O Transfer
  - Uncached load/stores to memory-mapped device registers transfer very few bytes
  - High DMA initiation overhead
  - User-level DMA has side effects
- Cache Block Transfer
  - \+ High bandwidth
  - \+ Memory buses are optimized for cache block transfer
  - \+ Cache coherency

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & spec.

Memory-Based Queue API

- Memory-Based Queue API vs. User-Level NIC API
  - + Decouples NIC from CPU
    - Sending/receiving packets = reading/writing queue memory
    - Both CPU and NIC can send/receive multiple packets to/from queues without blocking
  - + Avoids side effects by treating NIC queue accesses as side-effect-free memory accesses.

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & spec.

Proper Notification

- Interrupt
  - Heavyweight
  - Corrupts the cache(s). Adversely affects cache hit rate.
    - Results in added memory-bus traffic.
- Cache Invalidation
  - + "Non-intrusive"
    - NIC invalidates cached NIC register in CPU's cache.
    - CPU misses on cached but invalidated NIC register & gets valid NIC register from NIC.

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & spec.

Buffering Packets

- Use device memory of NIC
  - – Limited buffer space
- Use virtual memory
  - \+ Plentiful buffer space

  Issues
  - Address translation for users' virtual addresses in NIC.
  - Protected NIC access to main memory.

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & speculation

[NIC Access ≡ Memory Access] → Out-of-Order Access Possible

+ Additional scheduling flexibility in a dynamic pipeline.
  - Certain loads/stores
    - May be scheduled earlier than other loads/stores
    - May be completely bypassed
  - CPU may not need to stall ...

NIC Access ≡ Memory Access

I/O Access	Memory Access
Device on I/O bus	Memory on memory bus
Indirect via operating system (OS)	Direct via protected user access
Uncached NIC registers	Cached NIC registers
Ad hoc data movement	Cache block transfers
Explicit data movement via API	Memory-based queue
Notification via interrupts	Notification via cache invalidation
Limited device memory	Plentiful memory
No out-of-order access & spec.	Out-of-order access & speculation

Predictions for Next-Generation Supercomputer

- Bottlenecks
  - Application-to-memory interface
  - Application-to-network interface
  - Application-to-storage interface
- Fault Tolerance
  - Machine with 10K nodes (each similar to a desktop computer) should have a MTBF that is 10K times worse than a desktop computer!
  - The question is not *whether* anything is broken at any given time, but rather *how many* components are broken (and how the machine works in the presence of broken network links or nodes).

Status

- Internal interface: Memory access.
- External interface: Myrinet.
  - Problems: Myrinet performance degrades under heavy load.
    Nearly all other technologies are PCI I/O-based.
  - Solution: HiPPI-6400 when commercially available.
    Use Myrinet to prototype for now.
- Implementation of Intel x86-based simulator underway.
- DARPA SLAAC1 FPGA card for hardware prototype.
  - Linux device driver is done.
  - Porting of user-mode code is underway.

Future Work

- Finalization of internal/external interface design. (12/99)
- Simulator
  - Issues to address in presence of a cache-coherent NIC.
    - Optimized bus-based cache-coherency protocol. (4/00)
    - Scalability of system bus. (7/00)
  - NIC integration within CPU?
- Hardware Prototype
  - Initial design of hardware prototype. (2/00)
  - Completion of initial prototype. (7/00)
  - Simulation results guide evolution of prototype. (Ongoing …)
- Continuing discussions with Intel.