# Network Interface Cards (NICs) as First-Class Citizens

Wu Feng[*†]

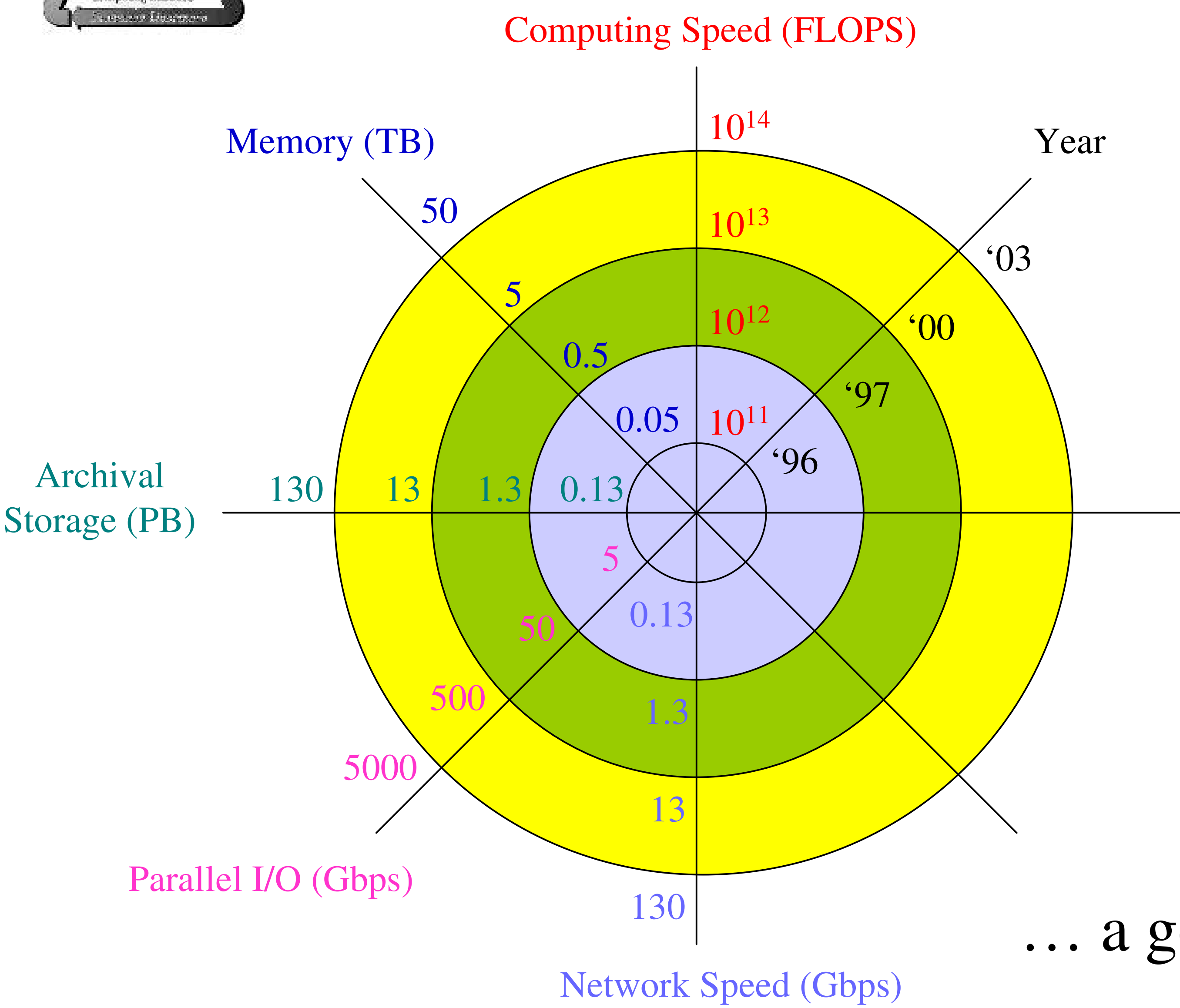
feng@lanl.gov

Ronald G. Minnich[*] and Fabrizio Petrini[*]

[*] Los Alamos National Laboratory
Los Alamos, NM 87545

Purdue University[†]
W. Lafayette, IN 47907

# 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

| Component | "Latency" | "Bandwidth" |
|---|---|---|
| CPU | 1-2 *ns* | 3.6 *Gips* |
| DRAM access time | 60-100 *ns* | 6.4 *Gbps* |
| Network link | 1 $\mu s$ | 6.4 *Gbps* |
| Memory bus | 10 *ns* | 6.4 *Gbps* |
| I/O bus | 15 *ns* | 1.1 *Gbps* |
| Appl-to-network (TCP/IP) | 100-150 $\mu s$ | 0.25-0.50 *Gbps* |
| Appl-to-network (OS byp) | 3 $\mu s$ | 0.60-0.99 *Gbps* |

Goal
- Alleviate app/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.

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

SGI XIO? 6.4 Gb/s max, 0.8 Gb/s delivered.

Problems: Directory-based cache coherency (ccNUMA) & 10:1 CPU:NIC ratio.

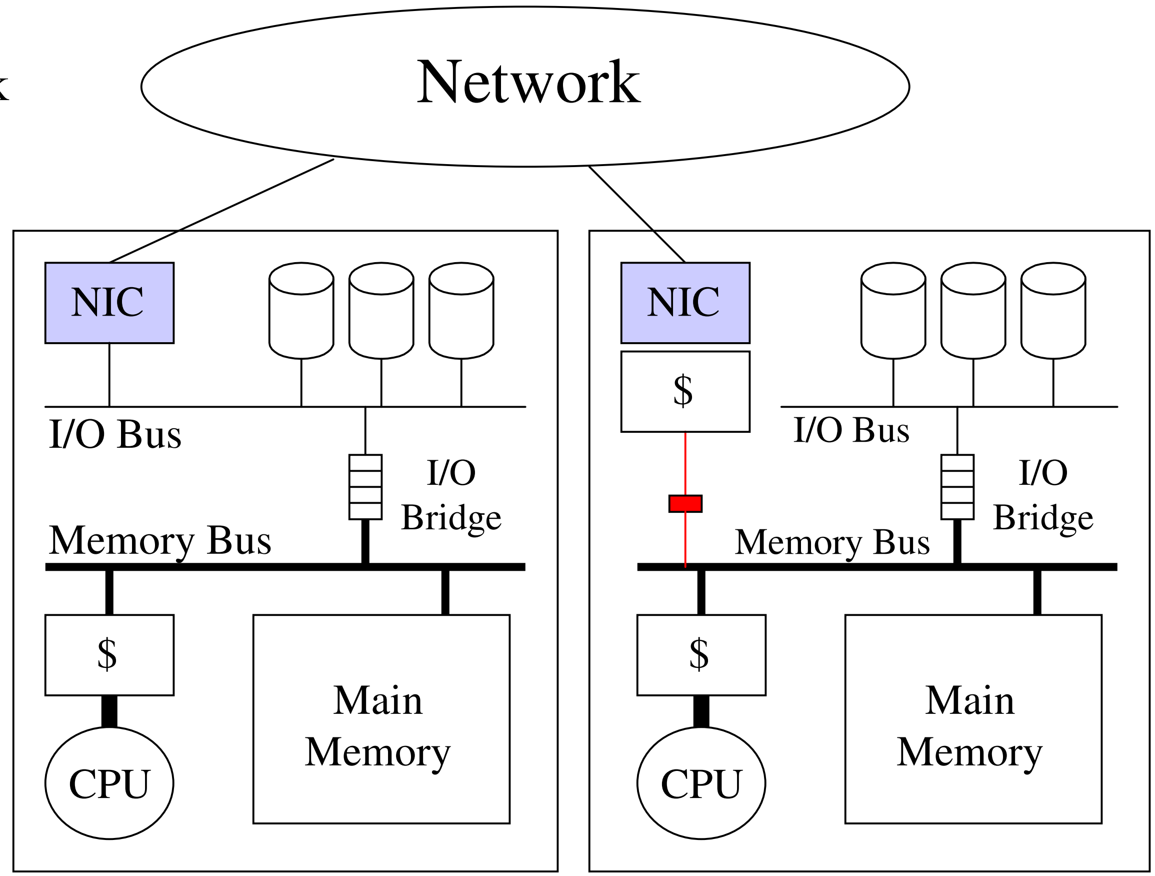
# NICs as First-Class Citizens

Goals

- Alleviate application/network bottleneck.

↓

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




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 & speculation |

# Status

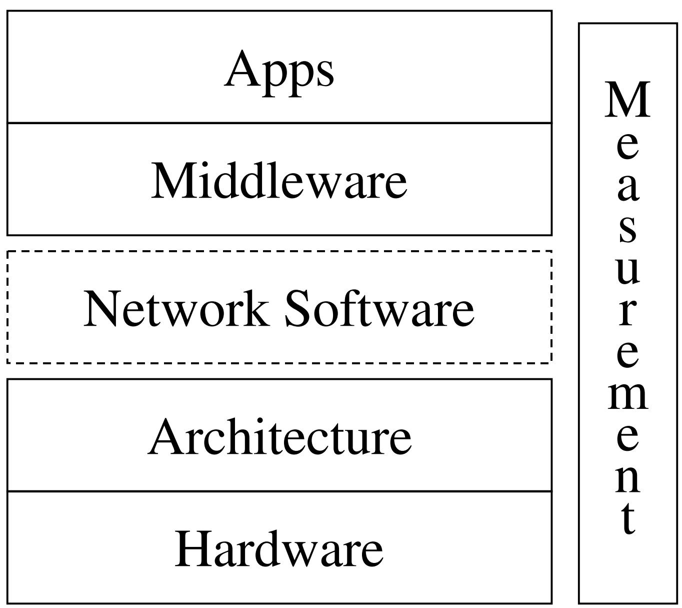
- Internal interface: Memory access.
- External interface: Myrinet.
  - Problems: Myrinet performance degrades under heavy load. Nearly all other technologies are PCI I/O-based.
  - Solution: HiPPI-6400 when commercially available. Use Myrinet to prototype for now.
- Implementation of Intel x86-based simulator underway.
- Re-evaluation of FPGAs to implement our NIC processor: SLAAC instead of RCA-2 & Pamette.

# Future Work

- Finalization of internal/external interface design. (12/99)
- Simulator
  – Issues to address in presence of a cache-coherent NIC.
    - Bus-based cache-coherency protocol. (5/00)
    - Scalability of system bus. (10/00)
  – NIC integration within CPU?
- Hardware Prototype
  – Identification of existing CAD environment to support work. (11/99)
  – Completion of initial prototype. (10/00)
  – Simulation results guide evolution of prototype. (Ongoing …)
- Continuing discussions with Intel.

# DOE NGI Program Structure

- "Regular" NGI PI meetings
  - To address interface issues between NGI technologies.
  - To stay abreast of the work of fellow NGI colleagues.




Network software implies
- High-speed protocols
- QoS protocols
- OS/network interface

# DOE NGI Program Structure

- More tightly-coupled collaboration across academia, government labs, and industry.
  - Why?
    - Advance research & development at a faster rate.
    - Enhance visibility of the overall DOE NGI program.
    - Ensure that our basic research has practical application.
  - Examples
    - HiPPI-6400 (LANL and SGI)
    - NICs as First-Class Citizens (LANL, Purdue, and Intel)