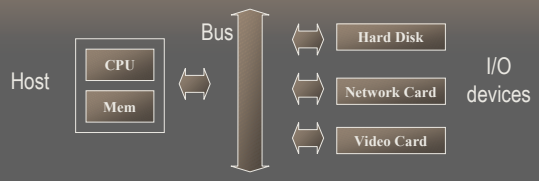
Split OS

Matias Cuenca, Kiran Nagaraj, Kiran Srinivasan, Florin Sultan, Ricardo Bianchini, Richard Martin, Thu D. Nguyen and Liviu Iftode

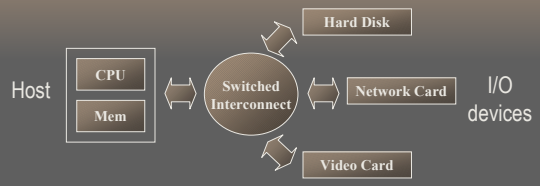
Disco Laboratory, Rutgers University
http://discolab.rutgers.edu

Traditional Computer Architecture



- Devices connected by a shared I/O bus
- Poor scalability
- Limited reliability: a buggy I/O controller can easily crash the system
- Traditional OS runs entirely on the host

New Architecture for Servers

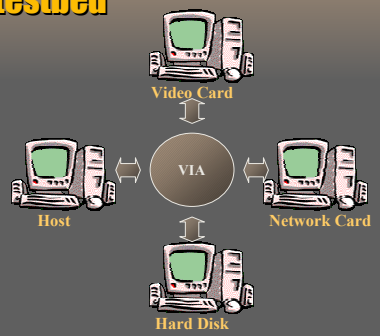


- Devices connected to the host through message passing I/O fabric
  - Increased scalability, reliability and usability
- Intelligent devices
  - RISC processors and memory on board
- What should the operating system look like?
  - Split OS

Split OS

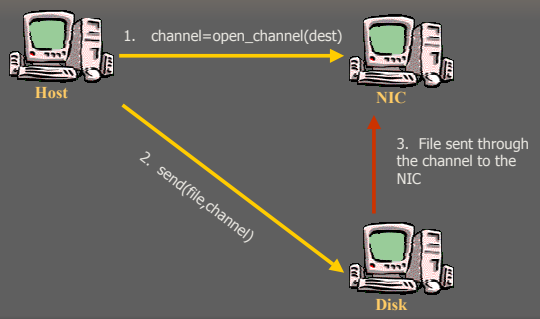
- How to split and replicate the OS state and functions between the host and the intelligent devices?
  - Split to leverage device intelligence to increase performance
  - Replicate to tolerate failures
- How to use the idle cycles of the intelligent devices?
  - Applications code (disklets, etc.)
  - OS code (OS layers, caching, prefetching, probing, housekeeping chores, etc.)
- How to eliminate host from device to device communication path (e.g. disk to network)?

Our testbed

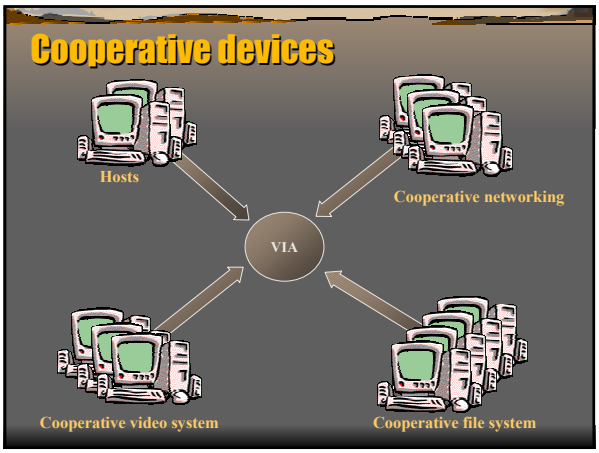


Emulate a cluster of intelligent devices with a cluster of computers connected by a VIA SAN

Example of how I/O devices can bypass the host



Cooperative devices



First steps

- Split the file system between the host and the disks
- Define an API that enable devices to bypass the host
- Distribute the state of the TCP/IP stack among several NICs to provide fault tolerance at the connection level