TCP Servers:

Offloading TCP Processing in Internet Servers. Design, Implementation, and Performance

M. Rangarajan, A. Bohra, K. Banerjee, E.V. Carrera, R. Bianchini, L. Iftode, W. Zwaenepoel.

Presented by:

Thomas Repantis

`trep@cs.ucr.edu`

Overview

To execute the TCP/IP processing on a dedicated processor, node, or device (the TCP server) using low-overhead, non-intrusive communication between it and the host(s) running the server application.

Three TCP Server architectures:

1. A dedicated network processor on a symmetric multiprocessor (SMP) server.
2. A dedicated node on a cluster-based server built around a memory-mapped communication interconnect such as VIA.
3. An intelligent network interface in a cluster of intelligent devices with a switch-based I/O interconnect such as Infiniband.

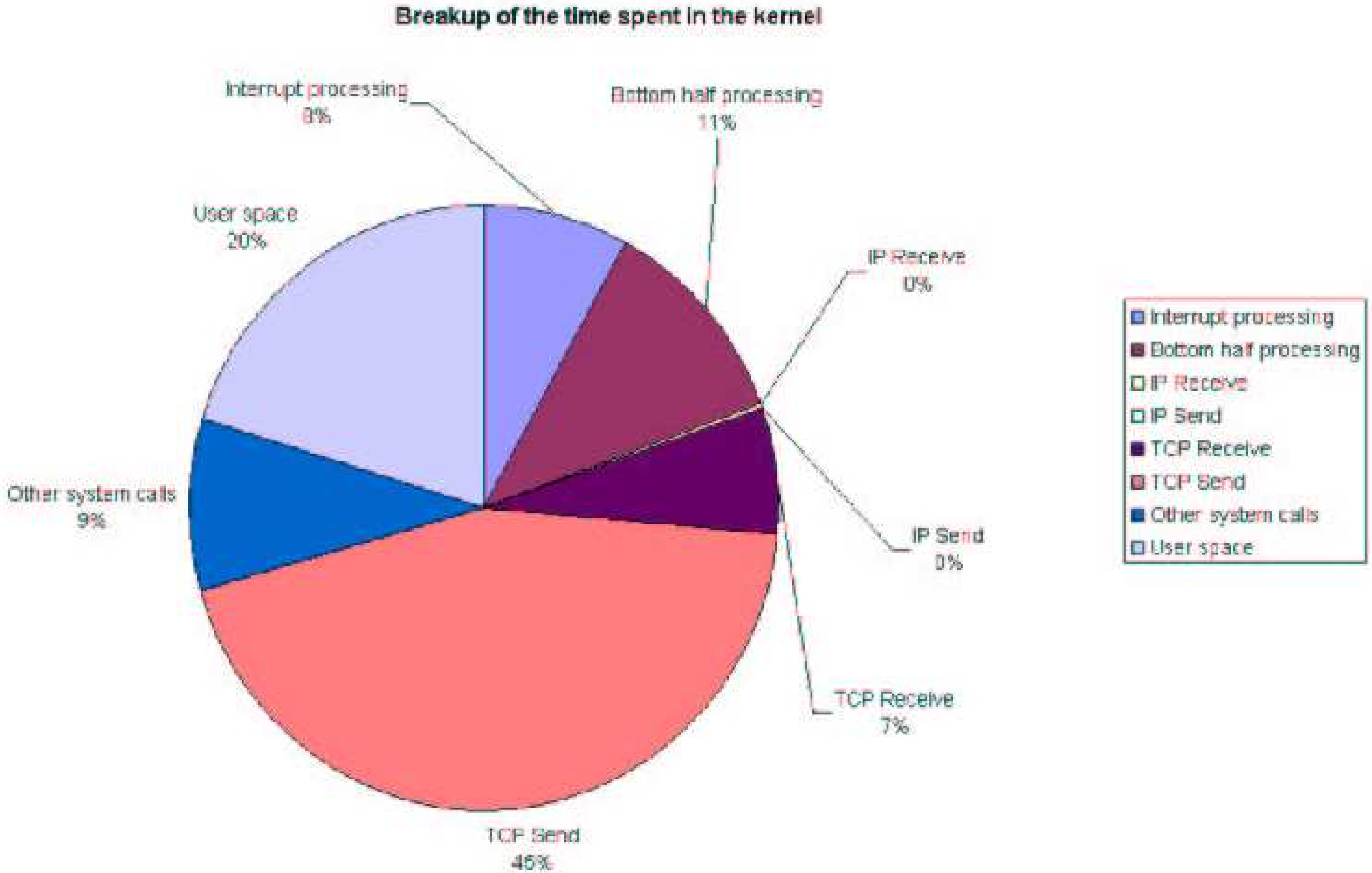
Introduction

- The network subsystem is nowadays one of the major performance bottlenecks in web servers: Every outgoing data byte has to go through the same processing path in the protocol stack down to the network device.
- Proposed solution a TCP Server architecture: Decoupling the TCP/IP protocol stack processing from the server host, and executing it on a dedicated processor/node.

Introductory Details

- The communication between the server host and the TCP server can dramatically benefit from using low-overhead, non-intrusive, memory-mapped communication.
- The network programming interface provided to the server application must use and tolerate asynchronous socket communication to avoid data copying.

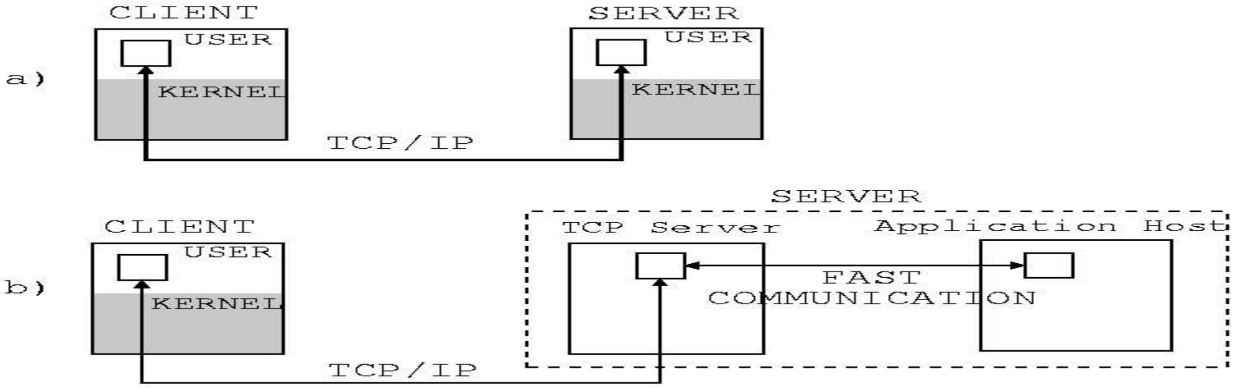
Apache Execution Time Breakdown



Motivation

- The web server spends in user space only 20% of its execution time.
- Network processing, which includes TCP send/receive, interrupt processing, bottom half processing, and IP send/receive take about 71% of the total execution time.
- Processor cycles devoted to TCP processing, cache and TLB pollution (OS intrusion on the application execution).

TCP Server Architecture



- The application host avoids TCP processing by tunneling the socket I/O calls to the TCP server using fast communication channels.
- Shared memory and memory-mapped communication for tunneling.

Advantages

- Kernel Bypassing.
- Asynchronous Socket Calls.
- No Interrupts.
- No Data Copying.
- Process Ahead.
- Direct Communication with File Server.

Kernel Bypassing

- Bypassing the host OS kernel.
- Establishing a socket channel between the application and the TCP server for each open socket.
- The socket channel is created by the host OS kernel during the socket call.

Asynchronous Socket Calls

- Maximum overlapping between the TCP processing of the socket call and the application execution.
- Avoid context switches whenever this is possible.

No Interrupts

- Since the TCP server exclusively executes TCP processing, interrupts can be apparently easily and beneficially replaced with polling.
- Too high polling frequency rate would lead to bus congestion while too low would result in inability to handle all events.

No Data Copying

- With asynchronous system calls, the TCP server can avoid the double copying performed in the traditional TCP kernel implementation of the send operation.
- The application must tolerate the wait for completion of the send.
- For retransmission, the TCP server can read the data again from the application send buffer.

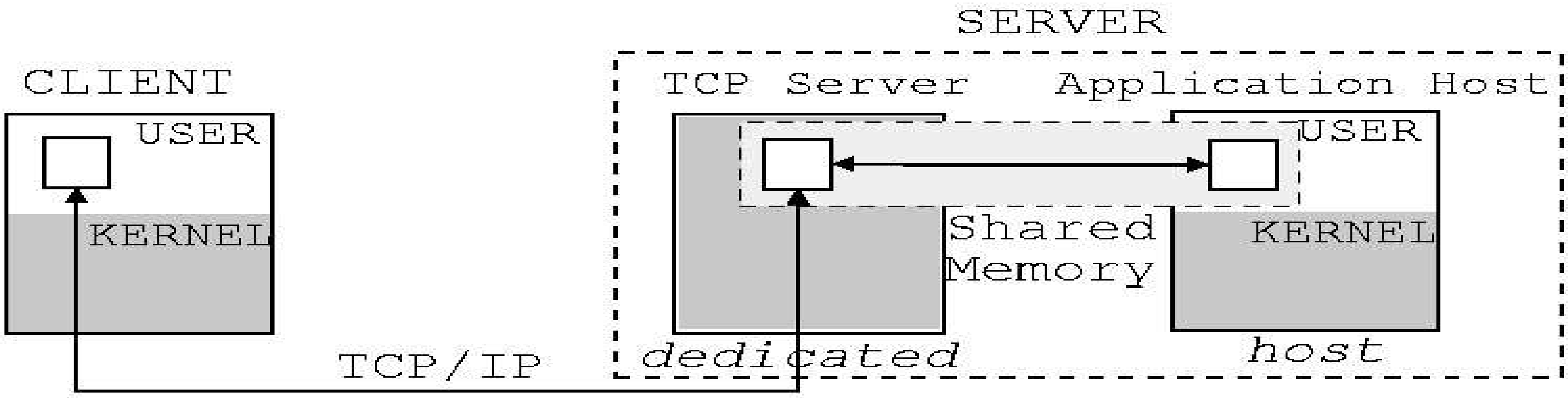
Process Ahead

- The TCP server can execute certain operations ahead of time, before they are actually requested by the host.
- Specifically, the accept and receive system calls.

Direct Communication with File Server

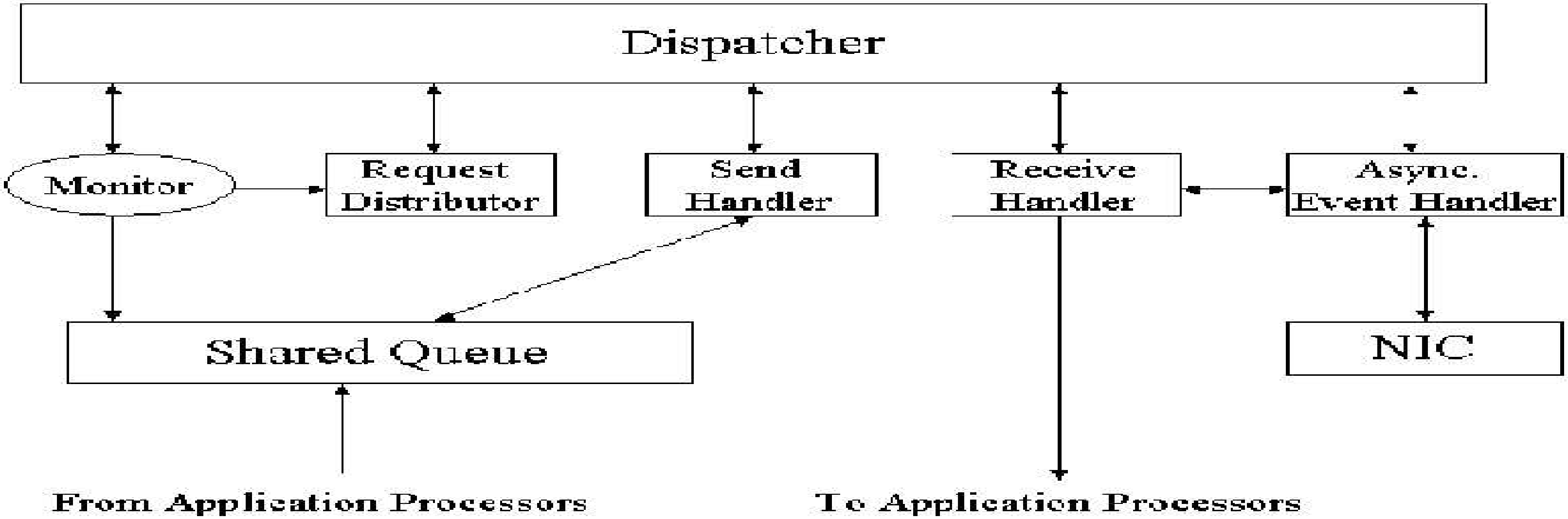
- In a multi-tier architecture a TCP server can be instructed to perform direct communication with the file server.

TCP Server in an SMP-based Architecture



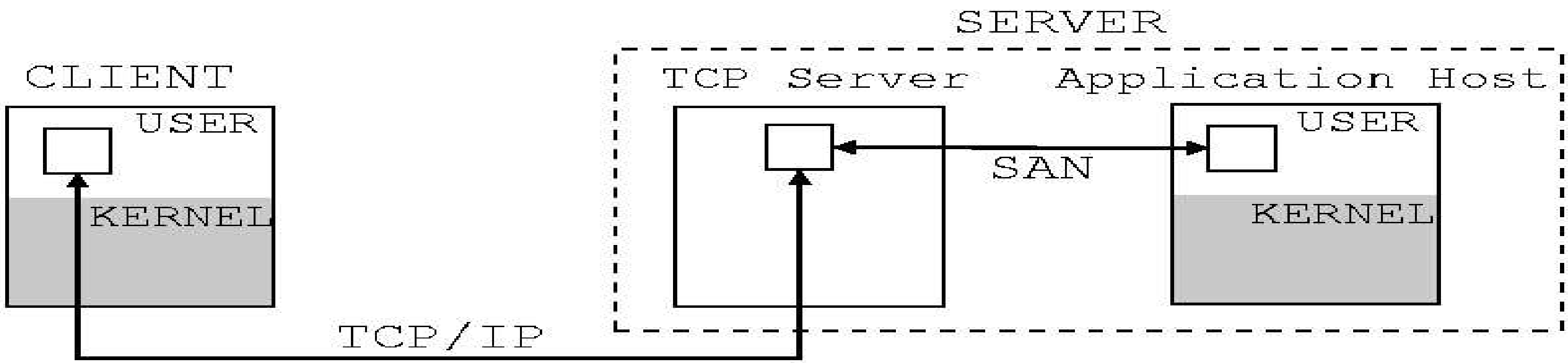
- Dedicating a subset of the processors for in-kernel TCP processing.
- Network generated interrupts are routed to the dedicated processors.
- The communication between the application and the TCP server is through queues in shared memory.

SMP-based Architecture Details



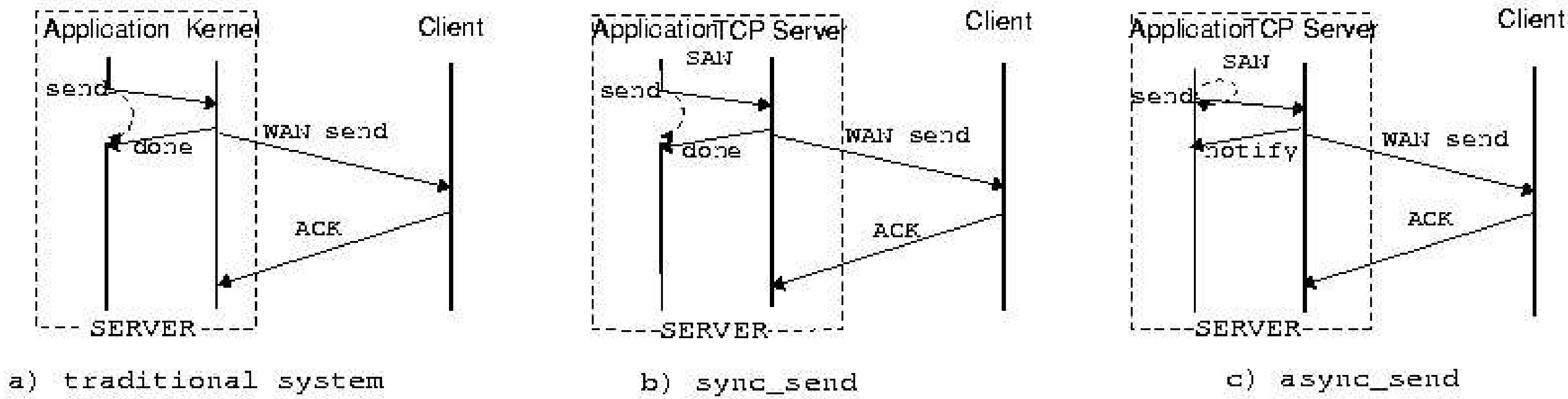
- Offloading interrupts and receive processing.
- Offloading TCP send processing.

TCP Server in a Cluster-based Architecture



- Dedicating a subset of nodes to TCP processing.
- VIA-based SAN interconnect.

Cluster-based Architecture Operation

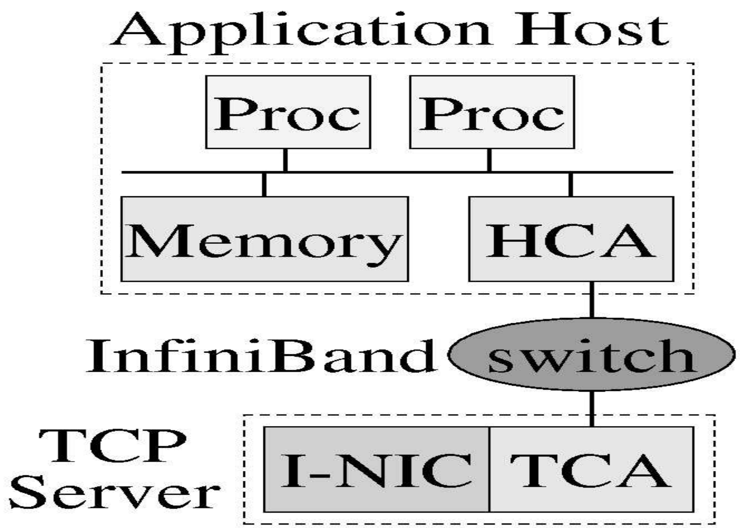


- The TCP server node acts as the network endpoint for the outside world.
- The network data is transferred between the host node and the TCP server node across SAN using low latency memorymapped communication.

Cluster-based Architecture Details

- The socket call interface is implemented as a user level communication library.
- With this library a socket call is tunneled across SAN to the TCP server.
- Several implementations:
  1. Split-TCP (synchronous)
  2. AsyncSend
  3. Eager Receive
  4. Eager Accept
  5. Setup With Accept

TCP Server in an Intelligent-NIC-based Architecture

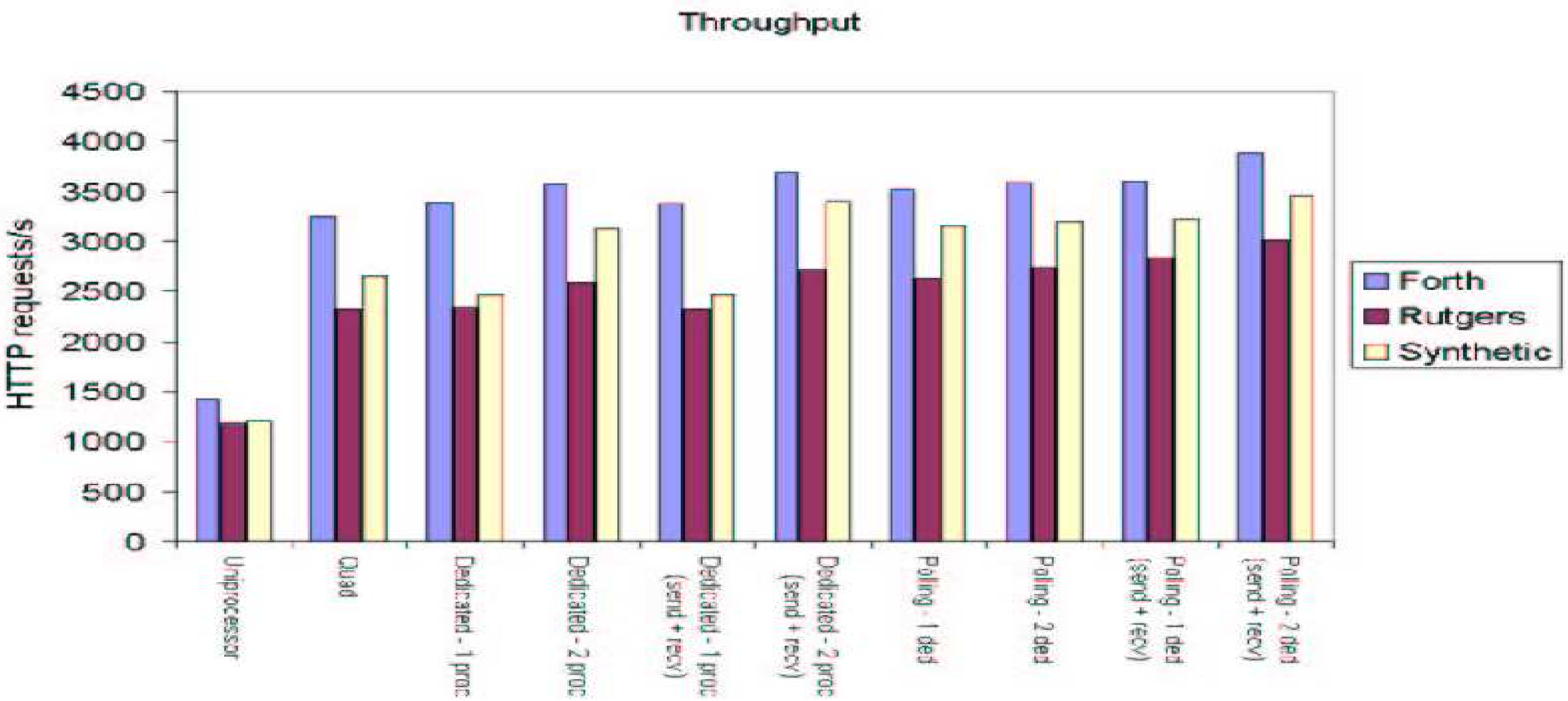


- Cluster of intelligent devices over a switched-based I/O (Infiniband).
- The devices are considered to be "intelligent", i.e., each device has a programmable processor and local memory.

Intelligent-NIC-based Architecture Details

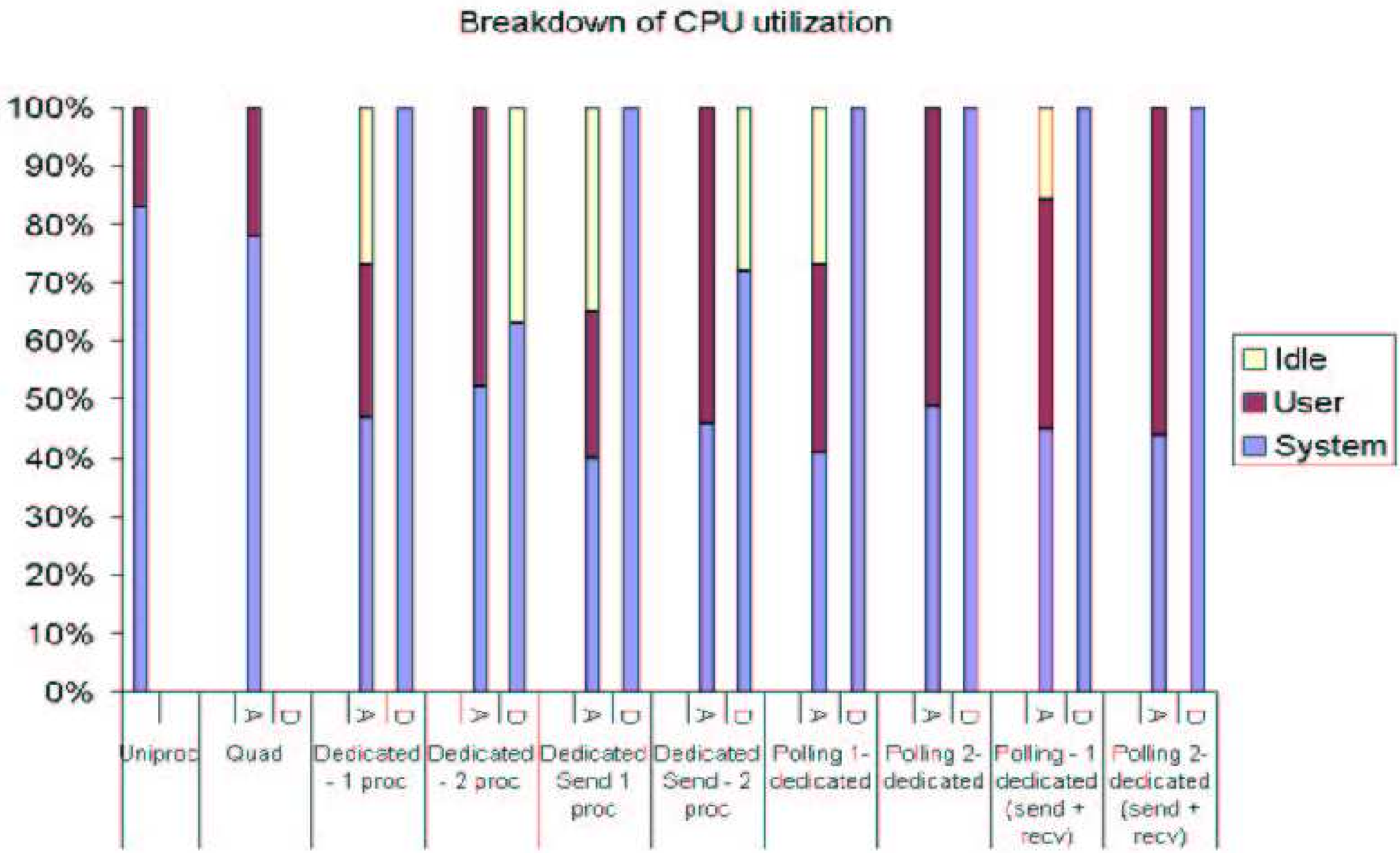
- Each open connection is associated with a memory-mapped channel between the host and the I-NIC.
- During a message send, the message is transferred directly from user-space to a send buffer at the interface.
- A message receive is first buffered at the network interface and then copied directly to user-space at the host.

4-way SMP-based Evaluation



- Dedicating two processors to network processing is always better than dedicating only one.
- Throughput benefits of up to 25-30%.

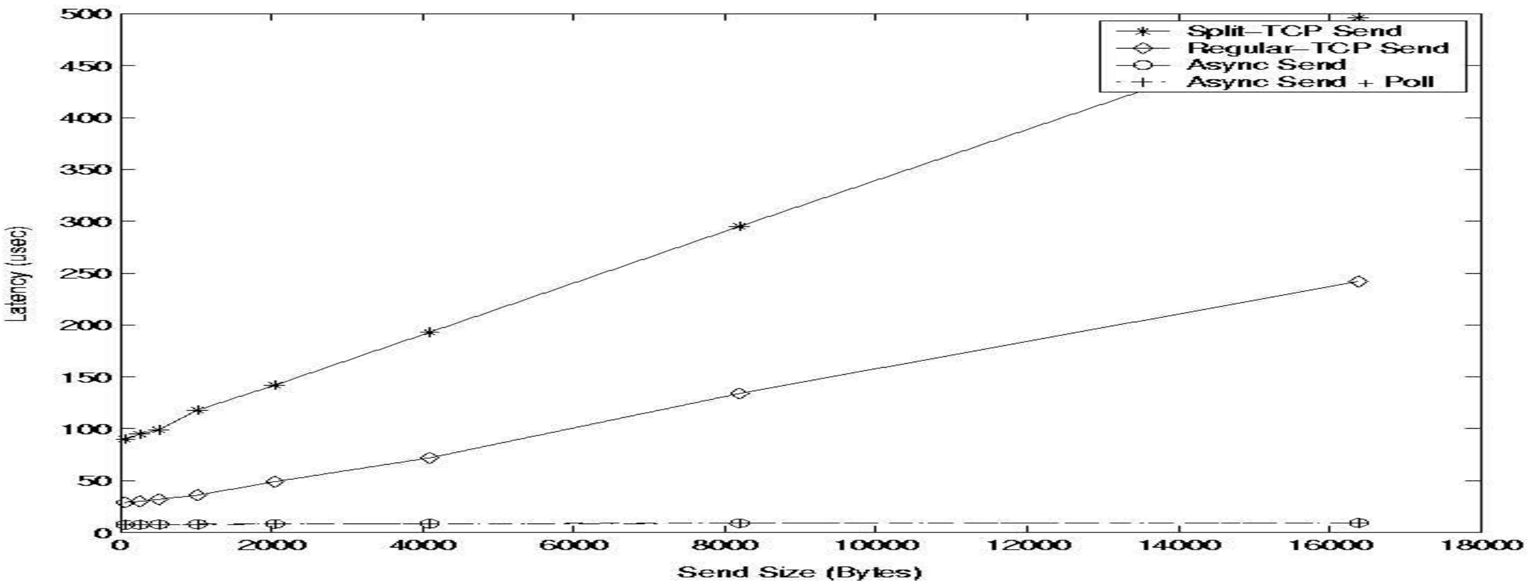
4-way SMP-based Evaluation



4-way SMP-based Evaluation

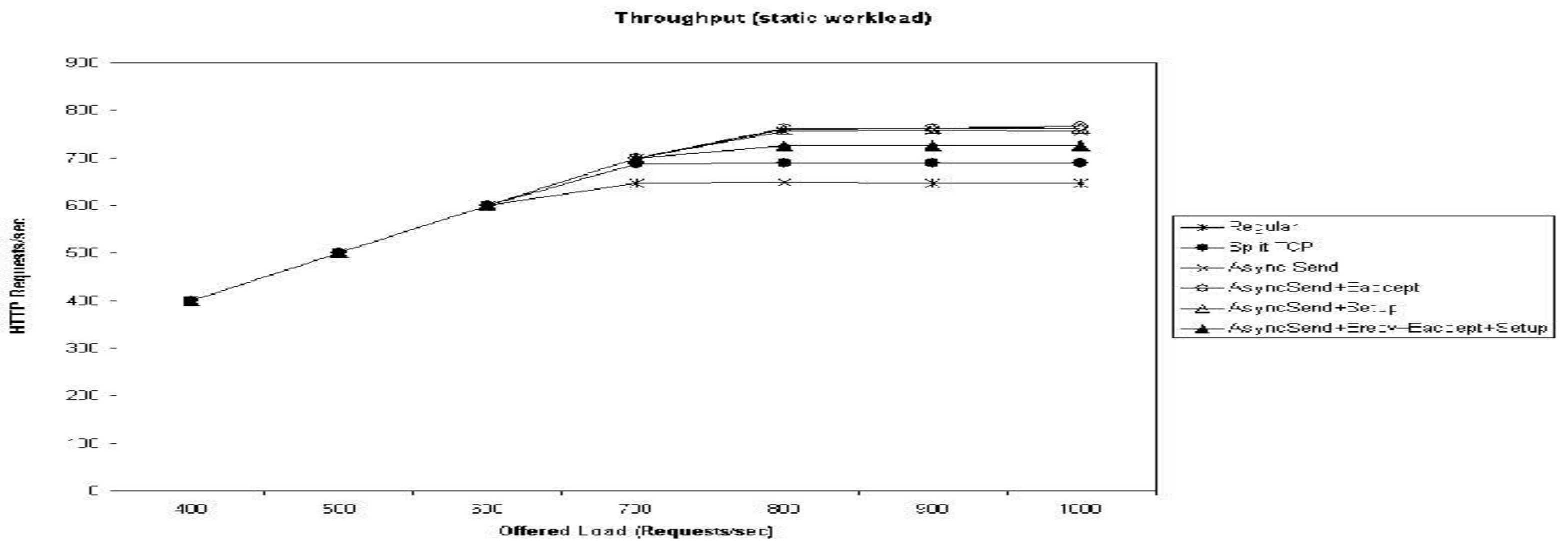
- When only one processor is dedicated to the network processing, the network processor becomes a bottleneck and, consequently, the application processor suffers from idle time.
- When we apply two processors to the handling of the network overhead, there is enough network processing capacity and the application processor becomes the bottleneck.
- The best system would be one in which the division of labor between the network and application processors is more flexible, allowing for some measure of load balancing.

2-node Cluster-based Evaluation for Static Load



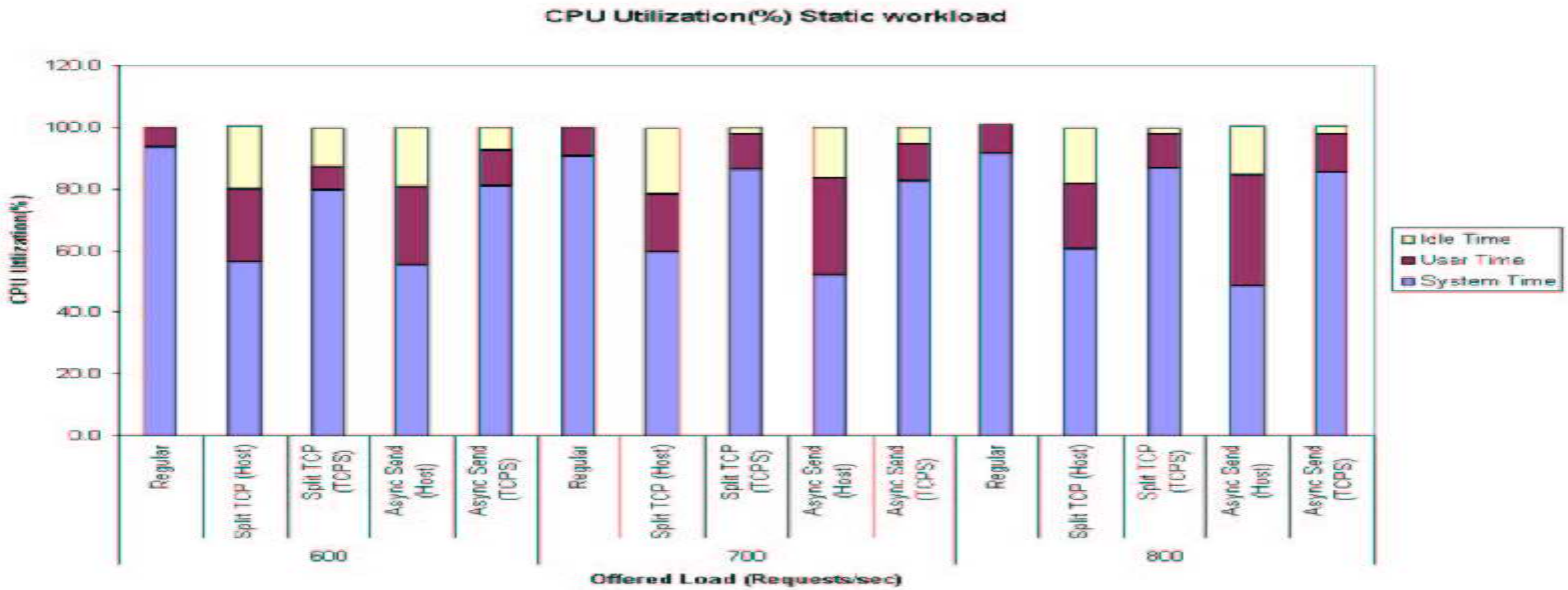
- Asynchronous send operations outperform their counterparts

2-node Cluster-based Evaluation for Static Load



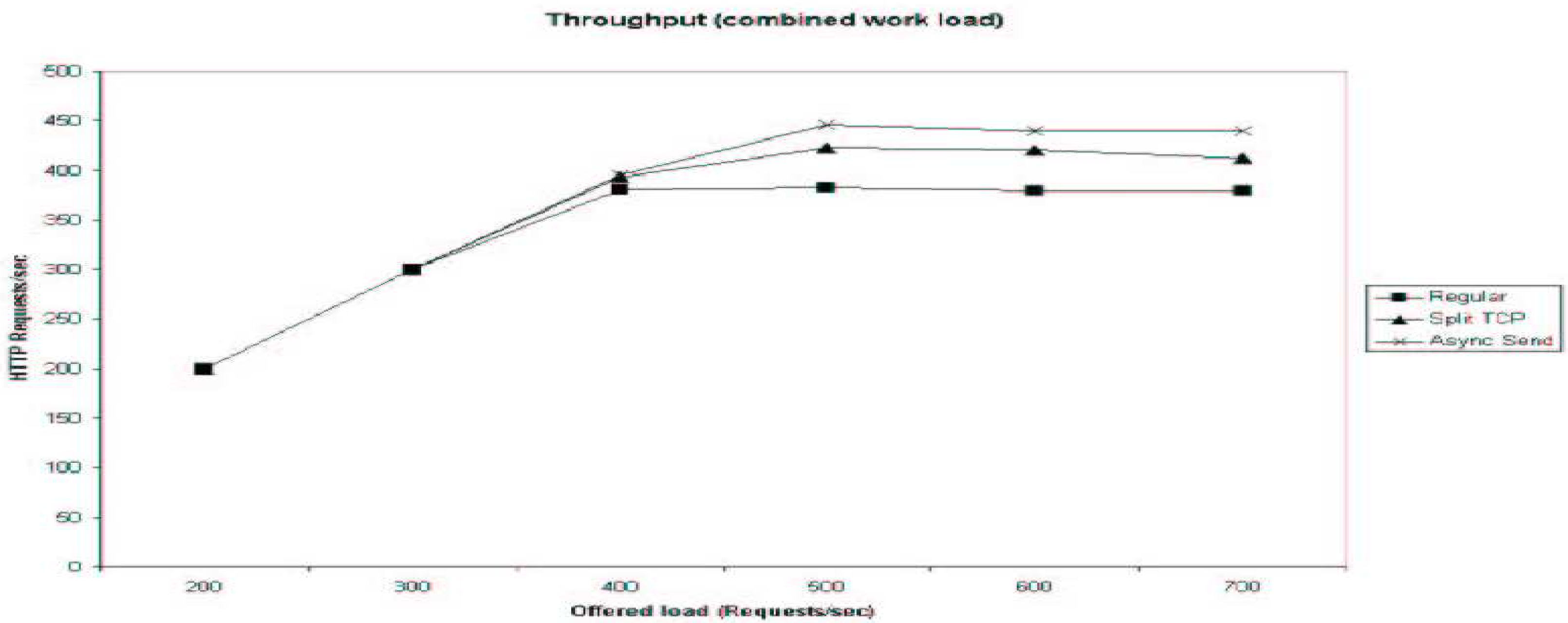
- Smaller gain than that achievable with SMP-based architecture.
- 17% is the greatest throughput improvement we can achieve with this architecture/workload combination.

2-node Cluster-based Evaluation for Static Load



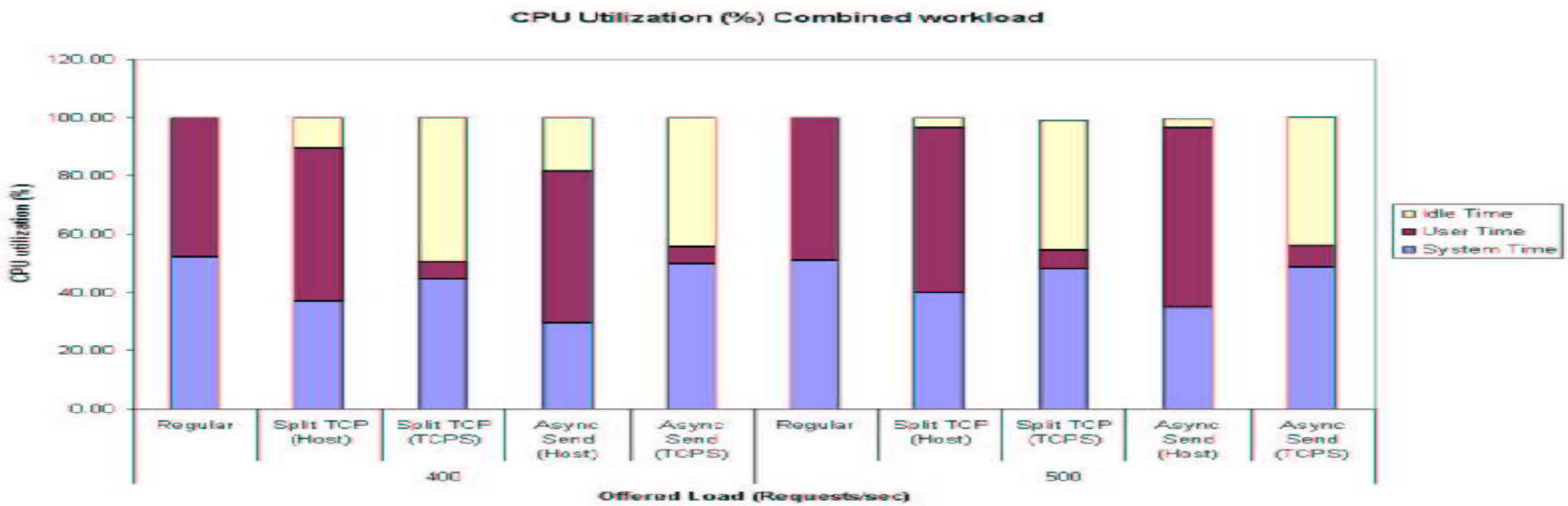
- In the case of Split-TCP and AsyncSend the host has idle time available since it is the network processing at the TCP server that proves to be the bottleneck.

2-node Cluster-based Evaluation for Static and Dynamic Load



- Split TCP and Async Send systems saturate later than Regular TCP.

2-node Cluster-based Evaluation for Static and Dynamic Load

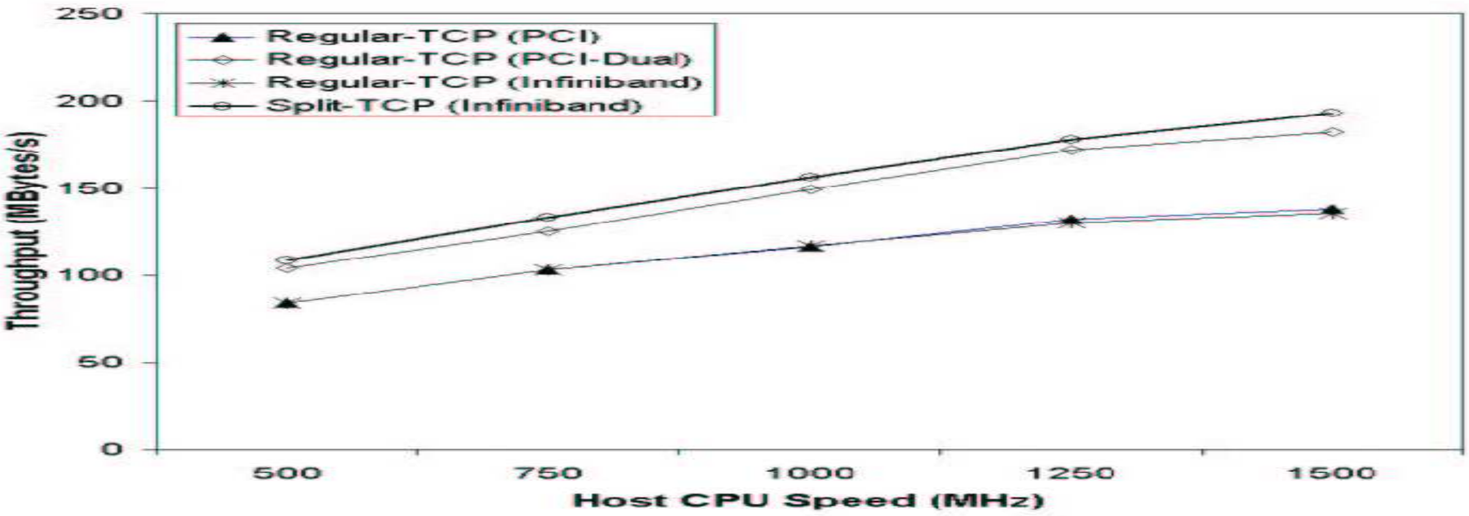


- At an offered load of about 500 reqs/sec, the host CPU is effectively saturated.
- 18% is the greatest throughput improvement we can achieve with this architecture.

2-node Cluster-based Evaluation for Static and Dynamic Load

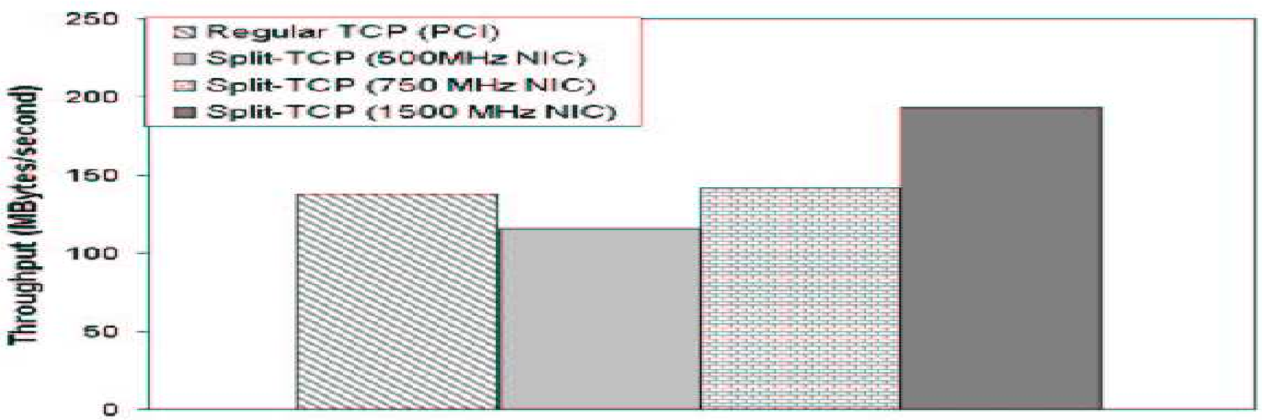
- Balanced confgurations depend heavily on the particular characteristics of the workload.
- A dynamic load balancing scheme between host and TCP server nodes is required for ideal performance in dynamic workloads

Intelligent-NIC-based Simulation Evaluation



- For all the simulated processor speeds, the Split-TCP system outperforms all the other implementations.
- The improvements over a conventional system range from 20% to 45%.

Intelligent-NIC-based Simulation Evaluation



- The ratio of processing power at the host to that available at the NIC plays an important role in determining the server performance.
- In Split-TCP the processor on the NIC saturates much earlier than the host processor or the network.

Conclusions about TCP Servers 1/2

- Offloading TCP/IP processing is beneficial to overall system performance when the server is overloaded.
- An SMP-based approach to TCP servers is more efficient than a cluster-based one.
- The benefits of SMP and cluster-based TCP servers reach 30% in the scenarios we studied.
- The simulated results show greater gains of up to 45% for a cluster of devices.

Conclusions about TCP Servers 2/2

- TCP servers require substantial computing resources for complete offloading.
- The type of workload plays a significant role in the efficiency of TCP servers.
- Depending on the application workload, either the host processor or the TCP Server can be- come the bottleneck.
- Hence, a scheme to balance the load between the host and the TCP Server would be beneficial for server performance.

Thank you!

Questions/comments?