

# Evaluating Network Processing Efficiency with Processor Partitioning and Asynchronous I/O

Tim Brecht, University of Waterloo

G. (John) Janakiraman, HP

Brian Lynn, HP

Vikram Saletore, Intel

Yoshio Turner, HP

# Problem

- High-bandwidth TCP/IP processing saturates servers
- Why so expensive? [Clark et al, 1989]
  - Interrupts, system calls, data touching,...

# Improvements in Current Systems

- Network Interface Cards (NICs)
  - Interrupt moderation
  - Checksum and segmentation offload
- OS features
  - Interrupt affinity
  - `sendfile(out_fd, in_fd, offset, count)`
    - Zero-copy transmission of file data to network

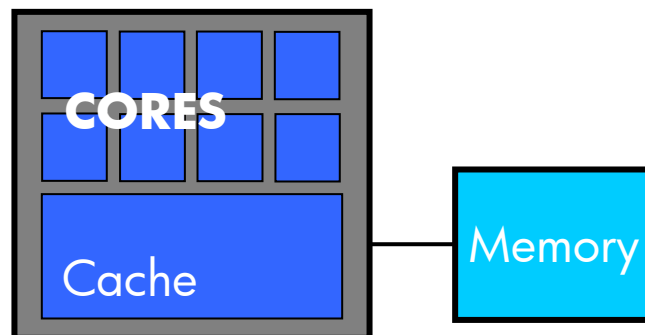
# Other Proposed Approaches

- TCP Offload Engine (TOE) NICs
  - Relieves host CPU and reduces device-OS interaction
  - Network stack in firmware: harder to upgrade and support
  - Resource limitations
  - Mostly for RNICs (TCP RDMA – Remote Direct Memory Access) used for storage and technical computing
- Network stack optimizations [e.g., PrecisionIO]
  - Replace socket buffer linked lists with contiguous buffers used as circular queues
  - Packet processing in user-level library

# Our Investigation

Evaluate a previously proposed approach for using multiple CPUs to improve network processing efficiency

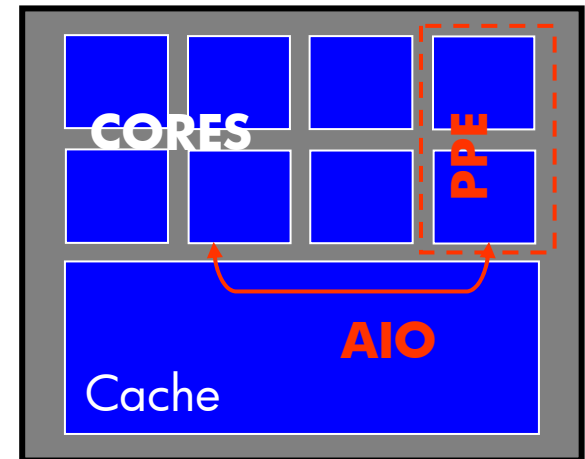
- Multi-core CPUs



- Previous studies showed significant benefits but did not consider recent OS and NIC improvements

# ETA+AIO Architecture

- Embedded Transport Acceleration (ETA):  
Assign some CPUs as dedicated Packet Processing Engines (PPEs)
  - Reduce processor resource contention
  - Eliminate NIC interrupts (PPE polling)
- Asynchronous I/O (AIO):  
“Request and Reply” model for I/O
  - Overlapped application and packet processing
  - Ideal for Single-Process-Event-Driven (SPED) application model
  - Application CPUs and PPEs communicate through shared memory without locks



# Contributions

- Prototype implementation of ETA+AIO architecture
  - Extended previous kernel-level implementation
- Performance evaluation
  - Web server workload
  - Comparison against best-practices system (baseline)
- Case study using AIO
  - Extended open source userver (SPED-based web server application) to use AIO
  - Informs the use of recent Opengroup Extended Sockets
  - See paper for details

# Host CPUs

## User Applications

### Direct User Adaptation Layer Provider Library

**Setup Operations:** Open/Close UAL, Bind  
Create/Destroy DTI, Register/Deregister Memory

**AIO Operations:** Connect/Listen/Accept  
Send/Receive/Shutdown

Open UAL/Close UAL  
Reg & Deregister Mem

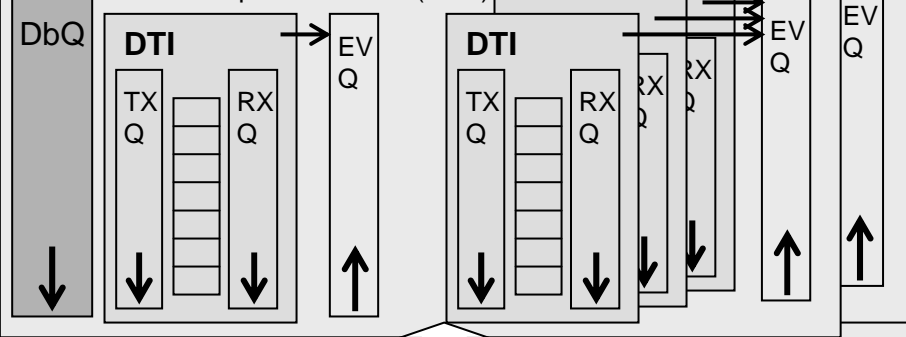
User  
-----  
Kernel

Kernel Agent

Shared  
Memory

Conn./Listen/Accept/Send/Recv  
User Adaptation Layer (UAL)

Direct Transport Interfaces (DTIs)



PPE  
CPU

PPE Kernel Agent Interface

PPE User Adaptation Layer Interface

TCP/IP Stack with  
Modified Interfaces

*PPE Polling  
Thread(s)*

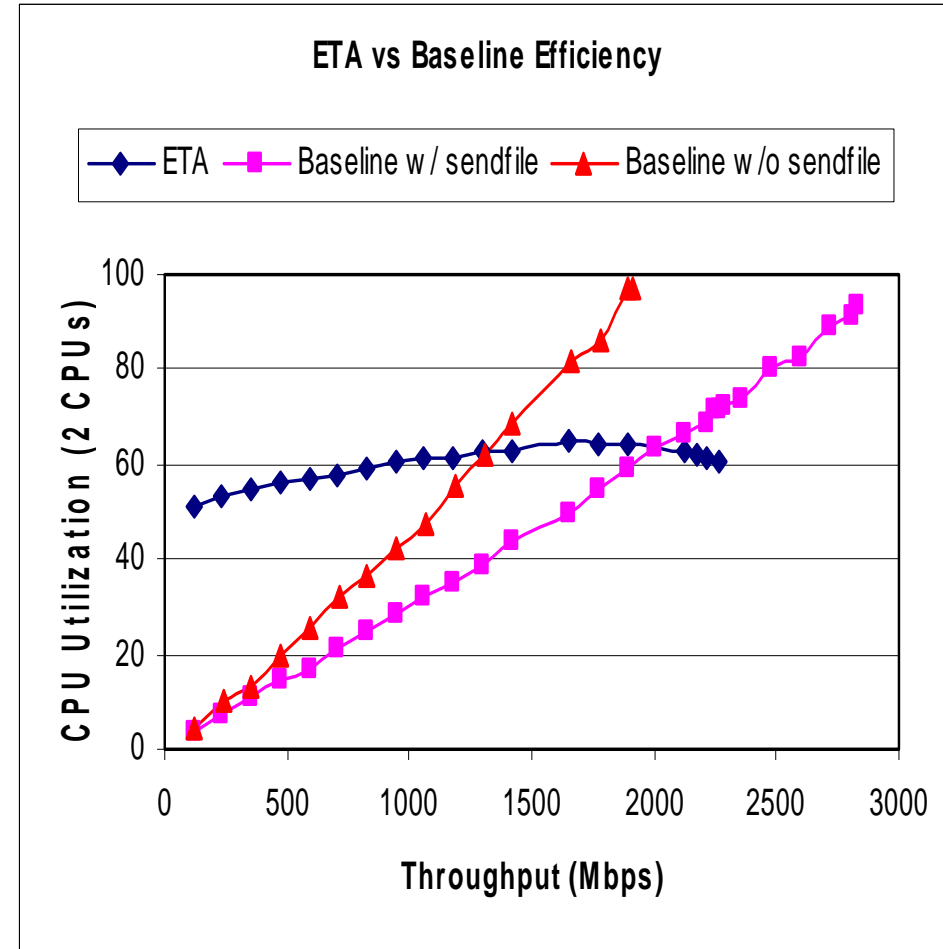
NIC Driver Modified for Polling Rx & Tx Packets

# Performance Evaluation Setup

- “SPECweb99-like” Workload
  - Web server application (userver):  
AIO vs. standard sockets in non-blocking mode
  - Static content only (file set fits in memory [51MB])
  - Clients: open-loop httpperf
- Server: 2-way ProLiant DL580
  - Intel Xeon CPUs (2.8GHz, L2 512KB, L3 2MB, HT disabled)
  - Intel PRO/1000 GigE NICs (MT dual-ported server NICs)
- Clients: HP Integrity rx2600 (per interface)
- Best-practice Linux configuration:
  - Kernel settings (socket memory, max file descriptors)
  - CPU affinity (application and interrupts)
  - Checksum offloading, interrupt moderation

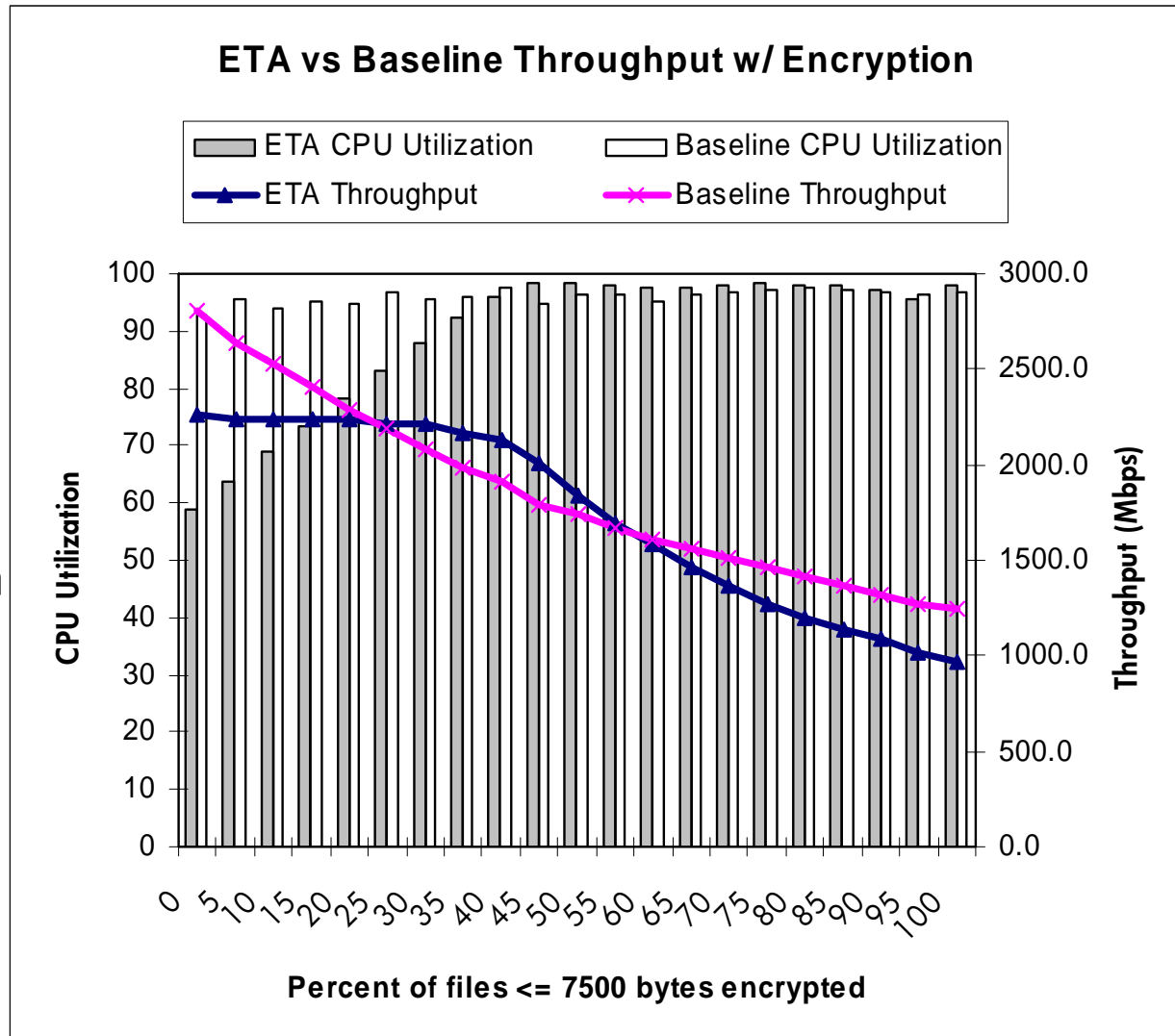
# Processing Efficiency and Peak Throughput

- ETA+AIO efficiency advantage
- ETA static CPU allocation limits peak throughput



# Impact of Application Load

- Encrypt some replies
  - App CPU cycles
  - Data touching
- Throughput advantage when workload mix matches CPU allocation
- Further motivation for dynamic allocation



# Detailed Analysis

- ETA prototype uses only one PPE CPU: efficiency similar to artificially scaled-up uniprocessor baseline
- CPU cycle breakdown near ETA max throughput

	<b>ETA+AIO</b>	<b>2-CPU Baseline</b>	<b>Scaled 1-CPU Baseline</b>
<b>Idle</b>	39.24%	29.30%	40.08%
<b>Userver</b>	6.89%	5.48%	5.32%
<b>Kernel</b>	8.31%	16.40%	12.09%
<b>TCP/IP</b>	25.02%	29.51%	25.54%
<b>Ethernet</b>	15.01%	19.19%	16.82%
<b>PPE Other</b>	5.43%		

- ETA reduces cost of TCP/IP (partitioning) and Ethernet (polling and partitioning)
- AIO reduces Kernel costs, adds DTI&event queue processing in both the PPE (2.7% productive) and user-level (userver)
  - ~2% improvement over scaled 1-CPU baseline

# Detailed Analysis: Cache Behavior

- No encryption:  
CPU partitioning increases L3 misses (Rx Copies and Host-PPE DTI)
- Encryption:  
CPU partitioning reduces cache contention with heavier application workload

## OProfile cache miss frequency (normalized to 1-cpu baseline)

**No Encryption (equal  
[normalized] throughput):**

	<b>ETA+ AIO</b>	<b>Baseline 2-CPU<sub>s</sub></b>	<b>Baseline 1-CPU</b>
<b>L3 Misses</b>	1.43	1.34	1.00
<b>L2 Misses</b>	1.01	1.16	1.00

**Encryption (40%, ETA+AIO has  
highest throughput):**

<b>L3 Misses</b>	1.00	1.38	1.00
<b>L2 Misses</b>	0.71	1.13	1.00

# Conclusions

- ETA+AIO is modestly more efficient than baseline
  - Avoiding multiprocessing overheads for network processing
  - Low overhead OS-bypass AIO programming interface
  - Reduced contention between application and network stack
- Peak throughput sensitive to workload mix:  
static allocation of resources
- Partitioning has two opposing impacts on cache behavior:  
shared vs. unshared data
- Further optimization and investigation
  - Signals
  - Dynamic CPU allocation
  - Benefits with multi-core processors (e.g., with shared L3 cache providing fast inter-core communication for DTIs)