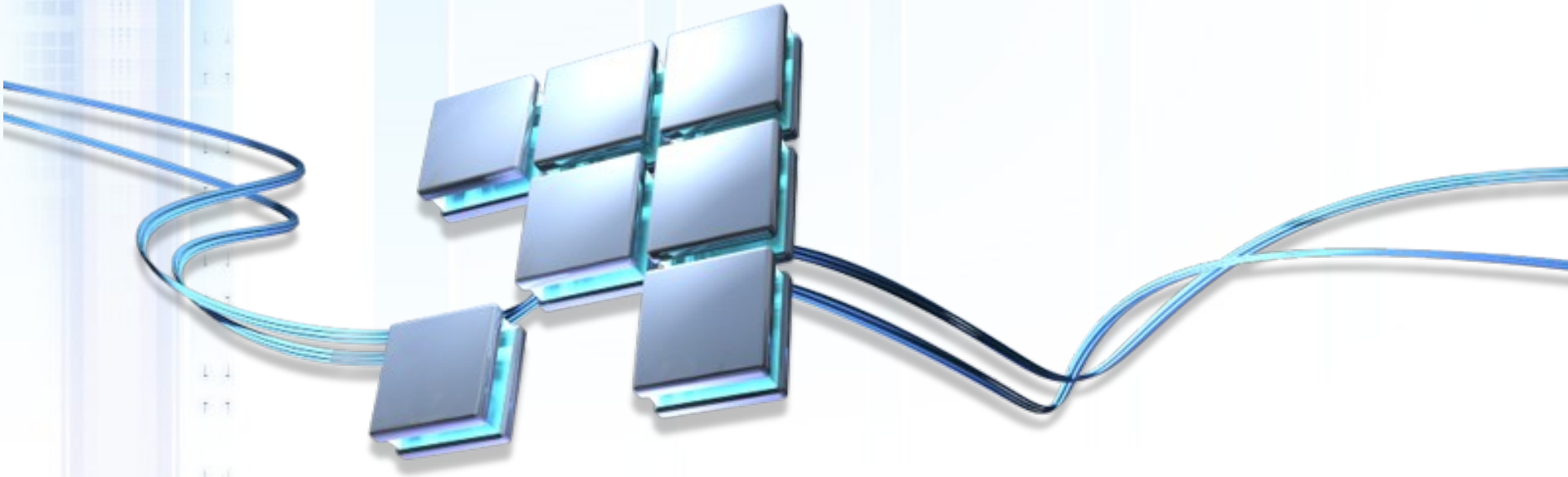
Accelerating Web Protocols Using RDMA

Dennis Dalessandro
Ohio Supercomputer Center

NCA 2007



Who's Responsible for this?

- Dennis Dalessandro
  - Ohio Supercomputer Center - Springfield
  - dennis@osc.edu

- Pete Wyckoff
  - Ohio Supercomputer Center - Columbus
  - pw@osc.edu

Note:

Our topic today is Web Servers but applicable to many client-server application domains.

Problem:

- Increase in demand for bandwidth
- Increase in demand for dynamic content
- Increase in number of clients
- How we interact with the web is becoming more and more complex, not simplifying
- Greater reliance on web based applications

Solutions:

- Distributed servers
  - High cost, complicated to maintain
- Get a more powerful server
  - Moore's Law coming to an end?
  - Demand means more upgrades more often
- Reduce availability
  - Not very likely!
- Offload network processing
  - Substantially less expensive
  - Works in HPC!

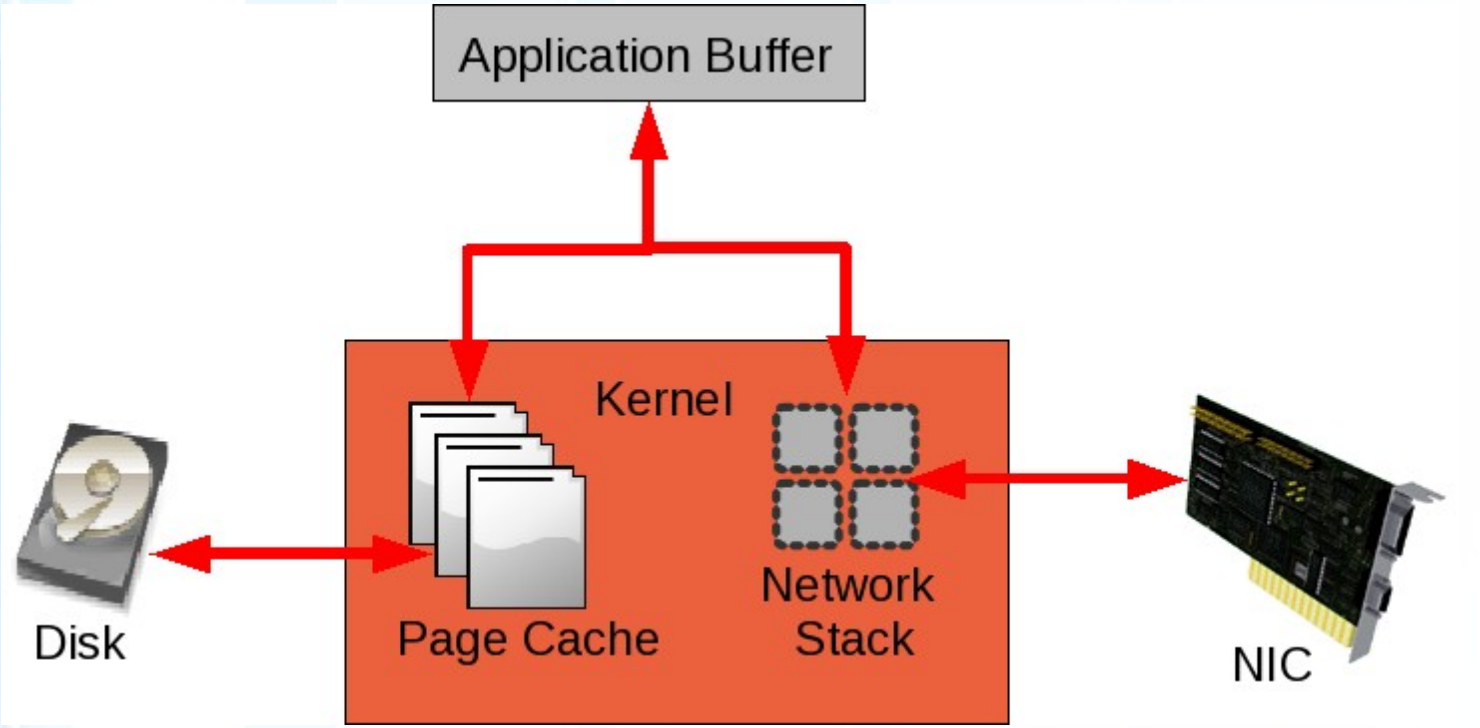
Network Processing Load

- Takes CPU power to generate content
- Takes CPU power to handle network processing
  - Multiple copies of data needed to get onto wire
- If CPU busy doing network processing it can not generate or retrieve content
- The opposite is true as well
- Naturally problem gets worse with more clients

Protocol Offload

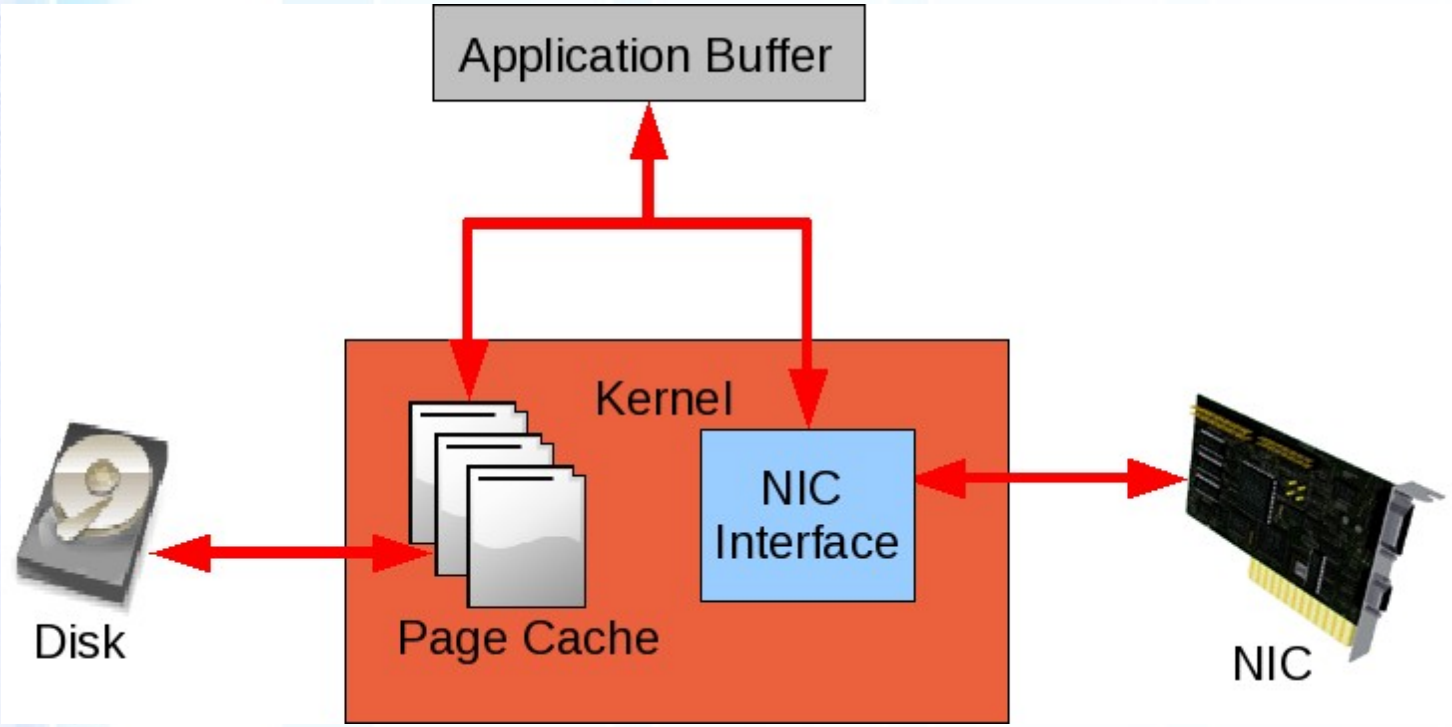
- NIC handles network related processing
- Removes biggest burden from the CPU
- Two common cases:
  - TCP Offload Engine (***TOE***)
    - CPU still has to move data to/from NIC
    - Leads to memory bottleneck at CPU
    - Partial solution
  - Remote Direct Memory Access (***RDMA***)
    - NIC is able to DMA data to/from memory
    - CPU not involved at all
    - Long used in HPC (InfiniBand, Myrinet, etc)

Normal (TCP) Network Processing



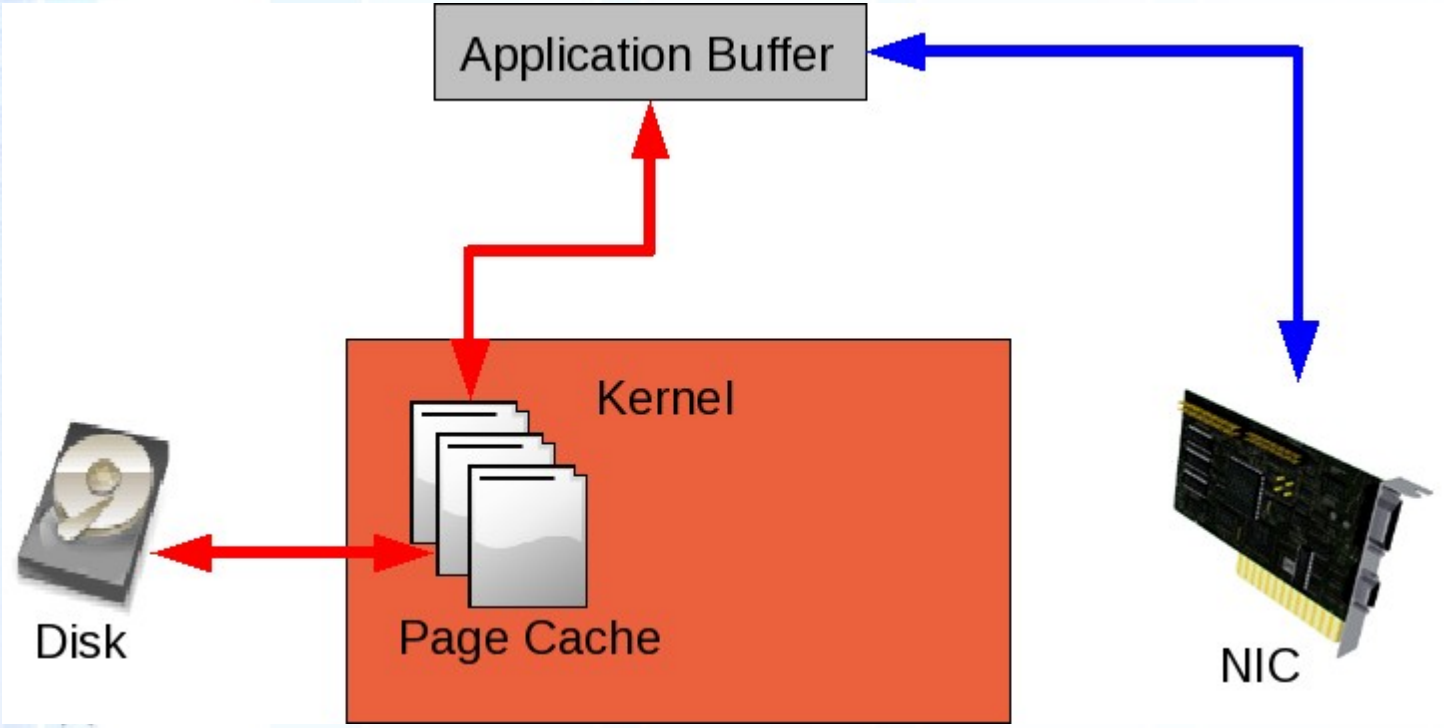
CPU Does Network Processing and Moves Data

TOE Network Processing



NIC Does Network Processing CPU Moves Data

RDMA Network Processing



NIC Does Processing And Moves Data

How can this work for Web Servers?

- iWARP is RDMA over ordinary TCP/IP
- Only server needs upgrade
  - Clients can communicate with software iWARP
    - via browser plugin or application mods
    - Makes for easy integration/adoption
  - iWARP card much cheaper than new server
- Changes to HTTP?
  - HTTP is simply application data
  - Only change needed is extending headers
  - Fully backwards compatible

HTTP Header:

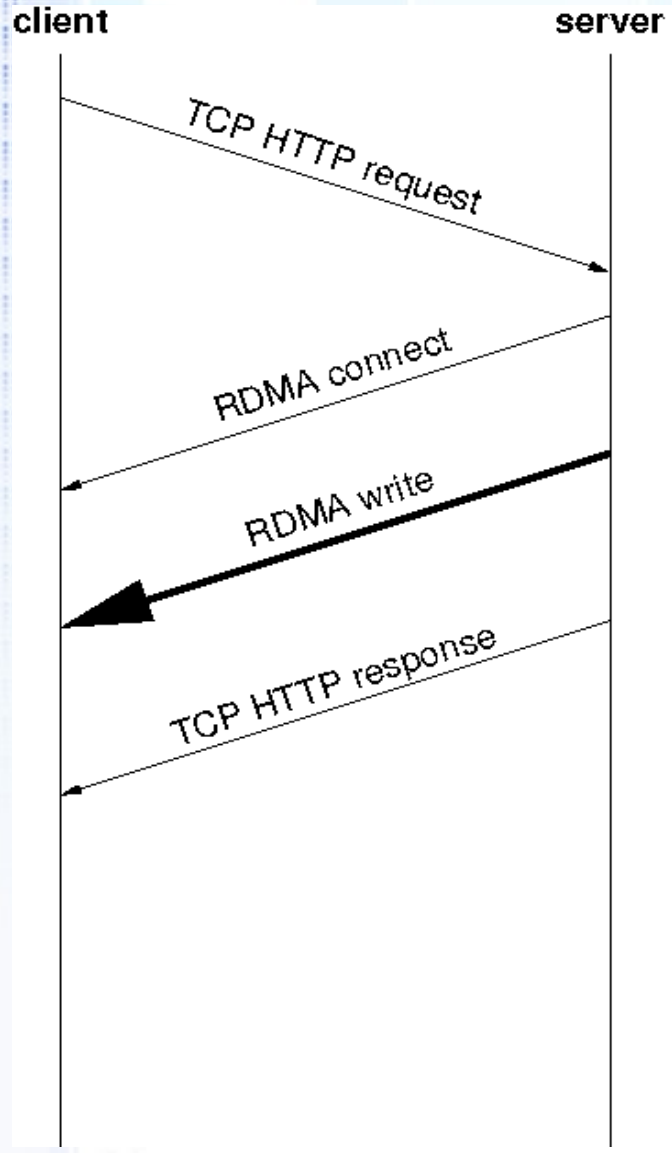
```
GET /index.html HTTP/1.1
Host: www.osc.edu
User-Agent: Mozilla/5.0
Connection: Keep-Alive
```

HTTP Header:

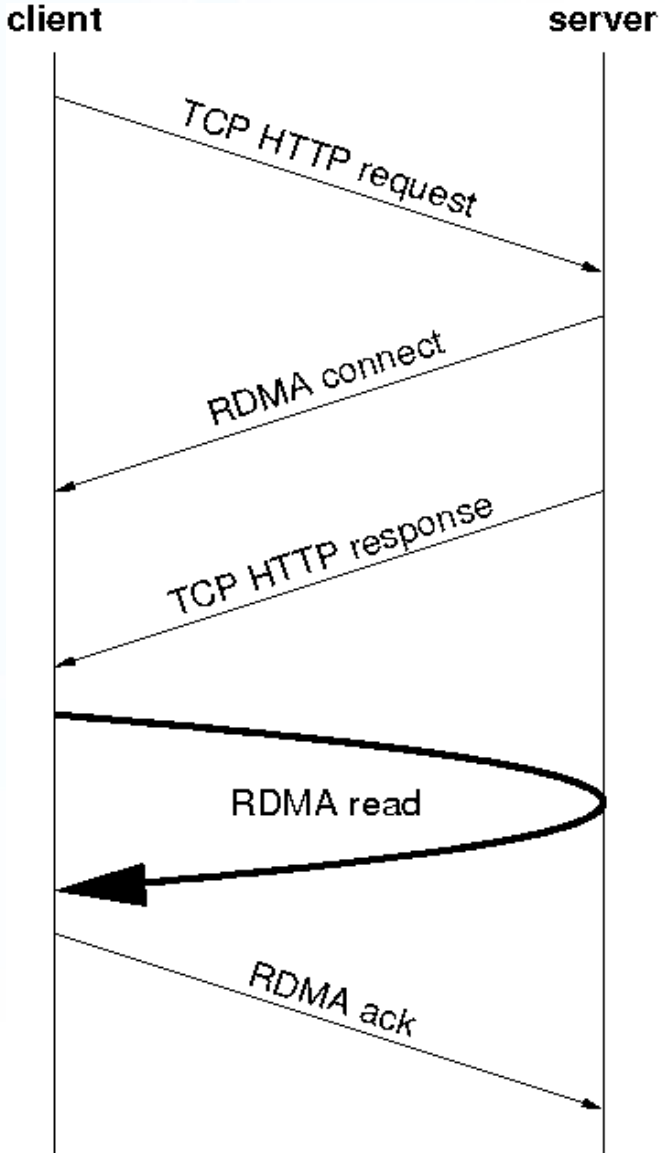
```
GET /index.html HTTP/1.1
Host: www.osc.edu
User-Agent: Mozilla/5.0
Connection: Keep-Alive
RDMA: server-writes, ip=10.0.0.15, port=3242,
        stag=642, to=0, maxlen=1048576
```

Get Request:

Server Writes



Client Reads



POST Request

- Similar to GET but:
  - Client Writes
  - Server Reads
- Not yet implemented
  - Planned for future work

RDMA Connection Issue

- Server establishes RDMA connection to client
  - Costly, especially in high latency environments
  - Ordinary TCP connection
- Why?
  - Need to transition ordinary TCP connection to iWARP
  - Not facilitated by todays software
    - RDMA connection represented by QP
    - TCP connection represented by FD
    - QP != FD (currently)

Memory Registration

- Two Methods
  - Static
  - Dynamic
- Necessary for RDMA
  - Ensure data stays put!
- Costly, proportionate to size
- Two phases:
  - Pin physical pages
    - Involves walking page tables
  - Inform adapter of physical address
    - Costly virt->phys translation

Static Registration

- Register large chunk outside of critical path
- In Apache: per client at connection time
  - Multiple transfers can reuse buffer
  - Not realistic as number of clients scales
- Still have cost to get file to user buffer
  - Results in the need for a **memcpy()**

Dynamic Registration

- Register buffer for each request
  - Very costly, proportionate to size
- Adds cost of deregistration
  - Constant cost, not a big deal
- Eliminates the memory copy
- The realistic approach, scales as clients

Lessons Learned

- Low CPU Utilization
  - Dynamic is best
  - Registration faster than **memcpy()**
- High CPU Utilization
  - Static is best
  - **memcpy()** faster than registration
- Reason: Registration is extremely CPU intensive

Implementation

- Does not modify Apache code
  - Why mess with a good thing?
- Get all the benefits of Apache for free
  - Efficient process management
  - Dynamic content generation (PHP/CGI)
- Makes use of hook infrastructure
- Resulting module known as **mod_rdma**

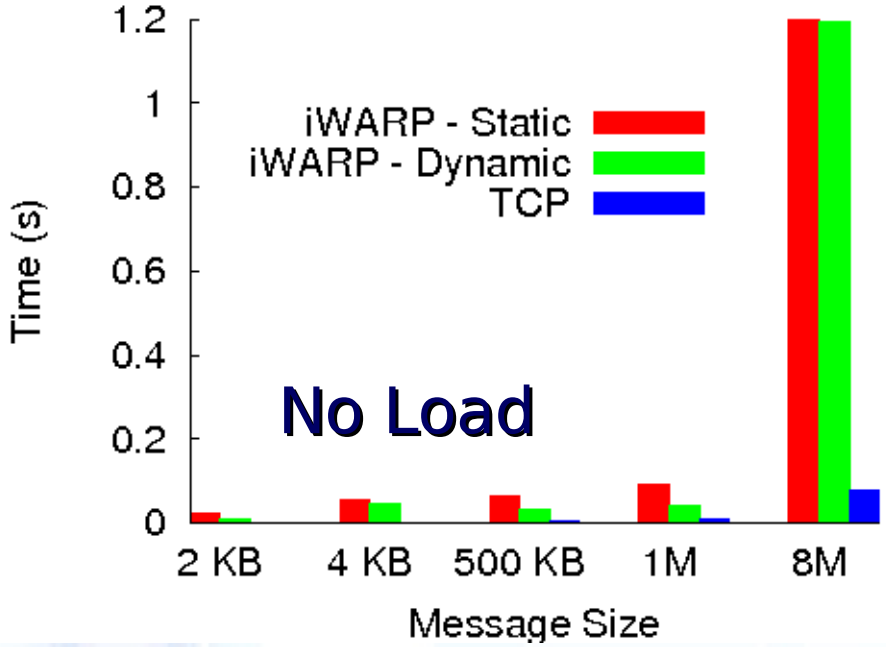
Hooks

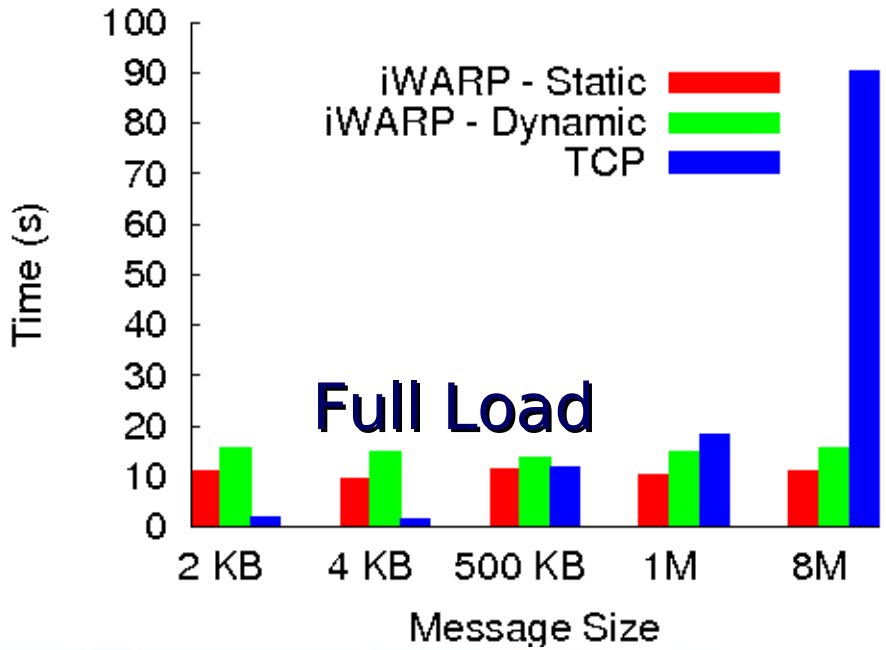
- **Child Init:**
  - Open and init dev
  - Once per proc
- **Pre-connection:**
  - Reg term handler
  - Once per TCP conn
- **Insert Filter:**
  - Make RDMA conn
  - Attach output filter
  - Each request
- **Output Filter:**
  - Do RDMA op
  - Pass on TCP Hdrs

Performance Analysis

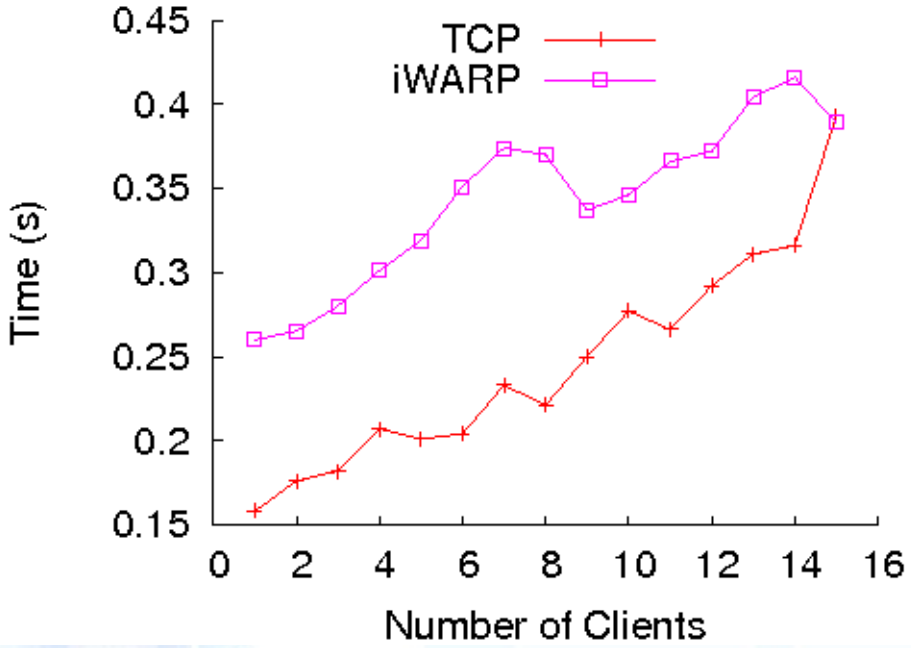
- Server outfitted with hardware iWARP
  - NE010 10 Gigabit iWARP Adapter (NetEffect)
  - Connected to Cisco 6506 switch
  - Apache with mod_rdma
- Clients equipped with 1 Gigabit Tigon3
  - Connected to Cisco 6506 switch
  - wget with linked in software iWARP
  - Only have 15 clients (switch capacity)
- To simulate heavy load use **cpu_eater**
  - lots of trig calculations
  - some 'nice' magic
  - results in nearly 100% CPU usage at all times

Single File Retrieval

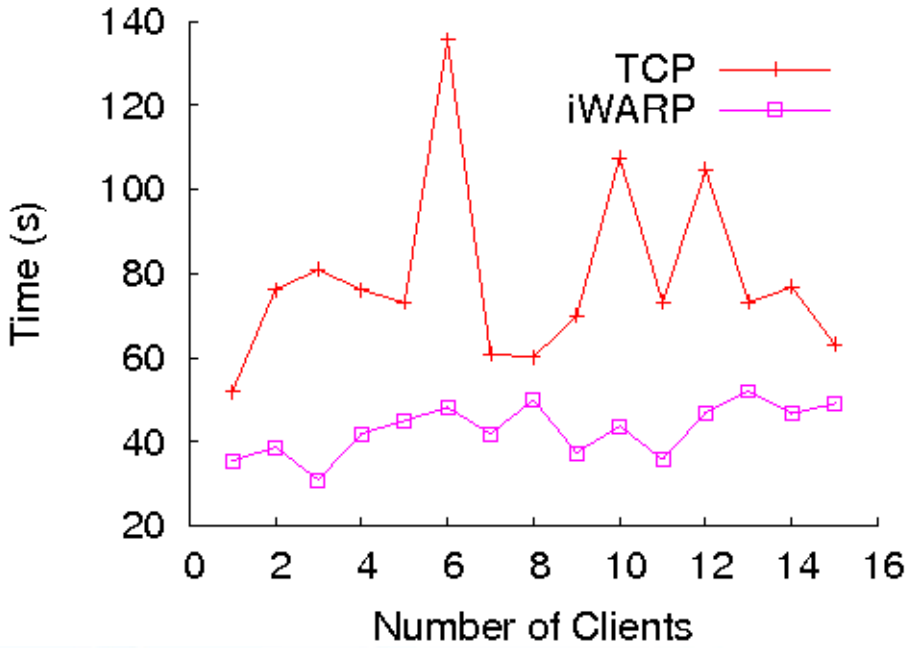




Multiple Clients



No Load

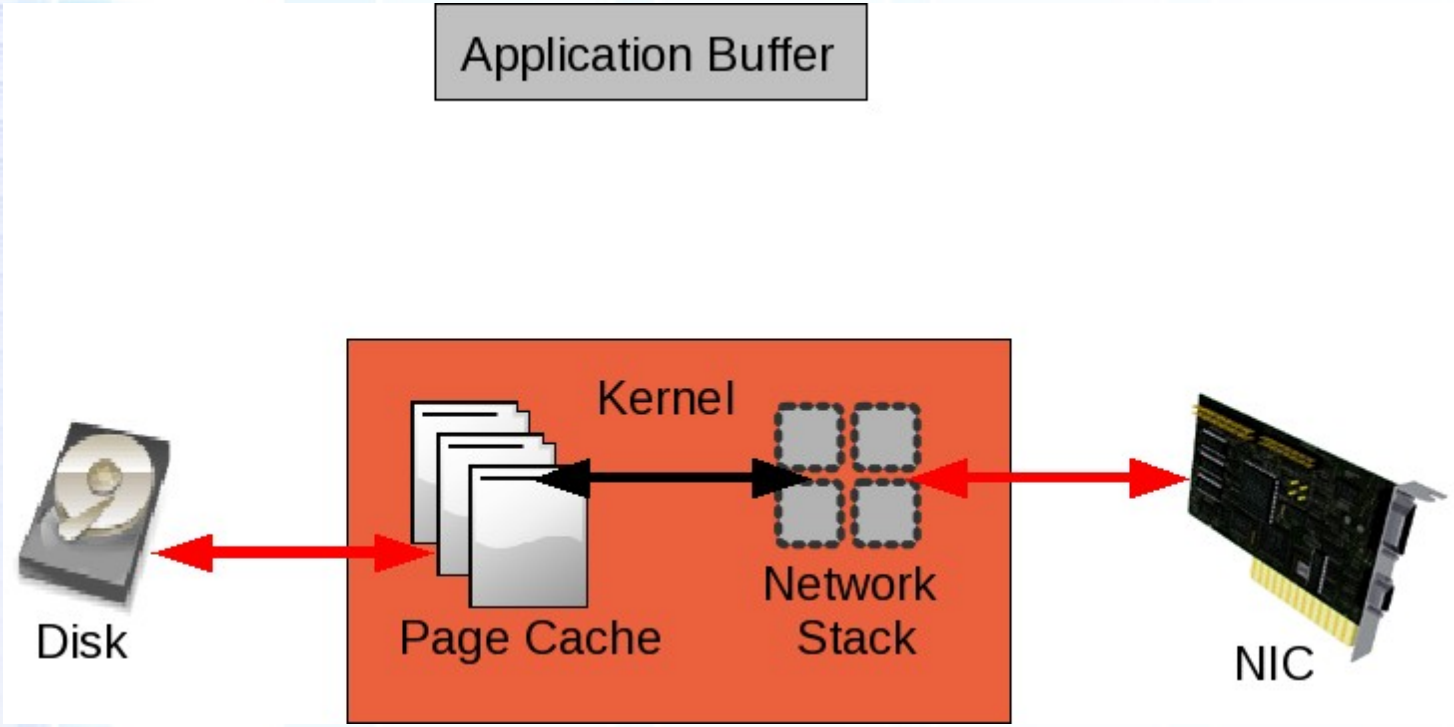


Full Load

Performance Improvement OK

- Expected much bigger benefit for iWARP
  - So Did We!
- Definite improvement under heavy load
- Definite improvement for large transfers
- Two costs to amortize to see benefits
  - Cost of RDMA connection
  - Cost of memory registration
- Something more fundamental at work here

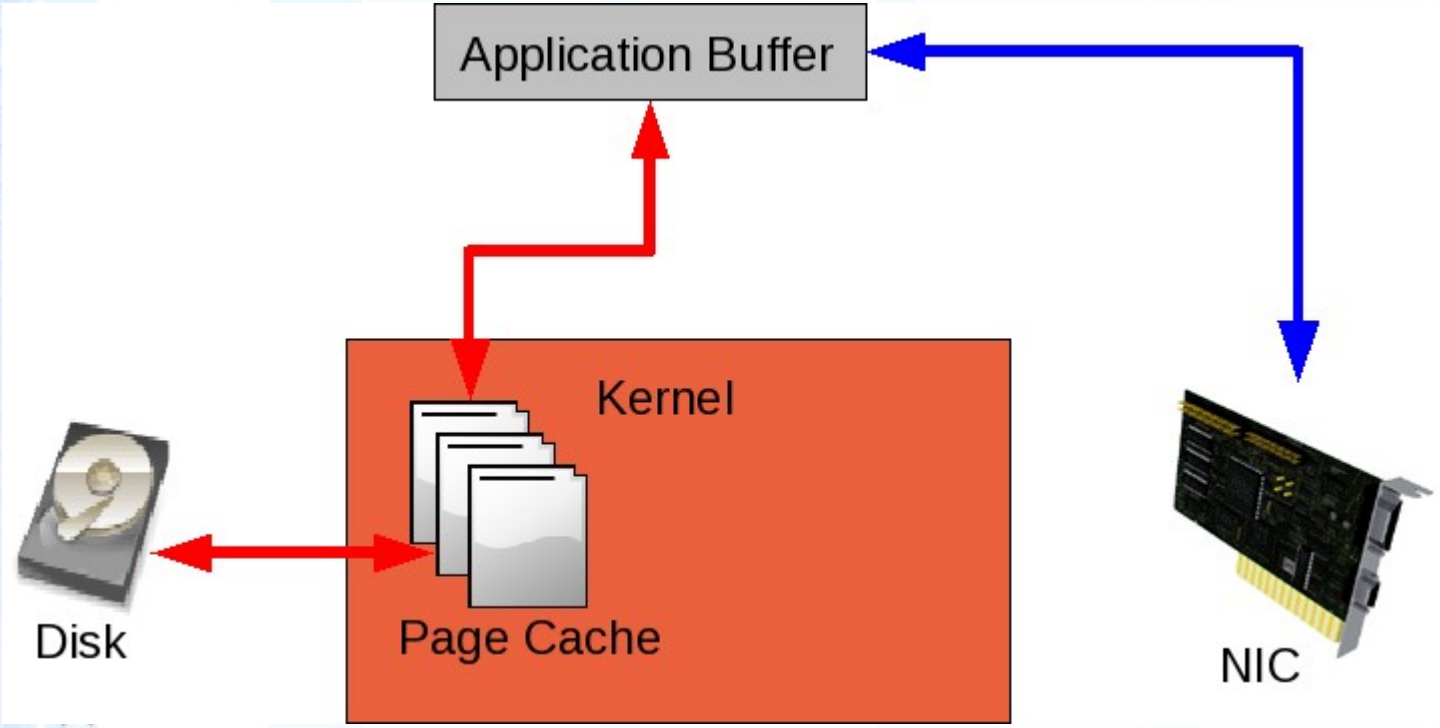
Apache Uses Sendfile



Zero-Copy way to send data direct from page cache

TCP/IP stack still processed on CPU of course

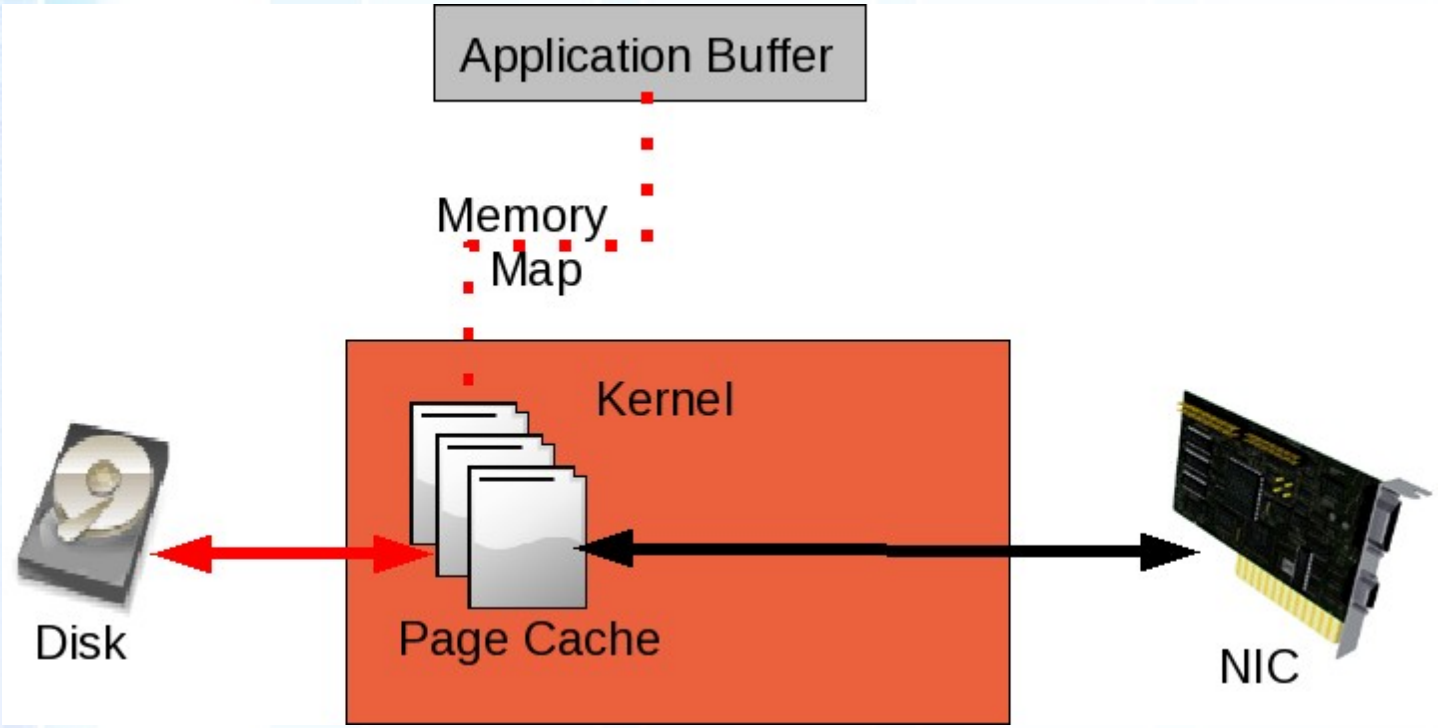
Recall RDMA – so much for Z-Copy



To get file data into buffer to register adds a copy!

RDMA APIs do not always map to sockets based applications

What about Memory Map?



Removes the copy but adds very costly virt->phys translation

Solution

- RDMA sendfile
  - Solves exactly this problem
- Use a kernel module to register memory
  - User code asks kernel to send a file
  - Kernel registers and pins down page cache
    - Avoids costly virtual to physical translation
    - Avoids copying data to user space
  - Kernel returns STag to user and user sends data
    - Kernel and user space can not share QP
    - Complicates things programatically but hidden away with rdma sendfile library

RDMA Sendfile

- Upcoming paper at Hot-Interconnects 07
- Solves problem for sending side only
  - Next step is to work on protocol to cooperate with recv side
- Working on integrating into mod_rdma
  - Will really show performance advantage
- Waiting on iWARP HW rev and source code access to integrate (NetEffect)

Future Work for mod_rdma

- RDMA Sendfile Integration
- Full SSL support
- Moving experiments to WAN
  - OSC Net (10 Gigabit WAN)
- Find suitable production application

Thanks!

- Any questions?
- For more info contact:

Dennis Dalessandro - Ohio Supercomputer Center
dennis@osc.edu

Software iWARP available on the web:
Ugly URL but link on www.osc.edu/~dennis