Advanced Operating Systems: Three Easy Pieces

NUMA and RDMA

Outline

- What is NUMA
- What is RDMA
- NUMA-aware RDMA-based end-2-end Data Transfer System

What is NUMA: Non-Uniform Memory Access

Shared Memory

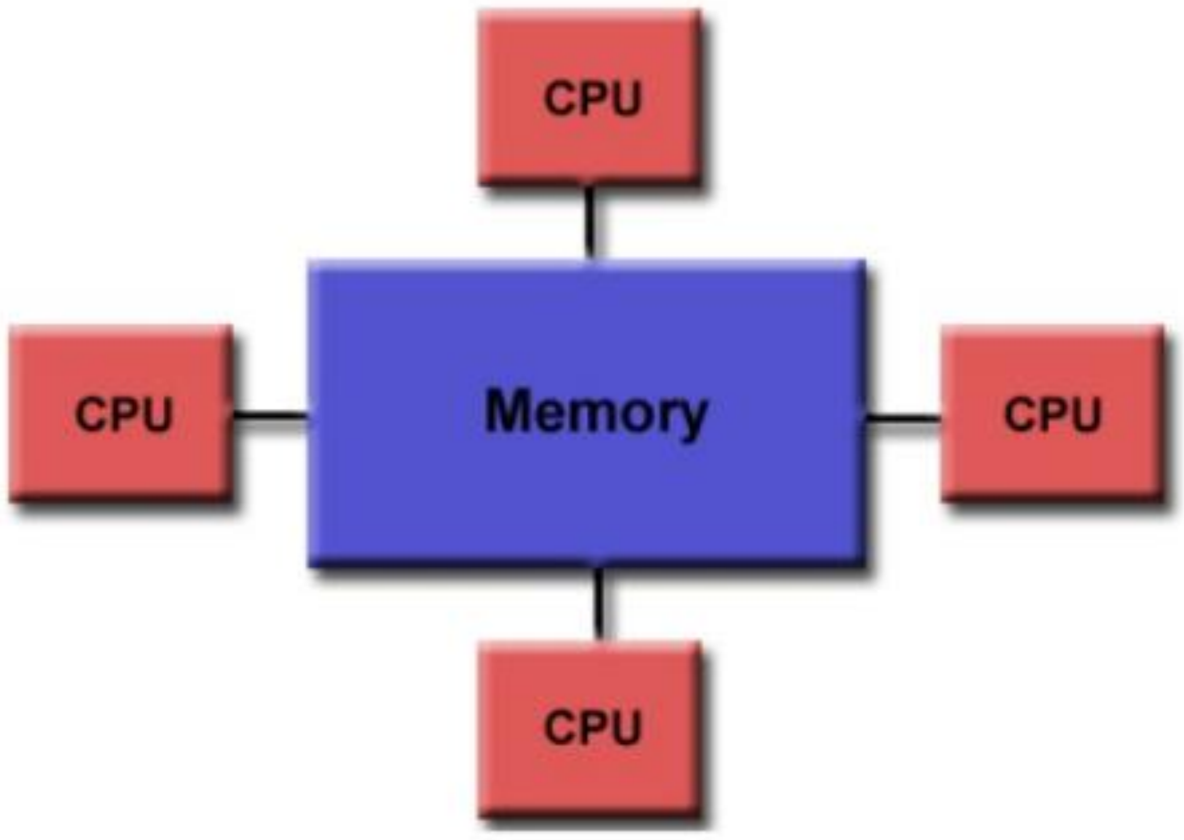
- ❑ Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.

- ❑Multiple processors can operate independently but share the same memory resources.

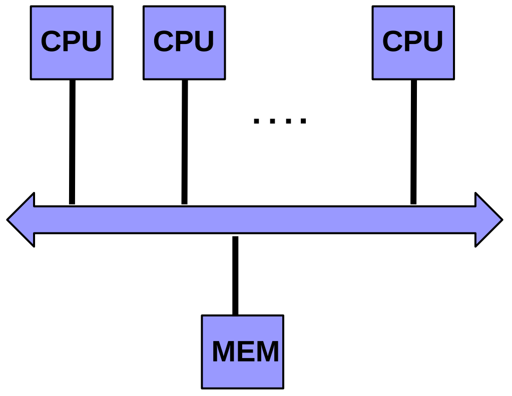
- ❑Changes in a memory location effected by one processor are visible to all other processors.

- ❑Shared memory machines can be divided into two main classes based upon memory access times: ***UMA*** and ***NUMA***.

Shared Memory - UMA



UMA



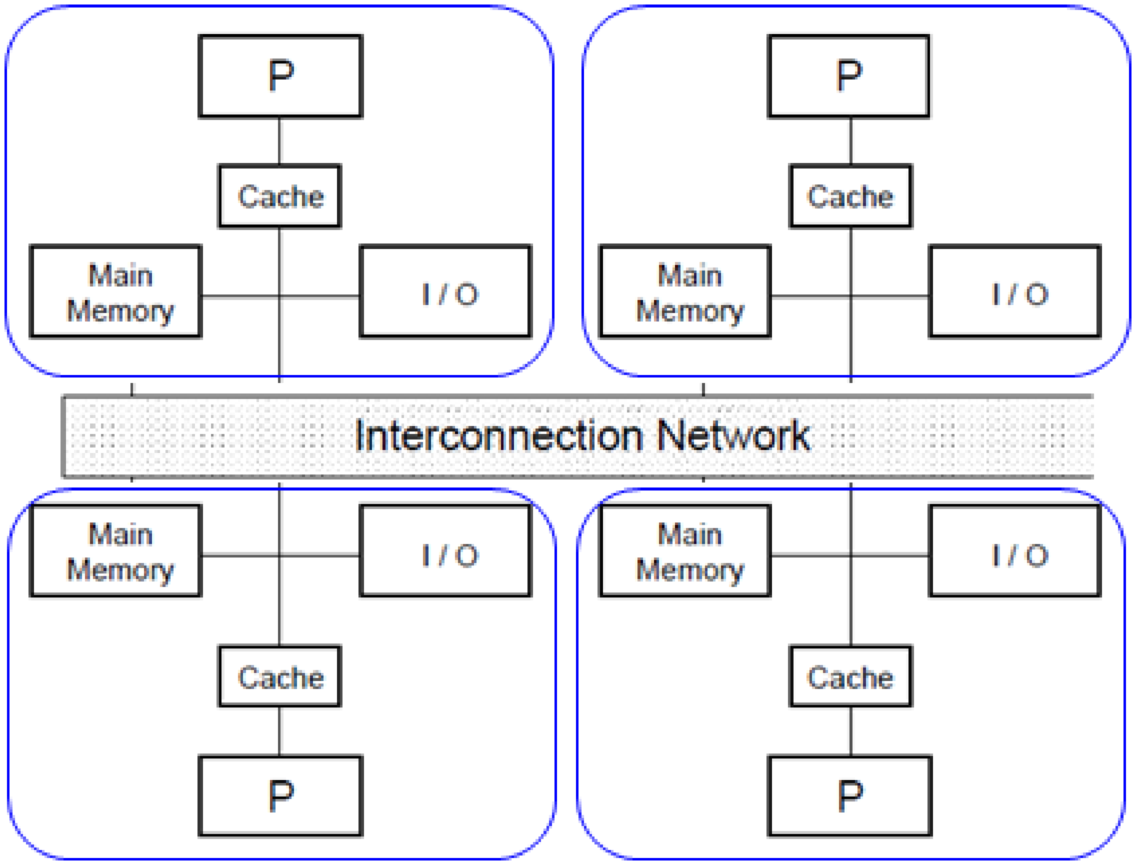
SMP
(UMA)

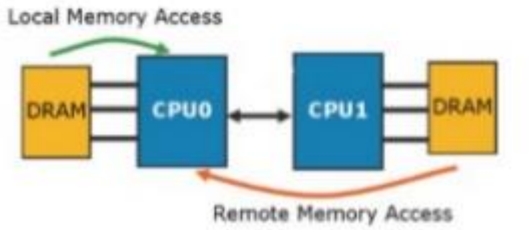
UMA: CPUs are equal distance (cost of access) to any memory location

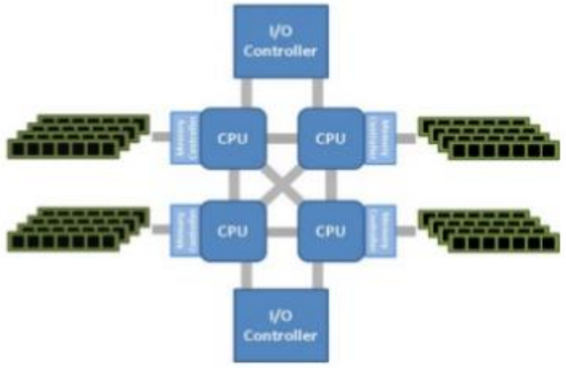
Uniform Memory Access - UMA

- Most commonly represented today by Symmetric Multiprocessor (SMP) machines
- Identical processors
- Equal access and access times to memory
- Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.

Shared Memory - NUMA







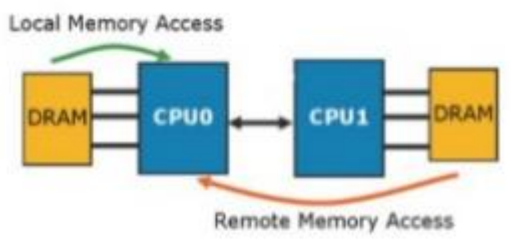
NUMA: CPUs have access to all memory but with access time depending on the target physical memory location

Non-Uniform Memory Access - NUMA

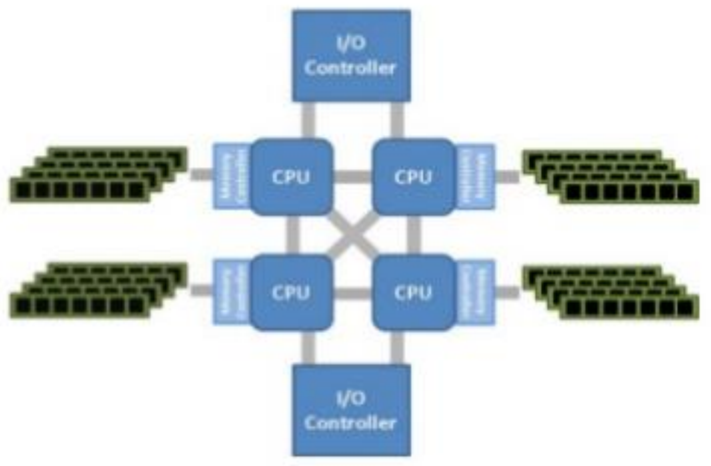
- Often made by physically linking two or more SMPs
- One SMP can directly access memory of another SMP
- Not all processors have equal access time to all memories
- Memory access across link is slower
- If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA

Non-Uniform Memory Access - NUMA

- With NUMA, it will take longer to access some regions of memory than others (i.e., depend if it is local or how remote)
- Designed to improve scalability beyond a large SMP
- A processor can access its own local memory faster than non-local memory



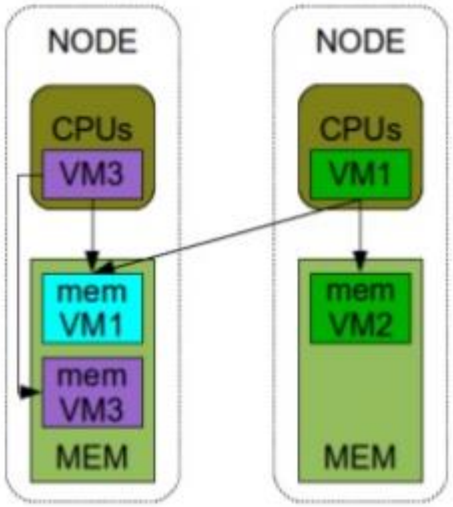
- **NUMA barriers:**
  - IO needs to be considered during placement and scheduling
  - Spread data between memories
  - Cache consistency



Non-Uniform Memory Access - NUMA

- Data request by more than one processor (conflicting requests)

- **Solution:** hardware implementation to solve most of these barriers



NUMA Advantages & Disadvantages

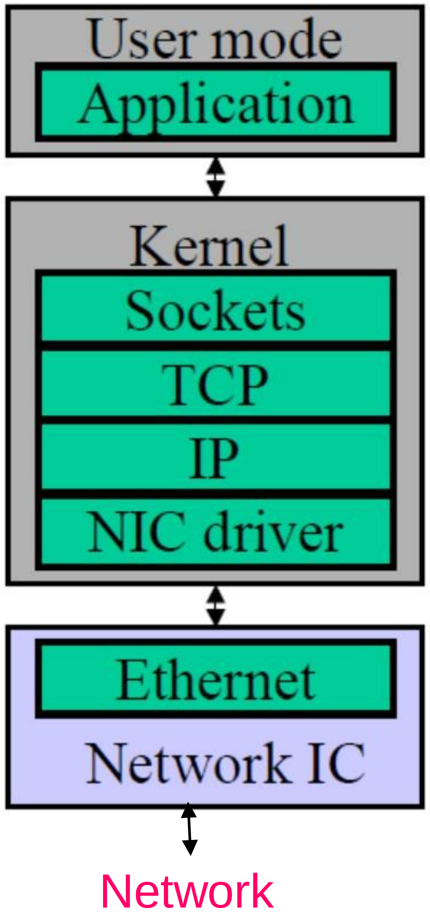
- **Advantages:**
  - Global address space provides a user-friendly programming perspective to memory and scalability beyond SMP
  - Data sharing through memory between tasks is both fast and uniform due to the proximity of memory to CPUs
  - Better horizontal scalability than UMA ↑
- **Disadvantages:**
  - **Primary disadvantage** is the lack of scalability between memory and CPU. Adding more CPUs will increase the traffic on the shared memory-CPU path, and for cache coherent systems, increases traffic associated with cache/memory mgmt. ← expects scalable network
  - **Programmer responsibility for synchronization** is needed to ensure correct access of the global memory – same as UMA
  - **Expense:** it becomes increasingly difficult and expensive to design and produce scalable system with very large number of CPUs: i.e., which memory region to use – local vs remote, etc.

What is Remote Direct memory Access (RDMA)

Network Communication Overview

- **Send**
  - Application buffer → socket buffer (kernel)
  - Attach header
  - Data is pushed to the NIC card buffer
  - Network protocol transmission

- **Receive**
  - NIC buffer → socket buffer (kernel)
  - Parsing header
  - Data is copied into App buffer
  - App is scheduled (context switch)



Network Communication Issues

- **Communication latency**
  - **Processing overhead** and sending/receiving messages overhead
  - **Network latency**
- **Message passing through the kernel**
  - **Low performance:** several memory copies (user1, kernel1, kernel2, user2) + translation between user and device driver
  - **Low flexibility:** protocol processing is inside the kernel
- **User-level Networking:** one of the 1st kernel-bypassing systems – protocol is in user space
- **RDMA**: Modern high-performance networking ↑

U-Net (User-level Networking) Goals

- **Move protocol processing parts** into user space (no context switch and data copy), i.e., user space to NIC memory bypassing the kernel

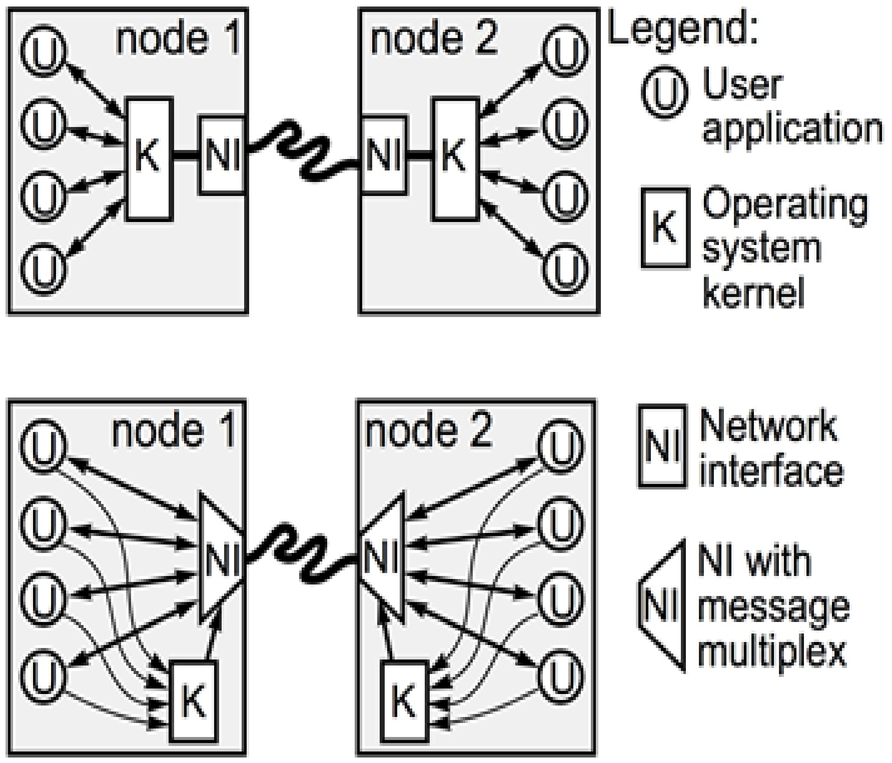
- **Remove the kernel completely** from data communication path

- **Focus on small messages, key goals are**:
  - Low communication latency in LAN setting
  - Exploit full bandwidth
  - Emphasis on protocol design and integration flexibility
  - Portable to off-the-shelf communication hardware

U-Net (User-level Networking) Architecture

- **Traditionally**
  - □□ Kernel controls the network
  - □□ All communications are via the kernel

- **U-Net***
  - □□ Applications can access network directly via Msg MUX
  - □□ Kernel is involved only in connection setup



- **Virtualize NI ➔** provides each process the illusion of owning interface to network

* **U-Net:** http://www.cs.cornell.edu/tve/u-net/

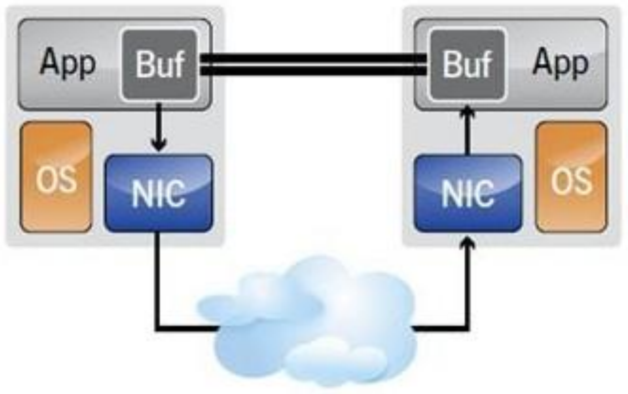
U-Net (User-Level Network Arch) Zero Copy

- **Base-level U-Net** (might not be ‘zero’ copy)
  - Send/receive needs a buffer
  - Requires a copy between application data structures and the buffer in the communication segment (NIC)
  - Can also keep the application data structures in the buffer (communication segment) without requiring any copy

- **Direct Access U-Net (true ‘zero’ copy)**
  - Span the entire process address space
  - But requires special hardware support to check address

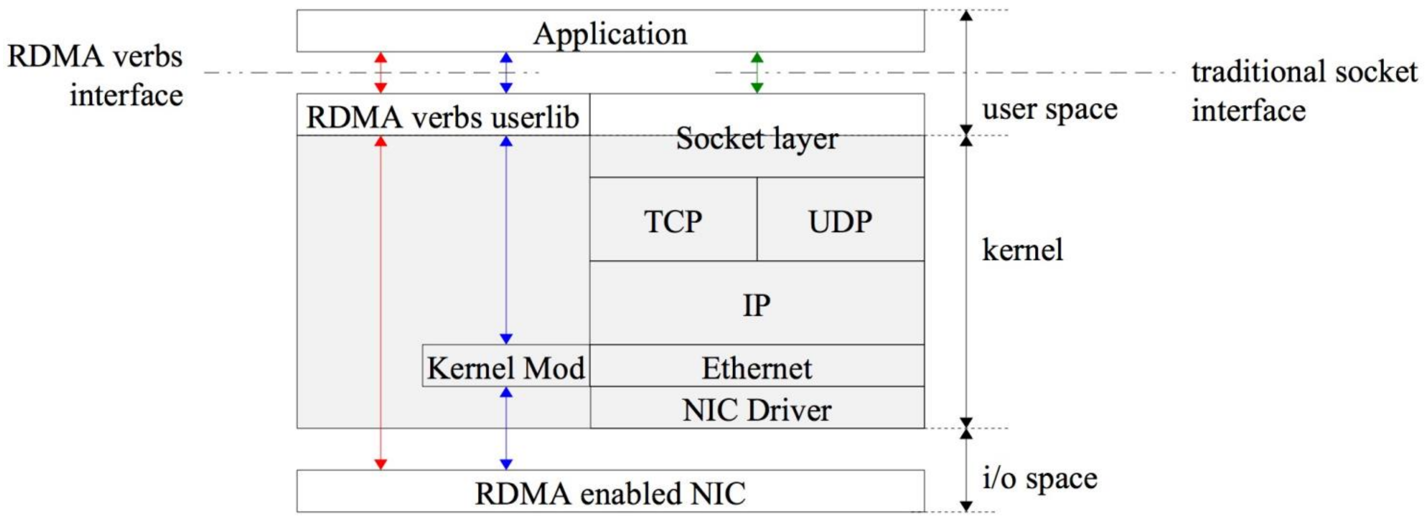
RDMA Goals

- Move buffers between two applications via network w/o kernel involvement



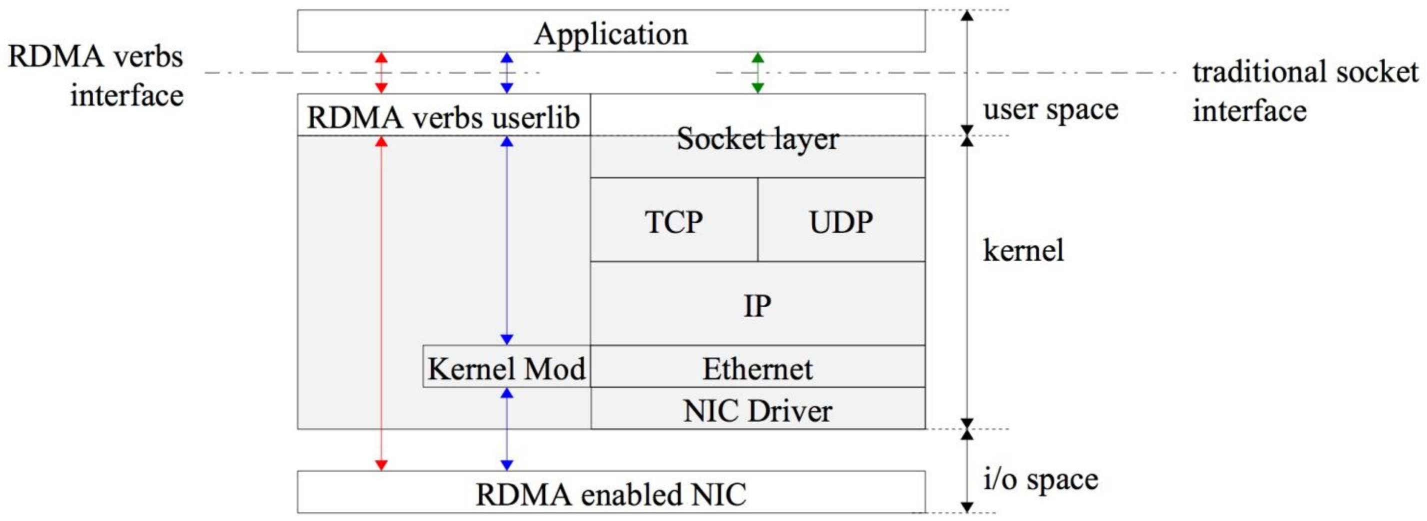
- **Once programs implement RDMA:**
  - Tries to achieve lowest latency and highest throughput
  - Smallest host CPU footprint

RDMA Architecture



- Traditionally, socket interface involves the kernel
- RDMA has a dedicated **verbs interface** instead of the socket interface
- Involves the kernel only on control path / setup time
- Can access RNIC (RDMA enabled NIC) directly from user space on data path bypassing kernel

RDMA Architecture



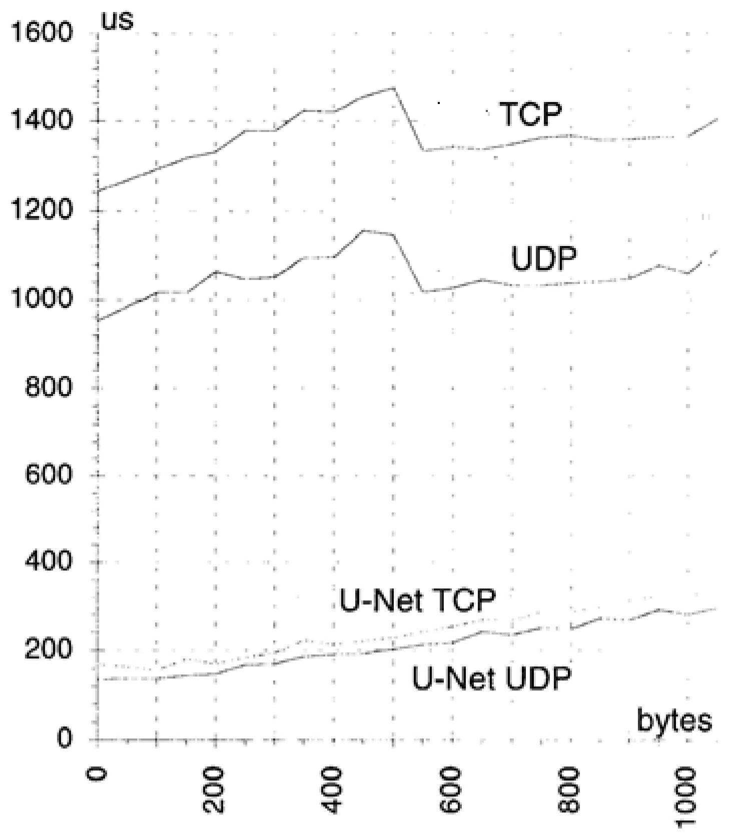
- To initiate RDMA, establish data path from RNIC (RDMA enabled NIC) to application memory
- **RNIC Verbs interface*** provide API to establish these data path
- Once data path is established, directly read-from/write-to buffers (effectively from/to RNIC)
- Verbs interface is different from the traditional socket interface.

* http://www.rdmaconsortium.org/home/RNIC_Verbs_Overview2.pdf

U-Net Performance

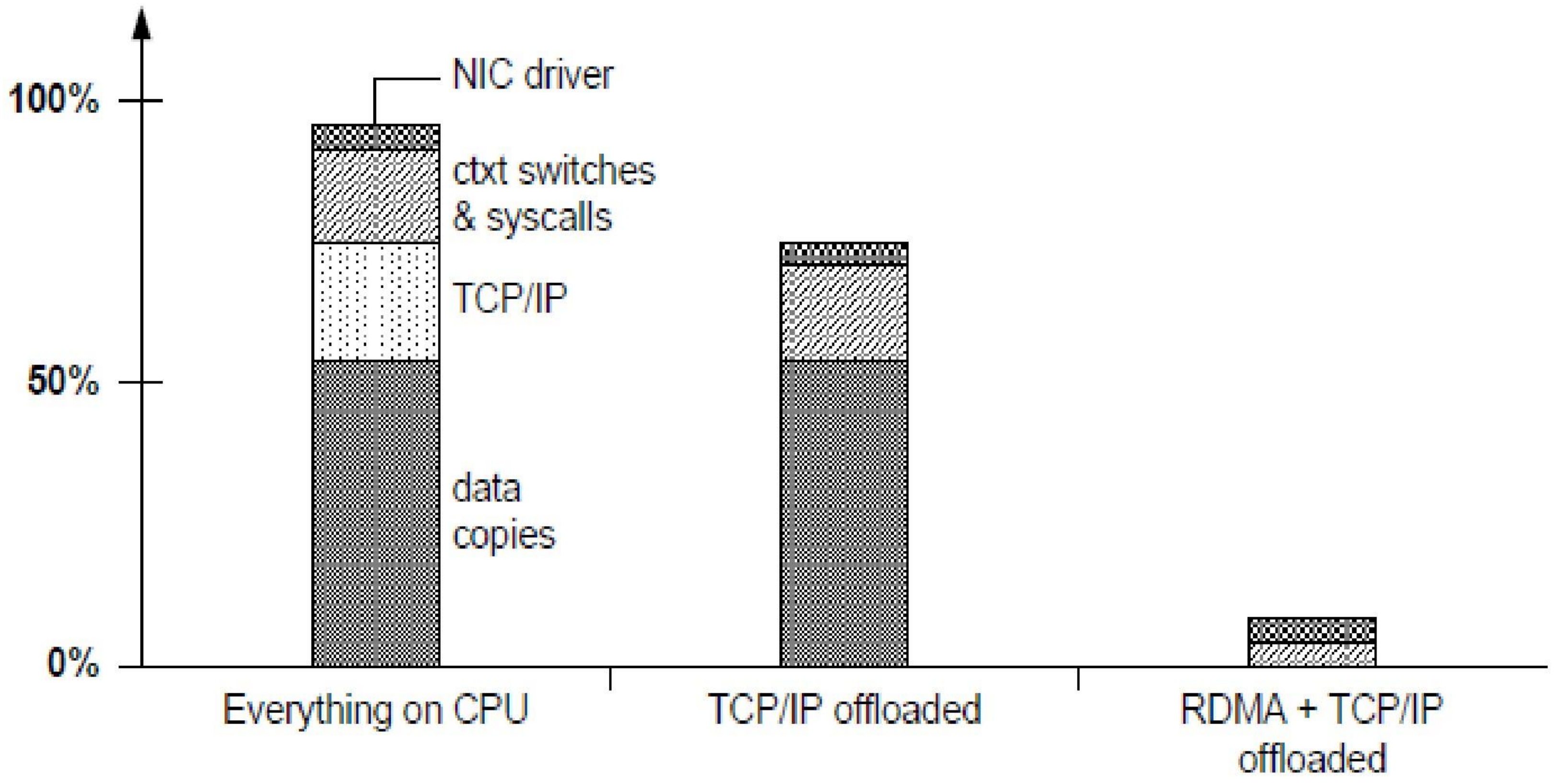
- ❖ End-to-end round trip latency

Round Trip latency



size of messages

RDMA Performance



Modern RDMA

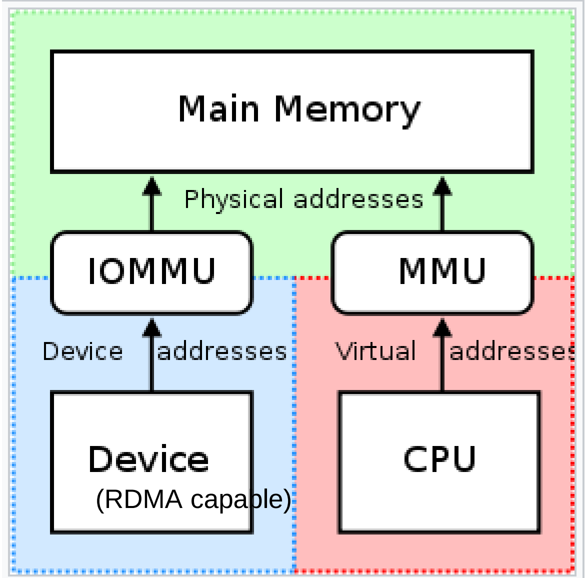
- **Several major vendors**: Qlogic* (Infiniband), Mellanox, Intel, Chelsio, and others
- **RDMA has evolved** from the U/Net approach to have three "modes"
  - **Infiniband (Qlogic PSM* API):** one-sided, no "connection setup"
  - **More standard:** "qpair" on each side, plus a binding mechanism (one queue is for the sends, or receives, and the other is for sensing completions). Both sides (RDMS Send and RDMA Responder) participate in client Read/Write.
  - **One-sided RDMA:** after some setup, allows one side to read or write to the memory on the other side, but pre-permission is required
  - **RDMA + VLAN:** needed in data centers with multi-tenancy

* **PSM**: Performance Scaled Messaging

* http://www.qlogic.com ➔ https://cavium.com/index.html
https://en.wikipedia.org/wiki/QLogic

Modern RDMA

- **Memory management is tricky:**
  - Pages must be pinned and mapped into IOMMU* (IOMMU is an MMU that translates IO contigous virtual address into possibly fragmented physical memory)
  - **Kernel will zero pages on first allocation request**: slow
  - If a page is a mapped region from a file, kernel may try to automatically issue a disk write after updates, costly (synchronous write)
  - Integration with modern Non-volatile RAM storage is "awkward"
  - On multicore NUMA machines, hard to know which core owns a particular memory page, yet this matters
- **Main reason we should care?**
  - RDMA runs at 20, 40Gb/s. And soon 100, 200… **1Tb/s**
  - But memcpy and memset run at perhaps **30Gb/s**



Comparison of the I/O memory management unit (IOMMU) to the memory management unit (MMU).

* https://en.wikipedia.org/wiki/Input%E2%80%93output_memory_management_unit

NUMA-aware RDMA-based end-2-end Data Transfer System

Overview

- Need to transfer large amounts of data long distances (end-to-end high performance data transfer), i.e., inter-data center transfer

- **Goal:** design a network to overcome three common bottlenecks of large-haul end-to-end transfer systems
  1. Achieve **100 Gbps** data transfer throughput

Approach

2. Use NUMA to reduce processing overhead when using UMA (kernel is involved)
3. Utilize full network bandwidth employing RDMA to provide low latency and high bandwidth

- **Build storage network with multiple storage components:**
  - Bandwidth equivalent to host’s processing speed
  - Enable RDMA networks use of SCSI commands and objects

Conclusion

http://www.cs.fsu.edu/~xyuan/cis5930.2014/present_Zach.pdf

- **Test results using LAN and WANs evaluating:**
  - **Back-end system performance with NUMA-aware tuning**:
    - Improve write operation bandwidth by ~20%
    - Utilizes CPU up to 3X less by using NUMA architecture
  - **Application performance in end-to-end LAN:**
    - Parallelized RFTP (RDMA-based FTP) has lower I/O request overhead than single threaded
    - Full bi-directional bandwidth is impossible due to resource contention
  - **Network Performance over a 40 Gbps RoCE (RDMA over Converged* Ethernet) long distance path in WAN:**
    - RFTP (RDMA-based FTP) can be scaled to more servers & longer distance

- **Converged Ethernet**: added buffers to the output side of switches with each port can stack several messages ← expensive

END