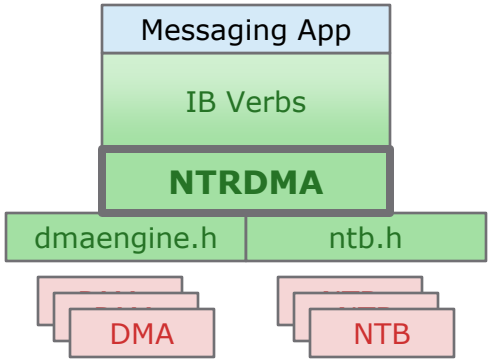


NTRDMA v0.1

An Open Source Driver for PCIe NTB and DMA

Allen Hubbe at Linux Piter 2015

EMC²

INTRODUCTION

- Allen Hubbe
- Senior Software Engineer
- EMC Corporation

NTB IN ENTERPRISE STORAGE

PROBLEM STATEMENT

- Connect nodes in enterprise storage array
  - Messaging and data protection
- Long supported by NTB interconnect
  - Fast and efficient as PCIe at low cost
  - Proprietary hardware and device drivers
- Evolution of software and hardware
  - RDMA programming abstraction
  - New open market platforms with NTB

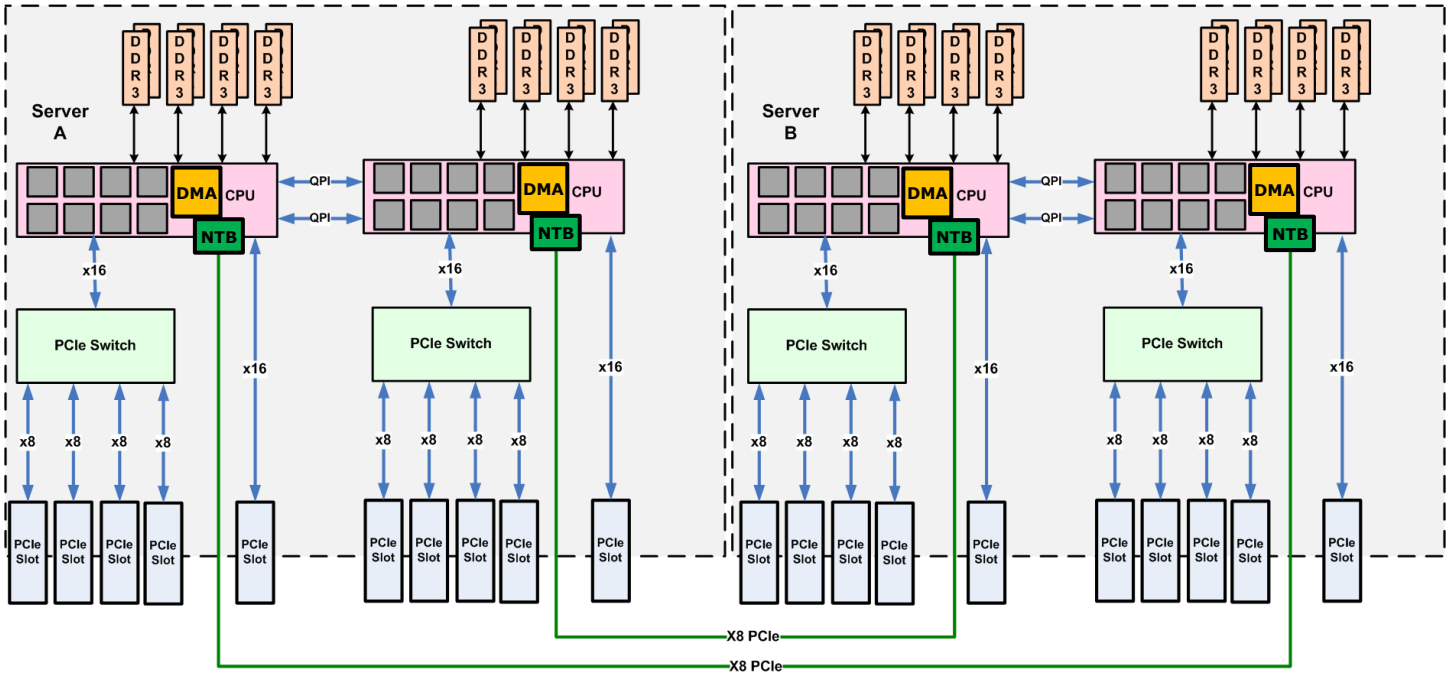
NTB IN ENTERPRISE STORAGE

BASIC COMPONENTS OF THE NTB INTERCONNECT

- NTB – Non-Transparent Bridge
  - A kind of PCI Express bridge device
- DMA – Direct Memory Access
  - A way to offload memory copy operations
- RDMA – Remote Direct Memory Access
  - A programming interface for remote memory operations

NTB IN ENTERPRISE STORAGE

EXAMPLE DUAL SERVER WITH MULTIPLE NTB



BEYOND ENTERPRISE STORAGE

- Messaging and Data Protection
  - Common problem domain across many fields
  - Common concerns: speed, efficiency, cost
- RDMA programming abstraction
  - Migration from proprietary to Open Fabrics APIs
- New open market platforms with NTB
  - Third party R&D, manufacturing, hardware support
  - NTB technology now available at small scale

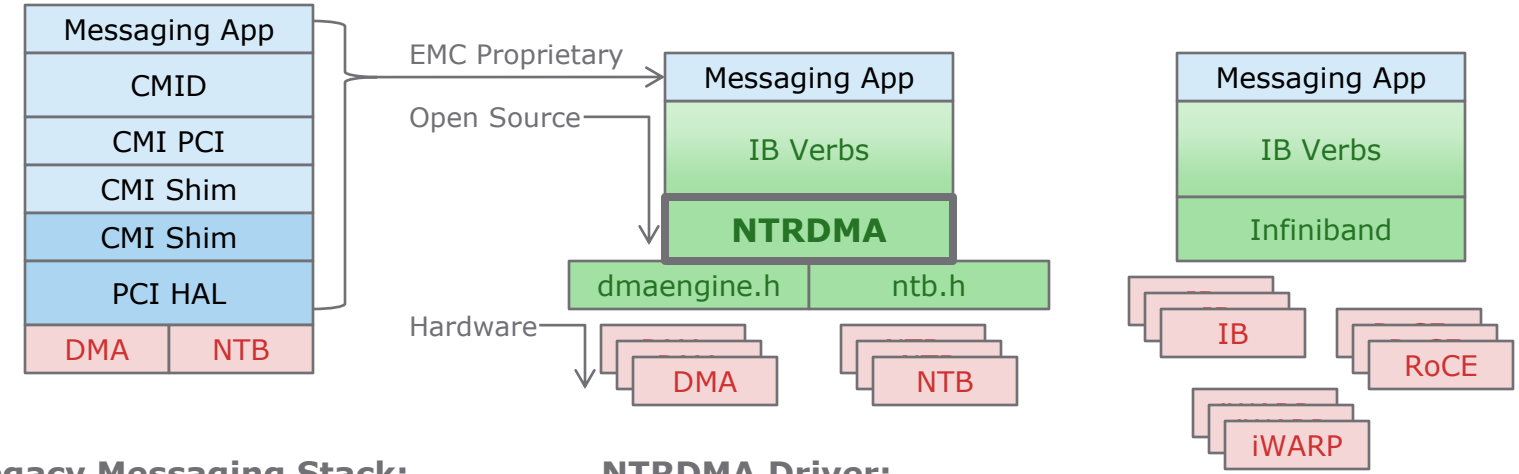
NTRDMA DRIVER

MOTIVATING FACTORS

- Transformation of EMC proprietary stack
  - Proprietary NTB and DMA hardware drivers
  - Proprietary RDMA programming interfaces
  - Tight coupling of hardware, drivers, applications
  - Fast, efficient, highly reliable, but inflexible
- Collaboration with Open Source
  - Access a wider diversity of specialist expertise
  - Share enterprise capabilities with the community

NTRDMA DRIVER

TRANSFORMATION OF EMC PROPRIETARY STACK



Legacy Messaging Stack:

Proprietary infrastructure with no deployment agility, investment relief, or opportunity to leverage alternate interconnects.

NTRDMA Driver:

Open Source hardware drivers with vendor and community support.

Open Source IB Verbs.

Flexible Deployment:

Scale up, down, and out, with different technologies.

NTRDMA DRIVER

SOFTWARE INTERFACES

- NTB Hardware Interface
  - With Intel NTB hardware driver (ntb_hw_intel)
  - Open Source contributions by EMC as of Linux v4.2
- DMA Engine Interface
  - With Intel DMA hardware driver (ioatdma)
- IB Verbs Interface
  - RDMA abstraction traditionally for Infiniband
  - Also supports RoCE, iWARP, and now NTRDMA

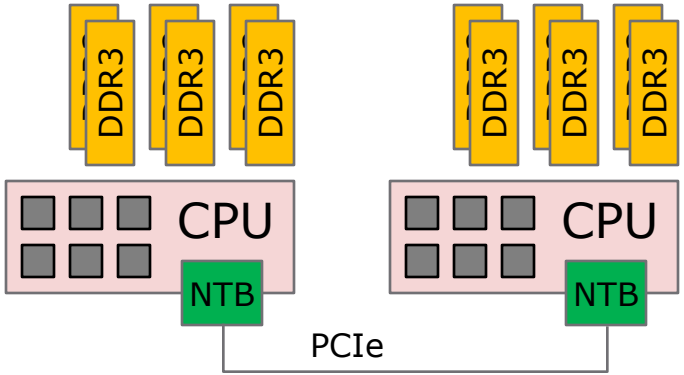
NTRDMA DRIVER

EMC OPEN SOURCE CONTRIBUTIONS TO NTB

- NTB before Linux v4.2
  - Intel devices only
  - Ethernet transport only
- EMC contributions
  - Patch series a1bd3ba, News on Phoronix
  - Apply Linux device model to NTB hardware
  - NTB hardware and transport abstraction
  - Bug fix, stability, and performance improvements

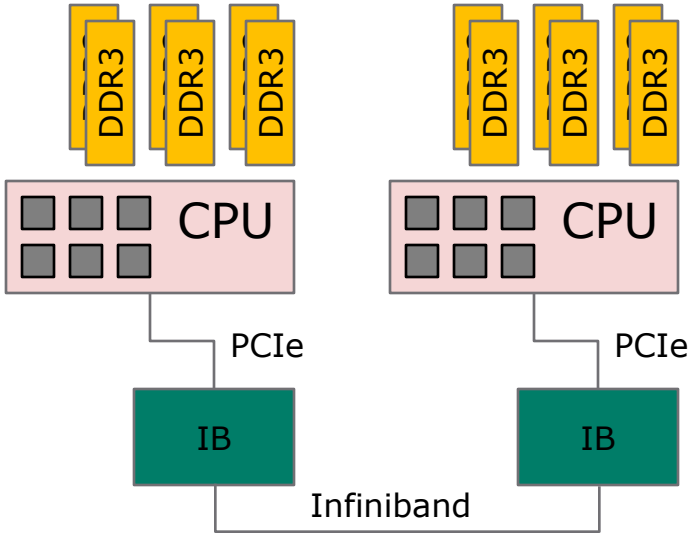
NTRDMA DRIVER

NTRDMA VERSUS INFINIBAND



- Internal network
  - Less hardware to maintain
  - Inaccessible to the customer
  - Single peer, point to point *

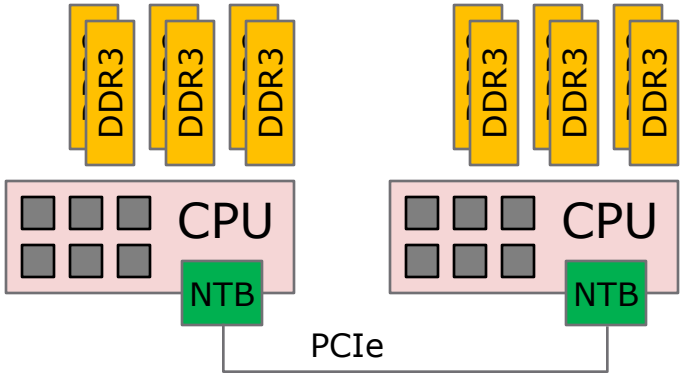
* New NTB hardware may begin to support multiple peers



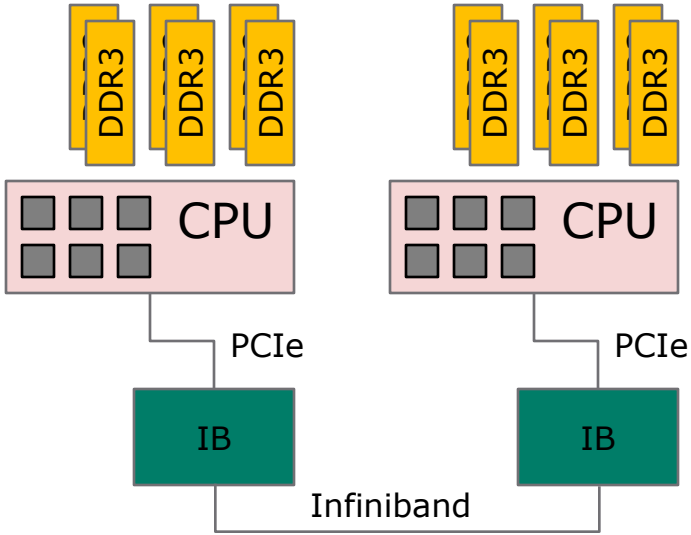
- External network
  - Extra costs for maintenance and service
  - Vulnerable to improper customer access
  - Multiple peer, fabric, switching, routing

NTRDMA DRIVER

NTRDMA VERSUS INFINIBAND



- Only simple operations
  - Write across PCIe bus
  - Interrupt the peer
  - Direct access to peer memory
- RDMA operations in software
  - Need to mitigate software overhead



- Supports complex RDMA operations
  - Work Requests and Completions
  - Memory Regions and Queue Pairs
  - Protected access to peer memory
- Complete hardware implementation

NTRDMA DRIVER

DOES THIS THING ACTUALLY WORK?

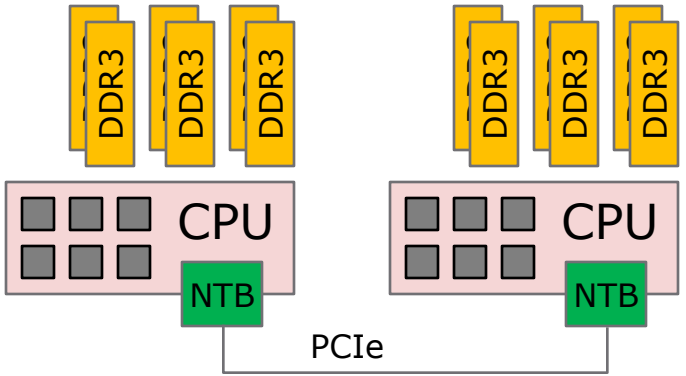
- Multiple NTB devices
  - multiple RDMA devices
  - multiple Ethernet devices
- NUMA Support
  - kernel space & user space
- Performance Demo
  - RDMA Performance
  - Ethernet Performance

LIVE DEMO

NTRDMA DRIVER

THEORY OF OPERATION

- RDMA Operations
  - Move data from src to dst, where src or dst is remote
  - Place data directly into peer memory (zero copy)
  - DMA offload bulk data copy

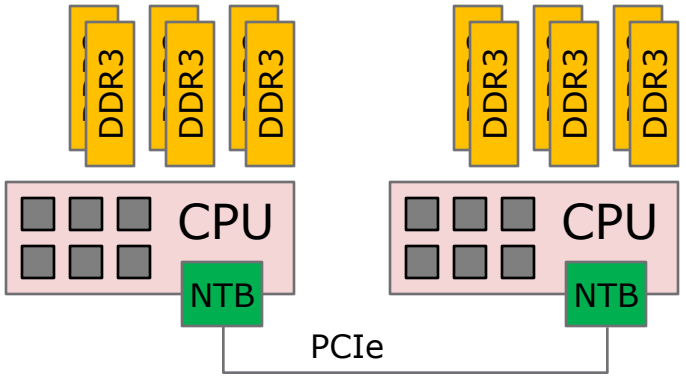


- Other Requirements
  - Minimize protocol overhead, small writes, interrupts
  - Avoid PCI READ across NTB bus, other hardware errata
  - Maintain system stability throughout link or peer failures

NTRDMA DRIVER

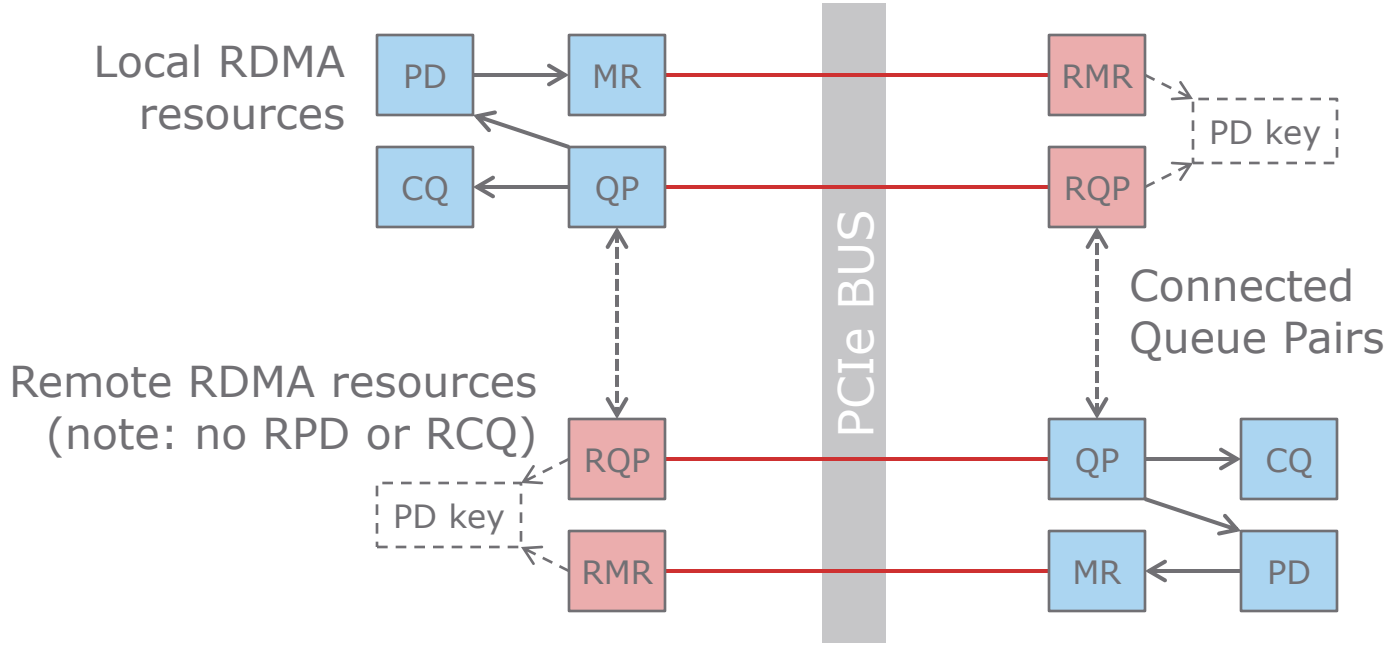
THEORY OF OPERATION

- Local Resources
  - MR, QP, PD, CQ
  - RDMA resources created or registered by local app
- Remote Resources
  - RMR, RQP
  - RDMA resources created or registered by remote app
  - Sending side uses to correctly place data on the peer
  - There is no RPD or RCQ



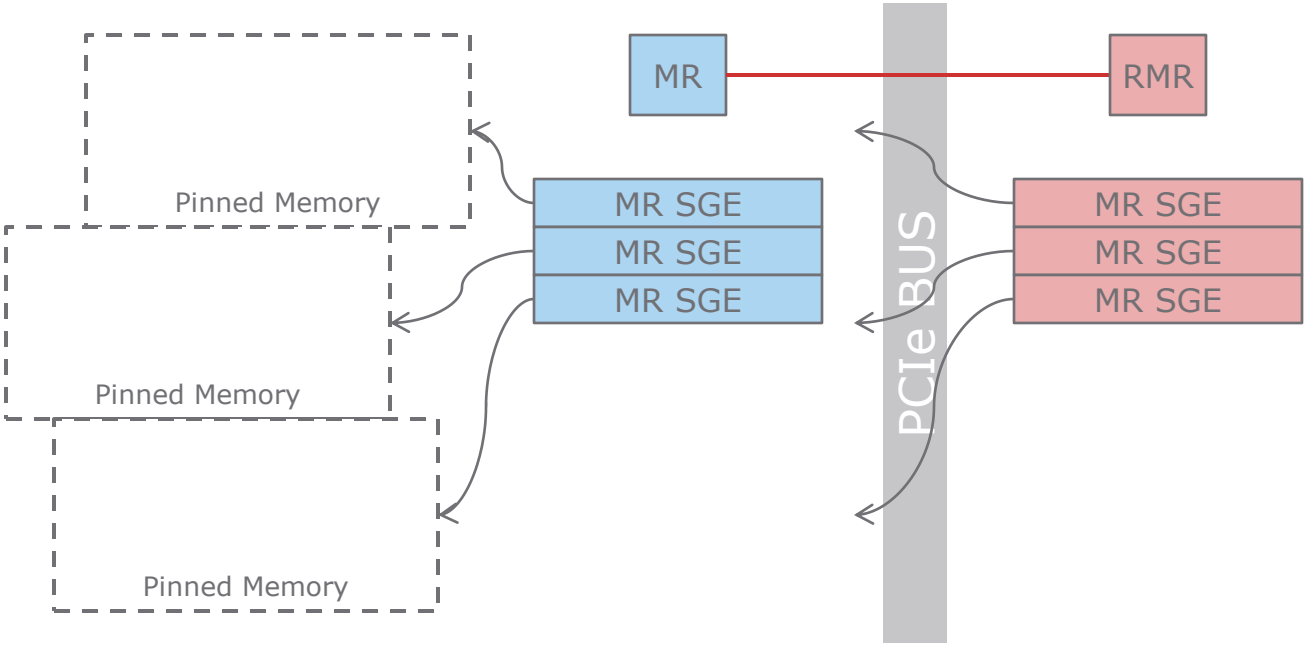
NTRDMA DRIVER

REMOTE RESOURCES



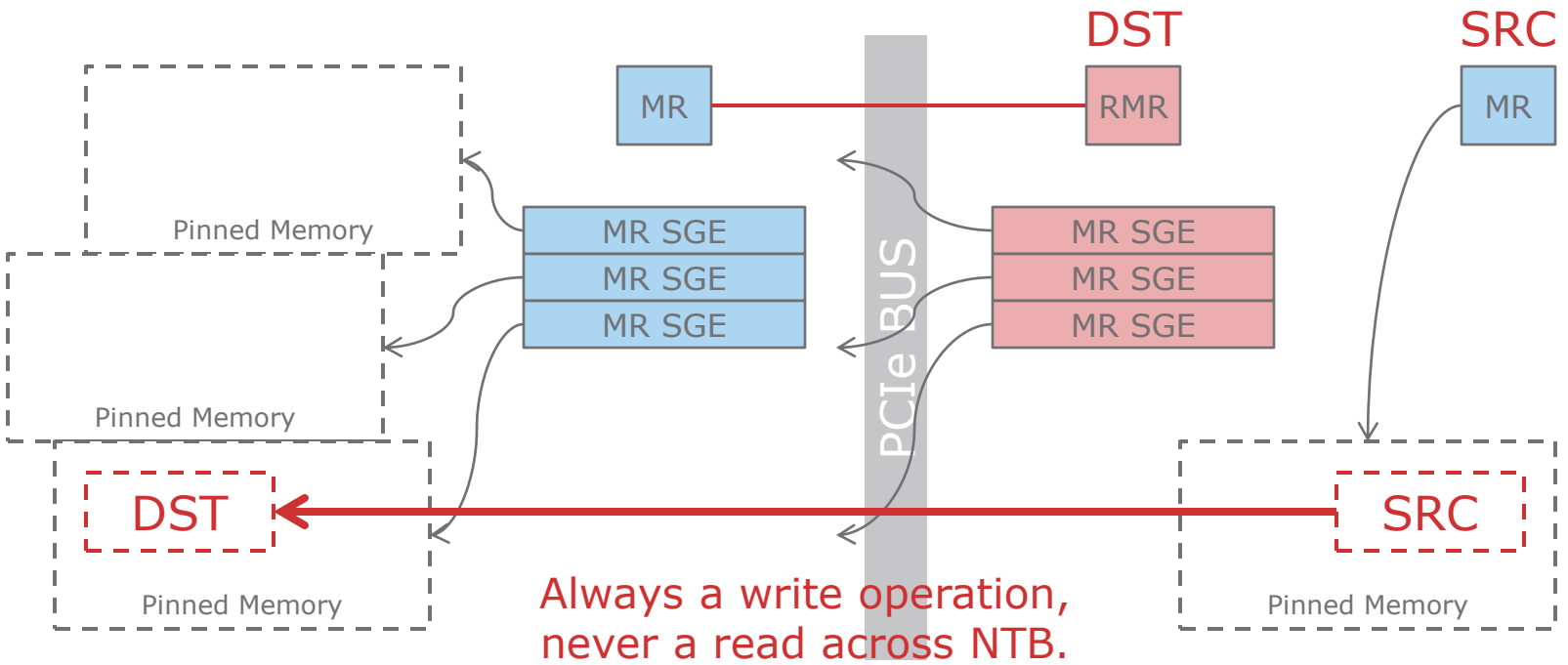
NTRDMA DRIVER

MEMORY REGIONS



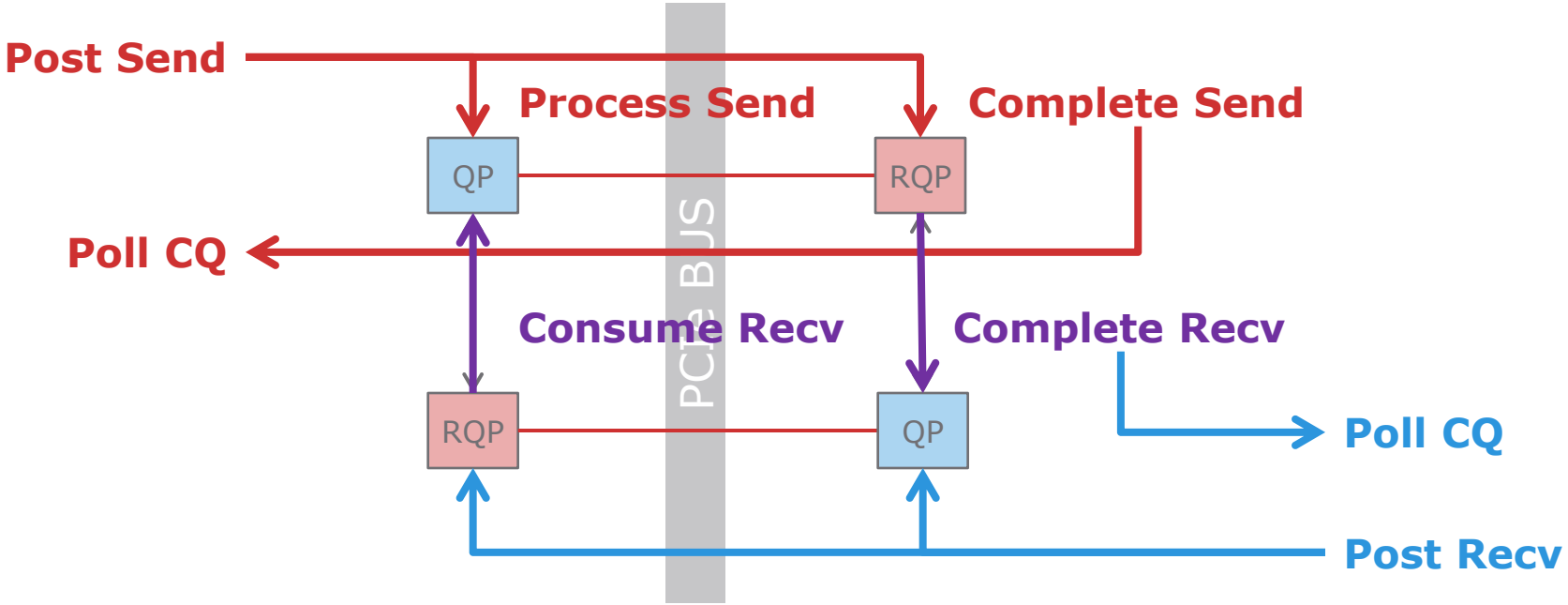
NTRDMA DRIVER

DATA TRANSFER DIRECTION



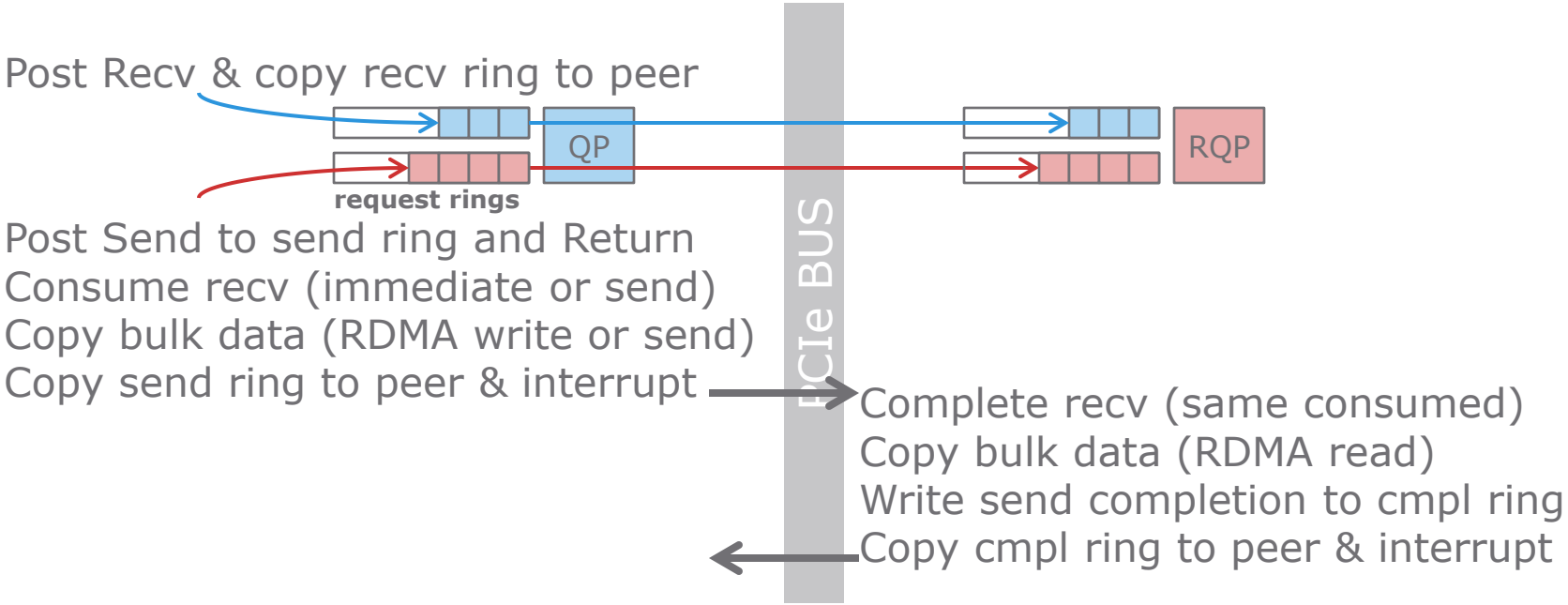
NTRDMA DRIVER

QUEUE PAIRS



NTRDMA DRIVER

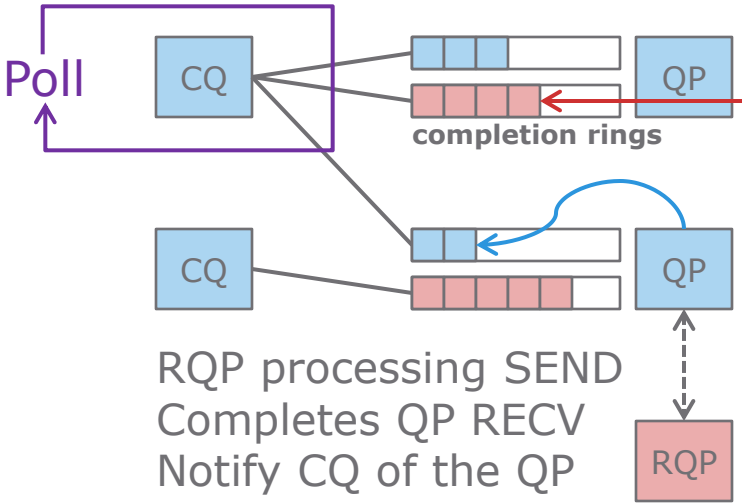
WORK REQUESTS



NTRDMA DRIVER

WORK COMPLETIONS

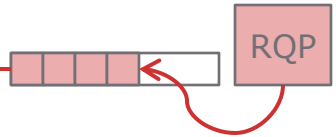
Application may poll at any time.
CQ reads from the QP ring buffers.



RQP processing SEND
Completes QP RECV
Notify CQ of the QP

PCIe BUS

Complete SEND
Copy ring buffer data
Interrupt to notify CQ



Notify CQ only wakes the application polling thread.

Application can affine thread so polling runs on the correct CPU.

NTRDMA DRIVER

PROTOCOL OVERHEAD

- Small operations are bad for DMA throughput
  - Trade-off: can deliver interrupts to the peer sooner
- Solution: process batches of requests at a time
  - Four small operations per batch data transfer
    - Single copy of contiguous ring metadata (small-ish)
    - Update the ring index*
    - Update the virtual doorbell*
    - Deliver interrupt to the peer*

      * immediate data operations, four bytes each

NTRDMA DRIVER

HARDWARE ERRATA

- Doorbell and Scratchpad can lock the CPU
- Solution: Never use the doorbell registers
  - never read doorbell register in interrupt handler
  - never write doorbell register to interrupt the peer
  - write msi data to directly to peer lapic
- Solution: Scratchpad only during link initialization
  - Scratchpad used initially to communicate buffer address
  - Handshake to use the buffer, and stop using scratchpads

NTRDMA DRIVER

TCP BACK END

- TCP = Zero Hardware Requirements
  - Full **functional equivalence**
    - Ethernet, RDMA
  - Not a performance back end
- How does that work?
  - Send to kernel socket instead of prepare dma
    - very simple protocol: header+data, destination address, length
  - Mapping uses all kernel-virtual/logical addresses
    - mmap user pages instead of pinning for DMA

NTRDMA AND YOU

NTRDMA IS OPEN SOURCE

- Use it in your product
  - Looking to work with early adopters
  - Looking for more diverse applications
- Contribute to the code
  - RDMA Connection Manager (rdmacm)
  - IB Verbs asynchronous events (ibv_get_async_event)
  - High QoS Queue Pairs (CPU memcpy)
  - Libfabric, Qperf, bug fixes and reports

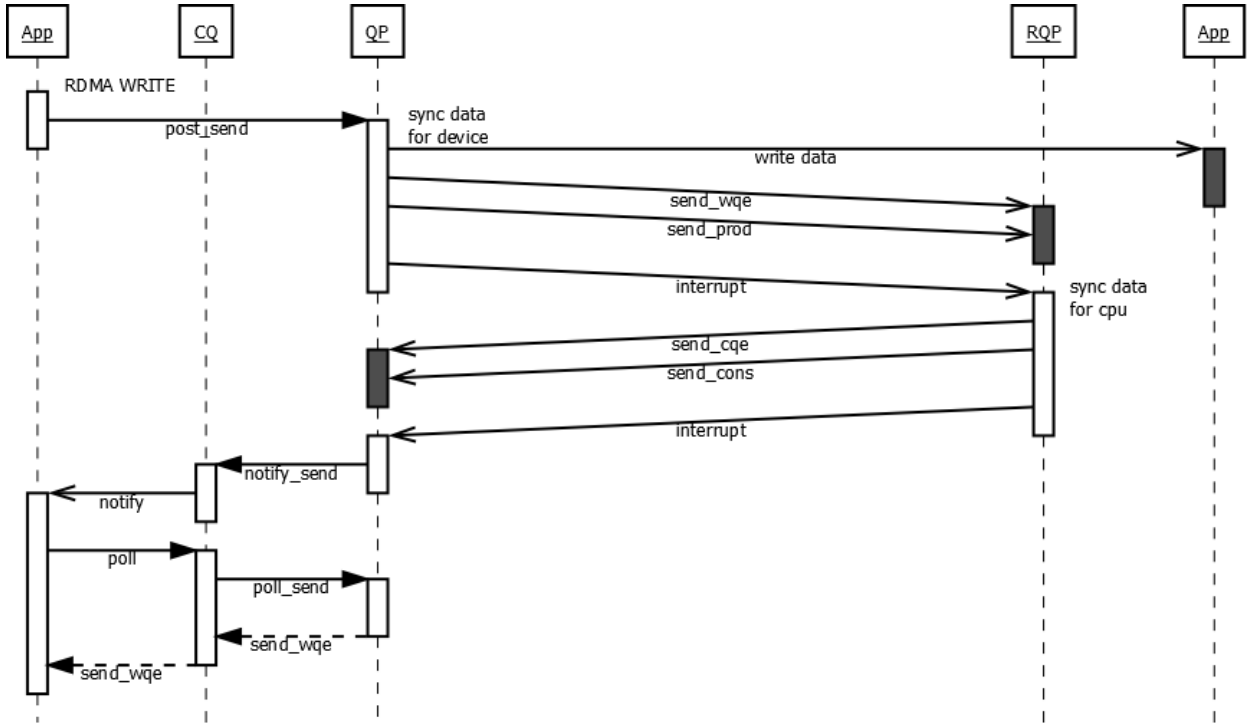
NTRDMA v0.1

An Open Source Driver
for NTB and DMA
github.com/ntrdma

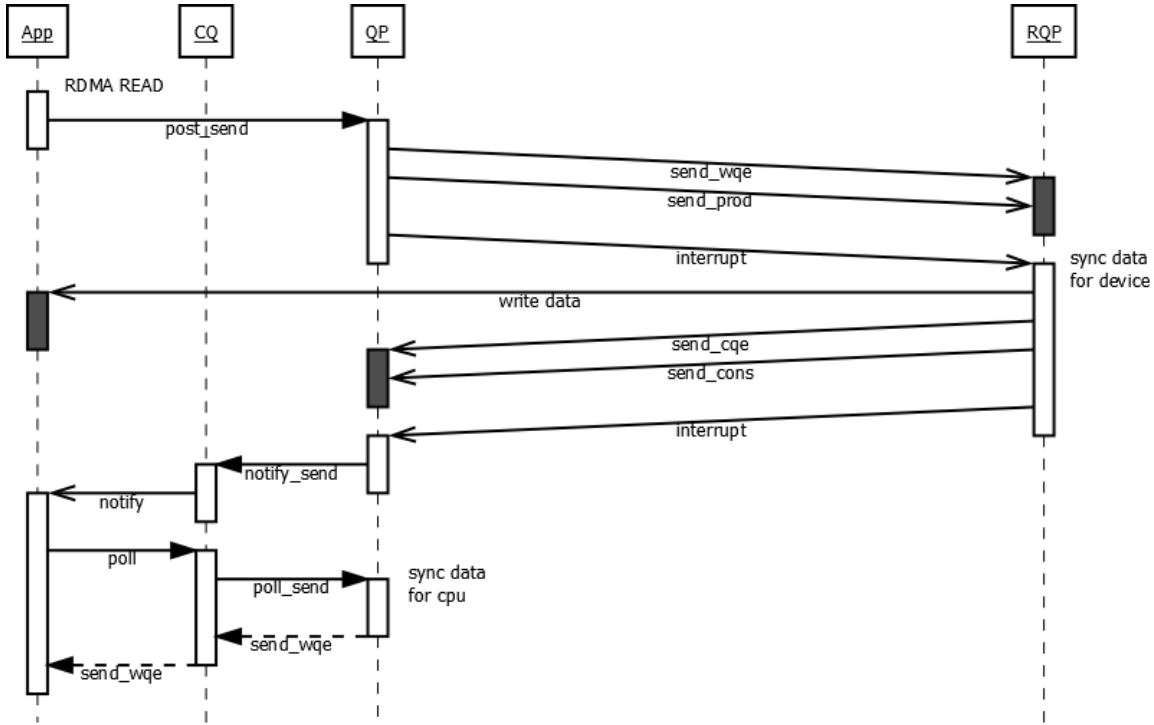
EMC²


EMC²

RDMA WRITE



RDMA READ



SEND AND RECV