After 10+ years of NTB in specialized hardware, PCI-express Non-Transparent Bridge technology is making its entrance into retail off the shelf server solutions. Linux, with its selection of open source drivers for NTB, is strategically positioned to unlock the value of this low cost, low latency, high bandwidth interconnect.
What is NTB

- What is a PCIe Bridge?
  - Forwards PCIe traffic between buses
  - Attach CPU to multiple end point devices
What is NTB

- PCIe “Non-Transparent” Bridge
  - Forwards PCIe traffic between busses like a bridge
  - CPU sees the bridge as an end-point device
  - CPU does not see devices on the other side
  - Other side is typically attached to another CPU
NTB Functions

• Memory Window
  – PCIe memory aperture allocated to the NTB
  – PCIe writes (and reads) are translated across
    • Normally configured to access peer memory

• Doorbell Registers
  – Interrupt the NTB driver on the peer

• Scratchpad Registers
  – Register-size storage visible on both sides
NTB Features

• Low Cost
  – Already present on CPU or PCIe bridge chips

• High Performance
  – PCIe wire speed: NTB connects PCIe buses

• Internally Wired
  – Not accessible to customer, low maintenance
  – External setup also supported (redriver and cable)
Supported Topologies

• Linux NTB API supports a single peer
• Seeking NTB Drivers from PCIe switch makers

**B2B:** Back to Back
(typical, and recommended)

**PRI:** Primary
(aka TB or VIRT)

**SEC:** Secondary
(aka RP or LINK)

Side without NTB sees secondary side of peer's NTB

Also will be supported for switches (pending hardware drivers)
Other Topologies

- Linux NTB API **does not** currently support
- Seeking leadership from NTB Fabric makers

**Chained:** PRI/SEC with translation offsets overlapping memory windows

**Fabric:** mesh, star, or ring topology with PCIe or proprietary interconnect
NTB Applications

• Enterprise Storage
  – data protection in the write cache
  – internal network device

• Embedded Systems
  – simple, low level, low cost interconnect
  – component isolation and device fail-over

• Other Applications
  – DRBD distributed block device
NTB Hardware

• Historically: Original Design Manufacturing
  – Contract a hardware vendor to build servers
  – Available as ODM for 10+ years (2004?)

• Today: Retail / Off the Shelf servers with NTB
  – Immediately available, and in small quantities
  – Decreased up front and research costs
Why Now?

...and why at a Linux conference?

• Retail / Off the Shelf Hardware with NTB
  – Small budgets demand Linux and OSS

• What value is NTB without drivers?
  – Intel NTB Hardware Driver (since Linux v3.9)
  – NTB Ethernet Device (since Linux v3.9)
  – NTB Hardware API (since Linux v4.3)
  – NTB RDMA Drivers (available out-of-tree)
Linux NTB Development

• Maintainers
  – Jon Mason <jdmason@kudzu.us> (lead)
  – Dave Jiang <dave.jiang@intel.com>
  – Allen Hubbe <allen.hubbe@emc.com>

• Git Repo: github.com/jonmason/ntb

• Mailing List: linux-ntb@googlegroups.com

• IRC Channel: #ntb on irc.oftc.net
Linux NTB Driver Stack

- **ntb_hw_xxx, ntb**: Hardware drivers and abstraction layer
- **ntb_transport**: Generic CPU and DMA-assisted data transfer
- **ntb_netdev**: Ethernet device using ntb_transport
- **ntb_tool, perf, pingpong**: Examples and test drivers
- **ntrdma, ntb_scif**: RDMA over NTB drivers for IB Verbs
RDMA over NTB

• Could be the “killer app” for NTB
  – Low cost, high performance, *internally wired*
  – IB Verbs: easily swap for Infiniband, RoCE

• Driver alternatives
  – RDMA over NTB: NTRDMA or SCIF
  – RDMA verbs in NTB hardware
Legacy Proprietary 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

• Non-Transparent RDMA (NTRDMA)
  – Intended purpose is RDMA over NTB

• What Works
  – IB User Verbs, performance tested

• What Needs Work
  – RDMA Connection Manager (not implemented!)
  – Needs wider deployment and testing
  – github.com/ntrdma
SCIF

- **Scientific Communications InterFace (SCIF)**
  - Supports RDMA to Intel® Xeon Phi™ coprocessor
  - *Could be made to support RDMA over NTB*

- **What Works**
  - RDMA over Phi, performance tested on Phi

- **What Needs Work**
  - NTB to SCIF driver adapter (not implemented!)
  - github.com/sudeepdutt/mic
NTB: Try it Out, Make it Better

• NTB Hardware Makers
  – Send your NTB drivers! (Thanks Intel, AMD)
  – NTB Fabrics: help us improve the API

• Interested, or already using NTB?
  – Go get some hardware: now you can!
  – Looking for RDMA driver users, developers
  – Come say hi on #ntb
Questions?

• What is NTB?
  – NTB features, example server, topologies
  – NTB applications in storage, embedded

• Why now, why Linux?
  – Retail servers with NTB, users will run Linux

• Development
  – Maintainers, mailing list, IRC, components
  – NTB HW drivers, NTB Fabric API, RDMA
BACK UP SLIDES
Security

• **Attack Surface**
  – Peer has read/write access to physical memory

• **Mitigations**
  – Limit size of NTB memory window
    • RDMA memory window is *all of dynamic memory*
  – Use IOMMU/virtualization hardware (Intel VT-d)
    • Intended to isolate hardware in virtual environments
    • NTB should only access its dma-mapped memory
Example Server with NTB

Server A

CPU

MEM

MEM

MEM

MEM

MEM

MEM

MEM

MEM

DMA

RP

RP

RP

NTB

Server B

CPU

MEM

MEM

MEM

MEM

MEM

MEM

MEM

MEM

DMA

RP

RP

RP

NTB

IO DEV

IO DEV

IO DEV

IO DEV

PCIe Midplane
Example Server with NTB

Server A

Server B

CPU

MEM

MEM

MEM

MEM

MEM

MEM

CPU

MEM

MEM

MEM

MEM

MEM

MEM

DMA

RP

RP

RP

RPC

DMA

RP

RP

RP

RPC

IO DEV

IO DEV

IO DEV

IO DEV

IO DEV

IO DEV

IO DEV

IO DEV

IO DEV

IO DEV

SW

SW

SW

SW

PClE Midplane
Detailed NTB B2B Translation

Access from Node A

0x383ffe01000

PBAR23

0x383ffe00000 (BIOS set)

0x1000

PBAR2XLAT

0x1000000000 (reset default)

0x1000001000

SBAR2BASE

0x1000000000 (reset default)

0x1000

SBAR2XLAT

0x87f00000

Result on Node B

0x87f001000

Start of Node B memory location
B2B Workaround

Local NTB 0xA
Peer NTB 0xB
Peer MW 0xC
Peer MW 0xD

MW

NTB

PBAR01 0xA - 0xB
SBAR01

PBAR23 0xB - 0xD
SBAR23

PBAR45 0xD - 0x10
SBAR45

NTB

SBAR01 0xB - 0xC
PBAR01

SBAR23 0xC - 0xD
PBAR23

SBAR45 0xD - 0x10
PBAR45

MW

MW