Implementation of Application and a client driver on Non-Transparent Bridge

Sahana Y. S, Prof. Smitha G.R,
MTech, Assistant Professor,
Dept of ISE,
RVCE college, Bangalore, India

Abstract : The need of communication between two servers in the present world is Highly important. Though the performance is achieved speed is at risk in case of any two host servers. Achieving Data transferring speed up to 1GB is merely challenging. PCI is a multi-drop bus-based technology that was originally intended for compute applications, with the expectation that the host processor would control the entire system. In the PCI architecture, Non-Transparent Bridges are used to expand the number of slots possible for the PCI bus. Implementing an application and a client driver drives the Non-transparent bridge in achieving high speed upto 5GB/sec.

IndexTerms **- Peripheral Component Interface, NTB-Non-Transparent Bridge, Memory window, Doorbells,**

I. INTRODUCTION

In the present world, the need for communication between two servers is highly significant. While performance is achieved, in the case of any two host servers, speed is at risk. It is only difficult to achieve data transmission speeds of up to 1 GB, but this application slogan goes up to 5 GB/sec.Open sourcing this framework is the key inspiration. The Peripheral Component Interconnect Express specification defines a system with PCI Express as inherently hierarchical in that there is always a Root Complex that serves as master to which is connected either endpoints or switches. A Root Complex supports a "Root Port" that consists of a bidirectional data link to the other device. It's leadership in High Performance Computing is world wide The use of non-transparent bridges in PCI systems to support adapters in enterprise systems and multiple processors in embedded systems is well established. Nontransparent Bridging makes connection more feasible and enhance the communication speed.

II. DISTRIBUTED SYSTEM

A distributed system, also known as distributed computing, is a system with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent system to the end-user The machines that are a part of a distributed system may be computers, physical servers, virtual machines, containers, or any other node that can connect to the network, have local memory, and communicate by passing messages.

2.1 Benefits and challenges of Distributed system

There are three reasons that teams generally decide to implement distributed systems:

Horizontal Scalability—Since computing happens independently on each node, it is easy and generally inexpensive to add additional nodes and functionality, as necessary.

Reliability—Most distributed systems are fault-tolerant as they can be made up of hundreds of nodes that work together. The system generally does not experience any disruptions if a single machine fails.

Performance—Distributed systems are extremely efficient because workloads can be broken up and sent to multiple machines.

NTB in a distributed sytem is as shown below

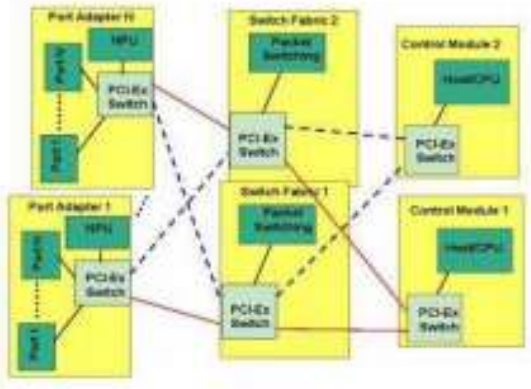

Figure 1

In the current example, using PCI Express switches, 2 controlling modules system and two switching fabrics designer modules linked together. The non-transparent bridging principle is exploited by these PCI Express switches. A chass control modularity controls the overall function of the system-units in many chass-based units. A backup chass control module is present in a HA environment hence designed that monitors upgradation of the device. The modularities usually referred to as secondary and primary, where the primry host actively controls the system while the secondry systems tracks the system.

III. RELATED MODULES

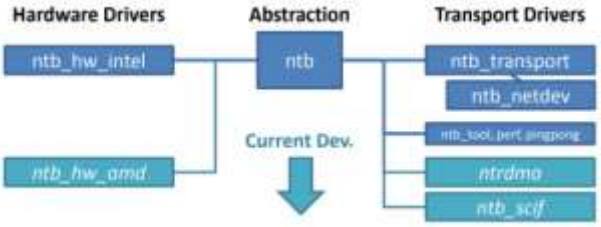

Figure 2

3.1 Iperf module

In the test directory, the NTB perf tool attempts to calculate as close to HW output to view potential performance with minimal software intervention on NTB. Although the calculated CPU copy output is very close to bare metal, due to having to use the Linux DMA driver, the DMA performance will not achieve a similar type of performance. The current perf driver only supports measurement of single direction efficiency. Of course, potentially, if you run the test long enough,can run experiments on both sides at the same time and interpret the findings on each side. Both sides need to have ntb perf loaded in order to get things underway. The defaults set by the perfect driver can be used to quickly detect if things are working correctly, but to get accurate results, a longer test is suggested. After loading both sides in dmesg (On Haswell platform): Intel(R) PCI-E Non-Transparent Bridge Driver 2.0 ntb-hw-intel 0000:00:03.0: Reduce doorbell count by 1 ndev-spad-write: NTB unsafe scratchpad access ntb-hw-intel 0000:00:03.0: NTB system. Implementation ofApplication and client driver for Non-Transparent Bridge

3.2 NTB transport

There are 4 kernel modules in the NTB subsystem: ntb.ko - the common NTB hardware driver glue ntb hw intel.ko - the Intel NTB hardware driver ntb transport.ko - the NTB transport ntb netdev.ko - the driver that exposes NTB as a virtual NIC Normally ntb hw intel.ko will be automatically loaded via the PCI device ID after the kernel has been installed. Ntb.ko will be loaded as a dependency. Ntb netdev.ko can be loaded, and ntb transport.ko would be loaded automatically. Ntb transport has some kernel parameters, although that may be of interest transport_mtu: For transport, the scale of the MTU. Remember, it is not possible to configure the MTU via an Ethernet system. This is set to 64k by default and seems to provide the best throughput efficiency. copy_bytes: The threshold at which the NTB can use the CPU to copy to resolve the latency Instead of using the DMA generator. This is set at 1k by design. Configuring this to 0 would force all copies of the DMA.

3.3 use_dma

This parameter allows data transport using DMA. It is switched off by default. It is found that CPU copying only with write-combining surpasses DMA depending on CPU SKUs. Root Port Mode / Transparent Bridge configuration This is what you need to do to load the NTB driver, unless the BIOS has support that makes the NTB on the RP side for some platform configurations. This parameter allows data transport using DMA. It is switched off by default. It isfound that CPU copying only with write-combining surpasses DMA depending on CPU SKUs. Root Port Mode / Transparent Bridge configuration This is what you need to do to load the NTB driver, unless the BIOS has support that makes the NTB on the RP side for some platform configurations.

IV. NTB INITIALIZATION

An NTB link operates with one side as a Root Complex and the other as an NTB port. When initializing an NTB port, host software on each side of the link must choose one of two modes of operation as shown below in Figure 3

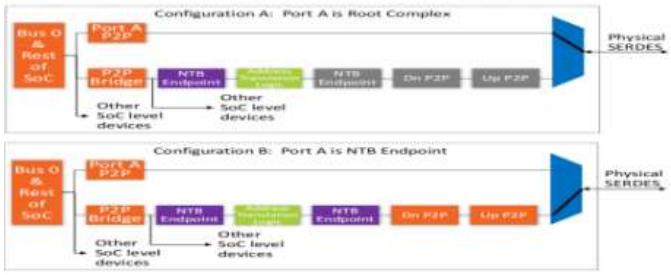

Figure3

V. MEMORY MAPPING

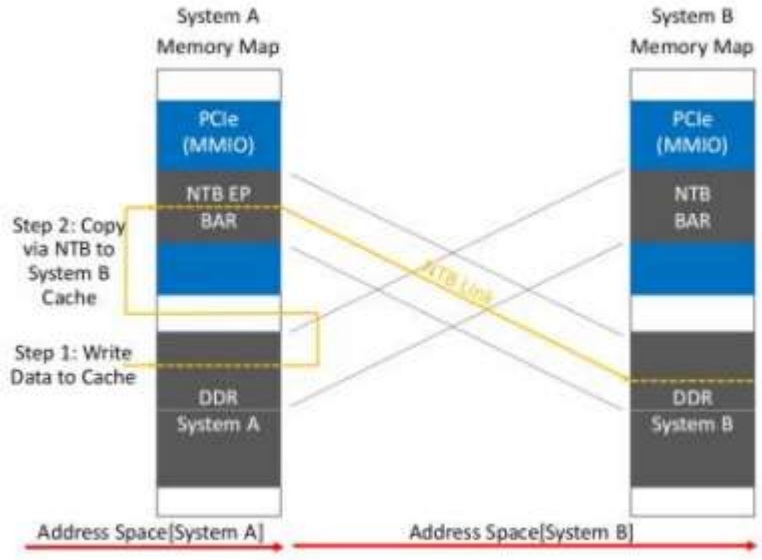

Figure 4

Once this mapping is setup, data is then sent across the link using address-based memory-tomemory copies. Bulk data transfers across the NTB port usually use writes rather than reads to maximize performance. While a processor can be used to do this copy, a DMA engine within the Processor SoC is the preferred method to offload this work from the processor. Because knowing the data has been committed to memory on the opposite write cache is key to closing the client's write command, systems frequently will perform a read after the last data is written. This read typically pushes all prior writes ahead to memory.While write cache coherency is the primary use case for NTB in storage applications, other structures and data may also be synchronized over the link. For example, storage systems must keep extensive meta-data that records among other things the mapping between the LBA (logical block address) of the client's request and the location of the block in the storage appliance's physical storage.

VI. DOORBELL AND REGISTER

aples/Snowy Owl supports a variety of hardware resources that allow the two NTB endpoints to communicate with each other. These include. Sixteen doorbell interrupts Sixteen 32-bit scratchpad registers with an associated semaphore

VII. RESULTS

7.1 Mode selection

Mode selection is a key aspect of system and initialization design. If System A and B always know which mode to operate in, initialization becomes relatively straightforward. To enable this simple case, the Naples and Snowy Owl SBIOS supports either compile-time or user-selectable options for choosing which of the two modes Port A can operate in. A more challenging use case occurs when both hardware and software must be identical on both sides of the link and neither system has a priori knowledge of which mode to use. Naples/Snowy Owl supports this initialization scenario. During initialization, firmware detects when the link has been "cross-connected" such that both sides of the link are operating in Root Complex or NTB Endpoint mode. When the cross-connection state is detected by firmware, SBIOS will randomly switch modes and retrain. As both sides repeat in this cycle, a functional asymmetric configuration will soon be found.

7.2 Hotplug operation Naples/Snowy Owl fully supports both planned and surprise hot plug events as described below. Configuration A shown above in Figure 11 results in enumeration discovering a Root Port, a remote NTB Endpoint and a local (primary) NTB Endpoint. Once initialization is complete, the configuration A side will have an active Root Port A, an active remote NTB Endpoint downstream from this Root Port and an inactive local NTB Endpoint present and their drivers loaded. On the Configuration B side, an inactive Root Port A and an active primary NTB Endpoint will be present together with their drivers.Hence, both the Root Port and primary NTB Endpoint driver will have an active and inactive state depending on the configuration. The inactive driver state discussed here is not to be confused with the case where the PCIe device is logically removed from the system and the driver unloaded. Naples/Snowy Owl NTB Hot Plug support requires that the Root Port A and primary NTB Endpoint drivers are always loaded regardless of the configuration state.

7.3 Extraction

An NTB extraction event can occur in several different ways:

Link failure between two still operational systems (e.g. link cable is disconnected between two systems) or 1power down or actual physical extraction of the system on one side of the link. In both cases, an NTB device must deal with a physical link failure and loss of the device on the opposite side of the link. The process for dealing with this event depends on the configuration the system is in. When a system in configuration A suffers a physical link failure that cannot be restored by lower-level recovery processes, the following events occur: Hardware signals a physical link down event to the Root Port driver No logical-link level link-down event occurs at the Primary NTB Endpoint because the logical link was already down before the physical link failure Software will cause the system to unload the downstream remote NTB Endpoint driver that is now no longer present System software then takes higher level implementation-specific actions such as transitioning the system from a dual-redundant to failovermode operational state

7.4 Insertion

An NTB insertion event can occur in several different ways: 1) the link is re-established between two still operational systems (e.g. link cable is plugged back in) or 2) a FRU is inserted into the system and powered up. In either case, an NTB device must deal with the physical link being re-established and devices reappearing on the opposite side of the link. The process for dealing with this event depends on the configuration the system is in. When the link is re-established to a system in configuration A, the following events occur: Underlying hardware and firmware train the physical link and establish link operation Hardware then signals a physical link-up event to the Root Port driver No logical-link level link-down event occurs at the Primary NTB Endpoint because the logical link was already down before the physical link failure Software re-enumerates devices downstream from the Root Port Software will reload the downstream NTB Endpoint driver that is now present System software then takes higher level implementation-specific actions such as transitioning the system from a failover to a dualredundant-mode op Insertion of a new device may come up with the wrong configuration if had no prior knowledge of the proper configuration. In this case, a cross-link configuration will occur where both sides of the NTB link are configured the same way. As already described above, firmware on both sides will detect this situation and randomly swap configurations until retraining is successful.

7.5 Successful NTB link implementation

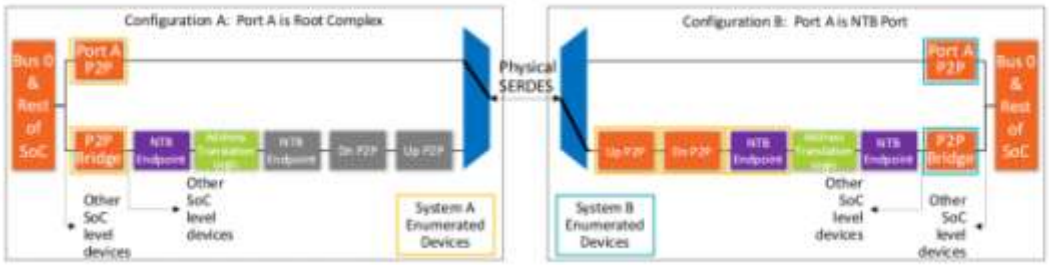

Figure 5

With the help of above operations, the NTB can be implemented successfully cent percent as shown in figure 5

VIII. CONCLUSION

Using the above technology, PCIe with further speed can be achieved even more than 5GB/sec with all other forms of data like images, audio, video etc. PCIe as the ubiquitous connectivity fabric between CPUs, GPUs adds additional challenges during debugging of distributed systems.Since many peripheral devices can't be connected to CPU and GPUs This concept of PCIe with NTB is a very good concept to connect any number of peripheral hardwares. The Root Complex is master because it always performs device enumeration, configuration and initialization activities for all devices in the system. Typically devices with Root Complexes also have processors and attached memory.

IX. REFERENCES

[1] D. Muench, O. Isfort, K. Mueller, M. Paulitsch, and A. Herkersdorf, "Hardware-Based I/O Virtualization for Mixed Criticality Real-Time Systems Using PCIe SR-IOV," *in International Conference on Embedded Software and Systems (ICESS), 2017.*

[2] D. Muench, M. Paulitsch, and A. Herkersdorf, "Temporal Separation for Hardware-Based I/O Virtualization for Mixed-Criticality Embedded Real-Time Systems Using PCIe SR-IOV," in *International Conference on Architecture of Computing Systems (ARCS), 2018.*

[3] D. Muench, "IOMPU: Spatial Separation for Hardware-Based I/O Virtualization for Mixed-Criticality Embedded Real-Time Systems Using Non Transparent Bridges *(TR-TX4- 399)," Airbus Group, Tech. Rep., 2015.*

[4] PCI-SIG, "Single Root I/O Virtualization and Sharing Specification 1. *International Conference on Architecture of Computing Systems (ARCS),2010.*