University of Amsterdam

SNE
System and Network Engineering

ClearStream: Prototyping 40 Gbps Transparent End-to-End Connectivity

Cosmin Dumitru, Ralph Koning, Cees de Laat

March 10, 2011

Abstract
The ever increasing demands of data intensive eScience applications have pushed the limits of computer networks. With the launch of the new 40 Gigabit Ethernet standard, 802.3ba, applications can go beyond the common 10 Gigabit/s per data stream barrier for both local area and, as it shall be presented in this report, wide area setups. This report focuses on the setup and results of two 40 Gbps Ethernet technology demonstrations, the first at GLIF 2010 in Geneva, Switzerland and the second at Supercomputing 2010 in New Orleans, USA.

SNE technical report SNE-UVA-2011-02
`http://www.science.uva.nl/research/sne/reports/`

1 Motivation and Novelty

The dual 40/100 Gigabit Standard 802.3ba [1] is the next evolution of the Ethernet standard. It increases the maximum speed at which Ethernet frames can be transmitted and defines new physical (PHY) standards for the transport of data over copper or optical media. It was was approved in its final state in July 2010 with vendors announcing hardware that implements it (and its drafts) in September 2009. In Q3 of 2010 40GE client adapters became available in limited supply from Mellanox Inc., the ConnectX2 40GE Network Interface Card (NIC). At the same time several vendors , like CIENA, Extreme Networks and Broadcom, started offering optical switch modules or development platforms.

The technology demos presented at GLIF2010 and Supercomputing 2010 focused on the transport of 40 Gigabit Ethernet (40GE) application traffic over wide area network (WAN) links. Their purpose was to investigate and then validate the current technology stack from optical (physical) layer to application layer and lead the way to adoption of 40GE in both carrier and datacenter environments. National research and education networks (NREN's) like the Dutch SURFnet, can profit from this throughput increase as they can provide a better service for network intensive scientific applications. While 40 gigabit WAN transport has already been adopted by carriers, until now it was limited to the edge of the datacenter where more 10GE links would be multiplexed into a single 40 gigabit WAN link using SONET encapsulation, for example. Layer 2 (MAC) bonding of the 10GE links is possible on the server side but the per stream throughput is limited to the rate of a single 10GE interface. Additionally, this solution requires more network cards, optical cables and more configuration then a regular network link. In this light the 40GE interface provides a single channel that requires no special configuration and permits transfer speeds of above 10 Gigabit/s. Alternative technologies like Infiniband Quad Data Rate (IB QDR) already provide 40 Gigabit/s bandwidth but they are limited to the premises of the datacenter and although methods of extending them over WANs exist, they are still in an emerging state.

2 Setup

This section will describe the experimental setup used during the two technology demos mentioned beforehand. Three main components comprise the setup: the servers, the LAN component and the WAN component.

2.1 Servers

The servers used the Mellanox ConnectX2 40GE NICs which, at the time the demos were performed, were the only available 40GE cards. The cards use

Figure 1: Server Hardware Specifications

Server Model	Supermicro H8DTT-HIBQF	Dell R815
CPU Model	Intel Xeon E5620 2.4GHz	AMD Opteron6172 2.1GHz
Core Count	2 x 4 cores	4 x 12 cores
RAM	24GB	128 GB

the PCI-Express Gen 2.0 8x peripheral interface which supports a maximum of 40Gigabit/s raw transfer rate. During the demos two models of servers were used: OEM Supermicro H8DTT-HIBQF and Dell R815. The hardware specifications of the servers are displayed in figure 1.

Both server models supported the PCI-E Gen 2.0 peripheral interface used by the NICs. Besides the CPU frequency, core count and RAM memory one very important architectural aspect is the way the CPUs are connected to each other and to the north bridges/PCI-E bridges. While the Supermicro motherboard uses only one chip to connect the CPUs to the peripherals, the DELL R815 uses two chips allowing more PCI-E devices to be connected to the machine. This creates a non-symmetric setup which as it will be presented in section 5, creates performance inconsistencies.

The Dell R815 uses the AMD Magny Cours platform employing 48 CPU cores in a chassis of only 2U. The cores are grouped into 4 CPUs or packages, each package having two six-core CPUs that share a common L3 cache.The Hypertransport interconnect enables communication between the nodes or PCI-E bridges. A more in-depth description of the platform can be found in [3].

The Supermicro server offers a lower core density, only 16 cores, with two blades fitted in a 2U chassis. It uses the Intel Nehalem architecture[2] having two packages per blade, each having one node with 4 cores. Although the Hyperthreading feature was available it was disabled during all the tests and demos performed. Inter-node (CPU) communication and PCI-E connectivity to the CPUs is provided by the the Quick Path Interconnect (QPI) interface.

2.2 LAN Connectivity

For LAN connectivity two different Ethernet switches were used. In the GLIF demo (figure 3) the servers were connected to an Extreme X650 Ethernet Switch equipped with a four port QSFP+ 40GE module and in the Supercomputing demo (figure 2) to an 18 port Broadcom Layer 2 switch. QSPF+ optical modules provided the connectivity together with 12 pair MPO fibers (SR4 standard). In the SR4 standard, four multimode fibers are used for the RX side and 4 multimode fibers for the TX side. There are multiple standards for short range and long range 40GE connectivity but SR4 was the one available at the moment the demos were performed. While

Figure 2: Supercomputing 2010 Network Diagram

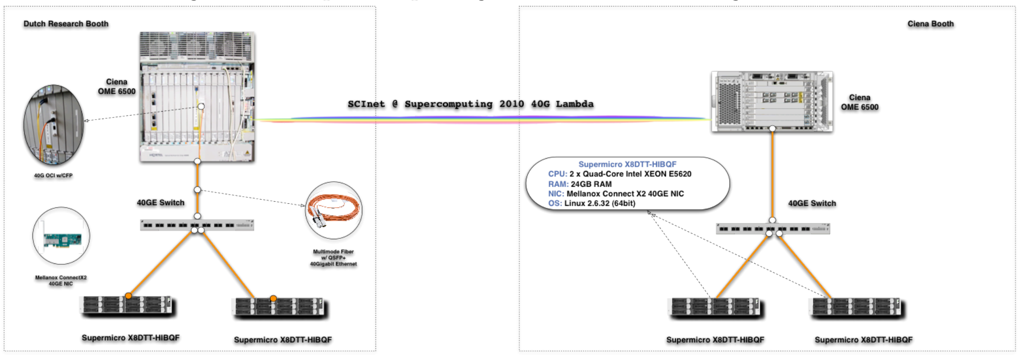
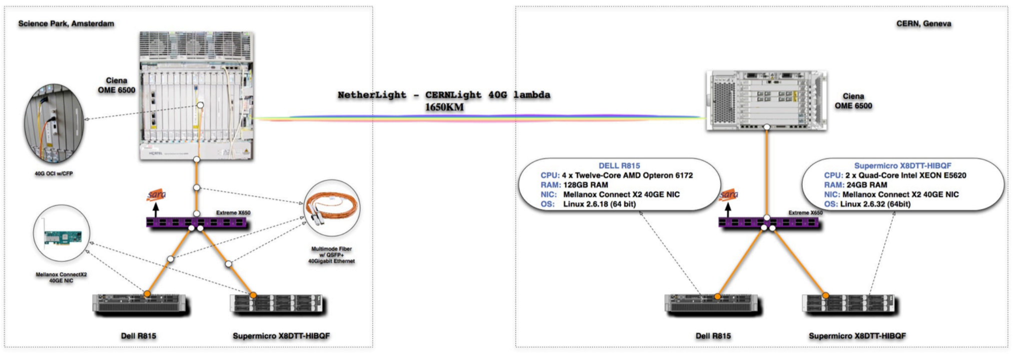


Figure 3: GLIF 2010 Network Diagram


Science Park, Amsterdam
Ciena OME 6500
40G OCI w/CFP
sara
Extreme X650
Mellanox ConnectX2 40GE NIC
Multimode Fiber w/ QSFP+ 40Gigabit Ethernet
Dell R815
Supermicro X8DTT-HIBQF
NetherLight - CERNLight 40G lambda 1650KM
CERN, Geneva
Ciena OME 6500
DELL R815
CPU: 4 x Twelve-Core AMD Opteron 6172
RAM: 128GB RAM
NIC: Mellanox Connect X2 40GE NIC
OS: Linux 2.6.18 (64 bit)
Supermicro X8DTT-HIBQF
CPU: 2 x Quad-Core Intel XEON E5620
RAM: 24GB RAM
NIC: Mellanox Connect X2 40GE NIC
OS: Linux 2.6.32 (64bit)
sara
Extreme X650
Dell R815
Supermicro X8DTT-HIBQF

on the logical level there is only one 40GE interface available to the server the QSPF+ optical module has the function to convert the electrical signal sent by the network card to four 10Gbit/s optical signals which are sent over the four fiber strands. The PHY component of the 40GE standard specifies the way in which the four signals are synchronized and multiplexed on the receiving side. A detailed description of the PHY component of the 802.3ba is presented in [1].

2.3 WAN Connectivity

In both demos the Ethernet switches were connected to a CIENA ActivFlex 6500 optical switch equipped with a prototype card supporting a CFP optical module that provided one 40GE port. The 40GE signal was encapsulated in an OTU3 frame and sent to the receiving side over a DWDM system. On the WAN side a single wavelength was used to carry the data signal. At the receiving side a similar setup was used employing also an ActiveFlex 6500 with the same type of cards and optical modules. The distances varied from less than 1km of fiber at Supercomputing to 1650km of dedicated dark fiber lit by SURFnet at GLIF2010.

2.4 Applications

The application suite was chosen in order to examine two aspects of the new 40GE technology. First, we were interested to see maximum throughput achievable using the 40GE cards and current hardware, and second, we targeted a real life application that could benefit from the speed increase.

For achieving maximum throughput we selected `iperf` as a traffic generating application. While it is not the only application in its class, we chose `iperf` because of its reliability and also because of the easy way to increase the number of data streams, and hence throughput by using multiple flows. In the initial experiments we used also the `netperf` application obtaining the same results as with `iperf`.

For a real world application we settled on using DiVinE, a tool for LTL model checking and reachability analysis of discrete distributed systems[10]. DiVinE leverages multiprocessing in order to distribute its verification task to networked nodes. Each DiVinE process is able to produce under ideal conditions - single machine, only internal communication, Intel Xeon E5550 2.4GHz - around 400Mbps of peak traffic. DiVinE was used before [8] to evaluate the performance of experimental optical networks. It uses MPI to orchestrate the multiprocess communication and achieve massive parallelism. Having this in mind we created a setup where two machines with 48 CPU cores were connected over the 40GE link, running DiVinE. This experiment was performed only during GLIF2010.

3 Tuning

In order to achieve maximum throughput the server configuration needed extra tuning. Both servers ran CentOS 5.5, a free Linux distribution derived from Redhat Enterprise Linux. The OS choice was based on the list of supported OS' by Mellanox, the 40G network card manufacturer. The Dell R815 machine based on the AMD Magny Cours many-core platform required a number of optimizations at the kernel level and because these optimizations were already performed by the kernel maintainers of the Redhat distribution, during the demos the default distribution kernel was used, namely 2.6.18. On the other hand one can expect that a newer version of the Linux kernel will boost performance in a number of subsystems amongst which networking. Therefore, the Supermicro machines used a more recent kernel version. This kernel was compiled from the vanilla Linux sources obtained from kernel.org and no extra patching work was done. In the testing process before the demos, a similar kernel was installed on the Dell R815 but no notable network performance differences were observed yet the NUMA nodes were not correctly detected.

Multi queue support [11], also known as Receive Side Scaling (RSS), was enabled on the Mellanox cards by default. The mlx4_en driver creates

Figure 4: Kernel parameters used during the demos (taken from /etc/sysct.conf)

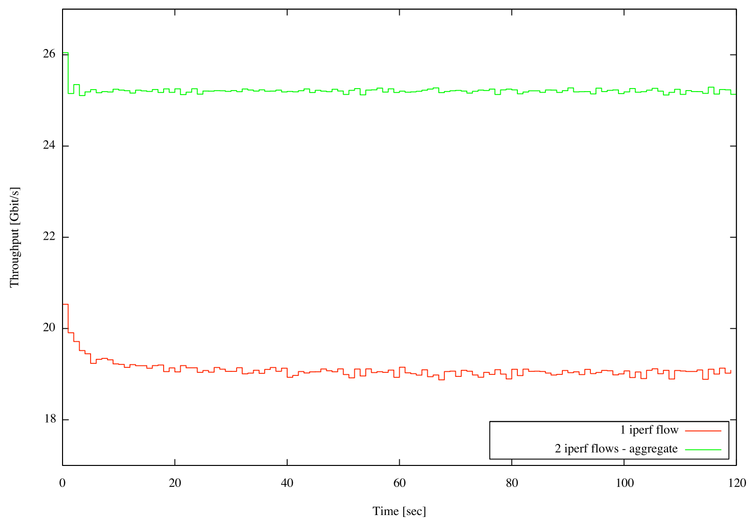
```
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.core.netdev_max_backlog = 25000
net.core.rmem_max = 102400000
net.core.wmem_max = 102400000
net.core.rmem_default = 67108864
net.core.wmem_default = 67108864
net.core.optmem_max = 67108864
net.ipv4.tcp_mem = 33554432 33554432 67108864
net.ipv4.tcp_rmem = 4096 33554432 67108864
net.ipv4.tcp_wmem = 4096 33554432 67108864
```

a number of receive (RX) queues or rings that is equal to the number of online CPUs (or cores). When a network packet is received it is placed in a RX queue and the network card triggers an interrupt signaling the CPU to handle the incoming data. Each incoming IP packet is sent to the appropriate RX queue according to an algorithm that hashes the IP information in the header of the packet. An IP flow to be always handled in-order and by the same core and also multiple flows will be distributed in a fair manner to the available cores. A similar feature is used for the transmission part, the driver creating a number of TX queues that each handle one IP flow at the sender side.

The networking and in particular the TCP settings of the Linux kernel were adapted for the distance and expected throughput as presented in figure 4. During the GLIF2010 demo the distance between the communicating servers was approximately 1650km which translated in 17ms of round trip time (RTT). This latency was consistent throughout the experiments with 0% link loss and 0.2ms jitter.

TCP window size scaling (`net.ipv4.tcp_window_scaling`) was turned on during all the experiments and demos and the congestion algorithm used was HTCP[7]. It was chosen for its fast increase of the window size to the optimal value and not for its congestion behavior as the 40Gigabit line was clear channel with no additional traffic interfering with the test data streams.

Figure 5: Local ***iperf*** *tests - single and multiflow*



4 Results

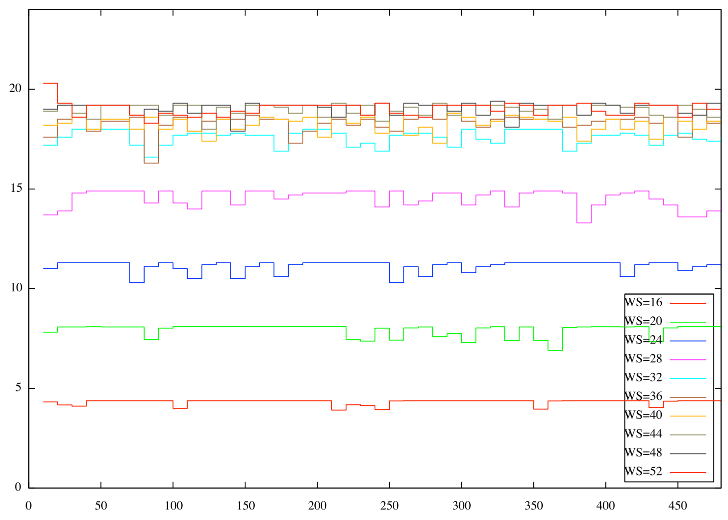
Initial tests were performed with 3 meters of multimode ribbon fiber connecting the two Supermicro servers mentioned in section 2. This was done to evaluate the performance of the setup under ideal conditions and to discover potential issues and optimal tuning parameters. In figure 5 we present the throughput of this setup when using a single flow `iperf` running for a timespan of 5 minutes. A similar experiment was performed with two `iperf` flows. Due to the multiqueue features mentioned in the previous section, this maximized the use of the network card. Increasing the number of flows didn't further increase the aggregated throughput.

The card uses a PCI-E 2.0 8x interface which allows of a maximum 8 x 5GT/s. This data rate translates into 32 Gbps of useful data as the PCI-E protocol uses a 8/10bit encoding scheme. The maximum theoretical 32Gbps data rate is further decreased by the inherent overhead of the PCI-E protocol and of the network protocols (Ethernet, IP) that support the communication. Therefore, the application transfer rate observed using `iperf` comes close to the theoretical maximum achievable and we conclude that in our tests the PCI-E interface becomes saturated, this being the maximum rate at which the card can send or receive data.

Once the setup was tested under ideal conditions we moved to performing the demonstration over the WAN infrastructure. The GLIF2010 demo involved a series of experiments that targeted throughput and protocol behavior.

One of the experiments consisted in validating the behavior of the TCP protocol over the long distance high performance link. Due to the 17ms RTT

Figure 6: Influence of the TCP Window size - GLIF2010 - Single iperf flow

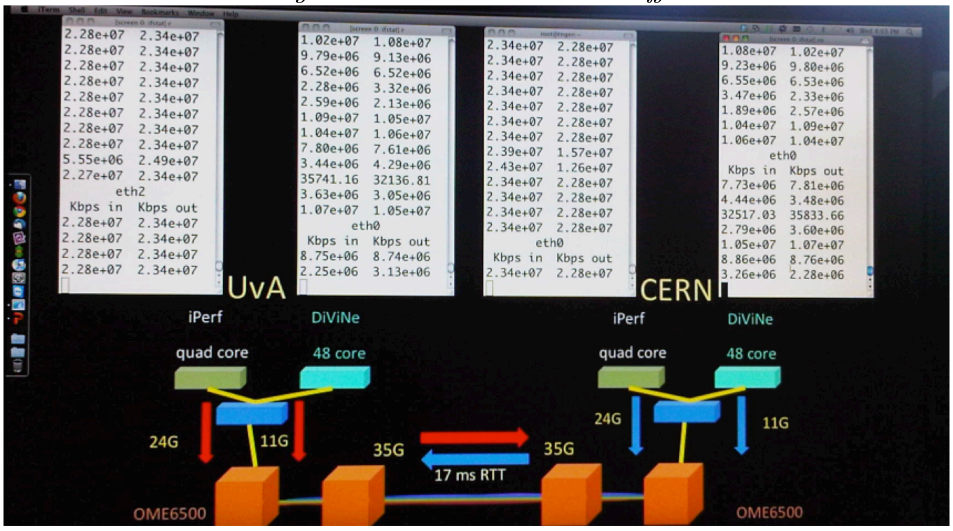


the TCP window size needs to be adjusted in order to achieve the maximum throughput. The ideal TCP window size is obtained by calculating the Bandwidth Delay Product or BDP. Assuming that the speed of the link is 19Gbps the value of the window size would be $WS = 19Gbps \times 17ms = 40.38MB$. Figure 6 presents the results of multiple throughput tests where the window size varied from a low value to the calculated optimal value and beyond. The maximum transmission rate is achieved as expected, around the optimal value. Further increasing the window size does not affect the throughput rate in any way. This confirms that the TCP protocol and its current Linux implementation is still well suited for speeds beyond 10Gbps.

During GLIF2010 in parallel with the throughput tests we ran the DiVinE application obtaining peaks of 11Gbps of aggregated traffic. The application creates a number of TCP connections in order to perform message passing. When using large states, DiVinE is not very sensitive to latency as each MPI process performs local computation for the length of a processing cycle (iteration) and then exchanges state with other processes which are either local (on the same machine) or remote. Given the relatively short availability of the WAN link we were unable to perform a very thorough evaluation of DiVinE. The availability of a higher number and faster CPUs would have enabled higher aggregated throughput, as DiVinE is able to scale beyond 94 cores. The throughput rate observed varied throughout the experiments as it depended on the application state.

In figure 7 we present a snapshot of live traffic statistics running during the GLIF2010 demo. The speeds were measured with a 1 second sample time.

Figure 7: GLIF 2010 Live traffic



```
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
5.55e+06  2.49e+07
2.27e+07  2.34e+07
       eth2
 Kbps in  Kbps out
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
2.28e+07  2.34e+07
```

```
1.02e+07  1.08e+07
9.79e+06  9.13e+06
6.52e+06  6.52e+06
2.28e+06  3.32e+06
2.59e+06  2.13e+06
1.09e+07  1.05e+07
1.04e+07  1.06e+07
7.80e+06  7.61e+06
3.44e+06  4.29e+06
35741.16  32136.81
3.63e+06  3.05e+06
1.07e+07  1.05e+07
       eth0
 Kbps in  Kbps out
8.75e+06  8.74e+06
2.25e+06  3.13e+06
```

```
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.39e+07  1.57e+07
2.43e+07  1.26e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
2.34e+07  2.28e+07
       eth0
 Kbps in  Kbps out
2.34e+07  2.28e+07
```

```
1.08e+07  1.02e+07
9.23e+06  9.80e+06
6.55e+06  6.53e+06
3.47e+06  2.33e+06
1.89e+06  2.57e+06
1.04e+07  1.09e+07
1.06e+07  1.04e+07
       eth0
 Kbps in  Kbps out
7.73e+06  7.81e+06
4.44e+06  3.48e+06
32517.03  35833.66
2.79e+06  3.60e+06
1.05e+07  1.07e+07
8.86e+06  8.76e+06
3.26e+06  2.28e+06
```

5 Performance Analysis

AMD Magny Cours It is interesting to note that at GLIF2010 the throughput of the link was not maximized, only about 80% being used. This was because the 48 core AMD server was unable to reach the same I/O performance as the 8 core Intel server even when using plain `iperf` with multiple threads. To explain this we need to look at the machine architectures used in both servers. While both are NUMA (Non-Uniform Memory Architecture), each CPU having its own local memory, the AMD server is more complex as it has six times more cores and uses two PCI-Express bridges. In [3] the authors present multiple possible CPU I/O topologies, each suited for either I/O performance or low latency (small diameter of the CPU network) the later being the recommended setup for server manufacturers. We assume that the I/O topology of the AMD Magny Cours platform used in the Dell R815 server is the low latency variant presented in figure 8 . Two of the CPU packages are not connected directly to the PCI-E bridges but through the other two CPU packages - in figure 8 P5-P7 and P4-P6.

In figure 10 we present the node matrix as listed by `numactl --hardware`.

To explore this asymmetry and see its impact on I/O performance, we performed a simple experiment where a Supermicro Intel server and a Dell AMD server were connected back to back using 40GE. The Intel server was configured as an `iperf` server (receiver) and the AMD server as a single threaded sender with the `iperf` process forced to run on a specific core (figure 11). We used the Intel server as a receiver because in our experiments it could receive data at rates higher than the AMD server could send. This allowed us to change parameters on the AMD server without worrying that

Figure 8: Simplified view of the AMD Magny Cours IO Architecture - Dual HexaCore Quad Socket

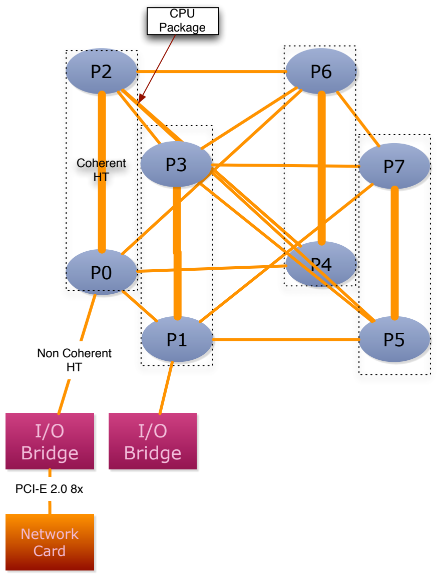


Figure 9: Simplified view of the Intel Nehalem IO Architecture - Quad Core Dual Socket

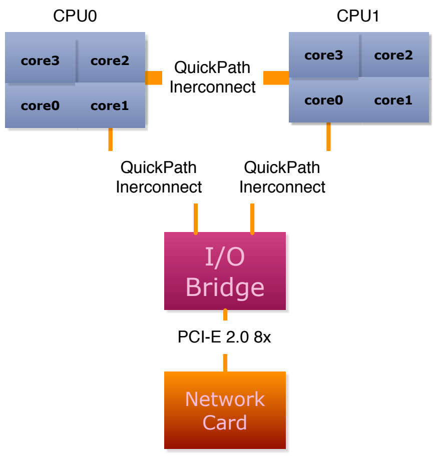
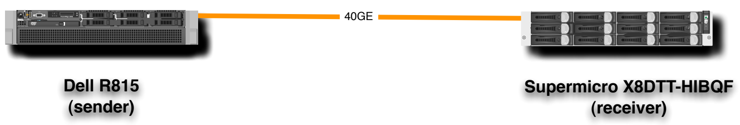


Figure 10: CPU node distance as listed by `numactl --hardware` for Dell R815

```
node distances:
node   0   1   2   3   4   5   6   7
  0:  10  16  16  22  16  22  16  22
  1:  16  10  16  22  22  16  22  16
  2:  16  16  10  16  16  16  16  22
  3:  22  22  16  10  16  16  22  16
  4:  16  22  16  16  10  16  16  16
  5:  22  16  16  16  16  10  22  22
  6:  16  22  16  22  16  22  10  16
  7:  22  16  22  16  16  22  16  10
```

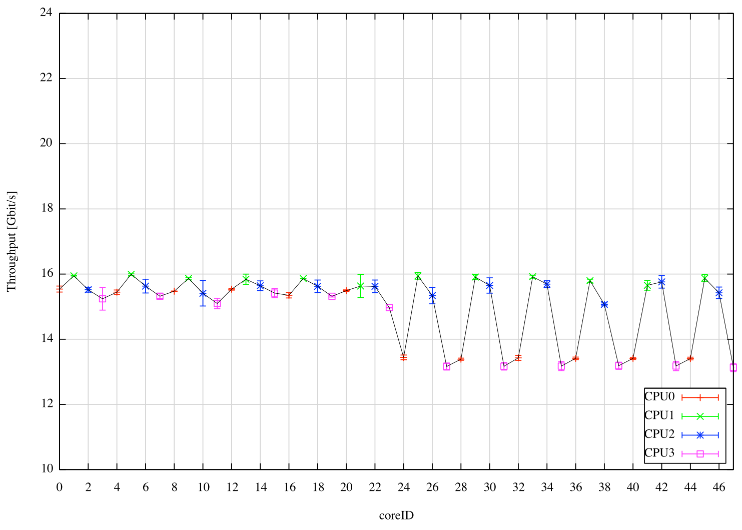
Figure 11: Simple experimental setup to determine I/O Performance - Dell AMD



the receiver would not be able to cope with the incoming data.

Setting the CPU affinity for the `iperf` process was done using the `numactl` utility. This was repeated for each of the 48 cores available for a period of 5 minutes. By analyzing the results of the average per core throughput(figure 12) we can observe that some cores perform better than the others. When mapping the coreID over the given CPU topology, a correlation between the core location I/O stands out. The mapping of the cores to CPU packages on our setup followed the rule: if one core has id i then it is located on package i *modulo* 4, so for example core 12 is located on package 0. The graph clearly shows that even for single threaded applications which do heavy I/O traffic the placement of the process on specific cores heavily impacts the overall performance. In a multithreaded setting this affinity issue is overcome by the overall interconnect bandwidth to the PCI-E bridge. Our tests showed a maximum of 20Gbit/s when running `iperf` with multiple threads and a similar Supermicro Intel-equipped receiver. Because the Supermicro Intel machine can receive more than 20Gbit/s, as shown in the previous section, we believe that the bottle neck is caused by the limitations of the I/O architecture used in the Dell AMD server. This means that for single threaded `iperf` the bottle neck lies closer to the CPU cores. It is

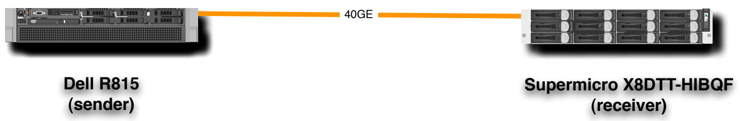
Figure 12: Average throughput of iperf with core pinning



difficult to pinpoint this without extensive investigation at the OS level or even deeper at the chip level, yet we suspect a number factors that together cause this behavior. The asymmetry of the I/O topology adds latency to any I/O transfer done by the cores not directly connected to CPU1. Coming back to the network adapter, it uses multiple transmit queues to distribute the data flows evenly to the cores. In our experiments the interrupts were handled mostly by core 1, so for each packet a DMA request would have to take place in order to take the data from memory and send it to the network adapter. In terms of interconnect bandwidth, the Hypertransport links are not equally shared between the cores[3] and even if the AMD Magny Cours has a novel cache coherency mechanism, we can expect that still some of the traffic is cache coherency related, leaving less available bandwidth for data intensive applications. Unfortunately, proving this hypothesis is beyond the scope of this paper yet in [6] the authors suggest that cache coherency can have an important impact on network traffic.

Intel Nehalem Intrigued by the performance variations of the AMD Magny Cours, we were curious to see if the less complex Intel Nehalem would present similar issues. Because we could not create a stable receiver that could receive at a higher rate than the sender, like in the case of the AMD Magny Cours, we connected to Supermicro Intel servers back to back to cover all the possible cases of application to core mapping(figure 13). Because the multi queue receive and transmit (also known as Linux scalable I/O[9]) mechanism uses a hash based on the IP header (source&destination IP and port) we patched the `iperf` source code so that for any single flow run, the same source port would be used. This assured that a TCP flow

Figure 13: Simple experimental setup to determine I/O Performance - Supermicro Intel

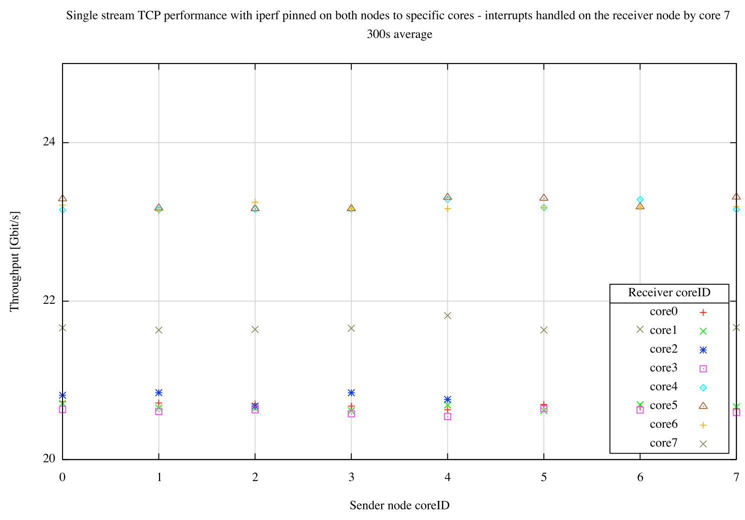


would always be handled by the same core as on a Linux system if the source port is not specified by an application, the OS picks a random one from the ephemeral range.

With the use of `numactl` the `iperf` server and client process were forced to be executed on only one specific core. This resulted in 64 measurements each lasting for 5 minutes. Figure 14 presents the averaged throughputs mapped by core. The core to CPU mapping is more simpler and more intuitive than in the AMD case: the first four cores belong to CPU0 while the remaining four to CPU1. A clear pattern emerges: when the application runs on the same CPU as the core that handles the interrupts the performance increases by approximately 10%. Not surprisingly the same performance increase is not achieved when `iperf` and the interrupts are handled by the same core, the throughput increasing with only 1 Gbit/s. This most likely is caused by the extra time needed by the CPU core to change context and handle the interrupts. The location of the application on the sender node does not influence the performance in a very significant way and therefore we conclude that the limitation is at the receiver node.

Network Drivers A deeper understanding of the processes that take place in the kernel on the receiver was needed and therefore, we decided to inspect the mlx4_en driver source code [5]. As mentioned before the driver has multiqueue support and by default it creates a number of receive queues equal to the number of online CPUs, as seen by the kernel. While the driver is more complex and has support for advanced features like TCP offloading and Generic Segmentation Offload (GSO), we will not go into the details of these. The receive procedure follows a standard NAPI network driver approach[4]: a socket buffer structure , also named `skb`, is pre allocated in order to handle the incoming data from the network. When the data is received from the network, it is buffered on the network card and the CPU is notified via an interrupt about its arrival. The CPU then polls the network card and issues a DMA transfer from the network card to the memory address stored in the `skb` structure. From here on, the kernel decides what to do with the received packet, as it gets passed to the upper layers of the networking stack. In the case of a packet destined to a local application, the payload is copied to the socket buffer belonging to the application. This

Figure 14: Average throughput of iperf with core pinning



means that the data is copied at least twice from the moment it is received by the network adapter, once to from the network card to kernel space and from there to user space . In the case of a NUMA machine the `skb` structure is allocated on the local node , the one that handles the receive queue from which the data originates. Obviously, if the application is running on a different node an inter-node transfer is needed and the topology, the memory and interconnect bandwidth and latency affect the overall performance. Our measurements clearly show degraded performance when inter-node transfer is involved. Alternative technologies, like Infiniband, offer zero copy and implicitly higher performance, but they break compatibility with the existing protocols or they require extra software to be installed or included with the user applications.

In our tests we assumed that the Mellanox ConnectX 2 40GE drivers have an optimal behavior and their performance is not affected in any way by the underlying architecture. It should be noted that because the ConnectX2 40GE is derived from the ConnectX2 10GE it uses the same driver and supports the same kernel module parameters.

We believe that our measurements give a good insight on the capabilities of the current server hardware when faced with I/O intensive applications. Given the current state of the networking stack on the Linux OS, it is very probable that similar performance inconsistencies will occur with future multicore architectures and next-gen network adapters and the approach presented in this paper could be used to investigate them.

6 Conclusion

In this report we have presented the results of two technological demonstrations performed at GLIF2010 and Supercomputing 2010 and a short analysis of the performance of 40GE when combined current state-of-the-art server hardware . The GLIF2010 demo was, to our knowledge, the first time when a 40GE signal was transported over more than 1600KM of fiber. We investigated the current performance of the 40GE technology and presented an in-depth analysis of the results. We conclude that given the current PCI-E 2.0 interface and CPU micro-achitectures the 40GE standard is not yet used to its full potential. A modern server can not fully utilize the available bandwidth and while it can saturate the I/O bus this leaves little room for a real application running on the machine. When using a more capable machine the I/O limitations stand out even more giving 40GE little advantage over other existing technologies. With the introduction of the new PCI-E 3.0 computer interface in late 2010 together with faster CPUs, we expect that the full potential of 40GE will be unleashed. With the current available technology, the bottleneck has moved one level higher, from the network to the computer's internal interconnect. At the moment, the number of applications that can leverage this bandwidth increase is still limited. New research and applications are needed to promote 40GE from an exotic network protocol to a commodity interconnect.

Acknowledgments The GLIF 2010 and Supercomputing 2010 demos would have not been possible without the support of : Erik-Jan Bos, Gerben van Malenstein, Roeland Nuijts - SURFnet, David Yeung, Jan-Willem Elion, Harry Peng, Kevin McKernan, Martin Bluethner, Rodney Wilson - CIENA, Kees Verstoep, Henri - Bal Vrije Universiteit Amsterdam, Erez Cohen, Bill Lee - Mellanox, CERN, CineGrid.

References

[1] Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications amendment 4: Media access control parameters, physical layers and management parameters for 40 gb/s and 100 gb/s operation. *IEEE Std 802.3ba-2010 (Amendment to IEEE Standard 802.3-2008)*, pages 1 –457, 22 2010.

[2] J. Casazza. First the tick, now the tock: Intel microarchitecture (nehalem). *Intel Corporation*, 2009.

[3] Pat Conway, Nathan Kalyanasundharam, Gregg Donley, Kevin Lepak, and Bill Hughes. Cache Hierarchy and Memory Subsystem of the AMD Opteron Processor. *IEEE Micro*, 30(2):16–29, March 2010.

[4] Linux Foundation. napi article on linuxfoundation.org. `http://www.linuxfoundation.org/collaborate/workgroups/networking/napi`. [Online; accessed 20-February-2011].

[5] Mellanox Inc. Mellanox mlx4 en driver source code hosted on lxr.linux.no. `http://lxr.linux.no/linux+v2.6.37.1/drivers/net/mlx4/`. [Online; accessed 20-February-2011].

[6] Amit Kumar and Ram Huggahalli. Impact of cache coherence protocols on the processing of network traffic. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 40, pages 161–171, Washington, DC, USA, 2007. IEEE Computer Society.

[7] D. Leith and R. Shorten. H-TCP: TCP for high-speed and long-distance networks. In *Proceedings of the 2nd Workshop on Protocols for Fast Long Distance Networks*, Argonne, Canada, 2004.

[8] Jason Maassen, Kees Verstoep, H.E. Bal, Paola Grosso, and C. de Laat. Assessing the impact of future reconfigurable optical networks on application performance. *IPDPS '09 Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, 2009.

[9] Niels Provos, C. Lever, and S.N. Alliance. Scalable network I/O in Linux. In *Proceedings of the USENIX Annual Technical Conference, FREENIX Track*, volume 19, 2000.

[10] Kees Verstoep, H.E. Bal, J. Barnat, and L. Brim. Efficient large-scale model checking. In *IEEE International Symposium on Parallel&Distributed Processing*, number 201. IEEE, 2009.

[11] Z. Yi and PJ Waskiewicz. Enabling Linux network support of hardware multiqueue devices. In *Proc. of the 2007 Linux Symposium*, pages 305–310.