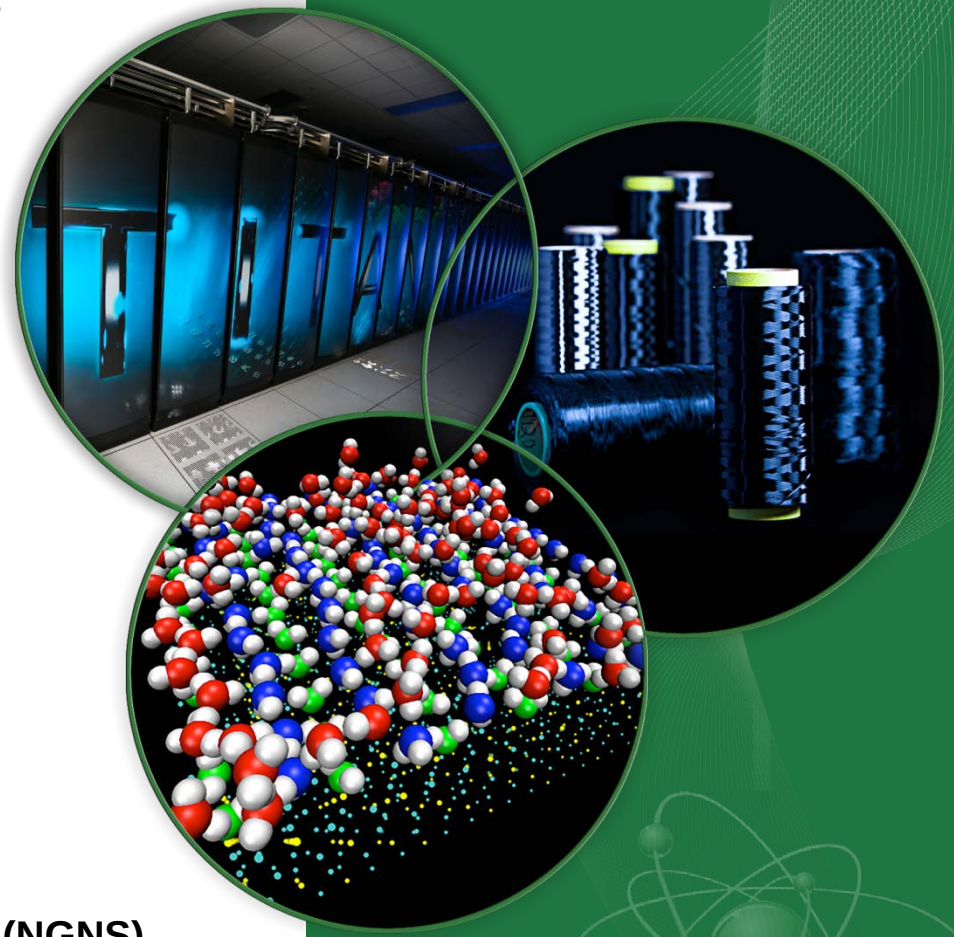
Synthesis of Source-to-Sink High-Performance Flows



Nagi Rao, Brad Settlemyer, Josh Lothian

Oak Ridge National Laboratory

Raj Kettimuthu, Venkat Vishwanath

Argonne National Laboratory

Chase Wu

University of Memphis

ASCR Next-Generation Networks for Science (NGNS)

Principal Investigators Meeting

September 16-17, 2014, Rockville, MD

ORNL is managed by UT-Battelle
for the US Department of Energy

Argonne NATIONAL LABORATORY

THE UNIVERSITY OF MEMPHIS

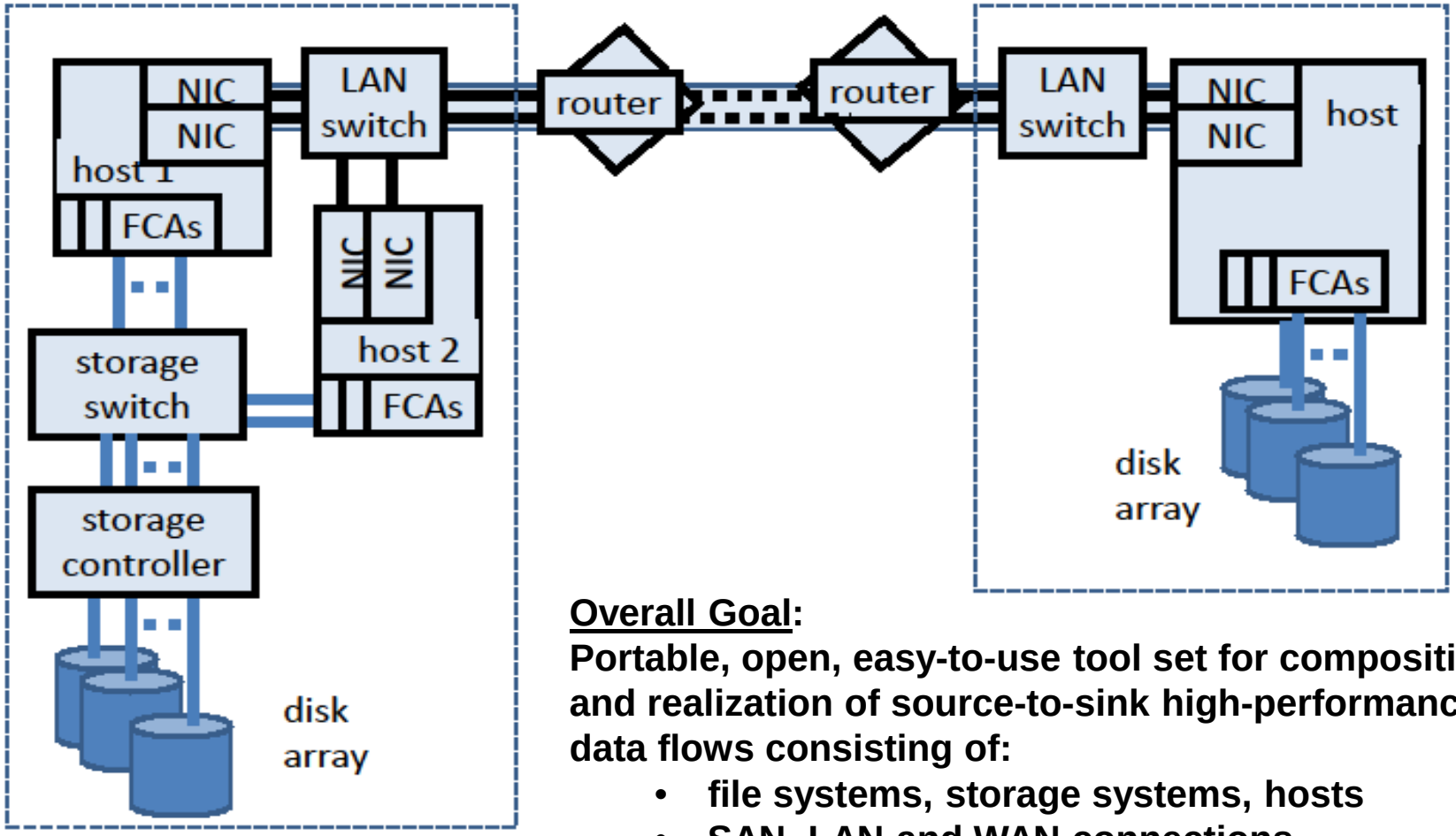
OAK RIDGE National Laboratory

Outline

- Project Overview
- Test-Beds and Flow Composition Tests
- Network Interfaces, APIs
- XDD: Disk-to-Disk Transfers
- Transport profiles and protocols

Project Title:
Technologies and Tools for Synthesis of Source-to-Sink High-Performance Flows
Collaboration: ORNL, ANL and UM

Multiple Multi-Bus Host-to-Host Disk/File Transfers



<u>Overall Goal</u>:
Portable, open, easy-to-use tool set for composition and realization of source-to-sink high-performance data flows consisting of:

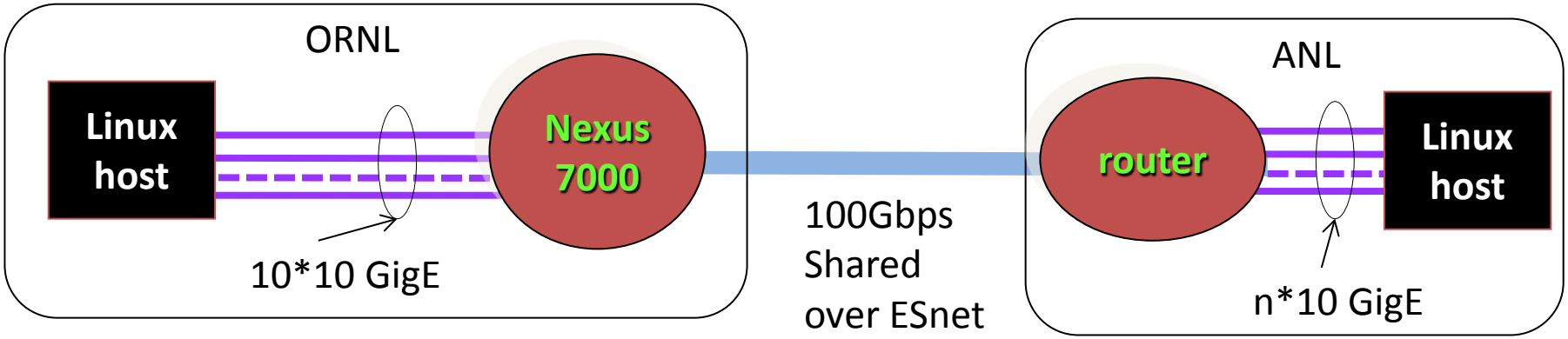
- **file systems, storage systems, hosts**
- **SAN, LAN and WAN connections**

Significant leverage from DOD projects and testbed

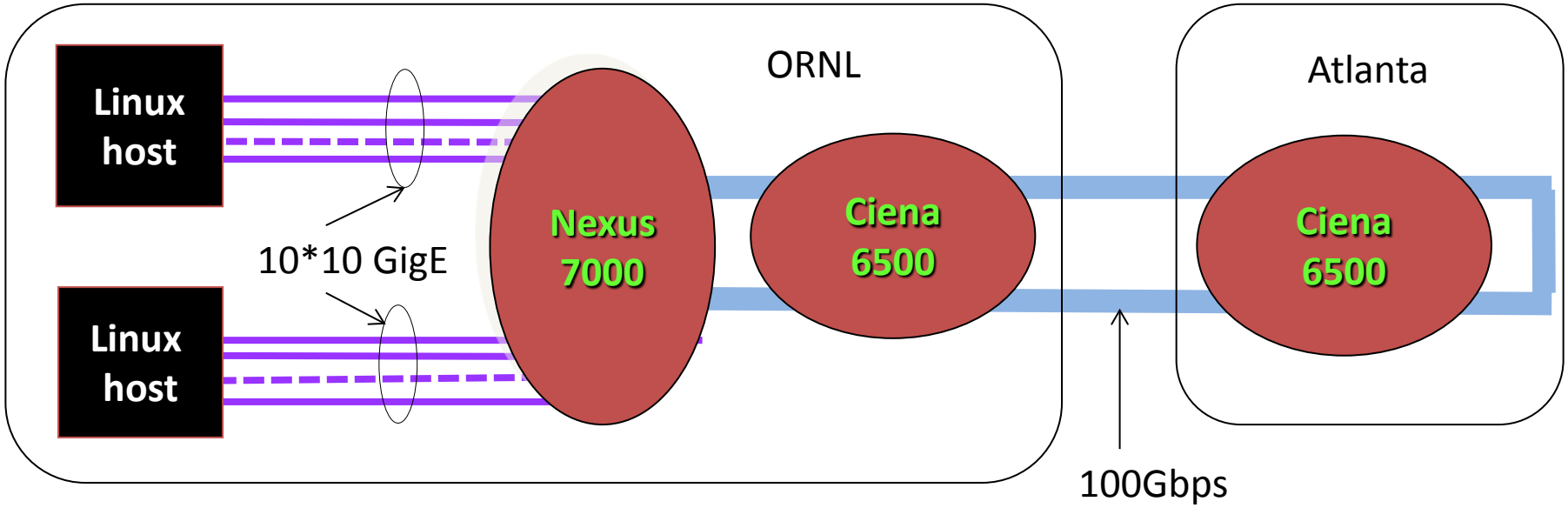
Project Tasks Overview

ORNL	**Network Infrastructure and Path Provisioning**: (Rao) Establishing static and dynamic local and long-haul connections. **Flow Composition Methods**: (Rao) Synthesizing source-to-sink flows from constituent storage-network flows. **Storage and File Systems Methods**: (Settlemyer) Optimizing storage flows and their transition to network flows, generating storage throughput profiles, and specialization for Lustre file system. **Multi-Core-Bus Host Systems Optimizations:** (Lothian) Optimal I/O card placements, IRQ de-confliction, thread and memory assignments, and developing scripts.
UM	**Transport and Network Profiles tasks**: (Wu) Generation of throughput profiles for UDP, TCP, and RDMA protocol; and testing in pair and integrated configurations.
ANL	**User and Application Tools**: (Kettimuthu) Integration of l modules of XDDC versions, developing interfaces and APIs, and packaging and delivery of open-source transport tools within the team and to broader community. **Integrated Test Designs and Execution**: (Vishwanath) Designing and coordinating integrated tests for XDDC versions, and communicating the feedback for improving the modules, and specialization for GPFS file systems.

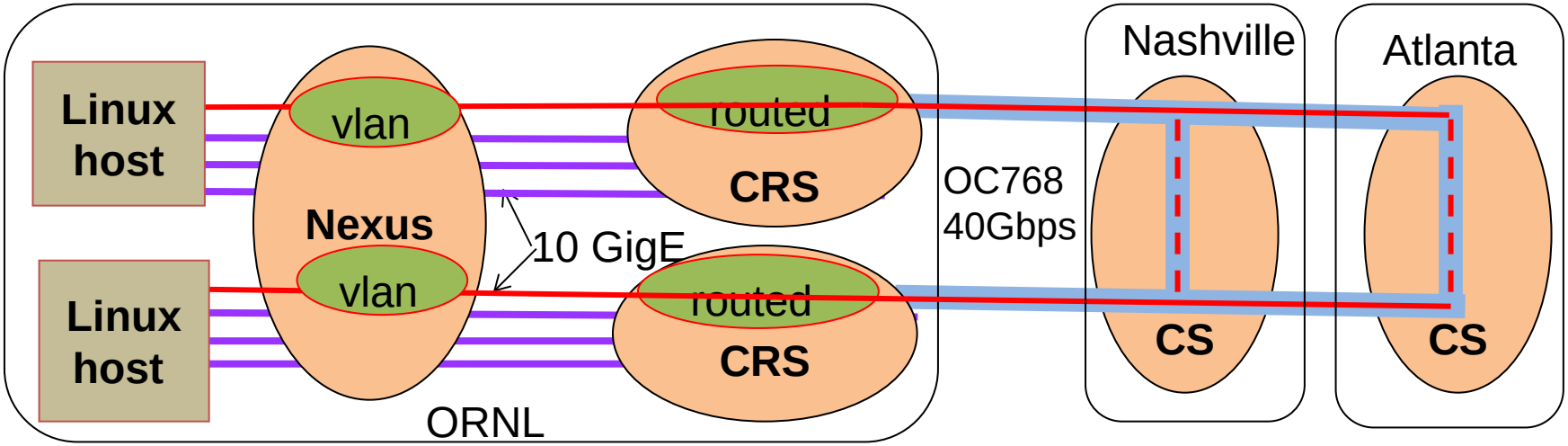
ORNL-ANL Connection: Under Progress



100Gbps Host and Edge Configurations: leveraged DOD testbed



40Gbps Flow Composition: physical connection



Four separate IP streams

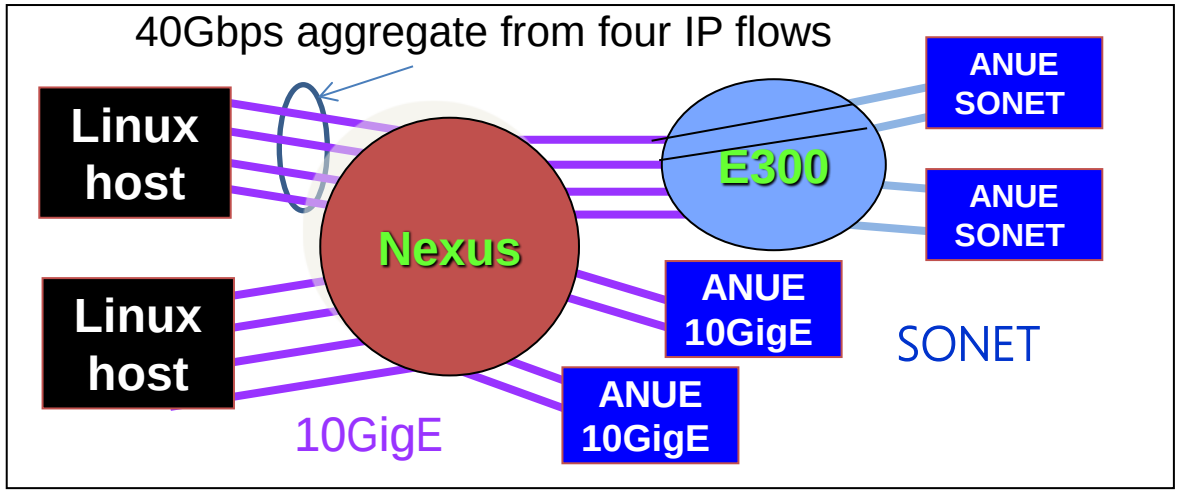
Achieved ~38 Gbps

loop	rtt ms	flow 1 Gbps	flow 2 Gbps	flow 3 Gbps	flow 4 Gbps	total Gbps	peak percent
local	0.17	9.89	9.89	9.88	9.89	39.55	98.88%
Nashville	8.24	9.03	9.65	9.55	9.73	37.96	98.85%
Atlanta	19.20	9.69	9.25	9.33	9.65	37.92	98.75%

TABLE I
THROUGHPUT MEASUREMENTS ON PHYSICAL TESTBED CONNECTIONS.

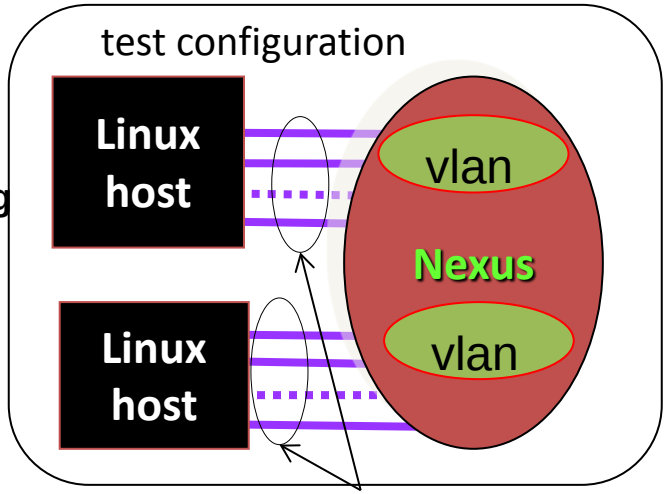
40Gbps Emulation Testbed used for testing

- arbitrary connection lengths, not feasible over physical infrastructure
- corrections for differences from physical connections
- differential regression method



100G Flow Composition

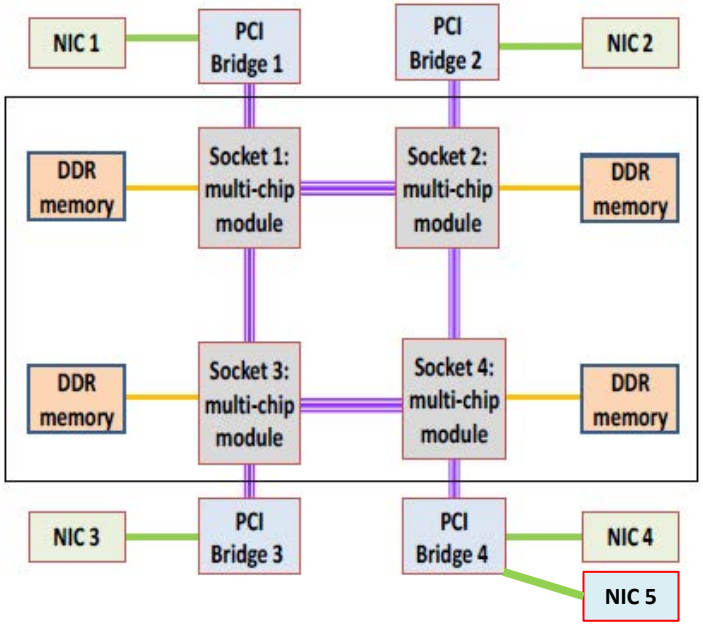
- Ten separate IP streams with separate IP addresses
  - Five dual-port Myrinet 10GigE NIC connections to Nexus 7000 Separate VLAN connections for ten10GigE connections
- Host optimization – Buffer tuning, IRQ de-confliction/balancing
  - Hosts have 48 processor cores
  - Buffer tuning enabled ~9.89 Gbps per flow NIC Interrupts handled by assigned processor cores in IRQ map
    - default IRQ map has overlaps, changed on reboots
    - detect overlaps and script IRQ maps with de-confliction
- Achieved ~95 Gbps locally and 72Gbps on ORNL-ATL link



10*10 GigE
48-core HP servers

Host Profiling: Automated several components

- Inventory profile: CPUs, cores; Sockets, dyes, and PCIbusses; NICs, HCAs, HBAs
- Throughput profiling:
  - Network throughput profile
  - IO and file throughput profile
  - composite profile
- Host Configuration:
  - IRQ de-confliction; NIC/HCA/HBA placement
- Storage and file system configuration:
  - HBAs/HCAs; file systems, #threads, NUMA
- Transport selection configuration:
  - TCP/UPD; #flows/threads; NUMA placement



Aggregate TCP throughput – IRQ de-confliction/balancing

[screen 0: bash] nrao@bohr03:~/100G/bohr3-4

File Edit View Search Terminal Tabs Help

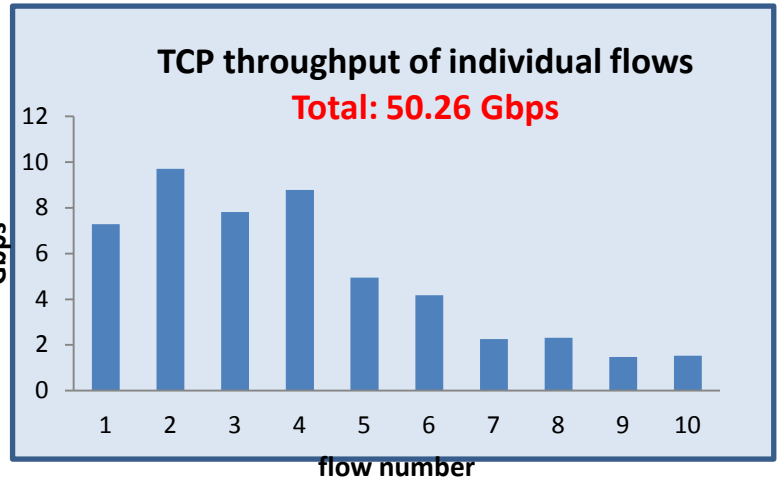
[screen 0: bash] nrao@bohr03:~/100G/bohr3-4 | 3nr@pcwf02548:~

```
Socket 0 PUs: 0  4  8  12  16  20  24  28  32  36  40  44
Socket 1 PUs: 1  5  9  13  17  21  25  29  33  37  41  45
Socket 2 PUs: 2  6  10  14  18  22  26  30  34  38  42  46
Socket 3 PUs: 3  7  11  15  19  23  27  31  35  39  43  47
 Host NIC Summary:
```

IRQ

IP	name	mac	socket	irq	socket:pu	conflict
10.101.4.5	b05-10-a	00:60:dd:45:e4:14	0	226	3:11 -	1
10.101.9.5	b05-10-b	00:60:dd:45:e4:15	0	227	2:14 -	1
10.101.3.5	b05-7-a	00:60:dd:45:dc:46	3	232	3:7 -	0
10.101.8.5	b05-7-b	00:60:dd:45:dc:47	3	233	3:3 -	0
10.101.2.5	b05-4-a	00:60:dd:45:ee:e4	2	230	2:6 -	0
10.101.7.5	b05-4-b	00:60:dd:45:ee:e5	2	231	2:2 -	0
10.101.1.5	b05-2-a	00:60:dd:45:e3:74	1	228	2:10 -	1
10.101.6.5	b05-2-b	00:60:dd:45:e3:75	1	229	1:13 -	0
10.101.0.5	b05-1-a	00:60:dd:45:dc:38	1	224	0:4 -	1
10.101.5.5	b05-1-b	00:60:dd:45:dc:39	1	225	3:15 -	1

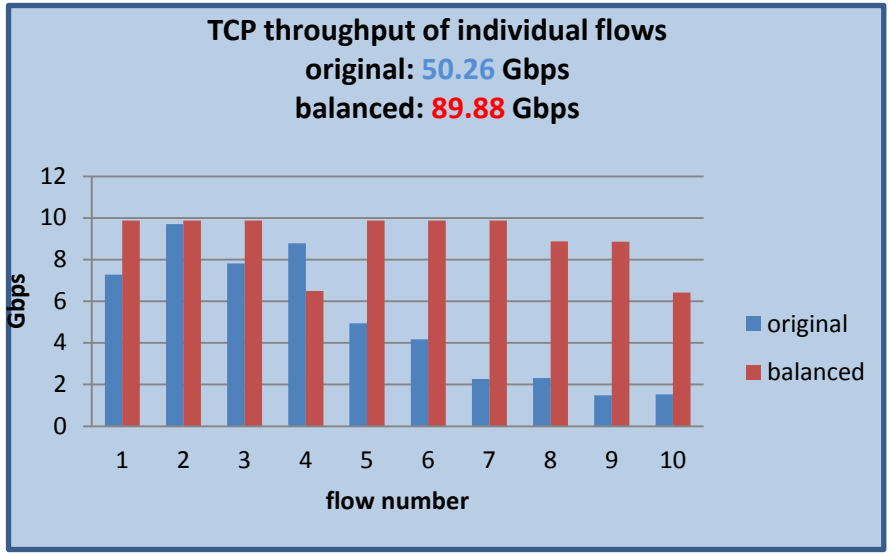
```
[nrao@bohr05 host_profile]$
```

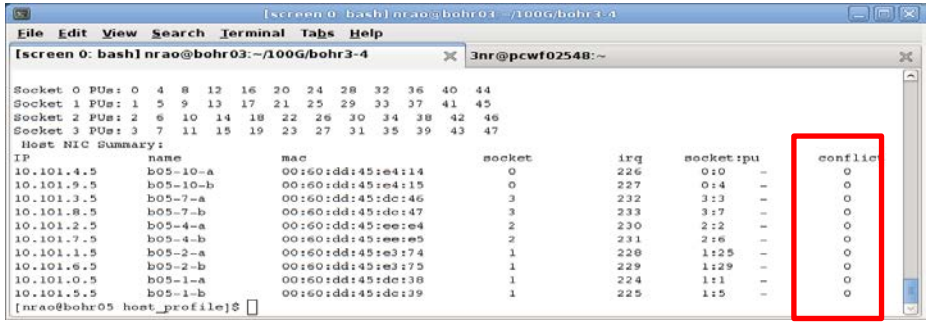


Utilizes outputs from ifconfig, lspci, lstopo and irq assignments

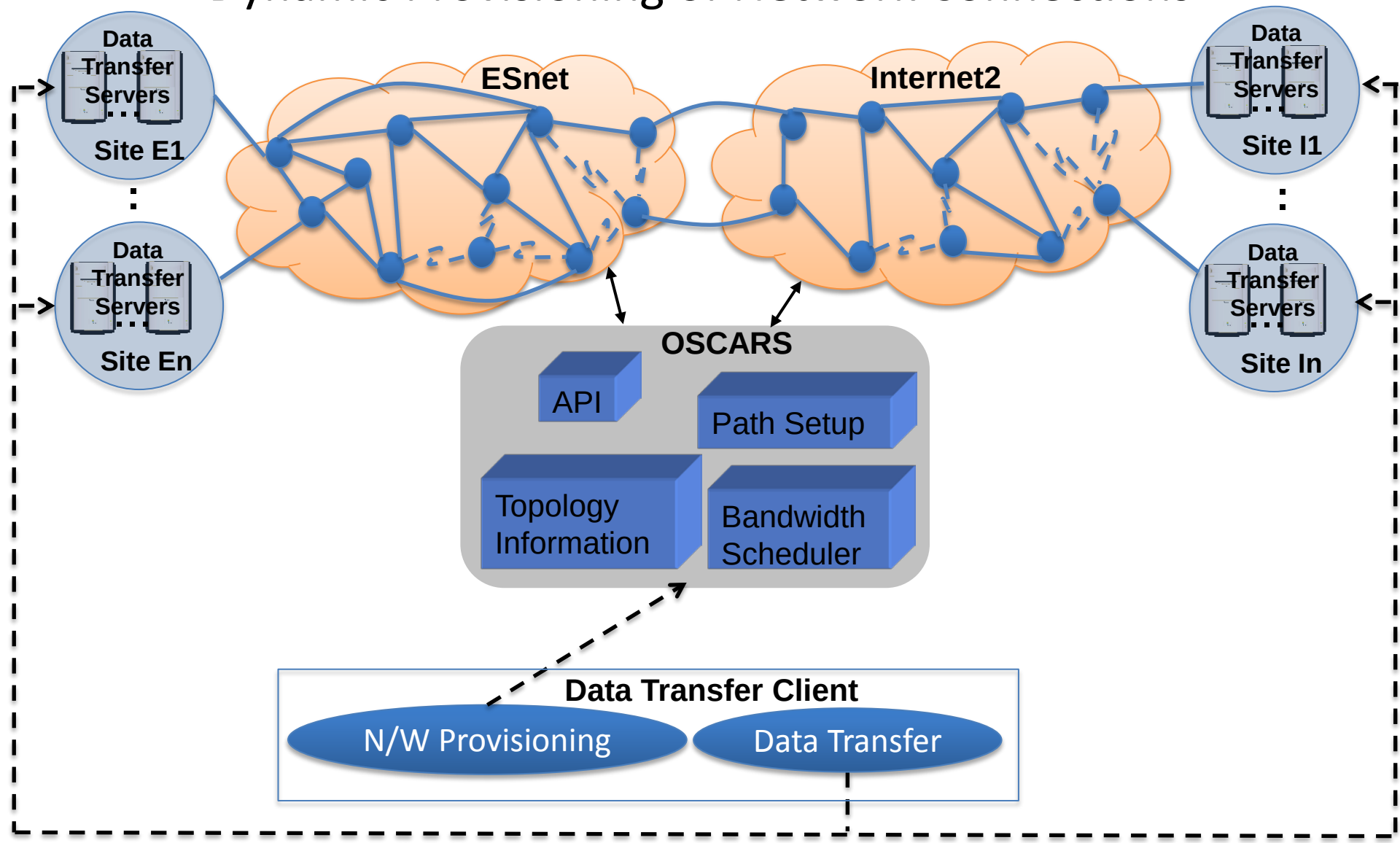
- provides summary: ip addr, name, MAC, socket, irq
- detects possible irq bottlenecks

Next steps: eliminate dependence on ifconfig, utilize libdnet





Dynamic Provisioning of Network Connections



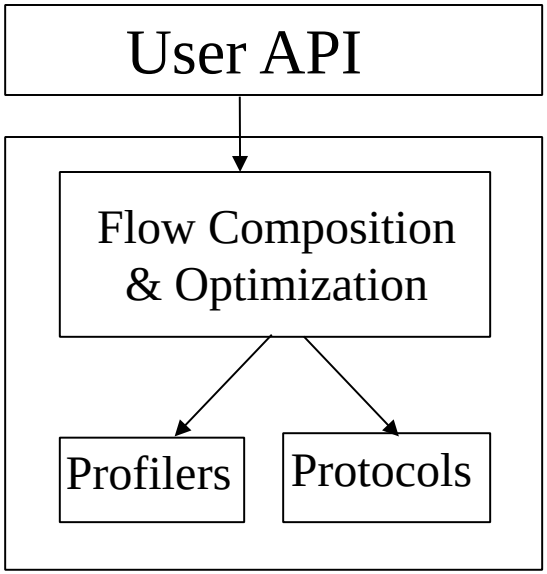
Challenges

- Authentication to OSCARS is based on Grid security
- Requires a certificate issued by OSG CA (replacement for DOEGrids CA) or some other well-known CA
- Had to contact the network administrator at both sites to get the certificate added to the authorized user database
- Though it is a one-time step, it makes the process complex
- Once the above steps are done, circuit establishment is straightforward
- But the setup delay is about 1 minute

Network Services Interface

- Network Services Interface - An open standard for dynamic circuit service interoperability.
- Reservation made in two phases.
- Phase I:
  - Resource availability is checked
  - If available resources are held, otherwise reservation request fails
- Phase II:
  - Requester either commits or aborts a held reservation
  - Timeout: If a requester does not commit for a certain period of time, a provider can timeout.
- Useful for co-reserving multiple resources.

User Tools and API



- Tools framework is a C library
- Simple user interface – connect, send, receive, close.
- Framework manages interactions between modules.
- 'Profilers' – network-profiling tools such as TPG (UM), host profiling tools (ORNL and ANL), and storage profiling tools (ORNL).
- 'Protocols' – TCP, UDP based.
- 'Flow composition & Optimization' – Network & storage flows, protocol tuning, parameter selection.

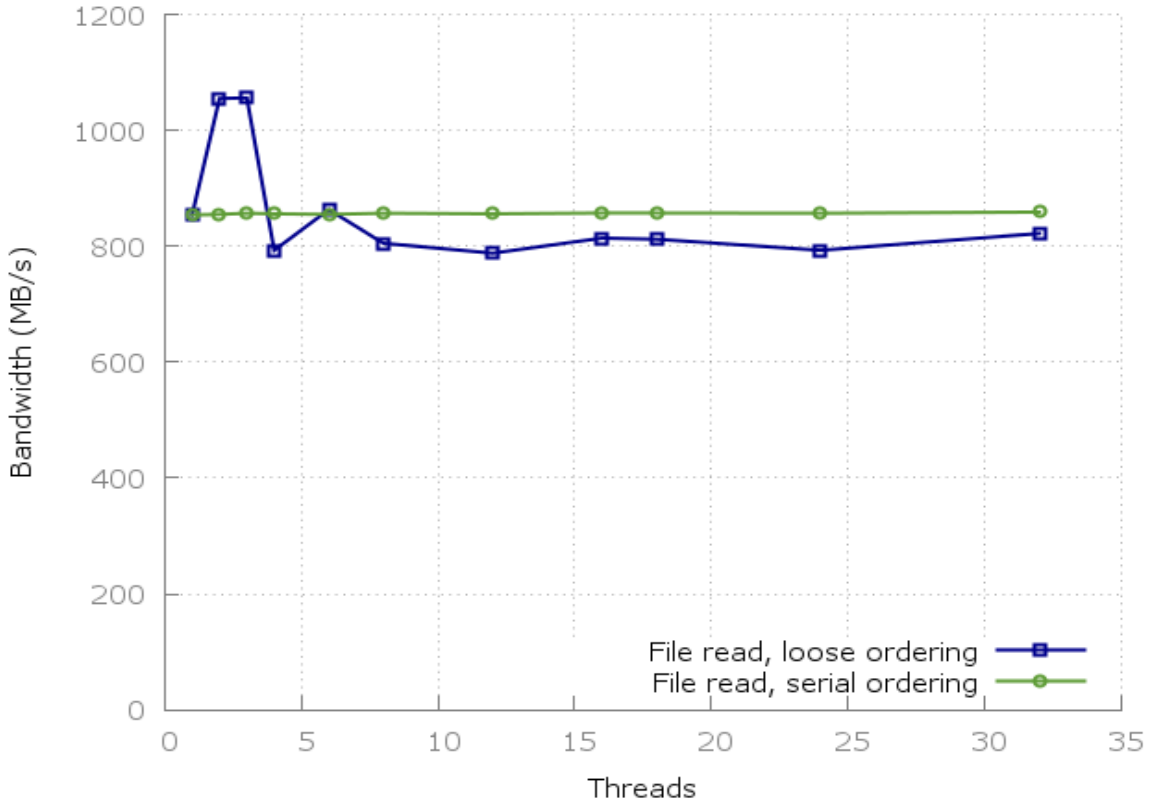
XDD: Disk-to-Disk Data Transfer Tool

- XDD started as a file and storage benchmarking toolkit
  - Many storage tuning options
- Added a network implementation, python frontend, multi-host coordination, and NUMA tuning options
  - One major feature is XDD can leverage multiple NIC's from a single process to better schedule storage access
- Adding profiling capabilities to sweep relevant tuning parameters
  - Identify storage performance modes for pairing with network performance profiles

Storage Profiling Parameter Examples

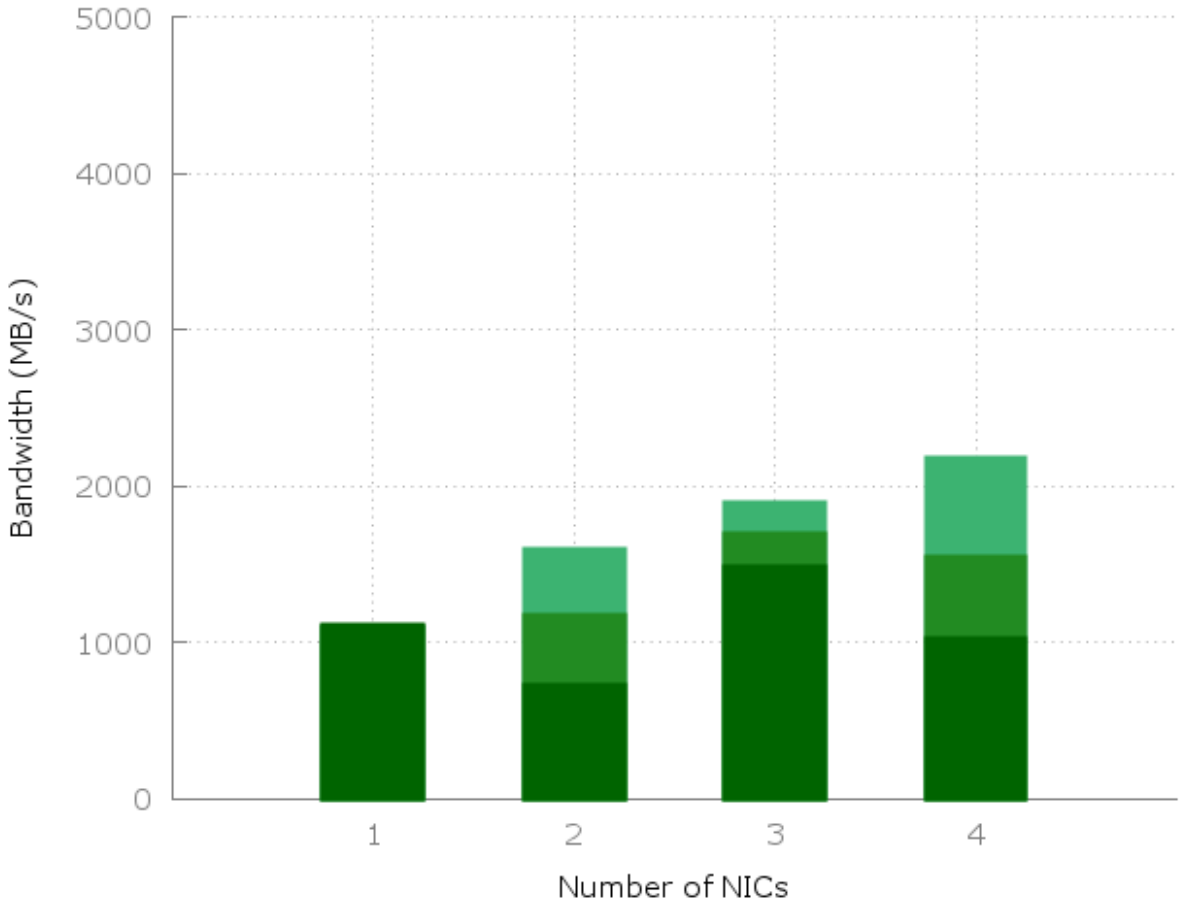
- Issue Order “None”
  - Each thread issue I/O’s without any ordering (often generates unstable performance)
- Issue Order “Serial”
  - Each request is issued in file order (high synchronization costs, but perfect streaming workload)
- Issue Order “Loose”
  - Each request is assigned in order, but allowed to complete in any order

Profiling Example



Opteron host, 3x 4Gbps FC HBA, 3x 7+1 Storage Array, 32MB requests

Multi-NIC Transfer with NUMA and Storage Order Tuning



Disk-to-disk transfer performance using XDD with no IRQ or NUMA tuning

Disk-to-disk transfer performance with network IRQ affinity tuning

Disk-to-disk transfer performance with tuned storage ordering and improved NUMA placements.

HP DL585 G7, 4x10Gbps NIC, 16x PCIe FusionIO (6.1 GB/s read 4.2 GB/s write)

XDDC: Composing XDD flows

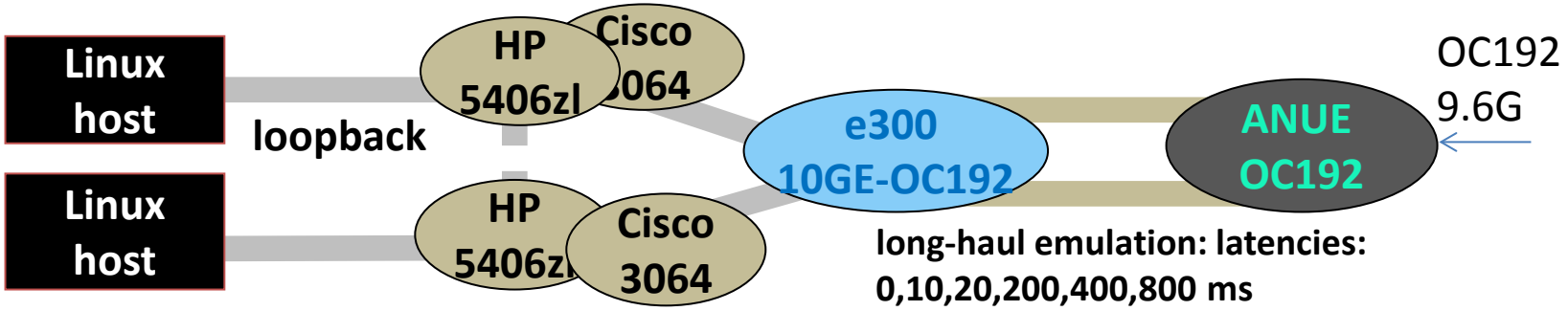
- Network and storage tuning optimizations aren't always complementary
  - Network may prefer high number of streams
  - Storage may prefer low thread counts
- Leverage profiling information to gather more complete understanding of performance modes
- Identify *compatible* network and storage performance modes and develop compositions based on that data
  - Given multiple endpoints and transient effects, this isn't exactly easy

Transport Considerations

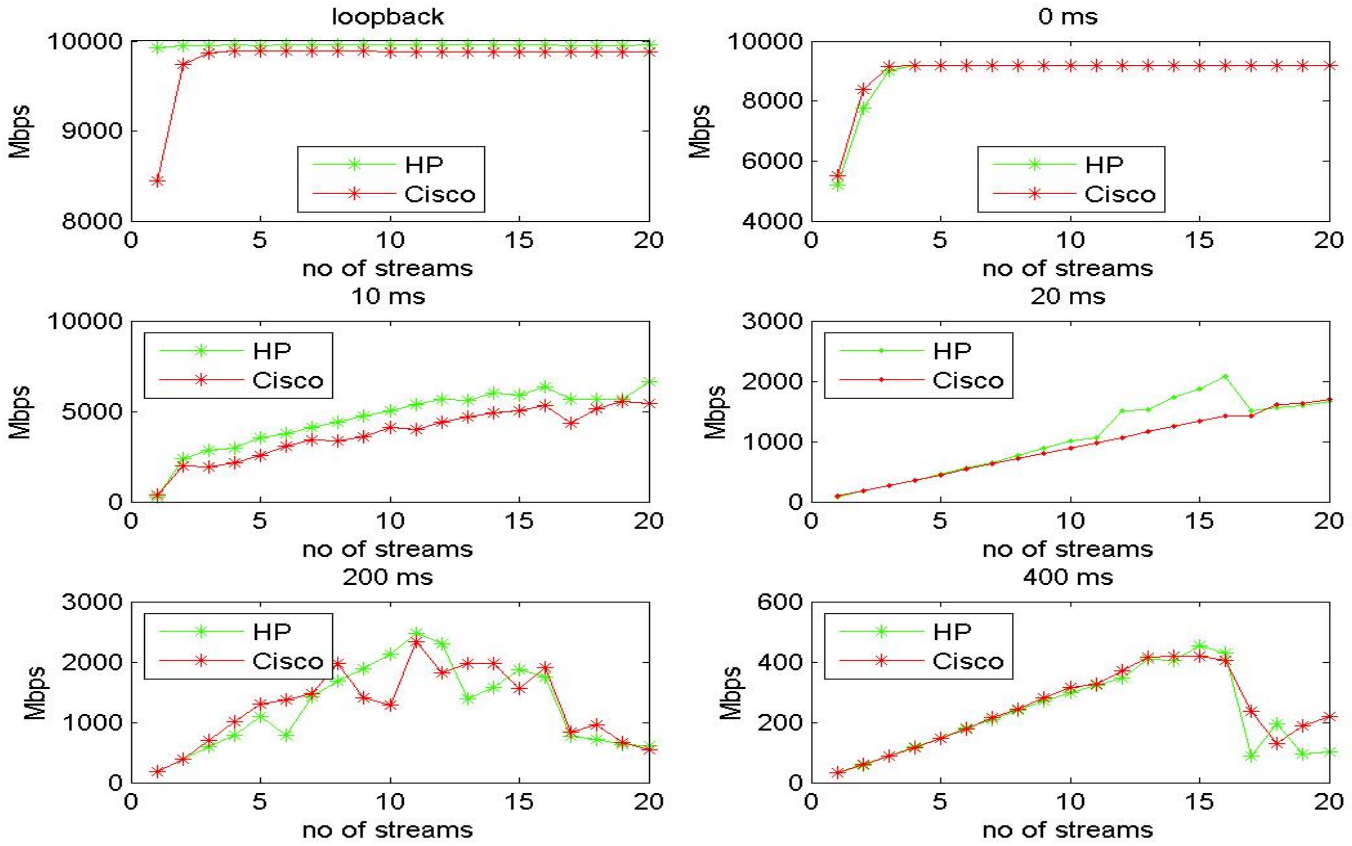
- Big data transfer in extreme-scale scientific applications requires end-to-end stable and high-bandwidth network connections
- Transport protocols play a significant role in utilizing the bandwidth resource of such dedicated channels in HPNs
- High-performance transport protocols are being rapidly developed and deployed, but maximizing their throughput over complex high-speed connections is challenging
    1) Their optimal operational zone is affected by the configurations and dynamics of the network, the end hosts, and the protocol itself
    2) The default parameter setting does not always yield the best performance
    3) Application users, who are domain experts, typically do not have the knowledge to choose which transport protocol to use and which parameter value to set
- We propose a Transport Profile Generator (TPG) to characterize and enhance the end-to-end throughput performance of transport protocols
    1) Automate the tuning of various transport-related parameters including socket options and protocol-specific configurations
    2) Support multiple data streams and multiple NIC-to-NIC connections

TCP Profiling

- Testbed: an emulated long-haul connection using Cisco 3064 & HP 5406zl switches
- Summary: TCP Throughput Comparison averaged over 10 runs

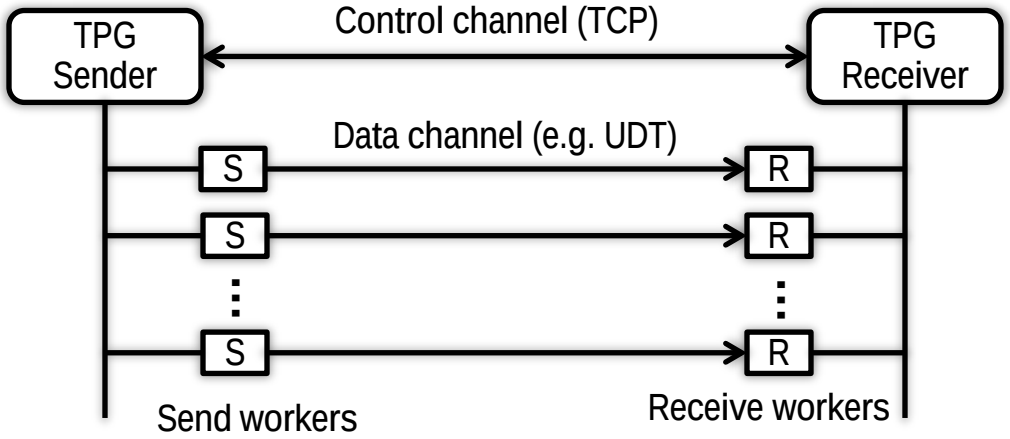


Performance profiles are "close" at all connection lengths.



Transport Profile Generator

- TPG Implementation:
  1) Consist of a pair of client and server, using a TCP-based control channel and a number of multi-threaded protocol-specific (TCP- or UDP-based) data channels
  2) Currently as a command-line tool, and will be integrated into XDDC with automatic parameter selection

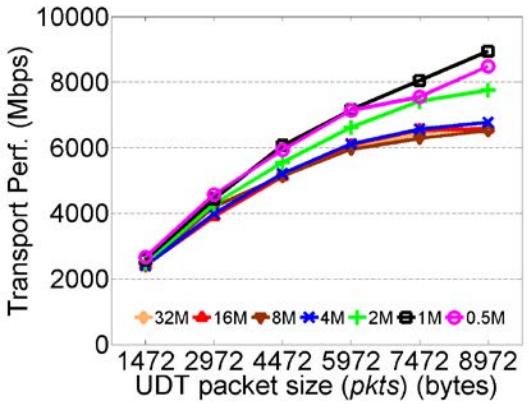


- A Typical Profiling Process
  1) Server starts listening on control channel
  2) Client connects to server, and exchange control parameter with server
  3) Client creates data streams on data channel, and connects them to server
  4) Assign each data stream to an independent sending/receiving thread
  5) Once a one-time profiling is completed, client and server exchange results
  6) Client exits, and server cleans up and listens for next profiling

UDT Profiling

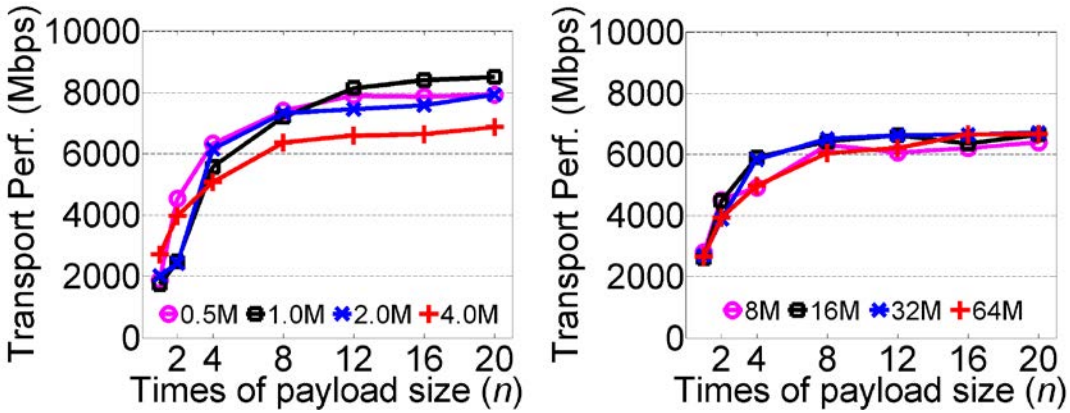
- Testbed: a local back-to-back 10Gbps link
- Summary: UDT throughput in response to different parameter settings

Packet Size



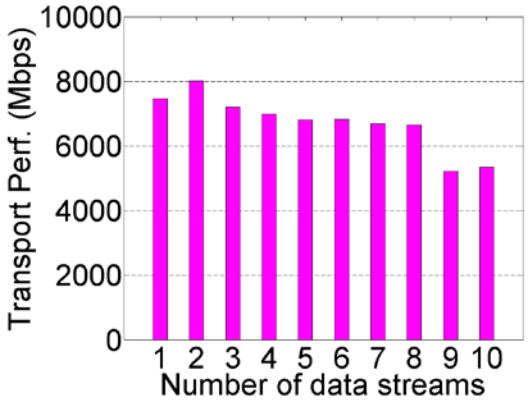
Average throughput vs. packet size.
1 stream, 120 seconds of transfer.

Block Size



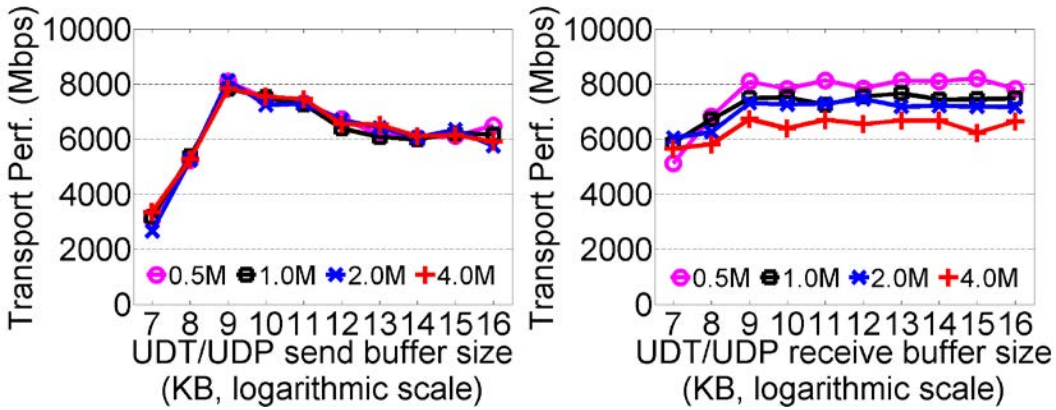
Average throughput vs. block size
8972B packet size, 8956B payload size, 1 stream, 120 seconds of transfer. Different curves correspond to different UDT/UDP send/receive buffer sizes.

Number of streams



Average throughput vs. number of streams.
1MB UDT/UDP send/recv buffer, 8972B packet size, 89559B block size, 1 stream, 120 seconds of transfer.

Buffer Size

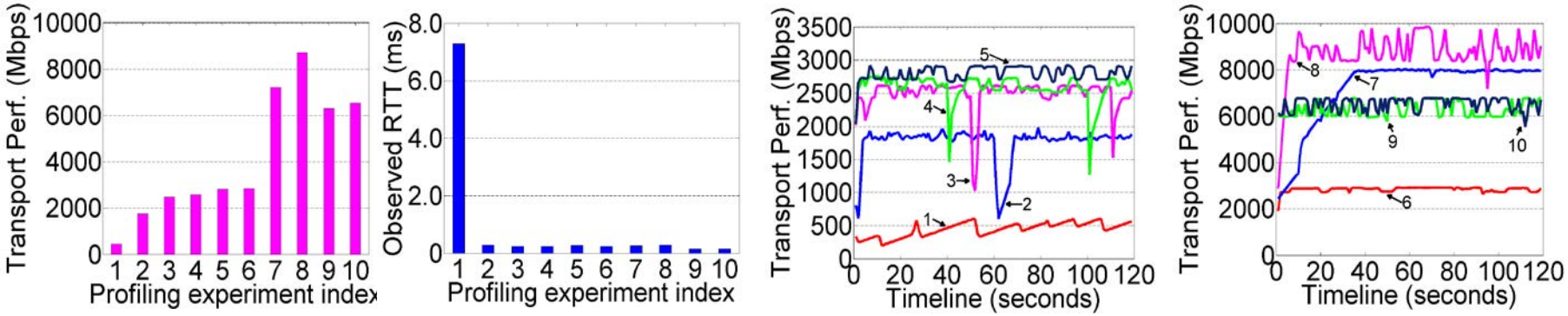


Average throughput vs. UDT/UDP send/recv buffer.
8972B packet size, 89559B block size, 1 stream, 120 seconds of transfer.

UDT Profiling

- Summary: throughput comparison between default UDT and TPG-tuned UDT

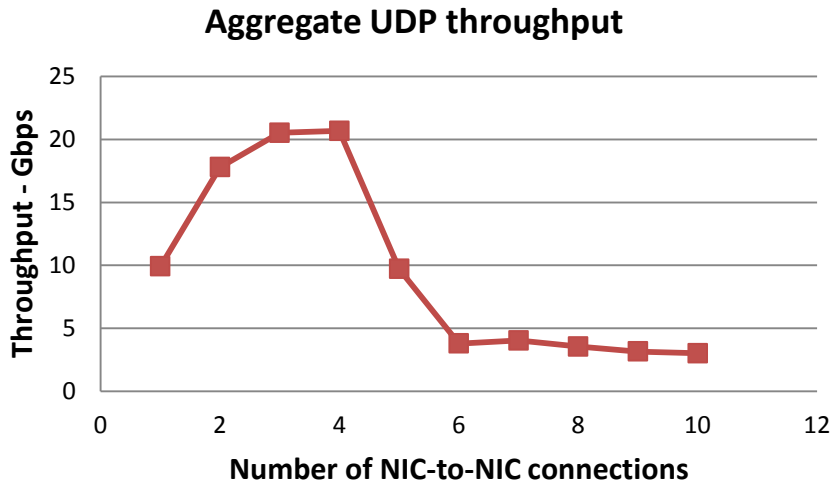
Index no.	Packet size (B)	Block size (B)	UDT sndbuf(M)	UDP sndbuf(M)	UDT rcvbuf(M)	UDP rcvbuf(M)	Perf. (Mbps)
1	1472	1455	10	1	10	1	450
2	1472	8955	10	1	10	1	1762
3	1472	65536	10	1	10	1	2487
4	1472	89559	10	1	10	1	2584
5	1472	89559	1	1	1	1	2810
6	1472	179119	1	1	1	1	2847
7	8972	89559	1	1	1	1	7216
8	8972	179119	1	1	1	1	8713
9	8972	179119	16	16	16	16	6300
10	8972	179119	32	32	32	32	6535

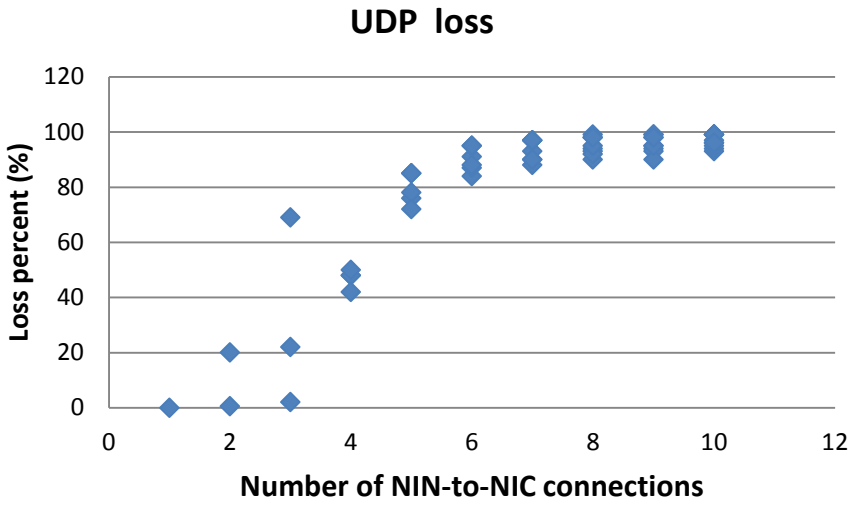


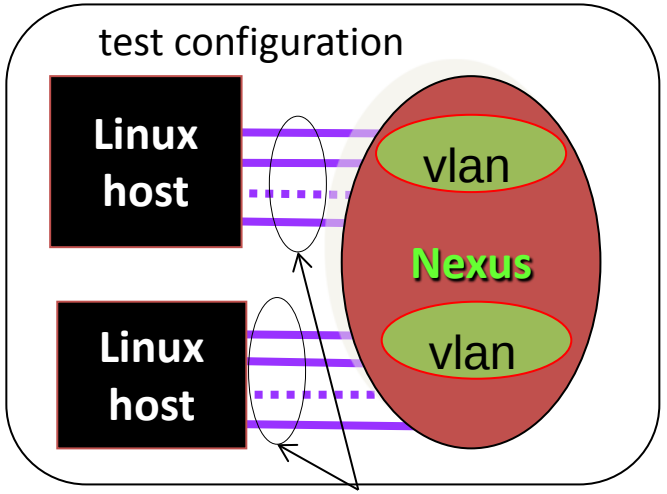
Throughput performance comparison between default UDT and TPG-tuned UDT: 120 seconds of transfer, 1 second sampling interval.
(a) Average thruput; (b) Average RTT; (c) Instantaneous thruput measurements in exp 1 to 5; (d) Instantaneous thruput measurements in exp 6 to 10.

UDP Profiling: Multiple NIC-to-NIC UDP Flows

- Testbed: two HP hosts with ten 10GigE NICs
- Summary: severe losses with more than 4 NIC-to-NIC UDP flows







10*10 GigE
48-core HP servers

Conclusions

- Host profiling software modules – first version
- XDD: disk-to-disk transport tool – using default network connections
- Transport profiling and methods - UDP profile under investigation
- Testing with physical and emulated connections

Next Steps

- Continued refinement of host profiling and configuration modules
- Network and disk flow matching and alignment of XDD
- Refinement of multiple NIC-to-NIC flow aggregations: TCP and UDP
- Testing over ORNL-ANL and ORNL 100G connections
- Aggregation of combinations of multiple 10G and 40G NIC flow aggregations: 3*40G; 2*40G+2*10G

"What question does your research motivate you to now ask?"

Do combined advances in OpenStack, Open Systems, and Software Defined Infrastructures provide ready-to-use solutions for complex high-performance data flows? Or, do we need to develop/evolve them for this class of scenarios?