7TH ANNUAL WORKSHOP ON EMERGING PARALLEL AND DISTRIBUTED RUNTIME SYSTEMS AND MIDDLEWARE (IPDRM)

SmartNIC Data Exchange Framework

Zackary Savoie, Anthony Sicoie, Ryan E. Grant
Queen's University

Introduction

Introduction – The Problem

We have overprovisioned hardware in the form of SmartNICs but don't know how to best use them

SmartNICs are not being widely used as application-specific accelerators but could be used as general accelerators

To best utilize these resources, we require inter-SmartNIC communication

Introduction – Our Solution

We built a flexible framework that supports varying network topologies for inter-SmartNIC communication

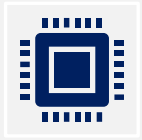
Low overhead and high frequency

- Not just for data gathering
- Enables decision making

Our tool enables users to develop their own system software or middleware without having to worry about the inter-SmartNIC communication layer

Background

SmartNICs – What are they and how are they used?



SmartNICs combine the network card functionality with both application-specific accelerators and general-purpose compute cores

Capable of so much more/Jack of all trades:

- System tuning
- Collective offloading
- Autonomous resource management

Modes of operation

- NIC Mode
- DPU Mode

Message Passing Interface (MPI)

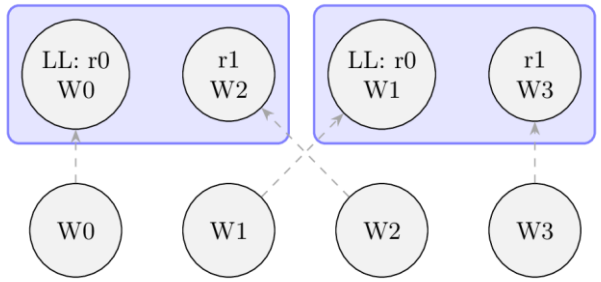
Communication Groups

Neighbourhoods

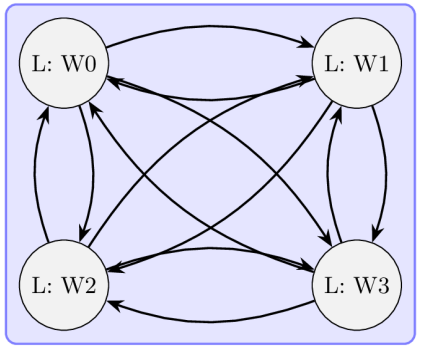
MPI Functions:

- MPI_Comm_split( )
- MPI_Allgather( )/MPI_Allgatherv( )
- MPI_Gather( )/MPI_Gatherv( )

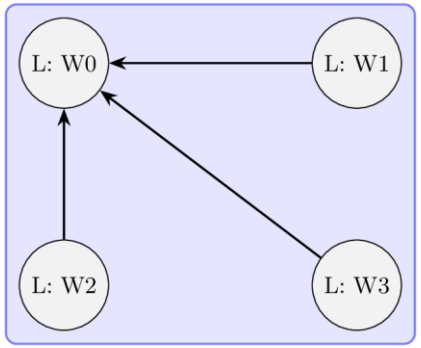
MPI_Comm_split( )

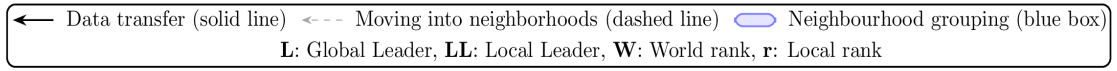


MPI_Allgather[v]( )



MPI_Gather[v]( )





Architecture

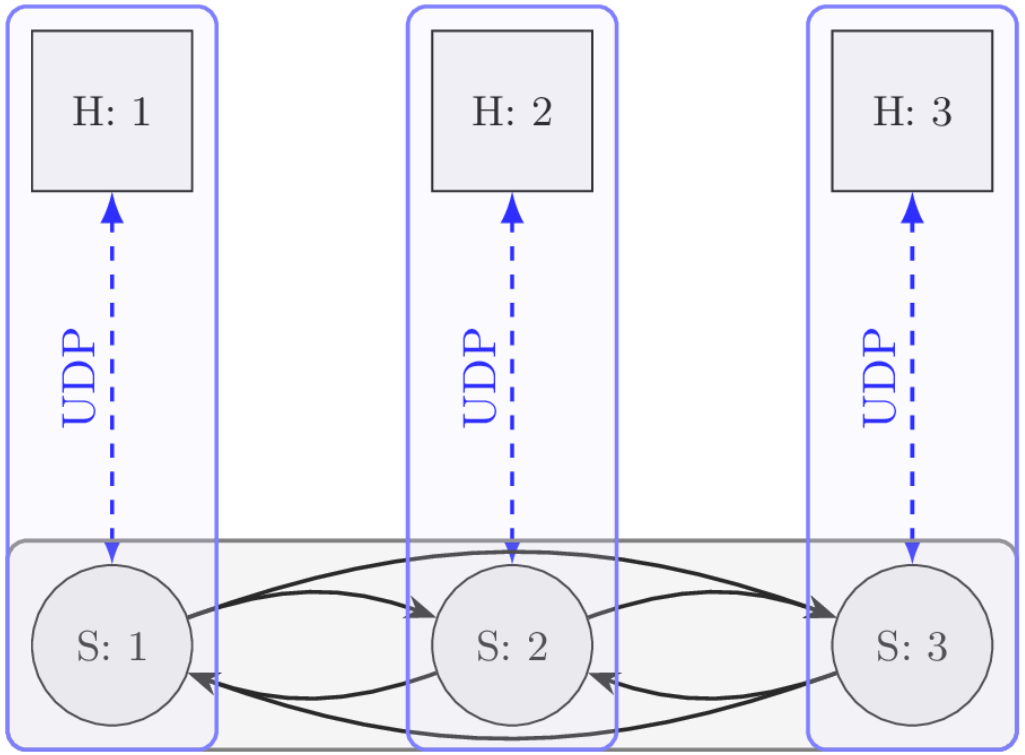
Architecture – Big Picture

Gather relevant host data

- SmartNIC sends request datagram
- Host returns appropriate data

Exchange data between SmartNICs

- Utilize inter-SmartNIC communication framework
- Select appropriate exchange topology based on the application



Architecture – Data Collection Framework Requirements

Requirements

Low Host-Side Overhead

Low Latency / High-Frequency Retrievals

Support for Diverse Information Sources

Counters in virtual files, logs in regular files, command outputs, RDMA-accessible application memory, etc.

Architecture – Data Collection Framework Implementation

Our solution: RPC-like server/client setup over UDP

- Lower latency than TCP, no argument passing, re-transmission if request datagrams lost (after timeout)

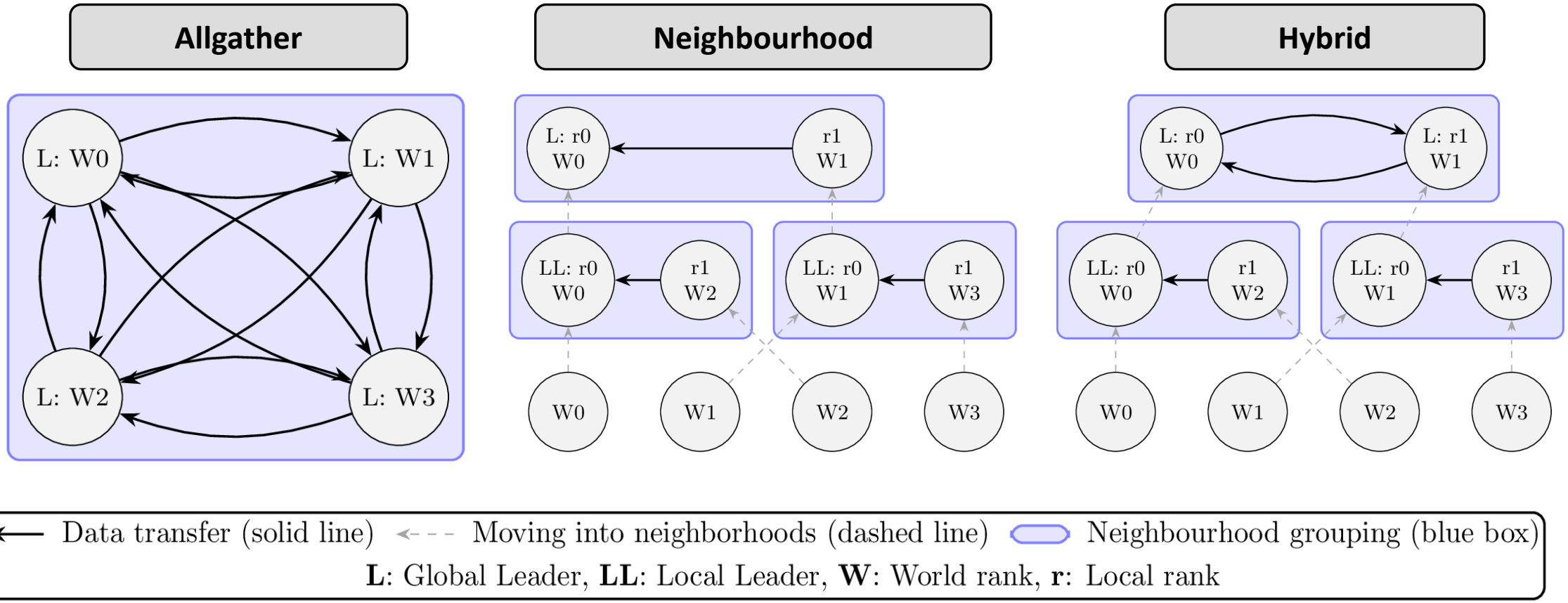
Step-by-step:

- A datagram-socket-based (UDP) server process is launched on each traditional host
- This process waits for a request message to be sent by this host's SmartNIC
- Each time a request datagram is received, process execution resumes and data is gathered, formatted, and returned to the requesting SmartNIC
- When a shutdown message is eventually received, the process closes its socket and terminates

The following information was transmitted during framework testing:

Source	Values / Fields
/sys/class/infiniband/*/ports/1/counters/port_rcv_data	Received data (all IB ports)
/sys/class/infiniband/*/ports/1/counters/port_xmit_data	Transmitted data (all IB ports)
/proc/stat	CPU cycles spent in each state, total running and blocked processes, and context switches

Architecture – Communication Types



Experimental Setup

Experimental Setup – Systems

System	CPU	Memory	Interconnects	OMPI Version	OS
Thor (16 Nodes)	Dual 16-core Xeon E5-2697A	256GB	Infiniband (200 Gb/s HDR)	4.1.7rc1	Rocky 9.4
BlueField-3 (Installed in Thor Nodes)	16 Armv8.2+ A78 Hercules cores	32GB	Infiniband (200 Gb/s HDR)	4.1.7rc1	Rocky 9.4
Rorqual (40 Nodes Used)	Dual 96-core AMD EPYC 9654	750GB	Infiniband (200 Gb/s HDR)	4.1.5	AlmaLinux 9.6

Experimental Setup – Goals of Testing

1. Measure latency of inter-SmartNIC exchanges using various topologies
2. Understand impact of message size on inter-SmartNIC latency
3. Study scalability of framework beyond 16x SmartNICs using x86 cores as proxies
4. Measure host-side resource utilization of framework

Experimental Setup – Notes for Interpretation

- “Exchange time” is measured as the average time from posting local data (MPI call) to when a complete system view is received by the global leader(s)
- Host data retrieval time is measured from request datagram transmission to data receipt on the SmartNICs
- “Small” messages are 48 B and “Large” messages are 64 KB
- “Neighbourhoods” runs have fixed neighbourhood sizes of 2 processes, “AllGather” runs have a fixed depth of 1, and “Hybrid” runs have neighbourhood sizes of 2 and a maximum depth of 3

Results

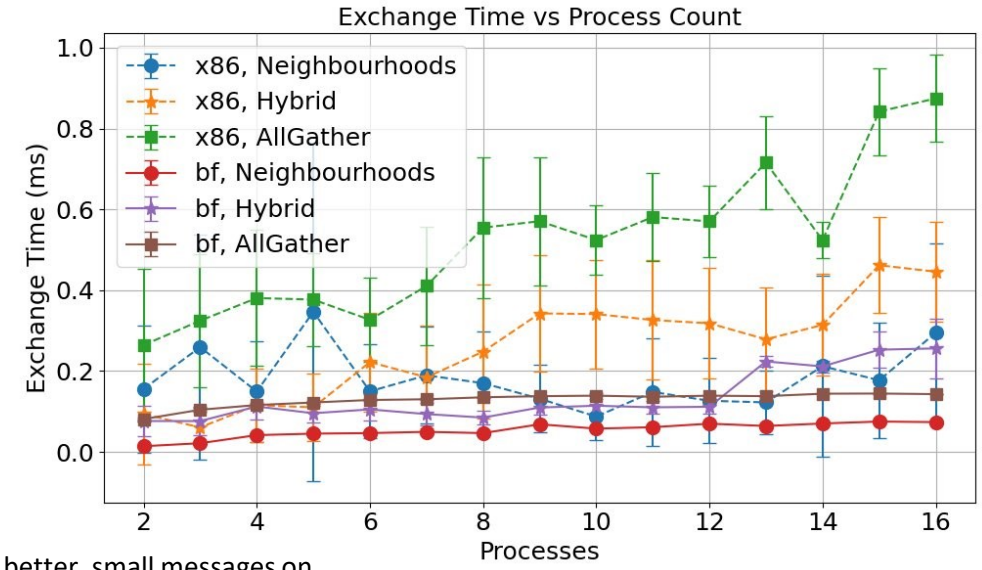
Results – Impact of Topology on Latency

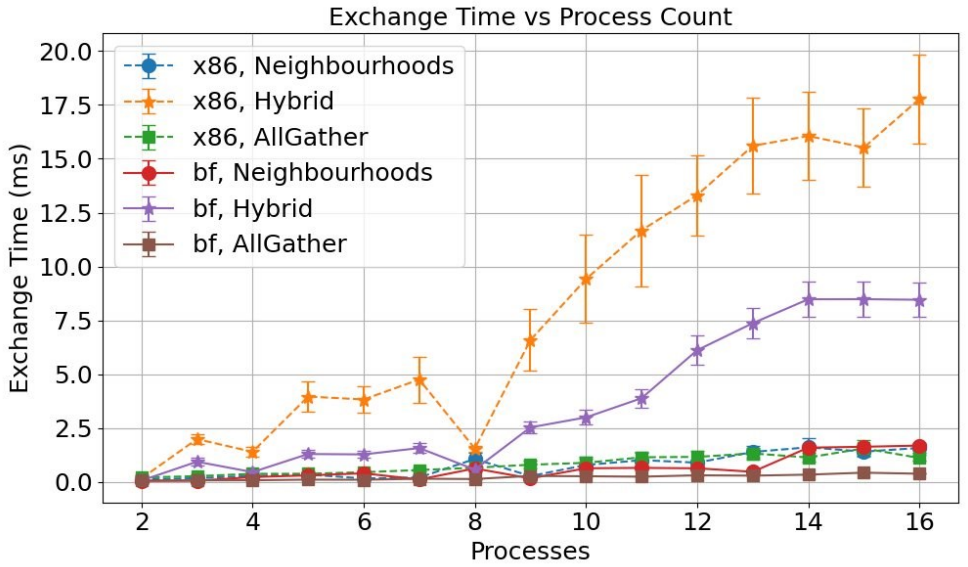
BlueFields exchanged data faster than x86 cores regardless of message size and topology

Narrowing of x86 vs BlueField performance delta attributable to weaker cores

Hybrid topology struggles with large messages (seen in later experiments too)

- Additionally, hybrid more sensitive to message size than process count



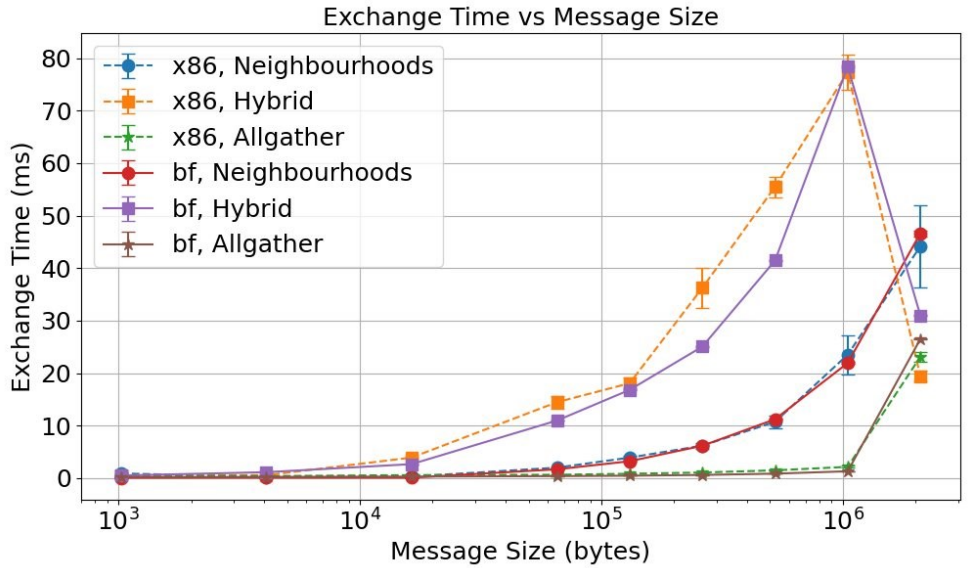


Note: Lower is better, small messages on left, large messages on right

Results – Impact of Message Size on Latency

Within expected message size ranges (<1 MB), relative performance of topologies constant

Anomalous dip in hybrid exchange times beyond 1 MB under investigation, otherwise hybrid remains slowest topology (partially attributable to use of variable allgather setup costs)



Note: Lower is better

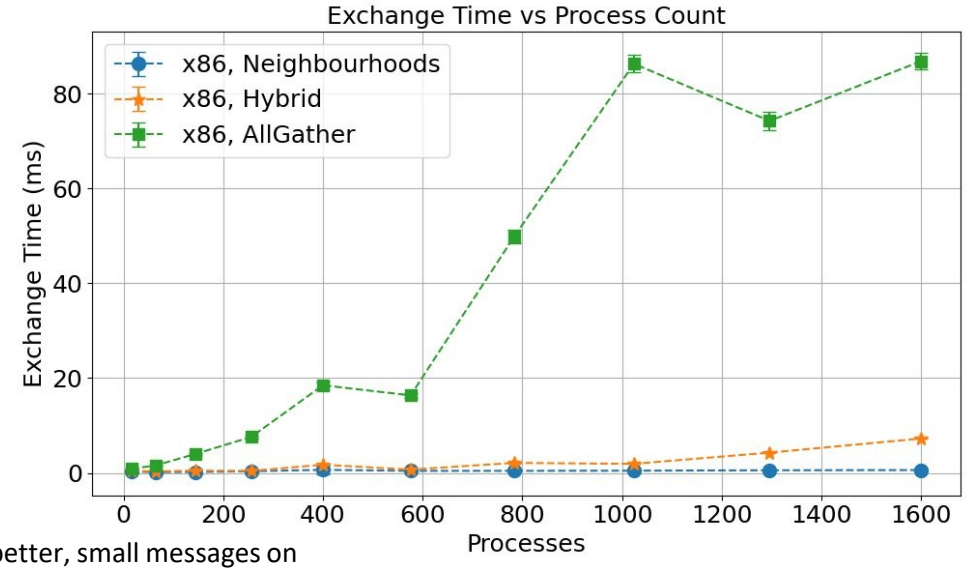
Results – Scalability Study

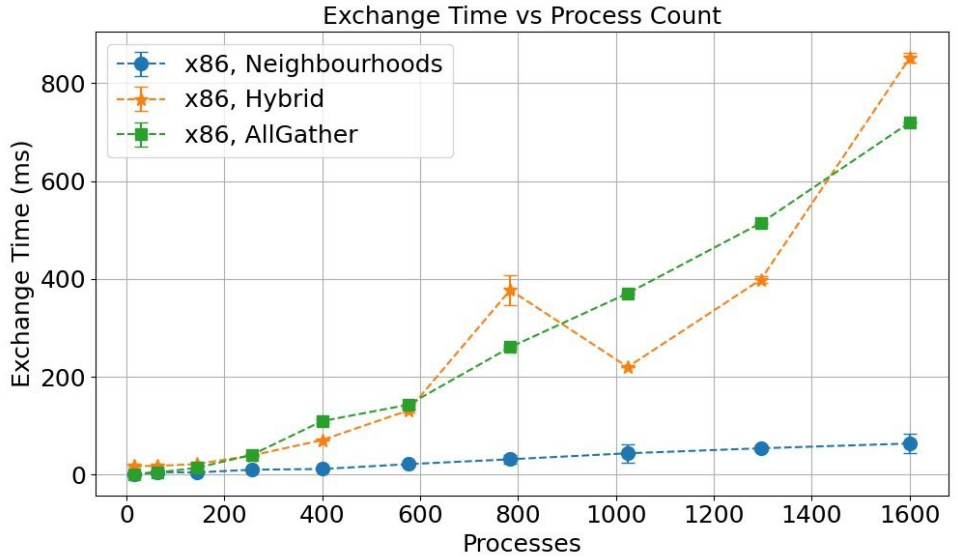
Equal number of processes-per-node and nodes used (ex. 1600 processes = 40 nodes, 40 P.P.N)

Advantage of using neighbourhoods evident for both message sizes

Hybrid topology's increased sensitivity to message size over process count seen again

AllGather topology struggles with large process counts regardless of message size





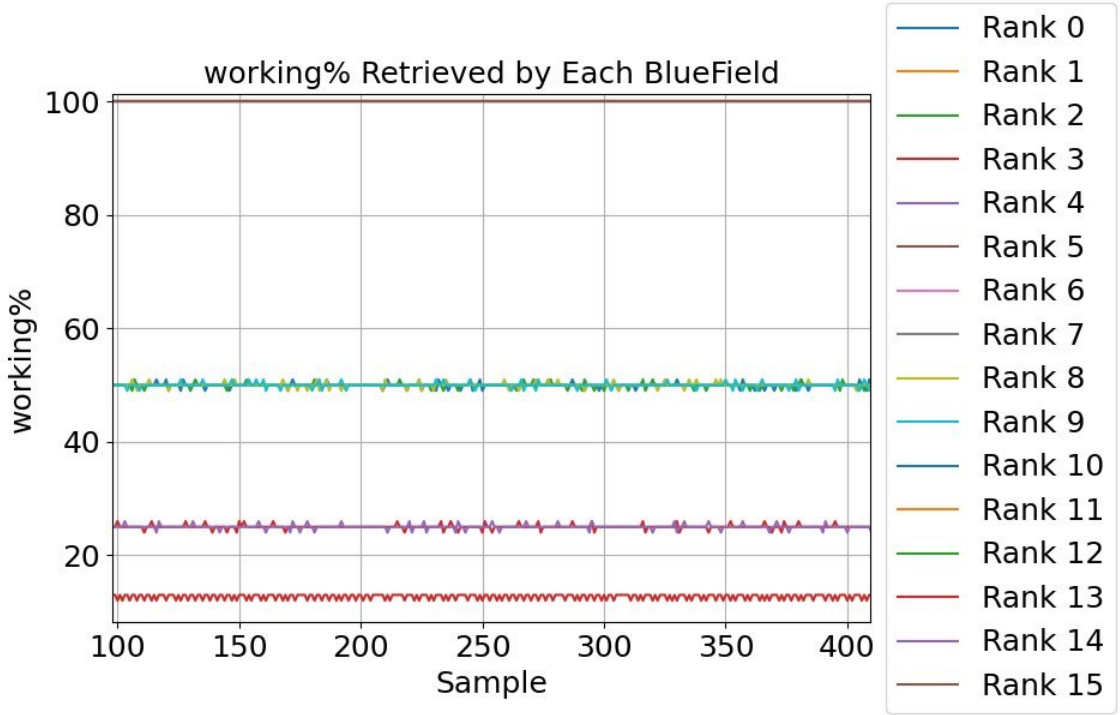
Note: Lower is better, small messages on left, large messages on right

Results – Sample Monitoring Application

Average of 790 μs to request and receive updated data (not exchange)

- Of this, 200 μs spent on host-side reading and sending data
- Remainder spent on transmission, receipt and unpacking
- Maximum sampling frequency of ~1250 Hz

Only 2640 KB of memory used to run server process

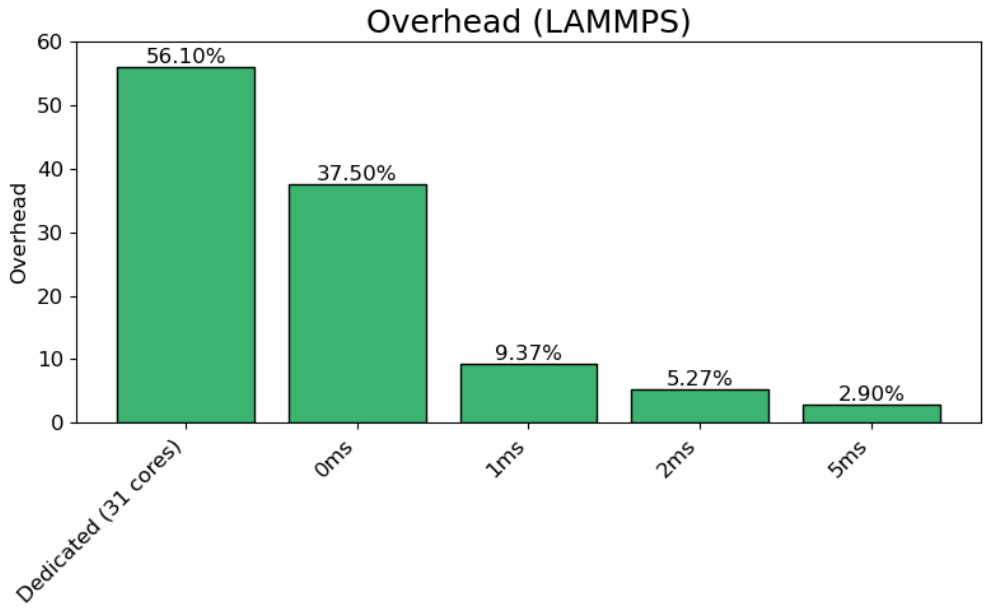


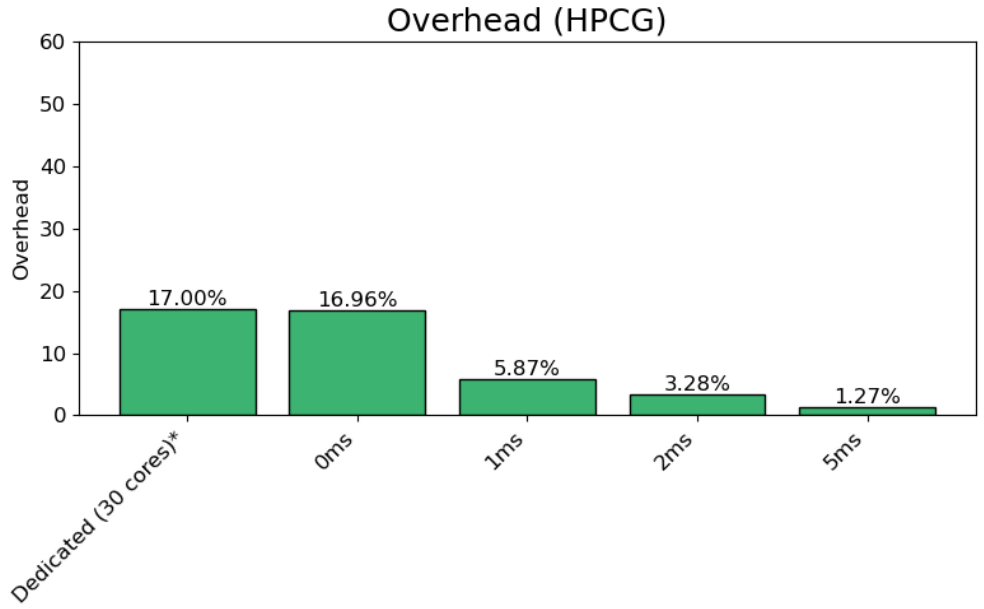
Results – Impact on Application Performance

Thor cluster used (16x 32-core nodes) with either one dedicated core per node* or oversubscription

Exchange framework polled with variable frequency to simulate different “decision times” on BlueFields

Overhead quickly decreases with frequency reduction, dedicating 1 of 32 cores predictably not economical





Note: Lower is better

Future Work

Future Work – SmartNIC Data Exchange Framework

Currently, only maximum neighbourhood size and depth control topology; this limits how well the topology can be matched to the hardware layout

- Ability to group processes into neighbourhoods arbitrarily will permit node-, row-, and rack-level neighbourhoods

The UDP-based solution unlikely to fully leverage high-speed interconnects, still requires host involvement

- RDMA-based host data retrieval will be implemented in the future to address these concerns

Future Work – Applications

Work Stealing

- SOTA uses dedicated communication threads already; offload them
- Good fit for AllGather

Dynamic Power Budgeting

- Manage at node-, rack-, or row-level
- Good fit for neighbourhood / hybrid

Shared Resource Access Control

- Prevent bottlenecks related to IO / disk or network usage
- SmartNICs have demonstrated ability to predict network traffic

Conclusion

Conclusion – How can this Framework be Used?

To integrate this into your project:

- Modify the host-side server to specify the data to gather, and modify the compute_host_load() function to properly fetch and format this host data for inter-SmartNIC exchange (this is the client code)
- Implement the BlueField-side update_global_view() function to act appropriately given received (exchanged) data (this is where decisions can be made with received data)

Init Functions

- createCommsArray()
- createNeighbourhoodView()

Exchange Functions

- compute_host_load()
- gatherCommData()
- update_global_view()

Further documentation is provided in our public repository

Conclusion – Summary of Work

github.com/labcaesar/bf_exchange_framework

Architecture	AllGather, Neighbourhood, and Hybrid topologies are supported to mesh with more applications	MPI is leveraged for inter-SmartNIC communication, UDP server/client system used for low-latency host-SmartNIC communication
Results	Neighbourhoods offer dramatically improved exchange times when scaling to large process counts, especially with large messages	Up to 1250 Hz host sampling achieved, with microsecond-range host CPU time per transfer and < 3 MB host memory use
Applications	Can be used for monitoring / logging out-of-the-box	Future applications include work stealing, power management, and shared resource access control, among others

doi.org/10.1145/3731599.3767460

github.com/labcaesar/
bf_exchange_framework

Thank You!

Questions?

doi.org/10.1145/3731599.3767460

Thank you to the Digital Research Alliance of Canada and the HPC Advisory council for allowing access to their systems. We acknowledge the support of Mitacs and the Natural Sciences and Engineering Research Council of Canada (NSERC), [ALLRP 5785392022].