Accelerating Intra-Node GPU Communication: A Performance Model for Multi-Path Transfers

AmirHossein Sojoodi, Mohammad Akbari, Hamed Sharifian, Ali Farazdaghi, Ryan E. Grant, and Ahmad Afsahi

Department of Electrical and Computer Engineering

Smith Engineering
Queen's University, Kingston, Canada

ExaMPI 2025,
St Louis, MO, USA, November 16th, 2025

1. HPC and Cluster Computing
2. Age of Memory Hungry Applications
3. GPUs and High-Bandwidth Interconnects
4. Multi-Path Communication
5. Empowered with Performance Modeling

Multi-Path Communication

- Use case: GPU-0 sends data to GPU-1
- Utilize available interconnects to transfer data more efficiently
- Requires extra care and optimization to find the splitting configuration

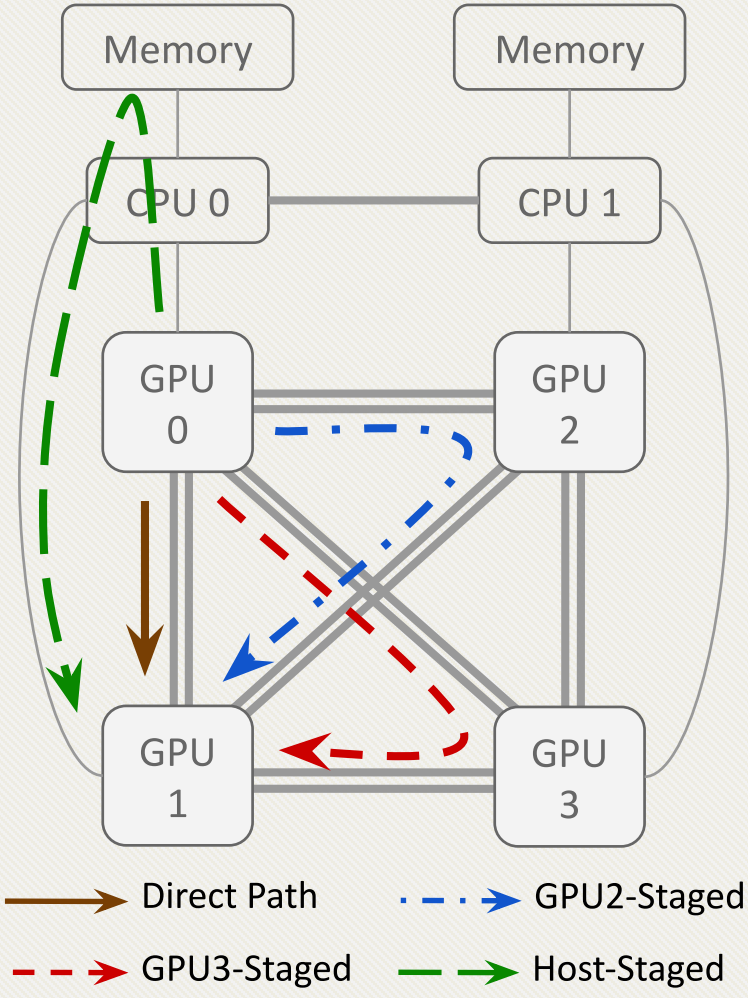


Figure 1: A multi-GPU data transfer

Performance Modeling to the Resuce

- Finding multi-path configuration depends on:
  - The message size
  - Number of paths
  - The characteristics of each path
  - Optimal chunk size (If pipelining is enabled in that path)
- Naïve solution: Exhaustive search on all the parameters.
  - Time consuming
  - Not portable
- Better solution: Modeling and Prediction!

1. MPI and UCX
2. GPUs and GPU-Enabled Clusters
3. UCX CUDA IPC Module

MPI and UCX

- When using MPI+UCX for intra-node GPU communication, the module "cuda_ipc" in UCX translates MPI transfers to one-sided PUT or GET.

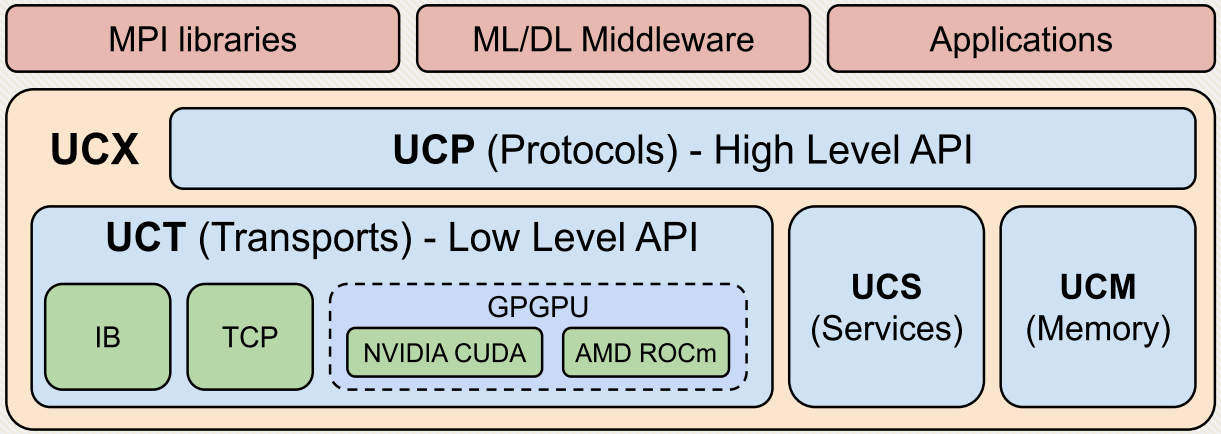


Figure 2: MPI and UCX high-level software stack

CUDA IPC mechanism in a nutshell:

1. Receiver gets IPC memory handle of its buffer
2. Receiver shares the memory handle with the sender
3. Sender opens the IPC memory handle (attach)
4. Sender initiates a **PUT** operation (or **GET** vice versa)

Performance Modeling

- Provides analytical models for the evaluation of communication costs based on various platform-specific parameters
- Famous examples: Hockney, LogP and its variants, Max-Rate, etc.
- Hockney's model, where:
  - **T** is the total communication time, **n** is the message size
  - α is startup latency, and β is the link bandwidth

$$T = \alpha + \frac{n}{\beta}$$

1. Extending Hockney's Model for Multi-Path
2. Incorporating Staging and Pipelining
3. Multi-Path Communication Framework

Design Objectives

1. Multi-GPU Awareness
   - Utilize the available GPUs and their interconnects
2. Path Selection
   - Use the selected GPUs (and the host) as staging devices
3. Overlap data transfers across all the paths
   - Maximize the overlaps to minimize the overall time
4. Apply Pipelining
   - To minimize the routing overhead at staging devices

Extending Hockney

- A system with *P* available paths a communication between two GPUs can be classified as (w/ or wo/ pipelining):
  - Direct path (no staging)
  - GPU-staged
  - Host-staged
- Communication time for path *i* can be written as two distinct communication with a synchronization cost at the staging device.

Notation	Description
T	Total communication time
n	Message size
α	Startup Latency
β	Bandwidth
p	Number of paths
T_i	Communication time of path i
α_i, β_i	Model parameters of path i
θ_i	Fraction of message on path i
ϵ_i	Synchronization overhead at staging device i

Table 1: Notations used in the model

$$T_i = \alpha_i + \frac{\theta_i n}{\beta_i} + \epsilon_i + \alpha_i' + \frac{\theta_i n}{\beta_i'}$$

Extending Hockney – Overall Time

- If we split the data across multiple paths, the total communication time would be the maximum of all of them:

$$T = \max_{i \in \{1,\ldots,p\}} T_i$$

- Therefore, we can find the optimal path configuration by answering this question: **"What allocation of data across the available paths ($0 < \theta_i < 1$) minimizes the total transfer time?"**

Notation	Description
T	Total communication time
n	Message size
α	Startup Latency
β	Bandwidth
p	Number of paths
T_i	Communication time of path i
α_i, β_i	Model parameters of path i
θ_i	Fraction of message on path i

Table 1: Notations used in the model

$$T_i = \alpha_i + \frac{\theta_i n}{\beta_i}$$

*To simply the notations and without loss of generality, we can assume we have no staging

Finding the Optimal Configuration

- The optimal configuration is proven to occur when the transmission times of all paths are equal.
- Omitting the proof and details, the data fractions assigned to each path can be analytically derived from the message size, the number of paths, and the characteristics of the extracted paths.

$$\theta_i = \frac{\beta_i}{\sum_{j=1}^{p} \beta_j} \left(1 - \frac{\alpha_i}{n} \sum_{j=1}^{p} \beta_j + \frac{1}{n} \sum_{j=1}^{p} \alpha_j \beta_j \right)$$

- Similar approaches can be done for cases in which staging and pipelining is enabled for any of the paths. (see the paper for more information)

Notation	Description
T	Total communication time
n	Message size
α	Startup Latency
β	Bandwidth
p	Number of paths
T_i	Communication time of path i
α_i, β_i	Model parameters of path i
θ_i	Fraction of message on path i
ϵ_i	Synchronization overhead at staging device i
α_i', β_i'	Parameters for the second link in staged transfer

Table 1: Notations used in the model

Framework Desing

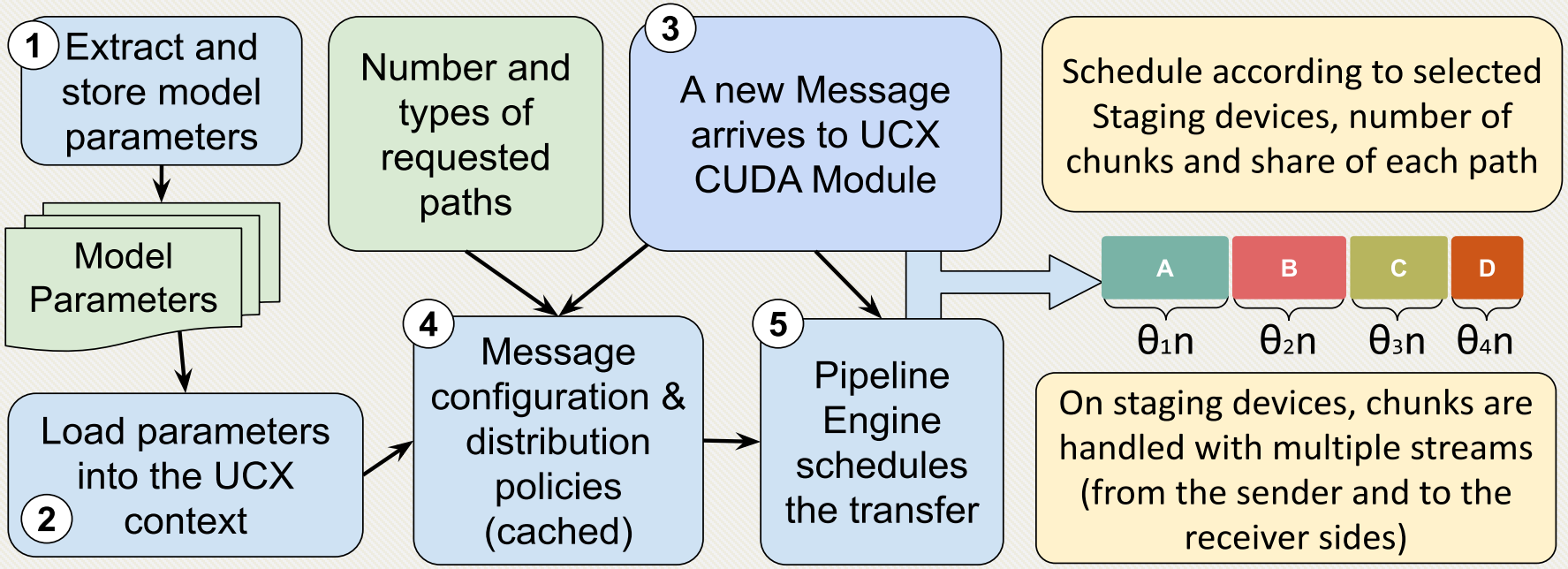
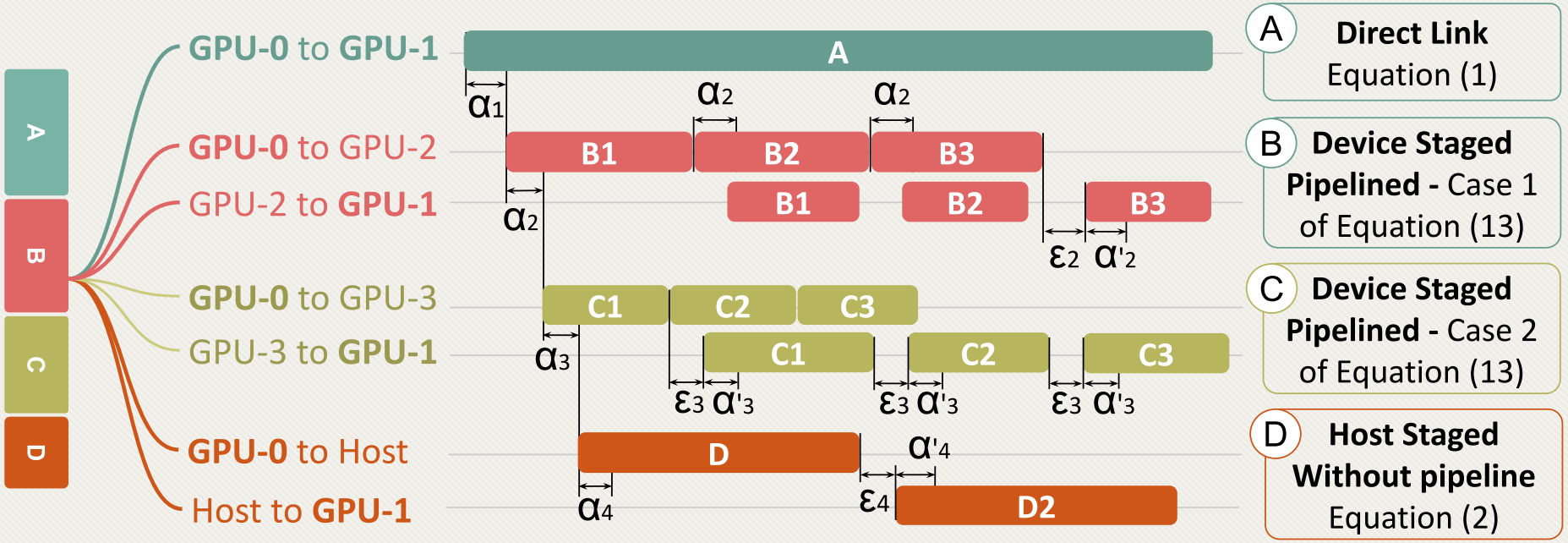


Figure 3(a): High-level design and implementation of the communication model

Different Cases in Action



*Overlapped startup and synchronization latencies are not shown

Figure 3(b): How a single data transfer from GPU-0 to GPU-1 is split into four parts and assigned to different paths.

A Microbenchmarks – OMB Unidirectional Bandwidth

B Microbenchmarks – OMB Bidirectional Bandwidth

C Microbenchmarks – OMB Alltoall

D Microbenchmarks – OMB Allreduce

Experimental Setup

- Beluga of Digital Research Alliance of Canada
  - 4 x NVIDIA V100 per node
  - Each GPU pair have two NVLinks
- Narval of Digital Research Alliance of Canada
  - 4 x NVIDIA A100 per node
  - Each GPU pair have four NVLinks
- MPI v5.0.4, UCX v1.14, and CUDA v12.6

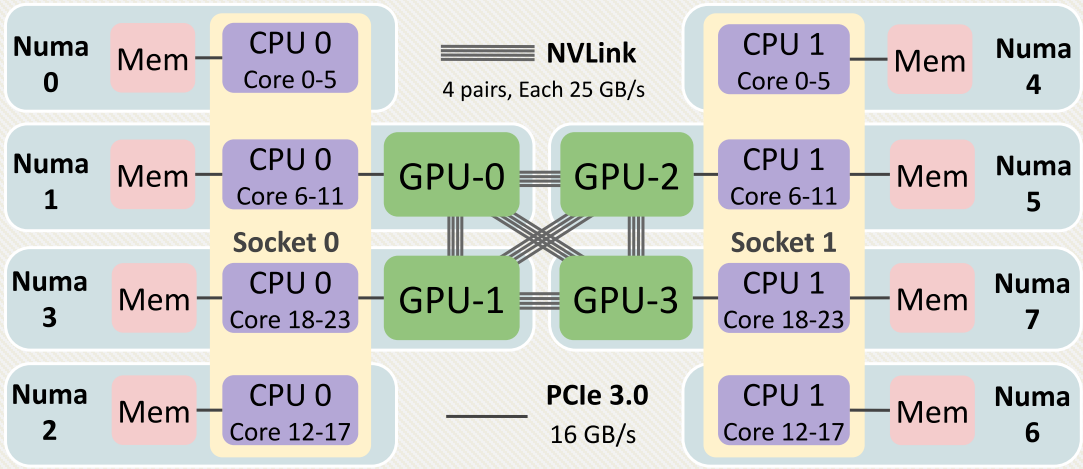


Figure 4: Narval Node Architecture

Performance Evaluation

- To evaluate the performance of our framework, we compare three different configurations:
  1. **Static Path Distribution:** Fixed pre-determined configuration extracted by exhaustive search (similar to the work in [1])
  2. **Model-Driven Prediction:** This is our proposed performance model, where we report the prediction of the model of the bandwidth (not the actual measurement)
  3. **Dynamic Path Distribution:** This is the implementation of our model in the runtime library to dynamically compute the model's parameters and to find the best distribution policy

[1] Amirhossein Sojoodi, Yıltan Hassan Temucin, and Ahmad Afsahi. 2024. Enhancing Intra-Node GPU-to-GPU Performance in MPI + UCX through Multi-Path Communication. In Proceedings of the International Workshop on Extreme Heterogeneity Solutions (ExHET)

OMB Unidirectional Bandwidth - Beluga

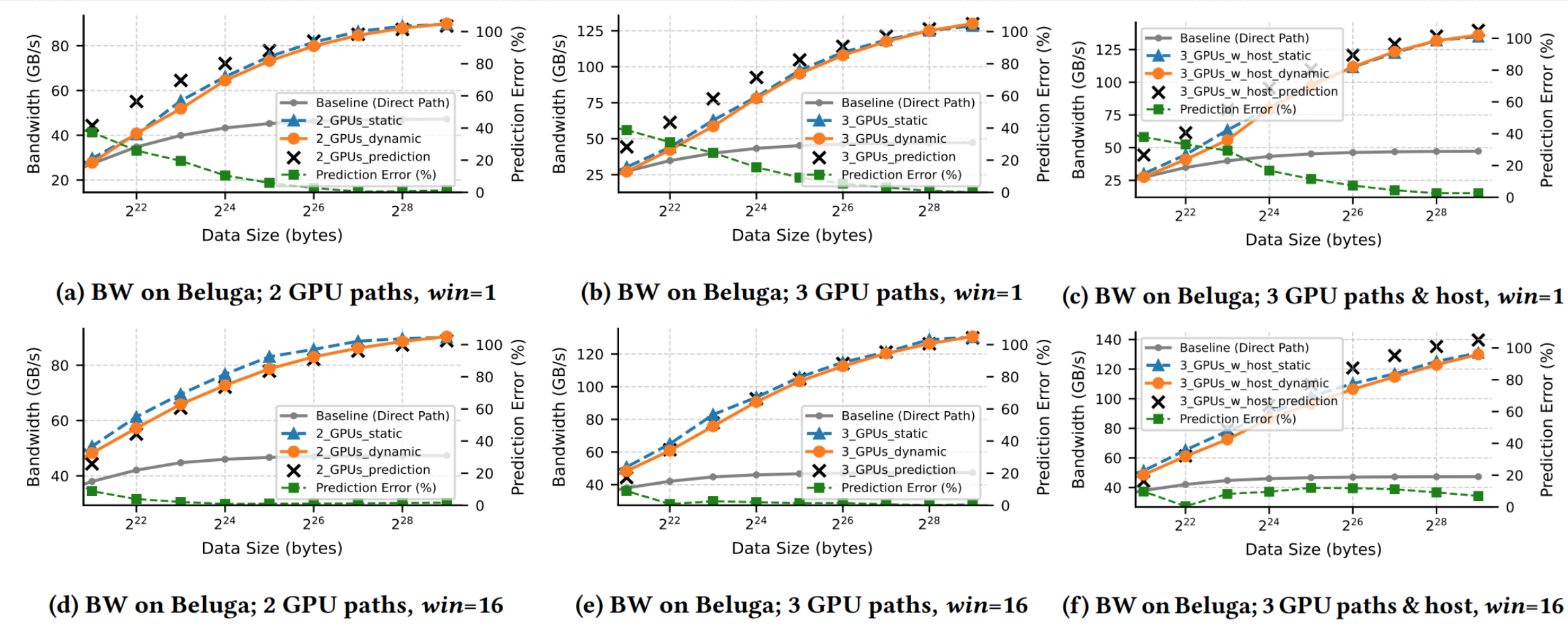


Figure 5: OMB unidirectional bandwidth tests on Beluga. Comparing the predicted and two measured configurations.

OMB Unidirectional Bandwidth - Narval

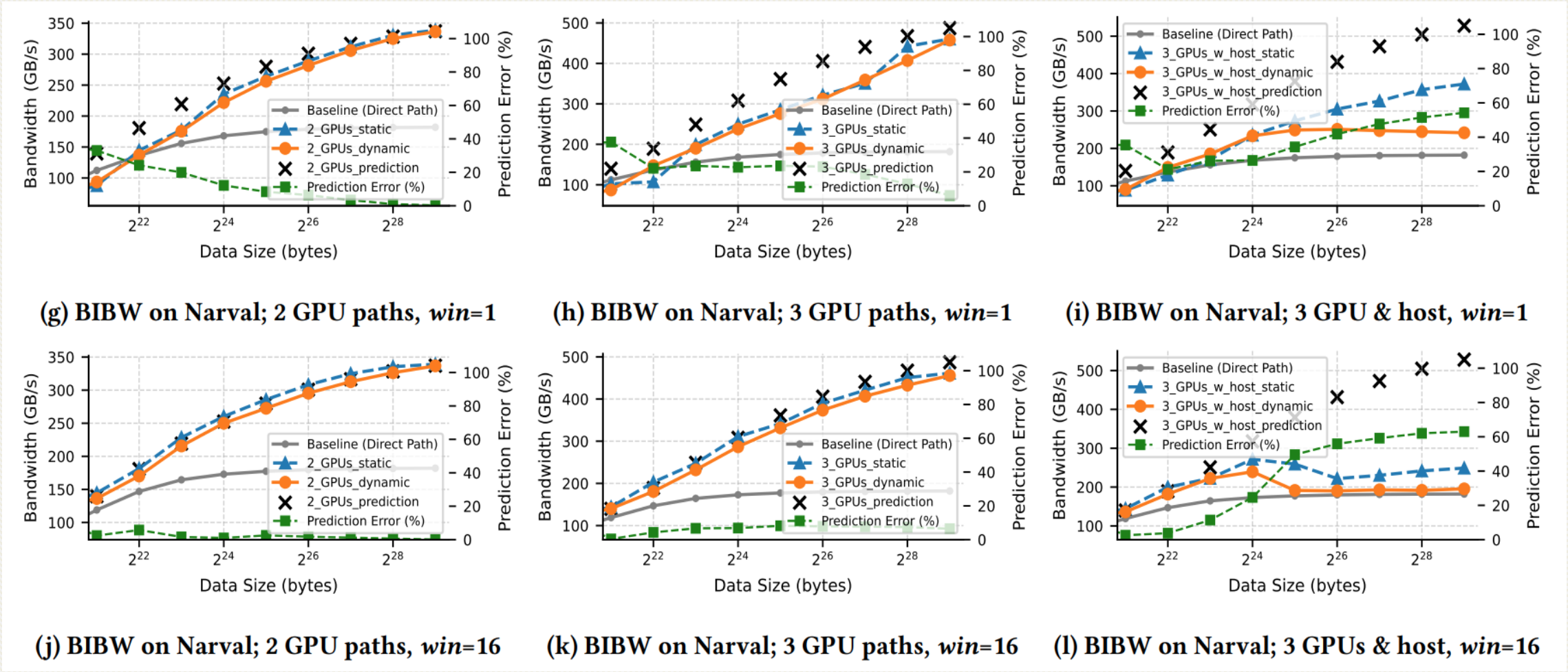


Figure 5: OMB unidirectional bandwidth tests on Narval. Comparing the predicted and two measured configurations.

OMB Bidirectional Bandwidth - Beluga

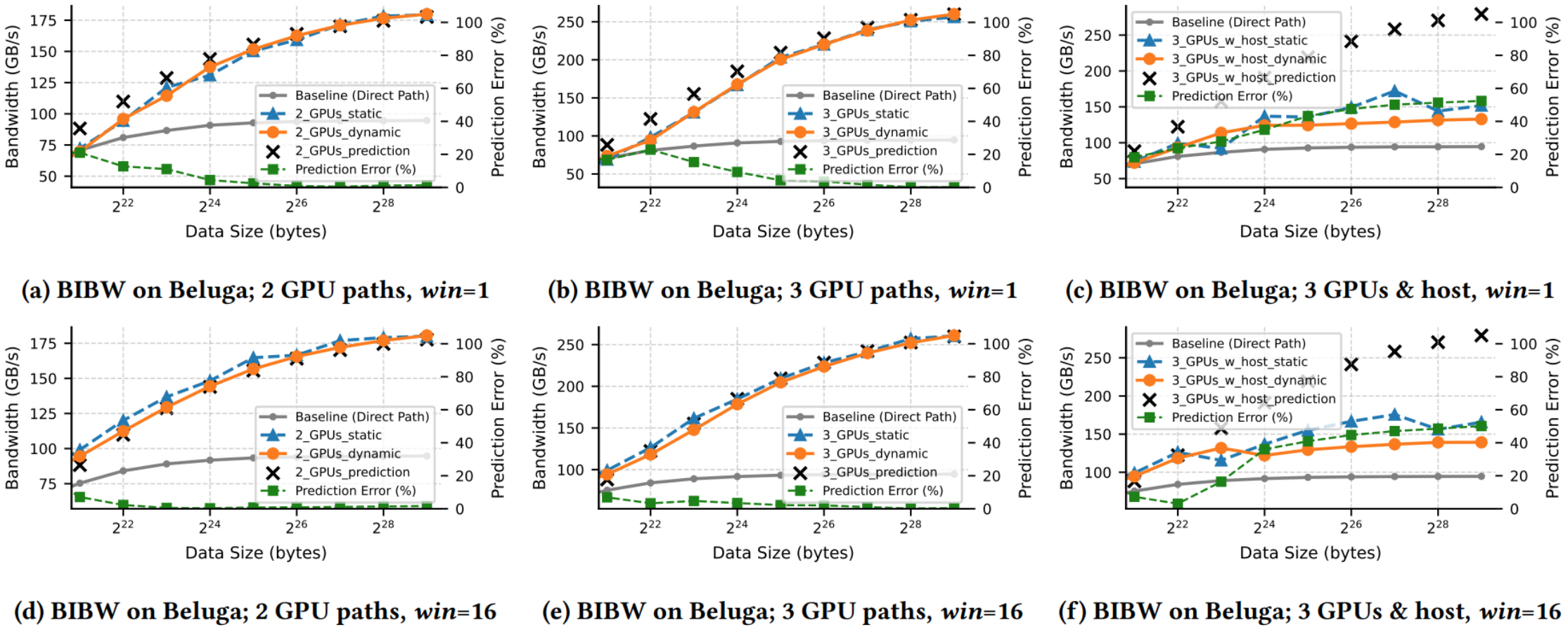


Figure 6: OMB bidirectional bandwidth tests on Narval. Comparing the predicted and two measured configurations.

OMB Bidirectional Bandwidth - Narval

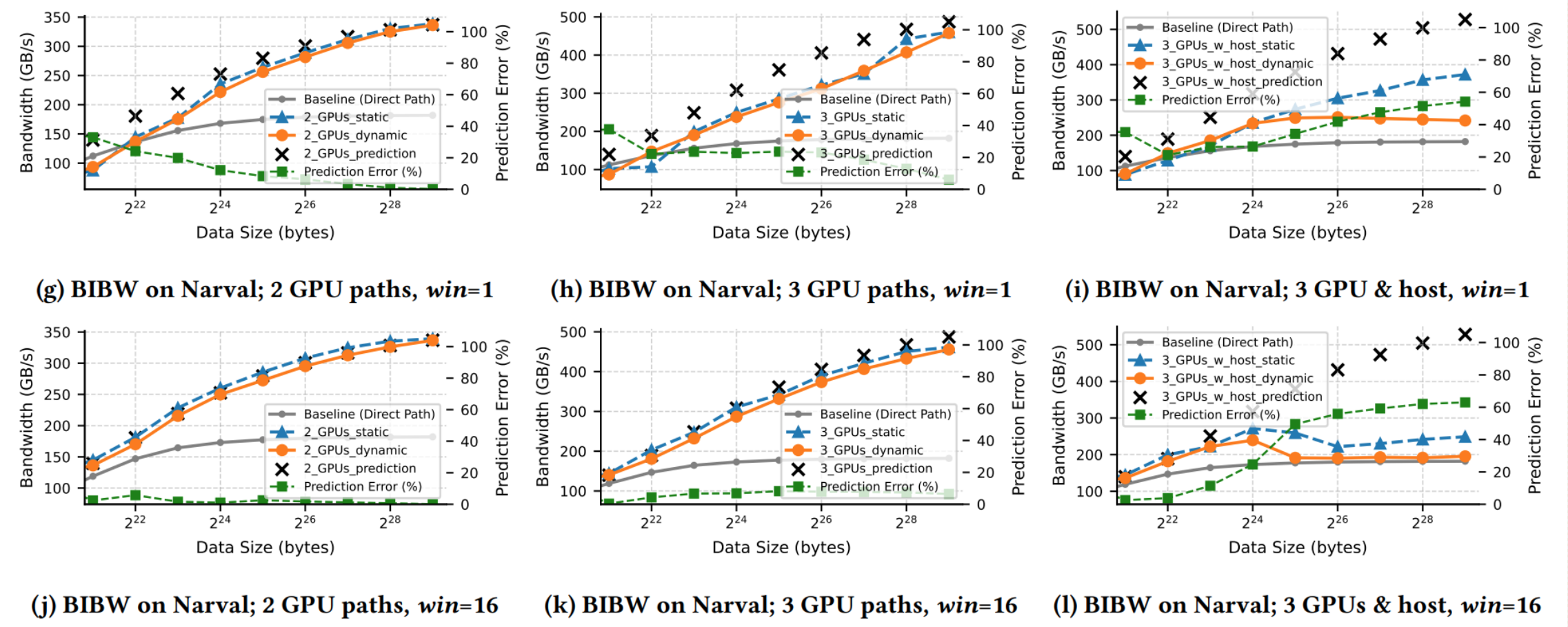


Figure 6: OMB bidirectional bandwidth tests on Narval. Comparing the predicted and two measured configurations.

OMB Allreduce Tests

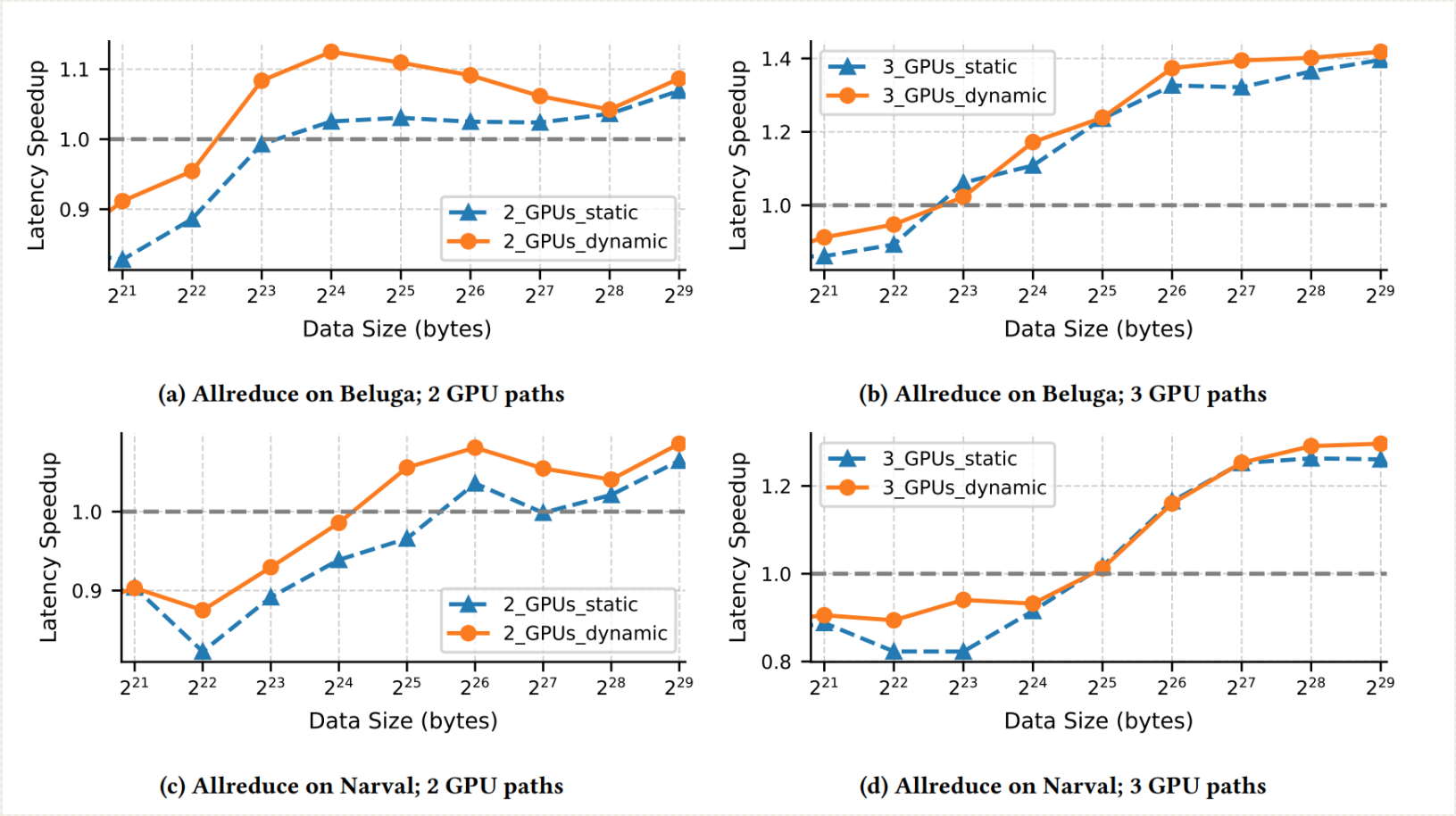


Figure 7: Latency speedup of MPI Allreduce tests on Beluga and Narval against MPI+UCC+UCX

OMB Alltoall Tests

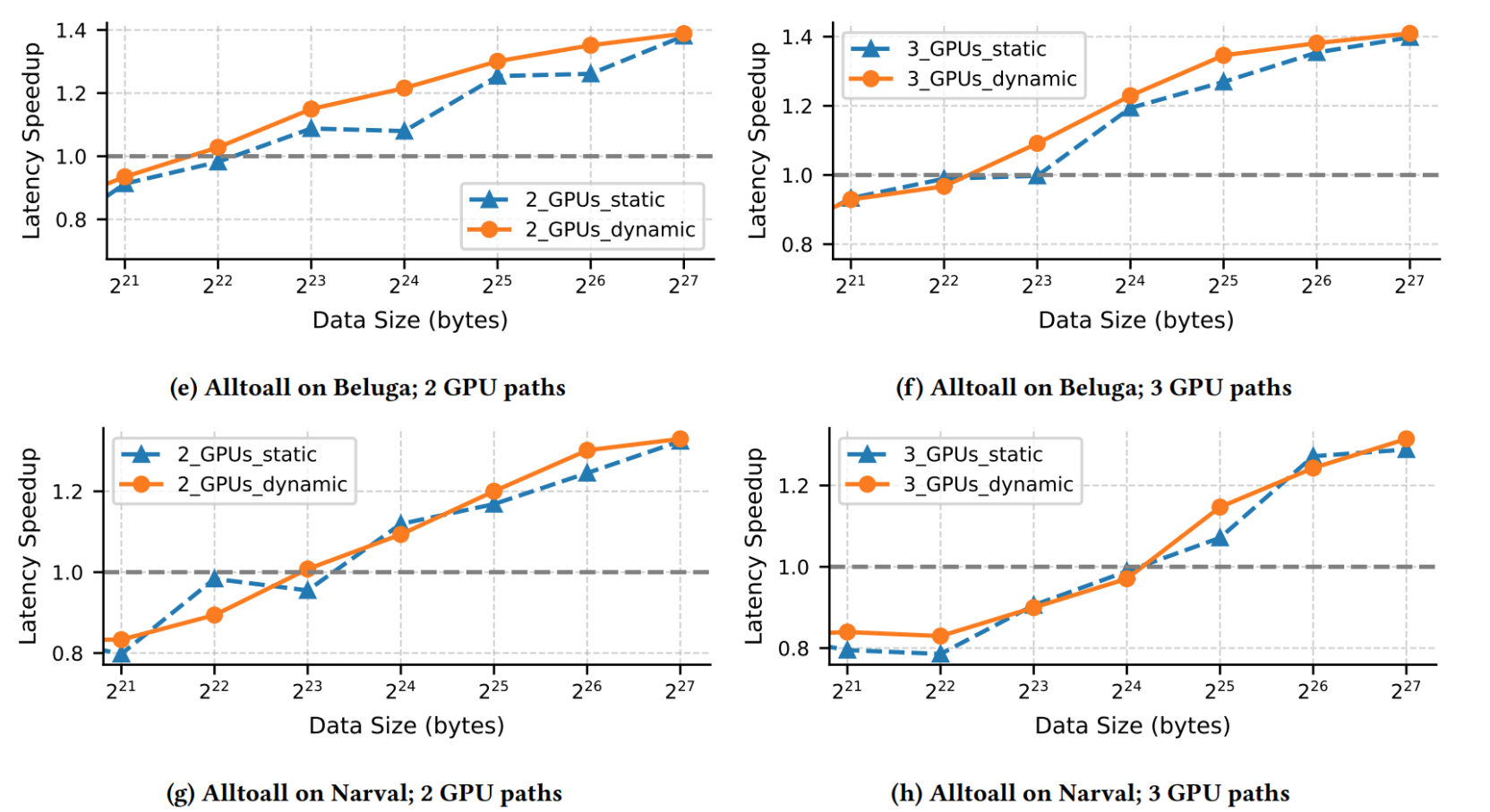


Figure 7: Latency speedup of MPI Alltoall tests on Beluga and Narval against MPI+UCC+UCX

1. Conclusions
2. Future Directions

Conclusions

- Designed and evaluated a performance model for multi-path intra-node GPU Communication
- Demonstrated up to 2.9x and 1.4x speedup comparing to single path default versions of P2P and Allreduce/Alltoall, respectively
- Achieving less than 6% error on average in predicting the optimal configuration.

Future Work

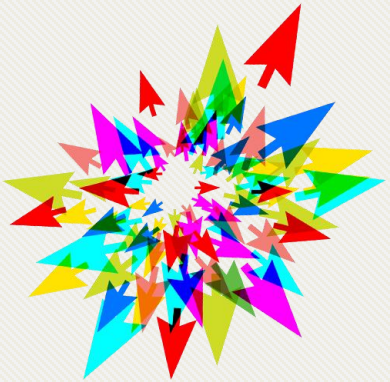
- Extend our model to support more complex communication patterns, such as multi-node settings.
- Explore the feasibility of utilizing other performance models, such as MaxRate, as the base of our model to capture more characteristics of the hardware.
- Investigate the application of our work on other architectures and interconnects, including NVswitch-based sysems and AMD GPUs.

Acknowledgments

- Digital Research Alliance of Canada
- Computations were performed on Narval and Beluga with support from Calcul Québec

NSERC CRSNG

compute canada | calcul canada



SciNet
ADVANCED RESEARCH COMPUTING at the UNIVERSITY OF TORONTO

SHARCNET™

Thank You!

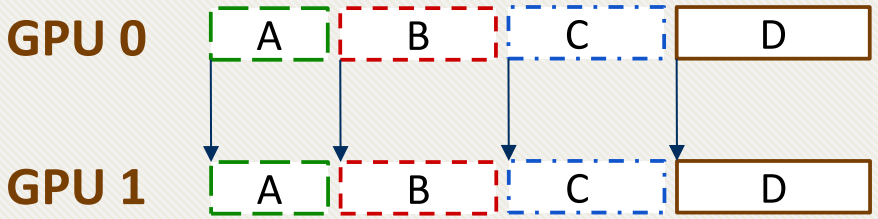
Instead of cursing the darkness, better light a candle!

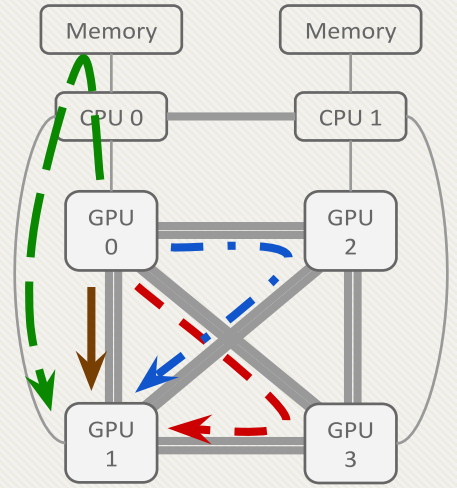
Questions, Comments, and Ideas are Welcome!

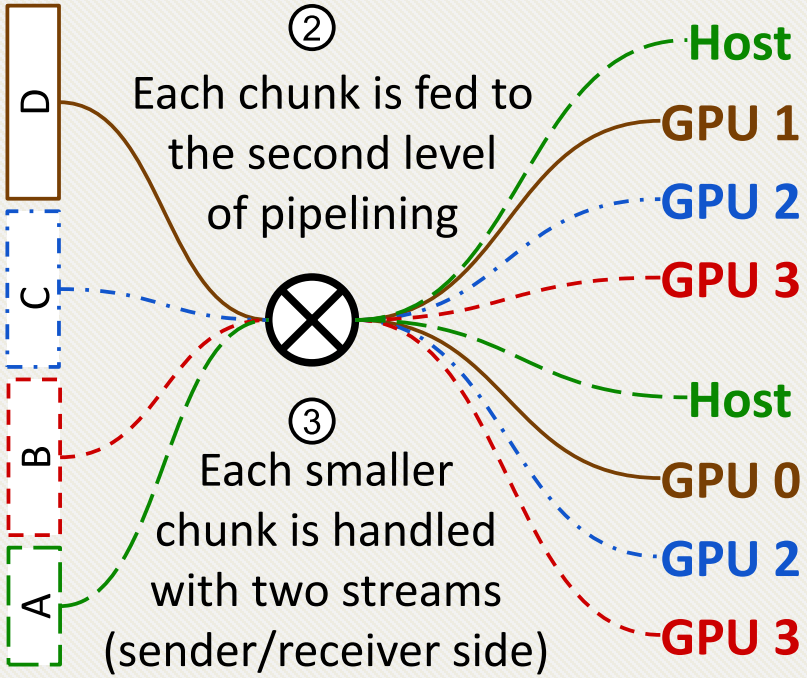
2-D Pipeline Design

Message is dynamically split.

① Chunk sizes vary depending on the # of paths and tunings







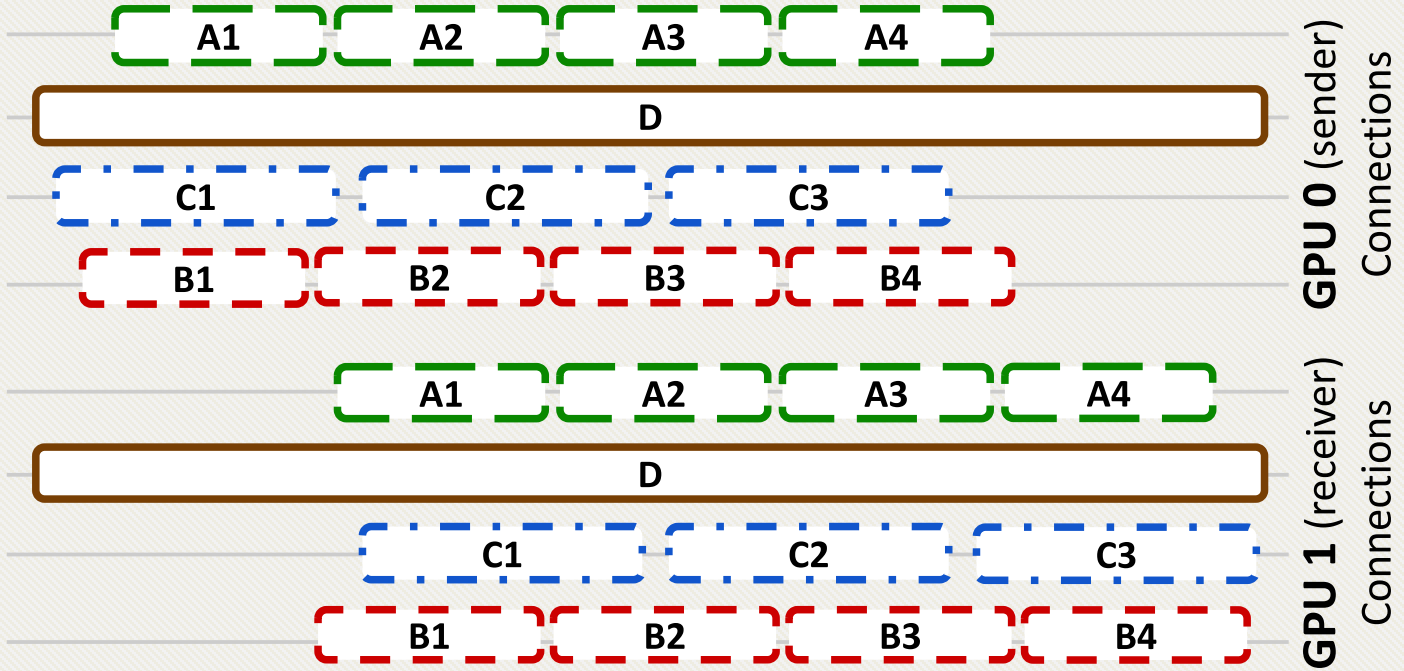


Figure 4: 2-D Pipeline in action