Experimental Validation of SimGrid Predictions for RTM Applications in Multi-Node Multi-GPU Environments

Arthur Andrade da Silva[1], **Vinícius Daniel Spadotto**[1], **Cristiano Alex Künas**[1], **Lucas Mello Schnorr**[1], **Phillipe Olivier Alexandre Navaux**[1]

[1]Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{aasilva, vinicius.spadotto, cakunas, schnorr, navaux}@inf.ufrgs.br

Abstract. *Simulation models calibrated on one hardware generation are reused without validation. A SimGrid/SMPI model of Fletcher, an anisotropic reverse time migration (RTM) application, calibrated on one GPU per node at 1 Gbit/s, predicted higher bandwidths raise the computing-time fraction, untested with multi-GPU nodes. We run Fletcher's CUDA backend on the Grid'5000 chuc cluster, four A100 GPUs per node, one to four nodes, at 1, 10, and 25 Gbit/s. The trend holds qualitatively, but diverges quantitatively with scale: overlap spreads from 1.3 to 38.2 points, with limited gains above 10 Gbit/s. A same-hardware simulation matches within 13% at 10–25 Gbit/s, but underestimates throughput by 40–50% at 1 Gbit/s beyond one node, from message sizes outside calibration. Direction generalizes across generations; magnitude does not.*

1. Introduction

Reverse Time Migration (RTM) produces subsurface images from seismic data recorded at the surface and is the reference method in oil and gas exploration. Its compute cost has made it a common workload in High-Performance Computing (HPC), motivating GPU implementations since earlier ports to CUDA [Micikevicius 2009]. Fletcher is among them, containing CPU and GPU backends. It solves wave propagation in anisotropic media with a tilted symmetry axis by finite differences over a regular 3D grid.

In distributed memory, the grid is partitioned across processes, and the stencil requires at each time step the values in the neighborhood of every point. Implementations satisfy this requirement in different ways. The one evaluated here exchanges ghost cells between adjacent subdomains via the MPI [Spadotto and Mello Schnorr 2026a]. The flexible decomposition strategy determines the ratio between the communication surface and the computational volume and, thus, how much of the exchange can be hidden under computation. GPU acceleration unbalances the two terms: time per step falls with higher computational throughput, while the exchanged volume remains the same, leaving less computation available to mask communication. On multi-GPU nodes, inter-node communication quickly limits halo exchange benefits and therefore scalability.

Fletcher's other limiting factors have already been characterized experimentally. [Rigon et al. 2024] overlapped single-node point-to-point GPU communication with grid computation, reducing the energy-delay product by up to 40%, and [Künas et al. 2024] overlapped input/output (I/O) checkpoints with computation, later evaluating compression to reduce I/O volume [Künas et al. 2025]. Communication across multiple nodes remains without direct measurement at the bandwidths of current hardware: the prior study of this implementation validated it up to 10 Gbit/s on CPU only, and up to 1 Gbit/s

with GPU [Spadotto and Mello Schnorr 2026a]. Beyond that point, the available evidence comes from simulation. A model built with SimGrid and its simulated MPI component (SMPI), calibrated on a cluster with one GPU per node and a 1 Gbit/s network, projected that higher bandwidths would raise the fraction of time devoted to computation and reduce total runtime [Spadotto and Mello Schnorr 2026b, Spadotto 2026].

The study by Spadotto and Mello Schnorr [2026b] is the only available simulation-based investigation into how inter-node bandwidth affects Fletcher's GPU performance at scale to date. Its main limitation is that validation occurs only on a limited hardware setup: one GPU per node with a 1 Gbit/s interconnect. As modern supercomputers typically comprise multi-GPU nodes and fast interconnects, we investigate whether the reach of the existing validation simulation still holds for platforms whose nodes contain four GPUs sharing a link up to twenty-five times faster than the initial study.

This paper reports on the validity envelope of that prediction by running Fletcher's CUDA backend across topologies from one to four nodes, under controlled bandwidth, with repetitions that quantify the variability of a shared infrastructure, and confronting the measurements against the equivalent simulated predictions. The predicted trend survives the change of hardware generation in direction but not in magnitude: the spread in effective overlap across bandwidths grows from 1.3 percentage points at one node to 38.2 at four, a scale dependence the original model does not express. A second simulation reuses the same calibration, described with a platform file for *chuc* instead of *poti*, and isolates where that divergence comes from: the network component alone is accurate within 13% at 10 and 25 Gbit/s, but underestimates throughput by 40 to 50 percent at 1 Gbit/s once more than one node is involved, a gap tied to message sizes the original calibration was never validated against. Both experiments share the infrastructure validation, per-repetition verification of network shaping, and provenance tracking of the framework released with this article [Cherrueau et al. 2022].

Section 2 covers domain decomposition, the conditions under which the Fletcher model was calibrated and validated, and related work. Section 3, the implementation and hypotheses. Section 4 presents our evaluation methodology. Section 5 presents the results and a discussion. Finally, Section 6 presents the main findings and future work.

2. Background and Related Work

Fletcher's domain decomposition determines how much data is communicated relative to the amount of computation. The calibration of the Fletcher model establishes the conditions under which its predictions have already been verified. The related work situates this study among earlier attempts to confront simulation with real execution.

2.1. Fletcher and Domain Decomposition

RTM propagates the seismic wavefield forward in time from the source and backward from the receivers. Cross-correlation between the two fields locates geological reflectors with greater precision than simplified methods [Zhou et al. 2018]. In distributed memory, the regular 3D grid is partitioned across processes, and subdomain borders are exchanged at every step. Under slices (1D), each process maintains two full-domain faces as communication surfaces, which do not shrink as the number of processes grows (Figure 1). Under blocks (3D), the decomposition used by [Pearson et al. 2020] in a related stencil application, the surface per subdomain decreases with partitioning, at the cost of more neighbors per process; this surface-to-volume ratio is the mechanism by which node count affects

performance, independently of bandwidth. The border data exchanged at each step are the ghost cells (Figure 1), whose volume depends on the adopted wave-equation formulation. In the isotropic case, propagation follows the acoustic wave equation

$$\frac{\partial^2 p(\vec{x},t)}{\partial t^2} = v(\vec{x})^2 \nabla^2 p(\vec{x},t), \tag{1}$$

where $p(\vec{x},t)$ is pressure, $v(\vec{x})$ is velocity, and $\vec{x} = (x,y,z)$ are the spatial coordinates. Fletcher's method extends this formulation to media with a tilted symmetry axis (TTI), replacing it with a pseudo-acoustic system coupling a main pressure field and an auxiliary one, parameterized by the medium's dip and azimuth angles [Fletcher et al. 2009]; this increases the volume of borders transferred per iteration relative to the acoustic case. Asynchronous point-to-point calls hide part of that cost by exposing the domain interior to computation while borders are in flight, a pattern used since the earliest multi-GPU finite-difference implementations in CUDA [Micikevicius 2009] and adopted here [Spadotto and Mello Schnorr 2026a].

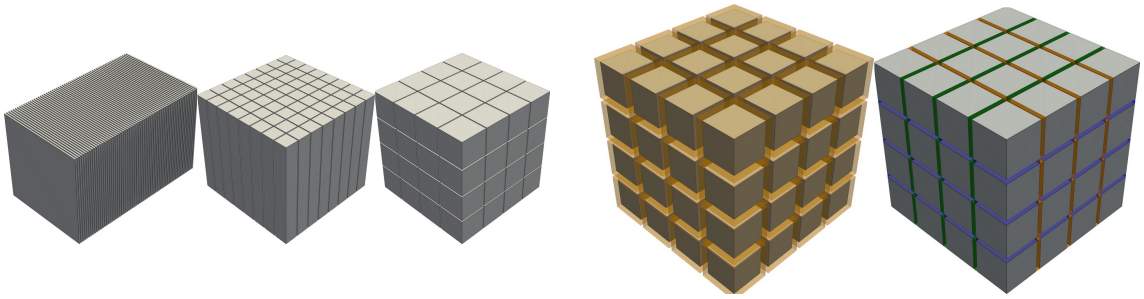

Figure 1. 3D domain decomposition (left) and ghost cells under block decomposition (right). Adapted from [Spadotto 2026].

2.2. SimGrid/SMPI and the Fletcher Calibration Point

The SimGrid/SMPI component [Casanova et al. 2014] runs unmodified MPI applications under simulated network and computation models, projecting performance on physically inaccessible platforms. The prior Fletcher study chose SMPI because it emulates the real source code, preserving effects that coarser-grained models ignore, such as PCIe bus transfers and network contention [Spadotto and Mello Schnorr 2026b]. Prior validation of SimGrid/SMPI covers flow-level TCP models for *Grid* and *Cloud* settings [Velho et al. 2013], unmodified MPI applications [Degomme et al. 2017], the calibration methodology used for the Fletcher model, error below approximately 15% for messages up to 1 GiB [Cornebize 2021], and large-scale platforms without accelerators, including Tibidabo [Bédaride et al. 2013], BigDFT [Degomme et al. 2017], and also experiments with StarPU [Stanisic et al. 2015]. Extension to accelerators covers a different scope: [Boughzala et al. 2020] model CUDA kernel energy consumption on a single GPU, without inter-node communication. Inter-node communication on multi-GPU nodes remains limited in coverage [Unat et al. 2026], a condition the Fletcher prediction requires.

2.3. Related Work

[Keller Tesser et al. 2017] validated dynamic load-balancing predictions for Ondes3D, an iterative seismic finite-difference application over-decomposed in AMPI/Charm++ and structurally akin to Fletcher, against real execution on the Parapluie cluster of Grid'5000,

in two phases: controlled validation, with simulated-versus-real makespan errors between roughly 1% and 9%, followed by simulation-guided exploration of parameters whose exhaustive testing on real hardware would be prohibitive. This work adopts the same incremental-validation principle under a different condition. Simulation and execution in Tesser et al. share the same hardware generation, with no GPUs involved; here, the execution target differs from the one the model was calibrated on. Within Fletcher itself, the bottlenecks already measured delimit what remains to be measured. None of those prior efforts [Rigon et al. 2024, Künas et al. 2024, Künas et al. 2025] reaches inter-node communication via MPI (Table 1). Closest to this work, [Spadotto and Mello Schnorr 2026a] reports simulated and real results for the multi-node MPI implementation, but only within its own calibration range: up to 10 Gbit/s on CPU, up to 1 Gbit/s with one GPU per node. Two things distinguish the present article: it measures outside that range, on four GPUs per node at up to twenty-five times the calibrated bandwidth; and it re-runs the same model against the test platform, separating the error owed to the network calibration from that owed to the hardware change (more details in Section 5.3).

Table 1. Comparison between this work and the closest related literature.

Work	Real hardware	Multi-GPU/node	Automated framework
[Keller Tesser et al. 2017]	Yes	No	No
[Rigon et al. 2024]	Yes	Yes (intra-node)	No
[Künas et al. 2024]	Yes	n/a	No
[Künas et al. 2025]	Yes	n/a	No
[Spadotto and Mello Schnorr 2026a]	Yes	Yes (multi-node only)	Yes
This work	Yes	Yes	Yes

3. The MPI-Based Fletcher Implementation Used in This Work

The parallel implementation and its physical evaluation at PCAD set the calibration point; its simulated extension yields the hypotheses this work confronts with real execution.

3.1. Implementation

The implementation evaluated here is the one built by Spadotto and Mello Schnorr (2026a), on which the simulation model was calibrated. It adopts a coordinator-worker MPI model, with cubic block decomposition when the process count factors evenly into three dimensions and slices otherwise, a distinction decisive for scalability (Section 3.2). Communication and computation overlap through asynchronous ghost cell exchange (Section 2.1), with backends in OpenMP for CPU and CUDA for GPU, the latter run here, where only ghost cell buffers cross the host-device boundary [Micikevicius 2009, Pearson et al. 2020]. Numerical correctness has been validated by [Spadotto and Mello Schnorr 2026a].

3.2. The Limits of the Prior Evaluation and the Resulting Hypotheses

The physical evaluation carried out by [Spadotto and Mello Schnorr 2026a] took place at the High-Performance Computing Park (PCAD) of UFRGS, on two clusters specified in Table 2: the CPU cluster *cei*, six nodes and 10 Gbit/s links, and the GPU cluster *poti*, five nodes, one NVIDIA RTX 4070 per node, and a 1 Gbit/s network. The two produced opposite behaviors. On *cei*, communication was masked by computation, with computational load fractions of approximately 75% at 256^3 and 87% at 512^3; on *poti*, that fraction stagnated around 20% regardless of problem size, with the 1 Gbit/s link

dominating runtime even at 512^3, since computation there was fast enough to no longer hide communication. Restoring masking would have required bandwidth PCAD did not offer: the evaluation was limited to five GPUs, one per node, and both multi-GPU decomposition and networks above 1 Gbit/s remained physically inaccessible. To reach those conditions, [Spadotto and Mello Schnorr 2026b] calibrated a SimGrid/SMPI model following Cornebize's methodology [Cornebize 2021], with network calibration error below approximately 15% for messages up to 1 GiB, and used it to evaluate the application at 10 and 100 Gbit/s, plus a $\{2, 2, 2\}$ block decomposition with eight workers the five-node cluster could not accommodate. Simulated computational load fractions reached approximately 65–80% at 10 Gbit/s and, at 100 Gbit/s, 85–90% at 256^3 and above 90% at 512^3, against the 20% measured at 1 Gbit/s. At 512^3, the eight-worker block decomposition attained a simulated throughput of 8,309 MSamples/s at 10 Gbit/s and 9,374 MSamples/s at 100 Gbit/s, while five-worker slices remained below the single-node baseline of 3,870 MSamples/s under both conditions.

These simulated results support the two hypotheses assessed in the present work. 1) Masking recovers as bandwidth increases, reversing the stagnation measured at 1 Gbit/s, but saturates above 10 Gbit/s, indicated by the mere 13% gain from a tenfold bandwidth increase. 2) Decomposition topology weighs more than available bandwidth, since slices do not surpass single-node execution under any simulated condition. Neither hypothesis was previously tested against real execution: the calibration infrastructure could not reproduce the required conditions, in particular, multiple GPUs per node and bandwidth above 1 Gbit/s. This work performs that test on the Grid'5000 *chuc* cluster.

4. Methodology

This section evaluates those hypotheses through real executions, organized in two phases of progressive departure from the calibration point (Section 4.1: platform; 4.2: metrics; 4.3: threats to validity), following the incremental validation principle of Tesser et al. (Section 2.3), which quantifies fidelity first under the conditions closest to calibration. In the first phase, the CUDA backend runs on A100 hardware with bandwidths of 1, 10, and 25 Gbit/s, at problem sizes in the prior study's GPU range (128^3, 256^3, 512^3), using block decomposition. *chuc*'s four-GPUs-per-node granularity does not accommodate a direct replica of the prior study's five-process slices configuration, so this phase concentrates on the dominant topology of the original simulated results. The 1 and 10 Gbit/s levels correspond to PCAD's physical condition and to the first simulated scenario; 25 Gbit/s covers the region of diminishing returns indicated by the simulation, obviating the 100 Gbit/s point. In the second phase, the same problem sizes run under the cluster's native bandwidth, a condition not evaluated in the original study. Absent a simulated counterpart, results are reported descriptively and supported by the per-run decomposition of time between MPI and CUDA. A strong-scaling sweep beyond the tested node range, using the original simulation and later verified on real hardware, was not carried out in this campaign and is left for future work (Section 6).

4.1. Platform

The experiments run on the Grid'5000 *chuc* cluster, at the Lille site [Balouek et al. 2013], with eight nodes, each with an AMD EPYC 7513, four NVIDIA A100-SXM4-40GB GPUs, 512 GiB of RAM, and links of up to 25 Gbit/s over 2× ConnectX-5 NICs, hardware capable of GPUDirect Remote Direct Memory Access (RDMA) [Potluri et al. 2013] not exercised here. The OpenMPI/UCX stack was compiled `--without-cuda` and

links no CUDA-aware UCX transport module, so no code path moves MPI buffers directly from device memory. All four GPUs of every allocated node are used in all configurations, so node count and GPU count scale together (Table 3). The campaign is capped at four of the cluster's eight nodes, a limit set by allocation constraints due to queue competition on this shared infrastructure. Table 2 compares this environment (last column) against the PCAD clusters, which differ jointly in GPU architecture, GPUs per node, interconnect, and communication path. The calibration point uses a consumer-class GPU (RTX 4070), whereas *chuc* employs datacenter-class accelerators (A100) with a distinct memory hierarchy. Hence, the extrapolation is subject to a change of network and accelerator class simultaneously.

Table 2. Hardware configuration used in the evaluation of the prior and this work.

	PCAD *cei* (CPU)	**PCAD *poti* (GPU)**	**Grid'5000 *chuc***
Nodes	6	5	8
Processor	2× Xeon E5-2699, 2.0 GHz	Core i7-14700KF, 3.4 GHz	AMD EPYC 7513 (Zen 3)
Cores/threads per node	24 / 48	20 / 28	32 cores, 1 CPU/node
RAM per node	256 GB DDR4	96 GB DDR5	512 GiB
GPUs per node	n/a	1× RTX 4070 (12 GB)	4× A100-SXM4-40GB
Network	10 Gbit/s	1 Gbit/s	25 Gbit/s (2× ConnectX-5)
Used in	Prior work	Prior work	This work

An experimental framework sustains reproducibility. Before each run, it validates the allocated infrastructure, including full GPU availability on each node, and aborts on a degraded allocation. The bandwidth of each shaping condition is imposed with `tc` (Linux Traffic Control) [Salim 2015] and a `tbf` queueing discipline, recorded in the metadata of every repetition; an independent `iperf3` [Energy Sciences Network (ESnet) 2026] probe exists but is not persisted systematically (Section 4.3). CSV-parameterized execution, checkpointing/resuming, and provenance tracking complete the design, following Grid'5000 orchestration practices such as EnOSlib [Cherrueau et al. 2022]. Source code of the Fletcher MPI version is available online[1].

4.2. Metrics

The primary metric is the **effective overlap ratio (ME)**, the fraction of wall-clock time per process devoted to computation rather than blocking MPI calls. We compute ME as the ratio between the computation time summed over all ranks and the total duration summed over all ranks. Less sensitive to GPU architecture than absolute throughput, it is reported per topology, and bandwidth in Table 3 and Figures 3 and 4, following the MPI overlap quantification of [Denis and Trahay 2016], revised by [Medvedev 2022] and used by [Spadotto and Mello Schnorr 2026b]. The **fidelity error** is the percentage difference between simulated prediction and real measurement under equivalent configurations, computed two ways (Section 5.3): with the compute component sampled live on real *chuc* hardware, and with it replaced by a fixed per-iteration constant from this campaign's traces. Both share the same network calibration, so divergence between them isolates the compute substitution from the network model. **Throughput** and **makespan** give the absolute reading, comparable to the prior study's MSamples/s figures. The **structural confirmation of shaping** verifies that nominal conditions were correctly configured, without independent per-run bandwidth measurement (Section 4.1). The **MPI-versus-CUDA time decomposition** grounds mechanistic explanations for observed divergences.

[1] `https://github.com/ArthurAndradee/simgrid-rtm-multigpu-validation/`

Table 3. Throughput (MSamples/s) and effective overlap ratio ME (%) per topology and bandwidth, mean ± 95% confidence interval (CI) over 5 repetitions, at the reference size N. Each node contributes four A100 GPUs; partitioning column gives the process grid used.

			Throughput (MSamples/s)			ME (%)		
Nodes (GPUs)	**Partitioning**	N	**1 Gbit/s**	**10 Gbit/s**	**25 Gbit/s**	**1 Gbit/s**	**10 Gbit/s**	**25 Gbit/s**
1 (4)	`1x2x2`	1344	3266.5 ± 100.1	3325.8 ± 98.0	3298.5 ± 49.9	84.7 ± 0.9	83.7 ± 3.7	85.1 ± 3.0
2 (8)	`2x2x2`	1728	4886.9 ± 139.7	7783.2 ± 170.0	7398.0 ± 749.8	44.9 ± 1.0	78.3 ± 2.9	74.4 ± 6.6
3 (12)	`2x2x3`	1920	2863.0 ± 6.1	5915.7 ± 143.0	5960.8 ± 74.8	28.9 ± 0.3	62.2 ± 0.8	63.1 ± 0.4
4 (16)	`2x2x4`	2176	3445.9 ± 40.8	8107.0 ± 394.8	8054.0 ± 247.9	26.2 ± 0.2	63.7 ± 2.6	64.3 ± 2.0

4.3. Threats to Validity

Hardware generation, network topology, and communication path vary jointly between calibration and test, so observed divergence reflects their combination, not an isolated factor. The nominal bandwidth labels also differ in meaning across platforms; the calibration point ran one process per node with the link to itself, whereas here four processes share the shaped link, so the effective per-process bandwidth at a given nominal value is not the same quantity. Software stack differences (MPI, CUDA, compiler versions) are documented by the framework's version fingerprint but not eliminated, and the Grid'5000 shared allocation model introduces measurement variability absent in simulation; each configuration is run five times, reported with the variability observed across repetitions [Hoefler and Belli 2015]. Shaping fidelity is verified structurally, through the `qdisc` configuration recorded per repetition; two isolated diagnostic probes predating the campaign indicate measured-to-nominal ratios near 0.94 at 1 and 10 Gbit/s, but not at 25 Gbit/s or during the campaign itself. I/O contention for the same link or storage, already characterized separately [Künas et al. 2024, Künas et al. 2025], is not measured here, and recalibrating a model for this hardware generation was ruled out by scope.

5. Results and Discussion

This section evaluates twelve combinations of topology and bandwidth. Single-node behavior establishes the baseline without network communication, multi-node behavior shows how the prediction's validity evolves with scale, and the discussion confronts both against the original projection.

Each topology is evaluated at a single reference problem size, following a weak-scaling design that keeps video memory (VRAM) occupancy per GPU roughly constant, at 80–87% of the A100's 40 GB capacity: $N = 1344$ at one node, $N = 1728$ at two, $N = 1920$ at three, and $N = 2176$ at four (Table 3). This isolates the effect of node count from that of local block size per GPU, which also influences the communication-to-computation ratio [Spadotto 2026]. A full sweep over N per topology was dropped, as it would multiply configurations without altering the comparison at constant occupancy.

5.1. Single-Node Behavior

In the single-node configuration, whose four GPUs are arranged as a `1x2x2` process grid (Table 3), ME is 84.7% at 1 Gbit/s, 83.7% at 10 Gbit/s, and 85.1% at 25 Gbit/s. The 1.3 percentage point spread across the three bandwidths is smaller than the 95% confidence interval of any individual measurement, of up to 3.7 percentage points, and throughput varies between 3266.5 and 3325.8 MSamples/s with no systematic relation to bandwidth.

Figure 2 arranges the twelve configurations as a grid, rows for node count and columns for bandwidth, each panel split into real (left) and simulated (right) execution,

one horizontal bar per MPI rank, elapsed time normalized by total duration, and annotated at the top, colors decomposing computation from the three MPI phases. One and two nodes use the reference sizes of Table 3 ($N = 1344, 1728$); three and four use the reduced reference size of Table 4 ($N = 1800$), re-measured so each panel compares real and simulated execution at the same N. In the first row, the three real panels have equivalent composition and durations of 68, 66, and 67 s, confirming the absence of a bandwidth effect at one node: with all four ranks on the same physical node, MPI traffic stays intra-node and never reaches the shaped interface. The simulated panels track the real ones closely (71, 74, 72 s), an early sign of the fidelity discussed in Section 5.3.

Figure 2. Temporal composition per rank, real (left) and simulated (right) execution, by node count (rows) and bandwidth (columns); median of 5 repetitions. Rows 3–4 use the reduced reference size of Table 4.

5.2. Multi-Node Behavior and Evolution with Scale

At two nodes (`2x2x2`, second row of Figure 2), ME is 78.3% at 10 Gbit/s and 74.4% at 25 Gbit/s, but drops to 44.9% at 1 Gbit/s, a spread of 33.4 percentage points across bandwidths, beyond what any individual confidence interval would explain (Figure 3). The spread grows to 34.2 percentage points at three nodes (28.9% against 62.2%–63.1%) and 38.2 at four (26.2% against 63.7%–64.3%). The jump occurs upon introducing the first additional node, up from 1.3 percentage points at one node, and subsequent growth is gradual.

Along each row of Figure 2, reducing bandwidth from 25 to 1 Gbit/s converts continuous stretches of computation into `MPI_Waitall` bands spread across all ranks, an effect absent in the first row and progressively more extensive in the following ones. Along each column, increasing the node count widens that waiting fraction even under

ample bandwidth, to a lesser degree. At the full reference sizes of Table 3, the makespan ratio between 1 and 10 Gbit/s goes from 1.03 at one node (68 against 66 s) to 1.67 at two (100 against 60 s), 2.11 at three (240 against 114 s), and 2.42 at four (290 against 120 s); 10 and 25 Gbit/s differ by at most 2 s in any topology.

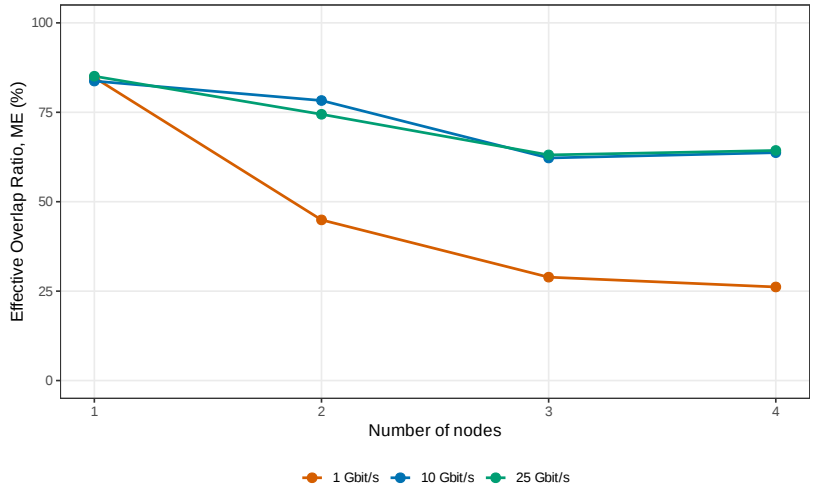

Figure 3. Effective overlap ratio (ME) as a function of node count, for the three bandwidth conditions. Node counts of 1–4 correspond to 4, 8, 12, and 16 A100 GPUs and to reference sizes $N = 1344/1728/1920/2176$.

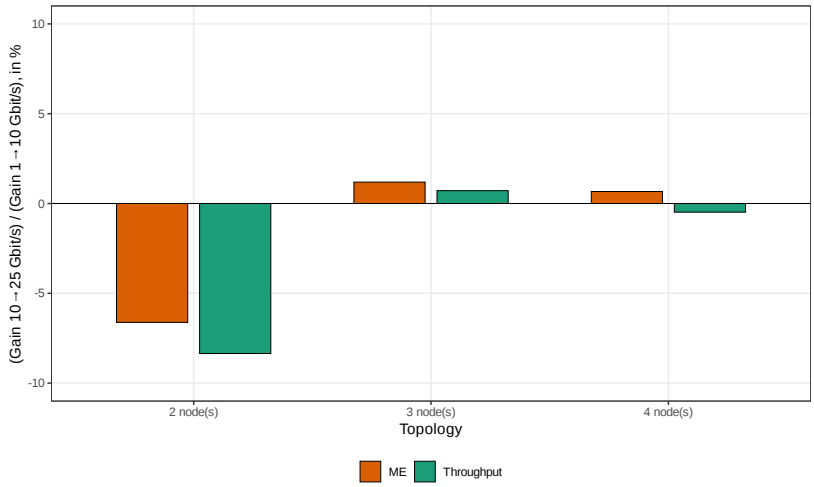

Figure 4. Ratio between the relative gain of the second bandwidth transition (10→25 Gbit/s) and that of the first (1→10 Gbit/s), per topology, for effective overlap and throughput. Equal gain in both transitions yields 100%.

Figure 4 compares the two bandwidth transitions available, from 1 to 10 Gbit/s and from 10 to 25 Gbit/s, by the ratio between the relative gain of the second and that of the first. The gain from the second transition is roughly 8% of the first at two nodes, and approaches zero at three and four; the reading holds from two nodes onward, since at one node the gain of the first transition is negligible and the ratio is not physically interpretable. At three nodes, the 0.85 percentage point difference between 10 and 25 Gbit/s is the only one, among the multi-node topologies, that exceeds the corresponding 95% confidence interval. This pattern of diminishing returns above 10 Gbit/s matches, in direction,

Table 4. Fidelity error (simulated vs. real) under online GPU sampling. One and two nodes use the reference size of Table 3; three and four use a reduced reference size ($N = 1800$, VRAM-limited under SMPI's per-node process sharing). Throughput error is relative; ME error is in percentage points.

Nodes	Band	N	Real thr.	Sim. thr.	Err. thr. (%)	Real ME (%)	Sim. ME (%)	Err. ME (pp)
1	1 Gbit/s	1344	3266.5	3153.4	-3.5	84.7	81.7	-3.0
1	10 Gbit/s	1344	3325.8	3051.6	-8.2	83.7	81.3	-2.4
1	25 Gbit/s	1344	3298.5	3156.4	-4.3	85.1	79.5	-5.6
2	1 Gbit/s	1728	4886.9	2440.1	-50.1	44.9	23.5	-21.4
2	10 Gbit/s	1728	7783.2	7603.3	-2.3	78.3	78.0	-0.3
2	25 Gbit/s	1728	7398.0	7422.7	$+0.3$	74.4	78.1	$+3.7$
3	1 Gbit/s	1800	2505.0	1412.3	-43.6	25.9	13.5	-12.5
3	10 Gbit/s	1800	6604.6	6979.7	$+5.7$	68.1	69.6	$+1.5$
3	25 Gbit/s	1800	6604.4	6231.5	-5.6	69.1	69.8	$+0.6$
4	1 Gbit/s	1800	2416.3	1439.0	-40.4	18.3	12.4	-5.9
4	10 Gbit/s	1800	7996.9	7044.5	-11.9	60.9	65.6	$+4.7$
4	25 Gbit/s	1800	8420.2	7338.3	-12.8	60.7	65.7	$+5.0$

the one projected by the prior study's simulated extension (Section 3.2); that projection was calibrated on different hardware and under the platform asymmetry discussed in Section 4.3, so the match is read here as qualitative, not quantitative.

Per-GPU throughput shows that the efficiency loss with scale is not confined to the most constrained bandwidth. At 1 Gbit/s, it falls from 816.6 to 215.4 MSamples/s between one and four nodes, a 73.6% reduction. At 10 and 25 Gbit/s, the drop is approximately 39% over the same interval, but it is not monotonic: a gain from one to two nodes ($+17.0\%$ and $+12.1\%$) is followed by a drop from two to three (-49.3% and -46.3%) and stabilization between three and four. The two-to-three drop, present at all three bandwidths, is larger than the remaining transitions. The slope of ME as a function of node count is -19.2 percentage points per node at 1 Gbit/s, against -7.6 and -7.4 at 10 and 25 Gbit/s, a degradation roughly 2.5 times steeper under constrained bandwidth; with only four topologies per bandwidth, these slopes describe the observed trend without a goodness-of-fit statistic.

5.3. Simulated Fidelity

Section 5.2 compares real execution on *chuc* against a prediction calibrated on different hardware; the remainder of this section isolates the model itself. Table 4 and Figure 2 reuse the same SimGrid/SMPI model, described with a platform file for *chuc* instead of *poti*, run under online GPU sampling and compared against real execution at the reference sizes of Section 5.1. Errors stay within 13% in most configurations, evidence the network model itself is broadly sound; the exceptions occur at 1 Gbit/s and are discussed below.

At 10 and 25 Gbit/s, the real and simulated panels of Figure 2 agree closely across all four node counts, reproducing the same computation-dominated composition. At 1 Gbit/s the simulated duration exceeds the real one beyond one node, most visibly at three and four, where simulated `MPI_Waitall` bands extend well past the real ones.

The exception at two nodes recurs at three and four. Throughput at 1 Gbit/s is underestimated by 40 to 50 percent, and ME by 6 to 21 percentage points, while errors at 10 and 25 Gbit/s stay below 13 percent for throughput and under 6 percentage points for ME. The gap is consistent across every multi-node configuration measured, pointing to a real limitation of the network calibration at low bandwidth rather than noise from a single run. It does not appear at one node, where no traffic crosses the physical network

(Section 5.1), supporting a network-side rather than compute-side cause.

A second, independent check isolates the network component directly. All twelve configurations were also evaluated with the compute component replaced by a fixed per-iteration constant extracted from this campaign's own traces (Section 4.2), needing no GPU access. At 1 Gbit/s, where the network genuinely limits throughput for two nodes or more, throughput is underestimated by 37% to 44%, matching the online-sampling result and pointing to the network model rather than the GPU sampling method as the source. At 10 and 25 Gbit/s, computation dominates enough in this injected-constant model that both bandwidths become indistinguishable in simulated time; the resulting error there reflects halo messages of approximately 12 MB, far outside the 64 KB largest message size in the calibration, and is not informative about the network model. This single execution per configuration carries no confidence interval, since the injected constants are fixed means, not repeated measurements.

5.4. Discussion

The predicted direction survives the change of hardware generation. Under any bandwidth, more nodes reduce ME, and the reduction is sharper the more constrained the bandwidth, consistent with the dependence of communication surface and computational volume on the number of neighbors per process established by [Spadotto 2026] for block decomposition. In a 3D block decomposition, the neighbor count per process is bounded at 26, reached once the grid has at least three subdomains along every axis; growing the node count still adds neighbors and data exchanged per iteration.

Its magnitude does not survive the same change. Quantitative validity diverges across bandwidths as scale increases. Under constrained bandwidth, the lengthening of the `MPI_Waitall` bands along the first column of Figure 2 suggests network capacity ceasing to keep up with the volume exchanged per iteration as more processes communicate. Above 10 Gbit/s the additional benefit is small; the same-hardware simulation of Section 5.3 attributes part of that pattern to a network model not calibrated for messages this large, rather than to the real network itself.

The remaining efficiency loss cannot be attributed to bandwidth alone. The per-GPU efficiency drop between two and three nodes persists under ample bandwidth. One explanation is the change in surface-to-volume ratio when the process grid moves from cubic (`2x2x2`, two nodes) to non-cubic (`2x2x3`, three), discussed in Section 2.1 and not isolated here. The original model, calibrated with one GPU per node and no communication beyond a single neighbor, approximates the direction of the bandwidth effect but not its magnitude.

For dimensioning, extrapolate the direction and measure the magnitude. Dimensioning decisions for stencil applications based solely on predictions calibrated at reduced scale tend to underestimate efficiency loss as node count grows, especially under constrained bandwidth. At the tested scales, the return on investing above 10 Gbit/s is limited; where Section 5.3 applies, part of the low-bandwidth error is attributable to the network calibration itself, which points to recalibrating for larger messages as a concrete next step, without ruling out other contributing factors.

6. Conclusion

This work determined the validity envelope of a simulation prediction reused beyond the hardware generation on which it was calibrated. The SimGrid model of Fletcher,

calibrated at 1 Gbit/s with one GPU per node [Spadotto and Mello Schnorr 2026b], was confronted with the CUDA backend on the Grid'5000 *chuc* cluster, under three bandwidth conditions and topologies from one to four nodes. The predicted masking trend holds qualitatively beyond the calibration generation, but its quantitative validity depends jointly on bandwidth and scale. The spread in ME across bandwidths grows from 1.3 percentage points at one node to 33.4 at two, 34.2 at three, and 38.2 at four, and the makespan ratio between 1 and 10 Gbit/s goes from 1.03 to 2.42 over the same interval. Above 10 Gbit/s the additional benefit is limited, with a gain of roughly 8% of that obtained in the first transition at two nodes and near zero at three and four. The degradation of ME with scale is -19.2 percentage points per node under 1 Gbit/s, against -7.6 and -7.4 under 10 and 25 Gbit/s, a dependence the original model does not express.

A second, same-hardware simulation traces part of that dependence to the network calibration rather than to physical network limits. Throughput and overlap match real execution within 13% at 10 and 25 Gbit/s, but underestimate throughput by 40 to 50 percent at 1 Gbit/s once more than one node is involved, a gap traced to halo messages roughly two orders of magnitude beyond the largest breakpoint the calibration covers. The direction of a prediction extrapolates across hardware generations; its magnitude requires measurement, and at low bandwidth, that measurement can be substituted by recalibrating the network model for larger messages. A stencil application with block decomposition bounds these findings, a reduced set of problem sizes, a ceiling of 25 Gbit/s, and GPU generation, network topology, and communication path varying jointly. Future work includes recalibrating the network model for message sizes near 12 MB, and a strong scaling sweep verified against real hardware at each additional node.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001, and by the CNPq/MCTI/FNDCT Universal project no. 408755/2025-3, and partially by Petrobras under Agreement no. 4600665938, project GPPD/UFRGS 2.20.2205 (361), Execution of Seismic Applications on Heterogeneous Computing Nodes, EXAFLOP II. The authors thank PCAD/UFRGS for infrastructure access in the prior study's evaluation, and the Grid'5000 testbed (*chuc* cluster), supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and other partners (see `https://www.grid5000.fr`).

References

Balouek, D., Carpen Amarie, A., Charrier, G., Desprez, F., Jeannot, E., Jeanvoine, E., Lèbre, A., Margery, D., Niclausse, N., Nussbaum, L., Richard, O., Pérez, C., Quesnel, F., Rohr, C., and Sarzyniec, L. (2013). Adding virtualization capabilities to the Grid'5000 testbed. In Ivanov, I. I., van Sinderen, M., Leymann, F., and Shan, T., editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer.

Bédaride, P., Degomme, A., Genaud, S., Legrand, A., Markomanolis, G. S., Quinson, M., Stillwell, M., Suter, F., and Videau, B. (2013). Toward better simulation of MPI applications on Ethernet/TCP networks. In *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation (PMBS 2013)*, volume 8324 of *Lecture Notes in Computer Science*, pages 1–20, Cham. Springer.

Boughzala, D., Lefèvre, L., and Orgerie, A.-C. (2020). Predicting the energy consumption of CUDA kernels using SimGrid. In *2020 IEEE 32nd International Symposium on*

Computer Architecture and High Performance Computing (SBAC-PAD), pages 191–198, Porto, Portugal. IEEE.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

Cherrueau, R.-A., Delavergne, M., Van Kempen, A., Lebre, A., Pertin, D., Rojas Balderrama, J., Simonet, A., and Simonin, M. (2022). EnOSlib: A library for experiment-driven research in distributed computing. *IEEE Transactions on Parallel and Distributed Systems*, 33(6):1464–1477.

Cornebize, T. (2021). *High Performance Computing: towards better Performance Predictions and Experiments*. PhD thesis, Université Grenoble Alpes.

Degomme, A., Legrand, A., Markomanolis, G. S., Quinson, M., Stillwell, M., and Suter, F. (2017). Simulating MPI applications: The SMPI approach. *IEEE Transactions on Parallel and Distributed Systems*, 28(8):2387–2400.

Denis, A. and Trahay, F. (2016). MPI overlap: Benchmark and analysis. In *2016 45th International Conference on Parallel Processing (ICPP)*, pages 258–267, Philadelphia, PA, USA. IEEE.

Energy Sciences Network (ESnet) (2026). iPerf3: A TCP, UDP, and SCTP network bandwidth measurement tool. `https://github.com/esnet/iperf`. Version 3.21.

Fletcher, R. P., Du, X., and Fowler, P. J. (2009). Reverse time migration in tilted transversely isotropic (TTI) media. *Geophysics*, 74(6):WCA179–WCA187.

Hoefler, T. and Belli, R. (2015). Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15)*, New York, NY, USA. Association for Computing Machinery.

Keller Tesser, R., Mello Schnorr, L., Legrand, A., Dupros, F., and Navaux, P. O. A. (2017). Using simulation to evaluate and tune the performance of dynamic load balancing of an over-decomposed geophysics application. In Rivera, F. F., Pena, T. F., and Cabaleiro, J. C., editors, *Euro-Par 2017: Parallel Processing*, volume 10417 of *Lecture Notes in Computer Science*, pages 192–205, Cham. Springer.

Künas, C. A., Freytag, G., Araújo, T., Machado, R., Morales, B., Sardinha, A., and Navaux, P. O. A. (2025). Compressão de dados para redução de E/S em simulações sísmicas de alto desempenho. In *Anais do XXVI Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 326–337, Bonito, Brazil. Sociedade Brasileira de Computação. (in Portuguese).

Künas, C. A., Freytag, G., and Navaux, P. O. A. (2024). Enhancing reverse time migration simulations in HPC systems through I/O and computation overlapping. In Guerrero, G., San Martín, J., Meneses, E., Barrios Hernández, C. J., Osthoff, C., and Monsalve Diaz, J. M., editors, *High Performance Computing: 11th Latin American High Performance Computing Conference, CARLA 2024, Santiago de Chile, Chile, Revised Selected Papers*, volume 2270 of *Communications in Computer and Information Science*, pages 68–82. Springer, Cham.

Medvedev, A. V. (2022). IMB-ASYNC: a revised method and benchmark to estimate MPI-3 asynchronous progress efficiency. *Cluster Computing*, 25(4):2683–2697.

Micikevicius, P. (2009). 3D finite difference computation on GPUs using CUDA. In *Proceedings of the 2nd Workshop on General Purpose Processing on Graphics Processing Units (GPGPU-2)*, pages 79–84, NY, USA. Association for Computing Machinery.

Pearson, C., Hidayetoğlu, M., Almasri, M., Anjum, O., Chung, I.-H., Xiong, J., and Hwu, W.-M. W. (2020). Node-aware stencil communication for heterogeneous supercomputers. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 796–805. IEEE.

Potluri, S., Hamidouche, K., Venkatesh, A., Bureddy, D., and Panda, D. K. (2013). Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with NVIDIA GPUs. In *2013 42nd International Conference on Parallel Processing (ICPP)*, pages 80–89. IEEE.

Rigon, P. H. C., Schussler, B. S., Künas, C. A., Lorenzon, A. F., Carissimi, A., and Navaux, P. O. A. (2024). Otimizando a implementação multi-GPU do método fletcher através da paralelização eficiente na computação e comunicação de dados. In *Anais da XXIV Escola Regional de Alto Desempenho da Região Sul (ERAD-RS)*, pages 5–8, Florianópolis, Brazil. Sociedade Brasileira de Computação. (in Portuguese).

Salim, J. H. (2015). Linux traffic control classifier-action subsystem architecture. In *Proceedings of netdev 0.1*, Ottawa, ON, Canada.

Spadotto, V. D. (2026). Distributed implementation of anisotropic RTM using the fletcher method with MPI and ghost cells. B.s. thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil.

Spadotto, V. D. and Mello Schnorr, L. (2026a). MPI-based 3D anisotropic RTM with fletcher's method and ghost cell halo exchange. In *2026 IEEE 38th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, Madrid, Spain. To appear.

Spadotto, V. D. and Mello Schnorr, L. (2026b). SimGrid to evaluate ghost cells effectiveness in high-performance computing applications. In *Latin America High Performance Computing Conference (CARLA)*, Córdoba, Argentina. To appear.

Stanisic, L., Thibault, S., Legrand, A., Videau, B., and Méhaut, J.-F. (2015). Faithful performance prediction of a dynamic task-based runtime system for heterogeneous multi-core architectures. *Concurrency and Computation: Practice and Experience*, 27(16):4075–4090.

Unat, D., Turimbetov, I., Issa, M. K. T., Sağbili, D., Vella, F., De Sensi, D., and Ismayilov, I. (2026). The landscape of GPU-centric communication. *ACM Computing Surveys*, 58(12):1–36.

Velho, P., Schnorr, L. M., Casanova, H., and Legrand, A. (2013). On the validity of flow-level TCP network models for grid and cloud simulations. *ACM Transactions on Modeling and Computer Simulation*, 23(4):23:1–23:26.

Zhou, H.-W., Hu, H., Zou, Z., Wo, Y., and Youn, O. (2018). Reverse time migration: A prospect of seismic imaging methodology. *Earth-Science Reviews*, 179:207–227.