jh250079

Study on Efficient File IO for GPU

Toshihiro Hanawa (The University of Tokyo)

Abstract

File I/O between GPU memory and storage is an increasingly critical bottleneck for large-scale HPC and AI workloads where data routinely exceeds GPU and host memory capacity. This project investigated efficient GPU file I/O across heterogeneous JHPCN systems along three themes: (1) characterising I/O behaviour of representative codes (GOTHIC, SCALE-RM); (2) benchmarking CPU-mediated I/O against GPUDirect Storage (GDS) on the NVIDIA GH200 Grace–Hopper Superchip in Miyabi-G, including a study of CPU/GPU first-touch effects on the NVLink-C2C link; and (3) prototyping an $\mathcal{H}$-Matrix application using GPU-initiated direct I/O via Big accelerator Memory (BaM) toward NVIDIA's SCADA architecture. On Miyabi-G, the NVLink-C2C fabric narrows the GDS/CPU bandwidth gap vs. x86+PCIe: GDS-NATIVE, GDS-COMPAT and SEC2 reach 9–13 GB/s on writes while POSIX `O_DIRECT` saturates at 2–3 GB/s; read bandwidth is largely insensitive to first-touch policy. For the BaM-based $\mathcal{H}$-Matrix application, batch pinning in the GPU page cache yielded 1.31× end-to-end and 1.71× read-time speed-ups over the CPU path.

1 Basic Information

1.1 Collaborating JHPCN centers

- Hokkaido University
- The University of Tokyo
- Institute of Science Tokyo
- Kyushu University

1.2 Theme Area

- Large-scale computational science area

1.3 Project Members and Their Roles

- T. Hanawa (U. Tokyo): Project administration, overall coordination, GPU I/O technologies
- Y. Miki (U. Tokyo): Computational science applications (GOTHIC astrophysics code)
- K. Nakajima, K. Yamazaki (U. Tokyo): Computational science applications
- S. Sumimoto (U. Tokyo): Storage technologies, parallel file systems
- S. Takeshima, S. Li (U. Tokyo): GPU I/O technologies, BaM/SCADA prototyping, $\mathcal{H}$-Matrix application design
- A. Kumar Paul, M. Dongare (BITS Pilani): File-system technologies, GH200 GDS benchmarking framework
- A. Raj Basani, A. Sawane (BITS Pilani): Data storage technologies, micro-benchmark development

2 Purpose and Significance of the Research

GPUs are the dominant compute engine in present-day HPC systems: in the most recent Top500 list more than 40 % of systems use GPUs as accelerators, and inside JHPCN every centre now operates at least one GPU-based system. The most recent example is the Miyabi system at the Joint Center for Advanced HPC (JCAHPC), which adopted the NVIDIA GH200 Grace–Hopper Superchip as the GPU-accelerated node and is the first major Japanese academic system built on this architecture.

In practical applications, data that GPUs operate on must ultimately be brought in from, and written back to, persistent storage. Because GPU memory is physically separate from host memory, the conventional I/O path requires an intermediate copy through CPU memory, which causes both raw transfer overhead and contention on the host memory bus. The aim of this project is therefore (i) to shorten transfer time itself by using GPU-direct file I/O paths such as GPUDirect Storage (GDS), and (ii) to design application structures that overlap computation with file I/O. We further extend the scope to GPU-initiated I/O, in which GPU threads themselves issue NVMe commands to local SSDs (Big accelerator Memory, BaM) and, prospectively, to local and remote SSDs (Scaled Accelerated Data Access, SCADA).

The GH200 deployed in Miyabi-G introduces a particularly interesting case. Its cache-coherent NVLink-C2C link between the Grace CPU and Hopper GPU means that GDS is technically available, but the actual data path and bandwidth profile differ substantially from those on x86+PCIe hosts. Characterising and exploiting this difference is one of the principal scientific goals of this fiscal year.

3 Significance as JHPCN Joint Research Project

This project requires three kinds of resources that JHPCN is uniquely able to provide together: (i) production-grade GPU systems with diverse host CPU architectures (x86, Arm, Grace); (ii) parallel file systems (Lustre, DAOS) and high-end NVMe SSDs co-located with those GPUs; and (iii) computational science end users in astrophysics, climate and materials science who can exercise these I/O paths under realistic scientific workloads.

Previous studies, including our own work in the predecessor projects jh220046, jh230027, and jh240081, have already shown that GDS performance differs significantly between local SSD and Lustre, and between different transfer sizes and thread counts. They have, however, mostly been carried out on x86 hosts. The GH200 platform now in operation in Miyabi-G has an Arm CPU and a cache-coherent CPU–GPU link, so its behaviour is qualitatively different and cannot be predicted from x86 measurements alone. By systematically running the same I/O paths across the GPU systems operated by each JHPCN centre, this project produces

a reference that the JHPCN centres themselves can use when planning the next generation of GPU clusters and storage systems.

In parallel, by collaborating with computational science groups (Miki on GOTHIC, the climate/weather group on SCALE-RM) and BITS Pilani on the file-system layer, we turn raw I/O characterisation into general-purpose libraries and benchmarking tools that can be deployed on every JHPCN GPU system.

4 Outline of Research Achievements until FY2024

This project is a continuous project, building on jh220046, jh230027 and jh240081. In those three years we built GDS-capable environments on the ITC experimental cluster, on virtual machines within mdx, and on Pegasus at the University of Tsukuba, and performed initial GDS evaluations there.

We also began the application of GDS to three classes of GPU applications: the gravitational octree code GOTHIC, the urban-scale meteorological code City-LES, and a representative machine-learning training workload. The principal academic output of this earlier phase was M. Tominaga's thesis work, which proposed a hint-based selection framework between GDS and CPU-mediated I/O for HDF5 on GPUs and was recognised as Best Master's Student paper at xSIG 2024 [6], with a companion poster at GTC 2024 [7].

The Tominaga framework had two structural limitations: it depended on user-supplied hints and used hard-coded transfer-size thresholds fitted on x86 hardware. Generalising it to the GH200 (Arm host with cache-coherent C2C) and to file formats beyond HDF5 was the natural next step.

5 Details of FY2025 Research Achievements

The work in FY2025 was structured along the three themes identified in the proposal. We summarise the achievements of each theme in turn.

5.1 Theme 1: Computational Science Simulations

5.1.1 GOTHIC (Astrophysics)

GOTHIC is a gravitational octree code optimised for GPUs in which essentially all compute kernels — tree construction, tree traversal, and time integration — run on the GPU. File I/O for initial conditions and for snapshot/restart files, however, still goes through HDF5 on the host CPU. On Miyabi-G we have completed an initial port of the I/O paths of GOTHIC to the GH200 environment and are analysing where, in a long N-body run, the HDF5 traffic becomes a wall-clock bottleneck. The GH200 results indicate that the conventional HDF5-via-host path suffers more from CPU-side overhead than was the case on Pegasus, which makes GOTHIC a particularly good test case for the GDS replacement path described in Theme 2.

5.1.2 SCALE-RM (Climate/Weather)

We replaced the City-LES target used in earlier years with SCALE-RM, the regional climate/weather code being used to simulate tropical cyclones and other large-scale phenomena. Such simulations need very fre-

quent output of high-precision history files in order to capture detailed time evolution, which makes them I/O-bound on current GPU systems. We are evaluating two strategies to reduce the I/O overhead: (a) batched asynchronous output that overlaps with the next time step, and (b) replacing the existing NetCDF-on-host path with a GDS-aware writer. Both directions are planned to feed into the framework described in Theme 2.

5.2 Theme 2: System Software for GPU File I/O

5.2.1 Initial benchmarking on Miyabi-G

We began with HDF5, the format used by GOTHIC and by many other JHPCN applications, and benchmarked CPU-mediated I/O against GDS on the Miyabi-G cluster (NVIDIA GH200, CUDA driver 550), using the `ymiki/h5-gds` repository as a starting point. We swept block size and CPU buffer size and compared the resulting throughput against numbers reported on x86+PCIe systems in the literature.

The trends did not match. In particular, the buffer-size dependency and the absolute throughput of the CPU-mediated HDF5 path on GH200 differed from earlier x86 measurements, which strongly suggested architecture-specific factors — most plausibly the cache-coherent C2C link and the way pages are placed when first touched on a unified-memory system. We therefore developed our own micro-benchmark on top of H5Bench, in which the I/O parameters that we wanted to study (buffer size, transfer size, thread count, first-touch policy, file format, file system) are first-class controls. This benchmark is the basis for the GH200 first-touch study reported below and has already been used to cross-validate the results with NetCDF (§5.2.4).

5.2.2 First-touch effects on GH200

On the GH200, the same physical page can in principle be mapped to both the Grace CPU and the Hopper GPU through the NVLink-C2C cache-coherent interconnect. When a page is first written, however, it is materialised in the memory that is local to the requesting agent, and subsequent accesses from the other agent then have to traverse C2C. This "first-touch" policy has direct consequences for file I/O, where the question is whether the I/O path materialises the buffer on the Grace side (CPU first-touch) or on the Hopper side (GPU first-touch).

We measured read and write bandwidth on Miyabi-G as a function of problem size (number of particles in a particle-data HDF5 file, ranging from 1 K to 134 M particles) for four I/O paths:

- **GDS-NATIVE** — cuFile with native GDS enabled.
- **GDS-COMPAT** — cuFile in compatibility mode (CPU staging path inside the cuFile library).
- **SEC2** — the cuFile SEC2 transfer mode.
- **DIRECT** — POSIX `O_DIRECT` read/write to a CPU buffer, baseline.

Figures 1 and 2 show the bandwidth when the buffer is first-touched by the *CPU*; Figs. 3 and 4 show the same measurements

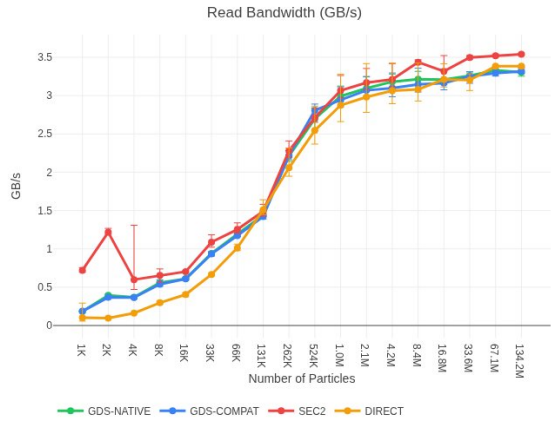

Fig. 1 Read bandwidth on Miyabi-G (GH200) when the buffer is first-touched by the CPU. All four I/O paths — including POSIX DIRECT — converge to ∼3.0–3.5 GB/s for problems above 1 M particles, with SEC2 leading by a small margin.

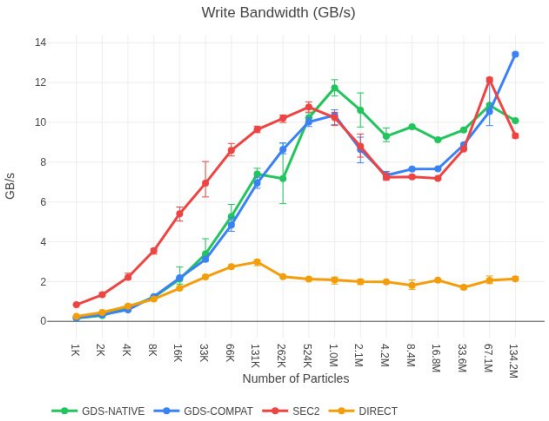

Fig. 2 Write bandwidth on Miyabi-G (GH200) under CPU first-touch. SEC2 peaks near ∼11 GB/s at intermediate sizes; GDS-COMPAT reaches the highest sustained bandwidth (∼13.5 GB/s) at 134.2 M particles, while GDS-NATIVE peaks near 12 GB/s before declining slightly. The POSIX DIRECT path is bandwidth-limited at ∼2–3 GB/s.

when the buffer is first-touched by the *GPU*.

The data show four robust effects on the GH200 platform:

1. **The DIRECT path is the bottleneck for writes, not for reads.** On the write path, POSIX `O_DIRECT` saturates at roughly 2–3 GB/s regardless of problem size or first-touch policy, while the three GPU-aware paths (GDS-NATIVE, GDS-COMPAT, SEC2) reach 9–13 GB/s. On the read path, by contrast, DIRECT is competitive with GDS, converging at ∼2.7–3.0 GB/s for large datasets — only a few percent below SEC2.
2. **Read I/O is largely insensitive to first-touch policy.** Comparing Figs. 1 and 3, the read curves are essentially indistinguishable. Whatever address-translation overhead the GH200 incurs when the GPU accesses Grace-resident pages, it does not noticeably affect read bandwidth at the file-I/O scale.
3. **Write peak depends on first-touch policy.** Under CPU first-touch (Fig. 2), GDS-COMPAT reaches the highest sustained bandwidth (∼13.5 GB/s) at 134.2 M particles. Under GPU first-touch (Fig. 4), the same paths peak at ∼10 GB/s near 524 K–1 M particles and then converge with the others at 6–9 GB/s for larger sizes. Compute latencies are slightly higher under CPU first-touch (because of the address-translation layer when the GPU accesses pages resident in system memory), but the I/O bandwidth trends remain in the same order of magnitude.
4. **SEC2 is consistently competitive with native GDS.** On both first-touch

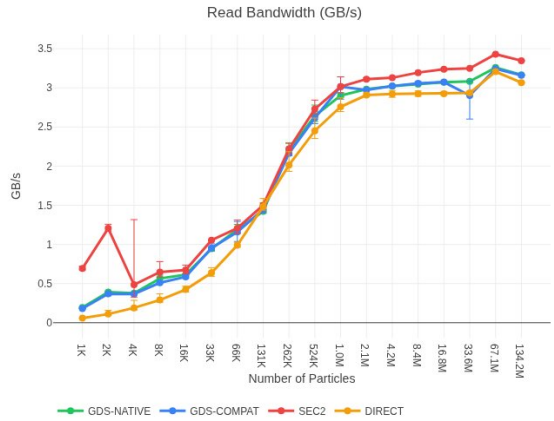

Fig. 3 Read bandwidth on Miyabi-G (GH200) when the buffer is first-touched by the GPU. The behaviour is remarkably similar to CPU first-touch (Fig. 1), indicating that read I/O is largely insensitive to initial page placement on the GH200; SEC2 again leads at ∼3.4 GB/s.

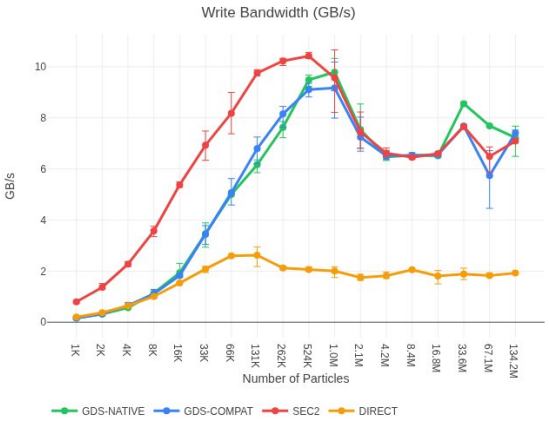

Fig. 4 Write bandwidth on Miyabi-G (GH200) under GPU first-touch. SEC2 peaks at ∼10.5 GB/s and GDS-NATIVE/GDS-COMPAT at ∼9–9.5 GB/s near 524 K–1 M particles; above 2.1 M particles all three GPU-aware paths converge in the 6–9 GB/s range. The DIRECT path remains flat at ∼2 GB/s.

policies, SEC2 leads on read for large problems and is the highest peak path on write at intermediate sizes. Because SEC2 does not require the same kernel-side configuration as native GDS, it is an attractive portability target across heterogeneous JHPCN GPU systems.

The read paths (Figs. 1, 3) converge for large problem sizes; the noise at ≤ 16 K particles reflects per-call latency, not sustained bandwidth.

5.2.3 Why the GDS/CPU gap is narrower on GH200

The most striking high-level observation from the four plots is that, on GH200, the gap between GDS and the non-GDS paths is much narrower than is typically reported on x86+PCIe systems. We attribute this to the NVLink-C2C cache-coherent fabric.

On a discrete x86+GPU node, GPUDirect Storage enables peer-to-peer DMA between the NVMe device and GPU memory over the PCIe bus, bypassing a CPU bounce buffer. The saving comes both from avoiding a second copy and from avoiding the host memory-bus contention it produces; the speed-up over a CPU-staged path is therefore large.

On GH200, the GPU is not a PCIe endpoint. The Hopper GPU is connected to the Grace CPU through NVLink-C2C, a high-bandwidth, cache-coherent link, and the NVMe SSDs attach to the Grace PCIe complex. When GDS is used, NVMe DMA still targets the application's registered buffer without a CPU bounce-buffer copy — so the core GDS advantage is preserved — but the data path itself is different. Depending on where the buffer is allocated and first touched, data may be DMA'd into Grace

LPDDR5X and accessed by the GPU over coherent C2C, or staged into Hopper HBM. In either case the transfer is DMA-driven and avoids host CPU involvement.

The crucial point is that, because NVLink-C2C is itself fast and cache-coherent, traversing system memory is no longer a PCIe bottleneck. The coherent fabric effectively levels the playing field between the DMA paths — which is why GDS-NATIVE, GDS-COMPAT and SEC2 all sit within a relatively narrow band on every plot. The DIRECT write path remains slow because it goes through the standard kernel write path with a user-buffer copy, not because of the C2C link.

5.2.4 Cross-validation with NetCDF

To confirm that these findings are platform-inherent and not artefacts of HDF5, we repeated the bandwidth sweep with NetCDF as the I/O layer. The same qualitative ordering of paths (SEC2 $\geq$ GDS-NATIVE $\approx$ GDS-COMPAT $\gg$ DIRECT for writes; all four paths within a few percent for reads at large sizes) and the same narrowness of the GDS/CPU gap were reproduced. This confirms that the bandwidth profile reported above is an intrinsic property of the GH200 platform — of the NVLink-C2C fabric and the cuFile transfer modes that run on it — and not of the choice of file format.

5.2.5 Implications for the framework

These results directly inform the framework that we are building on top of H5Bench. The decision rule between GDS and CPU-mediated I/O on GH200 cannot be the fixed transfer-size threshold derived in the Tominaga work [6]: on GH200 specifically, the right rule is closer to "always use a GPU-aware path on writes; reads are insensitive". First-touch state, file format and host architecture are therefore being exposed as runtime inputs to the framework. Methodology refinement is also ongoing: finer-grained profiling of individual DMA stages, more controlled memory-placement policies, and a broader sweep of cuFile configurations are planned for the remainder of the project.

5.3 Theme 3: Advanced GPU I/O — BaM and SCADA

NVIDIA and UIUC have proposed Big accelerator Memory (BaM) [1], in which GPU threads themselves issue NVMe commands directly to local SSDs, and GPU-Initiated Direct Storage (GIDS) [2], which extends the same idea to GNN workloads. Most recently, NVIDIA has disclosed Scaled Accelerated Data Access (SCADA) [3], which generalises BaM to tens of local and remote NVMe SSDs per GPU and targets PB-class data sets at sub-4 KB granularity. SCADA is not yet publicly available, so in FY2025 we used BaM as the prototype.

5.3.1 $\mathcal{H}$-Matrix as a target application

We chose the hierarchical matrix ($\mathcal{H}$-Matrix) boundary-element-method (BEM) solver as the HPC target application for BaM. Dense matrices arising from BEM scale as $\mathcal{O}(N^2)$, but $\mathcal{H}$-Matrix compression based on Adaptive Cross Approximation (ACA) reduces this to $\mathcal{O}(N \log N)$ in well-behaved cases. For problems large enough to be of practical interest the compressed matrix still does not fit in GPU memory, so the application has to stream the matrix from SSD on every it-

Table 1 End-to-end execution time on the Stanford Dragon $\mathcal{H}$-Matrix (42.16 GB, 30 BiCGSTAB iterations, 8 GiB GPU cache for BaM).

I/O path	Total [s]	Write [s]	Read [s]	Other [s]
BaM	1148.7	24.5	649.0	475.2
CPU path	1311.5	14.9	829.4	467.2
CPU path (`O_DIRECT`)	1301.4	14.7	823.3	463.4

eration of an iterative solver — exactly the regime where BaM should excel. The construction kernel is based on the HACApK library [4, 5], ported from Fortran to C++ for this work, and the solver uses BiCGSTAB.

5.3.2 Application design

The compressed matrix is split into batches sized so that each batch fits in a configurable fraction of GPU memory. For each batch, the dense leaves and the U, V low-rank factors are written to NVMe SSD; on the BaM path, this is done by GPU threads writing directly at offsets in the raw block device, while on the CPU path it is done through buffered `write()` or `O_DIRECT`. At solve time, each BiCGSTAB iteration sweeps every batch: the batch is read from SSD, the matrix-vector product contribution is added to the residual, and the next batch is read.

5.3.3 Evaluation

Evaluation was carried out on a node with one AMD EPYC 7713, an NVIDIA A100 (80 GB) and a KIOXIA CM6-V NVMe SSD (used as a raw block device for BaM, and through a filesystem for the CPU paths). The input model was the Stanford Dragon, configured to produce a 42.16 GB $\mathcal{H}$-Matrix (10 batches, 30 BiCGSTAB iterations for the I/O measurement).

Without batch pinning (Table 1), BaM was ∼1.28× faster on reads than the CPU path with the OS page cache enabled, and ∼1.27× faster than the CPU `O_DIRECT` path. Profiling with `iostat` and `perf` showed that, with the OS cache enabled, the CPU read path spent 64.6 % of CPU time in `copy_user_enhance_fast_string` and 13.6 % in `clear_page_erms`: in other words, the cost was dominated not by SSD bandwidth but by host page-cache management and the kernel/user copy. BaM avoids both costs by reading directly into GPU memory.

Table 2 Effect of pinning the first k batches in the BaM GPU page cache (32 GiB cache, 153 batches).

Configuration	Total [s]	Write [s]	Read [s]	Other [s]
BaM (none)	1225.6	25.7	680.8	519.1
BaM (20 pinned)	1182.9	25.6	658.7	498.6
BaM (50 pinned)	1143.7	25.2	593.8	524.7
BaM (80 pinned)	1081.6	25.1	542.7	513.8
BaM (110 pinned)	1071.9	25.8	499.9	546.2
BaM (120 pinned)	1026.2	25.2	480.4	520.6
CPU path	1352.6	18.6	820.6	513.4

The most striking result was obtained when the BaM cache size was raised to 32 GiB and the H-Matrix was split into 153 finer batches. With BiCGSTAB, the reuse distance for any one batch is essentially the entire matrix, and BaM's default round-robin replacement therefore behaves close to FIFO — so for a working set larger than the cache, hit rate collapsed to zero in our first measurements. We worked around this by pinning the first k batches in the GPU page cache (Table 2), exploiting the per-page reference counter that BaM maintains for inter-thread synchronisation. With 120 pinned batches the end-to-end time dropped to 1026 s —

1.31× faster than the CPU path, with read time alone 1.71× faster.

This is the first end-to-end demonstration of BaM speeding up an HPC numerical-linear-algebra application beyond the ML and graph workloads originally targeted, and provides a concrete case study for what SCADA will need to support when it becomes available. The pinning result also identifies a clear research direction (Section 6) towards solver redesign for GPU-initiated I/O.

5.4 Cross-cutting: hardware partner discussions

In FY2025 we also held regular technical discussions with KIOXIA on their Storage-Class Memory (XL-FLASH) and Super-High-IOPS SSD roadmap, which will sample in 2026 and aims at $> 10\,\mathrm{M}$ IOPS per device, and with NVIDIA on SCADA. The HPC use cases and access patterns identified in this project — in particular, the BiCGSTAB-style sweeping access in the $\mathcal{H}$-Matrix work and the HDF5 first-touch behaviour on GH200 — have been fed back to both vendors and are influencing our planning for the next-generation GPU system at the University of Tokyo.

6 Self-review of Current Progress and Future Prospects

6.1 Self-review against the FY2025 plan

The proposal identified three deliverables for FY2025: (a) extend the Tominaga GDS-vs-CPU framework beyond HDF5 and beyond x86; (b) bring the GH200 platform into the evaluation; and (c) prototype an HPC application on top of BaM as a stepping stone to SCADA.

Deliverable (a): The H5Bench-based benchmark is complete for HDF5 and NetCDF; binary back-ends are in development. The framework now exposes first-touch state as a runtime input, replacing fixed x86 thresholds. Substantially complete.

Deliverable (b): GH200 evaluation on Miyabi-G produced quantitative bandwidth profiles (Figs. 1–4) and an NVLink-C2C architectural explanation, cross-validated with NetCDF and shared with JHPCN centres planning Grace-Hopper systems. On plan.

Deliverable (c): The BaM $\mathcal{H}$-Matrix application is end-to-end functional; 120-batch pinning yields 1.31× end-to-end and 1.71× read-time speed-ups over the CPU path, already publishable [8]. On plan.

6.2 Future prospects

Three lines of work follow naturally from this year's results.

■Framework generalisation. HDF5 and NetCDF sweeps are complete; h5py and binary back-ends are next. We will run the same parameter sweep across all GDS-capable JHPCN GPU systems and publish the resulting decision rules as a JHPCN-internal reference.

■Solver redesign for GPU-initiated I/O. BiCGSTAB's near-FIFO cache behaviour under sweeping access is a structural mismatch for a finite-size GPU page cache. We plan block/restarted and s-step Krylov variants with finer-grained batching to overlap I/O and compute, targeting the high random-IOPS regime of SCADA and KIOXIA Super-High-IOPS SSDs.

■SCADA integration. The BaM $\mathcal{H}$-Matrix application will be ported to SCADA upon availability, extending evaluation to multiple local and remote SSDs beyond the single-PCIe-attachment regime of BaM.

■Application targets. On the application side, GOTHIC will be the first code into which the framework's GH200-aware decision rule is inserted, with SCALE-RM following once its history-file output has been refactored. We expect to report end-to-end speed-ups for both codes in FY2026.

References

[1] Z. Qureshi, V. S. Mailthody, I. Gelado, S. Min, A. Masood, J. Park, J. Xiong, C. J. Newburn, D. Vainbrand, I.-H. Chung et al., "GPU-Initiated On-Demand High-Throughput Storage Access in the BaM System Architecture," *Proc. ASPLOS 2023*, pp. 325–339, Mar. 2023.

[2] J. B. Park et al., "Accelerating Sampling and Aggregation Operations in GNN Frameworks with GPU-Initiated Direct Storage Accesses," *Proc. VLDB Endowment*, May 2024.

[3] C. J. Newburn, P. Prabhu, V. S. Mailthody, "Speed-of-Light Data Movement Between Storage and the GPU," NVIDIA GTC 2025, session S73012.

[4] A. Ida, T. Iwashita, T. Mifune and Y. Takahashi, "Parallel hierarchical matrices with adaptive cross approximation on symmetric multiprocessing clusters," *Journal of Information Processing*, vol. 22(4), pp. 642–650, 2014.

[5] A. Ida and T. Iwashita, HACApK code, `https://github.com/Post-PetaCrest/ppOpenHPC/tree/MATH/HACApK`.

[6] M. Tominaga, T. Hanawa, Y. Miki, "GPU 直接 IO を用いたファイル IO の高速化," *xSIG 2024*, Aug. 2024 (Best Master's Student Award).

[7] M. Tominaga, T. Hanawa, Y. Miki, "Toward Optimizing File IO on GPU Clusters," NVIDIA GTC 2024 poster, Mar. 2024.

[8] S. Takeshima, T. Hanawa, Y. Miki, A. Ida, "大規模データを指向した GPU 直接 IO に基づく $\mathcal{H}$-Matrix アプリケーションの設計と性能評価," IPSJ SIG Technical Report, Vol. 2026-HPC-203, No. 64, Mar. 2026.

[9] NVIDIA, "NVIDIA GPUDirect Storage," `https://docs.nvidia.com/gpudirectstorage/index.html`.