LLNL-CFPRES-2021260

Lawrence Livermore National Laboratory

GLANCED-IO: Taming IO Optimization for Deep Learning at Scale

H Zheng, H Gunawi, H Hariharan Devarajan, R Ross, W Nixon, R Sinurat, T Arcomano, P Carns, S Madireddy, S Foreman

Jul 2026

The 35th ACM International Symposium on High-Performance Parallel and Distributed Computing

Cleveland, OH, United States of America

2026-07-13 through 2026-07-16

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

GLANCED-IO

Taming I/O Optimization for Deep Learning at Scale

Ray A. O. Sinurat[1], William Nixon[1], Philip Carns[2], Huihuo Zheng[2], Sandeep Madireddy[2], Sam Foreman[2], Troy Arcomano[3], Robert Ross[2], Haryadi S. Gunawi[1], Hariharan Devarajan[4]

[1]University of Chicago [2]Argonne National Laboratory [3]Allen Institute for AI [4]Lawrence Livermore National Laboratory









HPDC '26 · Cleveland, OH · July 13–16, 2026

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-CFPRES-2021260

Scientific DL at scale is drowning in I/O

Weather, protein folding, genomics: **terabyte-scale datasets, read hundreds of times** over a training run → **petabyte-scale I/O**.

1000s	**TB → PB**	**Storage**
of GPUs per run	I/O over a run	strained to keep pace

"The dataset totals 16 TB in HDF5 …" , AERIS, SC'25

Storage must keep pace with computation, or data loading becomes the bottleneck.

Why I/O tuning is hard: three limitations

Three reasons, one at a time.

1 **Isolated metrics**
tune the application OR the system

Why I/O tuning is hard: three limitations

Three reasons, one at a time.

1. **Isolated metrics**
   tune the application OR the system

2. **Configuration-space explosion**
   jointly tuning both blows up the search

Why I/O tuning is hard: three limitations

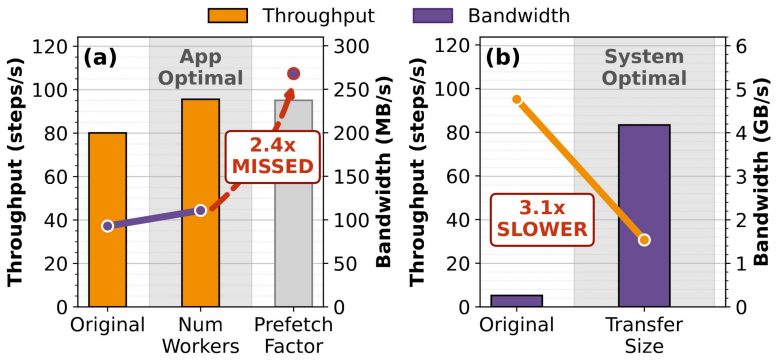
Three reasons, one at a time.

1. **Isolated metrics**
   tune the application OR the system

2. **Configuration-space explosion**
   jointly tuning both blows up the search

3. **Full-pipeline cost**
   every candidate runs the real GPU pipeline

Limitation #1: one metric hurts the other



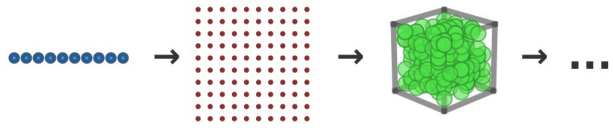
(a) App-only

Tuning for throughput stops early, misses **2.4×** of system bandwidth.

(b) System-only

Maximizing bandwidth makes the app **3.1× slower**.

Limitation #2: the configuration space explodes



1 param
10 configs

2 params
100 configs

3 params
1000 configs

Joint tuning is **O(N^K)**, K knobs × N values each → exponential blow-up.

Autotuners (DeepHyper)
need evaluation budgets → may miss the best config

ML methods (AIIO)
need terabytes of data collected over years

Limitation #3: every evaluation runs the full pipeline



× hundreds of configurations

= days to months of optimization, burning GPUs the whole time

What GLANCED-IO does

Jointly optimizes **application + system I/O** and **matches or beats state-of-the-art**, at a fraction of the cost.

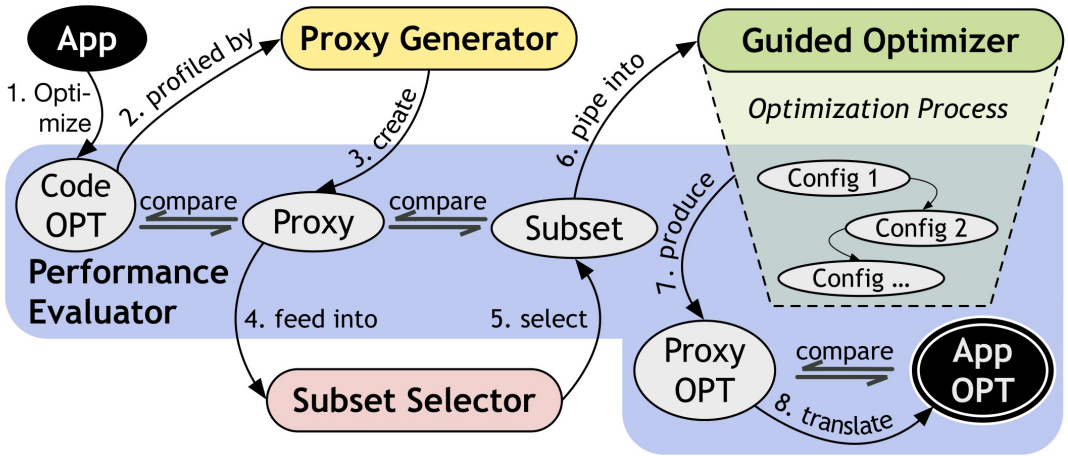
NO GPUs	NO pretraining	NO full-pipeline runs
GPU-free proxy evaluation	no offline data collection	evaluate on data subsets

Slow I/O wastes thousands of GPUs.
GLANCED-IO offers cheap, GPU-free, guided tuning.

GLANCED-IO at a “glance”

1.57× better than state-of-the-art (AIIO)	**1.8×** faster than the original pipeline	**2.3×** fewer configs than AIIO
~3.3× faster than DeepHyper	**93%** average proxy fidelity	**0** GPUs · no data collection

GLANCED-IO: the optimization pipeline



Four components: each removes one of the three limitations. **We walk through them next.**

Performance Evaluator: a joint metric

Solves #1 · Isolated metrics

Throughput ($\hat{t}$)
application speed

Performance Evaluator: a joint metric

Solves #1 · Isolated metrics

Throughput ($\hat{t}$)
application speed

+

Bandwidth ($\hat{b}$)
system utilization

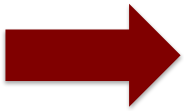
Performance Evaluator: a joint metric

Solves #1 · Isolated metrics

Throughput ($\hat{t}$)
application speed

+

Bandwidth ($\hat{b}$)
system utilization



One joint score

$$C \cdot \hat{t} + (1 - C) \cdot \hat{b}$$

C = 0.5 in this work (equal weighting)

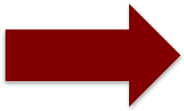
Performance Evaluator: a joint metric

Solves #1 · Isolated metrics

Throughput ($\hat{t}$)
application speed

+

Bandwidth ($\hat{b}$)
system utilization



One joint score

$$C \cdot \hat{t} + (1 - C) \cdot \hat{b}$$

C = 0.5 in this work (equal weighting)

- Both higher-is-better.
- Same metric validates fidelity at proxy → subset → app hand-offs.

Guided Optimizer: Greedy-OFAT

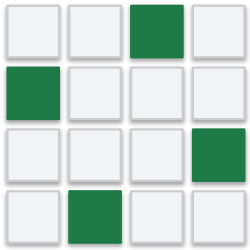
Solves #2 · Config-space explosion

More portable
Easier

Layer	Optimization parameter
App	Num Workers
	Prefetch Factor
	Dataset Access Pattern
System	PFS Striping
	Transfer Size

Less portable
Harder

Greedy-OFAT O(N^K) → **O(N·K)**



rows = knobs · columns = values · **green = best kept**

OFAT = one-factor-at-a-time:
sweep one knob, keep the best, move on

Portable knobs first *(ordering from framework docs + prior work)*: **everyone benefits.**

Proxy Generator: GPU-free proxy

Solves #3 · Full-pipeline cost

Proxy "mini-apps" are a standard HPC practice (e.g. Sandia's Mantevo); we bring it to DL I/O with DLIO.

Real application
GPUs + full pipeline
hours per evaluation

profile → strip GPU



DLIO proxy
CPU-only · no accelerator
minutes per evaluation

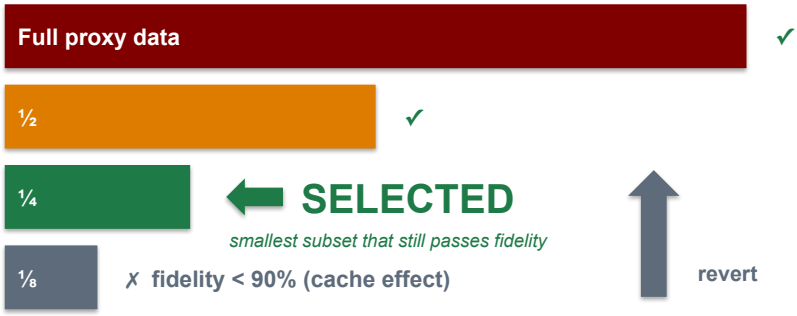
Proxy preserves:

- **I/O access pattern**
- **preprocessing time**
- **compute overlap**

Same I/O behavior, no GPU → **optimize a GPU workload without touching a GPU.**

Subset Selector: representative subset

Solves #3 · Full-pipeline cost



✓ **Halve until fidelity drops.**

Cost & storage shrink
in proportion to subset size

(e.g. a ¼ subset → ¼ the cost)

Storage is a hidden cost: every dataset variant is a full copy → **petabyte-scale**.
OFAT + subsets keep it tractable.

Evaluation setup

- **5 weather workloads on ERA5**
  SFNO, STORMER, TinySTORMER · 31 TB (0.25°) & 71 GB (5.6°), HDF5
- **Scale: 1 to 32 nodes**
  SFNO-SM 1 · STORMER-SM 2 · Tiny-LG 2 · SFNO-LG 16 · STORMER-LG 32
- **System: Tuolumne @ LLNL**
  4 AMD MI300A GPUs/node · Lustre @ 240 GB/s · 100 TB storage

Real HPC system, real weather models, real I/O pressure.

Baselines

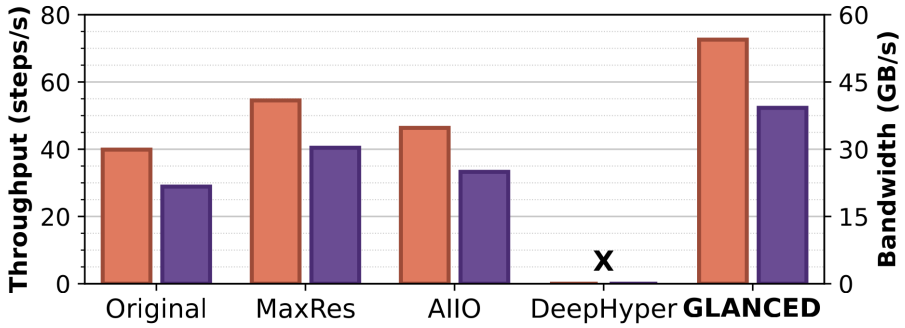
MaxResource
"use all the resources"

AIIO
ML-based, needs trained model

DeepHyper
autotuner, 12 h budget

Result: highest throughput, high bandwidth

SFNO-LG (orange = throughput, purple = bandwidth; **higher is better**)

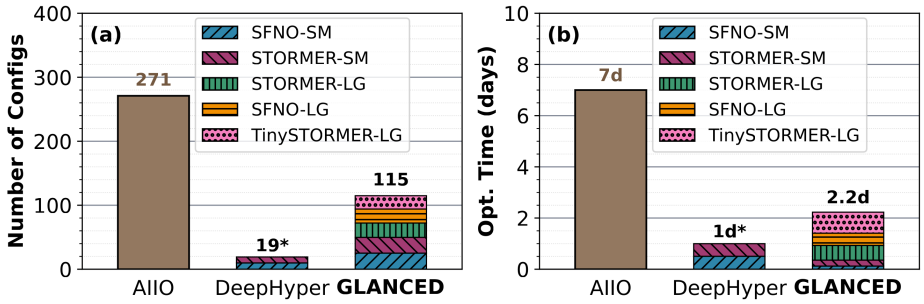


1.8×
over original pipeline

1.57×
over AIIO (SOTA)

MaxResource uses everything, and still loses.
AIIO chases bandwidth, tanks throughput; DeepHyper ✗.
No regressions on small workloads.

… for a fraction of the cost



AIIO's 7 d = upfront data collection to train its model (271 workloads, 1.2 B events), before any tuning
** DeepHyper: small-scale only, precomputing every dataset variant would exceed the 100 TB budget (petabyte-scale)*

2.3×	3.2×	93%	0
fewer configs vs AIIO	vs AIIO (2.2 d vs 7 d)	avg proxy fidelity	GPUs · data collection

Why we can trust the GPU-free proxy

Optimization runs on a proxy + subset; fidelity is validated at every hand-off.

93.7%
App → Proxy
proxy matches real I/O

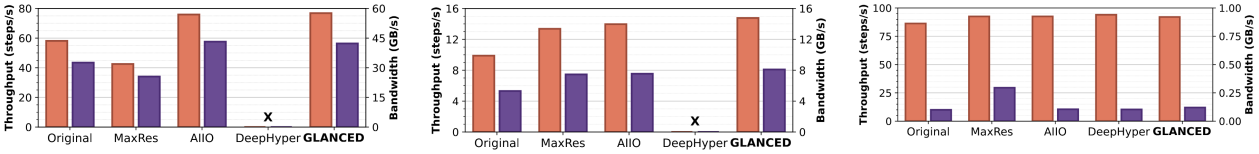
98%
Full → Subset
subset matches full proxy

86.7%
Proxy → App
gains transfer back

Ranking is preserved, enough to pick the right configuration **without a single GPU run**.
Subsets also cut each evaluation **1.6–8.5×**.

Robust across workloads: no regressions

Three I/O regimes (orange = throughput, purple = bandwidth; rightmost bar = GLANCED-IO)



I/O-bound

STORMER-LG / SFNO-LG

Big gains, balances workers + striping (up to 1.8×).

Compute-accelerated

TinySTORMER-LG

More loader parallelism; both metrics still improve.

Not I/O-bound

STORMER-SM / SFNO-SM

Already near peak, matches best, no regression.

Future work

Because each evaluation is cheap and GPU-free, we can:

Iterate the cheap loop
one OFAT sweep today; re-run it to refine (from the best config) and explore (random restarts)

Reuse modules in other tuners
drop the proxy + subset into autotuners like DeepHyper, making them GPU-free too

Broader workloads & frameworks
beyond weather, framework-agnostic by design (PyTorch / TF / JAX)

Summary

Joint metric
application + system, never trade one away

Greedy-OFAT
O(N^K) → O(N·K), no pretraining

Proxy + subset
GPU-free, 93% fidelity

1.57× over SOTA · **2.3×** fewer configs · **~3.3×** faster · **0** GPUs / data collection

Thank you! Questions?

Ray A. O. Sinurat · rayandrew@uchicago.edu

THE UNIVERSITY OF CHICAGO · Argonne NATIONAL LABORATORY · Lawrence Livermore National Laboratory · Ai2