

EuroMPI 2026: 33rd European MPI Users' Group Meeting

Adaptive compression techniques for ad-hoc parallel file systems based on MPI

Dario Muñoz-Muñoz

Felix Garcia-Carballeira

Alejandro Calderon-Mateos

Diego Camarmas-Alonso

Jesus Carretero

Vienna, October 7-9, 2026

<https://github.com/xpn-arcos/xpn>

- 1 Introduction
- 2 Related work
- 3 Expand Ad-Hoc parallel file system
- 4 Adaptive compression
- 5 Evaluation
- 6 Conclusions and Future Work

- 1** Introduction
- 2 Related work
- 3 Expand Ad-Hoc parallel file system
- 4 Adaptive compression
- 5 Evaluation
- 6 Conclusions and Future Work

Motivation

- **Current problem:** Exponential I/O growth in HPC creates bottlenecks in traditional shared file systems (Lustre, GPFS, BeeGFS)
- **Alternative:** Ad-hoc parallel file systems using local node resources as ephemeral, low-latency storage
- **Portability choice:** Some ad-hoc solutions (Expand Ad-Hoc) rely on **MPI** as communication engine
- **Critical limitation:** Co-location: application + ad-hoc layer share nodes, storage, CPU and MPI network → severe contention

Objectives and Proposal

- **Main objective:** Mitigate co-location contention by reducing data volume on network and local storage
- **Proposal:** Adaptive compression layer integrated in **Expand Ad-Hoc**
 - Intelligent decision: compress before transmission **only when beneficial**
 - Store compressed on server, keep random access
 - Controlled by `XPN_COMPRESSION`: 0 = Never, 2 = Always, 1 = Adaptive

- 1 Introduction
- 2 Related work**
- 3 Expand Ad-Hoc parallel file system
- 4 Adaptive compression
- 5 Evaluation
- 6 Conclusions and Future Work

Related work: File Systems

- **Traditional (Lustre, GPFS, BeeGFS):**
 - High capacity and persistent, but I/O contention under massive concurrent access
- **Ad-Hoc (BurstFS, GekkoFS, BeeOND, Expand Ad-Hoc):**
 - Leverage local storage of compute nodes
 - Provide high-bandwidth localized I/O, bypass central bottleneck

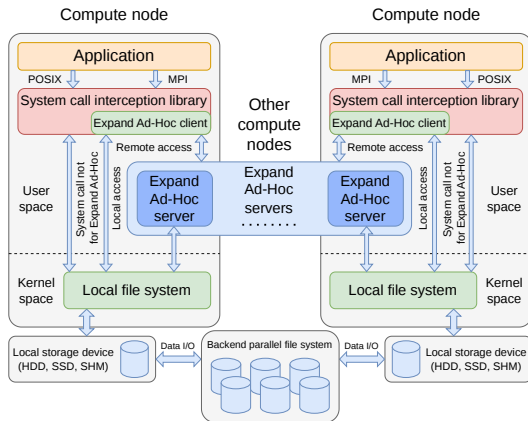
Related work: Compression

- **Algorithms: LZ4** / Snappy (speed, real-time I/O) vs Zstd / Zlib-DEFLATE (ratio, higher CPU)
- **Compression in MPI:** CoMPI (point-to-point + collectives), C-Coll (error-bounded lossy collectives), GPU-accelerated MPI_Alltoall
- **Compression in storage:** QZFS (Intel QAT hardware), SPIOC (parallel I/O compression), MRC middleware for FusionFS, Adaptive ZFS (entropy-based levels)

- 1 Introduction
- 2 Related work
- 3 Expand Ad-Hoc parallel file system**
- 4 Adaptive compression
- 5 Evaluation
- 6 Conclusions and Future Work

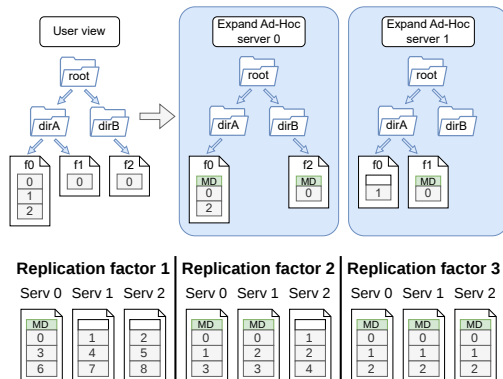
Expand Ad-Hoc Design

- Application and Expand Ad-Hoc execute on compute nodes
- Interception library (LD_PRELOAD)
- Client-server architecture (stateless) using MPI
- Local read bypass when data is on same node



Expand Ad-Hoc: Data and Replication

- On-demand subfile creation
- Distributed metadata in subfile header, hash-based master
- Block replication round-robin (fault tolerance)



- 1 Introduction
- 2 Related work
- 3 Expand Ad-Hoc parallel file system
- 4 Adaptive compression**
- 5 Evaluation
- 6 Conclusions and Future Work

Network Compression Design

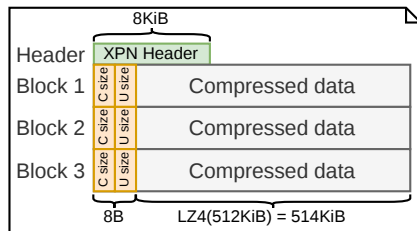
- **Write:** client sends header + payload; server writes to disk + ACK
- **Read:** client header request; server reads disk, sends response header + payload
- **Where:** compress on sender before TX, decompress on receiver
- **Algorithm: LZ4** — best compression/decompression speed, minimal CPU overhead

Adaptive Decision Model

- Telemetry per phase (compress, decompress, disk, network); network time = total – other phases (avoids buffering bias)
- Stability via EMA ($\alpha = 0,01$): $\mu_t = \mu_{t-1}(1 - \alpha) + v_t\alpha$ for avg_{comp} , avg_{decomp} , avg_{net} , avg_{disk} (MiB/s)
- Normalize network: $ratio_{cli_srv} = N_{clients}/N_{servers}$;
 $avg_{net} = mbps_avg_net / ratio_{cli_srv}$
- Decide for size S : $T_{no_comp} = S/avg_{net} + S/avg_{disk}$ vs
 $T_{comp} = S \cdot avg_{ratio}/avg_{net} + S/avg_{disk} + S/avg_{comp} + S/avg_{decomp}$; compress if
 $T_{comp} < T_{no_comp}$
- Calibration: compress 1/100 ops regardless of prediction

Disk Compression Format

- First 8 KiB: uncompressed XPN metadata
- Fixed segments: 8 B sizes (compressed + original) + payload
- 512 KiB block → 514 KiB reserved (max LZ4 output)
- Sparse-file friendly, $O(1)$ offset: $Block_{index}, Offset_{physical}$
- Full-block passthrough: no re-compress; reads offload decompression to client



Random writes via Read-Modify-Write
full writes bypass read.

- 1 Introduction
- 2 Related work
- 3 Expand Ad-Hoc parallel file system
- 4 Adaptive compression
- 5 Evaluation**
- 6 Conclusions and Future Work

Evaluation

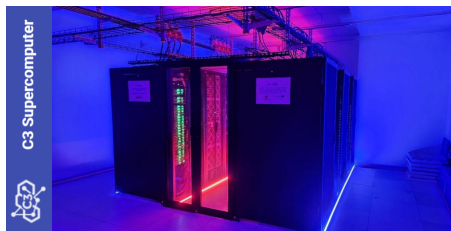
- Benchmark I/O capability + decision accuracy
 - **IOR Benchmark** (High / Medium / None patterns)
 - Policies: **Never** (baseline) vs **Always** (static) vs **Adaptive**
- Real-world application use
 - **Nek5000 turbPipe**: checkpoint 597.68 MiB, 100 steps

Evaluation: C3 supercomputer and Data

- **C3**: 90 nodes
- $2\times$ AMD EPYC 7713, 128 cores @2.8 GHz
- 3.84 TiB NVMe SSD / node
- 1024 GB RAM / node
- 100 Gbps HDR Infiniband
- MPICH 4.3.0-ofi, Rocky 8.10

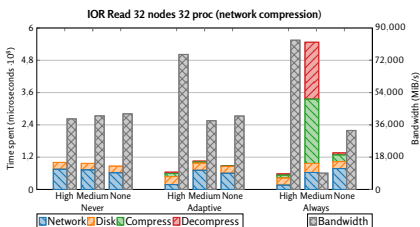
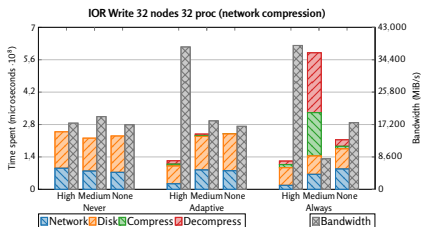
Compression ratio = uncomp/comp

Case	Ratio
IOR High	125
IOR Medium	1.984
IOR None	0.997
Nek5000	2.049



Evaluation: IOR network-only (32 nodes, low concurrency)

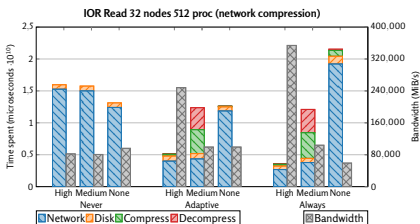
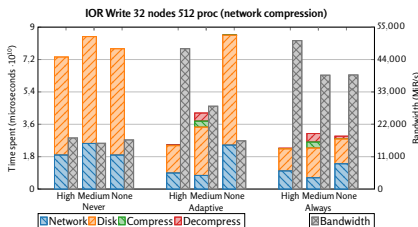
- 1 proc/node (32 clients); stacked time (lower better) + bandwidth (higher better)



- Adaptive $\approx$ Never on Medium (avoids overhead); $\approx$ Always on High; inactive on None

Evaluation: IOR network-only (32 nodes, high concurrency)

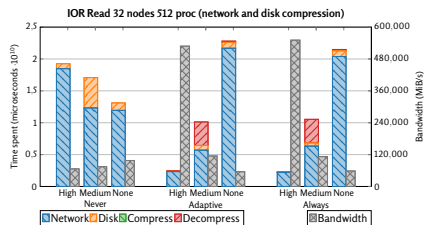
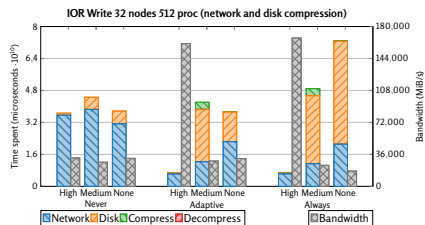
- 16 proc/node (512 clients); network congested → compression wins



- High: +190 % write, +330 % read; Medium: +148 % write, +29 % read

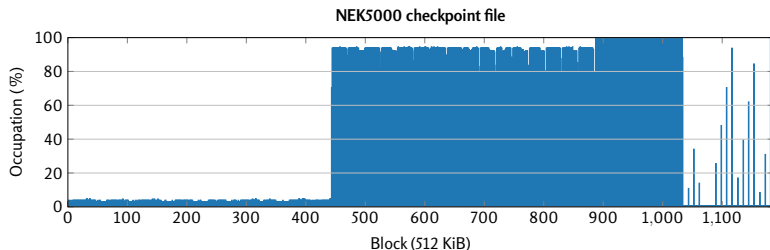
Evaluation: IOR network+disk (32 nodes)

- Same config with compressed disk format (full-block passthrough); reduces physical I/O



- High: +422 % write, +732 % read; Medium: +50 % write, +18 % read + disk-space saving

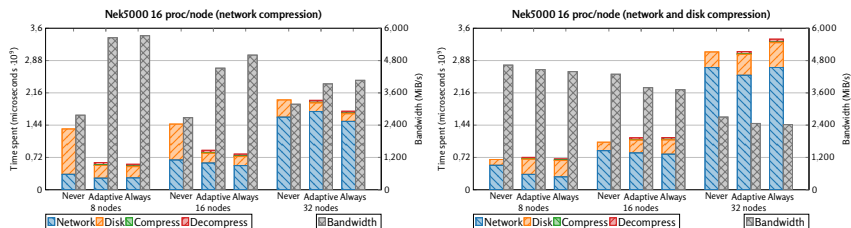
Evaluation: Nek5000 storage efficiency



- Compressed checkpoint occupation per 512 KiB block: high efficiency at start/end, medium section less compressible; total 48.79 % footprint

Evaluation: Nek5000 (8/16/32 nodes, 16 proc/node)

- 128/256/512 procs; network-only (left) vs network+disk (right)



- Adaptive $\approx$ Always; +107 % / +87 % / +28 % (63 % avg) network-only; disk -9 % avg (small unaligned writes $\rightarrow$ Read-Modify-Write) but -51.21 % disk space (291.60 of 597.68 MiB)

- 1 Introduction
- 2 Related work
- 3 Expand Ad-Hoc parallel file system
- 4 Adaptive compression
- 5 Evaluation
- 6 Conclusions and Future Work**

Conclusions and Future Work

- **Adaptive model:** Real-time telemetry (EMA) selects compression only when $T_{comp} < T_{no_comp}$
 - **IOR:** up to +732 % read / +422 % write (High, disk); +148 % write / +29 % read (Medium)
 - **Nek5000:** +63 % bandwidth avg; 51.21 % disk saving
- **Future work:** Cost of compressed disk in model, adaptive storage-layer policy, algorithms beyond LZ4 (ratio vs latency)

EuroMPI 2026: 33rd European MPI Users' Group Meeting

Adaptive compression techniques for ad-hoc parallel file systems based on MPI

Dario Muñoz-Muñoz

Felix Garcia-Carballeira

Alejandro Calderon-Mateos

Diego Camarmas-Alonso

Jesus Carretero

Vienna, October 7-9, 2026

<https://github.com/xpn-arcos/xpn>