Using Flash for Intermediate Data Management in HPC Applications

Giorgos Saloustros (gesalous@ics.forth.gr)

Institute of Computer Science (ICS),

Foundation for Research and Technology – Hellas (FORTH)

28th Jan 2026, HiPEAC 26, Krakow, Poland

EUPEX

FORTH INSTITUTE OF COMPUTER SCIENCE

EuroHPC Joint Undertaking

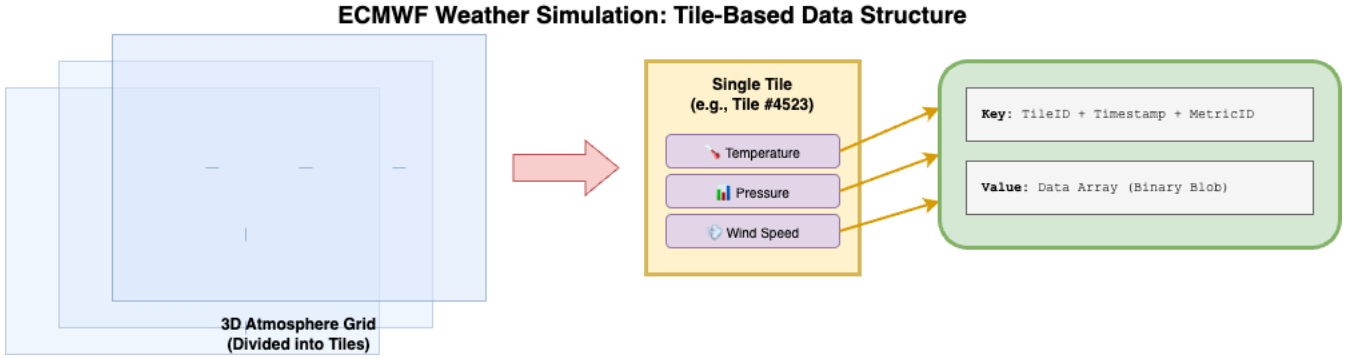
This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101033975. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Germany, Italy, Greece, United Kingdom, Czech Republic, Croatia.

The Intermediate Data Challenge in HPC

- Modern simulations (weather, physics) produce a large volume of transient data
  - Produced concurrently from thousands of compute nodes
- Need for immediate indexing for near-real-time analysis
  - Consumed by post-processing tasks before being archived or discarded
- Traditional Parallel Filesystems (e.g. Lustre) not designed for these workloads
  - No efficient inherent indexing mechanism (metadata overhead)
- The opportunity: Flash memory provides the IOPS and latency needed
  - Need the right software stack to expose it without disrupting the applications

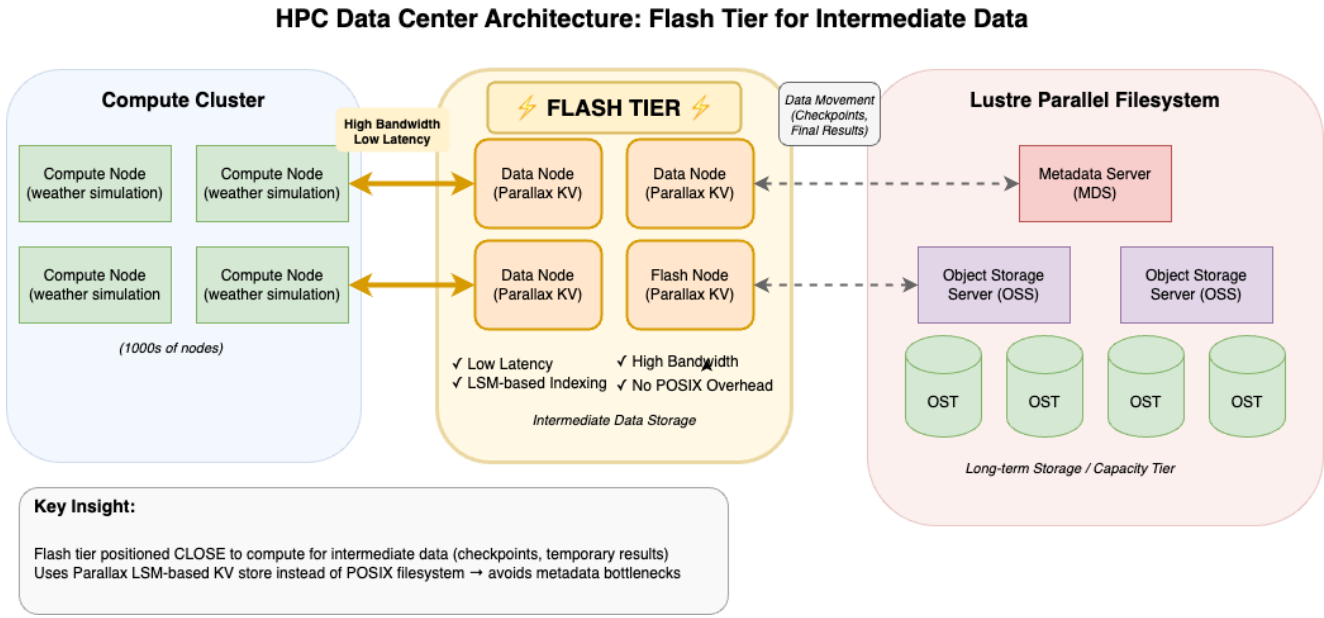
Weather Simulation Application (ECMWF)

- Runs across 100s-1000s of compute nodes
  - 4 times a day / tight time constraints for prognosis
- Atmosphere split into tiles
  - Keeps multiple metrics per tile
- Intermediate data are stored in Fields Database (FDB)
  - Implemented atop of POSIX filesystem (Lustre) (https://github.com/ecmwf/fdb)
- Both tile count and metrics per tile increase over time
  - Address challenge by using an flash tier for intermediate data
- How to use the flash tier in the weather prognosis without major disruption?



Flash Allocation and Exposure

- Data nodes equipped with flash devices
  - (NVMe SSDs)
- Allocate flash resources per job
  - Integration with Slurm
- Lifecycle management
  - Provision → Use→ Release



Ingestion Challenge: Fast Writes and on the Fly Indexing

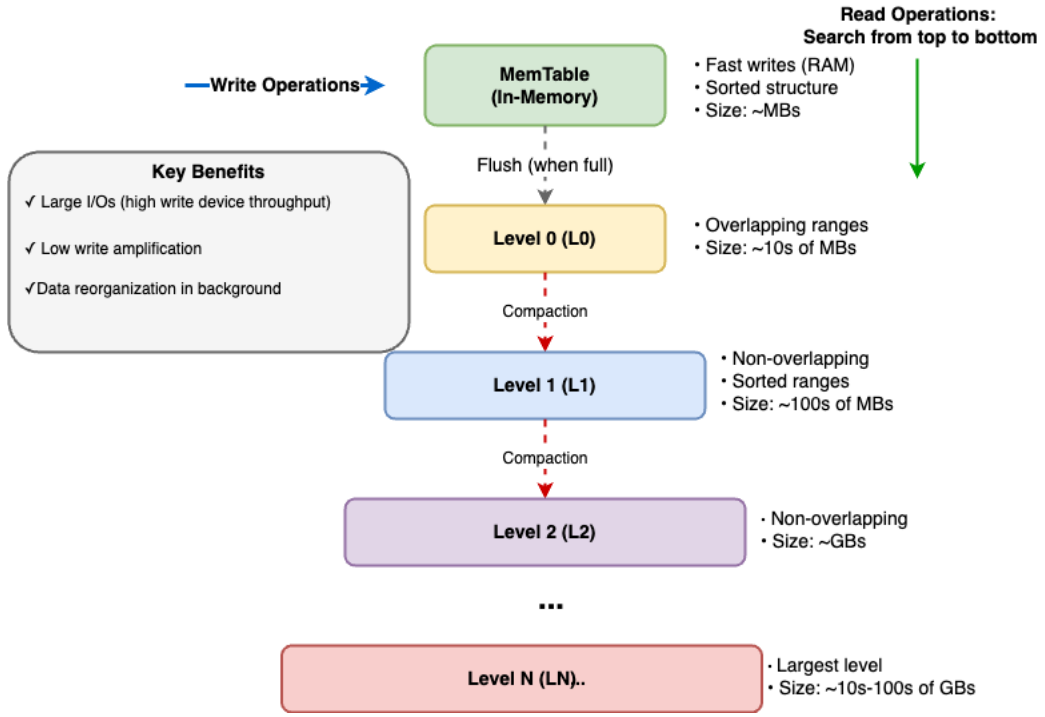
- Need to ingest data at high speeds (order of tenths of GB/s)
  - Thousands of compute nodes write simultaneously
- Must create searchable index during ingestion
  - Index must work efficiently regardless of I/O and data size
  - Write amplification must be controlled
  - Reads must remain fast even when executing in parallel with the ingestion phase

The Indexing Dilemma: POSIX and B-Trees Don't Scale

- Posix filesystems (LUSTRE)
  - Not designed for small files– metadata operations becomes bottleneck
  - Custom indexing withing the application adds complexity
- B-Tree writes generate high I/O amplification
  - Poor for flash - kills flash endurance
- LSM trees with lessons learned from the cloud come to the rescue

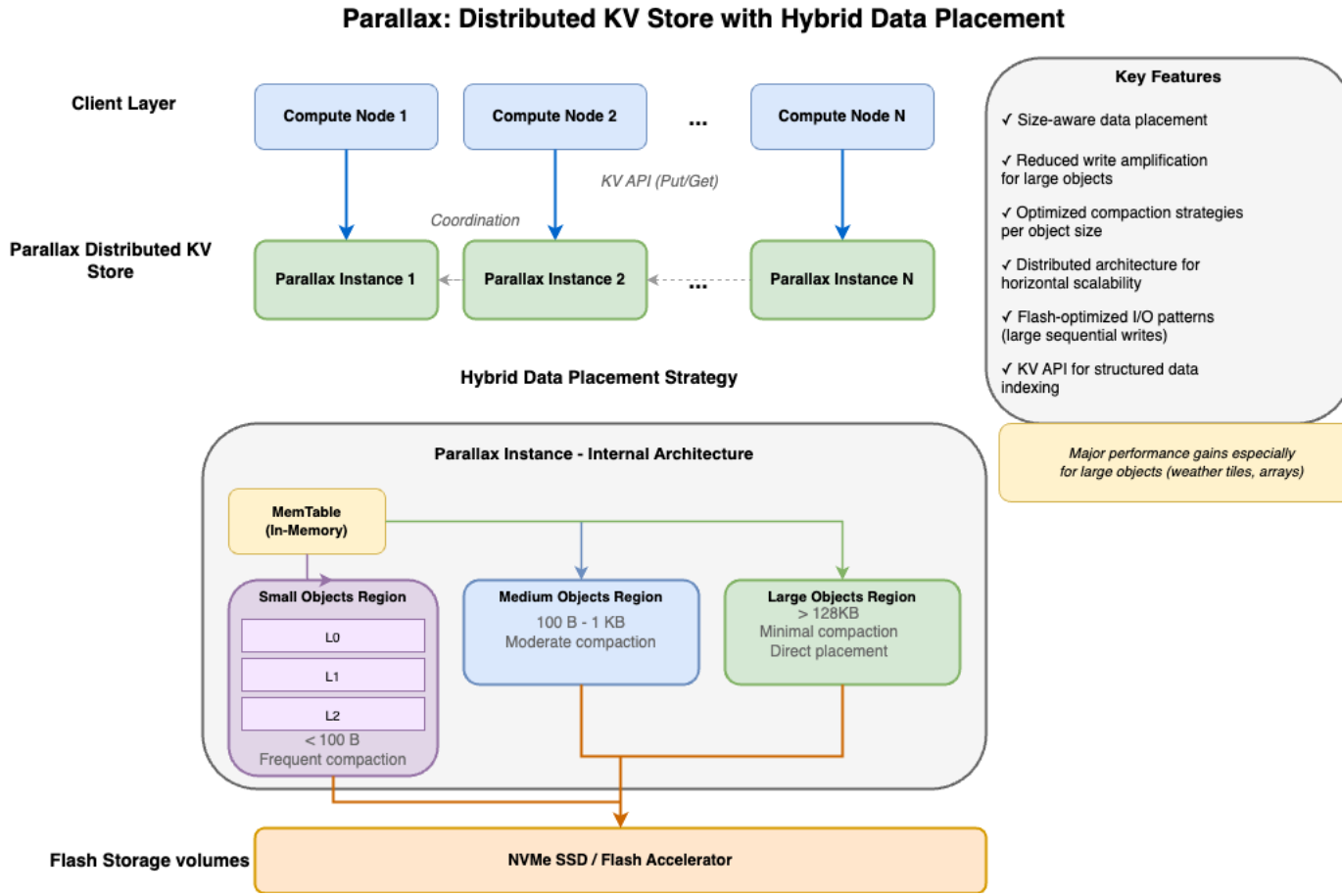
LSM-Trees: The Cloud's Answer to Write-Heavy Workloads

- Powers cloud infrastructure
  - Cassandra, RocksDB, LevelDB
- Guaranteed large I/Os (order of MB) regardless the I/O pattern
  - Batching of writes
  - Reorganization of the data
- Efficient reads comparable to a B-tree

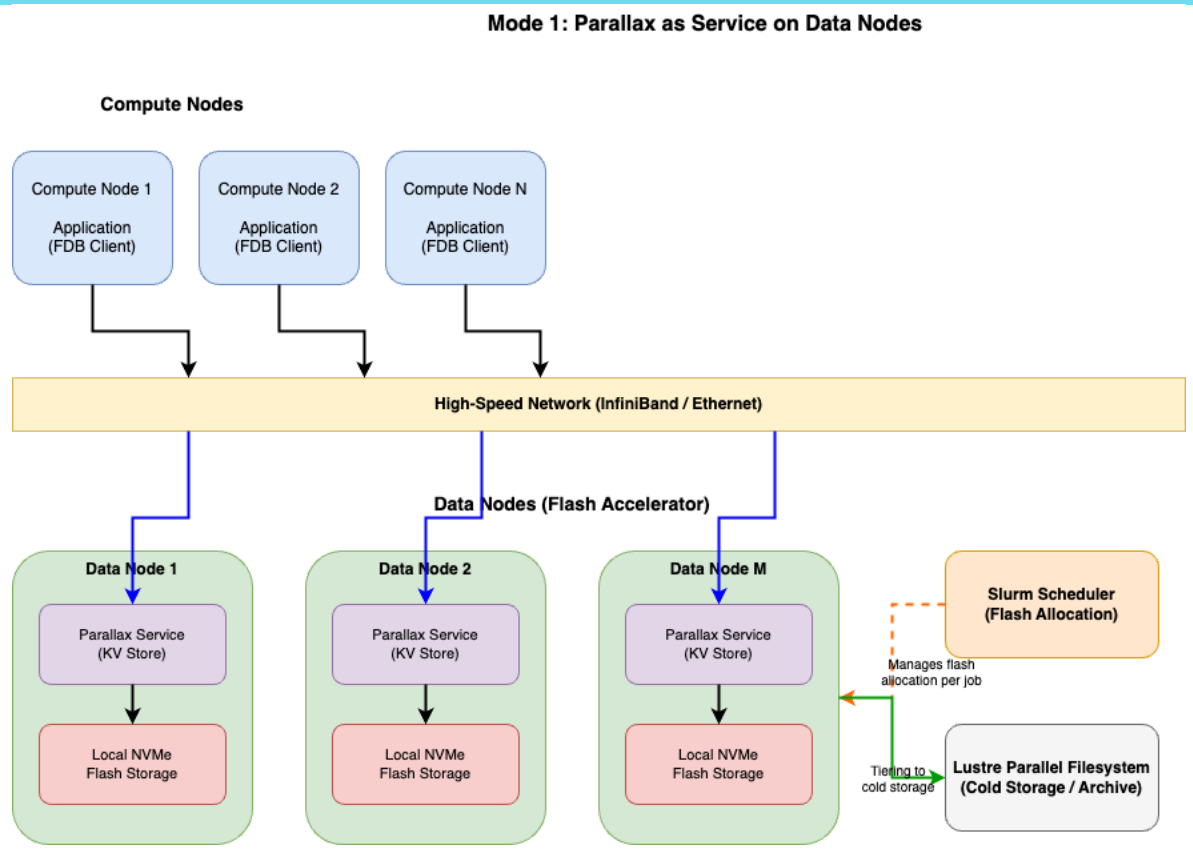


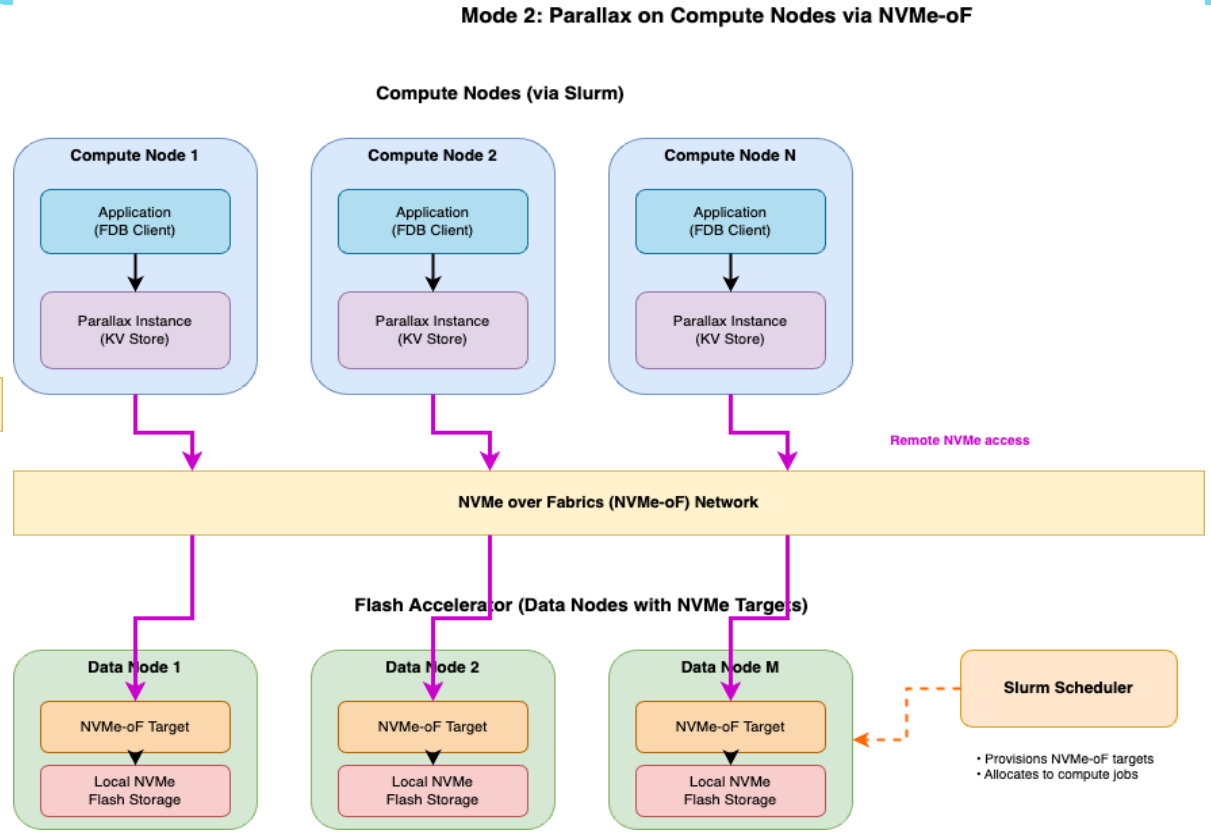
Parallax: LSM based KV Store for HPC

- Hybrid data placement to reduce amplification
- Different reorganization strategies according to object size
- Major gains especially for large objects
- Support for high speed networks
  - (Infiniband/BXI)



Deployment Modes

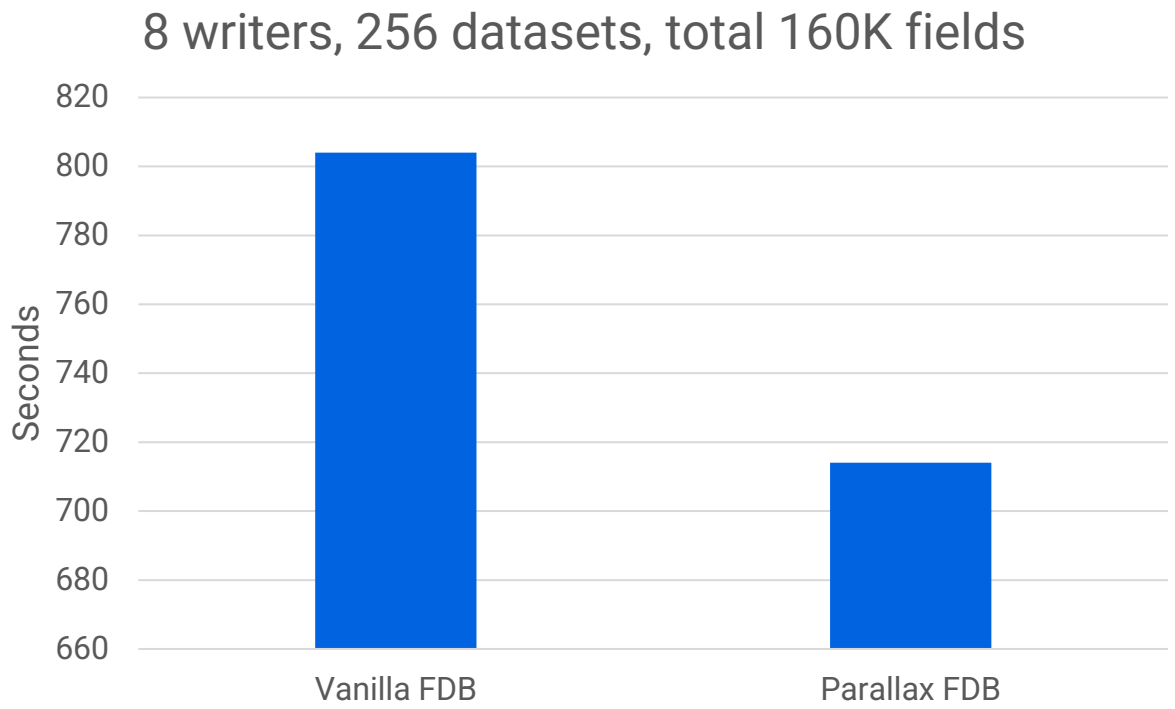




Preliminary Evaluation

- Single node setup
- 8 writers where they are a total of 256 tiles
- Stress the metadata part

Throughput for small KV pairs



EUPEX

Thank you! Questions?

- gesalous@ics.forth.gr

Backup Slides

Challenges: Provisioning and Indexing

- #1: Allocation & Exposure
  - Provision flash memory and assign it per job
  - Integrate with existing HPC schedulers (Slurm): provision → use → release
- #2: Data Indexing & Access
  - Efficient indexing for structured, tile-based data
  - Must handle heterogeneous I/O: small metadata, medium records, large arrays
  - Efficient ingestion of simulation concurrently with reads for post-processing tasks

Provisioned Flash with Key-Value(KV) Storage

- Flash tier for transient data
- Provisioning through Slurm
  - Storage lifecycle management per job
- Data flow:
  - Compute → Flash (hot) → Lustre (cold/archive)
- KV storage atop of flash