# Memory-Centric Storage for Exascale (MCSE)

**Dominik Vietinghoff[1], Julian Kunkel[2], Michael Kluge[3], Mirko Rahn[4], Olaf Krzikalla[1], Patrick Höhn[2], Philipp Falk[5], Philipp Zimmermann[4], Rui Machado[4], Sebastian Oeste[3]**

[1] DLR, [2] UGö, [3] TUD, [4] ITWM, [5] ThinkParQ

Contact: *julian.kunkel@gwdg.de*

## High-Performance Storage

High-Performance storage provides the hardware and software technology to query and store the largest data volumes at a high velocity of input/output. On exascale systems—i.e., systems processing $10^{18}$ floating-point operations per second—workflows will harness 100k of processors with millions of threads while producing or reading petabytes to exabytes of data.

## Motivation

- Increasing size of systems/applications/utilized datasets
- Increasing complexity of deep IO stacks
- Complex workflows lead to new challenges in data management
- Need for semantic aware storage for efficient IO

## Project Deliverables

- Analysis of existing IO semantics
- Unified interface for RAM and non-volatile memory
- IO-Verbs: Low-level IO API, alternative for POSIX-IO for parallel IO
- MCS2 (Memory-Centric Storage System) from IML (Infinite Memory Layer)
- Storage campaigns via MCS2 on HPC infrastructures

## Project Goals

- Analysis of relevant IO semantics
- API specification for IO-Verbs incl. demonstration
- Integration of IO-Verbs into IML and BeeGFS backends
- Porting IO-500 and FSDM to IO-Verbs
- High-level API for computational fluid dynamics

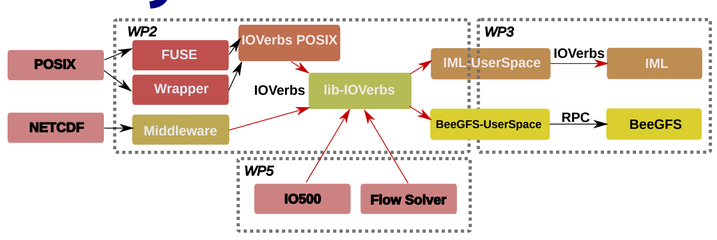
## Work Packages

WP7: Management

WP1: IO-Verbs
WP2: Implementation
WP3: MCS2
WP4: Campaign Storage
WP5: Application Porting

WP6: Evaluation

## IO Layers



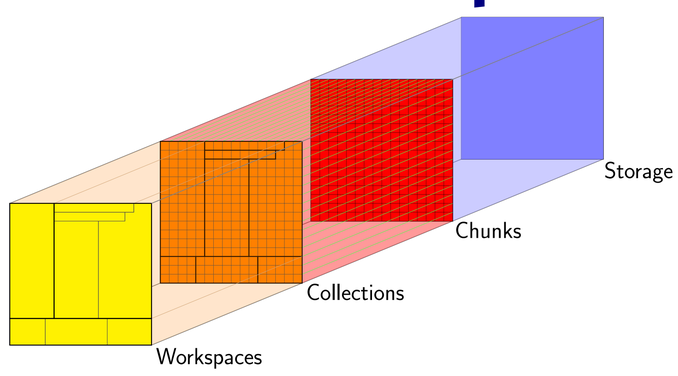

## IO-Verbs

- Support for IO semantics and parallel IO
- Low-level framework for higher-level APIs
- Transparent mapping of POSIX via compatibility library
- Integration with BeeGFS UserSpace client

## Applied IO Semantical Categories

- **Atomicity:** supports transactional groups of operations
- **Persistence:** control when data is written to non-volatile storage
- **Consistency:** defines when updates to shared data are visible for subsequent operations
- **Locality:**
  - Temporal locality: defines recently accessed data
  - Spatial locality: defines where in a distributed system the data is stored
- **Mutability:** defines whether data can be changed once written

## Structural Concept




## Analyzed IO Interfaces

- POSIX-IO
- MPI-IO
- HDF5
- S3
- NFS
- DAOS
- Zarr
- Lustre
- GekkoFS

## MCS2 (Memory-Centric Storage System)

- Extension of IML, addition of
  - IO-interfaces (via IO-Verbs)
  - Metadata
  - Persistence
- Provides applications abstract memory
  - Hides actual storage media
  - Supports violatile and non-violatile memory

## Campaign Storage

- Complex workflows lasting for weeks or months
- Integration of MCS2 with HPC-API HPCSerA
- Automation of data migration between storage media
- User specified requirements and required migrations

## Motivating Example

Facts about the models

- Stores data column-wise in memory
- Separates compute phase and IO phase

Existing NetCDF code for IO phase limits performance

```
size_t start[] = {0, 0};
size_t count[] = {nY, 1};
for(unsigned int col = 0; col < nX; col++) {
  start[1] = col; //select col (dim "x")
  nc_put_vara_float(dataFile, i_ncVariable, start, count,
    &i_matrix[col+boundarySize[0]][boundarySize[2]]);
}
```

## IO500 Example - MD Hard Write

Each process operates on an (independent) file

```
for (uint64_t i = items_start ; i < items_per_dir ; ++i) {
    fd = create(i);
    write(fd, buf, 3901)
    close (fd);
}
```

An alternative formulation

```
for(uint64_t i=0; i < items; i++){
    iov_collection_create(& batch, & address, & controlsCollection, & c);
    iov_write(& c, buf, 0, 3901);
    iov_collection_close(& c);
}
iov_batch_start(& batch, & list);
iov_wait(& list, & status);
```

## IO-Verbs in the Storage Stack

| | |
|---|---|
| **Application** | FlowSolver, IO500 |
| **Middleware** | MPI-IO, HDF5 |
| **System-Level** | POSIX, io_uring, **IO-Verbs** |
| **Storage-Level** | BeeGFS, Lustre, (p)NFS |
| **Physical Medium** | HDD, SSD, NVME, NVDIMM |

## Project Status

- Analysis of existing IO semantics
- Benchmarks of existing IO-stack in DLR FlowSolver
- Concept definition for IO-Verbs API
- Prototype of BeeGFS UserSpace and IML

## Next Steps

- Implementation of parallel IO in DLR FlowSolver
- First definition and reference implementation of IO-Verbs-API
- Increasing robustness of implementations
- Provide higher-level APIs to pass semantic information from the domain specific use-case to the storage system.

Funding code: 16ME0663K
Project duration: 1.10.2022 - 30.9.2025
Project lead: Prof. Julian Kunkel

Funded by the European Union
NextGenerationEU

SPONSORED BY THE
Federal Ministry of Education and Research