MAESTRO
DATA ORCHESTRATION

Middleware for Memory and Data-Awareness in Workflows

This project has received funding from the European Union's Horizon 2020 research and innovation program through grant agreement 801101.

www.maestro-data.eu

Authors
Prof. Dirk Pleiter
Dr. Manuel Arenaz

@maestrodata

/company/maestrodata

Motivation

- HPC and HPDA workloads are more and more I/O-intensive
- Performance bottlenecks are usually in the memory and storage systems
- Reducing and minimising data movement is very hard in general
- The HPC software stack was designed in a different era, to solve a different problem

The Maestro consortium is building a middleware library that characterises application data, reasons about how to load and store that data, assesses the cost of moving it and automates data movement across diverse memory systems

API Example

Producer

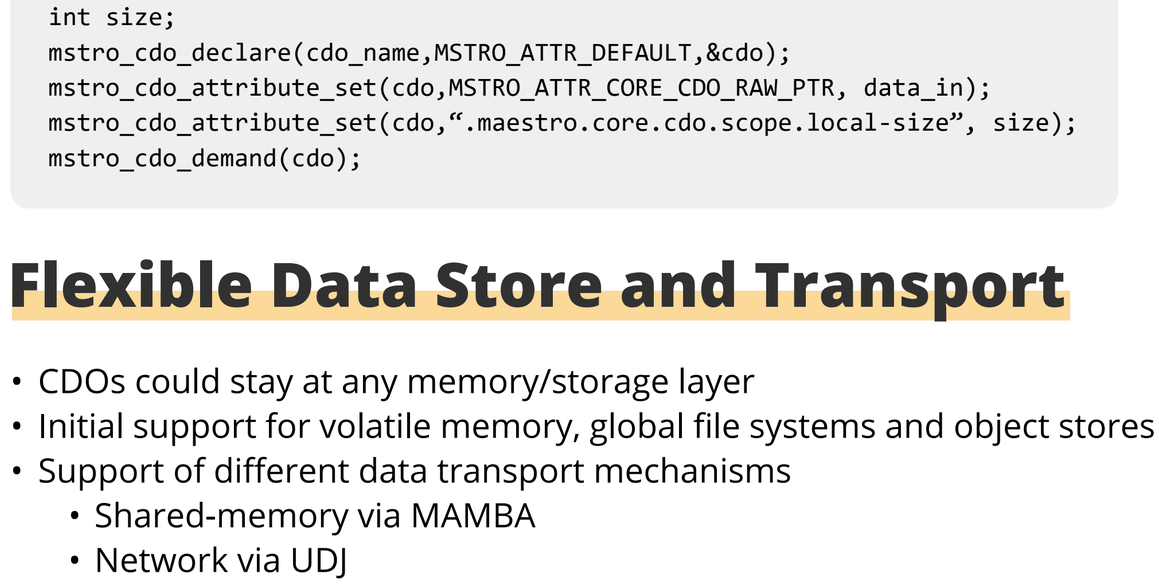
```
mstro_cdo cdo;
const char* cdo_name;
void data_out;
int size;
mstro_cdo_declare(cdo_name, MSTRO_ATTR_DEFAULT,&cdo);
mstro_cdo_attribute_set(cdo, MSTRO_ATTR_CORE_CDO_RAW_PTR, data_out);
mstro_cdo_attribute_set(cdo,“.maestro.core.cdo.scope.local-size”, size);
mstro_cdo_offer(cdo);
```

Consumer

```
mstro_cdo cdo;
const char* cdo_name;
void data_in;
int size;
mstro_cdo_declare(cdo_name,MSTRO_ATTR_DEFAULT,&cdo);
mstro_cdo_attribute_set(cdo,MSTRO_ATTR_CORE_CDO_RAW_PTR, data_in);
mstro_cdo_attribute_set(cdo,“.maestro.core.cdo.scope.local-size”, size);
mstro_cdo_demand(cdo);
```

Flexible Data Store and Transport

- CDOs could stay at any memory/storage layer
- Initial support for volatile memory, global file systems and object stores
- Support of different data transport mechanisms
  - Shared-memory via MAMBA
  - Network via UDJ
  - Global file system via POSIX I/O, MPI I/O
  - Object store via Mero using the Maestro I/O API (MIO)



Approach and Methodology

- Co-design: ascertain data movement and access requirements of target applications
- Develop new data-aware abstractions:
  - Used in any level of software (compiler, runtime, application)
  - Relevant for any type of data (array, file, unspecified)
- Design a middleware and library that enables:
  - Modelling of memory hierarchy
  - Reasoning based on cost of moving data objects
  - Automatic movement and promotion of data in memories
  - Powerful data transformations and optimisation
- Explore data-based performance portability of Maestro applications

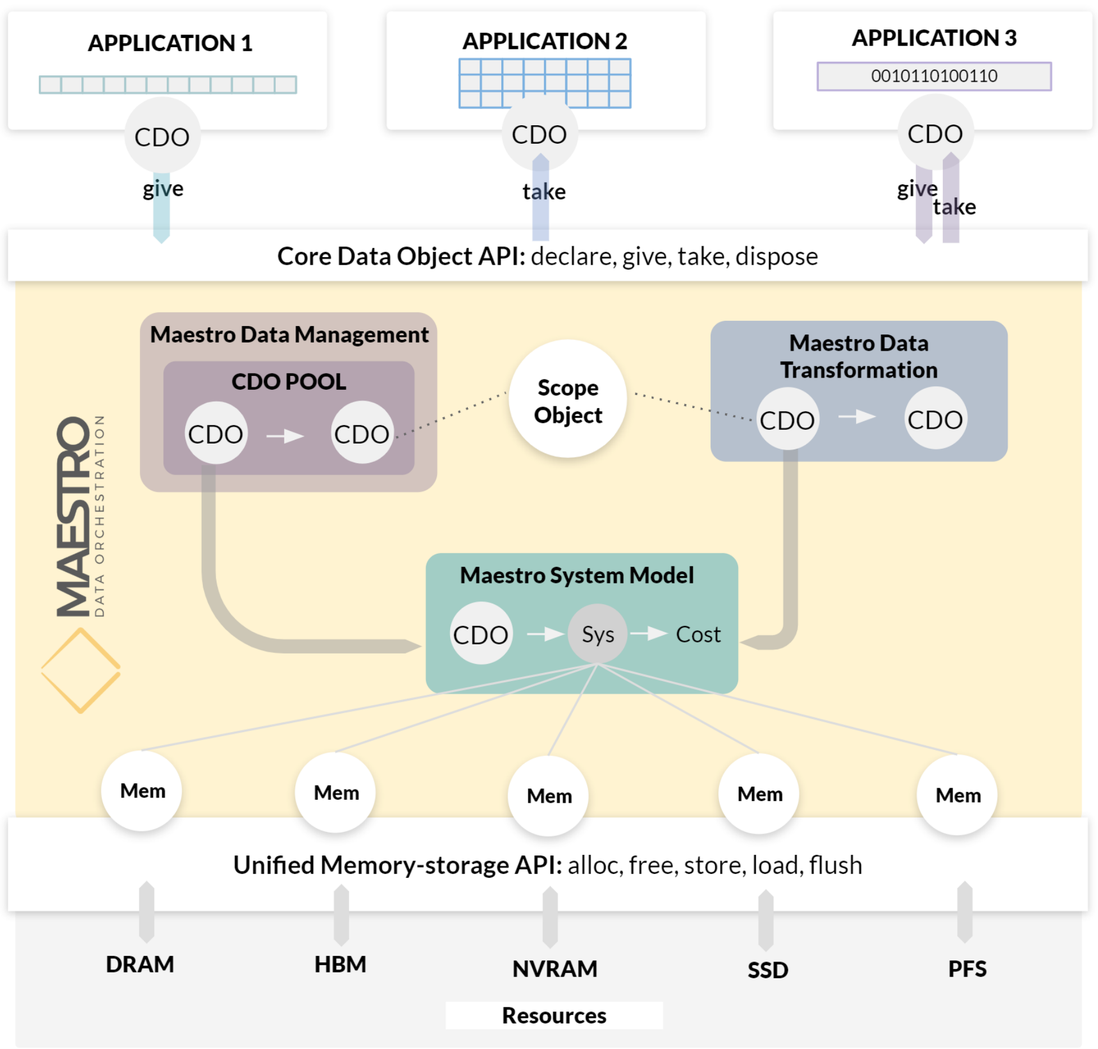


Figure 1: Design of the Maestro middleware. The CDO (Core Data Object) is at the heart of Maestro's design. It is used to encapsulate data and meta-data.

Selected Co-Design Use Cases

- **ECMWF's weather prediction workflow:** acquire and assimilate observations, produce numerical forecasts, post-process output and deliver products to customers.

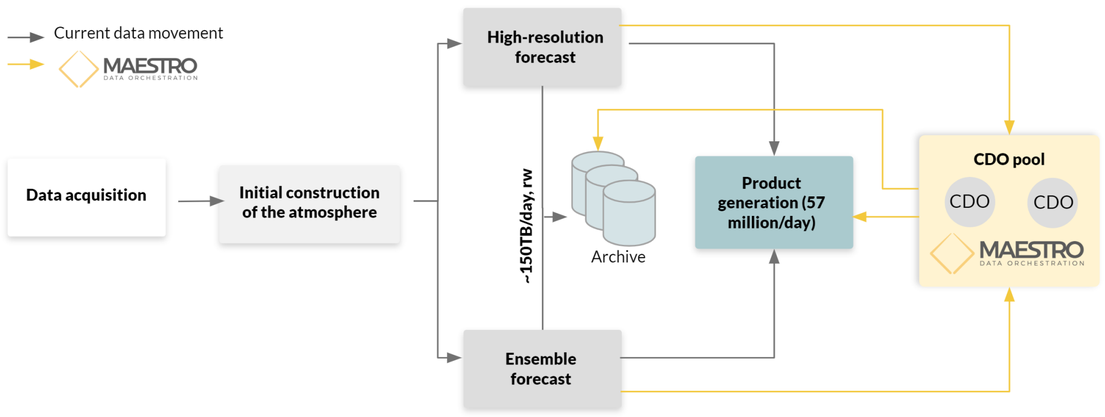


Figure 2: ECMWF workflow for weather forecast with and without the Maestro middleware to manage data movement.

Run time: 3-year project, started in September 2018
Current status: Description of the requirements of partners' applications and workflows in order to design the first specification of the Maestro middleware API
Next steps: Finalize the specifications of the Maestro middleware architecture and design the demonstrators for the ECMWF and SIRIUS use cases
Partners:

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE
cea
appentra make code parallel
ETH zürich
CSCS
ECMWF
SEAGATE
CRAY a Hewlett Packard Enterprise company