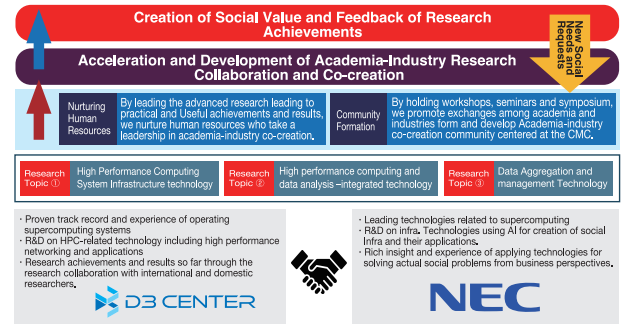


The Joint Research Laboratory for Integrated Infrastructure of High Performance Computing and Data Analysis

This joint research laboratory was established by NEC and D3 Center in May 2021 at the Suita Campus, the University of Osaka. In this joint research laboratory, we conduct R&D on essential computing and data infrastructure for academic use. Our R&D is aimed at supporting academic researchers in creating new social value through activities such as industry-academia projects, international collaborations, and community partnerships. We utilize D3 Center's large-scale computing (supercomputing) systems (SQUID, OCTOPUS, and mdx II) in our R&D.

- [Topic 1] High-performance computer system infrastructure technology that allows us to make maximum use of the hardware performance
- [Topic 2] High-performance computing and data analysis integrated computing technology that can accommodate a wide variety of computing needs
- [Topic 3] Data aggregation and management technologies that promote effective use of research data



Utility Functions for MPI Parallelization: Mutton

Background

Recent simulation programs commonly employ the MPI library for distributed-memory parallelization. However, MPI-based parallelization imposes a heavy burden on researchers, as all parallelization implementations must be written manually by researchers.

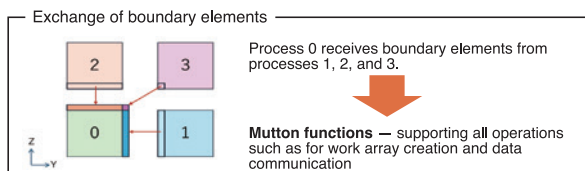
Our Approach: Mutton

To reduce this burden, our study developed Mutton (MPI utility function suite developed at the University of Osaka), a set of utility functions to support MPI parallelization. The initial version was designed for CFD simulations and currently supports the Fortran language.

Mutton can be applied to all or selected parts of MPI parallelization functions.

Main Functions

- MPI initialization and finalization
- Domain decomposition (2D partitioning)
- Data distribution, collection, and reduction
- Exchange of boundary elements between neighboring processes



Additional Features

- Functions for implementing **thread-overlap methods** to reduce communication time
- Functions to implement **assistant processes** that handle file I/O and real-time visualization concurrently with simulations



Mutton is publicly available at <https://www.nri.cmc.osaka-u.ac.jp/>

Research Poster session: "Divergence Prediction System for CFD Simulations" using Mutton functions

SCUP-HPC: System for Constructing and Utilizing Provenance on HPC Systems

Background

There is growing demand for Open Science, which promotes innovation through utilization of research data, ensuring transparency and reproducibility, and helping to prevent research misconduct. To ensure the transparency and reproducibility of computational experiments, provenance, which documents the data processing history, plays an important role.

Goal

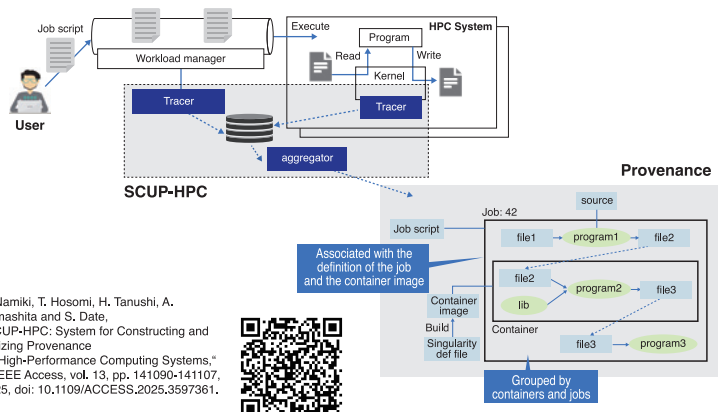
Provide a mechanism for recording the provenance of data generated within computer systems without imposing a burden on users.

Requirements:

- Record information necessary for reproducing and verifying research data.
- No modification of user programs or execution procedures is required.
- Minimize performance overhead.

Architecture

A tracer collects information about program execution and file access within the Linux kernel by using eBPF. By integrating information gathered from the workload manager, an aggregator constructs provenance that reflects the characteristics of HPC systems, such as information on jobs and parallel programs.



Y. Namiki, T. Hosomi, H. Tanushi, A. Yamashita and S. Date, "SCUP-HPC: System for Constructing and Utilizing Provenance on High-Performance Computing Systems," in IEEE Access, vol. 13, pp. 141090-141107, 2025, doi: 10.1109/ACCESS.2025.3597361.

Proof-of-Concept (PoC) System for Experimental Data Metadata Management

This PoC system generates, aggregates, and manages metadata in coordination with the transfer and storage of the experimental data.

- Creating metadata for experimental data (1)
- Transferring the set of experimental data and its corresponding metadata to a temporary storage area (2)
- Polling the storage directory for new experimental datasets and their corresponding metadata (3)
- Transferring the experimental data and metadata to Object Storage (ONION) and the metadata collection and management system, respectively (4)
- Retrieving the storage address (File Location URL) of the experimental data on Object Storage (ONION) and appending it to the metadata file (5)
- Registering the metadata in Apache Atlas (6)

