Cybermedia Center
Osaka University

Cybermedia Center

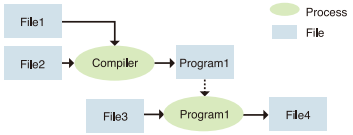
Osaka University, Japan

SC23 BOOTH 857

Provenance Management Framework for HPC Systems

Background

- Demonstrating the process used to generate the research data plays an important role in promoting the utilization of the data
- However, it is hard for researchers to take accurate records of the research processes ("provenance")



Example of Data Provenance

Goal

Allow users of HPC systems to construct provenance of data without end-user impact on usability and performance

Requirements

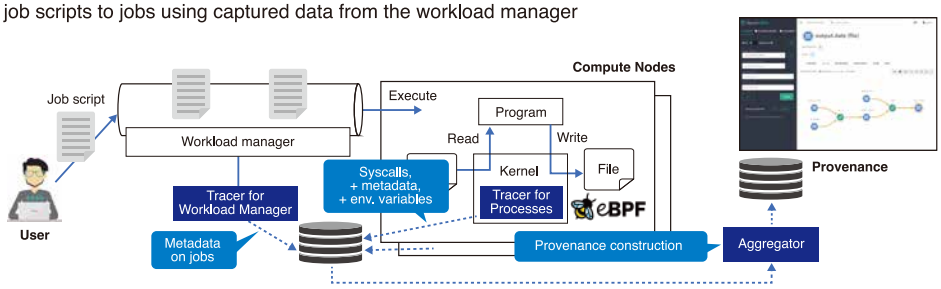
- Completeness: Preserving the operations to reproduce the data and the derivation
  - Questions need to be answered: Which are source files of the executed program?, Which job script needs to be executed to obtain the output file?
- Transparency: No modification on users' programs
  - Users have a lot of accumulated assets; introducing new frameworks/libraries could became a barrier
- Low-overhead: Minimal performance impact on users' programs
  - Performance is a top concern for HPC systems

Proposal

Capturing user operations from both a workload manager and an operating system kernel

Components:

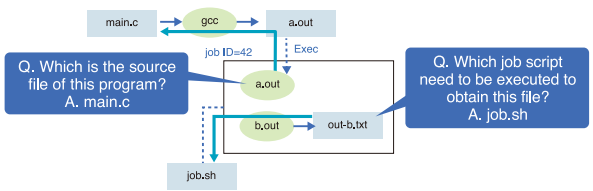
- Tracer for Workload Manager: Captures metadata (ID, job script, etc.) of jobs
- Tracer for Processes: Captures operations of running programs and metadata (job ID, full path and timestamps of files, etc.)
  - Record running programs and files that are read/written by the programs through tracing invocation of system calls in an OS
  - Record job IDs of the running programs from environment variables
  - Implemented by eBPF, the latest built-in tracing/observation mechanism of Linux kernel
- Aggregator: Constructing provenance from the captured data
  - Map a file read/written by a program to an input/output of the program and create input-program-output relationships in provenance
  - Group the input-program-output relationships by job using job ID
  - Link the job scripts to jobs using captured data from the workload manager



Constructed Provenance

Our system can construct provenance with program-to-source relationships and jobs, which are essential parts for reproduction in HPC systems in addition to input-program-output relationships

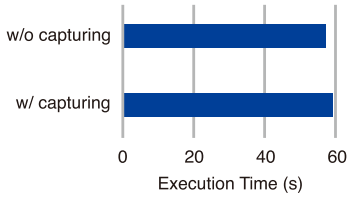
Example: Constructed provenance for building program a.out and executing the program and another program b.out in job job.sh



Performance Evaluation

A preliminary evaluation shows that performance overhead of the system is as low as 2.1% (1.21 seconds)

- Measuring performance overhead on computation caused by the capturing
- Application: BT-IO from NAS Parallel Benchmarks



Reference:
Yuta Namiki, Takeo Hosomi, Hideyuki Tanushi, Akihiro Yamashita and Susumu Date, "A Method for Constructing Research Data Provenance in High-Performance Computing Systems," 2023 IEEE 19th International Conference on e-Science (e-Science), Limassol, Cyprus, 2023, pp. 1-10, doi:10.1109/e-Science58273.2023.10254932

This work was carried out in Joint Research Laboratory for Integrated Infrastructure of High Performance Computing and Data Analysis https://www.nri.cmc.osaka-u.ac.jp/