Appendix: Artifact Description

Artifact Description (AD)

1 Overview of Contributions and Artifacts

1.1 Paper's Main Contributions

C_1 A methodology to replay HPC logs in any malleability-enabled computational cluster.

C_2 The design and study of a novel case study of a production supercomputer that initiates the adoption of malleable jobs into its traditional rigid workload.

C_3 The validation of MPI malleability using a real workload in a malleability-enabled supercomputer.

1.2 Computational Artifacts

The artifact source-code can be cloned from GitHub: https://github.com/siserte/dmr-poc.

A_1 `./dmr.h` and `./dmr.c` files

A_2 `./slurm-spawn/` directory

A_3 `./poc/` directory

A_4 `./mpdata-dmr/` directory

A_5 `./workload-generation/` and `./paper-experiments` directories

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_2, C_3	Figures 5, 7-15 Tables 1-2
A_2	C_2, C_3	Algorithm 2
A_3	C_1, C_3	Algorithm 1 Figures 1-3
A_4	C_2	Figures 4-5, 7-15
A_5	C_3	Figures 6-15 Tables 1-2

2 Artifact Identification

2.1 Computational Artifact A_1

Relation To Contributions

The artifact A_1 contains the Dynamic Management of Resources Library (DMRlib) which is responsible for linking malleable applications with the resource manager and orchestrates resources and MPI processes. In this regard, we leveraged DMRlib for designing the study case (C_2) and to validate MPI malleability in a supercomputer (C_3).

Expected Results

DMRlib automatically reallocates resources, redistributes data, and reshape the MPI process layout of the malleable applications (A_4) .

Expected Reproduction Time (in Minutes)

DMRlib is the framework so the time required to reproduce it corresponds to the setup which takes less than one minute.

Artifact Setup (incl. Inputs)

Hardware. Any HPC cluster. We need at least 3 nodes to evaluate DMRlib: one management node and two compute nodes to enable expansions to more than one node and shrinkages up to one node.

Software.

(1) GCC v10.4+
(2) Although in the paper we leveraged MPICH v3.2 (https://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz), Open-MPI v5+ is also supported.
(3) DLB v3.5.0 (https://pm.bsc.es/ftp/dlb/releases/dlb-3.5.0.tar.gz).
(4) DMRlib requires A_2.

Datasets / Inputs. Users need to define *communication efficiency* targets and *reconfiguration inhibition* periods. DMRlib is invoked by the jobs and the input is described in the experiments of A_5.

Installation and Deployment. DMRlib can be installed running `make` in the directory after configuring the environment variables for the dependencies. A_5 describes how to execute the experiments.

Artifact Execution

DMRlib relies on the resource manager A_2 and the malleable application A_4, so both have to be running. The experiment workflow is detailed in A_5.

Artifact Analysis (incl. Outputs)

As in the previous sections, DMRlib relies on other subsystems as A_5 output shows.

2.2 Computational Artifact A_2

Relation To Contributions

A_2 is the malleable resource manager of A_1 based on Slurm. In this regard, we leveraged A_2 together with A_1 for designing the study case (C_2) and to validate MPI malleability in a supercomputer (C_3).

Expected Results

The malleable version of Slurm is capable of reallocates resources previously assigned to running jobs. We expect that A_2 determines the most appropriate number of resources for each job depending on the users specifications and the cluster status.

Expected Reproduction Time (in Minutes)

Likewise A_1, A_2 is active during the whole experiment. Its setup time is around 10 minutes.

Artifact Setup (incl. Inputs)

Hardware. Any computer.

Software.

(1) GCC v10.4
(2) OpenSSL v1.0.2j (https://github.com/openssl/openssl/tree/OpenSSL_1_0_2j)

Datasets / Inputs. Slurm inputs are the submitted jobs as described in A_5.

Installation and Deployment. The installation is done with:

```
./configure                                          \
    --prefix=$DMR_PATH/slurm-install                 \
    --sysconfdir=$DMR_PATH/slurm-confdir             \
    --without-pmix  --with-ssl=$OPENSSL_PATH
make CFLAGS='-fcommon' CXXFLAGS='-fcommon'
make install
```

While the deployment is done during the experiments as shown in A_5.

Artifact Execution

A_2 relies on A_1 and the submitted jobs. The experiment workflow is detailed in A_5.

Artifact Analysis (incl. Outputs)

As in A_1, Slurm malleable relies on other subsystems as A_5 output shows.

2.3 Computational Artifact A_3

Relation To Contributions

A_3 is the workload submitter. It is responsible to extract users behavior and to adapt the information to the target cluster in order to be able to replay the workload (C_1) In this regard, we leveraged A_3 to validate MPI malleability in a supercomputer (C_3).

Expected Results

A_3 submits jobs to the queue handled by A_2 according to the input files.

Expected Reproduction Time (in Minutes)

A_3 is active during the whole experiment. The script is interpreted in execution time, so no compilation is required.

Artifact Setup (incl. Inputs)

Hardware. Any general-purpose computer that can run Python 3 interpreter.

Software. Python 3 interpreter.

Datasets / Inputs. A_3 expects users behaviors to be replayed. In the malleable experiments, it needs the behavior files for the *traditional users* and for the *generative user*.

Installation and Deployment. The Python scripts are executed during the experiments as A_5 describes.

Artifact Execution

A_3 submit jobs to A_2. The experiment workflow is detailed in A_5.

Artifact Analysis (incl. Outputs)

As in A_3 relies on other subsystems as A_5 output shows.

2.4 Computational Artifact A_4

Relation To Contributions

MPDATA is the scientific application executed by the PhD student. In this regard, it is essential in C_2.

Expected Results

MPDATA is leveraged to generate load for a given time depending on its configuration, so its results are irrelevant.

Expected Reproduction Time (in Minutes)

With the given configurations it can be completed between 36.5 and 1.45 hours, with 1 and 64 nodes, respectively.

Artifact Setup (incl. Inputs)

Hardware. Any computer.

Software. GCC compiler.

Datasets / Inputs. The input provided to MPDATA is a domain of $8,192 \times 1,024 \times 128$ cells that will run during $1,800$ steps.

Installation and Deployment. A_4 can be installed running `make`.

Artifact Execution

Since we only use one configuration of MPDATA, the generated binary can be launched as is.

Artifact Analysis (incl. Outputs)

The output is a message of "succesful execution" when finishing.

2.5 Computational Artifact A_5

Relation To Contributions

A_5 involves a series of scripts to validate MPI malleability in an actual systems (C_3) which coordinates the previous four artifacts.

Expected Results

A_5 provides all the necessary data to analyze the workload execution. Particularly, with A_5 can be created figures 6-15 and tables 1-2 and all the rational behind them.

Expected Reproduction Time (in Minutes)

Each experiments have run between 1.8 and 1.9 days.

Artifact Setup (incl. Inputs)

Hardware. Any HPC Cluster.

Software. Dependencies, scripts, and software in A_1, A_2, A_3, and A_4.

Datasets / Inputs. The experiments are launched with the inputs af the previous artifacts.

Installation and Deployment. Each experiment is launched with the `mnv_submission.sbatch` script configured with the desired strategy (see Section 4.4).

Artifact Execution

A worklflow consists of the task chain: T_1, T_2, T_3, T_4, T_5, and T_6.

T_1 consists of a batch job that launches the nested malleable Slurm (T_2) and the user-based submitter (T_3). In turn, T_3 will submit replayed jobs (T_5) and malleable jobs (T_4) to the Slurm deployed in T_2. The malleable jobs submitted in T_4 are reconfigured in T_6 with DMRlib, which takes actions based on T_2 rational.

Artifact Analysis (incl. Outputs)

A_5 generates all the necessary data to analyze the strategy evaluated in order to generete the paper results showed in sections 4 and 5.