Artifact Overview for PEANUTS

Kohei Hiraga[1][0000−0002−6743−436X] and Osamu Tatebe[1][0000−0003−4714−2164]

University of Tsukuba, Tsukuba Ibaraki, Japan hiraga@ccs.tsukuba.ac.jp

1 Getting Started Guide

1.1 Overview of the Software Stack

PEANUTS is integrated into the MPI runtime using OpenMPI's modular component architecture (MCA). To run OpenMPI with PEANUTS, the following software needs to be built together. PEANUTS is open-source and publicly available.

- **tsukuba-hpcs/peanuts**[4]: PEANUTS core libraries for C/C++
- **tsukuba-hpcs/ompi-peanuts**[6]: OpenMPI integrated with PEANUTS

The core library of PEANUTS, the C binding (libpeanuts_c), is linked with libmpi. ompi-peanuts internally calls the interface of libpeanuts_c, but this is done through a weak symbol. When building benchmarks that use ompi-peanuts, this linking relationship must be resolved either by solving it at build time or by loading libpeanuts_c with LD_PRELOAD at runtime. The latter method is adopted in Artifact evaluation.

Since correctly building OpenMPI and benchmarks with dependencies is very complicated, we use Spack [1], a build system for supercomputers, to perform the installation. We also provide packages for building PEANUTS using Spack.

- **tsukuba-hpcs/spack-packages** [8]: Spack packages for PEANUTS installation

Additionally, we have made the following references available:

- **range3/pmembench**[3] provides detailed information on the preliminary microbenchmarks and their results mentioned in the paper.
- **range3/rdbench**[2] provides the code for the RDBench benchmark used in the evaluation.
- **tsukuba-hpcs/mpiio-pmembb** [5] provides all the benchmark codes, raw logs, and visualization tools created for the evaluation in the PEANUTS paper.

1.2 Artifact Evaluation

PEANUTS has been evaluated on the Pegasus supercomputer at the University of Tsukuba. Pegasus is the only supercomputer in the world currently equipped with the latest and last generation of Intel Optane DCPMM, the 300 Series. Therefore, to reproduce the results in the paper, benchmarks need to be run on Pegasus. Unfortunately, it was not possible to create anonymous accounts on Pegasus for Artifact Evaluation.

As an alternative, we propose running PEANUTS on a local machine using a virtual cluster with Docker Compose. Additionally, it is possible to evaluate the performance of PEANUTS using our small-scale cluster chris9x, equipped with Intel Optane DCPMM 200 Series and InfiniBand EDR. Artifact reviewers can send us their SSH public keys, and we can create anonymous accounts on this cluster.

For Artifact evaluation, we have created a new repository for PEANUTS evaluation. This zip file is a copy of the archive from the following GitHub repository: **tsukuba-hpcs/peanuts-playground** [7].

For the actual Artifact evaluation procedure, please refer to the README.md in the peanuts-playground repository. The first part describes the evaluation method using Docker Compose, allowing you to start the evaluation immediately. The second part describes the evaluation method using the chris9x cluster. If you send us your SSH key, we can create an account for you.

If reviewers wish to evaluate PEANUTS on their own clusters, the hardware requirements include Intel Optane DCPMM and a cluster equipped with RDMA-capable NIC (RNIC). We have only tested RNIC with Mellanox InfiniBand. For the setup procedure, please refer to the setup instructions for the chris9x cluster in the README.md of the tsukuba-hpcs/peanuts-playground repository. The Spack configuration is detailed in spack/envs/chris90/spack.yaml, which can be used as a reference for setting up the environment.

1.3 Environment Setup Using Docker Compose

The following section is the same as README.md.

We have prepared a test environment for PEANUTS in a Docker container. While actual persistent memory is not available, you can verify the operation of PEANUTS. Using Docker Compose, we will build a virtual cluster consisting of four containers. MPI can be utilized between containers using OpenMPI-peanuts.

We are testing with VSCode and the devcontainer extension, so please follow the steps below to set up the environment:

1. Install VSCode (`https://code.visualstudio.com/`)
2. Install Docker by referring to Developing inside a Container (`https://code.visualstudio.com/docs/devcontainers/containers`)
3. Install the Remote Development extension (`https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.vscode-remote-extensionpack`)

Next, clone this repository, open it in VSCode, and build the devcontainer:

```
git clone --recursive git@github.com:tsukuba-hpcs/peanuts-playground.git
cd peanuts-playground
code .
```

Use the command palette to open the project in a container.
>**Dev Containers: Rebuild and Reopen in Container**
Once the container starts, build PEANUTS with the following commands. We use Spack for the installation of PEANUTS. Additionally, Python modules are installed with Pip for later visualization.

```
# in the container peanuts-h1

python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt

spack env create dev spack/envs/dev/spack.yaml
spack env activate dev
spack concretize -fU
spack install
```

1.4 Environment Setup Using a Real cluster chris9x

In the PEANUTS paper, the Pegasus supercomputer at the University of Tsukuba was utilized. However, Pegasus could not provide accounts for Artifact Evaluation. Instead, we explain how to run PEANUTS using the chris9x cluster owned by our laboratory.

The chris9x cluster consists of two nodes, chris90 and chris91, equipped with the second generation of Intel Optane DCPMM and InfiniBand EDR.

Perform the following tasks by connecting to chris90 via ssh. (Sorry, git clone and build will take a while due to poor NFS)

```
# checkout
git clone --recursive git@github.com:tsukuba-hpcs/peanuts-playground.git
cd peanuts-playground

# install python modules
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt

# prepre spack
source externals/spack/share/spack/setup-env.sh
```

```
# build PEANUTS and benckmarks and configuration tools with spack
spack repo add externals/spack-packages
spack env create peanuts spack/envs/chris90/spack.yaml
spack env activate peanuts
spack concretize -fU
spack install

# prepare interleaved devdax PMEM (requires root privilege)
# Preparation for DEVDAX is done in advance with root privileges.
sudo env PATH=$PATH ndctl destroy-namespace all --force
sudo env PATH=$PATH ndctl create-namespace --mode=devdax
sudo chmod 666 /dev/dax0.0
```

2 Step-by-Step Instructions

2.1 Run Benchmarks on a Virtual Cluster Using Docker Compose

Once the 'spack install' command is successfully completed, the following tools will be installed:

- OpenMPI integrated with PEANUTS (mpirun)
- Benchmarks:
  - ior
  - rdbench
  - h5bench_write / h5bench_read

Using these tools, you can run the benchmarks described in the PEANUTS paper. However, there are some limitations in the container environment:

- Since persistent memory is not available, we use the '/tmp/pseudo_pmem' file as a pseudo PMEM device.
- When running with process per node (PPN) > 1, 'MPI_Win_create' fails to register file-backed memory. Therefore, please run 'mpirun' with PPN=1.

We have prepared scripts to run the benchmarks inside the container:

- src/ior.sh
- src/rdbench.sh
- src/h5bench.sh

These benchmarks are intended to verify operation and are not suitable for reproducing the results of a paper. Execution is immediate.

When executed, the results will be output to the 'raw/' directory.

```
cd src
./ior.sh
./rdbench.sh
./h5bench.sh
```

Table 1. Chris9x cluster.

CPU	Xeon Gold 6326, 2.90GHz, 16 cores
Sockets/node	1
Num of Nodes	2
DRAM	DDR4 3200 MHz
PMEM	Intel Optane DCPMM 200 Series 128 GB * 8 slots
Network	InfiniBand EDR 100Gbps

Table 2. Pegasus compute node specifications.

CPU	Intel Xeon Platinum 8468, 2.1GHz 48 cores 1x
DRAM	DDR5 4400MHz 16GiB 8x
PMEM	Intel Optane DCPMM 300 Series 256GiB 8x
NW	ConnectX-7 InfiniBand NDR200 200Gbps 1x

2.2 Plot Results

We have also prepared Jupyter notebooks to parse the logs in the 'raw/' directory and create graphs:

- src/ior.ipynb
- src/rdbench.ipynb
- src/h5bench.ipynb

The required pip modules are installed in the '.venv/' directory. Open each Jupyter notebook using VSCode, select '.venv' from the 'Select Kernel' option in the upper right, and run the entire notebook.

If the benchmark fails to run, a corrupt results file may be generated and should be deleted.

2.3 Run Benchmarks on the Chris9x Cluster

The specifications of chris9x nodes are shown in Table 1. The chris9x cluster consists of two nodes, chris90 and chris91, equipped with the second generation of Intel Optane DCPMM and InfiniBand EDR.

The specifications of Pegasus' compute nodes are shown in Table 2.

Perform the following tasks by connecting to chris90 via ssh. (Sorry, git clone and build will take a while due to poor NFS)

```
# please use /shared/fish/$USER instead of /home/$USER
cd /shared/fish/$USER

# checkout
git clone --recursive git@github.com:tsukuba-hpcs/peanuts-playground.git
cd peanuts-playground
```

Table 3. Chris9x bandwidth upper bound.

Network	12.5 GB/s/node
PMEM write	15.7 GB/s/node
PMEM read	50.4 GB/s/node

```
# install python modules
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt

# prepre spack
source externals/spack/share/spack/setup-env.sh

# build PEANUTS and benckmarks and configuration tools with spack
spack repo add externals/spack-packages
spack env create peanuts spack/envs/chris90/spack.yaml
spack env activate peanuts
spack concretize -fU
spack install

# prepare interleaved devdax PMEM (requires root privilege)
# Preparation for DEVDAX is done in advance with root privileges.
sudo env PATH=$PATH ndctl destroy-namespace all --force
sudo env PATH=$PATH ndctl create-namespace --mode=devdax
sudo chmod 666 /dev/dax0.0
```

We have an interleaved PMEM namespace with 8 DIMMs. total size is 1 TB per node.

2.4 Run Benchmarks on the chris9x Cluster

We have prepared scripts for running benchmarks on the chris9x cluster with actual PMEM devices.

- src/ior-chris9x.sh
- src/rdbench-chris9x.sh
- src/h5bench-chris9x.sh

Before running, ensure that Spack is enabled.

```
source externals/spack/share/spack/setup-env.sh
cd src
./ior-chris9x.sh
./rdbench-chris9x.sh
./h5bench-chris9x.sh
```

PEANUTS automatically maps '/dev/dax0.0' into the virtual address space immediately after 'MPI_Init()' and registers PMEM to the RDMA-capable NIC with 'MPI_Win_create()'. Parameters can be passed to PEANUTS through the mpirun runtime command, but they are already set in the scripts above.

If other users are using '/dev/dax0.0', it cannot be executed simultaneously.

While IOR is run with various transfer sizes in the paper, the script executes it with a 32KiB transfer size. If you want to try other transfer sizes, uncomment the xfer_size_list variable in 'ior-chris9x.sh'.

2.5 Configuration and Estimated Runtime for Each Benchmark

The file **ior-chris9x.sh** writes 20 GiB of data per process to SSF. 16 processes per node (ppn), so 320 GiB/node. It executes write, remote read, and local read once per transfer size. Each mpirun execution takes about 1 minute due to initialization and termination processing. Therefore, the total execution time is: 2 (1 or 2 node count) * 3 (write, remote read, local read) * N (IOR transfer size) * 1 min = 6N min The total execution time is therefore about 6N min. If you try all transfer sizes, it will take 90 minutes or more. Experiments with smaller transfer sizes take more time.

rdbench-chris9x.sh creates 10 8 GiB files for strong scaling evaluation. Since rdbench also does the actual calculations, the run time is about 5 minutes per mpirun, for a total of 10 minutes.

h5bench-chris9x.sh writes 2.5 GiB of data per process to SSF. 40 GiB of weak scaling evaluation per node. Execution time is 4 mpiruns of a few minutes.

2.6 Plot Results on a Real Cluster

You can visualize the results on the chris9x cluster. To do so, connect to chris90 using VSCode with the Remote SSH extension. Alternatively, copy the log files to your local machine using scp and run Jupyter Notebook on VSCode within your local devcontainer.

Preliminary experiments have measured the parallel I/O performance of the PMEM devices on a single node of the chris9x cluster. The summary is shown in Table 3. 1 shows the results of the pmembench microbenchmark. Detailed results can be found in the document [3]. The peak write performance is 15 GiB/s, but it drops to about 10 GiB/s when accessed with 16 threads. This is a characteristic of the second generation of Optane DCPMM. For reads, it achieves around 42 GiB/s with 16 threads and a 32 KiB transfer size.

Regarding network performance, the chris9x cluster is equipped with InfiniBand EDR, providing 100 Gbps == 12.5 GB/s per node.

When visualizing the results of ior, success is indicated if the performance for 32 KiB and 2 nodes is close to:

- Write: 20 GiB/s
- Remote Read: 25 GB/s == 23.3 GiB/s
- Local Read: 84 GiB/s

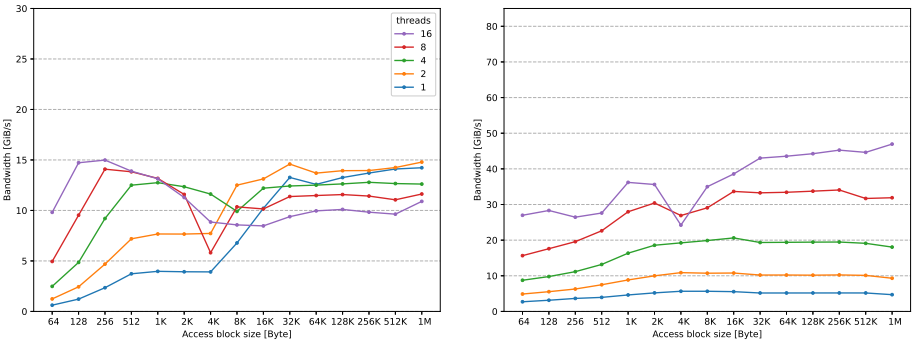

Fig. 1. Optane DCPMM 2nd generation performance measured on Chris90, left: writes, right: reads, with varying number of I/O threads and transfer size.

References

1. Spack: A flexible package manager supporting multiple versions, configurations, platforms, and compilers. (2024), `https://spack.io/`
2. Hiraga, K.: rdbench: 2D reaction-diffusion system benchmark using MPI and MPI-IO. (Jul 2022), `https://github.com/range3/rdbench`
3. Hiraga, K.: pmembench: Intel optane dcpmm performance comparisons for all generations (2023), `https://github.com/range3/pmembench`
4. Hiraga, K.: The core library of peanuts (2024), `https://github.com/tsukuba-hpcs/peanuts`
5. Hiraga, K.: Evaluation of peanuts (previously known as pmembb) and other systems using the pegasus supercomputer. (2024), `https://github.com/tsukuba-hpcs/mpiio-pmembb`
6. Hiraga, K.: Openmpi integrated with peanuts (2024), `https://github.com/tsukuba-hpcs/ompi-peanuts`
7. Hiraga, K.: Peanuts playground for testing (2024), `https://github.com/tsukuba-hpcs/peanuts-playground`
8. Hiraga, K.: Spack packages for peanuts installation (2024), `https://github.com/tsukuba-hpcs/spack-package`