PASS: A Power Adaptive Storage Server

Dedong Xie
University of Washington
USA
dedongx@cs.washington.edu

Theano Stavrinos*
Databricks
USA
theano.stavrinos@gmail.com

Jonggyu Park
University of Washington
USA
jonggyu@cs.washington.edu

Simon Peter
University of Washington
USA
simpeter@cs.washington.edu

Baris Kasikci
University of Washington
USA
baris@cs.washington.edu

Thomas Anderson
University of Washington
USA
tom@cs.washington.edu

A Artifact Appendix

A.1 Abstract

This artifact appendix provides a detailed process of derivation process for Figure 1, 2, 4-13, and Table 1. While we provide full design details and code for PASS, the experiments are resource-intensive running on servers with up to 10 NVMe SSDs, Intel Xeon CPUs, and high bandwidth network of 200Gbps NICs. To enable committee evaluation, we provide both package of code needed and a local setup with evaluation environment set with required hardware and software for reviewers to use.

A.2 Description & Requirements

A.2.1 How to access. The artifact is available online. Because we require a specific hardware setup that includes 10 NVMe SSDs and two servers connected with 200Gbps NIC, we provide network access to artifact evluators of our local setup.

A.2.2 Hardware dependencies. To run experiments of PASS, two servers are needed and connected through 100Gbps and 200Gbps RDMA network.

Target server:

- 10 NVMe SSDs (3.84 TiB each)
- Intel Xeon Gold 6430 processor
- 256 GiB DDR5 memory
- 100 GbE and 200 GbE RDMA NICs (e.g. ConnectX-5 and ConnectX-6)

Initiator server:

- Intel Xeon Gold 6530 processor
- 100 GbE and 200 GbE RDMA NICs (e.g. ConnectX-5 and ConnectX-6)

A.2.3 Software dependencies.

- OS: Ubuntu 22.04
- Kernel: Linux 5.15+ with NVMe over Fabrics and RDMA support
- SPDK: 23.09 with RDMA support
- Intel RAPL
- Programming Language: Python 3.10+
- Software: fio 3.38
- Other software: Filebench 1.4.9.1, RocksDB 9.8.0, YCSB 0.17.0

A.2.4 Benchmarks. The benchmark required is various fio jobs, Filebench workloads, db_bench workloads, and YCSB workloads. We have included the workload description files for these benchmarks in the package.

A.3 Set-up

We provide a local setup for evaluation so that no setup is needed if using our servers.
For those who want to setup their own environment, please follow the steps below to prepare the environment.

- Connect the two servers through the 100Gbps and 200Gbps network.
- Load the RDMA kernel module on both servers and assign IP addresses to the RDMA interfaces on both servers.
- Install SPDK from source with RDMA support.
- Install fio, Filebench, RocksDB, and YCSB.

A.4 Evaluation workflow

[Mandatory] This section should include all the operational steps and experiments which must be performed to evaluate your artifact is functional and to validate your paper's key results and claims. For that purpose, we ask you to use the two following subsections and cross-reference the items therein as explained next.

A.4.1 Major Claims. *Enumerate here the major claims (Cx) made in your paper. Follows an example:*

- *(C1): PASS achieves superior performance on microbenchmarks under power budgets.*
- *(C2): PASS improves performance of real application benchmarks.*
- *(C3): PASS provides performance isolation and robustness under power constraints.*

*Work done at the University of Washington

A.4.2 Experiments. *Experiment (E_{all}): [Run all experiments at once] [1 human-minutes + ~12 compute-hour]: run all experiments and get all figures of experimental results in the paper.*

We have provided a global script to run all experiments all at once and get results. The script is located in the top-level directory of PASS_AE named with `run_all_experiments.sh`.
[How to] To run the script, navigate to the top-level directory of PASS_AE and execute the following command:

```
bash run_all_experiments.sh
```

The expected result is all experiments used in the paper ran a round and that all figures 1, 2, 4-13 are generated automatically. We include reference results of each experiment under the directory named by figure number.
[Preparation] To prepare, ensure that all hardware and software requirements are met.
[Execution] The script will execute automatically by calling underlying experiment scripts to run each experiment, collect data, and generate each figure.
[Results] The results will be collected automatically into corresponding csv files under each figure's directory.
The mapping between the major claims and the collected results are:

- C1: Figure 1, 2, 4 and 5.
- C2: Figure 6, 7, and 8.
- C3: Figure 9, 10, 11, 12 and 13.

Experiment (E_n): [Individual experiments] [1 human-minutes + 0.25-4 compute-hour]: run of each individual experiments.

[How to] The PASS_AE directory is arranged in the structure of having top-level two parts, one being fio experiments, one being application benchmarks. Under each of these two parts, we further break to individual systems like the different application benchmarks used, and by purpose of fio experiments (motivation or microbenchmark).
To run an individual experiment, navigate to the corresponding directory with the name of the experimental result in the paper (e.g., fig1).
Then execute the shell script named `plot_fig1.sh` for example will run the experiment and produce results to csv file.
Please notice that different experiment runs for different durations. The microbenchmark susually runs under half an hour while the application benchmarks run for about 3 hours or up to 4 hours.
[Preparation] No specific preparation needed for running individual experiments.
[Execution] The script will automatically run the experiment. Please refer to the comments in the script to check its functionalities.
[Results] Results will be automatically collected for each figure by scripts called `extract_figureX_results.py` or `aggregate_experimentX_results.sh`. Running the corresponding `plot_figureX.py` can plot the results.

A.5 Notes on Reusability

The components in the artifact package includes the implementation of PASS with both offline profiling and online control.
For any setup, run the offline profiling part of PASS under directory `cpu_model` first then copy the generated CPU-side policy to the online control part of PASS.
The online controller part of PASS also relies on a power model of SSDs used, which is derived from the active power and idle power as well as bandwidth of SSD used.
With the offline and configuration parts set up, PASS can be used to control power usage.

A.6 General Notes

Connect to experiment setup: To access the setup of the experiments, please use the private key pair provided to access a jumper machine bicycle: bicycle.cs.washington.edu. After that jumper machine, please access to a evaluator account on monitor node of dock.cs.wahsington.edu.
Then please SSH to the experiment machine via `ssh kittitas1`.
The SSH private key is attached to this package and please use that to connect to the jumper machine.
A thorough instruction of how to connect is provided in the public repository in the file:
`artifact_evaluation_instructions.md`.
Avoid conflicts: Please wait for other evaluator's script to finish before running yours. As we have only one set of test servers, please time-sharing to use it as running concurrent job will cause interference and lead to inaccurate results. Our run all experiments script checks global lock to see if machines occupied by others. Please wait and retry later to avoid conflicts.