Appendix: Artifact Description/Artifact Evaluation

Artifact Description (AD)

I. Overview of Contributions and Artifacts

A. Paper's Main Contributions

C_1 **TurboFFT without fault tolerance** outperforms the popular open-source library VkFFT, and is comparable to the state-of-the-art closed-source library, cuFFT.

C_2 **TurboFFT with two-side fault tolerance** efficiently fuses the checksum computation into the FFT kernel, minimizing the fault tolerance overhead compared to exisiting offline fault-tolerant FFT (FT-FFT).

C_3 **TurboFFT's online error correction** protects FFT computation on-the-fly, obataining lower error correction overhead compared to the time-redundant recomputation in offline FT-FFT under error injections.

B. Computational Artifacts

Table I illustrates the relation between TurboFFT's contributions and the experimental figures presented in the paper.

.

TABLE I: Relation between TurboFFT (A_1) to C_{1-3}

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_1	Figure 1, 10-14, 21
	C_2	Figure 16-18, 19-20
	C_3	Figure 19-20, 22

TABLE II: Hardware Environment

System	Type	Description
System A	GPU	1x NVIDIA A100-SXM4-40GB
	GPU Power	400 W
	CPU	AMD EPYC 7713 64-Core Processor
	Cores per socket	64
	Threads per cores	2
	Memory	256 GB
System B	GPU	1x NVIDIA Tesla-T4
	GPU Power	70 W
	CPU	Intel(R) Xeon(R) Silver 4216 CPU
	Cores per socket	16
	Threads per cores	2
	Memory	192 GB

TABLE III: Software Environment

System	Software	Version
System A	`gcc`	`12.3.0`
	`cmake`	`3.24.3`
	`cudatoolkit`	`12.0`
	`python`	`3.10.14`
	`torch`	`2.5.1`
	`numpy`	`2.1.3`
	`matplotlib`	`3.8.4`
	`seaborn`	`0.13.2`
System B	`gcc`	`11.2.0`
	`cmake`	`3.26.4`
	`cudatoolkit`	`11.6`
	`python`	`3.9.18`
	`torch`	`2.5.1`
	`numpy`	`2.0.2`
	`matplotlib`	`3.9.2`
	`seaborn`	`0.13.2`

Artifact Evaluation (AE)

II. Introduction

This document demonstrates how to reproduce the results in the paper: **TurboFFT: Co-Designed High-Performance and Fault-Tolerant Fast Fourier Transform on GPUs**.

All supplementary files are available on Zenodo. The repository `PPoPP25_Artifact_TurboFFT.zip` consists of all code, including two **one-command scripts** `run_A100.sh` and `run_T4.sh` to reproduce all result figures list in Table I.

We executed all benchmarks in the paper using the hardware in Table II and software in Table in III.

III. Getting Started

This section guides you through the necessary steps to setup your machine. Please follow these steps before starting to reproduce the results.

A. Extract code repositories

Start by downloading `PPoPP25_Artifact_TurboFFT.zip`. Extract the archive into an empty directory and change into this directory using the following commands. Make sure that the absolute path to this directory contains no spaces.

```
# Create a reproduce directory
mkdir reproduce
cd reproduce
cp <path-to>/PPoPP25_Artifact_TurboFFT.zip ./
# Extract the artifact
unzip <path-to>/PPoPP25_Artifact_TurboFFT.zip
cd PPoPP25_Artifact_TurboFFT
```

Now, the folder reproduce contains all the necessary code to produce all results shown in the paper. The code consists of the `TurboFFT`, and `Common` repositores. `TurboFFT` includes the source code of high-performance FFT library shown in the paper, and the directory `Common` contains the cuda helper functions from `NVIDIA/cuda-samples`.

B. Install host machine compilation prerequisites

1) `GCC`: Please install a recent gcc version (>= 11.2.0).

2) `CMake`: Please install a `CMake` version (>= 3.24.3).

3) `CUDA Toolkit`: Please install CUDA Toolkit 12.0 for A100 machine or CUDA 11.6 for the T4 machine.

```
# Check the version after Installing
gcc --version
cmake --version
nvcc --version
```

C. Install host machine codegen & plot prerequisites

1) `Python`: Please install a recent version (>= 3.9).
2) `PyTorch`: Please install a recent version (>= 2.4).
3) `NumPy`: Please install a recent version (>= 2.0.2).
4) `Matplotlib`: Please install a recent version (>= 3.8.4).
5) `Seaborn`: Please install a recent version (>= 0.13.2).

```
# Sample instructions to set your python env
python -m venv .venv --prompt turbofft
source .venv/bin/activate
pip install upgrade
pip install torch
pip install numpy
pip install matplotlib
pip install seaborn
```

IV. Reproducing Results

The **one-command scripts** `run_A100.sh` and `run_T4.sh` allow you to regenerate **11 experimental result figures** on NVIDIA A100 GPUs (Figures 1, 10-14, and 16-20) and **2 experimental result figures** on NVIDIA T4 GPUs (Figures 21-22).

A. How to Run

1) **Ensure all dependencies are installed** (see Section III).
2) **Run the script**:
   - On NVIDIA A100 Machine:

   ```
   ./run_A100.sh
   ```

   - On NVIDIA T4 Machine:

   ```
   ./run_T4.sh
   ```

3) **View results**:
   - Experimental data will be available in the `artifact_data` directory.
   - Figures will be saved in the `artifact_figures` directory.

B. Workflow Overview

The scripts `run_A100.sh` and `run_T4.sh` execute the following steps:

1) **Environment Setup**: Configures environment variables.
2) **Code Generation**: Generates required CUDA kernels.
3) **Compilation**: Builds TurboFFT and related binaries.
4) **Benchmarking**: Runs benchmarks for TurboFFT.
5) **Plotting**: Produces figures matching the paper's results.

C. Runtime Details

Table IV shows the estimated execution time of `run_A100.sh` and `run_T4.sh`. The script `run_A100.sh` takes approximately **2 hours** on a machine with an AMD EPYC 7763 64-Core Processor and a NVIDIA A100 40GB GPU. The script `run_T4.sh` takes approximately **30 minutes** on a machine with an Intel(R) Xeon(R) Silver 4216 CPU and a NVIDIA T4 GPU.

TABLE IV: Estimated Execution Time

System	CodeGen	TurboFFT	Baseline cuFFT VkFFT	Plot	Total
`run_A100.sh`	20 s	15 min	90 min	3 min	2 hr
`run_T4.sh`	20 s	10 min	10 min	3 min	30 min