# Uncertainty-Aware Resource Allocation for Multi-Path Programs with In-Kernel Predictions (Artifact)

**Abigail Eisenklam**[1]
University of Pennsylvania,
Philadelphia, PA, USA

**Carlos A. Montenegro G.**[1]
University of California,
Santa Cruz, CA, USA

**Xian Wang**
University of Pennsylvania,
Philadelphia, PA, USA

**Yifan Cai**
University of Pennsylvania,
Philadelphia, PA, USA

**Robert Gifford**
University of Pennsylvania,
Philadelphia, PA, USA

**Linh Thi Xuan Phan**
University of Pennsylvania,
Philadelphia, PA, USA

**Ricardo G. Sanfelice**
University of California,
Santa Cruz, CA, USA

## Abstract

This artifact accompanies the paper *Uncertainty-Aware Resource Allocation for Multi-Path Programs with In-Kernel Predictions* appearing in ECRTS 2026. It is distributed as a Docker image and provides the scripts and datasets needed to reproduce the paper's core empirical results across six SPEC CPU 2017 benchmarks. The artifact trains and evaluates gradient boosting regression trees (GBRT) that predict two fine-grained execution properties of each benchmark (remaining execution time and next-window instruction rate) under different resource allocations, exports each model to a compact Q16.16 fixed-point binary, evaluates the speed, accuracy, and size of the models using a fixed-point C inference library, and applies weighted conformal prediction (WCP) to produce high-probability upper bounds on the prediction errors under different operating scenarios. The in-kernel decision logic, which implements MPORA: Multi-Path Online Resource Allocation, is included as source for inspection only, as it requires a custom kernel and specialized hardware to run.

**2012 ACM Subject Classification** Computer systems organization → Real-time operating systems

**Keywords and phrases** multicore, resource allocation, optimal control, learning, multi-path programs

**Digital Object Identifier** 10.4230/DARTS.12.2.9

**Funding** Partially supported by NSF Grants no. CNS-1955670, CNS-2039054, CNS-2111688, and CCF-2326606, by AFOSR Grants nos. FA9550-23-1-0145, FA9550-23-1-0313, and FA9550-23-1-0678, by AFRL Grant nos. FA8651-22-1-0017 and FA8651-23-1-0004, by ARO Grant no. W911NF-20-1-0253, by DoD Grant no. W911NF-23-1-0158, and by UC Alianza MX Grant no. SRPUCMX24-01.

**Related Article** Abigail Eisenklam, Carlos A. Montenegro G., Xian Wang, Yifan Cai, Robert Gifford, Linh Thi Xuan Phan, and Ricardo G. Sanfelice, "Uncertainty-Aware Resource Allocation for Multi-Path Programs with In-Kernel Predictions", in 38th European Conference on Real-Time Systems (ECRTS 2026), LIPIcs, Vol. 375, pp. 17:1–17:26, 2026. https://doi.org/10.4230/LIPIcs.ECRTS.2026.17

**Related Conference** 38th European Conference on Real-Time Systems (ECRTS 2026), July 7–10, 2026, Lund, Sweden

[1] Equal contribution.

## 1 Scope

The artifact reproduces the experimental results of the paper across the six evaluated SPEC CPU 2017 [2] benchmarks (`mcf`, `omnetpp`, `xz`, `xalancbmk`, `exchange2`, and `x264`), including:

- **Prediction accuracy, model size, and inference speed** of the XGBoost [1] models and their Q16.16 fixed-point C implementation, specifically the
  - normalized mean absolute error (NMAE),
  - coefficient of determination ($R^2$),
  - accuracy loss from floating-point to fixed-point inference ($\Delta$NMAE),
  - median per-sample prediction time, and
  - exported model size

  (corresponding to Table 2 of the related article).
- **NMAE versus observation-window size $\Delta$** aggregated across all six benchmarks for both targets (shown in Figure 2 of the related article).
- **Per-sample error distributions** for both prediction targets (i.e., the MAE histograms, an example of which is shown in Figure 3).
- **Operating-scenario distributions**, shown as the per-cluster `insn_sum` histograms that illustrate the distribution shift across operating scenarios (an example is given in Figure 4 of the related article).
- **Empirical coverage of weighted conformal prediction** per (benchmark, scenario, target), which should approach the target coverage $1 - \delta = 90\%$ in every scenario (Figure 5 of the related article).

The mapping between specific figures and tables in the paper and artifact outputs is recorded inline in the repository `README`: searching the `README` for a figure or table keyword (e.g. `FIGURE-1`, `TABLE-2`) locates the description of the corresponding artifact output.

The artifact also includes the fixed-point C inference library and the MPORA Linux kernel module source (`kernel/mpora.c`), which implements the in-kernel resource-allocation decision logic. This kernel code is provided *for inspection only*: it requires integration with a custom kernel and specialized hardware to execute as intended and is therefore not runnable within the artifact.

Absolute prediction-speed numbers are hardware dependent and will differ from the paper's measurements unless run on comparable hardware. The profiling datasets are provided as pre-collected CSV files because the profiling process itself takes several weeks on specialized hardware and is not reproduced by the artifact.

## 2 Content

The artifact package includes:

- **Datasets** (`profiles/`): one subdirectory per benchmark, each containing one normalized CSV per observation-window size ($\Delta \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$ seconds) together with the `.pkl` files holding the per-feature maximum values used for normalization. For each window size, the CSV file will contain profiles of the benchmark with various program inputs and allocations of last-level cache (LLC) and memory bandwidth. Therefore, each row of the CSV records the execution state of the benchmark at the end of one observation window when it is executed with a given input and resource allocation. The datasets total approximately 25 GB when the artifact is decompressed.
- **Pre-computed splits** (`splits/`): per-benchmark train/validation/test partitions, where the test set is further divided into WCP calibration, weight-network training, and WCP coverage-evaluation subsets.

- **Training and evaluation scripts**:
  - `train.py` (XGBoost training and evaluation),
  - `train_utils.py` (data loading, training, and C-side evaluation),
  - `plotting_utils.py` and `plot_delta_vs_metric.py` (figures),
  - `train_weight_net.py` and `train_weight_net.sh` (WCP weight-network training and coverage evaluation),
  - `cp.py` (the weighted conformal predictor),
  - `spec_input_feature_map.txt` (per-benchmark feature lists),
  - `requirements.txt`, and
  - `run.sh` (the main driver that reproduces all results).
- **Fixed-point C inference library** (`c_implementation/`): the Q16.16 fixed-point C inference code and its `Makefile`, against which the exported models are evaluated.
- **Kernel module** (`kernel/mpora.c`): the in-kernel MPORA resource-allocation decision logic. Provided view-only.
- **Generated outputs**: running `run.sh` produces trained models (float XGBoost boosters and their Q16.16 fixed-point binaries with metadata) under `models/`, and the metrics, histograms, line plots, scenario histograms, and WCP coverage report under `results/`.

## 3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The version published on DROPS is a gzip-compressed tarball of a Docker image (produced with `docker save`) that bundles the source code, all Python dependencies, and the profiling datasets. After downloading, ensure "Use containerd for pulling and storing images" is NOT selected in Docker Desktop settings. Then, start Docker and load and run the container with:

```
docker load -i DARTS-12-2-9-artifact-68c0aafc77a334178a287c40a84a49f5.tar.gz
docker run --gpus all -it mpora-artifact:v1.0
```

If running without an NVIDIA GPU, drop the `-gpus all` flag.

The source code is also maintained at `https://github.com/phan-lab/MPORA`; note that the profiling CSV files exceed GitHub's maximum file size and are therefore included only in the Docker image, not in the repository. The `README` in the GitHub repository provides instructions for pulling the latest image from Docker Hub.

Inside the container, results are reproduced with a single command, `./run.sh`. Individual XGBoost tasks and the WCP stage can also be run separately via `train.py` and `train_weight_net.py`; see the repository `README` for the full argument reference.

## 4 Tested platforms

The artifact requires Docker and, for GPU passthrough, the `nvidia-container-toolkit`. The XGBoost training/evaluation and plotting stages run on CPU. On an Intel® Xeon® Silver 4216 CPU @ 2.10 GHz, these stages (all six benchmarks × five $\Delta$ values, plus the aggregate plots) complete in less than an hour.

The WCP stage trains 24 MLP weight networks (6 benchmarks × 4 operating scenarios) and requires an NVIDIA GPU with CUDA support; a CPU fallback exists but is substantially slower. This stage completed in less than an hour on an NVIDIA RTX A2000; any comparable CUDA-capable NVIDIA GPU should run it efficiently.

## 5 License

The artifact is released under the MIT License.

## 6 MD5 sum of the artifact

68c0aafc77a334178a287c40a84a49f5

## 7 Size of the artifact

10.89 GiB

## References


1 Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi:10.1145/2939672.2939785.

2 Standard Performance Evaluation Corporation. SPEC CPU® 2017 Benchmark. https://www.spec.org/cpu2017/, 2017.