CacheFlow: Using Maximum Flow to Bound Cache-Based Preemption Delays (Artifact)

Tiancheng He
Department of Computer Science, Vanderbilt University, Nashville, TN, USA

Bryan C. Ward
Department of Computer Science, Vanderbilt University, Nashville, TN, USA

Abstract

This artifact accompanies the ECRTS 2026 paper *CacheFlow: Using Maximum Flow to Bound Cache-Based Preemption Delays* [1]. It provides the Rust plus Python implementation of every cache-related preemption delay (CRPD) analysis evaluated in the paper, the experiment harness used to compare them, the configurations that drive each figure in the paper, and a Docker-based build that fully reproduces the experimental claims of Sections 7.2–7.4. We claim all four ECRTS artifact-evaluation badges: Available, Functional, Reusable, and Results Reproduced.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Software and its engineering → Software verification and validation

Keywords and phrases Cache-related preemption delay; CRPD; max-flow; schedulability analysis; real-time systems; artifact

Digital Object Identifier 10.4230/DARTS.12.2.7

Related Article Tiancheng He and Bryan C. Ward, "CacheFlow: Using Maximum Flow to Bound Cache-Based Preemption Delays", in 38th European Conference on Real-Time Systems (ECRTS 2026), LIPIcs, Vol. 375, pp. 13:1–13:22, 2026. `https://doi.org/10.4230/LIPIcs.ECRTS.2026.13`

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

1 Scope

The paper introduces *CacheFlow*, a max-flow formulation of CRPD for uniprocessor fixed-priority scheduling. Schedulability and analysis-runtime claims are made against ten existing CRPD analyses across a six-dimensional synthetic design space. This artifact reproduces those claims:

- **Schedulability.** Figures 6, 7, and 8 of the paper, comparing CacheFlow variants against ten baselines as a function of system utilization and block reload time (BRT).
- **Harmonic refinement.** Figure 9, evaluating the periodic–harmonic refinement on strictly periodic task systems.
- **Analysis cost.** Figure 10, showing that the flow-based analyses scale gracefully to task-set sizes of 50, while Combined-Union-Multiset grows linearly to about 15 s per task system.

The contribution evaluated here is the implementation and the empirical evidence supporting the paper's quantitative claims; unrelated components of the broader development repository are not included in this artifact.

2 Content

The artifact is distributed as a single source tarball (`cacheflow-artifact-<sha>.tar.gz`, ∼1.7 MB compressed). Top-level layout:

`code/` Rust max-flow construction and solver, plus the Python experiment harness. The novel contribution is concentrated in `code/src/rta.rs` (graph builder and iterative response-time-analysis integration) and `code/src/dinic.rs` (Dinic's max-flow algorithm with budget-bounded early exit).

`schedcat/` Vendored task-model dependency. Used unmodified.

`experiments/` One subdirectory per experiment (`eval_draft`, `harmonic`, `timing_experiment`, plus the `sample` target used by the functional test). Each contains a paper-matching `config.yaml`, a fast-review `config_quick.yaml`, and a `run.sh` that selects between them via the `VARIANT` environment variable.

`artifact/` AE documentation (`README.md`, `REPRODUCIBILITY.md`, `REUSABILITY.md`), Dockerfile, helper scripts (`build_docker.sh`, `run_docker.sh`, `functional_test.sh`, `reproduce_figures.sh`, `verify_outputs.py`, `link_figures.sh`, `package.sh`).

The implementation is released under the MIT License (`code/LICENSE`); the vendored `schedcat/` retains its upstream license.

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). In addition, the artifact source tarball (`cacheflow-artifact-<sha>.tar.gz`) is hosted at the following OneDrive link:

`https://vanderbilt365-my.sharepoint.com/:f:/g/personal/bryan_ward_vanderbilt_edu/IgAelToW-N_aSZvt-tiNsDBaAcd82TEdfcy2NGaD_FZyS_c?e=Qdvz4U`

4 Tested platforms

The artifact has no special hardware or software requirements. The host machine needs:

- Linux or macOS with Docker Desktop or Docker Engine ≥ 20.
- About 5 GB free disk for the image and intermediate build layers, plus 50 MB for generated CSVs and PDFs.
- About 8 GB RAM is comfortable for every experiment; 2 GB is enough for the functional test.
- No GPU, accelerator, or real-time kernel is needed — the analyses are CPU-bound synthetic experiments.

5 License

The implementation is released under the MIT License (`code/LICENSE`).

6 MD5 sum of the artifact

785765534c0a062eff92c8e1dd7b36c3

7 Size of the artifact

1.55 MiB

Table 1 Wall times measured on an Apple M2 MacBook Air (8 cores, 8 GB RAM, Docker Desktop). `paper`-mode entries are extrapolations from per-design-point timings.

Stage	Figures	`quick`	`paper`
`build_docker.sh` (first build)	–	∼40 min	∼40 min
`functional_test.sh`	sanity	∼1 min	–
`eval_draft/run.sh`	6, 7, 8	∼3 min	∼12–24 h
`harmonic/run.sh`	9	∼11 min	∼12–24 h
`timing_experiment/run.sh`	10	∼2 min	∼24–48 h
`reproduce_figures.sh`	all	∼16 min	∼3–4 days

A Setup and Sanity-Check

Build and sanity-check:

```
tar -xzf cacheflow-artifact-*.tar.gz
cd cacheflow-artifact-*/artifact
./build_docker.sh                          # ~40 min first build
./run_docker.sh ./functional_test.sh       # ~1 min sanity check
```

The functional test prints `=== FUNCTIONAL TEST PASSED ===` after verifying that the Rust extension imports, all 94 unit tests pass, a 36-design-point experiment runs end-to-end, and four PDFs are produced.

B Reproducing the Paper's Results

The artifact ships two reproduction modes selected via the `VARIANT` environment variable. The recommended path for AE review is:

```
./run_docker.sh ./reproduce_figures.sh    # VARIANT=quick (default)
```

This runs all three experiments at a reduced sample count ($n_{\min} = 10$, $n_{\max} = 30$), preserves the qualitative trends of Figures 6–10 in approximately 16 minutes on an Apple M2 MacBook Air. After completion, a `figures/` directory at the project root contains paper-named symlinks (`Fig6.pdf`, `Fig7.pdf`, `Fig8_left.pdf`, `Fig8_right.pdf`, `Fig9_left.pdf`, `Fig9_right.pdf`, `Fig10.pdf`) for direct side-by-side comparison with the published paper. The companion `verify_outputs.py` script performs structural checks (row counts, dominance ordering, qualitative claims) and prints `PASS` or `FAIL`.

For pixel-level reproduction, the `VARIANT=paper` mode runs the exact Cartesian product specified in Table 1 of the paper ($n_{\min} = 30$, $n_{\max} = 1000$, the 95 % confidence-interval early exit at width 0.05, seed 12345):

```
VARIANT=paper ./run_docker.sh ./reproduce_figures.sh
```

This is multi-day on a laptop and is intended for camera-ready figure regeneration or for the Results Reproduced badge specifically. Table 1 gives measured wall times.

B.1 Figure-to-output mapping

Table 2 lists the source file backing each `Fig*.pdf` symlink. Side-by-side composites (Fig. 8 and Fig. 9 in the paper) are assembled by the paper's LaTeX build from the underlying panels; `Fig8_left.pdf` and `Fig9_left.pdf` are those panels.

Table 2 Paper figures and the artifact files that reproduce them. Files are produced by `reproduce_figures.sh`; `link_figures.sh` exposes them as paper-named symlinks under `figures/`.

Paper figure	Symlink	Source
Fig. 6 (BRT=8, 128 sets)	`Fig6.pdf`	`eval_draft/plots/brt_8-sets_128.pdf`
Fig. 7 (BRT=16, 128 sets)	`Fig7.pdf`	`eval_draft/plots/brt_16-sets_128.pdf`
Fig. 8 left (BRT=1)†	`Fig8_left.pdf`	`eval_draft/plots/brt_1-sets_128.pdf`
Fig. 8 right (BRT=4)	`Fig8_right.pdf`	`eval_draft/plots/brt_4-sets_128.pdf`
Fig. 9 left (harmonic, BRT=8)	`Fig9_left.pdf`	`harmonic/plots/brt_8-period_harmonic-p2.pdf`
Fig. 9 right (harmonic, BRT=16)	`Fig9_right.pdf`	`harmonic/plots/brt_16-period_harmonic-p2.pdf`
Fig. 10 (runtime vs. $\|\tau\|$)	`Fig10.pdf`	`timing_experiment/plots/sys_util_0.8.pdf`

† Only generated under `VARIANT=paper`; the BRT=1 sweep is omitted from `config_quick.yaml`.

B.2 Selective reproduction

Reviewers can run a single experiment using the `FIGS` environment variable:

```
FIGS=eval_draft ./run_docker.sh ./reproduce_figures.sh
FIGS=harmonic   ./run_docker.sh ./reproduce_figures.sh
FIGS=timing     ./run_docker.sh ./reproduce_figures.sh
```

C Verification

After `reproduce_figures.sh` completes, structural sanity checks can be run automatically:

```
./run_docker.sh python /workspace/artifact/verify_outputs.py
```

The script verifies that

1. every result CSV has the expected number of rows;
2. the dominance ordering claimed in Section 3 of the paper holds row-by-row (None ≥ Flow variants ≥ closed-form baselines);
3. Flow shows a measurable schedulability advantage at high BRT — the central claim of Section 7.2;
4. flow-based analysis runtime stays well under one second per task system at $|\tau| = 50$ — the central claim of Section 7.4.

The script exits non-zero on any failure and prints a one-line `PASS`/`FAIL` summary at the end. We have validated all four checks pass on the reference M2 MacBook Air configuration.

D Reusability

The implementation is structured for extension along three orthogonal axes, each documented in `artifact/REUSABILITY.md`:

- *New schedulability tests.* Subclass `Test` in `code/sched_tests.py` (pure-Python) or add a `#[pyclass]` Rust module under `code/src/` (Rust-backed). Declare the test's dominance relations and register it with the experiment manager.
- *New max-flow solvers.* Implement the `MaxFlowSolver` Rust trait (see `code/src/dinic.rs` as the canonical implementation) and parameterize `FlowState` on the new solver.

- *New design-space sweeps.* Drop a YAML config into `experiments/`; the harness handles parameter expansion, Bernoulli-estimator sampling with confidence-interval early exit, and CSV emission.

Reuse outside the paper's experimental setting (real-program profile data, custom period distributions, alternative cache models) is supported through the same surfaces, with the limitations made explicit in `REUSABILITY.md` §4.

References

1 Tiancheng He and Bryan C. Ward. CacheFlow: Using maximum flow to bound cache-based preemption delays. In *Proceedings of the 38th European Conference on Real-Time Systems (ECRTS 2026)*, volume 375 of *LIPIcs*, pages 13:1–13:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.ECRTS.2026.13.