Light Reading: Optimizing Reader/Writer Locking for Read-Dominant Real-Time Workloads (Artifact)

Catherine E. Nemitz ✉
University of North Carolina at Chapel Hill, NC, USA

Shai Caspin ✉
University of North Carolina at Chapel Hill, NC, USA

James H. Anderson ✉
University of North Carolina at Chapel Hill, NC, USA

Bryan C. Ward ✉
MIT Lincoln Laboratory, Lexington, MA, USA

Abstract

This paper is directed at reader/writer locking for read-dominant real-time workloads. It is shown that state-of-the-art real-time reader/writer locking protocols are subject to performance limitations when reads dominate, and that existing schedulability analysis fails to leverage the sparsity of writes in this case. A new reader/writer locking-protocol implementation and new inflation-free schedulability analysis are proposed to address these problems. Overhead evaluations of the new implementation show a decrease in overheads of up to 70% over previous implementations, leading to throughput for read operations increasing by up to 450%. Schedulability experiments are presented that show that the analysis results in schedulability improvements of up to 156.8% compared to the existing state-of-the-art approach.

2012 ACM Subject Classification Computer systems organization → Real-time system architecture; Computing methodologies → Shared memory algorithms

Keywords and phrases Reader/writer, real-time, synchronization, spinlock, RMR complexity

Digital Object Identifier 10.4230/DARTS.7.1.3

Acknowledgements DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering. © 2021 Massachusetts Institute of Technology. Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work. Work was supported by NSF grants CNS 1563845, CNS 1717589, CPS 1837337, CPS 2038855, and CPS 2038960, ARO grant W911NF-20-1-0237, and ONR grant N00014-20-1-2698. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGS-1650116. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was also supported by a Dissertation Completion Fellowship from the UNC Graduate School.

Related Article Catherine E. Nemitz, Shai Caspin, James H. Anderson, and Bryan C. Ward, "Light Reading: Optimizing Reader/Writer Locking for Read-Dominant Real-Time Workloads", in 33rd Euromicro Conference on Real-Time Systems (ECRTS 2021), LIPIcs, Vol. 196, pp. 6:1–6:22, 2021.
`https://doi.org/10.4230/LIPIcs.ECRTS.2021.6`

Related Conference 33rd Euromicro Conference on Real-Time Systems (ECRTS 2021), July 5–9, 2021, Virtual Conference

1 Scope

In this document, we provide an overview of how to replicate results from our paper. Figures 7, 8, and 9 and Table 2 can be replicated with our artifact package. These results are generated from two separate evaluations. The first is our overhead evaluation, which measures and compares the overhead of the locking protocols. The second is our set of schedulability experiments. Details of how to reproduce our results are given in the README files corresponding to those two experiments.

2 Content

Here we list the most relevant directories and files for both evaluations. All of the files and directories described for the overhead evaluation are in `overhead/`, and all of those for the schedulability evaluation are in `schedulability/`.

2.1 Overhead Evaluation

The artifact package includes:

- Readme guide: `overhead/README.md`
- Makefile: `overhead/Makefile`
- Experiment setup: `overhead/src/main_rw.c`
- Scripts: `overhead/scripts/`
- Plotting: `overhead/scripts/plots_rw.py`
- Example plots: `overhead/sampleplots/`

The code matching the algorithm described in Alg. 1 is here: `overhead/include/no-pfl.h`, and the version with overhead measurements can be found here: `overhead/include/pfl.h`.

2.2 Schedulability Evaluation

This portion of the artifact includes:

- Readme guide: `schedulability/README.TXT`
- Makefile: `schedulability/lib/schedcat/Makefile`
- Experiment setup: `schedulability/exp/pedf_lp.py`
- Plots: `schedulability/plots/`

The optimization problem constraints are in two files, listed with the following command.

```
ls schedulability/lib/schedcat/native/src/blocking/linprog/lp_rw_phase_fair*
```

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 is also available at `https://cs.unc.edu/~nemitz/papers/ecrts21_artifact.tgz`.

4 Tested platforms

4.1 Overhead Evaluation

The overhead experiment has been tested on both Linux and LITMUS^{RT} (an extension of Linux). This portion of our evaluation is in part based on an existing artifact [5] and is also based on a prior implementation of phase-fair locks [4]. A similar system setup is required, including:

- gcc
- Python 2.7
- Python3
- matplotlib, numpy, and pandas Python libraries

Our overhead experiment assumes the use of an x86 platform, and we tested it on a two-socket, 18-cores-per-socket platform, with two Intel Xeon E5-2699 v3 CPUs @ 2.30 GHz, 128 GB of RAM, and three levels of cache: per-core 32 KB L1 data and instruction caches, 256 KB L2 caches shared by pairs of cores, and 46,080 KB L3 caches shared by all cores on the same socket. We expect similar results from similarly-sized platforms. To reproduce our results, ensure that while the experiments are running, no other work is done on the machine. Additionally, frequency scaling should be turned off.

4.2 Schedulability Evaluation

The schedulability experiment was run on the 36-core platform described above, but the results do not depend on the platform. This portion of our evaluation is based on the artifact [2] that corresponds to the original presentation of the schedulability framework [3]. As such, we require the same system setup, including:

- Python 2.7
- Python NumPy Library
- Python SciPy Library
- SWIG 3.0
- GNU C++ compiler (G++)
- GNU Multiple Precision Arithmetic Library (libgmp)

For the linear programming solver, we used GNU Linear Programming Kit (GLPK) [1].

5 License

6 MD5 sum of the artifact

640c645393830f61d89f183bfa328079

7 Size of the artifact

2.2 MB

References

1 GLPK (GNU Linear Programming Kit). `https://www.gnu.org/software/glpk/`.

2 A. Biondi and B. Brandenburg. Artifact for Lightweight real-time synchronization under P-EDF on symmetric and asymmetric multi-processors. `http://people.mpi-sws.org/~bbb/papers/ae/ecrts16/pedf-synch.html`, 2016.

3 A. Biondi and B. Brandenburg. Lightweight real-time synchronization under P-EDF on symmetric and asymmetric multiprocessors. In *Proceedings of the 28th Euromicro Conference on Real-Time Systems*, 2016.

4 B. Brandenburg. *Scheduling and Locking in Multiprocessor Real-Time Operating Systems*. PhD thesis, University of North Carolina, Chapel Hill, NC, 2011.

5 C. Nemitz, T. Amert, and J. Anderson. Using Lock Servers to Scale Real-Time Locking Protocols: Chasing Ever-Increasing Core Counts (Artifact). *Dagstuhl Artifacts Series*, 4(2):2:1–2:3, 2018. `doi:10.4230/DARTS.4.2.2`.