# Know your Enemy: Benchmarking and Experimenting with Insight as a Goal

Mattia Nicolella*, Denis Hoornaert†, Shahin Roozkhosh*, Andrea Bastoni†, and Renato Mancuso*

*Boston University,   †Technische Universität München

*{mnico, shahin, rmancuso}@bu.edu, †{denis.hoornaert, andrea.bastoni}@tum.de

*Abstract*—**Available benchmark suites are used to provide realistic workloads and to understand their run-time characteristics. However, they do not necessarily target the same platforms and often offer a diverse set of metrics, leading to the lack of a knowledge base that could be used for both systems and theoretical research. RT-Bench, a new benchmark framework environment, tries to address these issues by providing a uniform interface and metrics while maintaining portability. This demo illustrates how to leverage this framework and its recently-added features to improve the understanding of the benchmarks' interaction with its system.**

*Index Terms*—**Benchmark, Profiling, Classification**

## I. Introduction

Benchmarking plays an indispensable role in the real-time community to evaluate, consolidate, and validate novel research. Using benchmarks to evaluate the performance of production systems is also of great value as simulators only partially depict real platforms' behavior. Benchmarking is thus beneficial for many aspects of the real-time research community: from system research to theoretical system modeling.

For system research, understanding the run-time behavior of realistic benchmarks is crucial to assess performance gains and showcase novel system designs. In such cases, *local knowledge* such as the maximal memory activity during the application run-time is valuable. On the other hand, theoretical research can benefit from using workload characteristics obtained empirically to test scheduling and regulation mechanisms against realistic loads. Informed improvements in the quality of regulation mechanisms require access to a coherent database of measurements providing a *global knowledge* via the aggregation of several performance metrics.

Despite its importance, to date, no existing reference knowledge base provides such *local* and *global* knowledge. Unfortunately, this leaves the community to rely on personal knowledge or experience. The problem is (at least partially) imputable to the high fragmentation of benchmark suites. In fact, the most commonly used benchmark suites in the community differ in several aspects, including (1) system compatibility, (2) requirements, (3) measured metrics, and (4) reporting formats.

To address these issues, we have created a new benchmark framework called RT-Bench[1] that was initially presented in [1] and that we are continuously improving and expanding. As part of the continued effort in the project, we hereby present the first milestone towards creating and establishing a public knowledge base of benchmark performance and profile characteristics. In the context of this demo, we will illustrate how the expanding knowledge base can be exploited to gain *wider* and *deeper* understanding of the interaction between realistic benchmarks and underlying hardware. To this end, we will demonstrate how to use RT-Bench, and showcase its recently-added features[2] to profile, extract, and report benchmarks' characteristics. Specifically, in our demo, we will:

- Present the latest advancements in the capabilities of RT-Bench originally presented in [1] that now includes 67 benchmarks.
- Illustrate how to interpret the profiles of complex benchmarks using key performance metrics. As an example, the profile of a benchmark issued from the `image-filters`[3] suite is provided.
- Provide a first comprehensive overview and classification of benchmarks issued from several suites.

## II. Focus of the Demo

This section presents and discusses the results of two classes of experiments to gain *global* and *local* knowledge about the benchmarks at hand. These experiments showcase the capabilities of RT-Bench to collect and export measurement data. In Section II-A, the *local* knowledge experiment leverages the capability of RT-Bench to simultaneously monitor several performance counters during the execution of a benchmark. On the other hand, the *"global knowledge"* experiments presented in Section II-B showcase the ability to directly contrast and classify benchmarks issued from different suites.

While RT-Bench is also compatible with `x86_64` Intel CPUs, our evaluation is based on ARM embedded platforms. In particular, we use a Raspberry Pi 3B+ to carry out the experiments. The system features (1) a four-CPUs (Cortex-A53) cluster operating at 1.5GHz[4], (2) per-core 32KB+32KB instruction and data caches, and (3) a 1MB shared last-level cache (LLC). We use a Linux kernel version 5.15.61, and RT-Bench applications have been compiled with GCC 10.2.1. The RT-bench benchmarks we used include the adapted version of the `image-filters` suite, the San-Diego Vision Benchmark Suite [2] (SD-VBS), and the TACLeBench suite [3].

---

[1]https://gitlab.com/rt-bench/rt-bench

[2]Comprehensive documentation on RT-Bench features can be found in the documentation: https://rt-bench.gitlab.io/rt-bench/

[3]https://gitlab.com/rt-bench/image-filters

[4]The frequency scaling governor is explicitly set to `performance`.
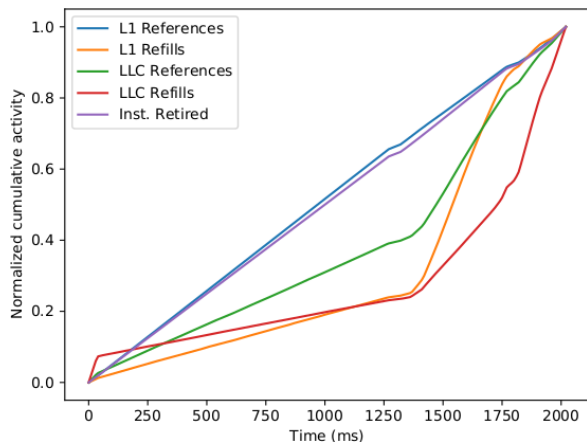
Fig. 1. Profiling of the `canny` image filter (see `image-filters` benchmark suite) via performances counters.



Fig. 2. Clustering of benchmarks and their suite based on instruction per clock (IPC) and last-level cache refills per clock (LLC-RPC).

### A. Local Knowledge - Profiling

For this experiment, we selected the `canny` benchmark from the `image-filters` suite to illustrate the profiling and analysis capabilities provided by RT-bench. In fact, due to its six-pass filtering, the benchmark implementing the Canny algorithm [4] is an ideal candidate. The benchmark is run using the `vga` input size (i.e., $640 \times 480$ pixels) on a single core, and all the supported performance counters have been monitored. Fig. 1 displays the normalized cumulative activity for each performance counter.

Fig. 1 clearly illustrates the fluctuations in resource demand taking place during a benchmark's execution. While the evolution of the instructions retired and the L1-D cache references remain stable throughout, the trend of other performance counters increases at various rates, hinting at the existence of several execution phases with distinct characteristics. In this case, such profiling is particularly interesting for budget-based memory regulation mechanisms.

### B. Global Knowledge - Benchmark Clustering

Using the aggregated performance counter activity and the timing measurements, it is possible to extract, compare, and classify benchmarks that belong to the various suites. Thanks to the wide choice of metrics reported, many classification criteria are available. Our demo will cover a handful of them. One such classification is reported in Fig. 2. Here, we showcase the relative positioning of the supported benchmarks w.r.t. their Instruction per Clock-cycles (IPC) and their Last-level Cache Refills per Clock-cycles (LLC-RPC). The figure is a visual representation of how CPU-bound vs. memory-bound the considered applications are.

Three observations can be made from Fig. 2. Firstly, only four benchmarks (all from SD-VBS) stand out w.r.t. how frequently they cause LLC refills. Secondly, all other benchmarks create less frequent LLC refills and are grouped together in a large cluster where only very few display an IPC higher than 1. Finally, most benchmarks lay in a dense low-IPC and low-refill frequency cluster. This cluster is mainly composed
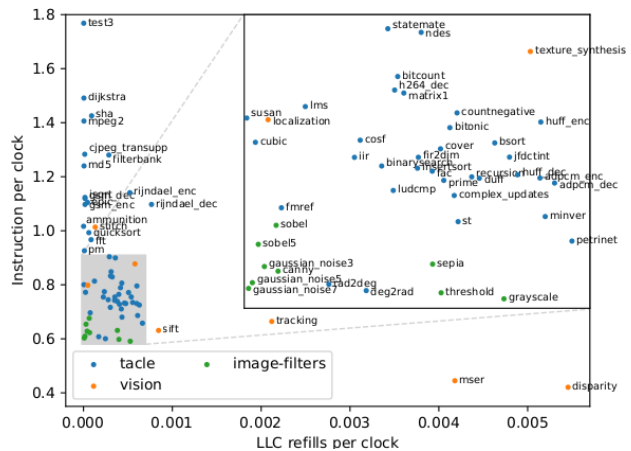
of TACleBench and image-filters benchmarks. The presence of the former can be explained by the small input data size used, whereas the latter can be explained by its reliance on costly floating point operations.

## III. CONCLUSION

In this article, we demonstrate how the recent advancements in RT-bench help get further insights into benchmarks.

The authors of this article and members of the RT-Bench project are committed to maintaining and expanding the tools, the supported benchmarks, and the measurement database. Future extensions include the analysis of Artificial Neural Networks models powered by widely used libraries (e.g., Tensorflow) and the capacity to define task sets to release simultaneously.

## REFERENCES

[1] M. Nicolella, S. Roozkhosh, D. Hoornaert, A. Bastoni, and R. Mancuso, "Rt-bench: An extensible benchmark framework for the analysis and management of real-time applications," in *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, 2022.

[2] S. K. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. B. Taylor, "Sd-vbs: The san diego vision benchmark suite," in *2009 IEEE International Symposium on Workload Characterization*.

[3] H. Falk, S. Altmeyer, P. Hellinckx, B. Lisper, W. Puffitsch, C. Rochange, M. Schoeberl, R. B. Sørensen, P. Wägemann, and S. Wegener, "TACLeBench: A benchmark collection to support worst-case execution time research," in *WCET 2016*, M. Schoeberl, Ed., vol. 55. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016, pp. 2:1–2:10.

[4] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.