USE-RB : Benchmarking how reasoners work in harmony with modern hardware

Christophe Gravier* and Julien Subercaze

Univ Lyon, UJM-Saint-Etienne, CNRS
Laboratoire Hubert Curien UMR 5516
F-42023 Saint Etienne, France
{christophe.gravier,julien.subercaze}@univ-st-etienne.fr
http://laboratoirehubertcurien.fr/

Abstract. As our computers embed more cores, efficient reasoners are designed with parallelization but also CPU and memory friendliness in mind. These latter contribute to make reasoner tractable in practice despite the computational complexity of logical fragments. However, creating benchmarks to monitor this CPU-friendliness for many reasoners, datasets and logical fragments is a tedious task. In this paper, we present the Université Saint-Etienne Reasoners Benchmark (USE-RB) that automates the setup and execution of reasoners benchmarks with a particular attention to monitor how reasoners work in harmony with the CPU.

Keywords: Reasoning, performance, benchmark, caches, memory.

1 Introduction

The number of Web applications relying on a triplestore and a reasoner has seen an exponential growth in the last years. This has resulted in new Web frontends for browsing, searching, and expressing complex queries over online data. As online data has never been as interlinked as today, the emerging challenge is to process these data in a computationaly efficient manner, especially when reasoning.

However, reasoning is a computational expensive task, and there are many logic fragments around, albeit no logic fragments fits all applications. For example, subsumption computation complexity in $\mathcal{SHOIN}$ description logic is decidable and exhibit a NEXPTIME time complexity. While such order of complexity may prevent the working ontologist to include a reasoner in their application – although the benefits to leverage implicit triples – the situation is not so desperate from practical point-of-view. Actually, even when the theoretical complexity seems to be intractable, there are optimized reasoners available [2, 4] that are usable for practical real world cases.

Historically, reasoning scalabilty has been tackled from a distributed computing point-of-view with practical system such as WebPie [1]. While these systems provided an unprecedented scalability, they fall short of scaling linearily

* This work has been supported by the CNRS PEPS-Secu project.

with the number of nodes in the cluster. One can observe that the most recent research have shifted towards parallelization on a single node equipped with many cores [2–4]. These approaches focus on designing in-memory, efficient and cache-friendly data structures and algorithms in order to win back otherwise lost CPU cycles as in hardware-unoptimized counterparts. By designing cache-friendly systems, data and code locality are expected to fully exploit CPU subsystems such as the prefetcher, the translation lookaside buffers and page management, to name a few. Actually, most of reasoning consists in memory I/O rather than raw arithmetic computations – few computations are made but usually data structures are to be traversed several times in different ways. For instance, systems such as RDFOx [2] are defying Amdahl law as they exhibit a close-to-linear scalability with respect to the number of threads devoted to the reasoning task. With major chips makers such as Intel running a many cores policy for the next years, one can expect that research efforts combined with technology advances will lead further reasoning performance to the real field. In order to sustain the research effort, it is therefore of the utmost practical interest to go deeper in understanding reasoners performance – examinating reasoners from a CPU-friendliness point-of-view is a mandatory effort.

Through USE-RB – University Saint-Etienne Reasoner Benchmark – we envision a CPU-friendly reasoners benchmark. USE-RB integrate all the facilities to plug any existing / to-be reasoners or datasets, to easily run and evaluate how a reasoner is working in harmony with the CPU. We believe that providing our benchmark to the community will contribute to the research for high performance many-cores reasoners.

2 USE-RB

2.1 Outline

USE-RB is expected to be configured with a list of reasoners, a list of datasets and a list of logic fragments. The cartesian product of these sets results in as many benchmark configuration to be run – a **benchmark task**. The module named **USE-RB**, the entry point of the execution of the program, is actually responsible for sequentially spawning as many instances of an external program named **ReasonersBenchmarked** – thus each benchmark task is run in isolation. A Java interface provides functional genericity in order to easily integrate new reasoner to the benchmark. Each execution of *ReasonersBenchmarked* can be parameterized with the number of warm-up iterations and the number actual iterations for which USE-RB will measure CPU counters.

2.2 Performance metrics

USE-RB track down CPU counters for each execution of a ReasonersBenchmarked instance. The primary counter is the wallclock time taken by the program to run one benchmark configuration – the actual number of CPU cycles,

translated to temporal units, used by the CPU for this process. Other metrics focus on hardware-specific counters for various CPU component, categorized as follows.

Instructions. Instructions metrics includes the number of branch mispredictions (branch misses) by the CPU. This is relevant for reasoners given the tremendous amount of data structures to traverse – these traversals are to be as much uniform as possible, therefore predictable for early CPU pipeline units. USE-RB also reports the total number of instructions, the number of instructions per CPU cycle, and the number of stalled CPU cycle per instruction.

Memory. Memory metrics includes the number of page faults, the number of transactional lookaside buffer loads and misses (total number and hit ratio). It also includes the number of stall CPU cycles when accessing any hierarchy of the memory.

Cache. Cache metrics are a sub category of Memory metrics – a highly prominent set of metrics when designing high performance applications, so that it falls into its own category in USE-RB. Cache metrics include the miss rate on all levels of cache. It also provides a per cache hierarchy level information on cache hit and misses. As for the first level of cache (L1), USE-RB differentiates data and instruction L1 caches. In case of a high cache miss rate, this allows to undersand whether the reasoner falls short to provide a good data or instruction locality.

2.3 Results and extensibility

USE-RB is shipped with vanilla datasets, logical fragments, and reasoners. It is natively able to run benchmarks by creating Benchmark configurations (see 2.1) by selecting one or several datasets, logical fragments and reasoners among :

– Datasets : 9 different sizes of a BSBM dataset (from 100,000 triples up to 100 million triples), Wikipedia Ontology, Wordnet ontology, Yago taxonomy, 7 LUBM datasets (from 1 and up to 100 million triples). We also ship dedicated datasets focusing on benchmarking the closure computation of the subsumption axioms.

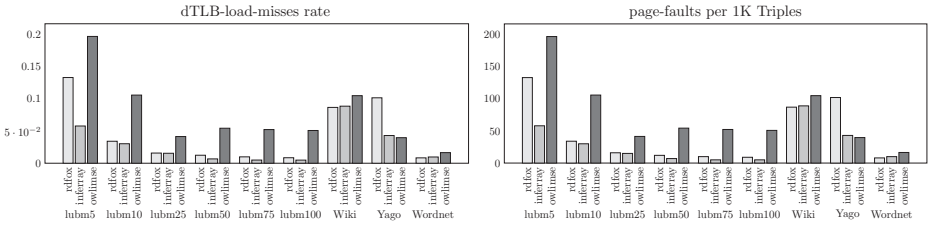

Fig. 1. Cache misses, TLB misses and Page Faults per triple inferred for the RDFS benchmark.

- Reasoners : JENA [5], OWLIM [6][1], SLIDER [3], SESAME [7], RDFOX [2], and Inferray [4].
- Logical Fragments[2] : RDFS default, RDFS-Full, Rho-DF, RDFS+.

One can easily add his/her own dataset, reasoner or logical fragment[3]. Figure 1 provides an example of figure that can be drawn from the execution of USE-RB. This figure reports data TLB misses and page faults per triple inferred for a RDFS-Full benchmark configuration[4]. Though the benchmark was run on all vanilla reasoner, we kept the most performant reasoners – omitted reasoners are outperformed by several order of magnitude as reported in [4].

3 Conclusion

In this paper, we presented USE-RB, a system for creating reasoners benchmarks and to observe how reasoners works in harmony with the CPU through the monitoring of CPU counters. USE-RB is publicly available at `https://github.com/telecom-se/USE-RB`. We believe that the presented benchmark is of the utmost practical interest for the researchers and industries who are willing to provide high performance reasoners. We also think that such frameworks are mandatory for promoting reproducibility of experiments, whenever possible. This framework also includes various popular datasets, reasoners, and implemented logical fragments – while providing the customizable features.

References

1. Urbani, J., Kotoulas, S., Maassen, J., Van Harmelen, F., Bal, H.: WebPIE: A web-scale parallel inference engine using MapReduce. In: Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 10, pp. 59-75 (2012)
2. Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems. In: AAAI, pp. 129-137 (2014)
3. Chevalier, J., Subercaze, J., Gravier, C., Laforest, F.: Slider: an Efficient Incremental Reasoner. In: SIGMOD, pp. 1081-1086 (2015)
4. Subercaze, J., Gravier, C., Chevalier, J., Laforest, F.: Inferray: fast in-memory RDF inference. In: VLDB, 9(6), pp. 468-479 (2016)
5. McBride, B.: Jena: A semantic web toolkit. In: IEEE Internet computing 6(6) pp. 55 (2002)
6. Bishop, B. et al: OWLIM: A family of scalable semantic repositories. In: Semantic Web 2(1), pp. 33–42 (2011)
7. Broekstra, J., Arjohn, K., Van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: ISWC, pp. 54–68 (2002)

[1] Requires a OWLIM-SE licence
[2] Logical fragments description in [4]
[3] `https://github.com/telecom-se/USE-RB`
[4] This exlcudes BSBM dataset since it does not support the expressivity of the RDF-Full logical fragment.