Energy-Efficiency of OWL Reasoners—Frequency matters*

Patrick Koopmann[1], Marcus Hähnel[2], and Anni-Yasmin Turhan[1]

[1] Institute of Theoretical Comp. Science, Technische Universität Dresden, Germany
firstname.lastname@tu-dresden.de
[2] Operating Systems Group, Technische Universität Dresden, Germany
mhaehnel@os.inf.tu-dresden.de

Abstract. While running times of ontology reasoners have been studied extensively, studies on energy-consumption of reasoning are scarce, and the energy-efficiency of ontology reasoning is not fully understood yet. Earlier empirical studies on the energy-consumption of ontology reasoners focused on reasoning on smart phones and used measurement methods prone to noise and side-effects. This paper presents an evaluation of the energy-efficiency of five state-of-the-art OWL reasoners on an ARM single-board computer that has built-in sensors to measure the energy consumption of CPUs and memory precisely. Using such a machine gives full control over installed and running software, active clusters and CPU frequencies, allowing for a more precise and detailed picture of the energy consumption of ontology reasoning. Besides evaluating the energy consumption of reasoning, our study further explores the relationship between computation power of the CPU, reasoning time, and energy consumption.

1 Introduction

Semantic technology applications often use ontologies and ontology reasoners as the core machinery to accomplish their tasks. Such applications are increasingly used on mobile devices [24]. Running times of ontology reasoning systems have been in the centre of attention of developers and users as long as these systems exist. On mobile devices, energy is a restricted resource, and as such at least as important as running times, but little is known so far about the energy consumption of ontology reasoners—although the motivation to investigate the energy consumption of reasoners is manifold.

Selection of the hardware for a reasoning task in an ontology-based mobile application requires knowledge on the energy consumption of carrying out this task. Reasoning could either be performed on a remote server or on the mobile device directly. While reasoning on a remote machine may save energy and computation

* This work is supported (in part) by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912 HAEC.

time, it can bring about problems in terms of privacy and security, and makes the service dependent on internet connectivity, as data has to be sent to another server. But if reasoning is performed on the mobile device (by reasoners such as Mini-Me developed specifically for mobile devices [17]) its energy consumption becomes relevant, as the energy available is simply limited by the battery.

Selection of the reasoner system for a reasoning task and ontology might regard its energy consumption. Little is known whether OWL reasoners differ as strongly in their energy consumption as they differ in the approaches they implement. Even for reasoners ported to and evaluated on mobile platforms [12,2] there is only little research on how the size, expressivity, and structure of the ontology and the performed reasoning task relate to energy consumption of the reasoner system.

Development of energy-efficient reasoners which use algorithms that behave energy-aware, requires detailed and reliable measurement methods for the hardware on which they are to be used. Such measurement methods should facilitate energy profiling of the different reasoning tasks—ideally for the individual components of the hardware.

Prediction functions for energy consumption trained by machine learning algorithms require reliable information about the energy consumption of a reasoning task at hand. While most research on prediction functions for reasoners focuses on running times, first research on predicting energy consumption has been undertaken in [6], albeit in a setup where only imprecise data on the energy consumption were available.

In this paper, we present an empirical study on the energy consumption (and running time) of OWL reasoners. We are not the first to address this topic. Motivated by the hard energy constraints of mobile devices, several research groups have evaluated the energy consumption of ontology reasoning on smartphones [16,23,6]. To the best of our knowledge, the first study on energy consumption of ontology reasoning was carried out by Patton et al. [16], who evaluated the energy consumption of answering SPARQL queries in the LUBM benchmark [8] and Schema.org [7]. They evaluated the reasoners Pellet [19], HermiT [5] and JENA rules [3] on the smartphone Samsung Galaxy S4. In order to measure the energy consumption, they replaced the battery of the phone with an external power supply that allows for power monitoring of the overall device. Based on their observations, they hypothesise that there is an almost linear relationship between execution time and power consumption of reasoners.

As the approach in [16] only works for smartphones with a replaceable battery, Valincius et al. [23] proposed an approach which uses the power management integrated circuit (PMIC) of the smartphone battery. Some of these PMICs, called *Fuel Gauge Chips* by Valincius et al., contain monitoring features that can be accessed by standard software libraries. Similarly to Patton et al., the authors evaluated SPARQL queries on the LUBM benchmark using the same

set of reasoners, but on a OnePlus One smartphone. They observed that the capacity of the battery affected the measured values. The capacity was reduced significantly by the experiments, so that experiments had to be rerun in different orders to compensate for this effect. This framework was later used by Guclu et al [6] to evaluate the ontology reasoners HermiT and TrOWL [21] on a Samsung Galaxy S6 and a Sony XPeria Z3, this time using a large set of ontologies taken from the OWL reasoner evaluation (ORE) competition from 2014 [1]. Their aim was to learn a prediction function for the energy consumption based on ontology metrics. The authors again found that the measured values differ significantly depending on the battery level of the device. This necessitated to incorporate the observed error rate in the interpretation of their measurements. The authors observed that performance and predictability of energy consumption can vary a lot depending on the hardware used. Moreover, contrary to the hypothesis by Patton et al., the execution time was not always linearly related to the energy consumption. The reason is that one of the smartphones, the Sony XPeria Z3, uses an ARM big.LITTLE architecture. This architecture allows the machine to switch operation freely between a slower, more energy-efficient cluster and a faster, less energy-efficient cluster, and makes it harder to obtain predictable measurements on energy consumption.

In conclusion, earlier studies on energy consumption of reasoners considered only a small set of available reasoning systems and ontologies (except the latter in [6]), and were only able to measure the energy consumption during reasoning for a smart phone as a whole and not for its components. All teams executed their experiments on Android smartphones, on which active background services, which have an impact on the overall energy consumption can only be controlled up to a certain point. Consequently, such measurements yield limited precision leading to uncertainties in the observed results. So far there are neither fine-grained, well-established methods of measurement nor benchmarks to assess the energy consumption of ontology reasoners.

To overcome the software and hardware related limitations that had an impact on the precision of energy measurements in these earlier evaluations, we used a different hardware setup in our experiments. More precisely, we chose a single-board computer with built-in sensors that measure power and energy consumption of various hardware components in a precise fashion. The device has a hardware architecture commonly found in Android smartphones, but gives the user full control over the hardware and software configuration. This way, we avoid the side-effects of unrelated tasks in the measurements, while obtaining results that can indicate energy consumption of mobile devices in general. The chosen hardware and architecture allows not only for more precise measurements of the energy consumption of ontology reasoning, but even for an evaluation of the energy consumption in regard of different CPU frequencies.

We used the ontologies from the ORE'15 benchmark [15] for computing ABox realisation in the OWL 2 DL and the OWL 2 EL profile. ABox realisation infers for all individuals in the data of the ontology to which of the named classes from the ontology they belong. Our choice is motivated on the one hand by

the relevance reasoning about individuals has to mobile semantic technology applications, and on the other hand by the fact that there are more OWL reasoner systems available that are capable of full ABox realisation than for (full SPARQL) query answering. To understand the impact of the hardware parameters on energy efficiency, we further chose to carry ou tour experiments under different CPU frequencies. We observed that, since the reasoning systems do not take full advantage of the computation power available, they perform more energy-efficient on lower CPU frequencies. For example, by reducing the CPU frequency from 2.0 GHz to 1.5 GHz, the energy consumption of reasoning is reduced by 43 percent on average, while the reasoning time is now increased by only 19 percent—giving rise to our claim that frequency matters.

The paper is structured as follows. In the next section we describe our experimental setup, used systems and data in detail. Section 3 lists and discusses our observations on running times, energy and power consumption as well as on the effects of the CPU frequency. The paper ends with conclusions and pointers to future work.

2 Experimental Setup

We describe our experimental setup and the rationale for its design in detail. The goal of this study is to obtain detailed measurements of the energy consumption of OWL reasoners. Furthermore, we want to explore the energy consumption and running time of OWL reasoners in regard of CPU frequency.

2.1 Experimental data and systems

Our experiments used a specific type of hardware, two sets of ontologies—one for the OWL DL and one for the OWL EL profile—and a set of reasoning systems capable of performing ABox realisation in at least one of the used profiles. For readers interested in access to the reasoners and ontologies used in our experiments, we provide links and further results online.[3]

Hardware. We performed our experiments on the ODROID XU3 by Hardkernel, a single-board computer with inbuilt-sensors to measure energy consumption of different components, whose asymmetric ARM big.LITTLE architecture provides an interesting trade-off space for software energy efficiency [9]. The ODROID XU3 has two clusters with 4 cores each:

- an ARM Cortex-A15 quadcore with 2.0 GHz maximum frequency (the 'big cluster'), and
- an ARM Cortex-A7 quadcore with 1.4 GHz maximum frequency (the 'little cluster').

[3] See `http://lat.inf.tu-dresden.de/~koopmann/energy-evaluation/`

Table 1. Metrics on the benchmark ontologies used in our evaluation.

Profile	#Ontologies	#TBox axioms average	#TBox axioms median	#ABox axioms average	#ABox axioms median
OWL DL	150	1,690	506	1,208	494
OWL EL	109	31,272	2,718	47,045	2,279

It further has 2 GiB of LPDDR3 RAM with a frequency of 933 MHz. Built-in sensors allow to measure the energy consumption of both clusters, the memory, and the GPU independently. As operating system, we used Arch Linux with Linux kernel version 4.9. We performed all experiments on the big cluster, while the evaluation environment was executed on the little cluster. To evaluate the impact of the CPU frequency on performance and energy consumption, we performed our experiments with CPU frequencies of 0.2 GHz, 0.5 GHz, 1.0 GHz, 1.5 GHz, and 2.0 GHz.

Ontologies. To get a balanced mix of ontologies with varying structures and properties, we took the ontologies from the benchmark used in the 2015 edition of the *OWL Reasoner Evaluation competition* (ORE'15) [15]. The competition evaluated OWL reasoners on reasoning time and success rates. It has different tracks for the reasoning tasks *consistency checking*, *classification* and *realisation* and provides for each of them a set of ontologies in the lightweight OWL EL profile and a set of ontologies in the expressive OWL DL profile. For both OWL profiles, we evaluated the reasoning task realisation, since it reasons over ABox data and is implemented in several reasoning systems.

At the ORE'15 ABox realisation track for the OWL DL profile, the reasoners used in our evaluation could only solve between 106 and 163 of the 264 ontologies. To obtain an experimental setup significant for the comparison of reasoners without overly many timeouts, we restricted the test set to the 150 ontologies of the track that had less than 10,000 statements. For ORE'15 ABox realisation track for the OWL EL profile, the situation was different. Here the dedicated OWL EL profile reasoner ELK could compute realisation for all but 7 of the ontologies at the competition, which is why we selected all of the 109 ontologies in the OWL EL profile of the ORE'15 realisation track for our experiments. Table 1 shows for both profiles the average and median of the number of TBox axioms and ABox axioms of the selected ontologies.

OWL reasoning systems. Our evaluation uses reasoners implemented in Java, as these can be executed directly on the ARM architecture of the ODROID. Reasoners not implemented in Java would require a recompilation from the sources for the architecture of the ODROID. Unfortunately, this technical constraint ruled out state of the art reasoners such as Konclude [20], Fact++ [22], PAGOdA [25] and ELepHant [18], and they are left for future work. Note that while PAGOdA itself is implemented in Java, the latest version has dependencies to the system-dependent datalog engine RDFox [14]. We used two sets of

reasoners each dedicated to the respective OWL profile. Of all of these reasoners, we used the latest version available on the official websites when we initiated the experiments (status January 2017).

To obtain a set of relevant OWL DL reasoners, we picked the four reasoners implemented in Java that performed best at the OWL DL realisation track at ORE'15. Listed according their performance at ORE'15, we used the following reasoners and versions:

- HermiT 1.3.8 [5],
- TrOWL 1.5 [21],
- Pellet 2.4.0 [19], and
- JFact 5.0.2.[4]

The reasoner TrOWL differs from the other reasoners in that for the OWL DL profile, it deliberately sacrifices completeness for performance, i.e. it does not guarantee to compute all instance relationships.

For the OWL EL realisation track of ORE'15, the four best performing reasoners were ELK [13], TrOWL, JFact, and Pellet. Since the latter two reasoners are optimised for more expressive ontology languages and perform significantly worse on OWL EL ontologies, we restricted our evaluation to

- ELK 0.5.0 [13] and
- TrOWL 1.5 [21].

ELK and TrOWL implement reasoning algorithms that are complete only for the OWL EL profile. ELK is the only reasoner that uses a dedicated multithreading implementation and thus implements parallel reasoning.

2.2 Setup of the Experiments

Our experiments are designed to investigate mainly the energy consumption and not so much the running times of the OWL reasoners. For this reason. we used a higher timeout of 10 minutes than the 3 minute timeout used at ORE'15, also to accommodate for the lower computation power of the CPU at lower frequencies. We ran each ABox realisation for the two OWL profiles for the respective set of ontologies and OWL reasoning systems. For each run, we logged the following information:

- histories of the different energy sensors,
- running time,
- CPU utilisation,
- number of CPU instructions, and
- cache references and cache misses per instruction.

[4] `http://jfact.sourceforge.net/`

The low-level information from the system was collected to gain insight on the causes for differing energy consumptions and reasoning times. Since all reasoners parsed the ontologies via the OWL API [11], they would use the same time for this task. We measured the time and energy used by parsing the ontologies once (for all) and excluded it in the measurements for the individual ABox realisation runs of the reasoners.

At ORE'15, the number of successful computations within the timeout was prioritised in the ranking, and computation time was only taken into account if the number of successful runs of all reasoners was the same for that ontology. Our focus here is on energy consumption of ontology reasoning, about which the number of timeouts hardly gives any insights. For this reason, and to allow for a meaningful comparison, we excluded those ontologies from the comparison which caused a timeout or an error for any reasoner at any frequency.[5] Excluding ontologies from the comparison that caused a timeout or an error for any reasoner at any frequency left us with 82 ontologies of 150 for the OWL DL profile and with 75 ontologies of 109 for the OWL EL profile. In the following, unless stated otherwise, all numbers refer to these sets of ontologies. For the interested reader, we provided corresponding numbers for the complete set of ontologies (including failed/incomplete computations), on our aforementioned webpage.

3 Observations

We report on our measurements carried out for the set of 82 OWL DL ontologies and 75 OWL EL ontologies for which ABox realisation was performed on the hardware and by the reasoning systems described in the last section. We start reporting on running times for ABox reasoning, because these values are important to put the measured values on energy consumption into perspective. We then turn our attention to power and energy consumption in Section 3.2, and discuss possible reasons for our observations in Section 3.3.

3.1 Running Times

The overall time required for ABox realisation is composed of the time needed for parsing the ontology and the time needed for the actual reasoning. To distinguish better the reasoning times of the different reasoners, we consider them separately.

Running times for parsing via the OWL API were measured for ontologies from both profiles. The obtained values are displayed in Table 2. The table relates to each frequency of the CPU the average, the standard deviation, and the median of running times for parsing the ontologies of the two profiles. As one could expect, the average running time, as well as the median of the running time, decreases as the frequency of the CPU increases. However, note that this increase is not proportional to the frequency of the CPU, as loading and parsing

[5] Note that this can lead to a different ranking as the one obtained at ORE'15.

Table 2. Execution time for parsing in seconds.

	OWL DL			OWL EL		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	91.05	35.20	97.61	143.10	56.00	186.41
0.5 GHz	53.15	14.60	74.73	104.92	25.50	143.92
1.0 GHz	61.84	8.00	82.81	92.50	14.00	156.55
1.5 GHz	45.70	6.00	61.03	84.58	10.50	150.08
2.0 GHz	37.60	5.00	50.06	76.30	9.00	137.34

Table 3. Reasoning time in seconds for the OWL DL profile.

	HermiT			JFact		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	84.85	205.16	11.58	115.82	282.06	14.32
0.5 GHz	34.45	82.59	4.75	48.79	116.42	6.05
1.0 GHz	18.20	43.33	2.61	27.42	64.35	3.41
1.5 GHz	12.98	30.55	1.92	20.72	47.73	2.66
2.0 GHz	10.33	24.06	1.58	17.40	39.36	2.26

	Pellet			TrOWL		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	31.16	64.12	11.43	4.98	7.76	2.97
0.5 GHz	13.74	30.79	4.74	2.12	3.41	1.26
1.0 GHz	7.80	18.00	2.63	1.20	1.98	0.69
1.5 GHz	5.92	13.85	1.97	0.91	1.56	0.51
2.0 GHz	5.02	11.91	1.63	0.77	1.33	0.43

requires frequent accesses to both hard drive and RAM. In general, the parsing times for OWL EL ontologies were higher than the ones for the OWL DL profile. This was to be expected, as the OWL EL ontologies were larger on average.

Running times for ABox realisation in the OWL DL profile were measured for the reasoners HermiT, JFact, Pellet, and TrOWL. The data obtained for running times is displayed in Table 3. Clearly, the running times of all reasoners decreased as the CPU frequencies increased. Here the difference between the reasoners in regard of the average and the standard deviation of reasoning times compared to the median is prominent. While HermiT and Pellet differ strongly on the average, their running times for the median are surprisingly close at all used CPU frequencies. TrOWL has the lowest running times, but recall that it implements an incomplete reasoning method and might miss inferences. The biggest decrease of running times consistently occurred when using a CPU frequency of 0.5 GHz instead of 0.2 GHz.

Running times for ABox realisation in the OWL EL profile were measured for the reasoners ELK and TrOWL and are displayed in Table 4. Here the difference

Table 4. Reasoning time in seconds for the OWL EL profile.

	ELK			TrOWL		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	19.77	22.79	11.51	30.51	114.45	2.99
0.5 GHz	8.36	10.19	4.85	13.13	51.51	1.24
1.0 GHz	4.78	6.37	2.70	7.42	29.92	0.75
1.5 GHz	3.74	5.34	2.03	5.76	24.39	0.51
2.0 GHz	3.23	4.66	1.68	4.78	20.70	0.47

between average and median is even stronger—while for ELK, the average is roughly the double of the median, for TrOWL, the average is almost an order of magnitude higher than its median.

When comparing the running times for parsing with those for computing ABox realisation, it is apparent that the overall running times are dominated by the parsing of the ontologies.

3.2 Energy and Power Consumption

Energy consumption of parsing for both profiles is displayed in Table 5. The energy consumption of both reasoners first decreased when the CPU frequency is increased up to 1.0 GHz. But the energy consumption increased again for frequencies higher than 1.0 GHz—unlike the running times. We will see a similar effect when regarding the energy consumption of ABox realisation, and give some explanations for this in Section 3.3. Similarly to the running times, the parsing of ontologies from the OWL EL profile consumed more energy, which can again be explained by the larger size of the ontologies.

Energy consumption for ABox realisation. In the OWL DL profile, there were 82 ontologies that could be realised by all reasoners in all frequency settings. The results of our energy measurements are shown in Table 6. In the OWL EL profile, there were 75 ontologies that could be realised by all the selected OWL EL reasoners in all frequencies. The measured values are displayed in Table 7. Note that there was generally a high standard deviation, and the average and median values often differed significantly, For example, while TrOWL had a higher average

Table 5. Energy consumption for parsing in Joule.

	OWL DL			OWL EL		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	7.31	6.33	2.47	50.98	11.27	94.58
0.5 GHz	3.97	3.18	1.53	37.48	6.56	85.10
1.0 GHz	3.07	2.45	1.43	37.18	5.22	86.87
1.5 GHz	3.60	2.50	2.11	56.53	6.89	128.38
2.0 GHz	5.59	4.84	3.72	106.77	10.82	250.05

Table 6. Energy consumption in Joule for reasoning in the OWL DL profile.

	HermiT			JFact		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	31.46	74.60	4.82	42.94	102.62	5.96
0.5 GHz	18.66	42.99	3.33	25.92	60.00	3.82
1.0 GHz	16.79	37.53	3.49	24.18	53.92	4.22
1.5 GHz	20.67	44.60	4.60	30.28	65.30	5.67
2.0 GHz	34.17	73.08	8.28	53.33	111.86	9.72

	Pellet			TrOWL		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	11.98	23.58	4.82	2.30	2.95	1.61
0.5 GHz	7.82	15.94	3.29	1.67	1.88	1.11
1.0 GHz	7.64	15.07	3.54	1.92	1.72	1.61
1.5 GHz	9.36	18.57	4.23	2.73	2.21	2.55
2.0 GHz	17.18	32.66	8.22	4.72	3.92	3.23

Table 7. Energy consumption in Joule for reasoning in the OWL EL profile.

	ELK			TrOWL		
Frequency	average	standrd. dev.	median	average	standrd. dev.	median
0.2 GHz	8.69	10.04	5.14	12.47	46.45	1.64
0.5 GHz	5.76	6.76	3.57	8.03	29.89	1.16
1.0 GHz	5.98	7.25	3.83	7.79	28.22	1.70
1.5 GHz	7.76	10.29	4.70	10.12	36.33	2.49
2.0 GHz	14.41	18.73	9.12	17.34	61.87	4.87

energy consumption in the OWL EL profile than than ELK, the median of the energy consumption of TrOWL was significantly lower than for ELK, indicating that TrOWL usually consumed less energy than ELK. In fact, if we consider all 90 ontologies for which ELK and TrOWL successfully performed realisation at 2.0 GHz, we see that for 70 of those, TrOWL consumed less energy than ELK. This indicates that choosing a reasoner specifically for a given ontology can provide significant savings in energy and reasoning time. Mobile applications that require simpler ontologies can therefore save significant energy and reasoning time by carefully selecting an appropriate reasoner for the ontology at hand.

Our measurements seem to confirm the hypothesis by Patton et al. that there is an almost linear relationship between energy consumption and reasoning time. Figure 1 plots for each reasoner the energy consumption against the running time for reasoning in the OWL DL ontologies at 1.0 GHz. For the other frequencies and the OWL EL profile, the picture looks similar.

However, when considering the average power consumption (i.e., energy per time) during realisation, one can note differences between reasoners as well as be-

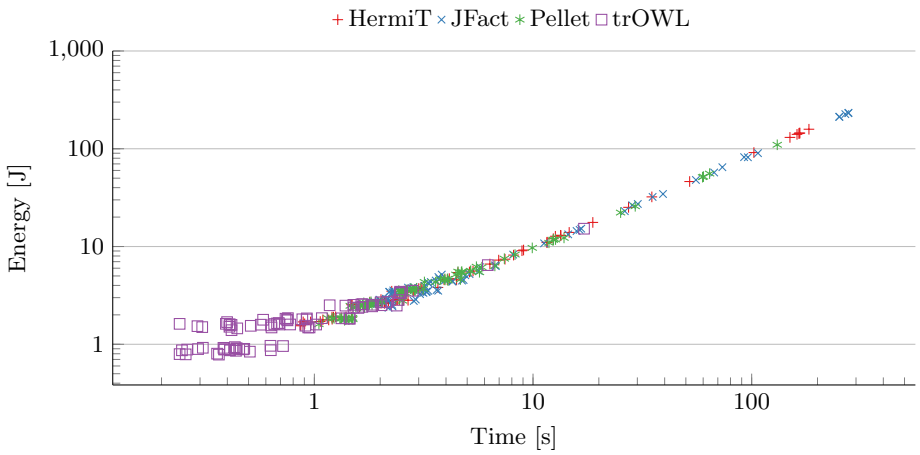

Fig. 1. Energy consumption in Joule in relation to reasoning time in seconds for the OWL DL profile (at CPU frequency of 1.0 GHz).

tween profiles. At 2.0 GHz, in the OWL DL profile, the lowest power consumption of the CPU was by TrOWL with 2.37 W, followed by Pellet with 2.52 W, with Hermit and JFact having the highest consumption of 2.62 W. In the OWL EL profile, the CPU consumed more power in general, with at 2.0 GHz, TrOWL consuming 2.60 W on average, and ELK 3.18 W. At the lower frequencies, though less power was consumed by the CPU, we obtain the same ranking among the reasoners. There were almost no differences in power consumption in the other components (memory, little cluster, GPU), which together accounted for between 0.26 and 0.28 W on average for all reasoners, profiles and CPU frequencies.

3.3 Impact of the CPU Frequency

We observed that the power consumption of the CPU grew exponentially with the selected frequency. While at 2.0 GHz, reasoning consumed 2.69 W on average for all reasoners and profiles, this value halves with each lower frequency used:

- 1.20 W at 1.5 GHz,
- 0.65 W at 1.0 GHz,
- 0.31 W at 0.5 GHz, and
- 0.16 W at 0.2 GHz.

In contrast, reasoning times were less than linearly affected by the CPU frequency, which is why the reasoning systems consumed significantly more energy at 2.0 GHz than at the lower frequencies.

The non-linear behaviour of the energy consumption in relation to the system frequency is to be expected for our system. The power that a system draws during execution is usually calculated using the following formula:

$$P = c \cdot f \cdot V^2,$$

where c is the capacitance of the system, f is the frequency and V is the core voltage. Since voltage has a quadratic influence on power, it is also the dominating factor for power efficiency, i.e., for power consumption in relation to work. The capacitance is a value specific to the system and very hard to determine for a specific instance of a system in practice. Many factors can change the capacitance of the system during runtime, because they disable unused parts of the chip. In general, the execution time decreases with increasing frequency, while power consumption increases.

Energy usage corresponds to execution time multiplied by average power used. To save energy, the decrease in execution time due to increased frequency has to outweigh the increase in power draw. Especially memory intensive workloads, such as those caused by reasoning systems, may benefit less from higher frequencies, because most of the cycles are spent waiting for memory, which is significantly slower than the CPU. While this waiting does not consume as much power as executing work, it still is not as efficient compared to a lower clocked chip that can run at a lower voltage.

We observed exactly this behaviour in our experiments. Interestingly, different reasoners benefited differently from the computation power available. Comparing reasoning times at 0.2 GHz with reasoning times at 2.0 GHz, we find for the OWL DL profile:

- HermiT taking 13.5% of the time,
- Pellet 14.5% of the time,
- TrOWL 14.8% of the time, and
- JFact 15.6% of the time

at the higher frequency. For performing realisation on the OWL EL ontologies,

- TrOWL took 14.5% of the time and
- ELK 15.1% of the time

at 2.0 GHz than it took at the frequency of 0.2 GHz. These values contrast the fact that the CPU is actually ten times faster at 2.0 GHz than it is at 0.2 GHz.

Based on this observation, it is no surprise that the lowest energy consumption occurred at the lower CPU frequencies, as can already be seen in Table 6 and Table 7. Figure 2 shows, for the OWL DL profile, at which frequencies how many ontologies caused the lowest energy consumption of the CPU. Figure 3 displays the same kind of information, but for the energy consumption of all measured components combined. Indeed, by using a lower CPU frequency, the energy consumption can be reduced significantly, with comparatively small increases in running time. For example, at 1.5 GHz, on average, realisation used only 57 percent of the energy used at the maximum frequency of 2.0 GHz, while taking only 119 percent of the time.

To get a clearer understanding of what happens at the lower frequencies, we first looked at the CPU utilisation, that is, the average number of active cores, during reasoning. We observed that higher CPU frequencies often caused a decrease in CPU utilisation, which means that on average, less cores were used

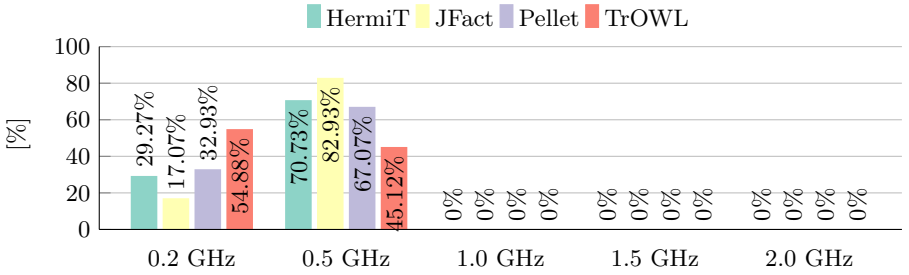

Fig. 2. Proportion of ontologies that caused the lowest energy consumption of the CPU. (E.g. with HermiT for almost 71% of the ontologies the CPU consumed least energy at 0.5 GHz.)

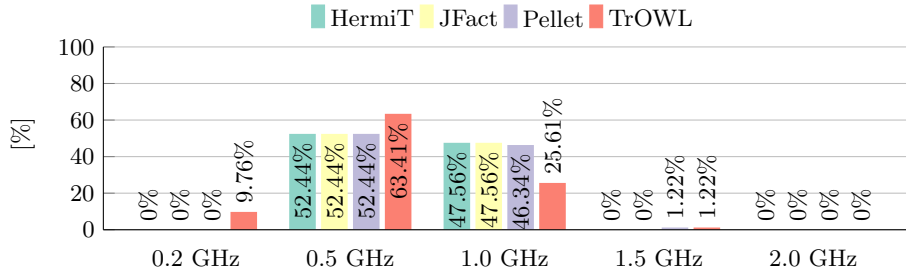

Fig. 3. Proportion of ontologies that caused the lowest energy consumption at a given frequency of all measured components. (E.g. with HermiT for almost 48% of the ontologies the measured components consumed least energy at 1.0 GHz.)

at the higher frequencies. For ELK, which had the highest CPU utilisation, the average CPU utilisation changed from 1.63 at 0.2 GHz to 1.42 at 2.0 GHz. For TrOWL, in both profiles, it even went from 1.21 at 0.2 GHz down to 0.85 at 2.0 GHz. For the other reasoners, the CPU utilisation was less affected by the CPU frequency, and generally lower than for ELK, which can be explained by the less dedicated use of multithreading. Note that using less cores means using less of the computation power available, and therefore has a negative effect on the reasoning time. The numbers also indicate that at higher frequencies, threads were often idle, and having a quadcore system was most useful when the CPU frequency was low.

The expected reason for why the reasoning systems did not make full use of the computation power available is that they are *memory-intensive* applications, and therefore slowed down by frequent accesses to the RAM. To confirm this hypothesis, and to understand better why different reasoners benefited differently from the CPU frequency, we examined the number of cache misses per CPU instruction measured during the experiments. A cache miss occurs when data accessed is not available in the cache, and has to be transferred from the (much

slower) RAM to the cache. Comparing the number of cache misses per CPU instruction, we found significant differences between the reasoning systems, which partly reflect the above observations. The lowest number of cache misses per CPU instruction was by HermiT (1.06%) and TrOWL (1.01% in both profiles), the highest was by JFact (1.28%) and ELK (1.34%).

Since ontology reasoning is memory-intensive by nature, higher numbers of cache misses cannot be avoided in general. However, the observed differences between the reasoning systems indicate that it might be possible to minimise the number of cache misses by a careful implementation with dedicated optimisations for cache-access. Optimisations like this have for instance already been applied fruitfully in the area of SAT-solvers [10,4]. Such optimisations to alleviate the memory bottle neck could potentially also improve performance of future OWL reasoning systems.

4 Conclusion

We evaluated energy consumption of computing ABox realisation in two OWL profiles with five state-of-the-art OWL reasoners that are implemented in Java. The experiments were run on a computer with built-in energy sensors, which allowed for more exact energy measurements than previously used methods for evaluating energy consumption of OWL reasoning.

Our empirical results confirm the hypothesis originally stated by Patton et al. that there is an almost linear relation between energy consumption and running time of OWL reasoning. However, different reasoners vary in the amount of energy they consume per second. Even though the energy consumption of reasoning is dominated by the energy consumption of the CPU, the performance of the reasoners is still affected significantly by the memory. The reason is that, as memory-intensive applications, reasoning systems regularly spend time waiting for data to be transferred from the RAM to the cache. As a result, they benefit less from higher CPU frequencies, while they consume significantly less energy at lower frequencies, and OWL reasoning turns out to be more energy-efficient on lower frequencies. The impact memory access have on reasoning times differs however from reasoning system to reasoning system. Determining (close to) optimal configurations for energy-efficiency and saving energy needs further investigations.

In the future, we would like to use our measurement framework to investigate the energy consumption of non-Java reasoners, and examine the energy consumption of query answering. We also see quite some potential in using our measurements to improve the results on predicting energy consumption of OWL reasoning obtained in [6]. In order to generate high quality prediction functions, one would need to relate the ontology metrics to the energy consumption of the OWL reasoners in regard of the hardware setup and the reasoner used.

References

1. Bail, S., Glimm, B., Jiménez-Ruiz, E., Matentzoglu, N., Parsia, B., Steigmiller, A. (eds.): Informal Proceedings of the 3rd International Workshop on OWL Reasoner Evaluation (ORE, CEUR Workshop Proceedings, vol. 1207. CEUR-WS.org (2014)
2. Bobed, C., Yus, R., Bobillo, F., Mena, E.: Semantic reasoning on mobile devices: Do androids dream of efficient reasoners? Web Semantics: Science, Services and Agents on the World Wide Web 35, 167–183 (2015)
3. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: Proceedings of the 13th international World Wide Web Conference (Alternate track papers & posters). pp. 74–83. ACM (2004)
4. Elffers, J., Johannsen, J., Lauria, M., Magnard, T., Nordström, J., Vinyals, M.: Trade-offs between time and memory in a tighter model of CDCL SAT solvers. In: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing - SAT. pp. 160–176. Springer (2016)
5. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
6. Guclu, I., Li, Y.F., Pan, J.Z., Kollingbaum, M.J.: Predicting energy consumption of ontology reasoning over mobile devices. In: International Semantic Web Conference. pp. 289–304. Springer (2016)
7. Guha, R.V., Brickley, D., Macbeth, S.: Schema.org: Evolution of structured data on the web. Communications of the ACM 59(2), 44–51 (2016)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2), 158–182 (2005)
9. Hähnel, M., Härtig, H.: Heterogeneity by the numbers: A study of the ODROID XU+E big.LITTLE platform. In: 6th Workshop on Power-Aware Computing and Systems (HotPower 14). USENIX Association (2014)
10. Hölldobler, S., Manthey, N., Saptawijaya, A.: Improving resource-unaware SAT solvers. Proc. LPAR'10 6397, 519–534 (2010)
11. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
12. Kazakov, Y., Klinov, P.: Experimenting with ELK reasoner on Android. In: Proc. ORE'13. pp. 68–74 (2013)
13. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK. Journal of Automated Reasoning 53(1), 1–61 (2014)
14. Motik, B., Nenov, Y., Piro, R., Horrocks, I.: Combining rewriting and incremental materialisation maintenance for datalog programs with equality. In: IJCAI. pp. 3127–3133 (2015)
15. Parsia, B., Matentzoglu, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL reasoner evaluation (ORE) 2015 competition report. Journal of Automated Reasoning pp. 1–28 (2015)
16. Patton, E.W., McGuinness, D.L.: A power consumption benchmark for reasoners on mobile devices. In: Proc. ISWC'14. pp. 409–424. Springer (2014)
17. Scioscia, F., Ruta, M., Loseto, G., Gramegna, F., Ieva, S., Pinto, A., Di Sciascio, E.: A mobile matchmaker for the ubiquitous semantic web. In: Mobile Computing and Wireless Networks: Concepts, Methodologies, Tools, and Applications, pp. 994–1017. IGI Global (2016)

18. Sertkaya, B.: The ELepHant reasoner system description. In: Proc. ORE'13. pp. 87–93 (2013)
19. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: science, services and agents on the World Wide Web 5(2), 51–53 (2007)
20. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: system description. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 27, 78–85 (2014)
21. Thomas, E., Pan, J., Ren, Y.: TrOWL: Tractable OWL 2 reasoning infrastructure. The Semantic Web: Research and Applications pp. 431–435 (2010)
22. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. Automated reasoning pp. 292–297 (2006)
23. Valincius, E., Nguyen, H.H., Pan, J.Z.: A power consumption benchmark framework for ontology reasoning on Android devices. In: Proc. ORE'15. pp. 80–86 (2015)
24. Yus, R., Pappachan, P.: Are apps going semantic? A systematic review of semantic mobile applications. In: MoDeST@ ISWC. pp. 2–13 (2015)
25. Zhou, Y., Cuenca Grau, B., Nenov, Y., Kaminski, M., Horrocks, I.: PAGOdA: Pay-as-you-go ontology query answering using a datalog reasoner. Journal of Artificial Intelligence Research 54, 309–367 (2015)