A Power Consumption Benchmark Framework for Ontology Reasoning on Android Devices

Edgaras Valincius, Hai H. Nguyen and Jeff Z. Pan

University of Aberdeen, UK

Abstract. Ontology reasoning, in particular query answering with Description Logics-based ontologies, is a power-consuming task, especially in mobile settings where such a resource is limited and shared by other processes. To be able to determine whether a reasoning task will be consuming significant amount of power, a benchmark framework will be needed for ontology designers and reasoner developers. In this paper, we report our work on a power consumption benchmark framework that can be used to evaluate and compare the level of battery consumption of different Description Logics reasoners on Android devices. To calculate the power consumption of the reasoners, we measured the current flow and the voltage of the integrated battery. We focus on Android-based devices and evaluate the framework using 3 popular DL reasoners.

1 Introduction and Background

The international Telecommunication Union revealed the ubiquity of mobiles, especially Android-based smartphones which takes around 77% of worldwide smartphone operating system market share [1]. To enable semantics data integration, some reasoning processes for Description Logics(DL)-based ontologies need to be performed. Although such reasoning tasks can be done on the cloud, with results being downloaded onto mobile devices afterwards, it has been indicated that local reasoning might produce results timely and they are robust to Internet connection unreliability [10], not to mention the ability to avoid privacy and security issues while uploading personal data to the cloud. Due to limited resources of mobile devices, the performance of reasoning tasks across different hardware platforms can vary and there has been not much research on how such reasoning tasks would affect the battery of mobile devices.

The closest work to ours is from Patton and McGuinness [6], which uses external power supply with fixed voltage instead of battery. The proposed framework system in this paper, however, is capable of measuring the power consumption of ontology reasoners from the operating system of the mobile devices, taking varying voltage of the battery into account. Our solution will thus make it easier for ontology designers and reasoner developer to test and evaluate their ontologies/reasoners in terms of power consumption efficiency, without having to connect mobile devices to external power. It is particularly important for smartphones with a built-in battery that cannot be taken out of the devices.

In our study, we used three ontology reasoners in this experiment: HermiT [7], Pellet [8], and AndroJena. In this work we investigate on how power is consumed on mobile-version of the above-mentioned reasoners on classification, instance retrieval (with and without inference) and query answering tasks. HermiT and Pellet can not be directly supported by Dalvik due to unsupportive Java classes. Yus et al. [11] provided the android-ported versions for them. AndroJena is an Android-ported version of the Jena reasoner that compatible with Dalvik. It should be noted that some other ontology reasoners can work on Android as well, such as ELK [5] and TrOWL [9]. A more comprehensive study will be part of our future work.

2 Our Approach

To calculate the overall power consumption, electrical energy over some time have to be measured. Our approach involves measuring the current flow (in Ampere) and the voltage of the battery (in Volt). Given these values change over time we can compute the amount of power consumption that has been used for a specific task. Note that we do not use existing android apps available on a market today to measure power consumption for two reasons: 1) With these apps, we would not be able to programmatically start and stop measuring the capacity of the battery when needed. 2) the accuracy of mentioned approach can be affected by the age of the battery due to the use of total capacity instead of the actual capacity.

To measure and calculate the battery information in mobile phones, hardware manufacturers have developed a range of chips, namely "fuel gauge chips". Most fuel gauges integrated into mobile phones are used for monitoring and just a few ones use their own sophisticated gauging algorithm to calculate the battery State-Of-Charge (SOC), available capacity, and other useful information such as current flow or voltage. In this experiment, we are not using gauging algorithms to calculate the battery SOC as this would depend on the types of fuel gauges used by phone manufactures. The calculation of power consumption is performed using the following equation:

$$Power\ Consumption = Current * Voltage * ErrorRate/Time$$

where $Current$ and $Voltage$ are retrieved using the properties in Android's `BatteryManager` class, `BATTERY PROPERTY CURRENT NOW` and `EXTRA VOLTAGE`, respectively. Note that since the value of $Voltage$ can be varied, it is necessary to update the changes by implementing a BroadcastReceiver which is triggered by a change of battery status. All values for this experiment had to be retrieved periodically to make sure the results are accurate and the power consumption can be calculated. To do so, a timer was set up to check and store the values necessary to calculate the power consumption for every second.

To determine the error rate of this or similar experiments, the actual capacity of the battery had to be compared with the total capacity of the battery. Total capacity was calculated by dividing the factory-determined mili-amphere-hours

(mAh) from the percentage (100) of the battery. Then the actual capacity per 1% of the battery was measured using the current sensor and both values compared.

Before Android Lollipop (starting with the Android API level 21), it was almost impossible for mobile-apps developer to access accurate information regarding battery usage from the "fuel gauge sensors" using a software approach. Since Android API level 21, information such as the State-of-charge (SOC), voltage, instantaneous battery current, and others provided by the fuel gauge sensors can be accessible for developers by a range of new properties and methods in BatteryManager class. Note that different smartphones will use different hardware, in particular different fuel gauge sensors, and hence Android API Level 21 has enabled a generic interface for apps developers to access battery information in an abstract and standardised way.

Although the power measuring techniques using integrated battery's current sensor was suggested around 5 years ago, they have been facing many compatibility issues. The main issue is that different manufacturers provide their devices with different type of current sensors. As a result, it has been lack of a standard way for Android-based apps developers to access battery information to calculate power consumption accurately. We believe our approach is future-proof, in that it is targeted for phones with Android API Level 21 and above while later generations of Android-based phones will need to support at least Android API Level 21. Given current rapid changes in the smartphone markets, it is reasonable to develop an approach that is compatible to current and later version of Android smartphones.

3 Evaluation

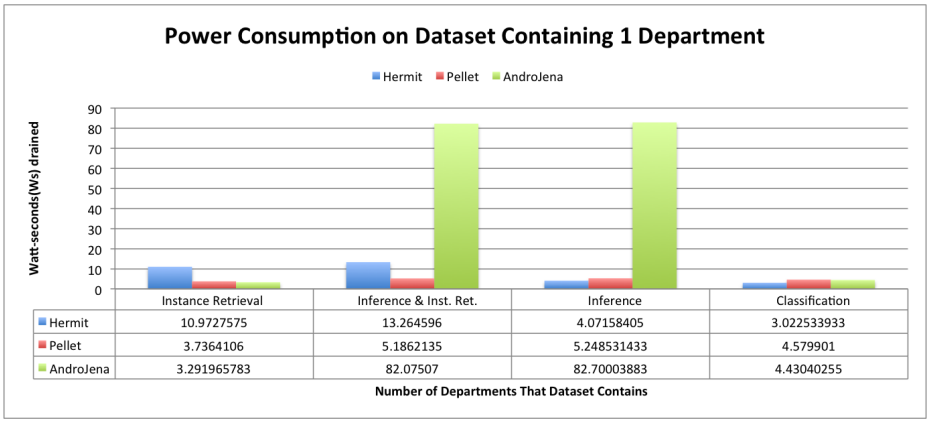
3.1 Experiment Setup

The experiment was done using a OnePlus One device, with 2.5GHz processor, 3GB RAM and 3100mAh battery. We used the datasets generated by the LUBM benchmark generator [2]. Four datasets of different sizes (in terms of the number of departments, ranging from 1, 5, 10 to 15) were generated. We chose to perform four different reasoning tasks for each reasoner: simple instance retrieval, complex query answering with the inference, which is the act or process of deriving logical conclusions from premises known or assumed to be true, just inference, and TBox classification. The last two tasks were to see if reasoning tasks with less or no involvement of individuals can affect the power consumption of different reasoners.

For each reasoner, there were 16 simulation tests performed in total (4 reasoning tasks and 4 datasets of different sizes). To prevent bias caused by the order of running the reasoners, we ran the 3 reasoners in all possible orders, e.g., 6 runs for 3 reasoners per test and got the average timing. In addition, the conditions of the experiment are kept the same for all the runs/reasoners, including current temperature and the level of the battery at the beginning of every test.

3.2 Results

After executing the test queries on the mobile reasoners, we compared the results returned by 3 reasoners to verify correctness. It was found that all reasoners produce the same answers to given queries except one case with the AndroJena reasoner. For classification reasoning tasks, AndroJena produced incomplete result. According to Jena's documentation [4], relationships between classes, when using Jena's OntModel, will be reported if they were asserted explicitly by the document. Otherwise reasoner flags plausible relationships by blank nodes. The class subsumptions, produced by other two reasoners, can be reflected in the output. This explains why the output of the SPARQL queries produced by Jena reasoner differs from ones produced by Pellet and Hermit.



	Instance Retrieval	Inference & Inst. Ret.	Inference	Classification
Hermit	10.9727575	13.264596	4.07158405	3.022533933
Pellet	3.7364106	5.1862135	5.248531433	4.579901
AndroJena	3.291965783	82.07507	82.70003883	4.43040255

Fig. 1. Comparison of reasoners, in respect to dataset containing 1 department.

Figure 1 (for dataset with 1 department) and Figure 2 (for dataset with 15 departments) both show that different reasoners perform differently with each reasoning task. Firstly, instance retrieval task is the suggestion of LUBM project and was also evaluated in [6]. The results that from our experiments and ones reported in [6] both suggested that HermiT reasoner consumes much higher power than its competitors. AndroJena and Pellet reasoners, on the other hand, were using power consumption similarly. Although, it was recorded the slightly higher power drain from the Pellet reasoner.

The second task, Inference and Instance Retrieval, was using Query 10 from LUBM's set of test query and differs from Query 6, 7, 8 and 9, which were used in [6], in that it only requires the (implicit) subClassOf relationship between GraduateStudent and Student.

Our Query 3 (for inference only) and Query 4 (for T-Box classification task) were to inspect the power consumption for reasoning services with less or no involvement of individuals. These were not evaluated by the work [6]. Interestingly, HermiT outperformed the other reasoners in these tasks.

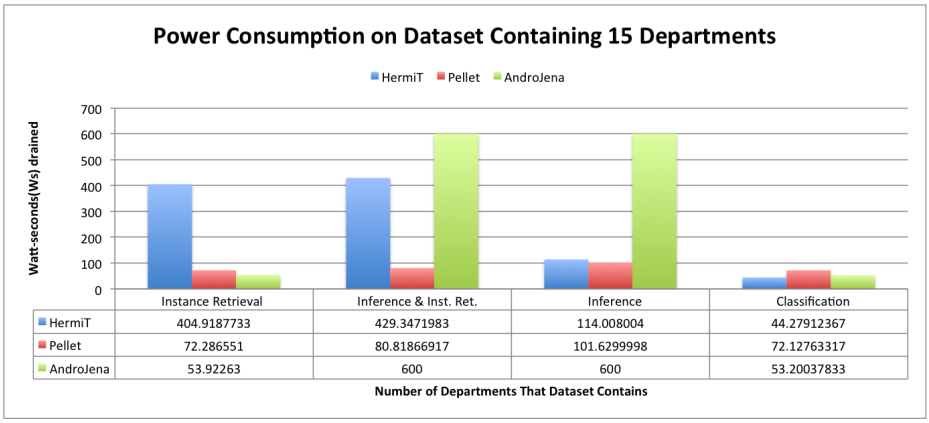


Fig. 2. Comparison of reasoners, in respect to dataset containing 15 departments.

In [6], the largest dataset that authors used contained only 4 departments. This paper, however, aims to test reasoner's scalability and hence the experiments were performed against a wide range of datasets, from ones with 1 department to ones with 15 departments. AndroJena cannot cope with large datasets so it had to be aborted while performing two reasoning tasks including logical inference, due to the boundaries that were set. Nevertheless, the rest of the reasoners performed well and produced the results.

It was found out that voltage is reducing when the state of charge is decreasing. Moreover, some tests took up to 51 minutes to complete their task. During these tests the phone battery drained about 21% of the capacity and the voltage decreased by 220mV. This happened because the permutation launches 3 reasoners, one after another a multiple times. Hence the results produced at the beginning of the simulation test were slightly different than the ones received later on during the same test, due to the decrease of voltage.

3.3 Discussion

This experiment proved that reasoners can be successfully compiled on the android mobile device. The results revealed that voltage is affecting the outcome of the reasoning task. It was calculated that during the most time consuming test, used in this experiment, voltage can differ by about 220mV. It also caused the power consumption values calculated using varying voltage differ.

The results also suggested that the effect of different reasoning tasks and the size of dataset to reasoners are important. AndroJena reasoner, for example, was found to require much more hardware resources than other reasoners when performing reasoning tasks involving inference and running on datasets containing more than 21 thousand triples. Therefore, the reasoner had to be manually suspended from occupying all the memory and getting into the outOfMemory error. HermiT revealed its advantages over other reasoners when performing complex

tasks. However, simple tasks such as instance retrieval were causing HermiT to take longer time and drain much more power. Pellet, on the other hand, was showing stable results during the whole experiment. It was in average the fastest and the least power consuming reasoner. Moreover, Pellet was also capable to use either OWL or Jena APIs, so it has higher compatibility over other reasoners, because it can support, for example, both SPARQL and SPARQL-DL queries.

4 Related Work

The most relevant work to this paper is Paton and McGuiness's work [6], which uses external power supply with fixed voltage instead of battery. Our setting is closer to the real world set up. Furthermore, their approach only works for smartphones with replaceable batteries but not for ones with built-in battery, which is in fact the case for current generation of smartphones. Note that our approach does not only work for Google Nexus devices but for any Android 5.0 and later devices (with Android API level 21 and later).

William Van Woensel et al. in [10] conducted the performance analysis (instead of power consumption) of mobile reasoners. The Benchmark data and rule-sets in their work were selected from [3]. The experiment was conducted on Android platform but the framework was claimed as a cross-platform solution.This is an advantage compared to our approach which depends on the use of Android API 21 to measure current flows and voltage. Given the dominance of Android-based smartphones in the market, we believe that our approach is a simpler yet efficient solution. Besides, unlike [10] our approach does not require the datasets to be converted into another format before running the benchmark.

The main idea in paper [12] is to use battery built-in voltage sensor, measure time period and the state of discharge (SOD), i.e., the percentage of the rated battery energy that has been discharged. To calculate power consumption using those values the variation of SOD based on voltage needs to be determined within the period of testing time. An issue with this approach is that the formula $P \times (t_1 - t_2) = E \times (SOD(V_1) - SOD(V_2))$ calculating the power consumption is using the total capacity of battery (E) that is decreasing due to ageing.

5 Conclusion

We have presented an approach to measuring power consumption for Android-based DL reasoners in a software level instead of the hardware levels as in previous works. By using new features provided in Android API Level 21, it is now possible to access battery information such as voltage, current flows in order to calculate the accurate power consumption of reasoning tasks. To evaluate the approach, we have conducted some initial experiments to compare the power consumption of different reasoners w.r.t. different reasoning tasks against datasets of different scales. Reasoners were compared in term of power consumption while running on mobile device and using the smart battery's sensors instead of external equipments.

Bibliography

[1] International Data Corporation. Smartphone os market share, q4 2014. http://www.idc.com/prodserv/smartphone-os-market-share.jsp, 2014. Accessed 29 March 2015.

[2] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3), 2005.

[3] IMPACT-AF. Integrated management program advancing community treatment of atrial fibrillation. http://impact-af.ca/. Accessed 29 March 2015.

[4] jena.apache.org. Ontmodel. https://jena.apache.org/documentation/javadoc/jena/com/hp/hpl/jena/ontology/OntModel.html. Accessed 29 March 2015.

[5] Yevgeny Kazakov and Pavel Klinov. Experimenting with elk reasoner on android. In *ORE2013*, 2013.

[6] Evan W. Patton and Deborah L. McGuinness. A Power Consumption Benchmark for Reasoners on Mobile Devices. In *ISWC2014*, 2014.

[7] Rob Shearer, Boris Motik, and Ian Horrocks. Hermit: A highly-efficient owl reasoner. In *OWLED*, volume 432, 2008.

[8] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51 – 53, 2007. Software Engineering and the Semantic Web.

[9] Edward Thomas, Jeff Z. Pan, and Yuan Ren. TrOWL: Tractable OWL 2 Reasoning Infrastructure. In *the Proc. of the Extended Semantic Web Conference (ESWC2010)*, 2010.

[10] William Van Woensel, Newres Al Haider, Ahmad Ahmad, and Syed Sibte Raza Abidi. A cross-platform benchmark framework for mobile semantic web reasoning engines. In *ISWC2014*, pages 389–408, 2014.

[11] Roberto Yus, Carlos Bobed, Guillermo Esteban, Fernando Bobillo, and Eduardo Mena. Android goes semantic: Dl reasoners on smartphones. Citeseer.

[12] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010.