A Power Consumption Benchmark for Reasoners on Mobile Devices

Evan W. Patton and Deborah L. McGuinness

Tetherless World Constellation
Rensselaer Polytechnic Institute
110 8th Street, Troy NY 12180 USA
{pattoe,dlm}@cs.rpi.edu
http://tw.rpi.edu/

Abstract. We introduce a new methodology for benchmarking the performance per watt of semantic web reasoners and rule engines on smartphones to provide developers with information critical for deploying semantic web tools on power-constrained devices. We validate our methodology by applying it to three well-known reasoners and rule engines answering queries on two ontologies with expressivities in RDFS and OWL DL. While this validation was conducted on smartphones running Google's Android operating system, our methodology is general and may be applied to different hardware platforms, reasoners, ontologies, and entire applications to determine performance relevant to power consumption. We discuss the implications of our findings for balancing tradeoffs of local computation versus communication costs for semantic technologies on mobile platforms, sensor networks, the Internet of Things, and other power-constrained environments.

Keywords: reasoner, rule engine, power, performance, mobile, OWL

1 Introduction

The vision of the Semantic Web established by Berners-Lee, Hendler, and Lassila [4] has brought us a web with a variety of ontologies, interlinked datasets [5], and efforts such as Schema.org to standardize terminology and encourage webmasters to publish structured, machine-readable web content. Since 2001, we have also seen the rise of the smartphone as a dominant platform for web access. Smartphone use continues to grow. The International Telecommunications Union estimates that around 90% of the world's population has access to cellular connectivity versus 44% with wired broadband to the home.[1] The ubiquity of the mobile phone offers an opportunity to build previously impossible, content-rich applications that benefit from semantic technologies.

One challenge that semantic technologies face when deployed on mobile platforms like smartphones is the amount of energy available for the device to compute and communicate with other semantic agents on the web. For example,

[1] http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf

the Google Nexus One, one of the first Android smartphones, had a single core processor operating at 1 GHz and 512 MB of RAM. Samsung's latest offering, the Galaxy S5, has a quad core, 2.5 GHz processor and 2 GB of RAM, more than a 8-fold increase in processing power and 4-fold increase in capacity in 5 years. However, the battery capacity of the two phones are 1400 mAh and 2800 mAh, respectively, indicating that battery technology is progressing more slowly than processing technology, in a time period during which the complexity of applications has increased. We therefore need tools to help developers identify essential queries and to select ontologies of the appropriate expressivities for local reasoning or identify when off-device computation is a more practical use of devices' limited energy reserves.

Context awareness [29, 14, 18], ubiquitous computing [12], and other user-centric applications will benefit greatly by reasoning about different streams driven by smartphone sensors. However, rich data sources can pose new challenges. Access control [28] and privacy have always been critical topics to consider and are even more-so given revelations on weaknesses in various cryptography libraries, such as the OpenSSL Heartbleed attack.[2] Therefore, one scenario to consider is one where personal data are kept and used locally to perform computation rather than sending those data to an untrusted party. Alternatively, we may build applications that selectively expose context or perform computation in a context-sensitive way without sharing all inputs of those computations. Consider a scenario where a wine recommendation agent (e.g. [19]) wants to make a recommendation to a user, but only if the recommendation meets dietary, medical, or other restrictions available in a privileged document such as an electronic medical record. If a health agent were available to manage computation on the medical record, the wine agent would provide the recommendation to it. The health agent then responds affirmatively if the supplied recommendation is reasonable given the content of the medical record. The wine agent then makes its recommendation without ever having direct access to user's health information.

Democratization of application programming, accomplished via tools such as the MIT AppInventor, allows anyone to build mobile applications using predefined components. A linked data extension to AppInventor [21] allows users to take advantage of SPARQL on mobile devices. However, users are given no feedback about how their applications might affect the performance or battery life of their (or others') phones, where resources are relatively scarce. Therefore, a new set of tools are required to aid the AppInventor community to incorporate new technologies like those provided by the semantic web.

We introduce a methodology that we believe is broadly reusable and is specifically motivated by these different scenarios to evaluate the performance of semantic web technologies relative to the amount of energy consumed during operation. In particular, we are focusing on varying the reasoning engine but varying the query engine, ontologies, and datasets are all possible with our approach. Ultimately, these metrics will provide developers a deeper insight into power consumption and enable next-generation applications of semantic technologies for

[2] http://heartbleed.com/

power constrained devices. The remainder of this paper is organized as follows: Section 2 compares our contributions with related work on reasoner benchmarking; Section 3 describes our novel hardware and software configuration used for collecting performance data; Sections 4 & 5 discuss the ontologies, reasoners, and queries used for evaluating reasoner performance; Section 6 presents performance per watt findings using our approach; Section 7 discusses some unanticipated results and implications for mobile semantic web agents; and, Section 8 presents conclusions and opportunities for future research.

2 Related Work

While reasoner benchmarking is not new, performing benchmarks relative to system power consumption and the amount of inferred statements has not been previously explored to the best of our knowledge. We therefore look at a number of existing benchmarks related to processing time and memory consumption as well as evaluations performed by reasoner authors. We also consider some power related work done in other areas of computer engineering and computer science.

The Lehigh University Benchmark (LUBM) [8] and the extended University Ontology Benchmark (UOBM) [15] use an ontology written in OWL that models university concepts such as classes, departments, professors, and students. The goal of these benchmarks is to evaluate the scalability of inference systems using a controlled ontology and a random instance set generated based on statistical knowledge learned from real world data. We make use of LUBM as a portion of our power benchmark for reasoners that support OWL DL, but evaluate reasoners for memory and power consumption in addition to execution time.

The JustBench Framework [2] was developed to evaluate the performance of reasoners using the justifications of their entailments rather than by analyzing the reasoner's performance on the entirety of an ontology or a random subset. One of its goals is to aid ontology engineers in debugging performance of reasoners and ontologies. Bail et al. also highlighted five techniques for improving reasoning behavior: *a*) introduce further training for knowledge engineers; *b*) reduce expressivity to more tractable logics, such as the OWL 2 profiles; *c*) generate approximations in more tractable logics, e.g. OWL 2 EL approximation of a DL ontology; *d*) apply fixed rules of thumb during ontology design; *e*) and, apply analytical tools, such as profilers and benchmarks . They found that evaluating justifications was a good first step to understanding how reasoners performed over ontologies, but that such means are unable to test performance when no entailment exists. While this research is promising, the short amount of time required to generate a single justification is too small to be accurately detected by our hardware, making this method of evaluation infeasible for our purposes.

Seitz and Schönfelder [20] present an OWL reasoner for embedded devices based on CLIPS. The target device ran at 400 MHz with 64 MB of RAM. They evaluated the performance and memory use of the reasoner on OWL 2 RL ontologies and identified issues with existing reasoning systems on devices with limited resources. They benchmarked their reasoner using LUBM with 25,000 individu-

als. They found that runtime on the resource constrained platform runtime was $O(n^2)$ with respect to triples in their knowledge base. Memory consumption was also observed as $O(n^2)$. Our benchmark includes a power metric to complement their means of measuring CPU time and memory consumption to provide a more holistic view of the resources a reasoner is consuming.

Henson, Thirunarayan, & Sheth [9] presented a novel bit vector algorithm for performing a subset of OWL inference on sensor data in a resource constrained device that showed performance on explanation and discrimination in a knowledge base $O(n)$ compared with Androjena, a variation of Jena designed to run on Google's Android operating system, at $O(n^3)$. Their evaluation introduced a benchmark that used completely orthogonal bipartite graph matching to generate worst-case complexity in each system. While their benchmark was aimed at resource constrained devices, it used a particularly limited subset of OWL that would not be expressive enough to cover the use cases discussed in Section 1.

Lim, Misra, and Mo [13] present an numeric analysis of energy efficiency of database query answering in a distributed environment. They approximate the amount of energy consumed by wireless transmission of data in a sensor platform using the theoretical power values for communications hardware set in the IEEE 802.11 and BlueTooth standards. Our work captures actual power use, which includes energy consumption of the CPU and real-world noise that may include energy consumed for checking for and retransmitting error packets, energy used responding to multicast/broadcast from other devices on the network, among others. We also provide figures for cellular radios not considered in that work.

The field of high-performance computing, in particular the area of fully-programmable gate array (FPGA) design, has looked at benchmarking custom hardware designs in a power-aware manner. Tinmaung et al. [26] present a power-aware mechanism for synthesizing FPGA logic. Jamieson et al. [10] present a method for benchmarking reconfigurable hardware architectures for applications in the mobile domain. While these works are oriented around power optimization, they are primarily interested in hardware design for specific highly parallelizable problems. Our benchmark focuses on establishing metrics for the semantic web community and in particular those looking to deploy semantic web technologies on off-the-shelf mobile hardware.

3 Power Consumption Measurement Methodology

Our methodology uses a physical device setup to capture power measurements. Benchmark evaluation is performed on the device, discussed in Section 3.2, and we also capture baseline measurements to provide an understanding of how reasoning performance compares to other basic operations, e.g. "screen on" for an extended period of time or data access via the device radios.

3.1 Power monitor setup

The goal of our work is to establish a metric for determining how much energy is consumed during reasoning tasks so developers can be aware of the effects of

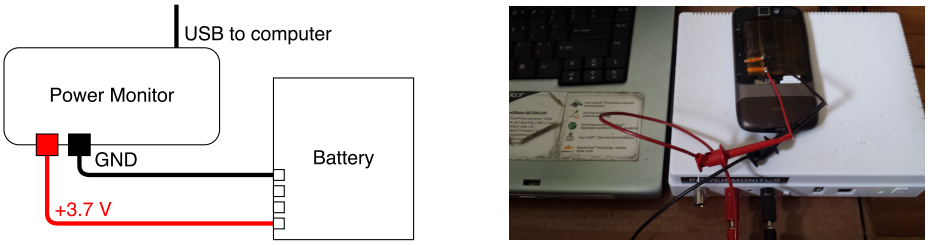

Fig. 1. Power bypass wiring for phone battery. Left, a conceptual diagram for bypassing the battery. Right, an example of the bypass implemented for the Google Nexus One.

semantic web tools on mobile device battery life. To this end, we need to provide a battery substitute that collects power samples during operation of the phone and use those data to compute the performance per watt of each reasoner.

We build our experimental framework around an external power supply sold by Monsoon Solutions[3] and use it as a battery substitute for Google Android smartphones. The power monitor captures voltage (2.1∼4.5 Volts) and current (max. 3.0 Amperes) drawn by the phone at a frequency of 5 kHz.[4] Following the company's instructions, we bypassed the battery of the smartphone as demonstrated in Figure 1. The leads connected to the battery are 85 mm for the positive lead and 77 mm for the ground, exclusive of the manufacturer's leads.[5] The power monitor is connected to a laptop running Windows XP SP3 to collect data. We use Cygwin as a scripting environment for interacting with the test setup. Our test script sends a start command to the device via WiFi after which both systems wait for a period of 10 seconds to allow the WiFi radio to enter its idle state so that power draw is minimal. The script then starts the power monitor software set to begin recording when power increases above 750 mW and to stop recording when power falls below 60 mW. These threshold values were determined by pilot trials where the changes in power consumption were manually noted by the researchers. The complete code is released under the GNU General Public License and made available on GitHub.[6]

3.2 Experimental Platform

We execute each test three times per trial, with thirty trials executed per query. The first execution is run in order to warm up Android's Dalvik virtual machine (VM) so that classes are linked appropriately and that this linking operation does not penalize any particular implementation. The second run is used to determine performance as well as measure the amount of energy consumed during

[3] http://www.msoon.com/LabEquipment/PowerMonitor/
[4] Power is voltage times amperage.
[5] Lead length affects the resistance between the power monitor and the phone, resulting in different power observations. We document the lengths here for reproducibility.
[6] https://github.com/ewpatton/muphro-test-framework

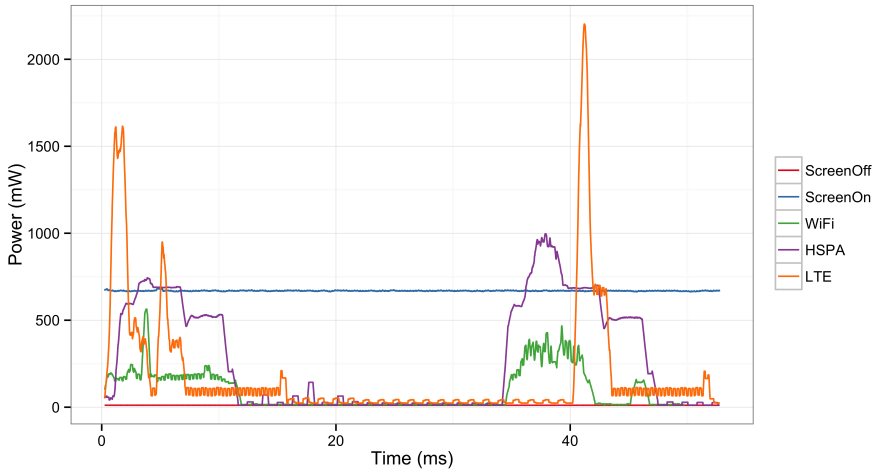

Fig. 2. Baseline power measurements for screen off (red), screen on (blue), data download over WiFi (green), HSPA+/3G (purple), and LTE/4G (orange). Note that the 4G/LTE radio consumes power at a rate of up to 2.5 times that of the 3G/HSPA+ radio, which in turn consumes power at a rate almost two to three times the WiFi radio. Thus, developers of distributed semantic web tools should carefully weigh the energy costs of WiFi vs 3G vs 4G for communication.

reasoning. The third run is performed using the Android Dalvik Debug Monitor Server (DDMS), which collects memory consumption usage whenever the Dalvik VM collects garbage. Note that we collect memory usage separately due to the fact that DDMS works over the Android Debug Bridge that communicates with the device either by Universal Serial Bus (USB) or by WiFi. We cannot use USB with the power monitor because the phone attempts to "charge" the power monitor, resulting in bad data, and using WiFi to communicate with the device during the tests would skew the power measurements. We therefore save the DDMS-oriented execution for WiFi when power is not being measured to prevent interfering with the primary focus of our evaluation. Thus, we are able to collect and monitor time to completion, memory requirements, and performance per watt estimations for our benchmark.

We execute our benchmark on Samsung's Galaxy S4 running the stock Android 4.3 operating system. The S4 has a quad-core 1.9 GHz CPU, 2 GB of RAM, and a 2600 mAh battery. We note that the Dalvik VM limits process memory to 128 MB, so any task that requires more memory than this will fail.

3.3 Baseline power measurements

Since we are concerned with evaluating semantic web reasoner performance, we desire the ability to control for other phone components that may skew results. We took baseline measurements to understand the effects of different subsystems

Table 1. Summary of tested ontologies

Ontology	Classes	Properties	Subclass	Subprops	Assertions
Schema.org	582	778	580	1	171
LUBM	43	32	36	5	103074

on power during common user actions or during normal system operation. We measured screen off with all radios off, screen on with all radios off, and tested two scenarios where we downloaded a 1 MB SPARQL/XML resultset from a remote server with an artificial latency of 30 seconds between the request and the start of the transfer over the WiFi, 3G, and 4G radios. Figure 2 charts the outcome of these results. Important points to note are that the 3G radio during transmission and broadcast can draw 1.56 Watt of power, which would cause the battery to deplete in just over 5 hours. The same download over WiFi requires significantly less power. The 4G radio draws even more power, requiring a peak of 4.00 W.[7] Overall, the amount of energy consumed by the WiFi, 3G, and 4G radios is 4.82 kJ, 14.4 kJ, and 9.77 kJ, respectively.

4 Ontology Selection

We choose to use a selection of different ontologies for evaluation to mimic different types of workloads that may be seen in real applications of the results of this work. The ontologies we selected are in RDFS and OWL DL. Table 1 provides a summary of the class and property hierarchies for the different ontologies used.

We use schema.org as it is widely known and gaining rapid adoption due to search engines backing it. The schema.org ontology is primarily composed of subclass axioms on primitive classes. Given the broad domain coverage of schema.org, we anticipate easy repurposing for driving user interfaces for interacting with linked data, such as those presented in our previous work on a mobile wine recommendation agent [19].

We include the Lehigh University Benchmark (LUBM), as it has emerged as a popular benchmark for evaluating the query answering runtime performance of reasoning engines using 14 queries.[8] Due to the memory limitations of the platforms in question, we evaluated LUBM on a subset of the full A-Box, limiting our queries to the first 4 files produced by the LUBM data generator (∼21 thousand triples). We intended to specify the knowledge base size explicitly, but the LUBM generator does not provide triple-level control over the output. The LUBM ontology is OWL DL, and thus we recognize that those reasoners that do not support OWL DL will potentially provide incomplete solutions.

[7] These peak times are taken directly from the raw data. The figure shown uses a moving average of one half second to smooth out excessive noise, which reduces the magnitude peak values in the visualization.

[8] http://swat.cse.lehigh.edu/projects/lubm/queries-sparql.txt

5 Reasoner Selection

We focus on Java-based reasoners to enable redeployment on mobile phones running the Android operating system.[9] We selected the reasoners based on their different algorithms (RETE networks, Tableaux, Hypertableaux) and supported expressivities in order to provide coverage of different techniques. Since we cannot evaluate all possible reasoners due to space constraints, most reasoners should be comparable to one of those identified here.

Jena Rules The Jena Semantic Web Framework [6] provides an implementation of the RETE algorithm [7] along with rule sets for RDFS, OWL Lite, and a subset of OWL Full. We used a variant of Apache Jena 2.10.0 with references to the Java Management Interfaces removed since they are unavailable on Android.

Pellet The Pellet reasoner [22] provides full support for OWL 2 DL, DL-safe SWRL, and uses the Tableaux algorithm. Due to this, its memory requirements often prohibit its use in memory-limited devices such as mobile phones.

HermiT The HermiT reasoner [16] uses a novel hypertableaux resolution algorithm to provide complete support for OWL 2. Due to the nature of this algorithm, it can perform more operations in polynomial time and reduced space, making it useful in more scenarios than the tableaux algorithm.

We intended to include the reasoners μ-OR [1], one of the first to be designed for OWL Lite inference on resource constrained devices, and COROR [24, 25], a Jena variant that loads rules selectively based on class expressions, but we were unable to obtain source code or binaries from the original authors.

6 Results

We hypothesize that the amount of energy used for reasoning will be linearly related to the amount of time required to perform the reasoning, a common assumption that needs to be validated. We also hypothesize that the mean power will be similar between reasoners as they are primarily CPU bound.

We define *effective performance* of a semantic software stack as

$$\rho_e(q) = \frac{1}{n} \sum_{i=1}^{n} \frac{results_q}{time_{q,i}}$$

where q is the query number, n is the number of trials run, $results_q$ is the number of query results found by the reasoner for query q, and $time_{q,i}$ is the execution

[9] We recognize that Google provides a native development toolchain for cross-compiling existing C/C++ libraries for Android, but leave an evaluation of reasoners written in these languages as future work.

Listing 1. SPARQL queries used for evaluating RDFS inference on the Schema.org ontology.

```
# query 1
SELECT ?cls (COUNT(?super) as ?supers) WHERE {
  ?cls rdfs:subClassOf ?super
} GROUP BY ?cls

# query 2
SELECT ?cls (COUNT(?property) as ?props) WHERE {
    ?cls rdfs:subClassOf schema:Organization .
    ?property schema:domainIncludes ?cls .
} GROUP BY ?cls

# query 3
SELECT ?property (COUNT(?cls) AS ?range) WHERE {
  ?property schema:rangeIncludes ?x .
  ?cls rdfs:subClassOf ?x .
} GROUP BY ?property
```

time for trial i. Mean power for a query is computed as:

$$\bar{P}(q) = \frac{\sum_{i=1}^{n} [\bar{P}_{q,i} time_{q,i}]}{\sum_{i=1}^{n} time_{q,i}}$$

where $\bar{P}_{q,i}$ is the mean power reported by the power monitor for trial i.

6.1 Schema.org

Schema.org provides an upper ontology in RDFS for describing content in microdata on the web. Backed by four of the world's major search engines, it is poised to dramatically change how structured data are published and consumed on the web. In a recent interview,[10] RV Guha of Google shared that over 5 million websites are now publishing microdata using Schema.org. Understanding how consuming the schema provided by Schema.org affects power consumption on a mobile device will enable developers to determine when the local consumption of Schema.org content is useful versus performing that consumption in the cloud. We consider that the classes and relations defined by Schema.org are useful for driving user interfaces for mobile linked data publishing applications and being able to query the schema efficiently is paramount to making this vision a reality. Three key relationships within the schema that we wish to evaluate

[10] http://semanticweb.com/schema-org-chat-googles-r-v-guha_b40607

Table 2. Summary of each reasoner on the three queries we designed for Schema.org's ontology. Left, the effective performance for each reasoner. Right, the performance per watt of each reasoner.

Query	1	2	3
HermiT	17.24	N/A	N/A
Jena	104.8	2.158	84.65

Query	1	2	3
HermiT	22.61	N/A	N/A
Jena	109.2	2.256	86.03

Table 3. Number of query answers for the subset of the 14 LUBM queries on which all reasoners returned results. Asterisks indicate those queries for which Jena's rule engine was unable to provide the complete result set found by HermiT and Pellet.

Query	1	3	4	5	6*	7*	8*	9*	14
HermiT	4	6	34	719	1682	67	1682	38	1319
Pellet	4	6	34	719	1682	67	1682	38	1319
Jena	4	6	34	719	1319	59	1319	15	1319

are subclass relationships (modeled using *rdfs:subClassOf*) and the *domainIncludes* and *rangeIncludes* properties. We provide three queries (Listing 1) to cover various common user interface needs: finding subtypes of the current type, identifying properties for which the current type is in the domain, and after selecting a property, finding valid range types, so an application can display relevant instances. We note that these queries are intended to stress how many and how quickly reasoners compute subclass and subproperty subsumptions.

Table 2 presents the reasoner evaluation results for the queries in Listing 1. We note that both HermiT and Pellet provide different challenges due to the fact that Schema.org's schema is not DL-compatible it lacks distinction between object and data properties. We were unable to execute queries 2 and 3 against HermiT using the OWL API due to this lack of delineation in the ontology and the API's inability to query at the level of rdf:Property. For all queries, Pellet used more than 128 MB of RAM, resulting in premature termination of the virtual machine so we exclude it from this analysis. A Mann-Whitney U test between HermiT and Jena's power measurements indicated no statistical difference ($U = 158$, $p = 0.8177$), thus the difference in performance per watt can be attributed entirely to the performance of each reasoner on the ontology.

6.2 Lehigh University Benchmark

Table 3 shows the number of query solutions found when all reasoners returned at least one answer. Due to their completeness, Pellet and Hermit generate the same answer set. Jena fails to return a complete answer set in 4 queries (noted with asterisks) and finds no results in four of five others.[11] Table 4 shows the effective performance for each reasoner in the different query conditions.

[11] No reasoner answered query 2 due to the subset of the dataset we are using. We assumed a uniform distribution in the types and quantity of triples generated by

Table 4. Performance (results/sec) for the different reasoners on LUBM queries, measured as query answers per second. Larger values indicate better performance.

Query	1	3	4	5	6*	7*	8*	9*	14
HermiT	0.0419	0.0635	0.365	7.466	42.48	0.631	16.94	0.378	33.37
Pellet	0.1893	0.2793	1.592	32.57	74.46	2.541	65.84	1.562	60.43
Jena	0.4511	0.6297	3.927	59.99	150.0	4.574	99.56	1.408	151.2

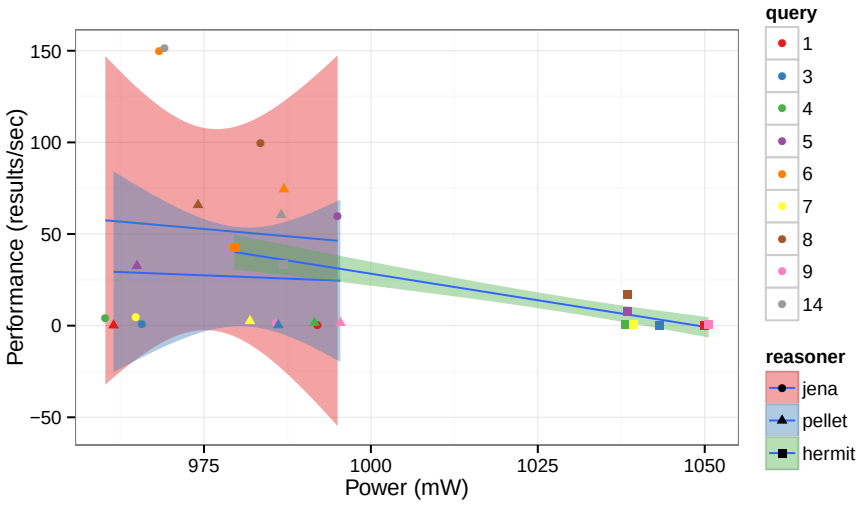

Fig. 3. Performance vs power for Jena, Pellet, and HermiT reasoners on LUBM queries. The shaded regions represent the standard error of the linear approximation of performance per watt.

A Kruskal-Wallis non-parametric one-way analysis of variance with mean power as a dependent variable and reasoner as the independent variable for each LUBM query presented in Table 3 showed significant difference in power drawn for all LUBM queries, with $p < 0.001$, with the exception of query 6 ($H = 3.08$, 2 d.f., $p = 0.214$). Pairwise Bonferroni-adjusted Mann-Whitney U comparisons on the differing queries indicated a statistically significant difference in the mean ranks of Hermit compared with Pellet and Jena. This finding is substantiated by Figure 3, which shows a tight clustering of HermiT's performance per watt measurements (except that for query 6) away from those of Jena and Pellet.

LUBM, but that assumption resulted in insufficient data to satisfy the patterns in query 2. Jena also found no results for queries 10-13.

7 Discussion

Ali et al. [1] and Henson et al. [9] both demonstrated scalable, low expressivity reasoners for resource constrained devices, but also highlighted that traditional inference systems on resource constrained devices are either very costly (as with Jena, HermiT) or impossible (as with Pellet). We affirmed their difficulty with deploying traditional reasoners on resource-constrained devices and have noted where appropriate when a test failed due to a limitation of the platform as compared with a limitation of the reasoning engine itself. As larger memory spaces become available on mobile platforms, we anticipate that these failures will become fewer.

One unanticipated interesting result was the difference in the device's power draw between HermiT's operation and that of Pellet and Jena. We looked at the experimental logs to determine possible causes and one significant factor that we have identified is memory consumption. HermiT's memory use is roughly double that of the other reasoners, causing the virtual machine to invoke the garbage collector more frequently. Since the short term garbage collector for Android runs concurrently on another thread, it is possible that the energy difference is the cost of the system engaging a second CPU core for garbage collection. This hypothesis is partially substantiated by one outlier (query 6) observed in Figure 3. When we compared the memory use on this query to others, it was roughly two thirds, more in line with memory consumption observed in Pellet. However, further experiments are required to isolate this effect.

Lastly, the combination of reasoning time and power are critical to understanding how to deploy semantic technologies on mobile platforms. In the case of Jena, the total amount of energy consumed ranged anywhere from 4.5 kJ to 8.9 kJ, indicating that it takes less energy to reason about some of these ontologies than it would to request the results from an external source via the 3G or 4G radios. However, HermiT and Pellet exhibit much higher energy consumption due to the longer running time of their algorithms, suggesting that it is always a better use of energy to offload OWL DL reasoning to an external resource when completeness is required by the application.

7.1 Experiences

Our original intention was to use the library OWL-BGP [11] as a SPARQL interface to reasoners designed for the OWL API (e.g. HermiT). However, due to a limitation in the Android Dalvik byte-code format, we were unable to successfully compile the library. In its place, we reimplemented the SPARQL queries used in each of the different test conditions as Java classes that would accept the OWL API's `OWLReasoner` class and generate a SPARQL result set by performing an appropriate set of operations on the reasoner.

Memory consumption was another challenge we addressed in multiple ways. In all tests, we observed that reasoning under Jena's OWL profile was impossible as out-of-memory errors were thrown prior to completion and thus we limited our tests to the RDFS profile. This evaluation performed on a Mid 2013 Macbook

Pro with a 2.8 GHz Intel Core i7 and 16 GB of RAM with a Java heap of 4 GB demonstrated that most reasoning operations would require at least 2.4 GB and in some instances, the Jena rules engine still exhausted the entire heap and failed to return results.

In order to execute the Jena reasoners, Hermit, and Pellet on the Android platform, we needed to make minor modifications to each library. While we will not explain details here, we note that all of the modified sources and binaries are made available under the appropriate source code license via our GitHub repository mentioned in Section 3.1.

The findings of this paper are subject to drastic changes in hardware design and power management that cannot be predicted from this work alone. For instance, advances in power management circuitry or power-efficient CPU architectures may have significant impact on power consumption that would improve the performance per watt observations that we have made (assuming such architectural changes do not negatively affect performance). Thus, we are making this software freely available under both the Apache 2.0 and GPL 3.0 licenses so that all of the tests performed herein can be repeated on alternative hardware as devices are made available.

8 Conclusions

We presented a novel, reusable, open-source methodology for evaluating semantic web reasoner performance on power-constrained hardware and evaluated three reasoners, Jena, Pellet, and HermiT, against standard benchmark ontologies and queries to establish the amount of power drawn during reasoning, information previously unknown in the field. The reusable methodology is one contribution and the use of the methodology to evaluate some best in class reasoners and standard benchmark ontologies is another contribution. We affirmed that single-threaded reasoners exhibit energy consumption linear in the amount of processing time and identified some discrepancies related to the effects of garbage collection on rate of energy consumption on an Android smartphone. We showed that incompleteness can greatly increase performance per watt if such incompleteness is acceptable in an application. Lastly, we demonstrated the effects of different smartphone features on power consumption to gain insights into the costs of communicating with external services to assist in making decisions about whether to perform processing on-device or off-device.

We found that for RDFS inference, the Jena RETE rule engine performed better than its complete OWL counterparts, a finding that is unsurprising given the overhead of the tableaux/hypertableaux algorithms. Another interesting finding is that while Jena was unable to perform OWL entailments on LUBM due to memory limitations, when executed with its RDFS transitivity rule set, it was able to answer some of the queries completely and others partially. This highlights how important it is for developers to identify essential queries and ontology expressivities to improve reasoning performance and reduce energy consumption.

Furthermore, we found a nearly linear relationship between energy required to complete an inference and the amount of computational time needed to perform the inference. This is due to the single-threaded nature of the design of the reasoners tested, but as our data show, there may be additional effects on power consumption outside of CPU runtime and we expect to see further differences as parallel and distributed techniques come to fruition.

8.1 Future Work

This benchmark lacks coverage for the complete OWL 2 semantics. Our examination focused on RDFS and OWL DL inference, primarily to exercise existing, well-known benchmarks used for evaluating reasoners in other contexts. A suite of different T-boxes and A-boxes providing coverage for all of the constructs in OWL 2 would allow us to more easily stress different parts of the reasoners and further research into parallelization techniques will enable us to gain a better understanding of the effects of communications on energy consumption.

We would also like to test parallel and distributed reasoning techniques such as those presented in [3, 27, 30, 17, 23]. Our findings in this work validate existing assumptions that power consumption for traditional RETE rule engines, tableaux, and hypertableaux reasoners will be linear in the compute time, but distributed algorithms will provide a richer execution strategy and power analysis that will benefit from the methodology we have outlined.

Due to the physical setup complexity and to ease adoption of power analysis of mobile semantic web applications, we intend to use a combination of factor analysis, clustering, and linear regression to build models that can consume power profiles of various devices, ontologies, and queries to generate expectations of energy consumption. This will eliminate the hardware barrier to entry by enabling an approximation of energy use. Furthermore, if the appropriate programming interfaces are available, we intend to build a reasoner that can take advantage of knowing the available energy remaining for computation to optimize its use of local versus remote resources.

Using the experiment metadata published by our framework, we intend to provide a portal of experimental conditions and evaluations with publicly accessible URIs. This portal will enable researchers to replicate results by providing a experiment URI to the benchmarking suite, which would download all the necessary resources and execute the experiment on a target device.

We plan to investigate more reasoners, such as those presented by [31], as well as those written in other languages, e.g. C or C++, to evaluate a wider breadth of technologies and to motivate extension of this work to other mobile platforms. In particular, we would like to provide a solution for phones without removable batteries, such as the Apple iPhone, so that practitioners can also assess performance characteristics of reasoners and ontologies for a greater breadth of devices. One possible approach may involve capturing battery level information from the operating system and running queries repeatedly and observing the reported drain. One challenge with this approach is the effective of battery degradation due to age.

Acknowledgements

Mr. Patton was funded by an NSF Graduate Research Fellowship and RPI. Prof. McGuinness and RPIs Tetherless World Constellation are supported in part by Fujitsu, Lockheed Martin, LGS, Microsoft Research, Qualcomm, in addition to sponsored research from DARPA, IARPA, NASA, NIST, NSF, and USGS. Our gratitude to Prof. Jim Hendler, Dr. Patrice Seyed, and Mr. Fuming Shih for their comments on early versions of this manuscript, and to the anonymous reviewers who provided detailed feedback to improve the manuscript.

References

1. Ali, S., Kiefer, S.: μOR - a micro OWL DL reasoner for ambient intelligent devices. In: Abdennadher, N., Petcu, D. (eds.) Proceedings of the 4th International Conference on Grid and Pervasive Computing. pp. 305–316 (2009)
2. Bail, S., Parsia, B., Sattler, U.: JustBench: A framework for OWL benchmarking. In: Proc. of the 9th International Semantic Web Conference. pp. 32–47 (2010)
3. Bergmann, F.W., Quantz, J.: Parallelizing description logics. In: Proceedings of the 19th Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence. pp. 137–148 (1995)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (May 2001)
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems 5(3) (2009)
6. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the semantic web recommendations. In: Proc. of the 13th Intl. World Wide Web conference on Alternate track papers & posters. pp. 74–83 (2004)
7. Forgy, C.: On the efficient implementation of production systems. Ph.D. thesis, Carnegie-Mellon University (1979)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmarking for OWL knowledge base systems. Web Semantics 3(2), 158–182 (2005)
9. Henson, C., Thirunarayan, K., Sheth, A.: An efficient bit vector approach to semantics-based machine perception in resource-constrained devices. In: Proc. 11th International Semantic Web Conference (2012)
10. Jamieson, P., Becker, T., Luk, W., Cheung, P.Y.K., Rissa, T., Pitkanen, T.: Benchmarking reconfigurable architectures in the mobile domain. In: IEEE Symposium on Field Programmable Custom Computing Machines. pp. 131–138 (2009)
11. Kollia, I., Glimm, B.: Optimizing SPARQL query answering over OWL ontologies. Journal of Artificial Intelligence Research 48, 253–303 (2013)
12. Lassila, O.: Using the semantic web in mobile and ubiquitous computing. In: Bramer, M., Terziyan, V. (eds.) Proceedings of the 1st IFIP WG12.5 Working Conference on Industrial Applications of Semantic Web. pp. 19–25 (2005)
13. Lim, L., Misra, A., Mo, T.: Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. Distributed and Parallel Databases 31(2), 321–351 (2013)
14. Lukowicz, P., Nanda, S., Narayanan, V., Albelson, H., McGuinness, D.L., Jordan, M.I.: Qualcomm context-awareness symposium sets research agenda for context-aware smartphones. IEEE Pervasive Computing 11(1), 76–79 (2012)

15. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Sure, Y., Domingue, J. (eds.) Proceedings of the 3rd European Semantic Web Conference. pp. 125–139 (June 2006)
16. Motik, B., Shearer, R., Horrocks, I.: Hypertableaux reasoning for description logics. Journal of Artificial Intelligence Research 36, 165–228 (October 2009)
17. Mutharaju, R., Maier, F., Hitzler, P.: A mapreduce algorithm for EL+. In: Proc. of the 23rd Intl. Workshop on Description Logics (DL2010). pp. 464–474 (2010)
18. Narayanan, V., McGuinness, D.L.: Towards situation aware smartphones via semantics and reasoning. In: Semantic Technology & Business Conference (2012)
19. Patton, E.W., McGuinness, D.L.: The mobile wine agent: Pairing wine with the social semantic web. In: Proc. of the 2nd Social Data on the Web workshop (2009)
20. Seitz, C., Schönfelder, R.: Rule-based OWL reasoning for specific embedded devices. In: Proc. 10th International Semantic Web Conference. pp. 237–252 (2011)
21. Shih, F., Seneviratne, O., Miao, D., Licardi, I., Kagal, L., Patton, E.W., Castillo, C., Meier, P.: Democratizing mobile app development for disaster management. In: Proceedings of the 2nd Workshop on Semantic Cities (2013)
22. Sirin, E., Parsia, B., Cuenca-Grau, B., 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 (June 2007)
23. Soma, R., Prasanna, V.: Parallel inferencing for OWL knowledge bases. In: 37th International Conference on Parallel Processing. pp. 75–82 (September 2008)
24. Tai, W., Brennan, R., Keeney, J., O'Sullivan, D.: An automatically composable OWL reasoner for resource constrained devices. In: Proceedings of the IEEE International Conference on Semantic Computing. pp. 495–502 (2009)
25. Tai, W., Keeney, J., O'Sullivan, D.: COROR: A composable rule-entailment owl reasoner for resource-constrained devices. In: Proc. of the 5th Intl. Symposium on RuleML. pp. 212–226 (2011)
26. Tinmaung, K.O., Howland, D., Tessier, R.: Power-aware FPGA logic synthesis using binary decision diagrams. In: Proc. of the 15th Intl. Symposium on Field Programmable Gate Arrays. pp. 148–155. New York, NY, USA (2007)
27. Urbani, J., van Harmelen, F., Schlobach, S., Bal, H.: QueryPIE: Backward reasoning for owl horst over very large knowledge bases. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) 10th International Semantic Web Conference. pp. 730–745. Springer-Verlag (2011)
28. Villata, S., Costabello, L., Delaforge, N., Gandon, F.: A social semantic web access control model. Journal on Data Semantics pp. 1–16 (2012)
29. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using OWL. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshop. IEEE (March 2004)
30. Williams, G.T., Weaver, J., Atre, M., Hendler, J.A.: Scalable reduction of large datasets to interesting subsets. Web Semantics: Science, Services and Agents on the World Wide Web 8(4), 365 – 373 (2010)
31. Yus, R., Bobed, C., Esteban, G., Bobillo, F., Mena, E.: Android goes semantic: DL reasoners on smartphones. In: 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013). pp. 46–52 (2013)