Thing Description Directory JSONPath Search Performance: Zion, TinyIoT ans WoT Hive Comparison

I. Introduction

Thing Description Directories are a crucial component of disseminating the Web of Things (WoT) since they allow users to store, fetch and filter Thing Descriptions (TD). One of the most exciting features of TDDs present in the W3C WoT Discovery Specification draft[1] is the ability to search TDs according to a known query language –i.e., jsonpath[2]. Due to IoT's very nature, we expected TDDs to host thousands of devices. Hence, scalability is a significant concern.

Given those raised concerns, we perform a comparative performance analysis between Zion[3], TinyIoT[4] and WoT Hive[5], those three being implementations of the W3C Web of Things Discovery. Our evaluation focused only on a feature defined in the standard: jsonpath search. We varied the size of the DB (number of TDs hosted) and performed several queries. Our results showcase that Zion increases its service time linearly and TinyIoT exponentially. Also, the WoT Hive Zion processing time is prohibiting high no matter the tested workload.

II. Performance Evaluation

Before each experiment, we populate the TDDs with TDs. We categorized TDs by their complexity; using the number of lines as the metric, we separated them as simple, medium, and complex. In each experiment, the distribution of TDs created in the TDDs was: 80% of simple TDs, 15% of medium TDs, and only 5% of complex TDs – mimicking a Pareto distribution. The TDs utilized in the experiments are publicly available in a GitHub repository[6].

We developed an open-source syntactical workload generator[7] that is responsible for performing queries to each TDD. Each experiment comprised a hundred calls sequential to the jsonpath search endpoint. For each call, we randomly select a query between:

- `$[?(@.properties.lightColor)]`: search TDs that have the `lightColor` property;
- `$[?(@.properties.lightColor @.properties.brightness)]`: search TDs that have both the `lightColor` and the brightness property;

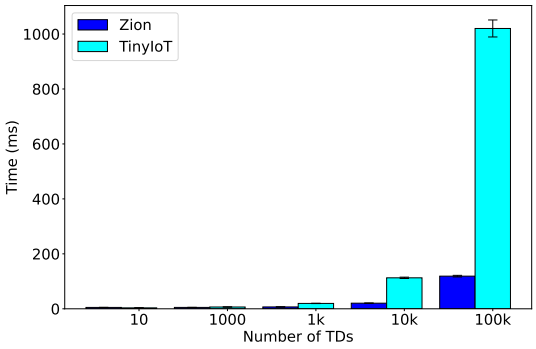

Fig. 1. Processing times for Zion and TinyIoT

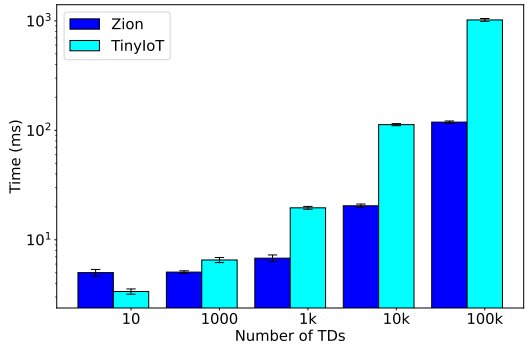

Fig. 2. Processing times for Zion and TinyIoT – y-axis logarithm scale

- `$[?(@.properties.sensorInformation)].properties..state`: search TDs with `state` as a sub-property of `sensorInformation` property.

Preliminary experiments unveiled that all three queries have similar processing times. In each replication, we reset the database and re-populated it. Additionally, we utilized Mersenne Twister[8] algorithm as the pseudo-number generator, and we seeded it manually with a prime number for each replication. There is a twofold reason to set replications seeds manually: (i) we guarantee fairness regarding the sequence of the pseudo-random number generated when comparing the different implementations – i.e., replication #12 of a given experiment with Zion was seeded as replication #12 of a given TinyIoT replication; (ii) We allow the performance analysis to

[1] https://w3c.github.io/wot-discovery
[2] https://tools.ietf.org/id/draft-goessner-dispatch-jsonpath-00.html
[3] https://github.com/vaimee/zion
[4] https://github.com/TinyIoT/thing-directory
[5] https://github.com/oeg-upm/wot-hive
[6] https://github.com/vaimee/tdd-workload-generator/tree/main/src/populate-db/examples-td
[7] https://github.com/vaimee/tdd-workload-generator
[8] http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/emt.html

be replicated by peer researchers and developers.

The workload generator and the analyzed TDD were deployed in the same machine, a personal computer with 12GB of RAM and an Intel Core i5-7200U CPU @ 2.50GHz × 4. Zion and TinyIoT were deployed as Docker containers. As for the workload, we varied the number of TDs in each experiment from 10, 100, 1,000, 10,000, and 100,000. Each experiment was replicated 30 times (except for 100k experiments, which were replicated ten times due to the long time – approximately 1 hour – to perform each replication), and asymptotic confidence intervals were computed at the level of 99%.

The results are depicted in Figure 1 and Figure 2 in logarithm scale. We removed WoT Hive from the graphs since it showcases an unfeasible high processing time in all tested workloads. In the lowest workload (10 TDs), the average processing time was 0.94s; with 100 TDs, the average was 1.85s, and with 1,000 TDs, the average was 8.61s. Further, during all Hive experiments, we face several error replies and denial of service from the server. Unveiling that Hive is not an application ready to be put into production or used in any environment that is not a prototype or highly experimental. The impact of the workload increase in Zion produces a linear increase in its processing time, as opposed to TinyIoT, which suffers an exponential growth in its processing time.