Semantic Data Provisioning and Reasoning for the Internet of Things

Altti Ilari Maarala
Department of Computer Science and Engineering
University of Oulu
Oulu, Finland
Email: ilari.maarala@ee.oulu.fi

Xiang Su
Department of Computer Science and Engineering
University of Oulu
Oulu, Finland
Email: xiang.su@ee.oulu.fi

Jukka Riekki
Department of Computer Science and Engineering
University of Oulu
Oulu, Finland
Email: jukka.riekki@ee.oulu.fi

***Abstract*—Semantic technologies could facilitate realizing features like interoperability and reasoning for Internet of Things (IoT). However, the dynamic and heterogeneous nature of IoT data, constrained resources, and real-time requirements set challenges for applying these technologies. In this paper, we study approaches for delivering semantic data from IoT nodes to distributed reasoning engines and reasoning over such data. We perform experiments to evaluate the scalability of these approaches and also study how reasoning is affected by different data aggregation strategies.**

I. Introduction

Internet of Things (IoT) contains a large amount of devices, i.e. IoT nodes that span diverse application domains such as logistics, industrial production, health care, and home automation. The data produced by the IoT nodes should be described in a commonly known and machine interpretable manner, as this would facilitate interoperability among a variety of applications and systems. Moreover, the data should be represented in a way that its meaning can be interpreted and shared efficiently, hence enabling reasoning of actionable knowledge.

Semantic Web technologies fulfil these requirements. Resource Description Framework (RDF)[1] enables integration of distributed semantic data and reasoning from it, and is capable for describing real-world phenomena. RDF databases, together with semantic query and reasoning techniques, provide sophisticated tools to manage semantic data. However, usually IoT nodes are resource constrained and loosely coupled, IoT data is highly dynamic and heterogeneous, the data should be accessed in real-time, and the variety of data formats can be large. Finally, the large amount of data amplifies these challenges.

Here, we study how the Semantic Web technologies fit in this equation. In our earlier work, we have studied lightweight data formats compatible with Semantic Web [1]. In this article, we focus on providing large amounts of sensor data, aggregating this data for distributed reasoning engines, and evaluating the latency of the whole process of delivering sensor data, reasoning actionable knowledge from the provided sensor data, and storing the inferred facts in a knowledge base.

[1]http://www.w3.org/TR/rdf11-primer/

We compare different approaches of delivering semantic data, that is, delivering data in different formats that can be supported by IoT sensors and aggregated to the actionable model for reasoning engines. Moreover, we compare alternative approaches for aggregating data and distributing reasoning. We focus on scalability, that is, on studying the different configurations' performance by delivering variable sized volumes of real sensor data with different amount of IoT nodes. We also study different data aggregation strategies' effect on the amount of inferred facts. These studies are the main contribution of this paper.

We do not provide a general architecture for IoT applications, but focus on a general process of data provision and reasoning. Protocols and data repository solutions also have a significant effect on latencies and resource usage. However, we assume that each message is delivered with the same protocols and along the same path, and we use one well-known RDF database to store RDF triples.

We present related work in Section II and details of our experiments and results in Section III. We conclude the article with discussion and propose future research directions in Section IV.

II. Related work

A. Data and representation

RDF is the most widely adopted data model on Semantic Web. It is based on description of relations between subjects, predicates, and objects (triple) linked in RDF graphs. New knowledge can be deduced by reasoning from RDF graphs with standard Web Ontology Language (OWL)[2] ontologies and rule-based languages.

Alternative representations for RDF include Notation 3 (N3)[3], Turtle[4], and N-Triples[5]. They are also based on the triple structure, but differ in expressive power. However, these representations are all designed for Web applications. Resource usage is critical for Internet of Things but was not emphasized when these languages were designed.

[2]http://www.w3.org/TR/owl2-overview/
[3]http://www.w3.org/TeamSubmission/N3
[4]http://www.w3.org/TR/turtle/
[5]http://www.w3.org/TR/n-triples/

978-1-4799-5154-3/14/$31.00 ©2014 IEEE

JSON for Linked Data (JSON-LD)[6] allows an RDF graph to be serialized in compact JSON format. Entity Notation (EN) [2] is another lightweight data format designed for resource-constrained devices and networks. It resembles the triple structure of RDF.

B. Reasoning and ontologies

Reasoning is a way to acquire new knowledge from RDF graphs and advanced knowledge structures, including ontologies. Ontologies organize information and represent knowledge in a formal way. Bikakis et al. [3] pointed out several benefits of rule-based reasoning, including simplicity, flexibility, formalism, expressive power, modularity, high-level abstraction, and integration with ontologies. Moreover, they suggested combining ontologies and rule-based techniques for reasoning in IoT [4].

Reasoning engines (i.e. a reasoner) can handle a comprehensive set of vocabularies and most semantic data formats, which enables integration and interpretation of heterogeneous IoT data, and thus improve interoperability. Moreover, OGC Sensor Web Enablement Domain Working Group [7] and Semantic Sensor Networks Incubator Group [8] have been fostering interoperability of sensor networks by standardization and providing high level ontologies. We focus on low level ontologies and predefined rules for reasoning in specific contexts.

Large amount of IoT data is provided by diverse heterogeneous sources and the context of data may change frequently, thus challenging reasoning on IoT. Distributed reasoning offers an approach to tackle this challenge [4]. Bikakis et al. [3] pointed out the computational, communication, scalability and availability advantages of distributed reasoning in large, dynamic environments. Prasad et al. [5] noted advantages such as concurrency, modularity, and robustness.

Distributed reasoning has been exploited in Multi-agent systems. Most Multi-agent systems are developed for specific environments and demands [6]. However, interconnected IoT environments consist of loosely coupled devices and services, where flexibility, integration and interoperability are preferred. Urbani et al. [7] propose distributed reasoning with MapReduce model for greater scalability. Cheptsov et al. suggest a general platform for distributed Web scale reasoning [8]. Similar approach is experimented in [9] with standalone setup on traffic prediction workflow. However, that work focuses on reasoning with static Web data from a data centric perspective. We focus on aggregation of dynamic data for physically distributed reasoner nodes in real-time fashion.

Also stream reasoning [10] focuses on reasoning with dynamic data. Le-Phuoc et al. [11] propose a framework for scalable querying of linked stream data. Streaming query engines perform reasoning with extended, SPARQL [9] based query languages over continuous data streams managed by Data Stream Management Systems (DSMS), whereas we concentrate on distributed rule-based reasoning. However, both distributed DSMS systems and streaming SPARQL query engines are new research topics. Evaluations of streaming query engines have been made based on relatively simple queries and static data [12][13].

[6]http://www.w3.org/TR/json-ld/

[7]http://www.opengeospatial.org/projects/groups/sensorwebdwg

[8]http://www.w3.org/2005/Incubator/ssn/

[9]http://www.w3.org/TR/sparql11-query/

C. RDF databases

RDF Databases [14] are capable to store, manage and provide semantic IoT data in RDF format, and integrate distributed RDF data. For IoT, features like SPARQL 1.1 standard queries, graph updates, inference from RDF graphs, federation, concurrency control, good scalability and performance would be preferred. Moreover, geospatial features such as GeoSPARQL querying is a definite advantage, as IoT data is often related to locations.

The current RDF databases supporting these features are: Virtuoso, Jena TDB, Sesame, Oracle Spatial and Graph, BigData and AllegroGraph. Many evaluations of RDF databases have been published [15][16], but distribution, federation, and concurrency issues have not attracted much attention. NoSQL databases [17] have been experimented for RDF data management as well. However, these databases are not specialized for RDF; thus, querying of RDF graphs, RDF schemas, and expressive ontologies are not directly supported.

III. Experiments

A. Setup

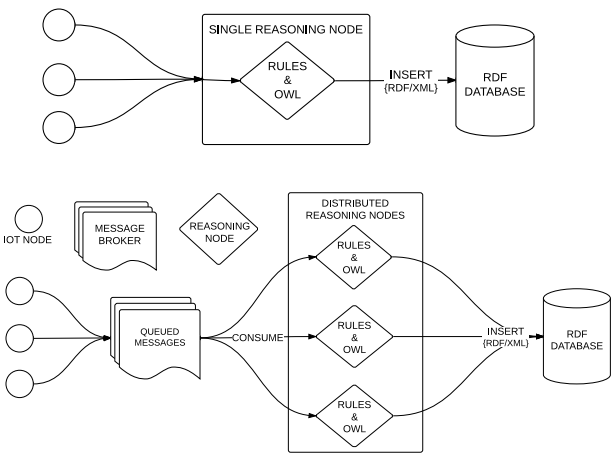

Fig. 1. System architectures for the experiments.

We performed experiments with both a system with a single reasoning node and a system with distributed reasoning nodes.

Figure 1 (top) presents the system with a single reasoning node. The operation starts by aggregating a sequence of messages from IoT nodes, in other words, by collecting a set of messages into an RDF data model. Then, a reasoning engine performs reasoning over this data, and the reasoned knowledge is stored into the RDF database.

Figure 1 (bottom) shows the system in which data and reasoning tasks are distributed physically to the reasoning nodes. Each reasoning node hosts several reasoner instances

TABLE I
IMPLEMENTED RULE SET

Rule	Clause
Low speed	IF Observation hasVelocity<25km/h THEN ns:LowSpeed
Jam	IF LowSpeed hasDuration>90s AND LowSpeed hasAverageSpeed<20km/h THEN ns:Jam
Long stop	IF Observation hasVelocity<3km/h THEN Stop AND Stop hasDuration>3min THEN ns:LongStop
Speeding	IF Observation hasVelocity>100km/h THEN ns:Speeding
Left turn	IF LowSpeed[1] hasDirection(a) AND LowSpeed[2] hasDirection(b) AND a=b-90deg OR a=b+270deg THEN ns:LeftTurn
Right turn	IF LowSpeed[1] hasDirection(a) AND LowSpeed[2] hasDirection(b) AND a=b+90deg OR b=a-270deg THEN ns:RightTurn
U-Turn	IF LowSpeed[1] hasDirection(a) AND LowSpeed[2] hasDirection(b) AND a=b-180deg OR b=a+180deg THEN ns:U-Turn
High acceleration	IF Observation[2] hasVelocity(v2) hasTmeStamp(t2) and (v2-v1)/(t2-t1)>2.5m/s^2 THEN ns:HighAcc
High deacceleration	IF Observation[2] hasVelocity(v2) hasTmeStamp(t2) and (v1-v2)/(t2-t1)>2.5m/s^2 THEN ns:HighDeacc
Crossing	IF LeftTurn hasLocation(x) AND RightTurn hasLocation(x) THEN ns:Crossing
Jam area	IF Jam[1] hasLocation(x) AND Jam[2] hasLocation(x) AND Jam[3] hasLocation(x) THEN ns:JamArea
Go-slow area	IF HighDeacc[1] hasLocation(x) AND HighDeacc[2] hasLocation(x) AND HighDeacc[3] hasLocation(x) THEN ns:GoSlowArea
U-Turn area	IF U-Turn[1] hasLocation(x) AND U-Turn[2] hasLocation(x) AND U-Turn[3] hasLocation(x) THEN ns:U-TurnArea
Stopping area	IF LongStop[1] hasLocation(x) AND LongStop[2] hasLocation(x) AND LongStop[3] hasLocation(x) THEN ns:StoppingArea

(as separate threads), hence serving multiple IoT nodes concurrently. The operation is otherwise similar with the single reasoning node, but IoT nodes produce data to a message broker and the data is consumed by several reasoning nodes. Moreover, data aggregation is divided between the message broker and the resoning nodes: the broker is responsible of content-based aggregation (i.e. of dispatching messages to reasoning nodes based on message content) and the nodes collect messages to an RDF data model until a predefined limit is reached (e.g. time interval or amount of messages).

These two systems were deployed on 11 servers within the same 1Gb/s sub-network. Each server machine has 16 to 32 cores and at least 64 GB of main memory. Each reasoning node was run on a separate server, the message broker and the database both had their own servers, and one server simulated the IoT nodes. The single reasoning node runs on a machine with 32 cores and 128 GB of main memory. The maximum amount of reasoner threads in the reasoning node equals to the amount of IoT nodes.

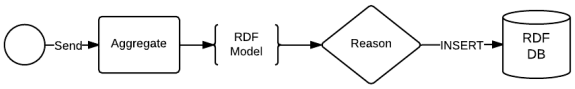

Fig. 2. Delivery process.

Figure 2 presents the implemented process. We create different scenarios from the data set by varying the data formats, amount of IoT nodes sending data to the system, and the number of messages sent by one node. Different data aggregation strategies can be realized by controlling the amount of aggregated messages, by selecting messages based on sources (i.e. IoT nodes) and content, and by controlling the interval messages are aggregated before triggering reasoning. We focus on scalability, thus, we measure the latency of the whole process instead of separate reasoning tasks. That is, the latency from delivering different volumes of data through aggregation and reasoning process to storing reasoned facts into the database is measured.

We selected the Jena[10] rule reasoner for these experiments, because it supports a comprehensive subset of OWL 2 language, several data formats, and a range of external reasoners. ActiveMQ[11] message broker was selected for dispatching messages as it is scalable, highly configurable, and fast enough. The two most feasible solutions for a backend RDF database are Virtuoso and Sesame. The selection turned in favour of Sesame, because of its integration and interoperability capabilities.

IoT nodes deliver data in different semantic data formats: RDF/XML, N3, JSON-LD, and EN. These representations have the required expressive power and enable inference [1]. However, Jena does not support EN format directly; thus, EN is transformed to RDF data model, which caused approximately 2% overhead. JSON-LD is supported through an integration module and other formats are built-in.

B. Dataset, scenario, and rules

We use real Global Positioning System (GPS) data collected by taxi cabs. The dataset includes 5 543 348 observations and 72 063 524 RDF triples, from 65 000 separate trajectories. The data consists of location coordinates represented as longitude and latitude, velocity, direction, timestamp, and vehicle identifier denoting the individual taxi cab.

Our scenario is about detecting different events and situations from GPS observations of taxi cabs. We design an ontology model and a set of rules for this scenario to reason facts from the GPS data, such as traffic jams, turns, speeding, stopping for a long time, strong acceleration and deceleration, and areas where taxis often stop for a while.

Table I shows the used rule set in pseudo code[12]. The rules process sequences of individual GPS observations dispatched

[10]http://jena.apache.org/documentation/inference/

[11]http://activemq.apache.org/

[12]Values between square brackets refer to sequence numbers and values between parentheses refer to actual property values. As our focus is on overall performance, the actual reasoning is not emphasized here. For example, in a real service, recognizing a traffic jam requires more than three samples and last four location based facts requires inference at long intervals from stored statements.

by IoT nodes. That is, we deduce from consecutive observations by comparing changes in direction and velocity. A sequence of observations is first aggregated and then the rules are processed. Messages are grouped in the message broker by vehicle identifier to guarantee that each aggregated set contains observations from a single vehicle. The rules are expressed in Jena rule format in forward chaining incremental manner. As incremental rules work, an inferred LowSpeed fact fires the rule that has LowSpeed fact in its clause. For example, a left turn can be thought to happen only after a taxi has driven at a relatively low speed.

The actual reasoning is performed on RDF data. The facts are expressed with RDF statements inferred from aggregated RDF model and OWL ontology. An inferred instance inherits all properties that the original RDF observation resource describes as RDF triples such as longitude, latitude, velocity, direction, timestamp and vehicle identifier.

C. Results

We perform three experiments on scalability. First, we compare single and multiple reasoning nodes with different data formats (Figure 3). Second, we evaluate distributed reasoning with different amounts of IoT nodes producing one million messages (Figure 4). Third, we evaluate distributed reasoning with different amounts of reasoning nodes (Figure 5). Moreover, we study the influence of different data aggregation strategies on the amount of inferred triples (Figure 6) and we compare also latency at different stages to analyze its cause (Figure 7).

In the first three experiments, we use size based aggregation strategy: for each task, 100 messages are aggregated into an RDF data model (this value is chosen based on the fourth experiment). Each reasoner thread processes one task at a time as determined by the aggregation strategy. A reasoner continues reasoning until all rules in Table I are processed over the aggregated data. Bandwidth usage of different data formats is proportional to payload sizes: N3 uses 74%, JSON-LD uses 54%, and EN uses 10% of the bandwidth that RDF/XML uses.

As seen from Figure 3 (top), the latency of the single reasoning node increases significantly when RDF/XML is used and the amount of IoT nodes exceeds 50. Moreover, the latency with the 50/1000 data set is higher than with 10/5000 data set. This increase can be explained by limited server resources and lack of load balancing. That is, with 10 IoT nodes load distribution is more uniform between reasoner threads than with 50 nodes. When the number of IoT nodes exceeds 50, high rate of context switching between threads in server processors leads to poor performance. Because the processing of RDF/XML requires more memory and computing resources than other formats, the server is not able to handle reasoning tasks in parallel in a reasonable time with 100 IoT nodes. With other formats, the latency starts to increase significantly when the amount of IoT nodes exceeds 50. It should be noted that these are scalability tests. Hence, we do not measure the processing time of a single reasoning task. For example, when 100 nodes send 10000 messages each (totaling to 12 million

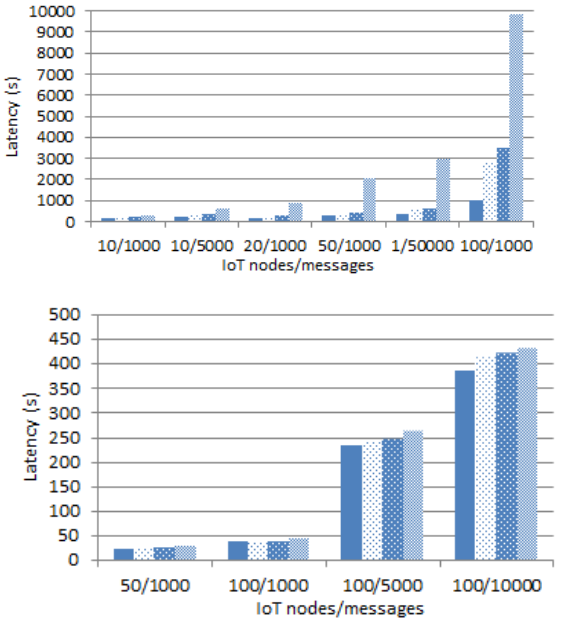

Fig. 3. Comparison between single (top) and multiple (bottom) reasoning nodes. Data formats in each set are in the following order: EN, JSON-LD, N3, and RDF/XML.

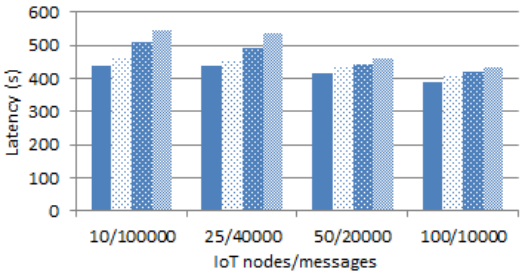

Fig. 4. Distributed reasoning with different IoT node configurations. Data formats in each set are in the following order: EN, JSON-LD, N3, and RDF/XML.

triples) and aggregation size is 100 messages, we measure the latency influenced by 10000 separate reasoning tasks.

As seen from Figure 3 (bottom), for multiple reasoning nodes, increasing the data set size causes quite linear increase in latencies. Moreover, increasing the amount of IoT nodes does not cause high increase on latencies as is the case for a single reasoning node. In contrast, it can be noted from the second experiment (Figure 4) that latency decreases when the total amount of messages is kept at one million and the amount of IoT nodes is increased from 10 to 100. This phenomenon can be explained by better bandwidth utilization and hardware resource utilization of the distributed reasoning nodes. EN outperforms also in this experiment, but latency has smaller variations between different formats because of the load balancing performed by the message broker, thus, memory and computing resources suffice for the task.

Figure 5 presents the measurements with different amounts of reasoning nodes and EN data format. The latency converges between six and eight nodes with the 100/1000 data set. With

the 100/10000 data set, the minimum is reached somewhere after eight nodes, which derives from better utilization of bandwidth and hardware resources with a greater amount of nodes.

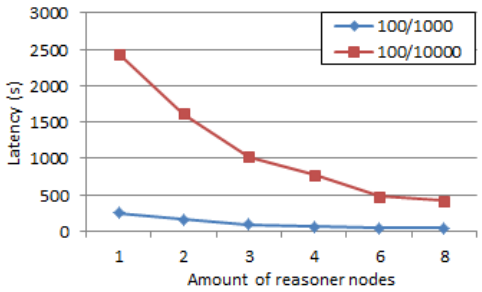

Fig. 5. Distributed reasoner with different reasoning node configurations.

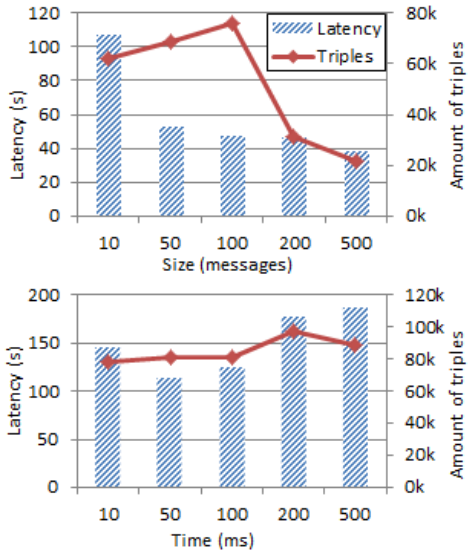

Fig. 6. Size (top) and time based (bottom) aggregation strategies.

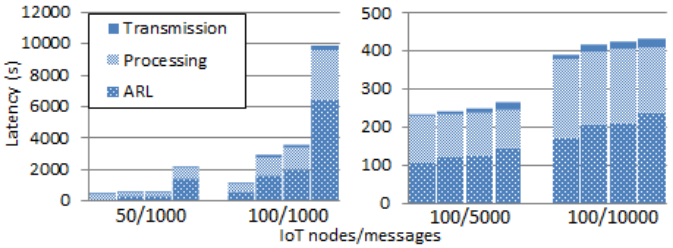

Fig. 7. Single node (left) and distributed (right) reasoning latencies in different stages. Data formats in each set are in the following order: EN, JSON-LD, N3, and RDF/XML.

We also studied the effect of size based and time based aggregation strategies to reasoning latencies and quality of reasoning (Figure 6), with the 100/1000 data set and EN data format. Here, (completion) size refers to the amount of aggregated messages and (completion) time to the time the aggregator collects messages.

Figure 6 (top) shows that the amount of inferred triples reaches its maximum value in the middle. Large completion size causes the amount of inferred triples to decrease. This is because data set contains also smaller sequence than chosen completion size, thus those sequences were not processed. That is why reasoning with size based strategy never reaches such high amount of inferred triples as time based strategy (Figure 6, bottom). Decreasing of completion size ($<$100) does not help, because inference chain is broken more often causing that all possible closures are not computed. Small completion size also increases latency, because more reasoner instances are started for reasoning tasks causing more context switching.

Figure 6 (bottom) shows that the number of inferred triples stays quite steady with different completion times. This is because reasoning is performed with all sequence sizes. High latency with 10 ms completion time derives from high amount of reasoning tasks as was with size based strategy. From 200 ms and onwards, the long waiting period starts to increase latency.

In the last experiment, we measure the Average Reasoning Latency (ARL) of all reasoning tasks per thread in a reasoning node. In the single node case (Figure 7, left), ARL is near 60% of total latency. In distributed case (Figure 7, right) ARL is near 50% of total latency. Transmission latency (including storing) is small, thus we can state, that total latency consist mostly of reasoning and message processing, including aggregation.

IV. Discussion

A large number of IoT nodes will be connected to the Internet and the amount of delivered data will increase greatly. This data can be used efficiently when interoperability is seamless and the data is easy to interpret [18]. Moreover, IoT applications require efficient reasoning methods to acquire knowledge in reasonable time. These problems can be tackled by describing the meaning of IoT data efficiently, combined with scalable, low latency and resource-conserving reasoning techniques.

We present in this paper our first experiments on distributed semantic data provisioning and reasoning within restricted contexts and real data. Our results show that current semantic technologies provide promising solutions for semantic data delivery and reasoning for real IoT systems. Distributed reasoners deduce facts from aggregated message sequences, that is, each reasoner operates on a small data sequence concurrently.

As expected, multiple reasoning nodes produce better scalability and smaller latency than a single reasoning node. For a single reasoning node, EN is the best data format to minimize latency and resource usage. When the amount of IoT nodes and messages is increased, RDF/XML shows a remarkable increase in latency compared with other data formats. However, data formats do not make a big difference with distributed reasoning nodes. EN is still slightly better than other alternatives. Moreover, increasing the number of messages introduces larger latency than increasing the number of IoT nodes. This means that the message broker handles load balancing between distributed reasoning nodes well. Large

data sets are handled successfully with distributed reasoning nodes. When 1 000 000 messages are sent, the latency with eight nodes is 25% of the latency with two nodes.

The advantage of message broker can be seen from the third experiment (Figure 5), where the latency with one node is only one-fourth of the latency experienced in the first experiment. For the single node, the lack of load balancing causes a large amount of concurrent thread executions and context switches in the server CPU (utilization was near 100%). Hence, the server runs out resources, which slows down all data processing.

The data aggregation strategy has a considerable effect on reasoning performance. Our aggregation strategy test shows that time based aggregation has more stable reasoning results when the amount of inferred triples is considered. In contrast, size based strategy can decrease reasoning latency, but some inferred triples can be lost if the aggregation size is not properly chosen. Balance between latency and quality requires choosing the right aggregation strategies and optimizing them carefully. The late stage of data aggregation decreased the performance. That is, the large amount of delivered messages causes processing overhead for the message broker and reasoning nodes, including message grouping and aggregation.

Overall performance could be improved by moving data aggregation and simple reasoning tasks to the IoT nodes, thus decreasing the amount of communication and the load of reasoning nodes. Though, this is only possible when data can be aggregated based on data providers. Sliding window aggregation techniques could improve the accuracy of the reasoning as inference chain would continue over sequences and all closures could be computed. Latency in reasoning stage could be improved by using some more computationally effective engines than Jena rule reasoner. Poor performance in single node case could be tackled by tuning the thread execution queue in the server (e.g. with message broker). Sesame RDF database did not show any significant degradation in performance.

Our scenario and rule set was quite simple. However, more complex scenarios can be implemented by using more diverse data, reasoning at several stages with forward-chained rules and delivering previous stages results as input for the next stages.

On open IoT environment, where connections are usually non persistent and network not reliable, decoupled IoT nodes are common. Hence, the system architecture should provide solutions for managing loose coupling and asynchronous messaging. Message-oriented publish/subscribe patterns are well known to handle loosely coupled message exchange. In addition, these solutions provide useful built-in tools for topic and content-based message routing, decomposition and aggregation; thus, enabling systems to adapt to dynamic environments, demands and contexts [19]. Hence, publish/subscribe could be used for context based aggregation.

In our future work, we will use more diverse data, reason in multiple contexts and with more detailed rules, and combine acquired knowledge between reasoners and with background knowledge. Federated RDF databases, state-of-the-art reasoning engines, and streaming SPARQL query engines would be interesting research topics as well.

ACKNOWLEDGMENT

This work was partly supported by TEKES as part of the Internet of Things program of DIGILE (Finnish Strategic Center for Science, Technology and Innovation in the field of ICT and digital business). We would thank Ekaterina Gilman and Susanna Pirttikangas for their advice for this work.

REFERENCES

[1] X. Su, J. Riekki, J. K. Nurminen, J. Nieminen, and M. Koskimies, "Adding semantics to internet of things," *Concurrency and Computation: Practice and Experience*, 2014. doi: 10.1002/cpe.3203

[2] X. Su, J. Riekki, and J. Haverinen, "Entity notation: enabling knowledge representations for resource-constrained sensors," *Personal and Ubiquitous Computing*, vol. 16, no. 7, pp. 819–834, 2012.

[3] A. Bikakis and G. Antoniou, "Rule-based contextual reasoning in ambient intelligence," in *Proc. International Symp. on Semantic Web Rules*, Washington, DC, 2010, pp. 74–88.

[4] A. Bikakis, T. Patkos, G. Antoniou, and D. Plexousakis, "A survey of semantics-based approaches for context reasoning in ambient intelligence," in *AmI Workshops*, Darmstadt, 2008, pp. 14–23.

[5] M. V. Nagendra Prasad, V. R. Lesser, and S. E. Lander, "Retrieval and reasoning in distributed case bases," *J. of Visual Communication and Image Representation*, vol. 7, no. 1, pp. 74–87, 1996.

[6] C. Badica, L. Braubach, and A. Paschke, "Rule-based distributed and agent systems," in *Proc. 5th International Symp. on Rules*, Barcelona, 2011, pp. 3–28.

[7] J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen, "Scalable distributed reasoning using mapreduce," in *Proc. 8th International Semantic Web Conf.*, Washington, DC, 2009, pp. 634–649.

[8] A. Cheptsov, M. Assel, G. Gallizo, I. Celino, D. Dell'Aglio, L. Bradesko, M. Witbrock, and E. Della Valle, "Large knowledge collider: a service-oriented platform for large-scale semantic reasoning," in *Proc. Int. Conf. on Web Intelligence, Mining and Semantics*, New York, NY, 2011, p. 41.

[9] E. Della Valle, I. Celino, D. Dell'Aglio, F. Steinke, R. Grothmann, and V. Tresp, "Semantic traffic-aware routing for the city of milano using the larkc platform," *IEEE Internet Comput.*, vol. 15, no. 6, pp. 15–23, 2011.

[10] A. Margara, J. Urbani, F. van Harmelen, and H. Bal, "Streaming the web: Reasoning over dynamic data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 25, pp. 24–44, 2014.

[11] D. Le-Phuoc, H. Q. Nguyen-Mau, J. X. Parreira, and M. Hauswirth, "A middleware framework for scalable management of linked streams," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 16, pp. 42–51, 2012.

[12] Y. Zhang, P. Duc, O. Corcho, and J. Calbimonte, "Srbench: A streaming rdf/sparql benchmark," in *Proc. 11th Int. Semantic Web Conf.*, Boston, MA, 2012, pp. 641–657.

[13] D. Le-Phuoc, M. Dao-Tran, M.-D. Pham, P. Boncz, T. Eiter, and M. Fink, "Linked stream data processing engines: Facts and figures," in *Proc. 11th Int. Semantic Web Conf.*, Boston, MA, 2012, pp. 300–312.

[14] D. Faye, O. Cure, and G. Blin, "A survey of rdf storage approaches," *ARIMA J.*, vol. 15, pp. 11–35, 2012.

[15] C. Bizer and A. Schultz, "The berlin sparql benchmark," *International J. On Semantic Web and Inform. Syst.*, vol. 5, no. 2, pp. 1–24, 2009.

[16] Y. Guo, Z. Pan, and J. Heflin, "Lubm: A benchmark for owl knowledge base systems," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 2, pp. 158–182, 2005.

[17] R. Hecht and S. Jablonski, "Nosql evaluation: A use case oriented survey," in *Proc. IEEE Int. Conf. on Cloud and Service Computing*, Washington, DC, 2011, pp. 336–341.

[18] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[19] E. Curry, "Message-oriented middleware," in *Middleware for Communications*, Q. H. Mahmoud, Ed. John Wiley and Sons, 2004, pp. 1–28.