Investigating Elastic Cloud Based Reasoning for the Semantic Web

Omer Dawelbeit, MSc, MBCS
School of Systems Engineering,
University of Reading
Introduction

- Part-time doctoral research at the School of Systems Engineering, University of Reading. Supervisor: Professor Rachel McCrindle.

- Full-time cloud computing professional at Appsbroker Consulting.
The current Web was designed for humans. Search is based on keyword occurrences. The Semantic Web [1] is an extension of the current Web that adds semantics to data.

Computers are able to answer queries based on the meaning of data. Semantic Web Data (knowledge) is represented in RDF or OWL.
A statement of knowledge is called a Triple:


Computers apply rule-based reasoning to infer new statements, for example:

**Input:** <John> <rdf:type> <Manager> and <Manager> <rdfs:subClassOf> <Person>

**Rule:** if a rdf:type B and B rdfs:subClassOf C then a rdf:type C

**Output:** <John> <rdf:type> <Person>

Computers infer knowledge using either forward reasoning or backward reasoning.
The Semantic Web enables computers to:
- Combine knowledge from different sources.
- Infer new implicit knowledge.
- Answer complex queries.

Find all Poets born in the UK before 1850 and wrote Epic Poem?

Book us a table for Saturday afternoon to eat Sushi in Covent Garden.
Problem Description

● The Semantic Web size is billions of statements.
● The data is highly skewed and inter-related.
● Reasoning and handling the data requires a great deal of computing power.
● Need efficient and scalable distributed/parallel algorithms.
● Need efficient data partitioning and assimilation between computing nodes.
● Need to cater for the storage of inferred knowledge.
Aim and Objectives

- Review the state of large scale distributed Semantic Web reasoning.
- Investigate how cloud computing features (Elasticity and Big data services) can address the current issues.
- Identify factors impacting the cost of cloud based Semantic Web reasoning.
- Develop a framework for elastic, cost aware cloud based reasoning (ECARF).
- Evaluate the framework through a cloud based prototype.
Embarrassingly Parallel [3] RDFS approach:
- Shows linear scalability.
- Generates large number of duplicates.
- Hard to extend to support richer logic.

MapReduce reasoning [4] on subset of OWL:
- Very high throughput.
- Difficult to store inferred data.
- Difficult to extend to support richer logic.

Peer to peer, Distributed Hash Tables [5,6]:
- Loosely coupled commodity computers.
- Suffers from load balancing issues.
Methodology

- Follow a design methodology.
- Literature review of current state of the art.
- Theoretical framework development (ECARF).
- Prototype development (AWS and/or GCP).
- Evaluation of ECARF in terms of cost, scalability and logic supported (RDFS, OWL 2 RL).
- Evaluation using a range of datasets:
  - DBpedia - Semantic extracts of Wikipedia.
  - SwetoDblp - Academic publications.
  - LUBM - Synthetic data benchmark.
Preliminary Framework

ECARF Formal Definition

ECARF is a 8-tuple, \((\Theta, \Sigma, T, \Gamma, \Delta, \rho, f, \sigma)\), where \(\Theta, \Sigma, T, \Gamma, \Delta\) are all finite sets and:

- \(\Theta\) is the set of coordinators,
- \(\Sigma\) is the set of processing nodes,
- \(T\) is the set of work items,
- \(\Gamma\) is the set of input to be processed,
- \(\Delta\) is the set of cloud services of mass cloud storage and big data service,
- \(\rho\) is the program embedded on a VM disk image and is capable of reading the node’s metadata,
- \(f: T \rightarrow \Sigma\), is the workload allocation function, and
- \(\sigma\) is the cost tracking function.
Preliminary Framework

- Coordinator receives requests for work.
- Partitions the work using bin-packing algorithm [7].
- Starts the required number of nodes, supplying tasks as metadata.
- Monitors nodes and terminate them once done.
- The overall resource usage is tracked.
Initial Results

(a) Runtime (min) vs. Num. of processing nodes
- Runtime decreases as the number of processing nodes increases.

(b) Load time (min) vs. Num. of processing nodes
- Load time decreases as the number of processing nodes increases.

(c) Cost ($) vs. Num. of processing nodes
- Cost increases as the number of processing nodes increases.

(d) Cost ($) vs. Runtime (min)
- Cost decreases as runtime decreases.
Conclusion

- Initial results are promising, closure of SwetoDblp in 24 minutes using only 4 nodes.
- Most of the processing of the massive data is handled by the cloud based big data services.
- Reported results outperforms DHT results.
- Results highlight the potential of the big data services for backward reasoning.
- To achieve Web-scale reasoning need to address the latency with data transfer.
- Need to also address issues with duplicate knowledge being inferred.
Thank you

Questions?

o.i.o.dawelbeit@pgr.reading.ac.uk
http://omerio.com
@omerio
References