LARGE-SCALE LINKED DATA PROCESSING: CLOUD COMPUTING TO THE RESCUE?

Michael Hausenblas[1], Robert Grossman[2], Andreas Harth[3] and Philippe Cudré-Mauroux[4]
[1]*Digital Enterprise Research Institute, NUI Galway, Ireland*
[2]*University of Chicago & Open Cloud Consortium, USA*
[3]*Institute AIFB, Karlsruhe Institute of Technology, Germany*
[4]*eXascale Infolab, Department of Informatics, University of Fribourg, Switzerland*
michael.hausenblas@deri.org, robert.grossman@uchicago.edu, harth@kit.edu, pcm@unifr.ch

Keywords: Linked Data

Abstract: Processing large volumes of Linked Data requires sophisticated methods and tools. In the recent years we have mainly focused on systems based on relational databases and bespoke systems for Linked Data processing. Cloud computing offerings such as SimpleDB or BigQuery, and cloud-enabled NoSQL systems including Cassandra or CouchDB as well as frameworks such as Hadoop offer appealing alternatives along with great promises concerning performance, scalability and elasticity. In this paper we state a number of Linked Data-specific requirements and review existing cloud computing offerings as well as NoSQL systems that may be used in a cloud computing setup, in terms of their applicability and usefulness for processing datasets on a large-scale.

1 Motivation

Although there are more and more data sources that are available over the Web, it is still relatively rare that a dataset is linked to another one. An important trend over the past decade has been the growing awareness of the importance of "light-weight" approaches to integrating data. The argument for lightweight approaches was best made in (Franklin et al., 2005). With these approaches the goal is to create loosely integrated "dataspaces" instead of completely integrated databases or distributed databases.

Early approaches for the lightweight integration of data using Web protocols include (Grossman and Mazzucco, 2002), which advocated using Universal Keys (columns of data identified by a Uniform Resource Identifier) and Web protocols to link columns of data in one table, identified by a Uniform Resource Identifier (URI), to columns in another table (identified by another Uniform Resource Identifier).

The most successful effort to date for light-weight Web data integration is based upon Tim Berners-Lee's Linked Data principles (Berners-Lee, 2006):

1. Use *URIs* to identify data elements.
2. Using *HTTP URIs* allows looking up a data elements identified through the URI.
3. When someone looks up a URI, provide useful information using standards, such as *RDF*.
4. Include *links to URIs in other datasets* to enable the discovery of more data elements.

In a nutshell, Linked Data is about applying the general architecture of the WWW (Jacobs and Walsh, 2004) to the task of sharing structured data on a global scale. Technically, Linked Data is about employing URIs, the Hypertext Transfer Protocol (HTTP) and the Resource Description Framework (RDF) to publish and access structured data on the Web and to connect related data that is distributed across multiple data sources.

The goal is to create a Web of Data broadly similar to the Web of Documents. Instead of using hyperlinks to connect documents, RDF links (in case of Linked Data) or Universal Keys (Grossman and Mazzucco, 2002) are used to connect data elements in different data sources. In this way, distributed data sources are easily integrated into a single global data space (Bizer et al., 2009; Bizer, 2009), enabling a new class of applications (Hausenblas, 2009) where the data integration effort is shared between data publisher, third-party services and data consumers.

Increasing numbers of data providers have begun to adopt Linked Data (Bizer et al., 2009). The most

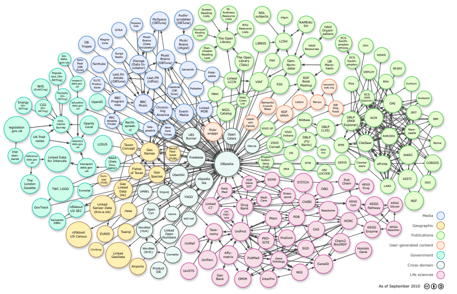

Figure 1: Topology of the Linked Open Data cloud with more than 200 datasets.

prominent example of the Linked Data principles applied to open data sources is the *Linked Open Data* (LOD) cloud[1] depicted in Figure 1. It currently contains over 200 datasets that contribute around 30+ billion RDF triples and over 400 million cross-data set links (Bizer et al., 2010). In the visualisation of the LOD cloud in Figure 1, each node represents a distinct dataset and arcs indicate the existence of links between data elements in the two data sets.

Another project that will be linking large amounts of data is the Open Science Data Cloud (OSDC) (Grossman et al., 2010). By the end of 2012, the OSDC plans to make available approximately 1 PB of scientific data across a variety of scientific disciplines, and then assign digital IDs and links to this data.

With projects like these, given the size of the data, *selecting an appropriate data management infrastructure for loading, managing and processing the data is critical.* In this paper, we consider some of the requirements, options, and issues that arise.

We illustrate some of the problems that arise with an example taken from our work in the LOD-Around-The-Clock (LATC) Support Action[2] and the PlanetData Network of Excellence[3]. Linked Data supports live querying (Umbrich et al., 2011); however, in typical usage scenarios, the time needed to load the remote data often makes it more practical to load the respective datasets into a central data store for processing. Take for example the case where data from DBpedia, Geonames, as well as statistical data from `data.gov.uk`—totalling some billion triples—are used in an application: one is likely better off to process this volume in place, rather than using the data live.

Currently, there exist three options to process Linked Data in a central location:

[1] `http://lod-cloud.net/`

[2] `http://latc-project.eu/`

[3] `http://planet-data.eu/`

- *Dedicated triple stores*[4], such as 4store, AllegroGraph, BigData, BigOWLIM, Virtuoso or YARS2, as well as triple stores in the cloud like the Talis platform[5] or Dydra[6].
- *Relational databases* along with i) built-in RDF support, for example Oracle 11g[7], or ii) RDB2RDF mappings, currently under W3C standardisation[8].
- *NoSQL* offerings.

In this paper, we focus on the last category. Sometimes the term *cloud computing* is used instead of *NoSQL*, since in practice, many cloud computing offerings (Armbrust et al., 2010) are NoSQL solutions and many NoSQL solutions are cloud-enabled. A good starting point for Linked Data processing with the cloud is Arto Bendiken's write-up on "How RDF Databases Differ from Other NoSQL Solutions"[9] as well as Sandro Hawke's "RDF meets NoSQL"[10].

We emphasise that this paper is concerned with the question to what extent NoSQL systems can be used to process Linked Data in a cloud computing setup. The more general question of the appropriate data management infrastructure for distributed data or science clouds is out of scope for the work at hand.

The remainder of the paper is structured as follows: in Section 2 we state requirements concerning Linked Data processing, then, in Section 3 we review systems in terms of their Linked Data processing capabilities and in Section 4 we compare the systems against the requirements stated earlier. Eventually, in Section 5, we conclude our survey and report on next steps.

2 Requirements

Based on the interactions with researchers and practitioners in the realm of various projects as well as drawing from own experience in the field of Linked Data processing in the past four years, we have identified a number of requirements in addition to the "hard" requirements *performance*, *throughput*, *scal-*

[4] `http://www.w3.org/wiki/LargeTripleStores`

[5] `http://www.talis.com/platform/`

[6] `http://dydra.com/`

[7] `http://www.oracle.com/technetwork/database/options/semantic-tech/`

[8] `http://www.w3.org/2001/sw/rdb2rdf/`

[9] `http://blog.datagraph.org/2010/04/rdf-nosql-diff`

[10] `http://decentralyze.com/2010/03/09/rdf-meets-nosql/`

ability and *elasticity* (Armbrust et al., 2010; Cooper et al., 2010; Dory et al., 2011):

URIs as Identifiers
Supporting URIs as primary keys. The first Linked Data principle (see above) suggests the use of URIs to name entities. The processing platforms must thus be able to use URIs as identifiers natively, or to map URIs to their own internal identifiers efficiently.

RDF Support
Importing and Exporting RDF datasets. The ability to import RDF data both in small chunks (for example, as RDF/XML files) and as large data dumps (for example, bulk loading of large N-Triples or N-Quads files) is essential, since in the LOD cloud data is typically exposed in an RDF serialisation.

Interface
The ability to serve information as HTTP, which is often used when browsing Linked Data sets and dereferencing URIs to get additional information about arbitrary data elements.

Partitioning
Support for logical partitions, for example via Named Graphs[11] (also sometimes referred to as "context") for managing the dataspace.

Update
Providing update facilities, for example via HTTP PUT/POST over an HTTP interface or via SPARQL update[12] to perform arbitrary inserts and updates on data.

Indexing
Support for a modular indexing sub-system that allows to use specialised indexing services, such as text indexing via the Semantic Information Retrieval Engine (SIREn)[13]. The ability to offer those indexes is important in many LOD applications, for example to support full-text search interfaces and co-reference services such as SameAs.org[14].

Inferencing
Support for reasoning, for example taking into account equivalence statements via `owl:sameAs` axioms as well as other logical constructs provided by RDFS and OWL (e.g., `subclasses`, `transitive properties`, etc.).

[11] `http://www.w3.org/2011/rdf-wg/wiki/TF-Graphs`
[12] `http://www.w3.org/TR/sparql11-update/`
[13] `http://siren.sindice.com/`
[14] `http://sameas.org/`

Rich Data Processing
Providing query facilities which can range, depending on the functionality and scalability requirements of the application, from simple Linked Data look-ups over triple pattern look-ups to conjunctive queries and finally full-fledged general SPARQL query[15] facilities (joins, aggregates, property paths, etc.) in order to perform rich, structured queries.

Efficient Graph Processing
Efficient support for path or transitive closure queries. As entities are interlinked on the LOD cloud, it is often necessary to follow series of links iteratively to resolve a given query. Such graph queries are very common in our context, however can be extremely expensive on some platforms, for example, on relational platforms where they often boil down to multi-joins.

In Section 4 we discuss the above listed, Linked Data-specific requirements along with the findings of this paper.

3 Data processing systems review

Following Cattel's terminology (Cattell, 2011) we understand data stores to include cloud computing as well as NoSQL offerings. In the following, we review several data stores, in alphabetic order, in terms of their capabilities to perform large-scale processing of Linked Data processing perspective. A number of runner-ups are discussed as well in the following.

3.1 BigQuery

BigQuery[16] is a cloud computing offering by Google, supposed to complement MapReduce jobs in terms of interactive query processing, introduced together with Google Storage and the Google Prediction API in early 2010. In late 2010 we looked into utilising BigQuery for Linked Data processing by developing the *BigQuery Endpoint* (Hausenblas, 2010a), an application deployed on Google App Engine that allows to load RDF/N-Triples content into Google Storage as well as exposing an endpoint allowing to query the data.

3.2 Cassandra

Apache Cassandra is a second-generation distributed database, bringing together Dynamo's (DeCandia

[15] `http://www.w3.org/TR/sparql11-query/`
[16] `https://code.google.com/apis/bigquery/`

et al., 2007) fully distributed design and Bigtable's column-family-based data model (Chang et al., 2006). Cassandra is an established NoSQL system used by some relevant companies[17] including Cisco, Facebook and Rackspace. There is a Cassandra storage adaptor for RDF.rb (Bendiken, 2010b) available, developed by Arto Bendiken and we developed CumulusRDF (Ladwig and Harth, 2010), which uses Apache Cassandra as a storage back-end. Brisk[18] is a Hadoop-style data processing framework built on top of the Apache Cassandra data store.

3.3 CouchDB

Apache CouchDB is a distributed, document-oriented database written in the Erlang; it can be queried and indexed in a MapReduce fashion. It manages the data as a collection of JSON documents and is used by Ubuntu, Couchbase and many more. Greg Lappen has provided a CouchDB storage adaptor for RDF.rb (Lappen, 2011). CouchDBs native language is JSON, hence it seems that efforts like *JavaScript Object Notation for Linked Data* (JSON-LD)[19] are a good fit for the data representation part. Only recently, a discussion took place on the CouchDB users list regarding "CouchDB vs. RDF databases" (Nunes, 2011).

3.4 Hadoop/Pig

Apache Hadoop is a software framework written in Java that supports reliable, scalable, distributed computing. Apache Pig[20] is a high-level data analysis language on top of Hadoop's MapReduce framework. The community discusses (Castagna, 2010) best practices for processing RDF data with MapReduce/Hadoop. Mika and Tummarello experimented with a system using Hadoop and Pig for SPARQL query processing (Mika and Tummarello, 2008). Tanimura et. al. (Tanimura et al., 2010) have reported on an extensions to the Pig data processing platform for scalable RDF data processing using Hadoop, somewhat related to what Sridhar et. al. (Sridhar et al., 2009) have suggested in their RAPID system. Arto Bendiken has developed RDFgrid (Bendiken, 2010a), a framework for batch-processing RDF data with Hadoop, as well as Amazon's Elastic Map Reduce (Amazon, 2011).

[17] `http://wiki.apache.org/cassandra/CassandraUsers`

[18] `http://www.datastax.com/products/brisk`

[19] `http://json-ld.org/`

[20] `http://pig.apache.org/`

3.5 HBase

Apache HBase is a distributed, versioned, column-oriented store modelled after Google' Bigtable, written in Java. A couple of institutions like Mendeley, Facebook and Adobe are using HBase. Gabriel Mateescu has provided an article (Mateescu, 2009) on how to process RDF data using HBase and Paolo Castagna has developed an experimental HBase-RDF implementation (Castagna, 2011). Sun and Jin (Sun and Jin, 2010) have proposed a scalable RDF store based on HBase.

3.6 MongoDB

MongoDB is a schema-free, (JSON) document-oriented database written in C++. It is used by an array of sites and providers including SourceForge, CERN, and Foursquare. Rob Vesse has reported (Vesse, 2010) on experiments he conducted with MongoDB as an RDF store and William Waites has provided a write-up on "Mongo as an RDF store" (Waites, 2010). Further, Antoine Imbert has developed `MongoDB::RDF` for Perl (Imbert, 2010).

3.7 Pregel

Pregel is a system for graph processing developed at Google (Malewicz et al., 2009). Similar to Hadoop, Pregel uses a set of nodes in a cluster to distribute work which is executed in parallel, with defined synchronization points to allow for exchange of intermediate results between the parallel processes. Unlike the MapReduce framework, for which an open source implementation is available in Apache Hadoop, Pregel is currently not available outside Google.

3.8 SimpleDB

Amazon SimpleDB is a distributed database/web-service written in Erlang. It is often used together with other Amazon Web Services (AWS) offerings such as the Simple Storage Service (S3), for example by Alexa, Livemocha or Netflix. Stein and Zacharias have summarised (Stein and Zacharias, 2010) their experiences with RDF processing in SimpleDB in their open source project Stratustore[21].

3.9 Riak

Riak is a Dynamo-inspired key-value store with a distributed database network platform and built-in

[21] `http://code.google.com/p/stratustore/`

MapReduce support. It supports high availability and is used in production by institutions such as Comcast, Wikia or Opscode. Andrew McKnight has shared (McKnight, 2010) his thoughts concerning SPARQL query processing on the Riak platform and we had a look into storing an RDF graph in Riak using HTTP Link headers (Hausenblas, 2010b) allowing for graph traversing.

3.10 Other Systems

There are a number of systems that would be capable of processing Linked Data in the cloud, however we are not aware of a cloud deployment or the features are not yet available; for sake of completeness, we list these systems in the following:

3.10.1 Distributed graph databases

- *Neo4j* is a graph database implemented in Java that has built-in RDF processing support, including indexing (Neo4jWiki, 2011). Further, *Gremlin*[22] is a graph traversal language that works over graph databases implementing the Blueprints interface[23], such as Neo4j or OrientDB[24] and *Graphbase*[25] is an implementation of the Blueprints interface on top of HBase.
- Microsoft's *Trinity* (Microsoft, 2011b) is a graph database over distributed memory cloud, providing computations on large scale graphs; it can reportedly be deployed on hundreds of machines. Further, Microsoft is building a graph library (Microsoft, 2011a) on top of their cloud computing framework Orleans that targets hosting very large graphs with billions of nodes and edges.
- *GoldenOrb*[26] is a cloud-based open source project for massive-scale graph analysis, building upon Hadoop, modelled after Googles Pregel (Malewicz et al., 2010) architecture.

3.10.2 Hybrid systems

- *MonetDB*[27] has support for RDF processing in the queue.
- *Sindice* (Oren et al., 2008), a semantic indexer, uses Hadoop and Lucence/SIREn to processes billions of triples.
- The *Large Knowledge Collider* project is working on a Web-scale Parallel Inference Engine[28], a MapReduce-based, distributed RDFS/OWL inference engine.
- Peter Hizalev reported (Hizalev, 2011) on a *Redis-based* triple database.
- Andy Seaborne reported (Seaborne, 2009) running TDB, a native persistent storage layer for the RDF processing framework Jena, on a cloud storage system.
- *SARQ*[29] is an open source text indexing system for SPARQL using a remote Solr server.

4 Discussion

Table 1 lists our Linked Data-specific requirements from Sec. 2 over the identified systems from Sec. 3.

The practical applicability of the systems surveyed varies: some systems represent first steps in mapping the RDF triple structure into a key/value-based storage layout, while others focus on optimising join processing capabilities.

Similarly, while some systems provide defined interfaces for insert, update and query, other systems are still in the prototype status which custom interfaces, often resembling those of the underlying processing system.

5 Conclusion

We have identified a number of requirements for cloud-based Linked Data management systems, surveyed existing offerings and analysed their capabilities with regard to the specified requirements. We have provided a first survey of the still evolving large-scale Linked Data processing landscape.

As becomes apparent from the plethora of systems surveyed and listed in Table 1, the burgeoning field of cloud-based Linked Data management is still fractured. Community-built benchmarks can serve as catalysts and help to unify a field. While the Wisconsin Benchmark (DeWitt, 1993) can be considered as the prototypical benchmark for parallel databases, it is rather outdated. In (Pavlo et al., 2009) Pavlo et. al. compared MapReduce with parallel databases, providing useful insights and guidance on what are important metrics. Most relevantly,

[22] https://github.com/tinkerpop/gremlin/wiki
[23] http://tinkerpop.com/
[24] http://www.orientechnologies.com/orient-db
[25] https://github.com/dgreco/graphbase
[26] http://www.goldenorbos.org/
[27] http://www.monetdb.org/Home
[28] http://www.few.vu.nl/~jui200/webpie.html
[29] https://github.com/castagna/SARQ

System	Backend	Identifiers	Interface	Partition	Update	Index	Query	Inference
(Hausenblas, 2010a)	BigQuery	URIs	Linked Data	quads	+	fixed	custom	-
(Ladwig and Harth, 2010)	Cassandra	URIs	Linked Data	quads	+	multiple	Linked Data lookups	-
(Tanimura et al., 2010)	Pig/Hadoop	URIs	custom	triples	-	fixed	SPARQL	rules
(Sridhar et al., 2009)	Pig/Hadoop	URIs	custom	triples	-	fixed	RAPID	-
(Mika and Tummarello, 2008)	Pig/Hadoop	URIs	custom	triples	-	fixed	SPARQL	forward-chaining rules
(Huang et al., 2011)	RDF-3X/Hadoop	URIs	custom	triples	-	fixed	SPARQL	-
(Sun and Jin, 2010)	HBase	URIs	custom	triples	-	fixed	SPARQL	-
(Vesse, 2010)	MongoDB	URIs	custom	triples	-	multiple	SPARQL	-
(Stein and Zacharias, 2010)	SimpleDB	URIs	custom	triples	+	multiple	SPARQL	-
(Hausenblas, 2010b)	Riak	URIs	HTTP	triples	-	fixed	custom	-

Table 1: Coverage of Linked Data processing capabilities.

Cooper et. al. (Cooper et al., 2010) introduced the *Yahoo! Cloud Serving Benchmark* (YCSB). Although generic and extensible by design, the YCSB is not directly suited for Linked Data benchmarking. Only recently, Dory et. al. (Dory et al., 2011) reported on elasticity and scalability measurements of cloud databases.

Our current work focuses on establishing a benchmark for Linked Data processing with cloud computing offerings[30].

We believe that a common, community-built benchmark could help to further identify and organise requirements, and in the process unite a fractured field towards a common goal.

[30] https://github.com/mhausenblas/nosql4lod

REFERENCES

Amazon (2011). Elastic MapReduce. http://aws.amazon.com/elasticmapreduce/.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53:50–58.

Bendiken, A. (2010a). RDFgrid. https://github.com/datagraph/rdfgrid.

Bendiken, A. (2010b). RDF.rb storage adapter for Apache Cassandra. https://github.com/bendiken/rdf-cassandra.

Berners-Lee, T. (2006). Linked Data, Design Issues.

Bizer, C. (2009). The Emerging Web of Linked Data. *IEEE Intelligent Systems*, 24(5):87–92.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked Data—The Story So Far. *Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22.

Bizer, C., Jentzsch, A., and Cyganiak, R. (2010). State of the LOD Cloud. http://www4.wiwiss.fu-berlin.de/lodcloud/state/.

Castagna, P. (2010). Best practices for processing RDF data using MapReduce. http://j.mp/processing-rdf-data-using-mapreduce-via-hadoop.

Castagna, P. (2011). HBase-RDF. https://github.com/castagna/hbase-rdf.

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *SIGMOD Rec.*, 39:12–27.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2006). Bigtable: a distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 15–15, Berkeley, CA, USA. USENIX Association.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud Computing*, SoCC '10, pages 143–154, New York, NY, USA. ACM.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41:205–220.

DeWitt, D. J. (1993). The Wisconsin Benchmark: Past, Present, and Future. In Gray, J., editor, *The Benchmark Handbook*. Morgan Kaufmann.

Dory, T., Mejias, B., Roy, P. V., and Tran, N.-L. (2011). Comparative elasticity and scalability measurements of cloud databases. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SoCC '11, New York, NY, USA. ACM.

Franklin, M. J., Halevy, A. Y., and Maier, D. (2005). From databases to dataspaces: a new abstraction for information management. *SIGMOD Record*, 34(4):27–33.

Grossman, R. and Mazzucco, M. (July/August, 2002). Dataspace - a web infrastructure for the exploratory analysis and mining of data. *IEEE Computing in Science and Engineering*, pages 44–51.

Grossman, R. L., Gu, Y., Mambretti, J., Sabala, M., Szalay, A., and White, K. (2010). An overview of the open science data cloud. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)*. ACM.

Hausenblas, M. (2009). Exploiting Linked Data to Build Web Applications. *IEEE Internet Computing*, 13(4):68–73.

Hausenblas, M. (2010a). BigQuery Endpoint. `http://code.google.com/p/bigquery-linkeddata/`.

Hausenblas, M. (2010b). Toying around with Riak for Linked Data. `http://webofdata.wordpress.com/2010/10/14/riak-for-linked-data/`.

Hizalev, P. (2011). Redis based triple database. `http://petrohi.me/post/6114314450/redis-based-triple-database`.

Huang, J., Abadi, D., and Ren, K. (2011). Scalable sparql querying of large rdf graphs. In *Proceedings of the 37st International Conference on Very Large Data Bases*.

Imbert, A. (2010). MongoDB-RDF. `https://github.com/ant0ine/MongoDB-RDF`.

Jacobs, I. and Walsh, N. (2004). Architecture of the World Wide Web, Volume One. W3C Recommendation 15 December 2004, W3C Technical Architecture Group (TAG).

Ladwig, G. and Harth, A. (2010). An RDF Storage Scheme on Key-Value Stores for Linked Data Publishing. Technical Report, Karlsruhe Institute of Technology (KIT).

Lappen, G. (2011). RDF.rb storage adapter for CouchDB. `https://github.com/ipublic/rdf-couchdb`.

Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G. (2009). Pregel: a system for large-scale graph processing - "abstract". In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, PODC '09, pages 6–6, New York, NY, USA. ACM.

Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G. (2010). Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 135–146, New York, NY, USA. ACM.

Mateescu, G. (2009). Finding the way through the semantic Web with HBase. developerWorks article, IBM.

McKnight, A. (2010). Sparql on Riak. `http://andrewmcknight.blogspot.com/2010/12/sparql-on-riak.html`.

Microsoft (2011a). Horton. `http://research.microsoft.com/en-us/projects/ldg/`.

Microsoft (2011b). Trinity. `http://research.microsoft.com/en-us/projects/trinity/`.

Mika, P. and Tummarello, G. (2008). Web semantics in the clouds. *IEEE Intelligent Systems*, 23:82–87.

Neo4jWiki (2011). Neo4j RDF/SPARQL Quickstart Guide. `http://wiki.neo4j.org/content/RDF_/_SPARQL_Quickstart_Guide`.

Nunes, D. (2011). CouchDB x RDF databases comparison. `http://comments.gmane.org/gmane.comp.db.couchdb.user/2334`.

Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., and Tummarello, G. (2008). Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1).

Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D. J., Madden, S., and Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 165–178, New York, NY, USA. ACM.

Seaborne, A. (2009). Running TDB on a cloud storage system. `http://seaborne.blogspot.com/2009/12/running-tdb-on-cloud-storage-system.html`.

Sridhar, R., Ravindra, P., and Anyanwu, K. (2009). RAPID: Enabling Scalable Ad-Hoc Analytics on the Semantic Web. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 715–730, Berlin, Heidelberg. Springer-Verlag.

Stein, R. and Zacharias, V. (2010). RDF on Cloud Number Nine. In *Workshop on New Forms of Reasoning for the Semantic Web (NeFoRS)*.

Sun, J. and Jin, Q. (2010). Scalable RDF store based on HBase and MapReduce. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 1, pages V1–633–V1–636.

Tanimura, Y., Matono, A., Lynden, S., and Kojima, I. (2010). Extensions to the Pig data processing platform for scalable RDF data processing using Hadoop. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pages 251 –256.

Umbrich, J., Hose, K., Karnstedt, M., Harth, A., and Polleres, A. (2011). Comparing data summaries for processing live queries over Linked Data. *World Wide Web*, pages 1–50.

Vesse, R. (2010). Experimenting with MongoDB as an RDF Store . Blog post, University of Southampton.

Waites, W. (2010). Mongo as an RDF store. `http://wwaites.posterous.com/mongo-as-an-rdf-store`.