



HAL
open science

AMADA: Web Data Repositories in the Amazon Cloud

Andrés Aranda-Andújar, Francesca Bugiotti, Jesús Camacho-Rodríguez,
Dario Colazzo, François Goasdoué, Zoi Kaoudi, Ioana Manolescu

► **To cite this version:**

Andrés Aranda-Andújar, Francesca Bugiotti, Jesús Camacho-Rodríguez, Dario Colazzo, François Goasdoué, et al.. AMADA: Web Data Repositories in the Amazon Cloud. ACM International Conference on Information and Knowledge Management, 2012, Maui, United States. hal-00730687v1

HAL Id: hal-00730687

<https://inria.hal.science/hal-00730687v1>

Submitted on 10 Sep 2012 (v1), last revised 19 Nov 2012 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AMADA: Web Data Repositories in the Amazon Cloud

Andrés Aranda-Andújar¹ Francesca Bugiotti^{1,3} Jesús Camacho-Rodríguez^{1,2}
Dario Colazzo^{1,2} François Goasdoué^{1,2} Zoi Kaoudi^{1,2} Ioana Manolescu^{1,2}

¹Inria Saclay—Île-de-France, France ²Université Paris-Sud, France ³Università Roma Tre, Italy
firstname.lastname@inria.fr

ABSTRACT

We present AMADA, a platform for storing Web data (in particular, XML documents and RDF graphs) based on the Amazon Web Services (AWS) cloud infrastructure. AMADA operates in a Software as a Service (SaaS) approach, allowing users to upload, index, store, and query large volumes of Web data. The demonstration shows (i) the step-by-step procedure for building and exploiting the warehouse (storing, indexing, querying) and (ii) the monitoring tools enabling one to control the expenses (monetary costs) charged by AWS for the operations involved while running AMADA.

Categories and Subject Descriptors

H.2.4 [Database Management]: Systems—*Concurrency, Distributed databases, Query processing*

General Terms

Design, Experimentation, Performance, Economics

Keywords

Cloud Computing, Web Data Management, Query Processing, AWS, Monetary Cost

1. INTRODUCTION

Increasing volumes of data are produced or exported into Web data formats. Among these, the W3C’s XML standard for structured documents (and in particular Web pages) and RDF for Semantic Web data are the best known. XML allows encoding complex documents whose structure may be constrained by an XML Schema. RDF graphs consist of triples of the form (s, p, o) stating that the subject node s has the property edge p whose value is the object node o . XML takes the lion’s share of Web content nowadays. At the same time, RDF is increasingly being used in numerous data sources such as in Linked Open Data.

To exploit large volumes of Web data, an interesting option is to warehouse it into a single access point repository. This typically involves some crawling or other means of identifying interesting data sources and loading these data sources into the repository where further processing can be applied. Huge data volumes have raised the need for distributed storage architectures and platforms typically deployed in a cloud environment, which provides scalable and

elastic resource allocation. *In this work, we consider hosting large volumes of Web data in the cloud, and their efficient storage and querying through a (distributed, parallel) platform also running in the cloud.* Such an architecture belongs to the general Software as a Service (SaaS) setting where the whole stack from the hardware to the data management layer are hosted and rented from the cloud.

In this context, an important challenge is *how to efficiently identify the parts of the data which need to be consulted in order to answer a given query.* In a cloud SaaS setting, efficient access path selection not only speeds up query processing, but also helps to reduce the monetary costs charged for querying, by avoiding work on some cloud machines that do not end up producing query results.

AMADA is a scalable platform for Web data management within the cloud, with a particular focus on this cloud-based data access path selection challenge [3, 4]. AMADA stores Web data in the cloud, and establishes indexes at various granularity over the data. Following other works [2, 11], AMADA uses the Amazon Web Services (AWS) cloud (<http://aws.amazon.com>), among the most prominent ones today. An important AWS feature is its elasticity, i.e., the ability to smoothly allocate computing power, storage, or other services, as the application demands vary.

The contribution of AMADA is twofold. First, AMADA presents a novel architecture harnessing the various subsystems of the popular cloud platform AWS for higher-level, efficient management of complex data. Second, AMADA includes an index-based mechanism for access path selection within a cloud-based repository, reducing query processing time as well as the warehouse operating costs.

The remainder of this work is structured as follows. Section 2 describes AMADA architecture while Section 3 outlines our Web data indexing algorithms. We present the demonstrated features in Section 4. Finally, we briefly discuss other related works in Section 5 and conclude in Section 6.

2. ARCHITECTURE

The design of AMADA was guided by the following objectives. First, we aimed to leverage AWS resources by scaling up to large data volumes. Second, we aimed at efficiency, in particular for the document storage and querying operations. We quantify this efficiency by the response time provided by the cloud-hosted application. Our third objective is to minimize cloud resource usage, i.e., the total work required for our operations. This is all the more important since in a cloud, total work translates into monetary costs. Fourth, the architecture should not be too closely tied to a specific XML or RDF query processor.

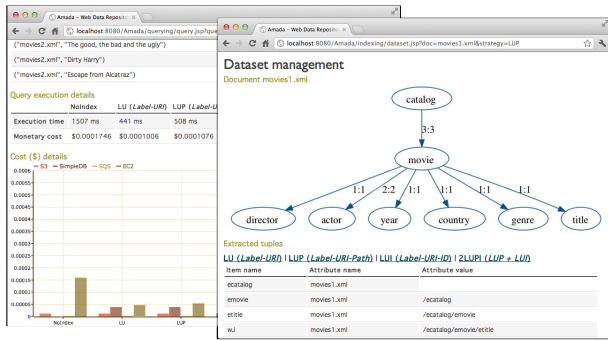


Figure 2: Demonstration interface.

attribute U_D , whose value is ϵ . Assume now that we want to evaluate the SPARQL query:

```
SELECT ?o WHERE {?s :hasAuthor "Foo" . ?s :hasTitle ?o . }
```

AMADA performs three *get* queries; one for each constant of the query: q_1 having key `phasAuthor`, q_2 having key `oFoo` and q_3 having key `phasTitle`. Then, it intersects the results of q_1 and q_2 to ensure that the `:hasAuthor` property and the "Foo" object value occur in the same dataset. The result is then *unioned* with the result of q_3 to obtain the datasets on which the SPARQL query will be evaluated. SPARQL semantics allows matches for this query to span over multiple datasets, i.e., query results are defined on a global "merged" graph.

Another RDF indexing strategy similar with ATT is ATS (for attribute subset) but it creates 7 item keys for each triple. Strategy ATS builds a larger index, in exchange for less *get* calls to identify the relevant graphs to a query.

Performance A detailed performance study of the XML side of AMADA is described in [14]. We compare four indexing strategies and show that indexing leads to cost savings that offset the index building and maintenance costs, in our experiments, after roughly 1000 queries [14].

4. DEMONSTRATION SCENARIO

The demonstration showcases loading and querying data in AMADA in real time within AWS. We use datasets about cultural artifacts, e.g., Open Data catalog of Bibliothèque Nationale de France (<http://data.bnf.fr/semanticweb>), as well as the general DBpedia corpus. AMADA displays in a Web-based interface (i) index entries to be added to SimpleDB, and (ii) our real-time rendition of the monetary costs entailed by index creation. Further, for a given query and indexing strategy, demo attendees will be shown (i) the calls to the SimpleDB API issued by the query processor to look up relevant datasets, (ii) the logical and physical plans for recombining (through intersection, join and/or union) the look-up results to identify the relevant datasets, and finally (iii) the AWS resource consumption associated to loading the respective datasets from S3 into EC2, processing the query there, and downloading the results from the cloud. AMADA includes an advisor tool, which based on the datasets and workload, provides index recommendations that depend on the relative importance given to the minimization of query response time and up-front monetary cost.

Figure 2 illustrates the interface; a more complete look-and-feel is available at <http://team.inria.fr/oak/amada/>.

5. RELATED WORK

An early work [2] has established the feasibility of building and exploiting B-tree indexes in S3 of AWS, while [11] fo-

cused on the problem of executing transactional workloads on cloud architectures, and AWS in particular. More recently, economic models for selecting indexes to materialize in a cloud were proposed in [9]. These works, however, considered regular, table-structured data, whereas we focus on irregular, tree- or graph-structured Web data.

Since the proposal of MapReduce [5] and the appearance of Hadoop, massively parallel data management using distributed infrastructures (a typical example of which are cloud-based) is a hot topic in industry and academia. For instance, XML query processing on top of Hadoop is studied in [10, 6]. Further, the increasing popularity of RDF has led to many recent works on parallelizing RDF processing using MapReduce [7, 8].

6. CONCLUSIONS AND PERSPECTIVES

AMADA exploits AWS components in order to achieve scalable storage and query processing for RDF and XML data. A main feature is content indexing, which is also the main focus of this demo proposal. Ongoing work on the platform includes (i) integrating content indexing within a generic, AWS-independent framework for massively parallel data management [1] and (ii) further investigating the impact of RDF graph partitioning on RDF query answering in a cloud context. Partitioning RDF graphs and efficiently parallelizing queries over triples entailed by the RDF semantics are open issues in this context.

Acknowledgments This work has been partially funded by the KIC EIT ICT Labs activities 11803, 11880 and RCLD 12115, as well as an AWS in Education research grant.

7. REFERENCES

- [1] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephelē/PACTs: a programming model and execution framework for web-scale analytical processing. In *SoCC*, 2010.
- [2] M. Brantner, D. Florescu, D. A. Graf, D. Kossmann, and T. Kraska. Building a database on S3. In *SIGMOD*, 2008.
- [3] F. Bugiotti, F. Goasdoué, Z. Kaoudi, and I. Manolescu. RDF data management in the Amazon Cloud. In *DanaC Workshop (collocated with EDBT/ICDT)*, 2012.
- [4] J. Camacho-Rodríguez, D. Colazzo, and I. Manolescu. Building Large XML Stores in the Amazon Cloud. In *DMC Workshop (collocated with ICDE)*, 2012.
- [5] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, 2004.
- [6] L. Fegaras, C. Li, U. Gupta, and J. Philip. XML Query Optimization in Map-Reduce. In *WebDB*, 2011.
- [7] J. Huang, D. J. Abadi, and K. Ren. Scalable SPARQL querying of large RDF graphs. *PVLDB*, 4(11), 2011.
- [8] M. Husain, J. McGlothlin, M. M. Masud, L. Khan, and B. M. Thuraisingham. Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing. *IEEE Trans. on Knowl. and Data Eng.*, 2011.
- [9] V. Kantere, D. Dash, G. Gratsias, and A. Ailamaki. Predicting cost amortization for query services. In *SIGMOD*, 2011.
- [10] S. Khatchadourian, M. P. Consens, and J. Siméon. Having a ChuQL at XML on the Cloud. In *A. Mendelzon Int'l. Workshop*, 2011.
- [11] D. Kossmann, T. Kraska, and S. Loesing. An evaluation of alternative architectures for transaction processing in the cloud. In *SIGMOD*, 2010.
- [12] T. Neumann and G. Weikum. The RDF-3X Engine for Scalable Management of RDF Data. *VLDBJ*, 19(1), 2010.
- [13] ViP2P web site. <http://vip2p.saclay.inria.fr>.
- [14] Technical report. http://jesus.camachorodriguez.name/_media/xml-aws/tech.pdf, 2012.