Enabling the auction model for the web of data with web preemption

GDD Team

LS2N, University of Nantes, Nantes, France
firstname.lastname@univ-nantes.fr

Abstract. Traditionally, querying the web of data is free of charge. However, ensuring availability of data and service incurs costs to data providers. The delayed auction model allows to fund the web of data using sponsorship but enabling this model while preserving queries performances is challenging. In this paper, we present an approach that enables the auction model while preserving the performances of queries. The key ideas are: (i) reindex bidding entities to ensure they appear first in query results, (ii) execute queries with web preemption to ensure lazy evaluation of queries. Experimental results demonstrate that our approach significantly outperforms a classical query evaluation approach in terms of time for first results.

1 Introduction

The web of data represents more than 38 Billions triples spread across upwards of 1300 sites. It can be queried for free by users worldwide. Albeit free, the access to this data does incur costs for the provider of the service. As data do not generate revenue yet, the economic sustainability of such services is under question: *Is there a way to have the web of data be free and financially sustainable at the same time?*

The delayed auction model [2] finances the web of data on the basis of advertisement, backing it up with proof of its economic viability. It puts forth the idea that entities of a graph could get bid by sponsors. Sponsored entities contained in a solution mapping would be returned with a higher priority by data services. Other solution mappings would be artificially delayed in order to incentivize sponsorship.

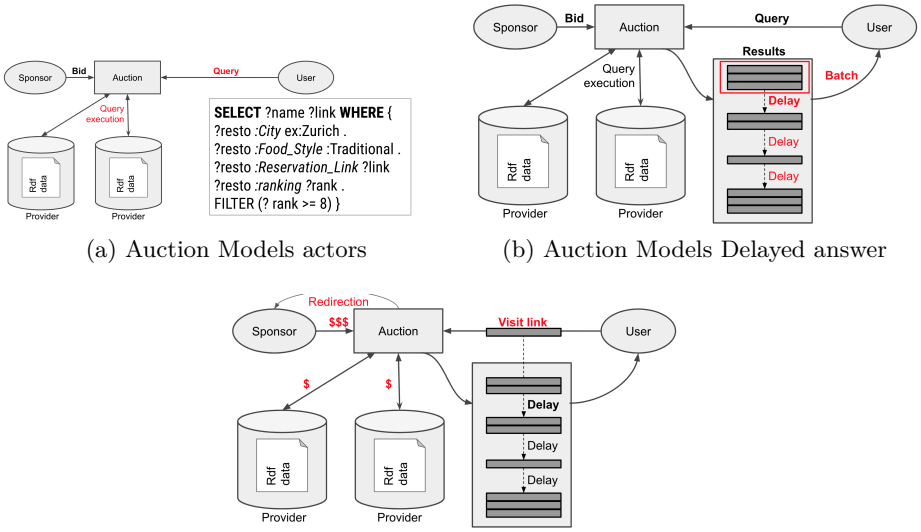
The proposal in [2] provides a naive appraoch for enabling the model. This naive appraoch does not allow a real exploitation of the auction model. To enable this model, we identify 2 issues: (i) The auction model forces an order on query results. Ordering results requires to collect all results first before ordering and delivering them. Consequently, executing queries with the auction model can slowdown query execution and negatively impact revenues, (ii) User are unlikely to consume all results of a query, however, the ordering constraint force the query engine to compute all of them. Consequently, most of the computation efforts are useless.

To address these issues, we propose to rely on web preemption [4] for the lazy evaluation of queries and on reindexing bidding entities to ensure orders at bid-time. This paper presents the following contributions:

1. An efficient execution model for the delayed auction model.
2. An experimental validation of the model. We compare the time for the first results of the execution model presented in [2] with our approach. We demonstrate that our model is four times faster than the naive approach.

This paper is structured as follows. Section 2 describes the delayed auction model. Section 3 details our approach and showcased its advantages. Section 5 presents related works. Section 6 concludes the work and and presents futur works

2 Background and motivations



(a) Auction Models actors

(b) Auction Models Delayed answer

(c) Auction Models Visiting links

Fig. 1: The Delayed Auction Model

Figure 1a presents the Delayed Auction model for financing the web of data as defined in [2]. The model includes four actors:

Users query the auctioneer free of charge, by sending SPARQL queries.
Data providers host RDF data and allow exclusive access to auctioneer.

Sponsors bid entities. A bid is a contract between sponsors and auctioneers. It promises money whenever an entity is visited, and expects this entity to be returned in priority, whenever it is part of query results.

Auctioneer is the interface between all of the previous actors. It handles and stores bid requests made by sponsors. It executes SPARQL queries and returns bidded entities first.

To illustrate, consider a user executes the following query Q_1 to retrieve restaurants in Zurich:

```
SELECT ?name ?link WHERE {
?resto :City ex:Zurich .
?resto :Food_Style :Traditional .
?resto :Reservation_Link ?link
?resto :ranking ?rank .
FILTER (? rank >= 8) }
```

The user sends Q_1 to the auctioneer. The auctioneer executes Q_1 by relying on existing data sources, then returns results where bid entities are returned first as described in figure 1b. It also delays the delivery of non-bid results, giving a competitive advantage to sponsored entities.

When a user visits a bided entity, the auctioneer notifies sponsors, collects revenue and redistributes revenue among data providers.

2.1 Enabling the Delayed Auction Model with Query rewriting

A straightforward approach for enabling the delayed auction model is to rewrite queries with *order by* clause such that bid entities appear first in results. To illustrate, consider the set of the following RDF facts in the auctioneer database.

```
:entity :bid bid_amount
:entity :sameAs auctionID
:entity :sponsor :sponsorID
```

The bid links an arbitrary entity `:entity` with a bid amount `bid_amount`, a sponsor `:sponsorID` and sponsored link `auctionID`.

```
select * where {
  ?resto :type :restaurant .
  ?resto :City ex:Nantes .
  ?resto:Food_style :Street .
  ?resto :review :Decent
}
```

(a) Original query

```
select coalesce(?id, ?restaurant) where {
  ?resto :type :restaurant .
  ?resto :City ex:Nantes .
  ?resto :Food_style :Street .
  ?resto :review :Decent .
  optional {
    ?resto auction:bid ?bid .
    ?resto auction:id ?id
  }
} order by ?bid
```

(b) Rewriting of the query with order by

Fig. 2: Enabling delayed auction model with rewriting

Given the previous representation of bids, this query can be rewritten as in figure 2. This rewriting features 3 additional clauses:

COALESCE performs a replacement of the name of sponsored items by their identifiers as a validation mechanism for the work of the auctioneer.
OPTIONAL performs two extra joins to retrieve the bid amount and the identifier of the item if a bid is defined on it.
ORDER BY orders the results by bid amounts.

The rewriting approach requires to acquire all results :

- the `OrderBy` clause requires to collect all results for a query before delivering the first results. Consequently, the time for first result is at least equal to the execution time of the original query. For example, in section 4, we run both queries of the figure 2. The original query take 349ms to deliver the first result while the rewritten query takes 1,314s. Time is money [3]:

  > "If the time to display search results is increased by 500ms, the revenue is reduced by 20%"

- Not all results are consumed, only first results are likely to be consumed by end-users. Computation is money too.

2.2 Problem Statement

Given a graph G, a SPARQL query Q and a set of bid entities B belonging to G. As in the delayed auction model [2], we suppose that:

Only one entity is sponsored in a given result there will be at most one bid on a given result entry.
The entity that is sponsored is known This ties in with the previous point. Before executing our queries, we have knowledge of the entity that is sponsored and that we are trying to order on.

We denote $[\![Q]\!]_{\mathcal{G},\mathcal{B}}$, the evaluation of Q on G where the results of the evaluation is ordered by $\mathcal{B}$. The problem is to compute the first results of $[\![Q]\!]_{\mathcal{G},\mathcal{B}}$ as fast as the first results of the $[\![Q]\!]_{\mathcal{G}}$.

3 Approach

The key ideas of our proposal are based on two main concepts:

Entity Reindexing: avoiding ordering by renaming entities such that they are indexed first.
Web Preemption: As a means to provide lazy evaluation of queries.

S	P	O
L'ardoise	City	Nantes
L'ardoise	Ranking	A
L'ardoise	Reservation	http://the-ardoise/res
Hunger Tame	City	New Orleans
Hunger Tame	Ranking	C
Hunger Tame	Reservation	https://hunger-tame.biz/res
jedzenie tutaj	City	Cracovie
jedzenie tutaj	Ranking	D
jedzenie tutaj	Reservation	http://jedzeni-etutaj/res.xml

Table 1: A sample of RDF Dataset

S	**P**	**O**
Hunger Tame	City	New Orleans
Hunger Tame	Ranking	C
Hunger Tame	Reservation	https://hunger-tame.biz/res
L'ardoise	City	Nantes
L'ardoise	Ranking	A
L'ardoise	Reservation	http://the-ardoise/res
jedzenie tutaj	City	Cracovie
jedzenie tutaj	Ranking	D
jedzenie tutaj	Reservation	http://jedzeni-etutaj/res.xml

P	**O**	**S**
City	Nantes	L'ardoise
City	New Orleans	Hunger Tame
City	Cracovie	jedzenie tutaj
Ranking	A	L'ardoise
Ranking	C	Hunger Tame
Ranking	D	jedzenie tutaj
Reservation	http://the-ardoise/res	L'ardoise
Reservation	https://hunger-tame.biz/res	Hunger Tame
Reservation	http://jedzeni-etutaj/res.xml	jedzenie tutaj

O	**S**	**P**
A	L'ardoise	Ranking
C	Hunger Tame	Ranking
Cracovie	jedzenie tutaj	City
D	jedzenie tutaj	Ranking
Nantes	L'ardoise	City
New Orleans	Hunger Tame	City
http://jedzeni-etutaj/res.xml	jedzenie tutaj	Reservation
http://the-ardoise/res	L'ardoise	Reservation
https://hunger-tame.biz/res	Hunger Tame	Reservation

Table 2: Original Indexes of sample of RDF Dataset

S	**P**	**O**
aaa-auction-bid-40	City	Cracovie
aaa-auction-bid-40	Ranking	D
aaa-auction-bid-40	Reservation	http://jedzeni-etutaj/res.xml
Hunger Tame	City	New Orleans
Hunger Tame	Ranking	C
Hunger Tame	Reservation	https://hunger-tame.biz/res
L'ardoise	City	Nantes
L'ardoise	Ranking	A
L'ardoise	Reservation	http://the-ardoise/res

P	**O**	**S**
City	Cracovie	aaa-auction-bid-40
City	Nantes	L'ardoise
City	New Orleans	Hunger Tame
Ranking	A	L'ardoise
Ranking	C	Hunger Tame
Ranking	D	aaa-auction-bid-40
Reservation	http://jedzeni-etutaj/res.xml	aaa-auction-bid-40
Reservation	http://the-ardoise/res	L'ardoise
Reservation	https://hunger-tame.biz/res	Hunger Tame

O	**S**	**P**
A	L'ardoise	Ranking
C	Hunger Tame	Ranking
Cracovie	jedzenie tutaj	City
D	jedzenie tutaj	Ranking
Nantes	L'ardoise	City
New Orleans	Hunger Tame	City
http://jedzeni-etutaj/res.xml	jedzenie tutaj	Reservation
http://the-ardoise/res	L'ardoise	Reservation
https://hunger-tame.biz/res	Hunger Tame	Reservation

Table 3: Reindexing bid entities

3.1 Entity Reindexing

Traditionally, RDF data are indexed at least with 3 index according to the Subject (SPO), to the Predicate (POS) and Object (OSP). For instance, figure 2 shows the three index of the the sample dataset in figure 1 (we omit prefix for simplicity).

Entity reindexing renames entities at bid time such that bided entities appear first in the index.

```
INSERT {
  :aaa-auction-bid-40 ?p1 ?o1;
    auction:bid 40 ;
    auction:sponsor :sponsorID ;
    owl:sameAs :jedzenie_tutaj .
  ?s2 ?p2 :aaa-auction-bid-40 .
}

DELETE {
  :jedzenie_tutaj ?p1 ?o1 .
  ?s2 ?p2 :jedzenie_tutaj .
}

WHERE {
  :jedzenie_tutaj ?p1 ?o1 .
  ?s2 ?p2 :jedzenie_tutaj .
}
```

Fig. 3: An insert/delete/where query abstracted by a bid

Re-indexing is implemented as a SPARQL update query. Basically, the query renames an entity with a prefix including the bid. To illustrate suppose a sponsor bids 100 euros for the entity `jedzenie tutaj`. This entity is renamed as illustrated in figure 3.

Updating the data in this fashion requires an additional space compared to the rewriting approach. In both approaches, databases are equivalent in terms of information, however, the identifier for sponsored entities simply *happens* to be different on each database.

Tables 2 show the index state before renaming. Tables 3 describe the index after renaming. As we can see, the entity `jedzenie tutaj` has been renamed in the indexes. As the query engine scans the index according to the order of the index, if `jedzenie tutaj` belongs to results, it will be delivered first.

3.2 Web Preemption

Originally web preemption[4] is defined as the capacity of a web server to suspend the execution of a running query and to resume the execution of a suspended query.

Web preemption was proposed as a solution for the problem of congestion on web servers. It proceeds by taking queries into an execution queue. When available, the server takes the first query in the queue and constructs a query execution plan. It then executes the query until a time quota is reached. When the quota is hit, the query is suspended, and results are returned to the client, with the interrupted query execution plan. The client can then resend the interrupted plan to resume the execution of the query.

We can extend the triggers to interrupt query execution to include the generation of a certain number of sponsored results. Doing so effectively recreates a batching mechanism and induces a delay. Both those features are integral to the delayed auction model. Moreover, it achieves lazy evaluation of queries, by letting the client decide on wether to continue the execution or not.

4 Experimental Study

We want to empirically answer the following question: Does the reindexing approach outperform the rewriting approach in term of time for first answer ?

We implemented the query rewriting approach and reindexing with infinite time quantum and reindexing with 100ms time quantum.

4.1 Experimental Setup

We use the WatDiv [1] SPARQL benchmark for both data and queries. The dataset contains 10^7 triples. Among all products (of which there are 25000), 5% are randomly sampled and assigned a random bid amount between 1 and 100. Bids are placed with different protocols as explained in the next sections.

We start with a workload of 12400 queries, of which there are SPARQL conjunctive queries with STAR, PATH and SNOWFLAKE shapes. Removing duplicate queries from the workload reduces it to around 4000 queries. Selecting again queries that return relevant results, i.e. results that contain sponsored entities, yields 220 queries. Those remaining queries constitutes our benchmark. They feature between 3 and 13 joins.

We use PostgreSQL as backend for Sage server [1] with B+ trees and 3 indexes SPO, POS and OSP.

4.2 Experimental Results

Figure 4 presents the time for first results for the three experimented approaches. With infinite time quantum, the time for first results is the same as query execution time. For all queries, it is better than the query execution time of query rewriting.

[1] `http:sage.univ-nantes.fr`

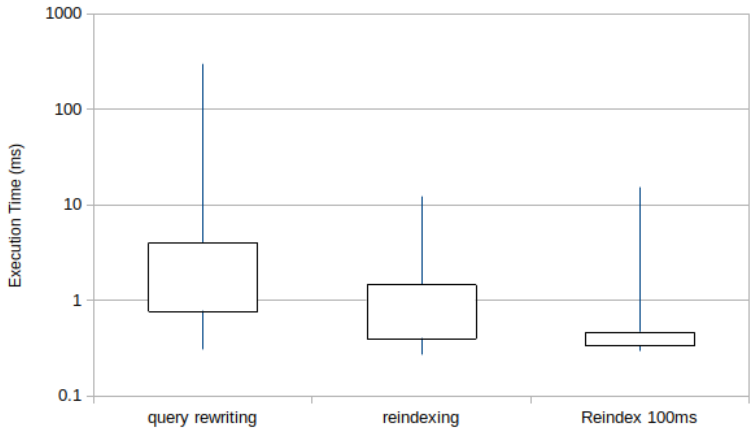


Fig. 4: Time for first results for query rewriting and reindexing

With a limited quantum, the time for first results is improved drastically. Using a 100ms time quatum is in average 4 times faster than rewriting the query. More precisely, half of the queries are 3.9 times faster and a quarter are 10 times faster than with infinite quantum.

5 Related works

Currently, the web of data is financed either through government funding or donations. However, when querying the web of data, more than one actual data source may provide the facts relevant for answering a single query. This means that in order to keep a query's answer complete and accessible, it is necessary for all of the data providers involved to be available. In other words, it is necessary to fund the whole federation of data sources. To the best of our knowledge, current donations target a particular dataset and not the whole federation.

With the goal of financial sustainability in mind, the example of the web of documents might seem like a good starting point. The world wide web becames sustainable through advertisement. As it is dependent on keywords users search, this form of advertisement allows for some amount of user ad targeting. But the web of data is mainly processed by machines; and machines do not read advertisement. It would seem like the model of the web of document cannot transition properly to the web of data.

The DBpedia Databus [2] is a platform that allows exchanging, curating and accessing data between multiple stakeholders. Any data entering the bus will be versioned, cleaned, mapped, linked and its licenses and provenance tracked. Hosting in multiple formats will be provided to access the data either as dump download or as API. Maybe talk about the DBpedia dataBus. Data sellers can

[2] `https://databus.dbpedia.org/`

link their offering to the open entities in the Databus. The DBpedia Association provides a business interface to allow guarantees, major improvements, stable maintenance, and hosting.

6 Conclusion and Future Works

In this paper, we demonstrated how web preemption and reindexing could be used to efficiently implement a prototype of the delayed auction model. This lays the foundation for a sustainable economic model for the web of data.

While this first implementation of the delayed auction model is a good step towards economic sustainability for the web of data, it adresses only a specific part of the challenges posed by the delayed auction model.

This opens a few perspectives. Mainly, it would be valuable to find a way to implement the extended auction model over our approach. This models advocates for the assignment of random slots in batches of query results. Random slots are filled with randomly selected results at the start of the batch process. This aims at reducing biases entailed by systematically favoring highest bid entities. Implementation of this extension is challenging in the context of web preemption, as random slot assignment is showcased with the assumption that all results can be batched simultaneously, which seems conflicting with lazy evaluation of queries.

Moreover, the delayed auction model is based on the assumption that only one sponsorship will ever be present in a given result which, in practice, is fairly likely.

References

1. Gao, L., Golab, L., Özsu, M.T., Aluç, G.: Stream watdiv: A streaming RDF benchmark. In: Groppe, S., Gruenwald, L. (eds.) Proceedings of the International Workshop on Semantic Big Data, SBD@SIGMOD 2018, Houston, TX, USA, June 10, 2018. pp. 3:1–3:6. ACM (2018), https://doi.org/10.1145/3208352.3208355
2. Grubenmann, T., Bernstein, A., Moor, D., Seuken, S.: Financing the web of data with delayed-answer auctions. In: Champin, P., Gandon, F.L., Lalmas, M., Ipeirotis, P.G. (eds.) Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018. pp. 1033–1042. ACM (2018), https://doi.org/10.1145/3178876.3186002
3. Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M.: Controlled experiments on the web: survey and practical guide. Data mining and knowledge discovery 18(1), 140–181 (2009)
4. Minier, T., Skaf-Molli, H., Molli, P.: Sage: Web preemption for public SPARQL query services. In: Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R.S., Zia, L. (eds.) The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019. pp. 1268–1278. ACM (2019), https://doi.org/10.1145/3308558.3313652