Traitement des requêtes d'agrégation sur un serveur SPARQL préemptif

Arnaud Grall, Thomas Minier, Hala Skaf-Molli, Pascal Molli

Traitement des requêtes d'agrégation sur un serveur SPARQL préemptif ⋆

Arnaud Grall[1,2], Thomas Minier[1], Hala Skaf-Molli[1], and Pascal Molli[1]

[1] LS2N – UNIVERSITY OF NANTES, FRANCE
{arnaud.grall,thomas.minier,hala.skaf,pascal.molli}@univ-nantes.fr

[2] GFI Informatique - IS/CIE, Nantes, France arnaud.grall@gfi.fr

1 Introduction

En suivant les principes du web des données, les fournisseurs de données ont publié des milliards de triples en format RDF (Bizer *et al.*, 2009; Schmachtenberg *et al.*, 2014). Il est possible de calculer des statistiques sur ces données en exécutant des requêtes SPARQL d'agrégation ; par exemple le nombre de propriétés par classe (Hasnain *et al.*, 2016), ou encore la durée de vie moyenne de scientifiques célèbres par pays. Cependant, le traitement des requêtes d'agrégation sur les serveurs SPARQL public reste difficile. En effet, la durée d'exécution des requêtes d'agrégation dépasse généralement les limites de temps autorisées par les serveurs SPARQL. On obtient alors des résultats partiels inutilisables dans le cadre de requêtes d'agrégation. (Polleres *et al.*, 2018; Soulet & Suchanek, 2019; Hasnain *et al.*, 2016).

Pour surmonter les limitations de quotas, les fournisseurs de données fournissent, en plus des serveurs SPARQL en ligne, des fichiers de sauvegarde contenant l'ensemble des données. Cependant, la ré-ingestion de milliards de faits RDF sur des ressources locales est extrêmement coûteuse et pose des problèmes de fraîcheur des données.

Récemment, des travaux de recherche ont été menés afin de construire des serveurs SPARQL fonctionnant sans limite de temps comme TPF (Verborgh *et al.*, 2016) ou SaGe (Minier *et al.*, 2019). Cependant, dans ces approches, les données à agréger sont d'abord transférées du serveur de données à un client intelligent qui effectue le calcul d'agrégation. La requête d'agrégation termine, mais le transfert de données est prohibitif.

Dans (Grall *et al.*, 2020), nous montrons comment il est possible d'étendre un serveur SPARQL préemptif avec un opérateur d'agrégation. Ce résultat est basé sur les propriétés de décomposition des fonctions d'agrégation. Dans notre approche, le serveur SPARQL préemptif est capable de calculer des agrégats partiels côté serveur pendant que le client intelligent combine ces agrégats partiels de manière incrémentale pour calculer les résultats finaux. Cette stratégie permet de réduire considérablement le trafic réseau lors des calculs d'agrégation.

2 Agrégation partielle et préemption web

Un serveur web préemptif permet de suspendre l'exécution d'une requête SPARQL après un quantum de temps afin d'en continuer l'exécution plus tard (Minier *et al.*, 2019). Malheureusement, il n'est pas possible de suspendre un opérateur d'agrégation. En effet, un calcul d'agrégation nécessite une structure de données dont la taille est proportionnelle à la taille de l'agrégat i.e. dans le pire cas, des données. Il faudrait donc au moment de la suspension, transmettre cet état du serveur web au client, puis dans le sens inverse au moment de la reprise.

Pour dépasser ce problème, nous calculons des agrégats partiels par quantum. En effet, la préemption web crée naturellement des partitions dans les résultats d'une requête. Comme les

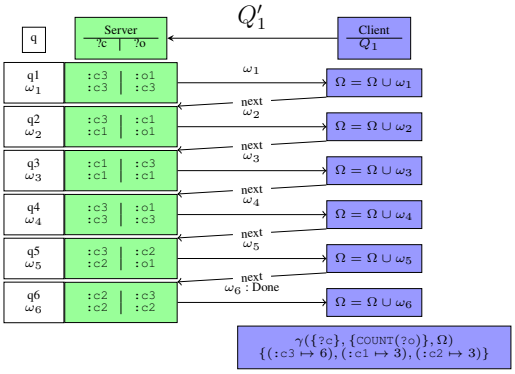
⋆. Cet article est un résumé en français de notre article publié à ESWC2020 (Grall *et al.*, 2020)

```
:s1 :p1 :o1 .
:s1 :a :c2, :c3.
:s2 :p1 :o1 .
:s2 :a :c1, :c3.
```

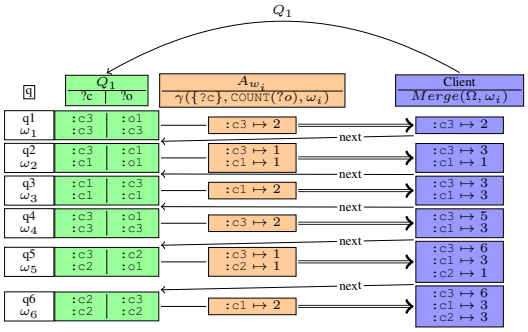
(a) $\mathcal{G}_1$

```
SELECT ?c
  (COUNT(?o) AS ?z)
WHERE { ?s :a ?c .
  ?s ?p ?o . ?s :p1 :o1}
GROUP BY ?c
```

(b) Requête SPARQL Q_1



(c) Evaluation de Q_1 sur $\mathcal{G}_1$ avec la préemption



(d) Evaluation de Q_1avec l'agrégation partielle Web originale (Minier *et al.*, 2019)

FIGURE 1 – Evaluation des requêtes d'agrégation SPARQL avec la préemption Web

TABLE 1 – Statistics of RDF datasets used in the experimental study

RDF Dataset	# Triples	# Subjects	# Predicates	# Objects	# Classes
BSBM-10	4 987	614	40	1 920	11
BSBM-100	40 177	4 174	40	11 012	22
BSBM-1k	371 911	36433	40	86202	103
DBpedia 3.5.1	153M	6 085 631	35 631	35 201 955	243

fonctions d'agrégation sont décomposables, nous calculons les agrégats partiels par quantum coté serveur et nous les fusionnons ensuite sur le client.

La figure (c) montre l'exécution de $Q1$ sur $\mathcal{G}_1$ sans agrégation partielle. Le corps Q'_1 de la requête est envoyé au serveur, qui renvoie les solutions (ω_i) en fin de chaque quantum (q_i). Tous les couples $<?c, ?o>$ sont donc transférés du serveur au client. Avec l'agrégation partielle (figure (d)), la requête $Q1$ est transmise au serveur, qui renvoie désormais les comptage partiels au lieu des couples. Le transfert de données est considérablement réduit.

3 Etude expérimentale

Nous voulons répondre empiriquement aux questions suivantes : (i) Quel est l'impact de l'agrégation partielle sur le transfert de données ? (ii) Quel est l'impact de l'agrégation partielle sur le temps d'exécution ?

Nous avons implémenté l'agrégation partielle comme une extension du moteur de requêtes SAGE [1]. Toutes les extensions et les résultats expérimentaux sont disponibles sur `https://github.com/folkvir/sage-sparql-void`.

Données et requêtes : nous avons construit un workload (*SP*) de 18 requêtes SPARQL d'agrégation extraites des requêtes de SPORTAL (Hasnain *et al.*, 2016) (requêtes sans ASK ni FILTER). La plupart des requêtes extraites ont le modificateur DISTINCT. Pour étudier l'impact de DISTINCT sur les performances des requêtes agrégées, nous avons défini un

1. `https://sage.univ-nantes.fr`

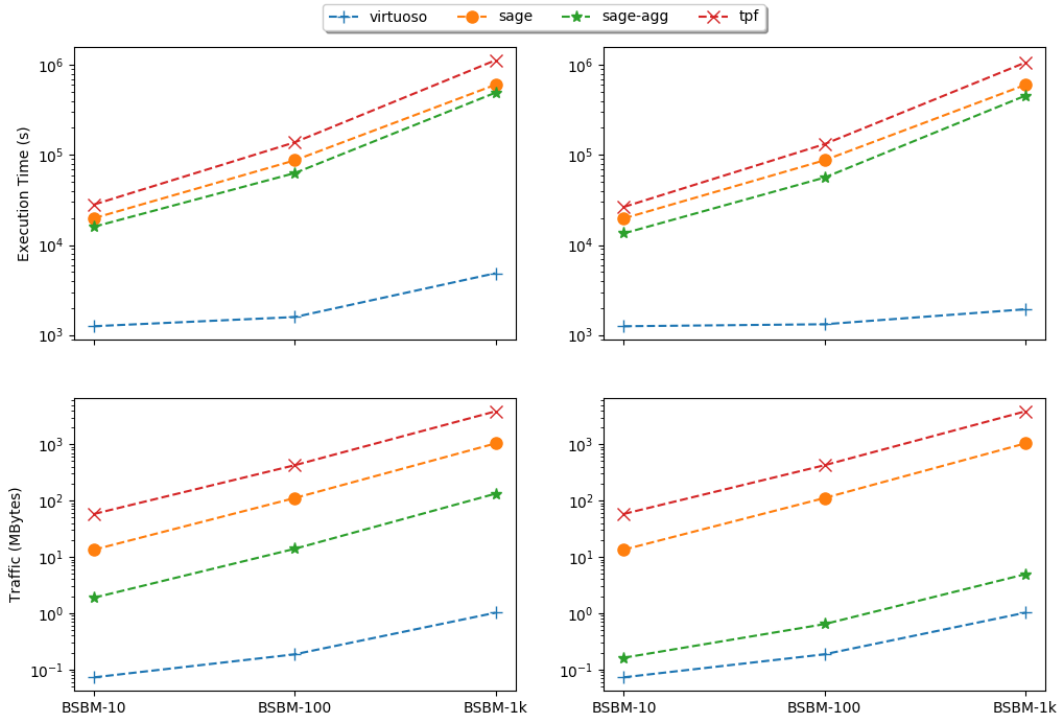


FIGURE 2 – Données transférées et temps d'exécution pour BSBM-10, BSBM-100 et BSBM-1k, lors de l'exécution des workloads *SP* (à gauche) et *SP-ND* (à droite)

nouveau workload, notée *SP-ND*, en supprimant le modificateur DISTINCT des requêtes de *SP*. Nous exécutons les workloads *SP* et *SP-ND* sur Berlin SPARQL Benchmark (BSBM) avec différents tailles et le fragment de DBpedia *v3.5.1*. Les statistiques de jeux de données utilisées sont détaillées dans le tableau 1.

Approches : Nous comparons les approches suivantes :

— SAGE : nous exécutons le moteur de requêtes SAGE query (Minier *et al.*, 2019) avec un quantum de temps de 150 ms et une page de taille maximale de 5000 résultats. Les données sont stockées sur un serveur PostgreSQL, avec des index sur *(SPO)*, *(POS)* et *(OSP)*.

— SAGE-AGG : est notre extension de SAGE avec les opérateurs d'agrégations partielles. Il fonctionne avec la même configuration que SAGE.

— *TPF* : nous exécutons le serveur TPF (Verborgh *et al.*, 2016) (sans cache Web) et le client Communica, en utilisant la taille de page standard de 100 triplets. Les données sont stockées au format HDT.

— *Virtuoso :* nous exécutons Virtuoso SPARQL endpoint (Erling & Mikhailov, 2009) (v7.2.4) **sans quotas** afin de fournir des résultats complets et un transfert de données optimal. Virtuoso est configuré avec *un seul thread* pour une juste comparaison.

Configurations des serveurs : les expérimentations sont menées sur Google Cloud Platform, avec un vCPU standard n1 2 : 2, 7,5 Go de mémoire avec un disque local SSD.

Métriques d'évaluation : Les résultats présentés correspondent à la moyenne obtenue sur trois exécutions successives de workloads. (i) *Données transférées* : est le nombre d'octets transférés au client lors de l'évaluation d'une requête. (ii) *Temps d'exécution* : est le temps entre le début de la requête et la production des résultats finaux par le client.

Résultats expérimentaux

Dans cette section, nous présentons seulement les résultats sur les jeux de données synthétiques (BSBM). Les résultats expérimentaux complets sont détaillés dans (Grall *et al.*, 2020).

La figure 2 présente les résultats en matière de transfert de données et de temps d'exécution pour BSBM-10, BSBM-100 et BSBM-1k. Les graphiques de gauche détaillent les résultats pour SP et les graphiques de droite, les résultats pour SP-ND. Virtuoso sans quota est présenté comme optimal en termes de données transférées et de temps d'exécution. Comme prévu, on observe les pires performances pour TPF. En effet, TPF ne prend pas en charge les projections et les jointures côté serveur. Par conséquent, le transfert de données est énorme même pour de petits ensembles de données. SaGe offre de meilleures performances que TPF principalement parce qu'il prend en charge la projection et les jointures côté serveur. SaGe améliore considérablement le transfert de données mais pas les temps d'exécution. En effet, les agrégations partielles permettent de réduire le transfert de données mais ne permettent pas d'accélérer le scan des données sur disque. En comparant les 2 charges de travail, nous pouvons voir que le traitement des requêtes sans `DISTINCT` (à droite) est beaucoup plus efficace dans le transfert de données qu'avec `DISTINCT` (à gauche). Pour les requêtes `DISTINCT`, les agrégations partielles ne peuvent supprimer que les doublons observés pendant un temps quantique uniquement et non ceux observées lors de l'exécution de la requête.

4 Conclusion

Dans (Grall *et al.*, 2020), nous avons montré qu'il est possible d'exécuter des requêtes d'agrégation sur des serveurs publics sans quotas et avec des transfers de données raisonnables. En perspective, nous projetons d'améliorer les temps d'exécution de requêtes d'agrégation en parallélisant les calculs d'agrégat partiels.

Références

Bizer C., Heath T. & Berners-Lee T. (2009). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, **5**(3), 1–22.

Erling O. & Mikhailov I. (2009). RDF support in the virtuoso DBMS. In *Networked Knowledge - Networked Media - Integrating Knowledge Management, New Media Technologies and Semantic Systems*, p. 7–24.

Grall A., Minier T., Skaf-Molli H. & Molli P. (2020). Processing SPARQL Aggregate Queries with Web Preemption. In *17th Extended Semantic Web Conference (ESWC 2020)*, The Semantic Web : ESWC 2020, Herkalion, Greece : Springer, Cham.

Hasnain A., Mehmood Q. & e Zainab ang Aidan Hogan S. S. (2016). SPORTAL : profiling the content of public SPARQL endpoints. *Int. J. Semantic Web Inf. Syst.*, **12**(3), 134–163.

Minier T., Skaf-Molli H. & Molli P. (2019). Sage : Web preemption for public SPARQL query services. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, p. 1268–1278.

Polleres A., Kamdar M. R., Fernández J. D., Tudorache T. & Musen M. A. (2018). A more decentralized vision for linked data. In *Proceedings of the 2nd Workshop on Decentralizing the Semantic Web (DeSemWeb 2018) co-located with ISWC 2018*.

Schmachtenberg M., Bizer C. & Paulheim H. (2014). Adoption of the linked data best practices in different topical domains. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference*, volume 8796, p. 245–260.

Soulet A. & Suchanek F. M. (2019). Anytime large-scale analytics of Linked Open Data. In *18th International Semantic Web Conference, ISWC*.

Verborgh R., Sande M. V., Hartig O., Herwegen J. V., Vocht L. D., Meester B. D., Haesendonck G. & Colpaert P. (2016). Triple pattern fragments : A low-cost knowledge graph interface for the web. *J. Web Sem.*, **37-38**, 184–206.