PubChemRDF on the Google Cloud Platform

Sunghwan Kim[1], Bo Yu[1], Jia He[1], Qingliang Li[1], Tiejun Cheng[1], Siqian He[1], Hannah Bast[2], Johannes Kalmbach[2] and Evan E. Bolton[1]

[1]National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA

[2]Department of Computer Science, University of Freiburg, Freiburg, Germany

Abstract

Knowledge graphs are powerful tools to exploit vast amounts of heterogeneous digital data on the Web. The Resource Description Framework (RDF) is a popular way to represent and manage knowledge graphs. Therefore, many public information resources provide RDF-formatted scientific data for data sharing and integration among scientific resources. An example of this is PubChemRDF, which refers to PubChem data in the RDF format, and where PubChem is a rather large and complex publicly available chemical data system. The present paper describes an ongoing effort to make PubChemRDF data available in cloud computing resources using two different RDF databases, Virtuoso and QLever. PubChemRDF data was loaded into a Virtuoso database in a series of virtual machines on the Google Cloud Platform (GCP). The Virtuoso database containing the PubChemRDF data was tested on these GCP virtual machines to investigate the effect of the number of virtual CPUs (vCPUs), memory size, and the disk storage type upon the query performance. In conjunction with the virtual machine pricing information, the query performance data was analyzed to find a cost-effective way to exploit PubChemRDF data in cloud environments. In addition, the query performance using QLever was investigated to explore its potential as an alternative to Virtuoso.

Keywords

PubChem, PubChemRDF, QLever, Virtuoso, Cloud Computing

1. Introduction

Over the past two decades, there was a rapid growth of digital data on the Web, accelerated by the rise of e-commerce and social media. Exploiting this enormous volume of heterogeneous data requires an appropriate way of organizing, storing, and querying it. Knowledge graphs are considered as a powerful tool for this purpose. Knowledge graphs [1, 2] represent data in a directed graph composed of nodes and edges, where the nodes correspond to entities of interest and the edges indicate the relation between the entities. The use of formal semantics in knowledge graphs enables machines to understand and process the data efficiently and unambiguously. Because of their machine readability, knowledge graphs are widely used in various AI systems, including recommender systems, question-answering systems, and information retrieval systems.

Resource Description Framework (RDF) (https://www.w3.org/RDF/) is a World Wide Web Consortium (W3C) standard and a popular way to represent and manage knowledge graphs. In RDF, the fundamental unit of a knowledge graph is a triple, which consists of a subject, an object, and a predicate defining their relationship. RDF is a standard model for data interchange on the Semantic Web (https://www.w3.org/2001/sw/wiki/Main_Page). Many attempts have been made to encode scientific knowledge in RDF, promoting data sharing and integration among different scientific domains. Examples are Medical Subject Headings RDF (MeSH RDF) [3], UniProt RDF [4], disGeNet [5], GlycoRDF [6], Disease Compass [7], Wikipathways RDF [8], and RDF-formatted data from the European Bioinformatics Institute (EBI) resources [9], including the Chemical Entities of Biological Interest (ChEBI) [10], ChEMBL [11], and BioModels linked dataset [12].

SWAT4HCLS 2025: 16th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences 2025

0000-0001-9828-2074 (S. Kim); 0000-0003-3952-8921 (B. Yu); 0000-0002-6513-8938 (J. He); 0000-0002-6453-236X (Q. Li); 0000-0002-4486-3356 (T. Cheng); 0000-0002-1707-4167 (S. He); 0000-0003-1213-6776 (H. Bast); 0000-0002-5582-1610 (J. Kalmbach); 0000-0002-5959-6190 (E. E. Bolton)

PubChemRDF [13, 14] refers to the RDF-formatted data from PubChem [15], a large-scale public chemical database that provides information on chemicals and their biological activities. Currently, PubChemRDF contains 18 billion triples (as of November 24, 2024). These triples are organized into multiple subdomains, based on the type of the subject of a triple. Users can download the entire PubChemRDF data set or only the data for desired subdomains from the PubChem File Transfer Protocol (FTP) site (https://ftp.ncbi.nlm.nih.gov/pubchem/RDF/) and import them into a triple (or quad) store or an RDF-aware graph database. In addition, PubChemRDF data can be accessed through a REST-ful interface. PubChemRDF enables integration of PubChem data with data from other resources across different domains. However, there is no open SPARQL query endpoint provided by PubChem.

SPARQL (https://www.w3.org/TR/sparql11-query/) is a query language for data stored in an RDF database. Open SPARQL query endpoints (https://www.w3.org/wiki/SparqlEndpoints) enable users to (remotely) query an RDF database and perform federated searches across multiple SPARQL endpoints. The reasons why PubChem does not provide an open SPARQL query endpoint are manyfold and beyond the scope of this paper. The expectation for PubChemRDF is that users will download PubChemRDF data and load it into their own (private) RDF database, where they will then use the data as needed or desired for their purposes typically via a SPARQL query interface. For many users, this is a substantial barrier in both technological expertise and time (e.g., to load the RDF data with each monthly release).

This paper describes our ongoing collaborative efforts to make PubChemRDF data available in popular cloud computing resources. PubChemRDF data were loaded into a Virtuoso database, which was made available in docker containers. The Virtuoso database containing the PubChemRDF data was tested on a series of virtual machines on the Google Cloud Platform (GCP) to investigate the effect of the number of virtual CPUs (vCPUs), memory size, and the disk storage type upon the query performance. In conjunction with the virtual machine pricing information, the query performance data was analyzed to find a cost-effective way to exploit PubChemRDF data in the cloud.

2. Methods

The overall strategy of this work can be considered as a three-step process. First, the PubChemRDF data was loaded into a Virtuoso database (version 7.2.14). Second, a Docker image was created, which in turn was used to generate a Docker container that can run the Virtuoso DB on a GCP virtual machine. Third, a set of sixteen SPARQL queries (represented as a set of use cases) available at the PubChemRDF Help page (https://pubchem.ncbi.nlm.nih.gov/docs/rdf-use-cases) were performed against the Virtuoso DB to check the query performance.

In addition, to explore the possibility of using database systems other than Virtuoso, QLever (version 0.5.15 from https://github.com/ad-freiburg/QLever) was tested on a local machine with the above-mentioned 16 queries. The resulting performance data was compared with those from Virtuoso.

2.1. Dataset

The PubChemRDF data for all subdomains (available in July 2024) was used in this study. The RDF data from ChEBI [9, 10] was also used because some of the SPARQL queries tested in this work required the ChEBI RDF data in addition to PubChemRDF. The size of the RDF data for individual subdomains is shown in Table 1. All RDF data was formatted in the N-Triples format (with the ".nt" extension), totaling 3 TB uncompressed. Most of this data was attributed to PubChemRDF, while the ChEBI RDF data was about 1 GB. These input RDF files were archived and compressed using the tar and gzip commands. The size of the final input RDF file (with the ".tar.gz" extension) was 159 GB. This file was made available for public access through the PubChem FTP site (https://ftp.ncbi.nlm.nih.gov/pubchem/).

2.2. Loading RDF data into a Virtuoso Database

The compressed RDF data (in the ".nt.gz" format) were loaded into the Virtuoso database (Version 7.2.14) on an in-house Linux machine (Almalinux 8.10, Intel Xeon Gold 5317 CPU, 12 cores/24 threads, 250 GB

Table 1
Size of the input RDF data and output Virtuoso database and the loading time. The output size of the Virtuoso DB and loading time are provided to give a perspective of the cost (file size and time) to users for different subdomains of PubChemRDF.

Subdomain	Input RDF (uncompressed)	Input RDF (compressed)	Output Virtuoso DB	Loading Time
all	3215.60 GB	159 GB	867.63 GB	9 hours 18 minutes
descriptor	1112.92 GB	43 GB	386.33 GB	3 hours
patent	567.03 GB	32 GB	91.42 GB	1 hour 36 minutes
compound	499.98 GB	12 GB	201.12 GB	1 hour 10 minutes
substance	353.07 GB	16 GB	82.93 GB	1 hour
reference	242.53 GB	23 GB	60.86 GB	54 minutes
synonym	139.23 GB	17 GB	67.75 GB	35 minutes
endpoint	109.87 GB	6.1 GB	35.20 GB	22 minutes
inchikey	57.66 GB	3.3 GB	45.58 GB	20 minutes
cooccurrence	53.79 GB	1.8 GB	6.65 GB	5 minutes
measuregroup	48.32 GB	4.5 GB	33.29 GB	13 minutes
author	21.30 GB	2.6 GB	17.96 GB	6 minutes
anatomy	1.66 GB	0.15 GB	1.07 GB	3 minutes
gene	1.64 GB	0.07 GB	0.50 GB	30 seconds
protein	1.52 GB	0.13 GB	0.68 GB	25 seconds
bioassay	1.11 GB	0.06 GB	3.06 GB	21 seconds
organization	1.07 GB	0.11 GB	0.76 GB	34 seconds
chebi	0.98 GB	0.08 GB	0.63 GB	40 seconds
grant	0.82 GB	0.07 GB	0.41 GB	29 seconds
pathway	0.36 GB	0.017 GB	0.30 GB	4 seconds
patentcpc	0.26 GB	0.013 GB	0.28 GB	10 seconds
taxonomy	0.22 GB	0.013 GB	0.25 GB	8 seconds
cell	0.09 GB	0.004 GB	0.10 GB	6 seconds
patentipc	0.06 GB	0.003 GB	0.18 GB	5 seconds
conserveddomain	0.04 GB	0.004 GB	0.15 GB	4 seconds
journal	0.03 GB	0.003 GB	0.13 GB	5 seconds
disease	0.02 GB	0.001 GB	0.09 GB	3 seconds
book	0.009 GB	0.0009 GB	0.08 GB	3 seconds
concept	0.004 GB	0.0002 GB	0.05 GB	2 seconds
source	0.0007 GB	0.00006 GB	0.05 GB	3 seconds

Table 2
Effects of the storage type for the input RDF data upon the time taken for loading the data into Virtuoso and QLever databases. The output in all cases was to a solid-state drive (SSD). In the case of input SSD, the input and output were on the same SSD.

Input RDF data storage	Loading time (hours)	
	Virtuoso	Qlever
Solid-State Drive (SSD)	10.52	12.52
Hard Disk Drive (HDD)	10.40	12.75
Network File System (NFS)	12.32	12.80

memory). (See Table 1.) To investigate the effects of the input RDF storage type upon the loading time, three disk types (where the input RDF data were stored) were considered: a solid-state drive (SSD), a hard disk drive (HDD), and a network filesystem (NFS). For all three cases, the Virtuoso database into which the RDF data was loaded was on an SSD. The loading times are shown in Table 2, along with those for the QLever database (to be discussed later).

Table 3
Specifications and estimated costs of virtual machines tested in this study. All virtual machines were of E2 machine type, with 1 TB of disk space.

Virtual machine	No. vCPUs	Memory (GB)	Storage type[1]	Estimated Cost [2]			
				Monthly			Hourly
				vCPUs + Memory	Storage	Total	
VM32-128	32	128	SSD	$881.50	$187.00	$1,068.50	$1.46
VM32-64	32	64	SSD	$727.70	$187.00	$914.70	$1.25
VM32-32	32	32	SSD	$650.80	$187.00	$837.80	$1.15
VM32-16	32	16	SSD	$612.35	$187.00	$799.35	$1.09
VM16-128	16	128	SSD	$594.55	$187.00	$781.55	$1.07
VM16-64	16	64	SSD	$440.75	$187.00	$627.75	$0.86
VM16-32	16	32	SSD	$363.85	$187.00	$550.85	$0.75
VM16-16	16	16	SSD	$325.40	$187.00	$512.40	$0.70
VM16-8	16	8	SSD	$306.17	$187.00	$493.17	$0.68
VM8-64	8	64	SSD	$297.28	$187.00	$484.28	$0.66
VM8-32	8	32	SSD	$220.37	$187.00	$407.37	$0.56
VM8-16	8	16	SSD	$181.92	$187.00	$368.92	$0.51
VM8-8	8	8	SSD	$162.70	$187.00	$349.70	$0.48
VM32-128B	32	128	Balanced	$881.50	$110.00	$991.50	$1.36
VM32-128S	32	128	Standard	$881.50	$44.00	$925.50	$1.27

[1] SSD (solid-state drive), Balanced (balanced persistent disk), Standard (standard persistent disk). [2] The costs were estimated based on the rates for the US-East 4 region (northern Virginia) and are provided to give a relative cost between the virtual machine instances.

2.3. Testing the Virtuoso database in virtual machines

Docker was used to create a Docker image for the environment necessary to run the Virtuoso database in virtual machines on the Google Cloud Platform (GCP). Multiple virtual machines were considered to investigate how query performances are affected by the number of CPUs, the memory size, and the disk storage type. Table 3 lists the hardware specifications of the virtual machines, along with their estimated costs.

All 15 virtual machines were of the E2 machine type, which is designed for low-cost, day-to-day computing (https://cloud.google.com/compute/docs/general-purpose-machines#e2_machine_types). The types of vCPUs used for the E2 machines can be Intel Skylake, Broadwell, and Haswell as well as AMD EPYC Rome and Millan (https://cloud.google.com/compute/docs/machine-resource). The processors are selected when a virtual machine is created. It is noteworthy that, while E2 machines can have up to 128 GB of memory, the minimum and maximum memory sizes allowed for them depend upon the number of vCPUs. The E2 machines with 32 vCPUs have at least 16 GB of memory and those with 8 vCPUs have no more than 64 GB of memory.

All virtual machines (except for VM32-128B and VM32-128S) had a 1000-GB SSD disk. VM32-128B and VM32-128S had a 1000-GB balanced persistent disk and a 1000-GB standard persistent disk, respectively. The latter two machines (with VM32-128) were used to investigate the effect of the disk type on the query performance and may be useful to help select the most cost-effective approach.

The docker image was used to create Docker containers in virtual machines and the loaded Virtuoso database was copied from Google Storage to individual virtual machines, using the "gsutil" tool (https://cloud.google.com/storage/docs/gsutil). This tool enables a faster file transfer, compared to the Linux "cp" command. For each virtual machine, the 16 SPARQL queries in the PubChemRDF Help page (urlhttps://pubchem.ncbi.nlm.nih.gov/docs/rdf-use-cases) were run and the query time was recorded.

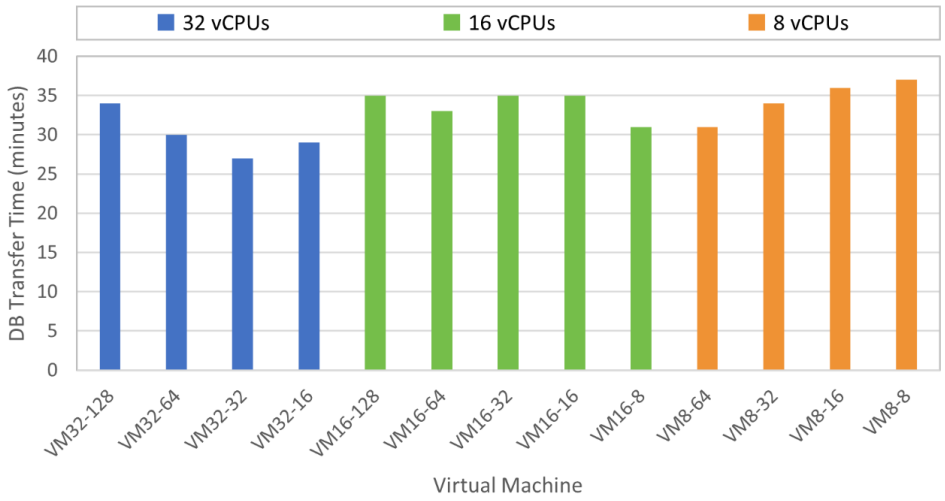

Figure 1: Virtuoso database transfer time for GCP virtual machines, which had 1000-GB solid-state drive (SSD) storage. The virtual machines are color-coded by the number of virtual CPUs (vCPUs): blue for 32 vCPUs, green for 16 vCPUs, and orange for 8 vCPUs. Virtual machines color-coded with the same color have different memory sizes. See Table 3 for the hardware specifications of the virtual machines.

2.4. Testing QLever on a local machine

The PubChemRDF and ChEBI RDF data were loaded into the QLever database [16] installed on the local Linux machine previously used in Section "2.2. Loading RDF data into a Virtuoso Database". The three types (SSD, HDD, and NSF) of disk storage for the input RDF data were tested and their loading times are reported in Table 2. The 16 SPARQL queries were run against both the Virtuoso and QLever databases on the local machine and their query performances were compared.

3. Results and discussion

3.1. Virtuoso query performance on a local machine

The RDF data from PubChem and ChEBI was loaded into a Virtuoso database. The size of the RDF data was 3 TB before compression and 159 GB after compression (see Table 1). The largest subdomain was the descriptor subdomain, which amounted to 353 GB uncompressed. This subdomain contains triples encoding various molecular descriptors computed for compounds, such as the molecular weight, XLogP value, hydrogen bond acceptor and donor counts, rotatable bond count, and chemical structure line notations (e.g., SMILES, InChI). Depending on the type of data storage where the RDF data was stored, the data loading into Virtuoso DB took from 10.4 to 12.3 hours (see Table 2). The Virtuoso database loaded with all data was 867.63 GB (Table 1) and was stored in Google Storage.

3.2. Transfer of virtuoso to virtual machines

A total of 15 virtual machines (Table 3) were created, each of which had a Docker container that could run the Virtuoso DB copied from Google Storage. As shown in Figure 1, it took 27-37 minutes to copy the Virtuoso database to virtual machines.

3.3. Virtuoso query performance on virtual machines

For each machine, the 16 SPARQL queries were tested, and the resulting query time is shown in Table 4.

The query time for the 16 queries performed on the virtual machines, which had a 1000-GB SSD disk, are compared in Figure 2. The total query time for the 13 virtual machines is compared in Figure 3.

Table 4
Query time in seconds for the 16 SPARQL queries performed on the Google Cloud Platform (GCP) virtual machines.

Query	VM32-128	VM32-64	VM32-32	VM32-16	VM16-128	VM16-64	VM16-32	VM16-16
1	106.46	73.65	68.85	65.79	77.80	88.30	97.78	75.51
2	5.79	3.03	3.08	2.93	3.37	3.57	3.83	3.38
3	341.30	336.01	363.40	358.95	353.79	338.85	362.15	367.71
4	243.76	228.96	226.23	227.95	238.42	231.21	238.94	234.24
5	60.87	55.94	62.97	58.87	72.11	81.75	74.04	65.03
6	1.29	1.19	1.21	1.16	1.29	1.78	1.64	1.40
7	51.83	50.94	52.25	51.94	60.85	66.45	65.74	55.53
8	16.62	15.32	13.22	13.08	15.25	17.73	17.14	14.66
9	267.96	266.65	277.20	281.36	272.99	286.54	280.95	294.82
10	21.38	19.80	22.41	20.76	23.37	25.79	27.65	23.20
11	0.17	0.11	0.18	0.36	0.22	0.16	0.19	0.20
12	16.67	16.26	15.87	17.60	18.11	18.99	20.26	17.83
13	0.25	0.25	0.25	0.28	0.27	0.38	0.31	0.26
14	0.14	0.11	0.17	0.18	0.16	0.16	0.23	0.15
15	0.48	0.42	0.48	0.45	0.48	0.56	0.58	0.49
16	25.31	21.99	24.00	24.06	24.32	27.83	29.49	26.00
Total	1160.26	1090.63	1131.78	1125.71	1162.80	1190.06	1220.91	1180.42

Query	VM16-8	VM8-64	VM8-32	VM8-16	VM8-8	VM32-128B	VM32-128S	
1	75.54	106.63	26.62	88.30	25.26	78.12	172.90	
2	3.24	4.68	4.84	4.04	3.38	3.47	8.19	
3	386.14	346.53	342.43	378.38	400.54	355.26	528.37	
4	234.77	240.61	250.04	240.83	238.05	230.43	291.16	
5	70.33	89.67	94.48	72.71	82.48	72.42	167.63	
6	1.51	1.63	1.65	1.78	1.53	1.46	7.67	
7	50.99	80.60	0.15	57.88	0.14	58.04	48.83	
8	16.25	20.33	17.12	18.99	18.70	15.28	140.63	
9	377.88	279.19	296.68	296.01	356.41	269.44	301.57	
10	21.57	23.11	38.62	24.96	25.81	23.77	22.83	
11	0.19	0.21	0.21	0.26	0.20	0.20	1.21	
12	18.27	20.99	20.30	22.53	21.46	19.23	120.14	
13	0.30	0.33	0.31	0.28	0.29	0.26	1.34	
14	0.20	0.16	0.23	0.25	0.22	0.19	1.21	
15	0.49	0.66	0.52	0.62	0.56	0.59	4.94	
16	27.54	29.46	35.63	28.06	34.19	26.99	89.76	
Total	1285.21	1244.80	1129.85	1235.89	1209.23	1155.18	1908.37	

Generally speaking, the machines with more vCPUs resulted in shorter query times. However, the effect of the memory size in the virtual machines is not obvious. This suggests that PubChemRDF can run on some of the smallest memory configurations tested, provided that sufficiently fast I/O is available. Given that the queries were run serially (as opposed to all 16 at once or in some other parallel fashion), additional CPUs may provide a greater degree of concurrency if many queries are desired to be run.

3.4. Effects of the data storage type upon query performances

Google Cloud provides various disk storage options for Compute Engine virtual machines (https://cloud.google.com/compute/docs/disks#disk-types). For the E2 machine types used in this study, four disk storage types are supported: SSD persistent disk, balanced persistent disk, standard persistent disk, and extreme persistent disk. Among them, the extreme persistent disk was the most expensive, due to the additional cost of provisioned Input/Output operations per second (IOPS) ($7,200 per month for 100000 provisioned IOPS). This is too expensive, considering that our project aims at providing

Table 5
Query time in seconds for the 16 SPARQL queries performed using Virtuoso and QLever databases on an in-house local machine. The queries were performed for three different disk types: solid-state drive (SSD), hard disk drive (HDD), and networking file system (NFS).

Query	Virtuoso			QLever		
	SSD	HDD	NFS	SSD	HDD	NFS
1	5.00	25.97	29.42	5.48	7.93	6.31
2	1.47	3.73	4.99	1.08	1.38	1.27
3	2.08	1566.01	12.39	1.18	3.38	2.34
4	183.84	206.45	212.37	1.38	3.80	9.84
5	404.87	428.74	625.88	21.27	27.43	21.83
6	0.36	1.78	3.72	5.26	5.42	5.51
7	0.06	0.38	0.64	5.40	5.52	5.56
8	11.19	92.54	101.94	5.78	6.99	7.40
9	0.37	1.27	1.65	8.60	9.53	9.52
10	304.09	310.78	317.10	12.48	12.46	12.35
11	1.90	5.32	7.37	1.05	1.17	1.11
12	7.19	40.40	38.07	1.59	4.00	3.17
13	0.25	1.23	1.06	1.07	1.25	1.29
14	0.07	0.57	0.55	1.08	1.28	1.20
15	0.97	9.29	7.08	0.06	1.43	3.99
16	296.84	299.45	306.83	1.15	1.29	1.21
Total	1220.55	2993.91	1671.06	73.91	94.26	93.90

low-cost access to PubChemRDF data through cloud resources. Therefore, the remaining three disk types were considered to investigate the effects of the disk types on the Virtuoso DB transfer time and the total query time. For this purpose, the data for VM32-128, VM32-128B, and VM32-128S (in Table 4) was used. The three machines had the same number of vCPUs and memory sizes (i.e., 32 vCPUs and 128 GB) but different disk types (SSD for VM32-128, balanced persistent disk for VM32-128B, and standard persistent disk for VM32-128S) (see Table 3).

Figure 4 compares the Virtuoso DB transfer time and the total query time for VM32-128, VM32-128B, and VM32-128S. The machines with SSD and balanced persistent disks (i.e., VM32-128 and VM32-128B) showed similar Virtuoso DB transfer time (34 vs. 35 minutes, respectively) and the total query time (1160 vs. 1155 seconds, respectively). However, the machine with the standard persistent disk (i.e., VM32-128S) gave much slower results (124 minutes for the DB transfer time and 1908 seconds for the total query time). It should be noted that the SSD disk is more expensive than the other two options ($187, $110, and $44 for SSD, balanced, and standard persistent disks, respectively). This pricing information and the performance data suggest that the balanced persistent disk is a cost-effective alternative to the SSD disk, with essentially the same performance at the time of our testing.

3.5. Comparison between Virtuoso and QLever on a local machine

While Virtuoso was used to test the query performance on virtual machines, there are other database applications that can load and exploit PubChemRDF data. One of them is QLever [16, 17], which is a high-performance open-source triplestore and graph database. To explore the potential of QLever as an alternative to Virtuoso, a QLever database containing the PubChemRDF data was built on an in-house Linux machine and the sixteen PubChemRDF use case queries were performed against the QLever database. The effects of the type of disk storage (i.e., SDD, HDD, and NFS) upon the performance of QLever were also investigated. The resulting data were compared with those from the Virtuoso database. The data loading times and query times for the two databases are listed in Table 2 and Table 5, respectively. Figure 5 compares the data loading times and total query times between Virtuoso and QLever.

Data loading to QLever took slightly more time than Virtuoso for all input RDF disk storage types.

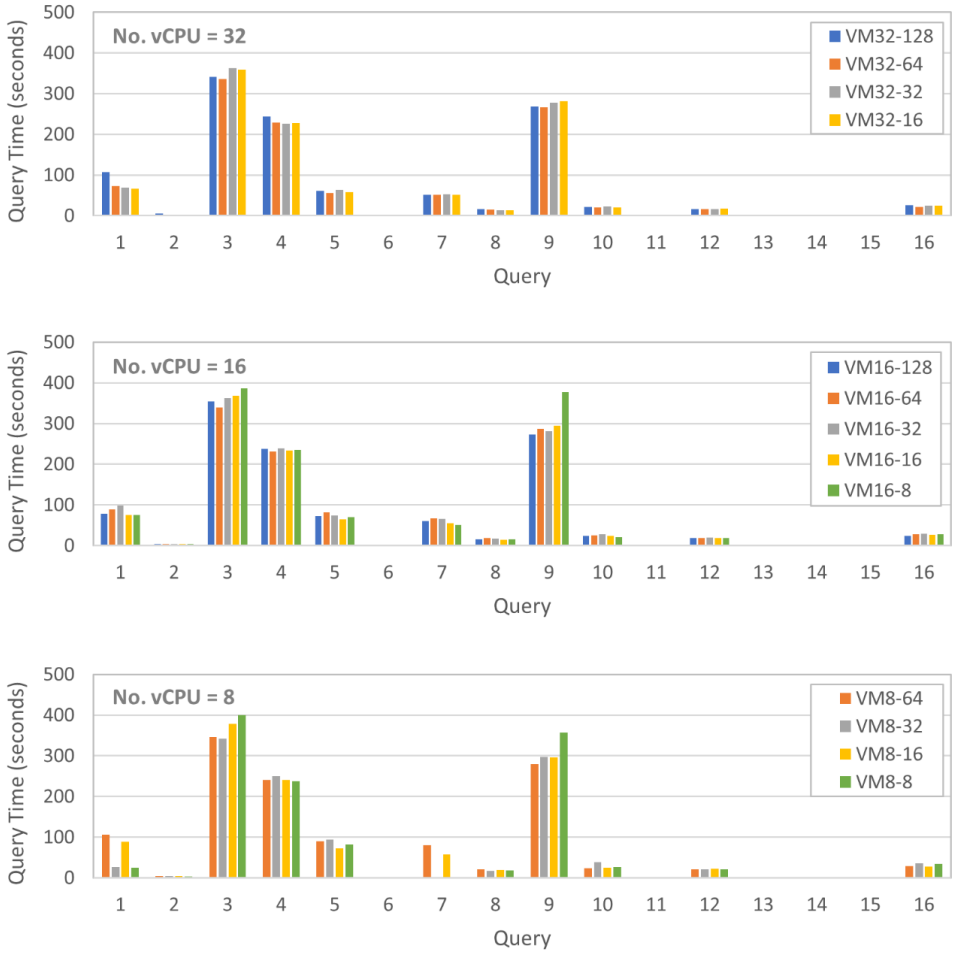

Figure 2: Query time for the 16 SPARQL queries executed on GCP virtual machines, which had 1000-GB solid-state drive (SSD) storage. The virtual machines are grouped by the number of virtual CPUs (vCPUs): 32 vCPUs (upper panel), 16 vCPUs (middle panel), and 8 vCPUs (lower panel). The virtual machines are color-coded by the memory size.

However, Virtuoso relies on a large amount of RAM for efficient loading and was configured with NumberOfBuffers = 15000000, which is the recommended setting when 180 GB of RAM are available. When using the recommended setting for 64 GB of RAM, Virtuoso's loading time more than triples. In constrast, QLever achieves the mentioned loading time when configured to use only 20 GB of RAM. While the Virtuoso database size was 867 GB, the QLever database was 355 GB, which is about 2.5 times smaller. That is, with a comparable loading time, Qlever achieves a much more compact database with a fraction of the RAM, which are both potential advantages over Virtuoso regarding the cost of cloud deployment.

In addition, QLever resulted in a substantially shorter total query time for all three disk storage types. For example, when the SSD was used, the total query times were 74 seconds and 1221 seconds for QLever and Virtuoso, respectively. That is, QLever is more than 16 times faster than Virtuoso for the PubChemRDF use case queries.

This stark difference in query time can be explained by the different architectures of the sytems. Virtuoso is primarily a relational database system, which translates SPARQL queries to SQL queries on the underlying relation database. This has the advantage of leveraging mature technology in relational databases, but has the disadvantage of making various optimization for RDF databases hard or impossible. QLever, on the other hand, is a native RDF database, which implements many RDF-specific optimizations.

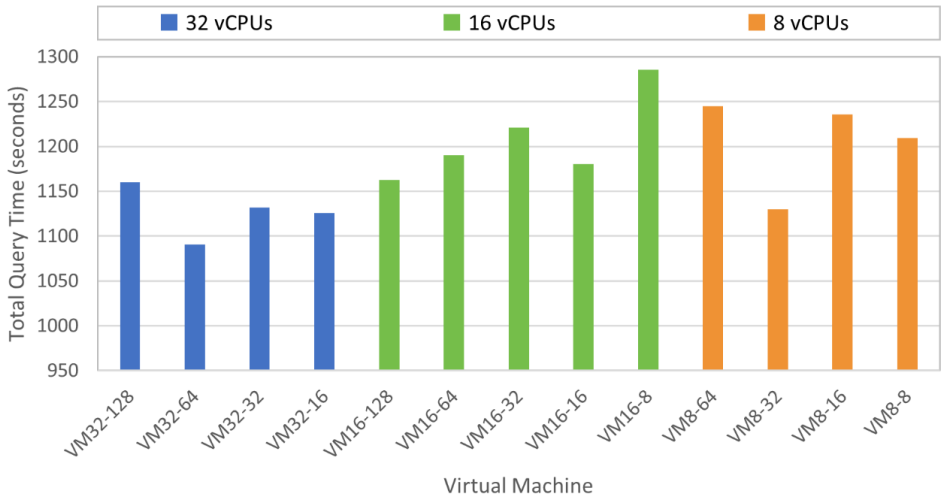

Figure 3: Total query time for GCP virtual machines, which had 1000-GB solid-state drive (SSD) storage. The virtual machines are color-coded by the number of virtual CPUs (vCPUs): blue for 32 vCPUs, green for 16 vCPUs, and orange for 8 vCPUs. Virtual machines color-coded with the same color have different memory sizes. See Table 3 for the hardware specifications of the virtual machines.

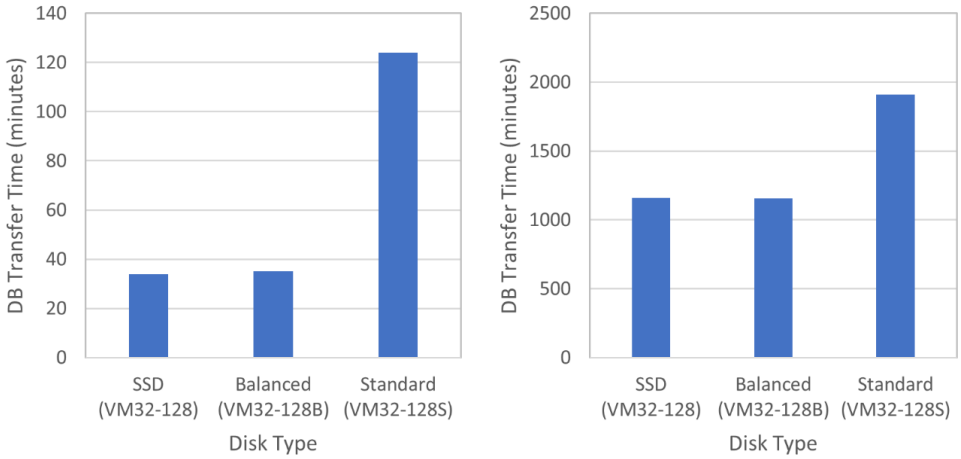

Figure 4: Effects of the data storage type upon the Virtuoso DB transfer time (in minutes) and the total query time (in seconds). The disk types, "SSD", "Balanced", and "Standard", mean "solid-state drive", "balanced persistent disk", and "Standard persistent disk", respectively.

One such optimization relevant for many of the 16 test queries is the handling of very large predicates. Specifically, PubChemRDF has three predicates with over 3 billion triples each: rdf:type, sio:SIO_000300, and sio:SIO_000008. They are central to the schema of PubChemRDF and used in all but one of the 16 test queries. QLever handles these predicates gracefully and reads only those parts of it that are relevant for the query. Virtuoso, on the other hand, has problems figuring this out and often has to read the complete predicate, which explains the much high query processing times for queries 4, 5, 10, and 16 in Table 5.

Further, as shown in Table 5 and Figure 6, Virtuoso's query times are considerably slower on HDD and NFS compared to SSD (145% and 34% slower, respectively), whereas QLever's query times are only 28% slower on both HDD and NFS compared to SSD. The main reasons for that are QLever's better locality of access and its more compact database. When ignoring queries 4, 5, 10, and 16, Virtuoso SSD is comparable to QLever SSD. However, when NFS and HDD are considered, even when ignoring queries 4, 5, 10, and 16, QLever is more than 4 times faster. This may suggest greater flexibility with

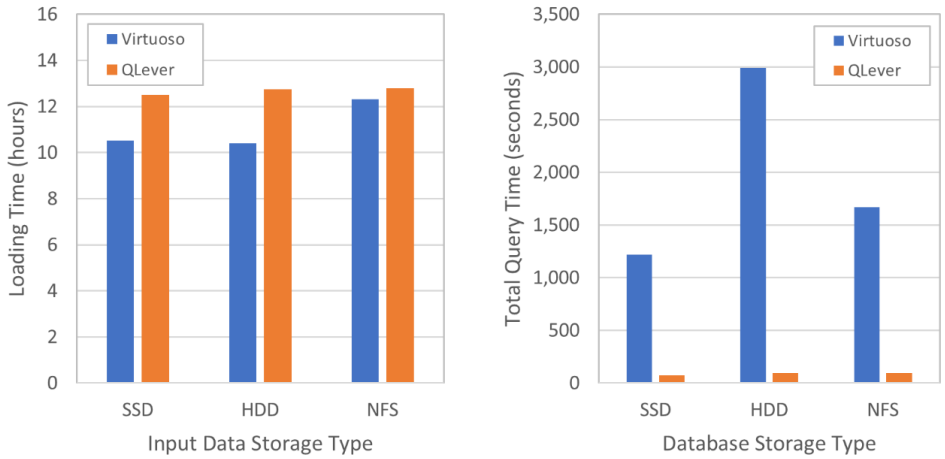

Figure 5: The query time (in seconds) for the 16 SPARQL queries performed against Virtuoso and QLever databases installed on an in-house Linux machine. Three data storage types were tested: solid-state drive (SSD, top), hard disk drive (HDD, middle), and networking file system (NFS, bottom).

QLever to explore optimal conditions within a given data center to balance speed and cost (e.g., no need for the additional step of making a local copy of QLever index files as a network storage may be sufficient or suggest that, on balance, SSD is worth the price if increased performance is desired).

When considering the total query time for all queries except for query 3, SSD is 17% faster than HDD and 36% faster than NFS for Virtuoso. In addition, HDD is about 16% faster than NFS, again when ignoring query 3. The PubChemRDF query 3 use case was repeated several times on HDD (as well as on NFS and SSD) for Virtuoso. The HDD appears to be somehow I/O bound within Virtuoso in a way that did not effect SSD or NFS, potentially reflecting an unfortunate query plan requiring substantially more I/O operations for query 3, likely resulting in a large number of cache misses (within Virtuoso, Linux, and the hard drive caches), and may reflect a major disadvantage to use HDD with Virtuoso at the full size of PubChemRDF, as some queries may be slower than expected due to the sheer size of the index files.

3.6. Installation guide for Virtuoso pre-loaded with PubChemRDF

A step-by-step guide (https://pubchem.ncbi.nlm.nih.gov/docs/rdf-cloud) was created to assist users in deploying a Virtuoso database pre-loaded with PubChemRDF data on a local machine or virtual machine on the Google Cloud Platform. It is noteworthy that it can take hours to deploy the Virtuoso database with all PubChemRDF data, which is hundreds of gigabytes in size. Therefore, a much smaller Virtuoso database containing a small subset of PubChemRDF data was created to enable users to quickly test-drive the installation process outlined in the step-by-step guide prior to working with the full-size version.

4. Conclusions

This paper describes PubChemRDF data being made available preloaded within the Virtuoso and QLever databases. Consideration and testing within the Google Cloud Platform (GCP) was performed using a set of sixteen use case queries found on the PubChemRDF Help page giving a sense of the speed and cost involved, including the effect of memory, CPU, and I/O type. Using the Virtuoso data system preloaded with PubChemRDF data the query speed on 15 virtual machines with different hardware specifications was performed. In general, the more vCPUs resulted in a marginally faster query result, while the impact of the memory size was not obvious. Concurrency (running multiple simultaneous queries) was not tested and having more CPUs available may provide a much greater benefit in that scenario. The

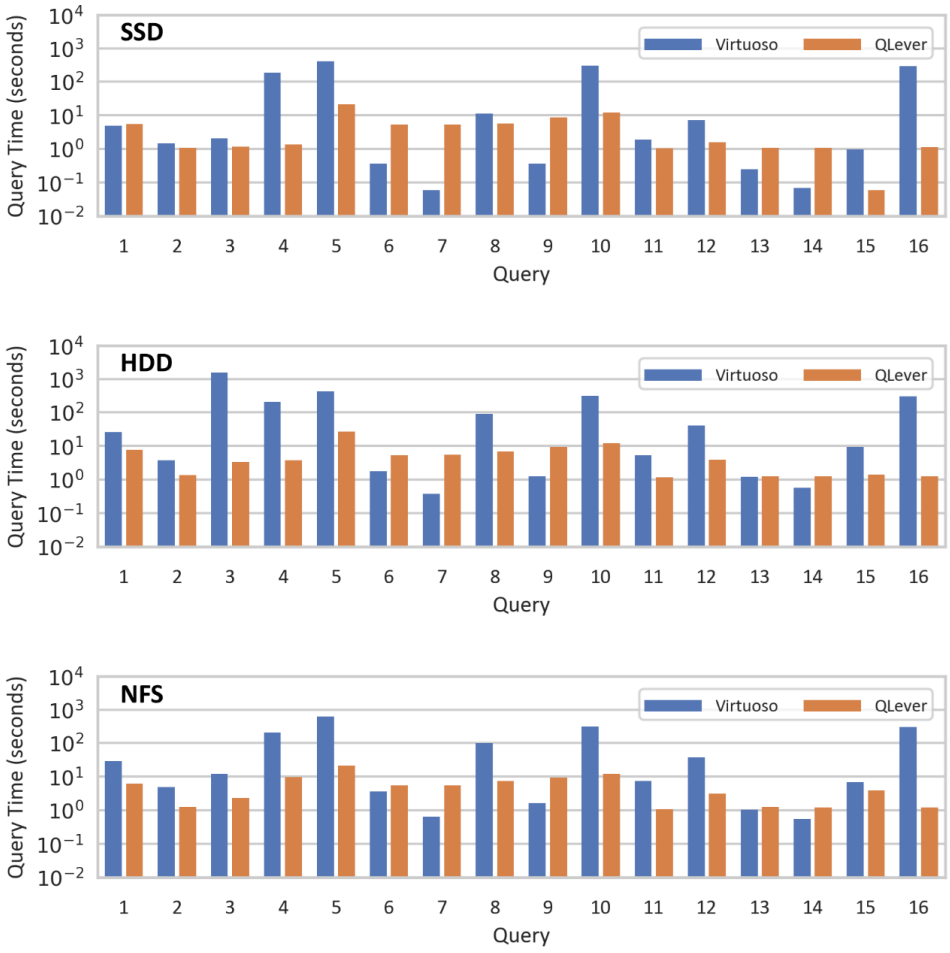

Figure 6: The query time (in seconds) for the 16 SPARQL queries performed against Virtuoso and QLever databases installed on an in-house Linux machine. Three data storage types were tested: solid-state drive (SSD, top), hard disk drive (HDD, middle), and networking file system (NFS, bottom).

virtual machine with the balanced persistent disk on GCP appears to be a good cost-effective alternative to a machine with an SSD disk.

While the Virtuoso database was used, there are other database applications that can load and exploit PubChemRDF data. In this work, QLever was shown to give substantially reduced total query times (compared to Virtuoso) when it was tested on a local machine. In addition, the index files were about 2.5 times smaller. QLever is even faster when using SSD (by more than 25%, compared to HDD and SSD). This result warrants further investigation of the use of QLever as an alternative to Virtuoso.

The PubChemRDF in the Cloud project aims to make PubChemRDF data readily available on major cloud computing resources, or in user-accessible local data centers, allowing users to exploit this data more readily and at a cheap cost (in time and money). While the work presented in this paper is focused only on GCP, we expect the results to be amenable to other virtualized and cloud computing resources, such as Amazon Web Service (AWS) and Mircosoft Azure.

Acknowledgments

This work was supported in part by the National Center for Biotechnology Information of the National Library of Medicine (NLM), National Institutes of Health. It was also supported in part by the Office of Data Science Strategy (ODSS), National Institutes of Health. We would like to acknowledge the DBCLS BioHackathon series and its participants for useful discussions.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] C. Peng, F. Xia, M. Naseriparsa, F. Osborne, Knowledge graphs: Opportunities and challenges, Artificial Intelligence Review 56 (2023) 13071–13102. URL: https://doi.org/10.1007/s10462-023-10465-9. doi:10.1007/s10462-023-10465-9.

[2] M. Kejriwal, Knowledge graphs: A practical review of the research landscape, Information 13 (2022). URL: https://www.mdpi.com/2078-2489/13/4/161. doi:10.3390/info13040161.

[3] B. Bushman, D. Anderson, G. Fu, Transforming the medical subject headings into linked data: Creating the authorized version of mesh in rdf, Journal of Library Metadata 15 (2015) 157–176. URL: https://doi.org/10.1080/19386389.2015.1099967. doi:10.1080/19386389.2015.1099967, pMID: 26877832.

[4] N. Redaschi, U. Consortium, Uniprot in rdf: Tackling data integration and distributed annotation with the semantic web, Nature Precedings (2009). URL: https://doi.org/10.1038/npre.2009.3193.1. doi:10.1038/npre.2009.3193.1.

[5] J. Piñero, J. M. Ramírez-Anguita, J. Saüch-Pitarch, F. Ronzano, E. Centeno, F. Sanz, L. I. Furlong, The DisGeNET knowledge platform for disease genomics: 2019 update, Nucleic Acids Res. 48 (2020) D845–D855.

[6] R. Ranzinger, K. F. Aoki-Kinoshita, M. P. Campbell, S. Kawano, T. Lütteke, S. Okuda, D. Shinmachi, T. Shikanai, H. Sawaki, P. Toukach, M. Matsubara, I. Yamada, H. Narimatsu, Glycordf: an ontology to standardize glycomics data in rdf, Bioinformatics 31 (2014) 919–925. URL: https://doi.org/10.1093/bioinformatics/btu732. doi:10.1093/bioinformatics/btu732.

[7] K. Kozaki, Y. Yamagata, R. Mizoguchi, T. Imai, K. Ohe, Disease compass– a navigation system for disease knowledge based on ontology and linked data techniques, Journal of Biomedical Semantics 8 (2017) 22. URL: https://doi.org/10.1186/s13326-017-0132-2. doi:10.1186/s13326-017-0132-2.

[8] A. Waagmeester, M. Kutmon, A. Riutta, R. Miller, E. L. Willighagen, C. T. Evelo, A. R. Pico, Using the semantic web for rapid integration of wikipathways with other biological online data resources, PLOS Computational Biology 12 (2016) 1–11. URL: https://doi.org/10.1371/journal.pcbi.1004989. doi:10.1371/journal.pcbi.1004989.

[9] S. Jupp, J. Malone, J. Bolleman, M. Brandizi, M. Davies, L. Garcia, A. Gaulton, S. Gehant, C. Laibe, N. Redaschi, S. M. Wimalaratne, M. Martin, N. Le Novère, H. Parkinson, E. Birney, A. M. Jenkinson, The ebi rdf platform: linked open data for the life sciences, Bioinformatics 30 (2014) 1338–1339. URL: https://doi.org/10.1093/bioinformatics/btt765. doi:10.1093/bioinformatics/btt765.

[10] J. Hastings, G. Owen, A. Dekker, M. Ennis, N. Kale, V. Muthukrishnan, S. Turner, N. Swainston, P. Mendes, C. Steinbeck, Chebi in 2016: Improved services and an expanding collection of metabolites, Nucleic Acids Research 44 (2015) D1214–D1219. URL: https://doi.org/10.1093/nar/gkv1031. doi:10.1093/nar/gkv1031.

[11] B. Zdrazil, E. Felix, F. Hunter, E. J. Manners, J. Blackshaw, S. Corbett, M. de Veij, H. Ioannidis, D. M. Lopez, J. Mosquera, M. Magarinos, N. Bosc, R. Arcila, T. Kizilören, A. Gaulton, A. Bento, M. Adasme, P. Monecke, G. Landrum, A. Leach, The chembl database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods, Nucleic Acids Research 52 (2023) D1180–D1192. URL: https://doi.org/10.1093/nar/gkad1004. doi:10.1093/nar/gkad1004.

[12] S. M. Wimalaratne, P. Grenon, H. Hermjakob, N. Le Novère, C. Laibe, BioModels linked dataset, BMC Syst. Biol. 8 (2014) 91.

[13] G. Fu, C. Batchelor, M. Dumontier, J. Hastings, E. Willighagen, E. Bolton, PubChemRDF: towards the semantic annotation of PubChem compound and substance databases, J. Cheminform. 7 (2015) 34.

[14] Q. Li, S. Kim, L. Zaslavsky, T. Cheng, B. Yu, E. E. Bolton, Resource description framework (rdf)

modeling of named entity co-occurrences derived from biomedical literature in the pubchemrdf, CEUR-WS.org, 2023, pp. 32–41. URL: https://ceur-ws.org/Vol-3415/paper-4.pdf.

[15] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, L. Zaslavsky, J. Zhang, E. E. Bolton, Pubchem 2023 update, Nucleic Acids Research 51 (2022) D1373–D1380. URL: https://doi.org/10.1093/nar/gkac956. doi:10.1093/nar/gkac956.

[16] H. Bast, B. Buchhold, Qlever: A query engine for efficient sparql+text search, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 647–656. URL: https://doi.org/10.1145/3132847.3132921. doi:10.1145/3132847.3132921.

[17] H. Bast, J. Kalmbach, T. Klumpp, C. Korzen, Knowledge Graphs and Search, 1 ed., Association for Computing Machinery, New York, NY, USA, 2024, p. 231–281. URL: https://doi.org/10.1145/3674127.3674134.