Stardog Benchmarking Report

Prepared By: Stardog Union
Last Updated: 06/20/2023

SECTION 1: SUMMARY

This report summarizes the results of performance benchmarks performed against Stardog 9.0.1 over a wide range of datasets and workloads. We have used public datasets and benchmarks (BSBM, LUBM, LSQB, YAGO, Wikidata) and followed the predefined benchmarking protocols where available. The benchmarks cover transactional queries (read and update) over local and virtualized graphs, reasoning queries, path queries, and bulk loading and measure the impact of concurrent users on the overall system including the High Availability cluster. We present both latency and throughput metrics for these benchmarks. We also compare the performance of Stardog 9.0.1 with the latest version of a commercial RDF Graph Database where applicable. This RDF Graph Database performs reasoning at data load time unlike Stardog which performs reasoning at query time and we discuss the implications of this difference in this report.

Here we provide some highlights of our key findings over these benchmarks:

- Stardog can load large datasets with **billions of triples** at speeds of **1 million triples per second** on commodity hardware. This speed is not specific to one dataset and the same speeds are sustained over different datasets.
- Stardog's bulk loading speed is **3-4 times faster** than the RDF Graph Database when reasoning is disabled for the RDF Graph Database at data load time and more than **10 times faster** than the RDF Graph Database when reasoning is enabled.
- Stardog can complete mixed read and update benchmark for BSBM dataset with **1 billion triples** with an average query execution time of **10 milliseconds** when queries are executed sequentially and under **70 milliseconds** when 64 queries are executed concurrently.
- Stardog consistently outperforms the RDF Graph Database at every concurrency level of the BSBM benchmark where Stardog is **50 times faster** with 64 concurrent users, **10 times faster** with 128 concurrent users, and **5 times faster** with 256 concurrent users.
- Using a 3 node Stardog HA cluster improves query latency **8 to 10 times** compared to a single Stardog server at high concurrency levels of the BSBM benchmark and improves query throughput by **2.5 times**.
- Stardog can complete the Wikidata Graph Pattern Benchmark against **16.7 billion triples** with query execution **under 100 milliseconds** for 92% of the queries and **under 1 second** for 99% of the queries in the benchmark.
- Stardog outperforms the RDF Graph Database for every query in the LSQB benchmark and for the majority of the queries Stardog is **3 to 9 times** faster.
- Stardog can scale to **1 trillion triples** by utilizing virtual graphs in a hybrid multi cloud setup with query answering times of **1 second or less** even for queries that require reaching out to multiple data sources. The Stardog setup is **98% cheaper** to run than the only competitor who could reach this scale.
- For reasoning queries, against the LUBM dataset of **133M triples,** Stardog completes 9 of the 14 queries in the benchmark under **100 milliseconds** even when reasoning is done at query time.
- Stardog reasoning performance is **competitive** with the RDF Graph Database for most of the queries in LUBM benchmark and is **significantly faster** for some of the queries which proves that query time reasoning is performant even at large scales while allowing **10 times faster** data loading times as explained above.

Please see the details of these findings in the following sections.

SECTION 2: BENCHMARKS

We present the detailed results for each benchmark in a separate section. The configuration details and the parameters used during the benchmarks are presented at the end. All the scripts used for testing are available upon request to reproduce these results independently.

Bulk Data Load Benchmark

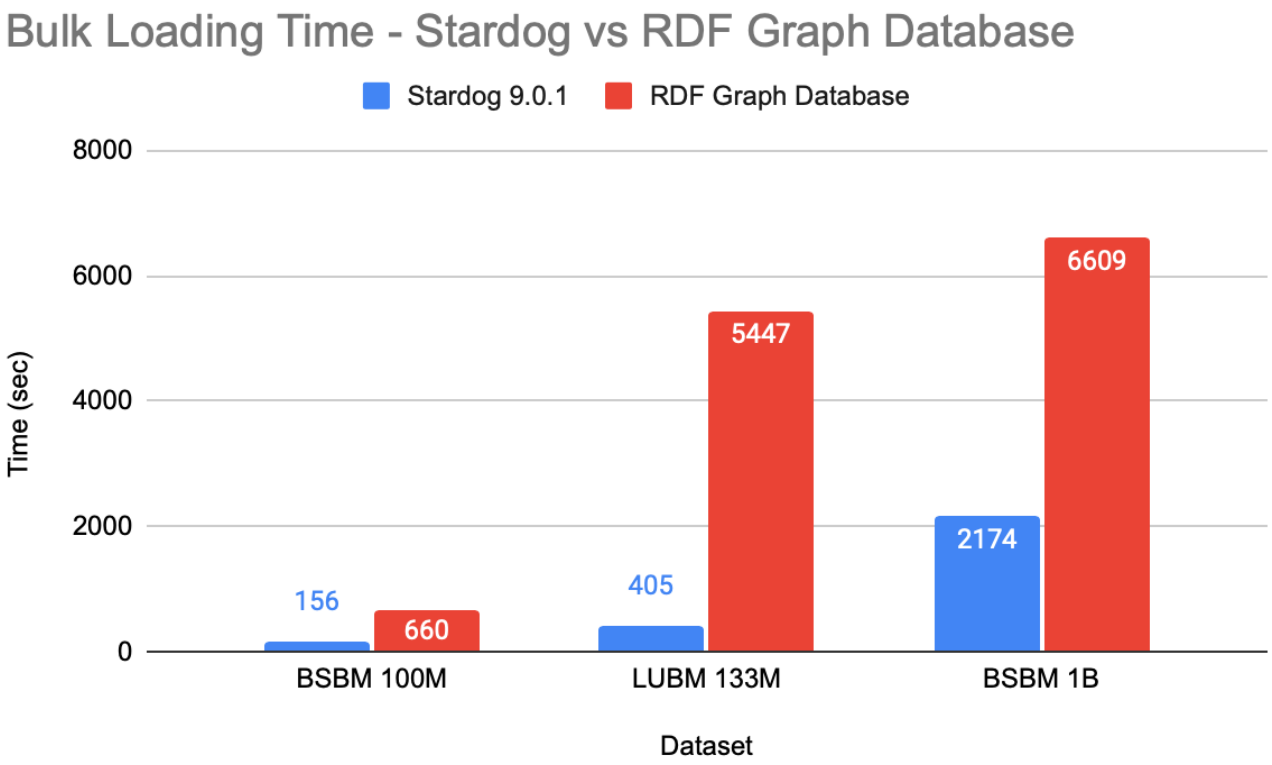
In this section we look at the performance of bulk data load which is used for initially importing large amounts of data into Stardog. In addition, Stardog uses this optimized data loading mechanism when creating distributed caches for virtual graphs. As mentioned at the beginning, the performance testing relies heavily on the specification of the hardware used. The number of CPUs, the available memory and the type of disk all impact the performance. For this reason, we provide loading times we have collected on different hardware configuration as summarized in the following table:

Dataset	Triples	EC2 Instance	Cost per hour	Disk	Loading Time (H:MM:SS)	Loading Speed (M triples/sec)
BSBM	100M	c5d.9xlarge	$1.728	NVMe	0:01:31	1.094
BSBM	100M	c5.4xlarge	$0.768	gp2	0:02:36	0.599
LUBM (1K)	133M	c5d.9xlarge	$1.728	NVMe	0:02:17	0.970
LUBM (1K)	133M	c5.4xlarge	$0.768	gp2	0:03:57	0.536
LDBC (SF10)	478M	c5d.9xlarge	$1.728	NVMe	0:07:37	1.045
BSBM	1B	c5d.9xlarge	$1.728	NVMe	0:15:58	1.043
BSBM	1B	m5.4xlarge	$0.768	gp2	0:21:29	0.775
BSBM	1B	r5.2xlarge	$0.504	gp2	0:36:14	0.459
LUBM(40K)	5.5B	c5d.12xlarge	$2.304	NVMe	1:16:09	1.168
BSBM	10B	c5d.12xlarge	$2.304	NVMe	2:56:41	0.940
Wikidata	16.7B	r5.8xlarge	$2.016	gp2	9:28:20	0.484

The c5d family of EC2 instances are CPU-optimized with local NVMe SSD disks that provide the fastest loading times. As shown in the table, Stardog achieves 1M triples per second loading throughput across different datasets with this configuration. As expected, the loading speed reduces

when less powerful machines are used but even at these levels 0.5M triples per second loading speed can be achieved even with significantly cheaper machines.

The following chart shows Stardog loading times compared with a commercial RDF Graph Database on some of these datasets where benchmarks for both systems were run on r5.2xlarge with identical memory settings. The chart shows the loading times expressed in seconds to make comparison easier.



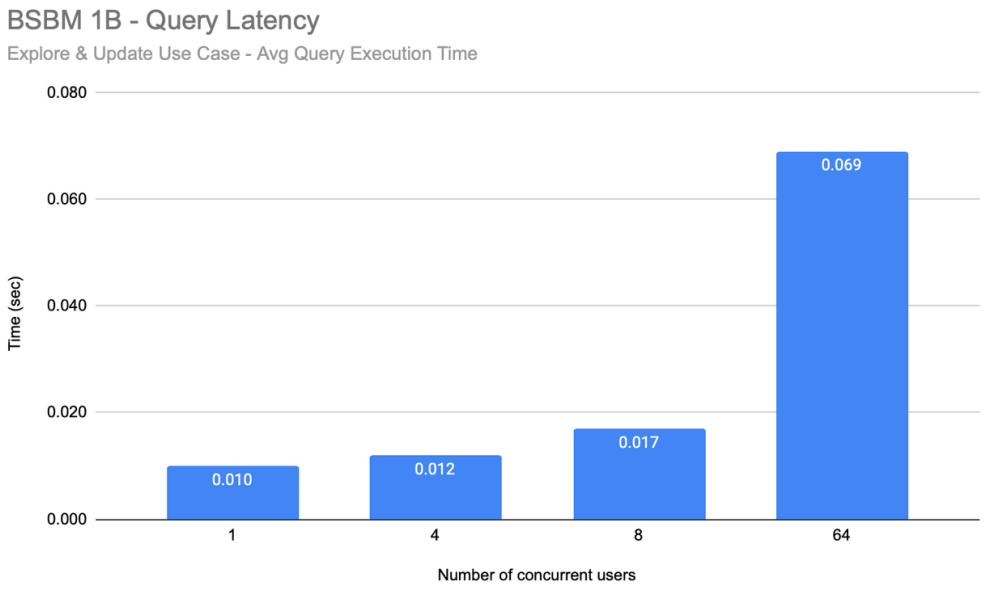
Stardog is 3 to 4 times faster while loading the BSBM dataset and more than 10 times faster for the LUBM dataset. To put the numbers in perspective the total loading time for Stardog was 6min45sec whereas the RDF Graph Database spent 1h30min47sec. The bigger difference in LUBM is due to the fact that reasoning is required for the benchmark as explained below so the RDF Graph Database needs to compute and materialize all the inferences upfront at data loading time. In contrast, Stardog computes inferences on-the-fly at query time as needed so there is no additional overhead during bulk loading or transactional updates.

Mixed Read and Update Benchmark

Berlin SPARQL Benchmark (BSBM) is built around an e-commerce use case in which a set of products offered by different vendors and consumers have posted reviews about products. The Explore & Update use case in this benchmark illustrates the search and navigation pattern of a consumer looking for a product while updates to the products and reviews are being applied. These are targeted queries that should be answered very quickly.

The benchmark consists of read and update query templates that are instantiated with different values for each execution. There are 11 read query templates and 2 update query templates grouped into query mixes with 30 queries. Each execution of the query mix uses randomized instantiations of these query templates.

The following chart shows the geometric average of query execution times against the BSBM dataset with 1 billion triples with increasing numbers of concurrent users. We use the parameters employed in the past official BSBM evaluations where the number of concurrent users was 1, 4, 8, and 64.

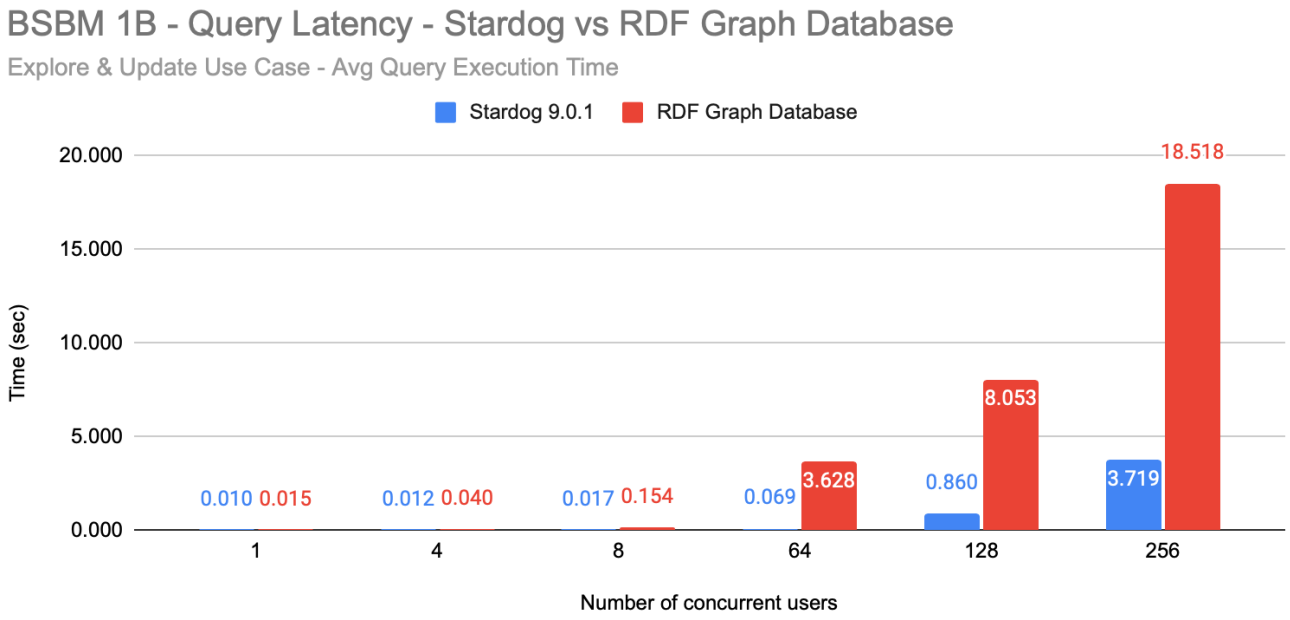


As the results show, the average query execution time ranges from only 10 milliseconds when there is a single user to 69 milliseconds when there are 64 concurrent users. The main reason for increasing query times is the fact that the Stardog server runs on a machine with 8 vCPUs. As we present in the next section, the query execution times under heavy load can be reduced by deploying Stardog's High Availability Cluster which results in higher query throughput.

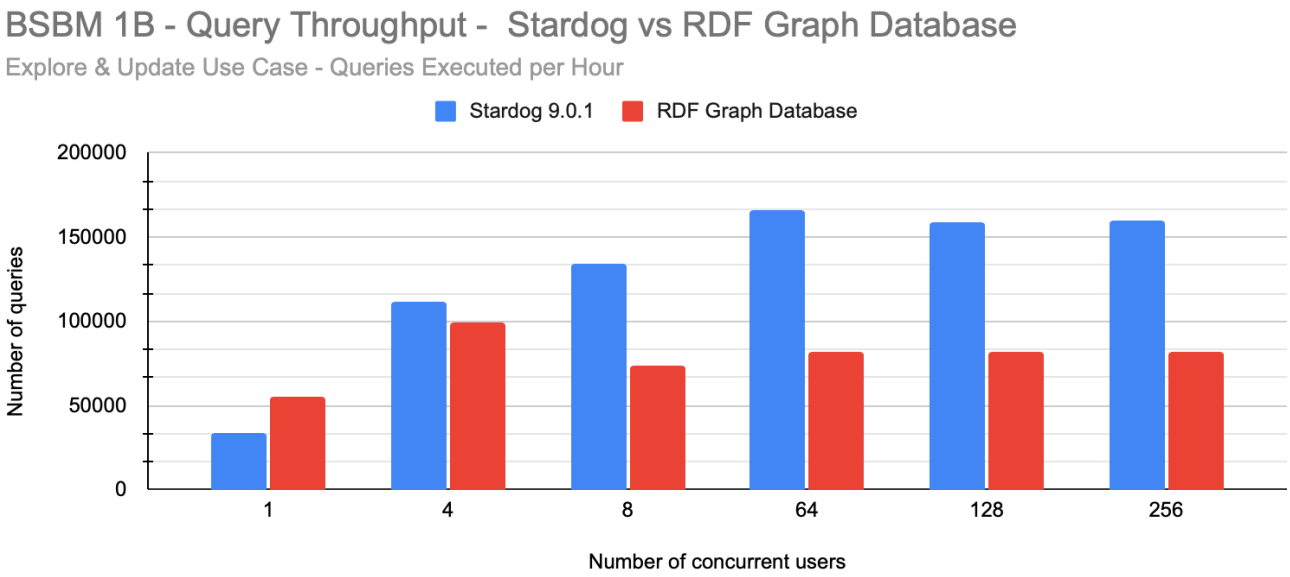
The following table shows the throughput results for this benchmark measured as the number of queries executed per hour (higher is better). The results show that overall query throughput continues to increase with increasing number of concurrent users even though the average query execution times increase as shown in the above table.

Concurrent Users	1	4	8	64
Queries per Hour	33,314.83	111,225.54	133,829.00	165,513.28

We next compare our results with the RDF Graph Database in the same benchmark run on the exact same hardware. For this comparison we go beyond the settings used in past evaluations and test both systems with also 128 and 256 concurrent users. The following chart shows the comparison of average query execution times for both systems. As the results show Stardog consistently outperforms the RDF Graph Database at every concurrency level but the difference between systems becomes more visible as the concurrency level increases. The RDF Graph Database is 50 times slower with 64 concurrent users, 10 times slower with 128 concurrent users, and 5 times slower with 256 concurrent users.



The breakdown of execution times for query templates shows that Stardog is faster than the RDF Graph Database on 10 of the query templates and slower on the remaining 3 templates. If we look at the query throughput numbers, we see that Stardog outperforms the RDF Graph Database on all concurrency levels except for the lowest concurrency level.

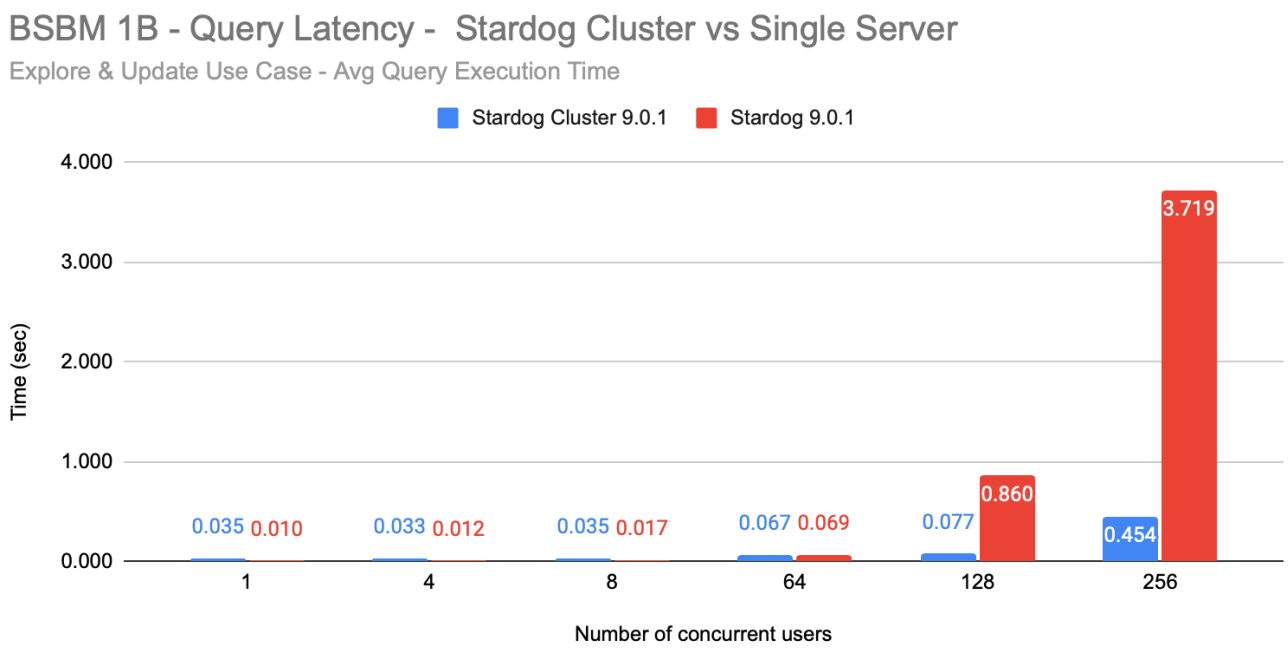


The RDF Graph Database throughput degrades when the concurrency level goes over 4 whereas Stardog throughput continues to increase with the load. Stardog throughput is 2 almost exactly 2 times higher than the RDF Graph Database at concurrency levels 64, 128, and 256. As we show in the next section Stardog throughput can be further improved by using the High Availability cluster.

High Availability Cluster Benchmark

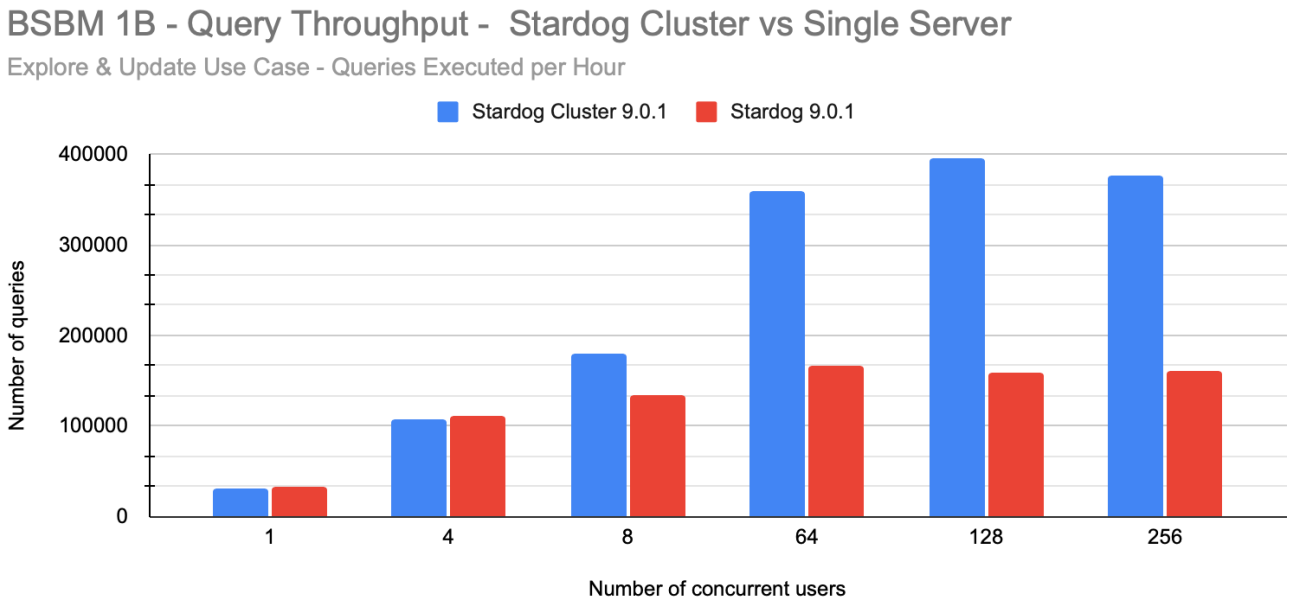
The Stardog offers a High Availability (HA) Cluster with several built-in capabilities for hot standby nodes, geographically redundant nodes, cache nodes for big data environments, and the ability to scale up and down. The HA cluster automatically handles data replication across multiple Stardog servers automatically ensuring all Stardog servers are synchronized at all times and self-heals if there is a failure.

Using an HA cluster introduces some small performance overhead compared to using a single Stardog server because a load balancer is needed to distribute the queries over multiple servers and data changes need to be replicated but as we show in this section the increased capacity offered by multiple Stardog servers more than offsets these factors.



In order to measure the performance of the Stardog HA cluster we repeated the BSBM1B Explore and Update benchmark against a cluster with 3 Stardog servers. The following chart shows the comparison of average query execution times for a single Stardog server and a 3-server Stardog cluster.

When the system is under low load (8 or less concurrent users) we see that the single server offers lower latency due to the reasons mentioned above. But as the load on the overall system increases, we see the benefits of the HA cluster clearly. In the most extreme case of 256 concurrent users the Stardog HA cluster outperforms the single Stardog server significantly. Tripling the number of Stardog servers results in 8 to 10 times faster query execution times because once a server is oversubscribed the queuing and context switching times result in additional performance degradation especially for fast queries.



Another way to compare the HA cluster and the single server performance is by looking at the throughput numbers calculated as the number of queries executed per hour as shown in the next chart. The single Stardog server exhibits slightly better query throughput when the concurrency load is low, but the HA cluster throughput is better at higher load.
Notice that the query throughput for the cluster is about 2.5 times better than the single server under heavy load. The difference here is not as high as query execution times mainly because slower queries in the mix have more effect on the overall throughput whereas each query contributes evenly for average execution times.

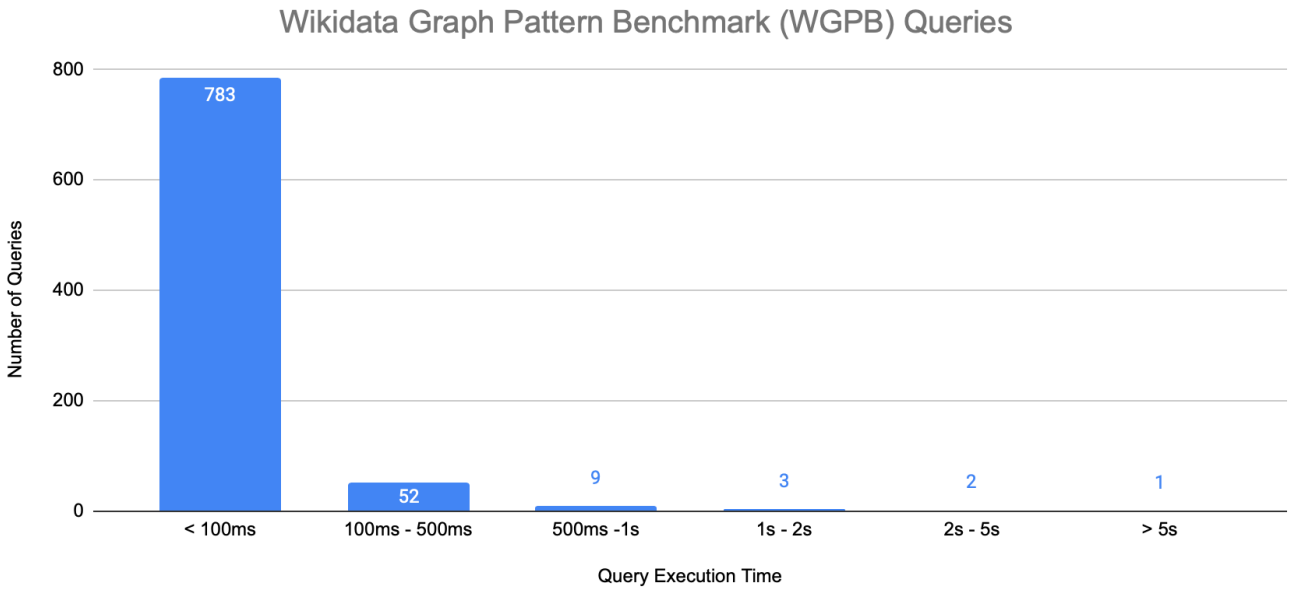
Overall, these results show that Stardog HA cluster not only improves the system robustness by eliminating single point of failure but it also improves the overall performance and throughput especially when the system is under heavy load.

Large Scale Read Benchmark

Wikidata is a free and open knowledge base that serves as the central storage for the structured data of Wikipedia. It is one of the largest publicly available RDF datasets that is constantly growing and contained 16.7 billion triples at the time of our benchmarking. Wikidata Graph Pattern Benchmark (WGPB) comes with 850 queries that test different join patterns.

The following chart shows the histogram of query execution times for the 850 WGPB queries. As the results show, the response times are less than 100 milliseconds for 92% of the queries and less than 1 second for 99% of the queries in the benchmark, highlighting that Stardog is great for both interactive real-time applications, and large reporting and data processing solutions even at

exceptionally large scales.

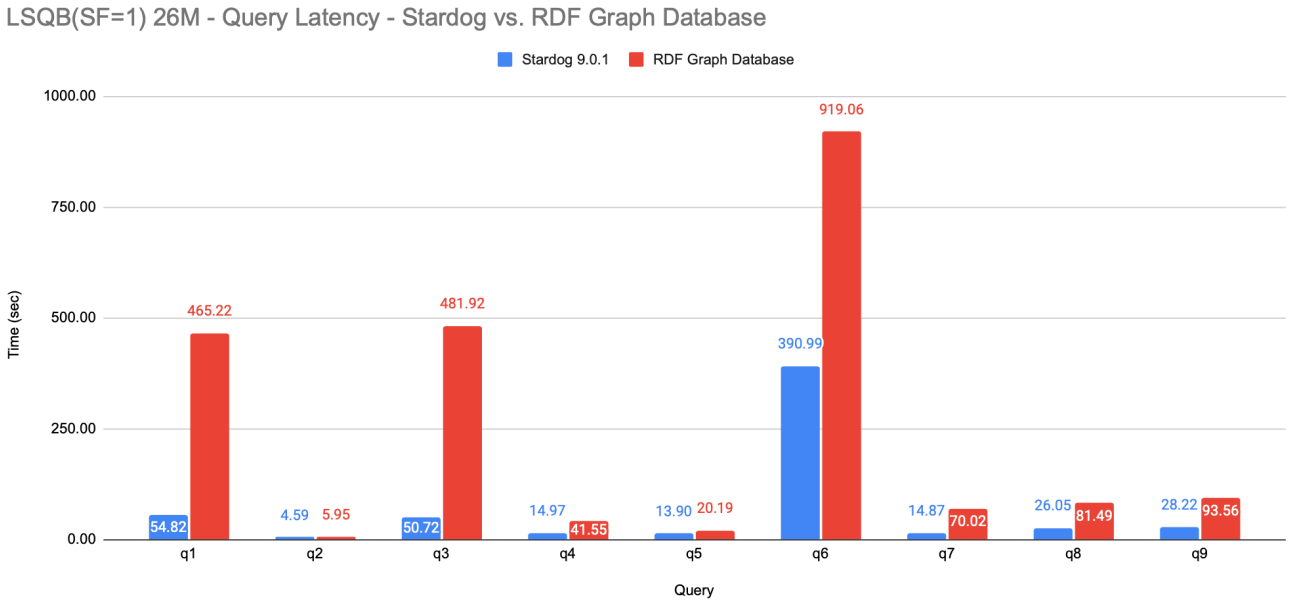


Subgraph Matching Benchmark

Labelled Subgraph Query Benchmark (LSQB) is a benchmark from Linked Data Benchmarking Council (LDBC) solely focusing on subgraph matching. The primary goal of this benchmark is to test the query optimizer (join ordering, join types selection) and the execution engine (join performance, support for worst-case optimal joins) of graph databases. The dataset is based on a representation of social networks with people, forums, messages, and locations.

There are 9 queries in the LSQB benchmark that require the query engine to process large amounts of data even though the dataset itself is relatively small. At scale factor 1, there are only 26M triples in the dataset but the queries may end up matching nearly half a billion distinct subgraphs. For this reason, the execution time of these queries are measured in seconds and not milliseconds.

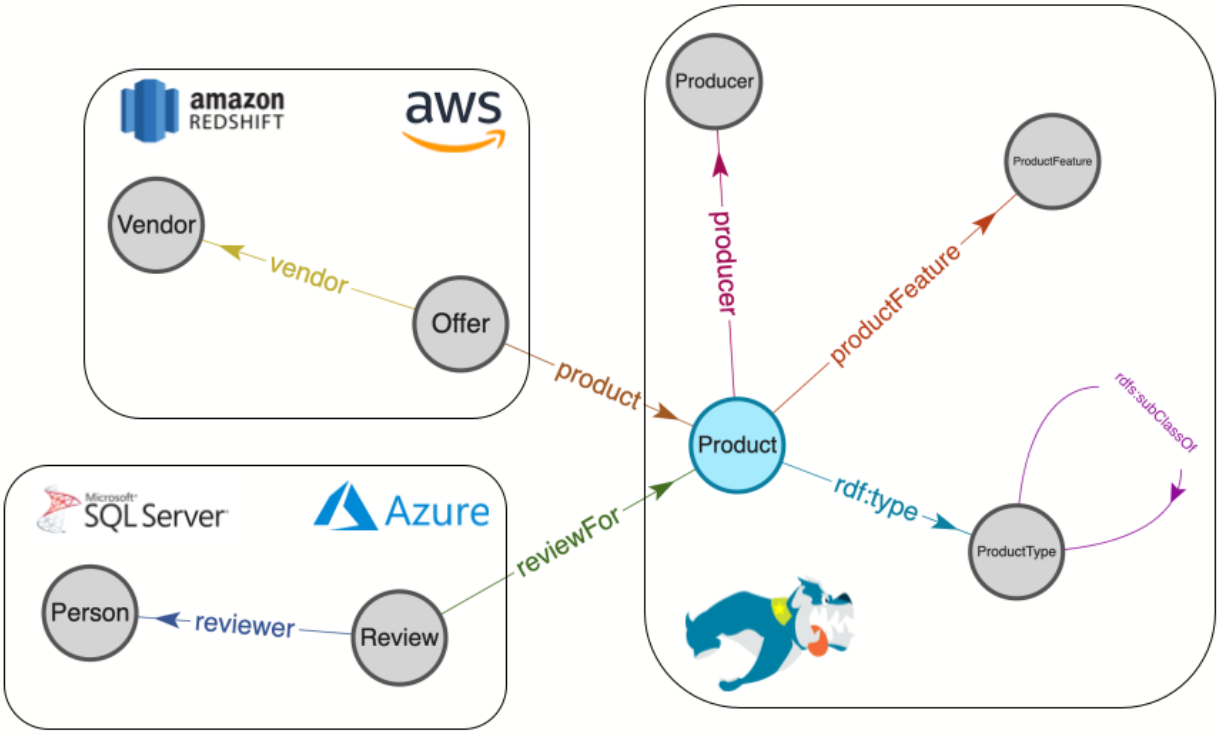
The following chart shows the average query execution times for LSQB queries for Stardog and the RDF Graph Database. As the results show, Stardog is significantly faster on all queries and the difference between two systems is most noticeable on queries that take more time. For a majority of the queries Stardog is 3 to 9 times faster than the RDF Graph Database.



Virtual Graph Benchmark

All the existing SPARQL benchmarks solely focus on data that has been fully materialized as an RDF graph in a local index. But Stardog's unique virtualization capability allows it to query external data sources as virtualized graphs. In order to test the Stardog performance over hybrid multicloud data sources an independent benchmark was carried out by McKnight Consulting over a massive Trillion Triple Knowledge Graph on a modified version of the BSBM benchmark**.** You can find more details about this benchmark in the Trillion Edge Knowledge Graph report. We summarize the main findings of the report in the remainder of this section.

In this version of the benchmark the BSBM data has been partitioned into three parts: products, offers, and reviews. Each part was loaded into a different system: products into Stardog, offers into Amazon Redshift in AWS and reviews into SQL Server in Azure:



The size of the data distributed over the three data sources looks as follows:

Data Source	Graph Type	Number of Entities	Number of Triples	Data size
Stardog	Materialized	8.8 billion	115 billion	6.1TB
SQL Server	Virtualized	30 billion	220 billion	4.5TB
Redshift	Virtualized	57 billion	660 billion	2.8TB

The following table shows the average query execution times for each query template in the benchmark. Since most of the data is stored in external data sources, we have used the Explore use case of BSBM without the updates. We followed the same BSBM benchmarking protocol with randomized query mixes so no same query is executed twice to avoid caching query results. Average query execution time for each query has been one second or less.

Queries	Average Number of Results	Average Execution Time (sec)
1	10.0	0.445
2	18.3	0.010
3	9.8	0.769
4	10.0	1.107
7	10.9	0.049
8	3.4	0.034
9	6.0	0.017
10	1.3	0.015
11	10.0	0.015
12	8.0	0.022

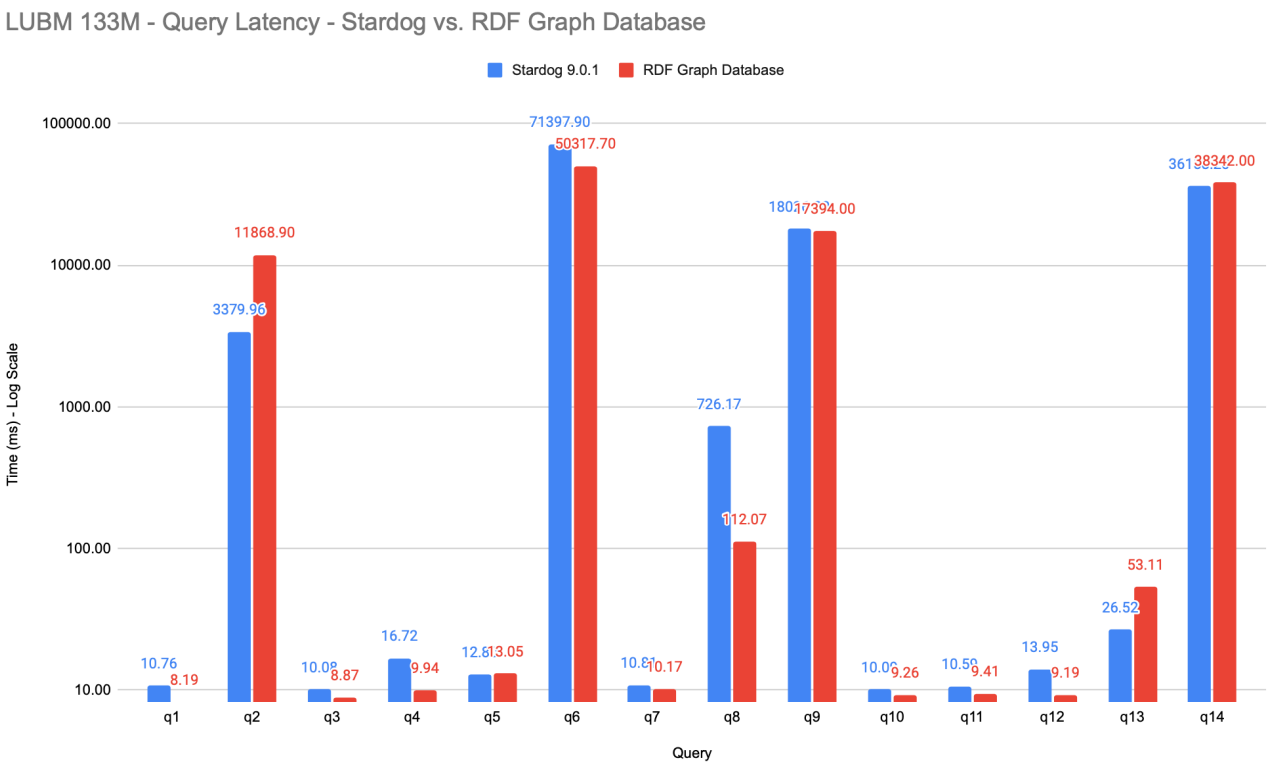
The report also discusses how these results compare with other products in scale and operational cost. First, many products do not have any results close to this scale; the highest scale benchmark results published by Ontotext GraphDb is 17 billion, Neo4j, 14 billion, TigerGraph, 67 billion. In addition, the report found Stardog 98% cheaper to run than the only competitor who could reach this scale, AnzoGraph. The configuration used in this benchmark is very different from all the other benchmarks discussed in this report so please refer to the McKnight Consulting report for details.

Reasoning Benchmark

All the benchmarks mentioned so far perform queries over explicit data whereas the Lehigh University Benchmark (LUBM) tests the performance of reasoning queries over an ontology. This benchmark is used and referenced frequently in the literature because it is the benchmark that has been around the longest.

The benchmark comes with a fixed set of 14 queries that requires inferences to be considered, and most return no results otherwise. Stardog uses query-time reasoning which means every input query is rewritten by taking the benchmark ontology into account to return the additional inferred results. This requires more work during query time but we show that the Stardog reasoning performance is as good as systems that compute inferences at data load time and materialize them upfront.

The following chart shows the average query execution times for the 14 LUBM queries using Stardog and the RDF Graph Database.. The characteristics of queries in the LUBM benchmark vary widely which is why there is a large gap between query execution times for different queries regardless of the system used which is why the following chart uses the log scale.



Both systems answer 9 of the benchmark queries very quickly under 100 milliseconds. Even though the RDF Graph Database is typically faster for these queries in around 10 milliseconds. There are also some queries where Stardog outperforms the RDF Graph Database. For example, query 2 involves traversing department hierarchy in universities and even though Stardog does this on the fly the query execution is still 3 times faster. Both systems take much longer time to answer query 6 simply because this query returns more than 10M results. The query-time reasoning involves more overhead for queries that return massive result sets. If the same query is executed with a limit then both systems execute the query in similar times.

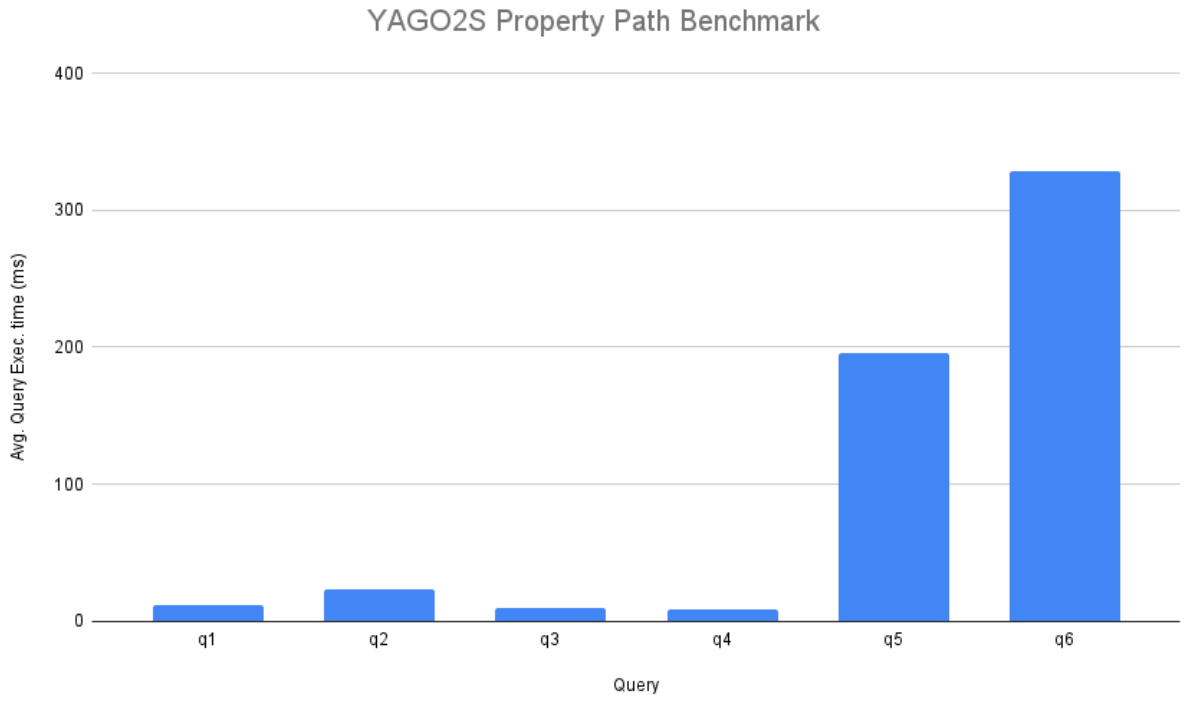
In summary, the RDF Graph Database does better than Stardog in this benchmark but the difference is typically minimal (a few milliseconds) and in general not large enough to justify the 10 times difference in data loading speeds mentioned in the first chapter.

Path Benchmark

YAGO is a large knowledge base with general knowledge about people, cities, countries, movies, and organizations that is continuously being updated. YAGO2S version of the dataset extracts and merges information from multiple sources such as Wikipedia, Word-Net, Geonames, the Universal WordNet, and WordNet Do- mains. The resulting dataset has 33.6M triples. The YAGO2S property path benchmark tests the performance of path queries that require the query engine to recursively

traverse the links in the graph to answer queries which are not covered in the other benchmarks. The benchmark comes with 6 queries that test different navigational patterns over locations.

The following chart shows the average query execution times of Stardog for the queries in the YAGO2S benchmark. The results show average execution time for these queries are extremely low with the hardest query taking slightly more than 300ms.



SECTION 3: CONFIGURATION AND SETUP

All the experiments have been run on an EC2 instance type r5.2xlarge (8 cores, 64GB RAM) using Stardog version 9.0.1 unless otherwise specified in the above sections. The gp2 general purpose SSD disks were used as the home directory to store the databases. Stardog memory uses a combination of Java heap memory and native memory, so its memory was configured as `-Xmx20g -XX:MaxDirectMemorySize=28g`.

All benchmarks except the bulk loading benchmarks use default Stardog options. For bulk loading, Stardog memory mode has been set to BULK_LOAD in stardog.properties.

All the benchmarks for the RDF Graph Database were run on the exact same hardware configuration as Stardog. The RDF Graph Database uses only Java heap so its memory was configured with the option "-Xms 48g -Xmx 48g" to have the same total memory limit as Stardog. Data loading into the RDF Graph Database was performed using an offline data loading tool provided by the RDF Graph Database and performs faster than online data loading. Thus, the loading command was used while the RDF Graph Database server was not running, and the server was started after the data was

loaded. In the case of the LUBM benchmark where reasoning is needed, the reasoning materialization was performed after the initial data loading. The additional time spent for reasoning is included in the data loading time in the above charts.

For the BSBM benchmark, we have used the procedure that has been employed in past official evaluations with the `bsbmdriver` that comes with bibm-0.7.8 library. As in the official evaluations we used 1, 4, 8, and 64 threads for concurrent users but additionally we ran tests with 128 and 256 threads. Before starting the benchmarks, a warmup run with 8 threads and 512 iterations were performed to get the system to a steady state. After that, the actual benchmark runs were run with separate warmup iterations with the same random seeds specified in the official evaluation. The same process was employed to run the BSBM benchmarks against the RDF Graph Database.

Number of Threads	Driver Parameters
1	-seed 9834533 -runs 25 -w 100
4	-seed 8188326 -mt 4 -w 8
8	-seed 9175932 -mt 8 -w 16
64	-seed 4187411 -mt 64 -w 128
128	-seed 8763581 -mt 128 -w 128
256	-seed 1218313 -mt 256 -w 256

For LUBM and YAGO benchmarks, a set of fixed queries were executed against each system using curl to send queries via the SPARQL protocol. The first execution of the query mix was discarded as a warmup iteration and three additional iterations were completed afterwards. The reported query execution numbers are the average of these runs.