Comparing Open Source and Proprietary Database Solutions for Querying Spatio-Temporal Data: SpaceTime vs Geomesa

Dinko Židić, Toni Mastelić
Ericsson Nikola Tesla
Split, Croatia
{dinko.zidic, toni.mastelic}@ericsson.com

***Abstract*—Spatio-temporal data represents a subset of high volume multidimensional data defined by its temporal dimension, i.e., a timestamp, as well as its (geo)spatial dimension, i.e., a location. Growth of Internet of Things and Earth Observation use cases that produce and utilize such data revealed that traditional relational databases as well as noSQL databases are not adequate, hence creating a demand for novel solutions. Recent studies that cover several spatio-temporal databases highlight Geomesa as the most prominent solution. The purpose of this paper is to expand the previous studies by executing performance and scalability benchmarks in order to compare Geomesa with a proprietary solution from Mireo, namely SpaceTime database. Performance benchmarks are used to evaluate the impact of data clustering and indexing techniques on response time, while scalability benchmarks are used to evaluate database distribution and resource usage. Results indicate that SpaceTime outperforms Geomesa in almost all benchmarks due to better CPU and memory utilization and resilience to slower network speeds, except the case when running on hard drive.**

***Index Terms*—spatio-temporal data, databases, benchmarking, Geomesa, SpaceTime**

I. Introduction

Spatio-temporal data is part of high-volume multidimensional data defined by a temporal dimension (timestamp), as well as a spatial dimension (location) [1]. Internet of Things [2] and Earth Observation [3] use cases produce and utilize such data for tackling problems such as global warming, earthquakes, urban planing, modern transportation, health care, agriculture and many more [1].

Classic relational databases along with noSQL databases are not adequate to handle such data types due to their high volume and multidimensional characteristics [1]. On one hand, handling spatio-temporal data is not straightforward due to the complexity of the data structures that require careful analysis of dimensions, together with the representation and manipulation of the data [4]. Therefore, its structural organisation and indexing technique are the most important factors that affect access performance to raw data [5]. Consequently, the main thing is to select a good indexing technique with low space requirements and accomplish optimal storage data clustering [5]. On the other hand, the scalability is highly affected by the clustering of high volume data due to the requirements for large storage space [2]. To fulfill those requirements spatio-temporal databases are commonly distributed to compensate single nodes inefficiency due to lack of parallelism, scalability and I/O bottleneck [1].

Simplest approach to index multidimensional data is utilizing spatial indexing techniques as crude form of spatio-temporal indexing, where time is treated as a new dimension, while other more complicated indexing techniques are only extensions of existing spatial methods [5]. Tamas Abraham and John F. Roddick [5] survey covers detailed concepts of spatio-temporal databases, including data indexing techniques. Furthermore, DataReply [6] performed benchmarks of multiple big data technologies for ingesting and querying spatio-temporal datasets. All benchmarked technologies are open-source, namely: Hive, MongoDB, GeoSpark, Elasticsearch, Geomesa and Postgres-XL. The benchmarks utilize three datasets with results including query and ingest performance, along with setup and configuration, infrastructure complexity, community support, monitoring tools availability and complexity of implementing geospatial queries. Moreover, along with spatial, temporal and spatio-temporal queries named as simple queries, string queries and complex queries are also used. From their performance results it is noticeable that Geomesa outperforms all other open-source solutions.

The goal of this study is to expand on DataReply survey to include proprietary solutions, hence by comparing the performance of querying two spatio-temporal databases, namely Geomesa and SpaceTime. Former is an already mentioned open source solution developed by LocationTech and is considered state-of-the-art in the context of spatio-temporal databases, while latter is a proprietary solution developed by Mireo. Comparison is done by performing multiple performance benchmarks to evaluate the impact of indexing methods and data clustering on query response times. Additionally, scalability benchmarks are performed to test databases distribution and resource usage impact.

Results show that SpaceTime outperforms Geomesa in all benchmarks with response times in range of seconds compared to Geomesa's minutes. It exhibits better CPU and memory utilization due to superior caching capabilities, as well as the resilience to slower network speeds. The only case where Geomesa outperforms SpaceTime is when running on hard drive (HDD), specifically with smaller amounts of memory where SpaceTime is unable to utilize its caching capabilities.

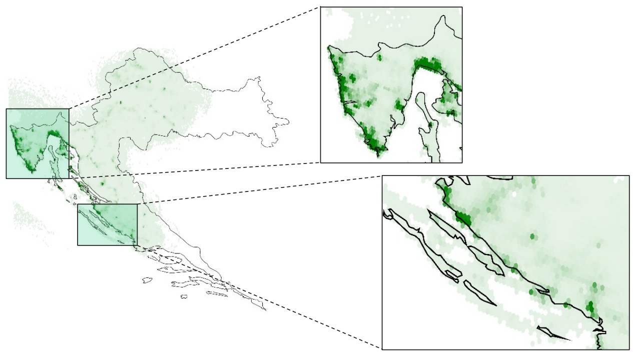


Fig. 1: Dataset spatial distribution heatmap

II. Methodology and Setup

Two spatio-temporal databases benchmarked in this paper include **Geomesa** and **SpaceTime**. Both of them represent solutions for handling spatio-temporal data, where former is an open-source solution developed by LocationTech without any known hard requirements, while latter is a proprietary solution developed by Mireo with hard requirements for solid state disk (SSD) and gigabit network. Their performance is tested with two groups of benchmarks. First group includes performance benchmarks that contain different sets of spatial, temporal and spatio-temporal queries used to test database indexing technique and data clustering, namely:

- **Spatial benchmarks** utilize only spatial queries to test spatial indices and spatial data clustering.
- **Temporal benchmarks** utilize only temporal queries to test temporal indices and temporal data clustering.
- **Spatio-temporal benchmarks** utilize spatio-temporal queries to test both spatial and temporal indices.

Second group includes scalability benchmarks used to test databases distribution and resource usage, namely:

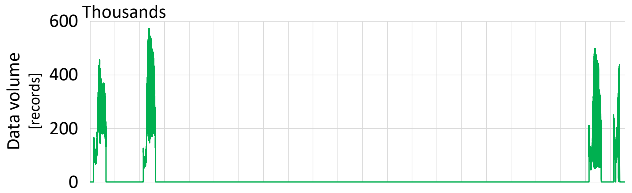
- **Data scalability benchmarks** utilize spatio-temporal queries for a variety of dataset sizes to test how database performance scales with regards to data volumes.
- **Resource scalability benchmarks** utilize spatio-temporal queries while running in various environments to test how databases perform with different CPU, memory, network and disk configurations.

A. Dataset

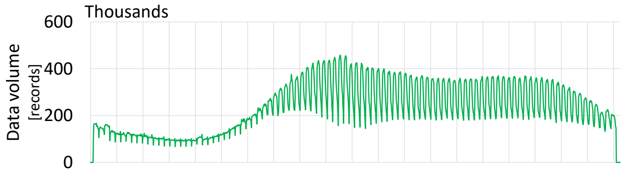
Dataset used in this study contains an anonymized telecom data from arbitrary base stations with a total of 1 162 400 595 records. The data is non-uniformly spatially and temporally distributed across Croatia with the timespan of 43 days as depicted in Figure 1 and Figure 2, respectively. Figure 1 shows that the most records are located over settlements, with the highest data density in city of Rijeka and Istria. Similarly, Figure 2a shows the temporal data distribution with the highest density during four specific days.

B. Configuration

Both databases are deployed on top of two virtual machines running on separate physical workstations, each virtual machine initially being configured with a configuration displayed in Table I.



(a) Whole dataset between 08/06/2018 19:12 and 20/07/2018 19:12



(b) One day between 09/06/2018 01:52 and 10/06/2018 01:52

Fig. 2: Dataset temporal distribution in minutes

The initial configuration is used for most benchmarks, except the resource related benchmarks where resources are intentionally reconfigured to study the performance on different resource configurations. SpaceTime is installed on top of Debian 9.11, while Geomesa is installed as a plugin to Accumulo database with Zookepeer and Hadoop as a coordinator and a backend, respectively, all running inside their own unlimited Docker containers on Ubuntu 20.04. Queries are executed from a separate machine using Scala scripts, where SpaceTime is accessed through its REST API, while Geomesa is accessed through GeoTools Java library. Both databases are indexed by their space and time data features.

TABLE I: Physical workstation initial configuration

Resource type	**Capacity**
CPU	12 cores (out of 16 cores)
RAM	96 GB (out of 120 GB)
DISK	500 GB SSD NVMe (out of 1 TB)
NETWORK	Gigabit

C. Queries

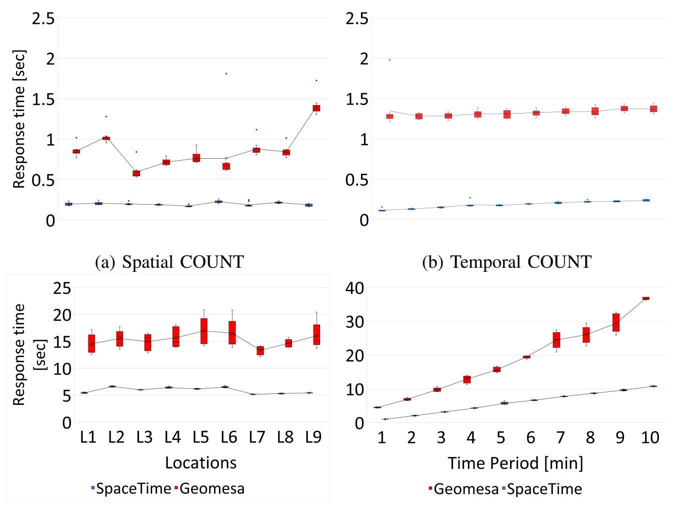
Several types of queries are used for benchmarking the database performances, including:

- **Temporal queries** request a set of data based on a time interval.
- **Spatial queries** request a set of data based on its location defined with geo-coordinates.
- **Spatio-temporal queries** request a set of data based on both time interval and a location.

Depending on a specific benchmark, above mentioned queries are executed as:

- **Count** – a database simply counts a set of data and returns a single integer.
- **Fetch** – a database fetches a set of data and returns the data itself over the network.

Each query is repeated 5 to 10 times in order to obtain statistically meaningful average values, as well as the deviation in performance for a specific query/configuration. Most of the obtained results are shown as distribution of data into quartiels, asserting mean value and outliers.



(a) Spatial COUNT (b) Temporal COUNT

(c) Spatial FETCH (d) Temporal FETCH

Fig. 3: Performance of spatial and temporal benchmarks

III. Performance benchmarks

This section presents the results of all mentioned performance benchmarks from Section II.

A. *Spatial benchmark*

Spatial benchmarks utilize spatial queries that either count or fetch approximately 1 million records for various geo-locations in order to test the performance of a database with regards to the geo-location being queried. A single test includes 9 geo-locations, where each test is repeated 10 and 5 times for count and fetch queries, respectively, which results with 135 queries being executed per database.

Figures 3a and 3c depict that SpaceTime performs better than Geomesa, as well as having more stable response times with lower deviation. However, they both follow a similar performance pattern with regards to the geo-location being queried.

B. *Temporal benchmark*

Temporal benchmarks utilize temporal queries that either count or fetch sequentially increasing datasets by increasing the requested time span. A single test includes 10 queries, where each query increases a requested time span by 1 minute, thus going from 1 to 10 minutes. Again, each test is repeated 10 and 5 times for count and fetch queries, respectively, which results with 150 queries per database.

Figures 3b and 3d depict that SpaceTime again performs better than Geomesa. For fetch queries it exhibits more stable performance, while Geomesa also gives more steeper increase in response time. However, for count queries, while giving better performance, SpaceTime gives steeper increase in response times unlike Geomesa's almost flat performance. More insights into this behaviour is given in Figure 4.

In Figure 4, it can be clearly seen that SpaceTime outperforms Geomesa by in average 8 times faster response time over a time span of 12 days. On one hand, SpaceTime gives an expected behavior by exhibiting longer response times as the requested time span increases, due to the fact that longer

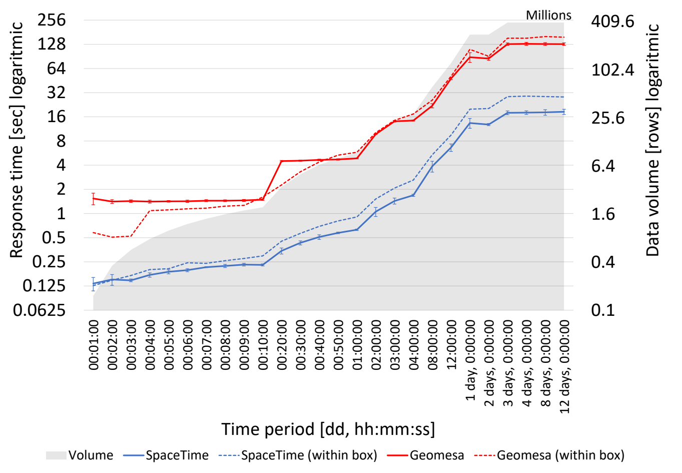


Fig. 4: Performance with regards to a longer time span

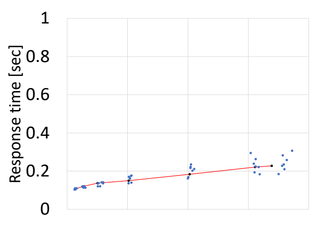
time span equals more data. On the other hand, Geomesa exhibits almost flat performance on some sequential queries, regardless of the larger amount of data that it needs to process with every next query (also seen in Figure 3b and 3d). This can be explained by the way Geomesa indexes its data by the temporal feature, where some temporally related data falls into favorable buckets and thus do not affect the performance in a significant manner, while SpaceTime indexes spatial and temporal features with a single weighted index and thus gives more balanced performance.

Figure 4 also depicts the results (dashed lines) of spatio-temporal queries that request the same data as their temporal counterparts. This was achieved by requesting the same time span, while providing a bounding box size of Croatia in order to cover all the data spatially. The results show that Geomesa improves its performance with such queries for smaller time spans and deteriorates for larger time spans. Similar behavior is exhibited by SpaceTime as well, while altogether giving better performance than Geomesa.

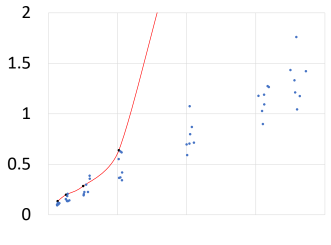
C. *Spatio-temporal benchmark*

Spatio-temporal benchmarks utilize spatio-temporal queries that either count or fetch sequentially increasing datasets. The datasets are increased by increasing the requested time span, while a spatial feature remains the same. The time span is increased 7 times in a manner to provide approximately the same amount of data for each geo-location, rather than the same time span. Consequently, each test contains 56 queries, namely 7 different time spans for 8 different geo-locations. Altogether, each database is benchmarked with 616 queries.

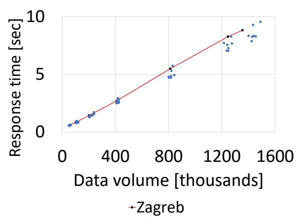
Figures 5a and 5b depict results for count queries where SpaceTime outperforms Geomesa. On one hand, SpaceTime also exhibits more stable performance independent from the specific geo-location, rather its performance is almost entirely dependent on the amount of data being requested. On the other hand, Geomesa shows significant dependence on a geo-location such as Zagreb, which increases the response times over 5 seconds regardless of the same amount of data being requested as with other queries.



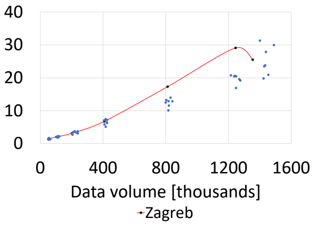
(a) SpaceTime COUNT



(b) Geomesa COUNT



(c) SpaceTime FETCH



(d) Geomesa FETCH

Fig. 5: Performance of spatio-temporal benchmarks

Figures 5c and 5d depict database performance for fetch queries. Obviously, these take more time than count queries as they send requested data over the network. However, the performance impact is similar as for the count queries, where SpaceTime gives more stable and faster response times than Geomesa, which again shows dependency on spatial distribution of the data.

In order to examine this performance dependency on spatial distribution, the results from the spatio-temporal count queries are plotted again in Figure 6, which now includes requested area size as well. It can be seen from the figure that SpaceTime shows no or very small dependency on an area size, while Geomesa gives better response times for larger areas. This is possibly due to different indexing schemes used by both databases. On one hand, SpaceTime splits its indexes with regards to the data, resulting with an evenly balanced indexing buckets. Consequently, each query regardless of its size is capable of utilizing parallel execution over multiple indexes. On the other hand, Geomesa splits its indexes with regards to space rather than data, which results in an unevenly clustered data, resulting with almost serial data retrieval for small areas.

Finally, all count queries are plotted in Figure 7 in order to compare database performances with regards to a type of query, namely spatial, temporal and spatio-temporal. On one hand, the figure shows that SpaceTime has low dependency on a type of query, i.e., its performance follows a similar curve for all types. On the other hand, the figure shows that Geomesa gives almost flat performance for temporal queries, scattered performance for spatio-temporal queries, and the best performance for spatial queries.

Performance dependency on a type of query is again possibly related to the database's different indexing schemes. Namely, SpaceTime utilizes combined weighted index for spatial and temporal features, which results with a single balanced index. Geomesa, on the other hand utilizes two separate indexes for spatial and temporal features, which results with the completely different performance patterns for all three types of queries.

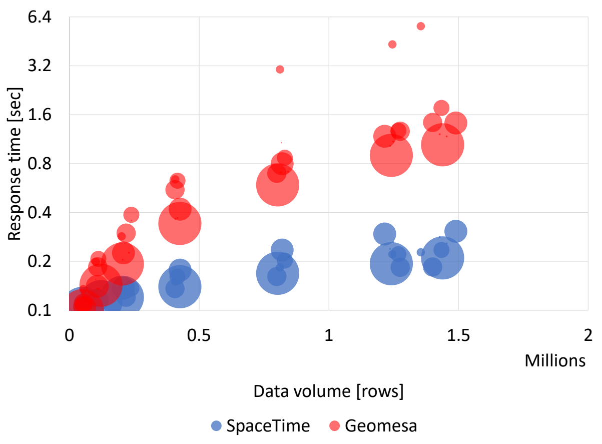


Fig. 6: Performance with regards to data volume and area size

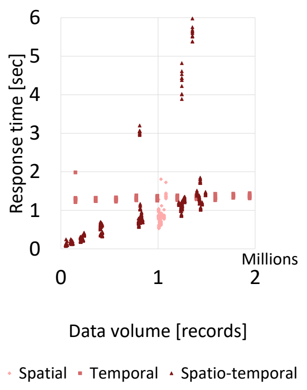
IV. Scalability benchmarks

Results from Section III show that the database performances depend mostly on the requested data volumes. Consequently, additional benchmarks are executed over larger datasets using spatio-temporal count queries. Besides scalability tests regarding data volumes, database scalability also depends on the amount of given resources, which is further explored with different resource configurations including CPU, memory, network and disk.

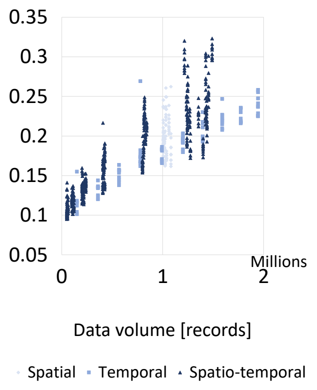
A. Data scalability benchmark

Area being queried covers a city of Rijeka as it contains most of the data from the complete dataset. Moreover, the queries span between 5 minutes and 2 hours with a step of 5 minutes, as well as for 2, 6, 42 and 44 days. All tests are repeated 10 times in order to obtain average values, which equals to 280 queries per database.

Figure 9 depicts the performance of both databases with regards to the requested data volumes, going from several hundreds of thousands of records to over 250 million records. The figure clearly shows that SpaceTime outperforms Geomesa on larger datasets, going from few times faster execution to over 17 times faster execution. The figure additional gives an insight that for even larger datasets this performance ratio would increase even more.



(a) Geomesa all COUNT queries



(b) SpaceTime all COUNT queries

Fig. 7: Performance of spatial, temporal or spatio-temporal benchmarks (*mind the different Y axis scale*)

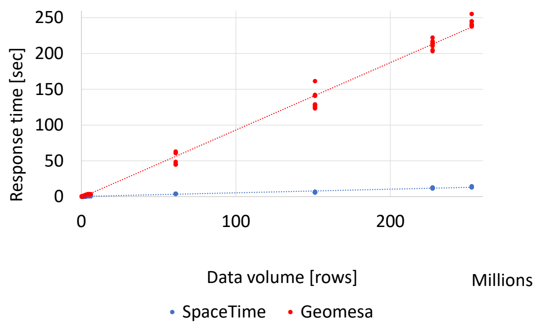


Fig. 9: Database scalability with regards to data volume

B. CPU related benchmark

For CPU related benchmarks 4 different configurations are used, namely 12 cores as a default setup, along with 8, 4 and 2 cores, which equals to 40 queries per database. All other resources are configured by default settings (Section II-B), while a subset of queries from the previous section (city of Rijeka with a time span between 2, 6, 42 and 44 days) is used.

Figure 8e shows that SpaceTime scales as expected with regards to number of CPU cores, i.e., response times increase as the number of cores goes down. However, Figure 8a shows that Geomesa does not follow the same expected behaviour, as its performance significantly degrades with 2 cores, while with more cores it does not improve as much. Moreover, it even exhibits better performance with 8 cores than with 12 cores. This behaviour might be related to the fact that a physical workstation has 16 virtual cores (2 x 8 physical cores), where perhaps those 4 extra cores exhibit some context-switching overhead rather than improve performance.

C. Memory related benchmark

For memory related benchmarks 6 different configurations are used, namely 96 GB of RAM as a default setup, along with 16, 8, 4, 2 and 1 GB of RAM, which equals to 60 queries per database. All other resources are configured by default settings (Section II-B).

Figure 8f shows that SpaceTime slightly losses performance once the allocated memory goes below 8 and 4 GB of RAM but continues to perform very well even with 1 GB of RAM. Geomesa, on the other hand, as shown on Figure 8b starts losing performance with 2 GB of RAM and completely degrades with 1 GB of RAM by giving almost 6 times longer response times. This is due to the much higher memory requirements by Geomesa compared to SpaceTime. During tests, SpaceTime was hardly using 200 MB of RAM when running idle along with the operating system, while Geomesa would consume over 1 GB of RAM while being idle.

D. Network related benchmark

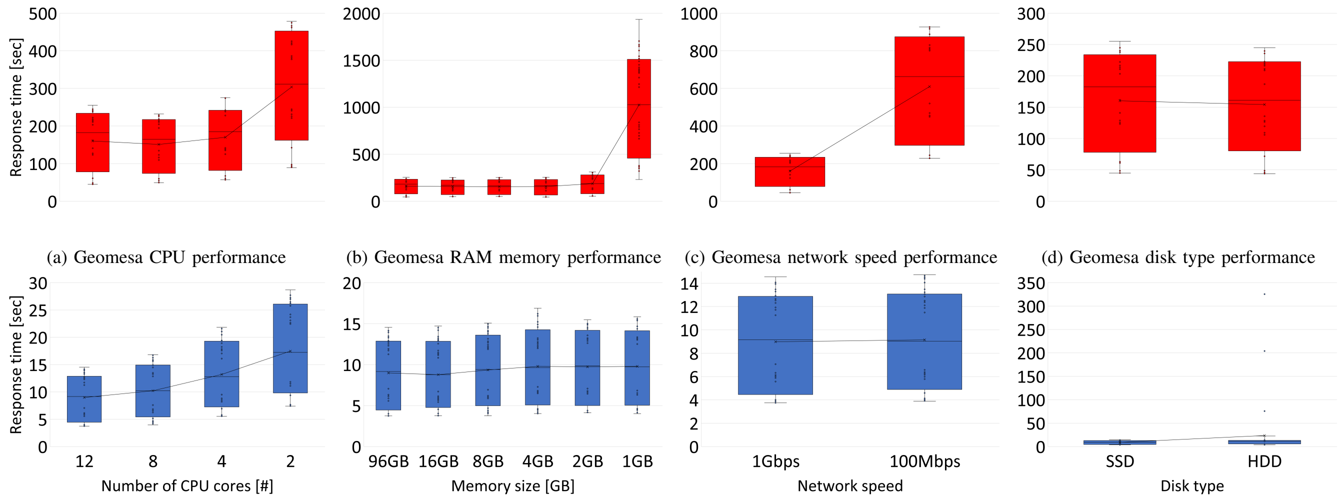
For network related benchmarks 2 different configurations are used, namely 1Gbps as a default setup, along with 100Mbps, which equals to 20 queries per database. All other resources are configured by default settings (Section II-B).

Figure 8g shows that SpaceTime performance does not depend much on the network speed. However, it must be noted that the executed queries are read queries, while for write queries it is expected that the performance would vary due to the fact that SpaceTime utilizes storage replication over network. Geomesa exhibits significant performance degradation (Figure 8c) even for read queries giving over 3 times slower response times for 100Mbps network. This might be due to the fact that Geomesa is essentially a plugin for Accumulo database that uses Hadoop as a backend storage. Consequently, it possibly creates a significant network overhead by running as a distributed database on top of so many layers.

E. Disk scalability benchmark

For disk related benchmarks 2 different configurations are used, namely SSD disk as a default setup, along with HDD, which equals to 20 queries per database. All other resources are configured by default settings (Section II-B).

One of the hard requirements of SpaceTime is the use of SSD. However, Figure 8h shows that SpaceTime seemingly performs the same on both SSD as well as on HDD, with only few outliers for HDD. Figure 8d shows that Geomesa gives slightly better performance while running on HDD. More insights on these unexpected behaviors is depicted below.



(a) Geomesa CPU performance (b) Geomesa RAM memory performance (c) Geomesa network speed performance (d) Geomesa disk type performance

(e) SpaceTime CPU performance (f) SpaceTime RAM memory performance (g) SpaceTime network speed performance (h) SpaceTime disk type performance

Fig. 8: Database performance with regards to scalability benchmarks

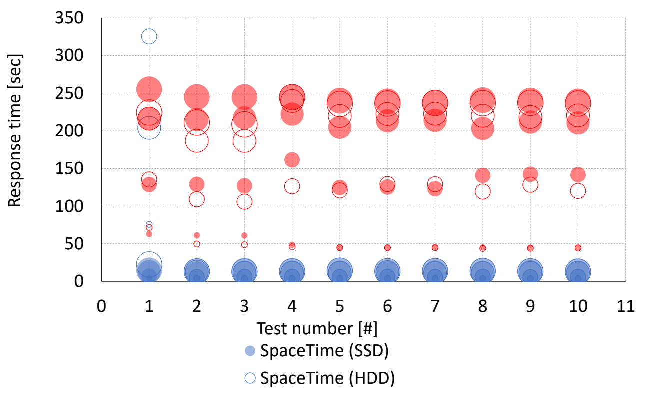


Fig. 10: Scalability performance by individual sequential tests

Figure 10 shows that SpaceTime, when run on SSD, has a very stable performance for all 10 tests. However, when run on HDD, the first test gives extremely poor performance, in some cases even worse than Geomesa on both SSD and HDD. All other tests from 2 to 10 perform the same as those on SSD. This is possible due to an excellent caching capabilities of SpaceTime. However, such performance on HDD cannot be expected in a production as these 10 queries are all the same and executed sequentially, which is not a realistic situation. Therefore, the first test gives a somewhat more realistic performance, which poor results can be explained by SpaceTime's heavy reliance on the disk speed.

Geomesa shows less stable performance by individual tests. While running on HDD, it even performs better than on SSD in three first tests, and then degrades, while on SSD it slightly improves over sequential tests. These results show that Geomesa does not handle caching very well, again possibly due to a layered architecture where each layer performs a caching operation of its own. Figure 11 shows even deeper insights into database performances when running on HDD, where additional tests are executed with only 2GB of RAM to throttle the caching capabilities.

SpaceTime, when run on SSD, shows excellent and stable performance which only increases as the data volume increases. When run on HDD, the first query with around 60 million records gives significantly worse results than SSD, even slightly worse than Geomesa. The second query with around 150 million records gives the worse execution, even 2 times worse than Geomesa and 46 times worse than SpaceTime on SSD. However, the third query, although having around 225 million of records, performs better than the second one that requests only 150 million records. This is due to the fact that the third query queries the same area of the city of Rijeka with the expanded time span, i.e., it contains the same data. Consequently, this is where SpaceTime caching capabilities kick in and provide better performance. This is also evident with the fourth query which gives almost SSD-like performance. However, in a test with only 2 GB of RAM that throttles caching capabilities, SpaceTime gives much worse results for all three queries, while only reaching SSD-like performance with the fourth one.

Geomesa, when run on SSD, gives almost the same performance as when run on HDD, where even HDD performs

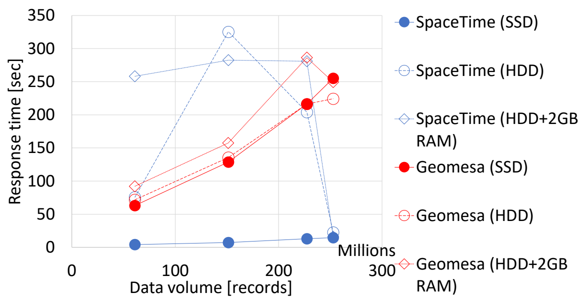


Fig. 11: Single test scalability performance

better for the fourth query. When throttled with only 2 GB of RAM it performs worse as it cannot cache properly. However, these results are very similar to the ones when Geomesa is run on SSD with 2 GB of RAM as seen in Figure 8b. Therefore, Geomesa does not utilize benefits of SSD considerably.

V. Conclusion

In this paper, a set of performance and scalability benchmarks were performed in order to compare open source solution for managing spatio-temporal data, namely Geomesa, and a proprietary solution SpaceTime developed by Mireo.

The results show that SpaceTime outperforms Geomesa in all performed benchmarks by giving response times in a range of seconds, unlike Geomesa that requires minutes. SpaceTime excels when executing different types of queries (spatial, temporal and spatio-temporal). When testing scalability with regards to the data volume it gives over 17 times faster response time for 250 million records and it is expected to perform even faster as the data volume increases. It better utilizes more CPU cores and it consumes significantly less memory. Its read queries are not as affected by the slower network as Geomesa's, and finally it utilizes caching capabilities much better then Geomesa when run on HDD instead of SSD. The only downside to SpaceTime is when running on HDD, where it performs worse than Geomesa. Therefore, the hard requirement for using SSD set by SpaceTime is confirmed.

VI. Acknowledgment

We would like to thank Mireo d.d., Zagreb, Croatia for providing access to SpaceTime database for benchmark testing.

References

[1] M. M. Alam, L. Torgo, and A. Bifet, "A survey on spatio-temporal data analytics systems," 2021.

[2] A. Oussous, F.-Z. Benjelloun, A. Ait Lahcen, and S. Belfkih, "Big data technologies: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018.

[3] M. Sudmanns, D. Tiede, S. Lang, H. Bergstedt, G. Trost, H. Augustin, A. Baraldi, and T. Blaschke, "Big earth data: disruptive changes in earth observation data management and analysis?," *International Journal of Digital Earth*, 03 2019.

[4] R. Nandal, "Spatio-temporal database and its models: A review," *IOSR Journal of Computer Engineering*, vol. 11, pp. 91–100, 2013.

[5] T. Abraham and J. F. Roddic, "Survey of spatio-temporal databases," 1998.

[6] "Benchmarking of big data technologies for ingesting and querying geospatial datasets." https://www.reply.com/en/topics/big-data-and-analytics/Shared Documents/DSTL-Report-Data-Reply-2017.pdf. [Online; accessed 02-July-2021].