Comparing Costs for Cloud-based Data Archives

LIBBY HEMPHILL, University of Michiagn, USA
JUNJIE XING, University of Michiagn, USA
LIZHOU FAN, University of Michiagn, USA

Research data management is an expensive enterprise. Computing infrastructure for storing, retrieving, and preserving data is one area of expenses, and computing infrastructure costs grow as the size and number of datasets and demands for their retrieval grow. This paper compares the costs and performance of two database infrastructures, PostgreSQL and Elasticsearch, for digital data archives. We used benchmarking experiments and data from social media to estimate the costs of loading, indexing, and querying data from these two databases. The results show that traditional relational open-source databases can be effective for large social science data and run on relatively low-cost computing infrastructure, where PostgreSQL queries can be faster and less expensive than Elasticsearch. PostgreSQL required higher up front costs and time, and adding computing resources did not improve Elasticsearch's query performance. These findings are useful for digital archives evaluating back-end storage systems.

ACM Reference Format:
Libby Hemphill, Junjie Xing, and Lizhou Fan. 2023. Comparing Costs for Cloud-based Data Archives. 1, 1 (May 2023), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Sharing and archiving data is important to enable researchers to verify, replicate, and extend one another's work. Caring for digital research data is a demanding and expensive enterprise [5, 12, 20, 24]. The specialized hardware required to provide long-term, secure, distributed access to research data (e.g., cloud servers, private networks) carries both monetary and environmental costs. Advances in cloud computing offer opportunities for flexible, scalable data storage and access systems. This paper examines whether these advances have made digital archiving more affordable or efficient by testing queries in multiple database structures and on various servers hosting terabytes of data from social media.

The [archive name removed for blind review] is developing infrastructure to archive and distribute data from social media platforms. The archive offers an opportunity to calculate the costs associated with a cloud-based data archive. In this paper, we address only questions related to loading data for storage, indexing data for retrieval, and querying data for access. We do not include costs related to staffing, data collection, user training, or long-term preservation. We also do not address collection development and access questions but acknowledge that social media users' preferences should influence archiving decisions [14, 17].

Our primary contribution is a comparison of the costs of two different database infrastructures, PostgreSQL and Elasticsearch, for digital data archives. We present findings from a set of benchmarking experiments that let us estimate

Authors' addresses: Libby Hemphill, University of Michiagn, Ann Arbor, MI, USA, 48109, libbyh@umich.edu; Junjie Xing, University of Michiagn, Ann Arbor, MI, USA, 48109, jjxing@umich.edu; Lizhou Fan, University of Michiagn, Ann Arbor, MI, USA, 48109, lizhouf@umich.edu.

Manuscript submitted to ACM

the costs of loading, indexing, and querying text and date data from these two databases. We found that PostgreSQL could return queries quickly, but it was more expensive overall because of the size of its data store and the separate index and ingest steps required. Elasticsearch was cost-effective and could run efficiently on moderate computing resources.

2 BACKGROUND

2.1 Digital Archiving and Big Data

The integration of digital archiving and big data has attracted increasing research attention: big data technologies can be employed to manage and preserve large digital collections, and digital archiving techniques can be used to ensure the long-term big data management and stewardship. Using big data technologies in digital archives enhances machine-actionability of archives and accelerates the shift to ensuring the findability, accessibility, interoperability, and reuse (FAIRness) of high volume digital assets [23]. A variety of recent research focuses on implementing and developing scalable and automated systems for preserving and managing large digital collections.

Some research focus on algorithms and infrastructures for storing, processing, and analyzing large digital collections. Some applications utilize distributed computing systems, such as Hadoop and Spark, to process and analyze complex types of big data in parallel [2, 26]. The use of distributed storage systems is also embedded in digital preservation workflows to ensure that digital content is properly preserved and managed over time [8].

Some other research uses machine learning and artificial intelligence techniques to extract insights and knowledge from big data in archives, which depends on the algorithm and infrastructure advancements [7, 15, 19, 25]. Machine learning and artificial intelligence are helpful in supporting decision-making in digital preservation, including the use of big data analytics to assess and mitigate the risks associated with preservation strategies and needs, for example, organizing and querying archives [4]. Machine learning techniques can also be useful for identifying the elements to preserve, depending on the relevance to archival goals in a digital collection [6].

2.2 Database Options for Digital Archives

The database system is the backend core of a digital archive. We mainly consider the two types of database system: (1) relational database, (2) NoSQL database. Relational database systems (RDS) are the most popular and most mature database systems in production. Schema designs influence how quickly and efficiently RDS can store and retrieve data. We include an RDS in our experiment because they are so popular and configurable. NoSQL database systems are gaining popularity because of their flexibility in data modeling. Unlike RDSs, which model data in table relations, NoSQL stores data in other formats, e.g. JSON, XML, key-values pairs, etc.. We include a NoSQL option in our experiments because they support JSON, the most popular format for social media data application programming interfaces (APIs).

2.3 Cloud Computing, Archival Practice, and the Environment

Following recent research [16, 21] on the energy considerations for machine learning tasks, we aim to highlight the environmental impacts of research data management. Individual researchers storing, indexing, and querying multiple copies of the same (or substantially similar) data multiplies both the financial and environmental costs of working with digital data. For instance, when multiple researchers query a platform's API for data from candidates for office, they create multiple copies of the same set of records, usually housed on individual researcher's computers. Carbon

footprints are receiving increased attention from the digital archiving community, especially in audio/visual archives where storage demands dwarf even our experiments [3].

Shared data infrastructure that uses a common data store reduces the computing demands for individuals, but it doesn't eliminate the costs. Hosting data on shared cloud services risks muddling the chain of custody, presents challenges for data security, and creates security and scalability trade-offs [9].

3 METHODS

3.1 Test Data

We used simplified Twitter data in all of our experiments. We stored only the author, tweet text, date, and hashtags. We used one year (2021) of the Twitter Decahose ("Decahose"), containing roughly 14.2 billion tweets. To store the data in PostgreSQL required roughly 7.1TB ($568/month to store in Amazon EBS); for Elasticsearch, Mapping 1 took 6.7TB ($536), and mapping 2 required 3.7TB ($296) of storage. Details on mappings are available below and in the Appendix.

3.2 System Setup

For software, we choose the following two database systems:

- PostgreSQL (v14.5)
- Elasticsearch (v8.4.2)

Database system configuration is a key factor to the system performance. Since both PostgreSQL and Elasticsearch have hundreds of settings, it is impossible test how every different configuration will effect the performance of the digital data archive. Therefore, to simplify the experiment setup, we decided to use the recommended configuration by PGTune[1] for PostgreSQL, and to use the default configuration for Elasticsearch. In order to test how hardware influences system performance, we also limited the number of CPUs, the size of the memory, .etc, to approximate the system performance of various EC2 instance types. For example, for c5.large, we set the configuration of PostgreSQL as recommended by PGTune for 2 CPUs, 4GB memory.

3.3 PostgreSQL Schema

We used text, timestamp, and tsvector data types to store data in PostgreSQL. We used the same indexing approach in both tests except that in Schema 2, hashtags are case-insensitive. Details are available in the Appendix in Table 4.

3.4 Elasticsearch Mapping

For Elasticsearch, we use the same field name and pre-processing methods (see the Appendix Table 3). We also designed two different mapping strategies in our experiment. For the first one, we utilize Elasticsearch's "dynamic mapping"[2] function and let the system choose the data type automatically.

In Mapping 1, Elasticsearch recognized 'created_at' as a text field, and we had to manually change the mapping to accommodate date queries. Mapping 1 also recognized 'hashtags' as text and therefore treated them as case-insensitive. In Mapping 2, we mapped 'hashtags' as keywords rather than text, and that rendered them case-sensitive.

[1] https://pgtune.leopard.in.ua
[2] https://www.elastic.co/guide/en/Elasticsearch/reference/current/dynamic-field-mapping.html

Instance type	vCPU	Memory	CO_2 per hour	$ per hour
c5.large	2	4	7.0	$0.085
c5.xlarge	4	8	14.1	$0.17
c5.2xlarge	8	16	28.0	$0.34
c5.4xlarge	16	32	56.1	$0.68
c5.9xlarge	36	72	126.2	$1.53
Avg. American			1872	

Table 1. Comparing EC2 instances types used in experiments. Prices are for on demand instances. Memory is measured in GiB. CO_2 estimates are from Teads Engineering and are measured in gCO_2eq. [3]

3.5 Benchmark Queries

We identified common query types from existing research about how researchers use Twitter data [13]. The following types of queries were most common: hashtags, tweet authors, and keywords within tweets. Based on those queries, we defined six types of queries that use hashtags, the tweet text, tweet authors, and date information (see Table 2). To make the benchmark queries reflect the information needs of social scientists, we selected several influential individuals, topics and events in 2021:

(1) **Individuals:** "Elon Musk", "Joe Biden"
(2) **News Press:** "CNN", "New York Times"
(3) **Social Movement:** "Stop Asian Hate", "Free Britney"
(4) **Hot Topic:** "Tokyo Olympics"
(5) **Natural Event:** "hurricane ida"

3.6 Estimating Costs

3.6.1 Computing Resources. We used five different Elastic Cloud Computing (EC2) instances from Amazon Web Services (AWS) for our benchmark experiments. We chose EC2 and AWS because they are widely used in university and non-profit settings and have robust user support communities. See Table 1 for a summary of the instances and their associated costs. We did not apply any of the discounts for which the authors are eligible; these costs are based on AWS's commercial rates as of January 2023.

3.6.2 Carbon Emissions. We use freely-available tools[4] to estimate the carbon emissions generated by our benchmark queries. Our carbon estimates are meant to illustrate the relative costs of various database approaches, not to be accurate calculations of the exact carbon emissions for any of our benchmarks. AWS recently introduced a tool for estimating an account's carbon impact [5], but it only allows account holders to estimate their own carbon emissions based on their AWS workload and not to forecast or compare various architectures. See Table 1 for a summary of the EC2 instances and their associated emissions estimates. For reference, Strubell et al. [21] estimates that the average American consumes 16,400 kilograms of CO_2 per year, and the average car consumes 57,153 kilograms of CO_2 in its lifetime.[6]

[3]https://engineering.teads.com/sustainability/carbon-footprint-estimator-for-aws-instances
[4]https://engineering.teads.com/sustainability/carbon-footprint-estimator-for-aws-instances
[5]https://aws.amazon.com/blogs/aws/new-customer-carbon-footprint-tool/
[6]per-capita consumption: https://bit.ly/2Hw0xWc; car lifetime: https://bit.ly/2Qbr0w1[21, p.3645]

Fig. 1. Comparing costs of different query types using either PostgreSQL or Elasticsearch databases. All queries in this chart were run on c5.2xlarge EC2 instances.

4 RESULTS

4.1 Costs of indexing and loading data

We used a c5.9xlarge EC2 instance for loading and indexing data. Both databases took roughly 50 hours to load our test data. Elasticsearch indexes as it loads, but PostgreSQL required an additional 144 hours of time to index the data in our schema. In total, Elasticsearch indexing and loading cost roughly 6.31 $kgCO_2eq$ and $76.50. PostgreSQL required an additional 18.2 $kgCO_2eq$ and $220.32 to index the data.

4.2 Costs of querying

We illustrate the costs of various queries on different databases in Figure 1. More detailed cost estimates are provided in Tables 7 in the Appendix.

We illustrate the impact of query result set size and EC2 instance type on Elasticsearch's performance in Figure 2. Expanding the computing resources available improves Elasticsearch's performance only to a point, after which increased computing actually decreases performance.

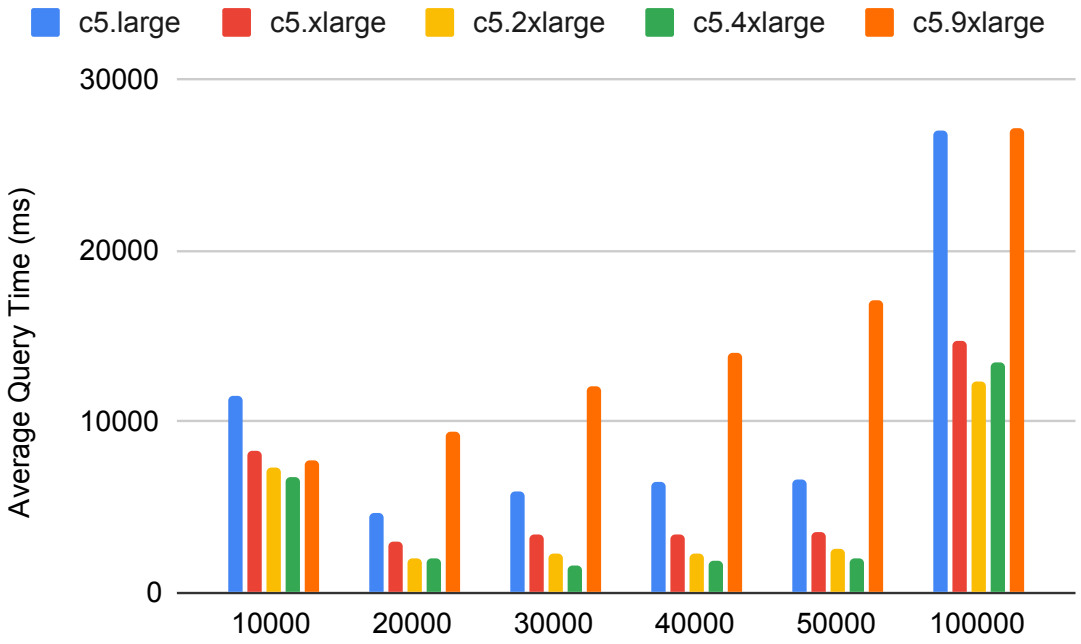

Fig. 2. Query times for Elasticsearch (Mapping 2) for full text search

4.3 Overall costs

Of course, the computing infrastructure runs 24/7 and not just when users submit queries. Our query time estimates are useful for comparing the databases' performance trade offs but not for determining the actual costs of running the archives' instances. Using a c5.2xlarge instance, the costs to set up and run our archives for one year are as follows:

- PostgreSQL: $10,092
- Elasticsearch Mapping 1: $6607
- Elasticsearch Mapping 2: $3552

The start up costs for PostgreSQL are higher, but it's case insensitive, flexible queries are faster and likely better for users. Unfortunately costs to load data recur anytime we add new data to the database. This is true for both PostgreSQL and Elasticsearch, but PostgreSQL is more expensive at load time.

5 DISCUSSION

We conducted a series of experiments with benchmark queries on Twitter data stored in PostgreSQL and Elasticsearch and serviced by varying levels of computing resources. We found that PostgreSQL's up front and data ingest costs are higher but that its queries are faster and more efficient in the long run. Elasticsearch provides fuzzy search and improved recall but requires higher ongoing computing costs to meet performance goals. Our findings are in line with earlier work on the cloud computing costs associated with digital archiving [22, 24].

Twitter data is born digital and undergoes many transformations and moves before arriving at an API endpoint; like the Facebook data Glassman [10] discusses, there are many representations of the same data. Each representation of a tweet is essentially a surrogate carrying the original information. As Shein and Lapworth [20] pointed out, the ongoing maintenance of digital surrogates can be expensive, sometimes even more expensive than taking custody of a physical object. Our inability to develop infrastructure for ethically and accessibly archiving large data, like social media records, undermines trust in archives [10]. Our paper helps articulate the costs and trade-offs involved in building such infrastructure provides important benchmarks for discussions about data structures and types of query support.

5.1 Comparing PostgreSQL and Elasticsearch

5.1.1 Costs. The costs, both carbon and dollars, for PostgreSQL and Elasticsearch databases are similar in the first year. PostgreSQL was 53% more expensive given its higher index and ingest costs. PostgreSQL is more expensive because the default compression algorithm is not as effective as Elasticsearch's, which is reflected by it's lower storage demands after loading and indexing. However, PostgreSQL queries are faster which means it can support more uses on the same infrastructure, and the overall carbon footprint is lower: the amount of carbon produced depends on the load on the EC2 instance while the dollar costs are fixed by time.

5.1.2 Features. Both databases supported the common user query types we identified from existing research (i.e., full-text, author). We were able to create indices and mapping that allowed for case insensitive search, but those increase query time in both databases. Future work should investigate query fuzziness specifically. For instance, for some research questions, researchers may not know the exact keyword(s) to search and would benefit from less precise matching than we tested. We expand on potential avenues for fuzzy search in the Appendix. In general, Elasticsearch provides better support for text search than PostgreSQL, but some features such as "edit distance" can be added via PostgreSQL modules. Archives will need to use data from user queries and other interactions with users to identify the fuzziness that would be useful and design databases and queries to support it.

5.2 Other Options for Large-scale Data Storage

5.2.1 Custodial Archiving. First we address other cloud-based services for hosting data for which the archive accepts custody. In these scenarios, the archive is responsible for storing and providing access to the files. In the next section, we address non-custodial approaches in which the archive does not directly manage the data in its collections.

Amazon Athena: Athena[7] is a serverless interactive query service. It provides query capacity on top of data lake storage, e.g. Amazon Simple Storage Service[8] (Amazon S3) using ANSI SQL.

However, Amazon Athena is not appropriate for hosting a large data archive for two main reasons: costs and query types. Athena's total cost includes (1) the cost of data storage, e.g. Amazon S3, (2) and Athena SQL query: $5.00 per TB of data scanned. In our experiment, the entire data size is around 6TB, thus each query will cost around $30.00. We can certainly reduce the cost as well as accelerate the query by partitioning the data and build partition indexes using AWS Glue Data Catalog. However, the archive would still need to spend time on data infrastructure configuration (i.e. data partitioning and partition index building). This produces the second drawback: efficiently partitioned Athena cannot efficiently support the different types of queries our users require.

[7]https://aws.amazon.com/athena/
[8]https://aws.amazon.com/s3/

Hosted RDS: Hosted, managed RDS services such as Amazon RDS or Cloud SQL (Google Cloud's offering) are too expensive for our purposes (e.g., storage alone is $1207/month for our data in Cloud SQL[9]). Staffing and computing create trade-offs for managed vs self-hosted RDSs. RDSs are mature enough that the expertise to manage RDS services is affordable for most archives that employ technical staff. Cloud computing, especially at discounted rates for universities, archives, and other non-profit organizations, is now inexpensive enough to be more cost-effective, even when system administration staff is required, than managed RDS options.

5.2.2 Non-custodial Archiving. Thus far we have assumed that the data archive must take physical custody of the data. For many archives, taking custody of large-scale data is not practical (i.e., it's too expensive or difficult to secure) or necessary. Instead, the archive could serve as a trusted intermediary between those with custody of the data (in our case, a social media platform company) and the archive's users (i.e., academic researchers). This postcustodial approach [11, 20] comes with other drawbacks, however. First, whether data custodians can be trusted to provide fair, perpetual access to the data remains to be seen. Second, the mechanism of access, such as an API, creates constraints and risks for data users. As Padilla [18] points out, social media platforms can monetize and monitor researchers' API calls, potentially threatening academic freedom. Acker and Kreisberg [1] also call attention to the competing interests between API developers and researchers, acknowledging that researchers' data needs are often not met by commercial APIs. A non-custodial, or postcustodial, approach to data archives would require archives and data custodians to work together to ensure secure, fair, accessible use for researchers.

6 CONCLUSION

In this paper, we compared the expenses and efficiency of PostgreSQL and Elasticsearch as database systems for storing digital archives. Our findings indicate that

(1) Traditional relational open source database structures can work for large social science data and run on relatively inexpensive computing infrastructure but have higher start up costs;
(2) PostgreSQL queries can be faster and less expensive than Elasticsearch; and
(3) Additional computing resources do not improve Elasticsearch query time performance.

Our findings will be useful to any digital library or archive evaluating back-end storage systems and those estimating and/or articulating computing costs.

ACKNOWLEDGMENTS

REFERENCES

[1] Amelia Acker and Adam Kreisberg. 2020. Social media data archives in an API-driven world. *Archival Science* 20, 2 (June 2020), 105–123.
[2] Muhammad Babar, Fahim Arif, Mian Ahmad Jan, Zhiyuan Tan, and Fazlullah Khan. 2019. Urban data management system: Towards Big Data analytics for Internet of Things based smart urban environment using customized Hadoop. *Future Generation Computer Systems* 96 (2019), 398–409.
[3] Itza A Carbajal and Michelle Caswell. 2021. Critical Digital Archives: A Review from Archival Studies. *Am. Hist. Rev.* 126, 3 (Nov. 2021), 1102–1120.
[4] Jan de Mooij, Can Kurtan, Jurian Baas, and Mehdi Dastani. 2022. A Computational Framework for Organizing and Querying Cultural Heritage Archives. *Journal on Computing and Cultural Heritage (JOCCH)* 15, 3 (2022), 1–25.
[5] Ryan Deschamps, Samantha Fritz, Jimmy Lin, Ian Milligan, and Nick Ruest. 2020. The cost of a WARC: analyzing web archives in the cloud. In *Proceedings of the 18th Joint Conference on Digital Libraries* (Champaign, Illinois) *(JCDL '19)*. IEEE Press, 261–264.
[6] Lizhou Fan, Zhanyuan Yin, Huizi Yu, and Anne J Gilliland. 2022. Using machine learning to enhance archival processing of social media archives. *Journal on Computing and Cultural Heritage (JOCCH)* 15, 3 (2022), 1–23.
[7] Marissa Friedman, Cameron Ford, Mary Elings PI, Vijay Singh, and Tracey Tan. 2021. Using AI/Machine Learning to Extract Data from Japanese American Confinement Records. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2210–2219.

[9] https://cloud.google.com/sql/pricing#mysql-pg-pricing

[8] Mariagrazia Fugini and Jacopo Finocchi. 2022. Data and Process Quality Evaluation in a Textual Big Data Archiving System. *ACM Journal on Computing and Cultural Heritage (JOCCH)* 15, 1 (2022), 1–19.
[9] Anne J Gilliland. 2017. *Conceptualizing 21st-Century Archives.* American Library Association.
[10] Dominique Glassman. 2020. Facebook is creating records — but who is managing them? *Arch. Manuscripts* 48, 1 (Jan. 2020), 45–58.
[11] F Gerald Ham. 1981. Archival Strategies for the Post-Custodial Era. *Am. Arch.* 44, 3 (1981), 207–216.
[12] Margaret Hedstrom. 1997. Digital Preservation: A Time Bomb for Digital Libraries. *Comput. Hum.* 31, 3 (May 1997), 189–202.
[13] Libby Hemphill, Margaret L Hedstrom, and Susan Hautaniemi Leonard. 2021. Saving social media data: Understanding data management practices among social media researchers and their implications for archives. *J. Assoc. Inf. Sci. Technol.* 72, 1 (Jan. 2021), 97–109.
[14] Libby Hemphill, Angela Schöpke-Gonzalez, and Anmol Panda. 2022. Comparative sensitivity of social media data and their acceptable use in research. *Sci Data* 9, 1 (Oct. 2022), 643.
[15] Hermann Kroll, Jan Pirklbauer, Florian Plötzky, and Wolf-Tilo Balke. 2022. A Library Perspective on Nearly-Unsupervised Information Extraction Workflows in Digital Libraries. In *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*. 1–11.
[16] Da Li, Xinbo Chen, Michela Becchi, and Ziliang Zong. 2016. Evaluating the Energy Efficiency of Deep Convolutional Neural Networks on CPUs and GPUs. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*. ieeexplore.ieee.org, 477–484.
[17] Catherine C Marshall and Frank M Shipman. 2012. On the institutional archiving of social media. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries - JCDL '12* (Washington, DC, USA). ACM Press, New York, New York, USA, 1.
[18] Thomas G Padilla. 2018. Collections as Data: Implications for Enclosure. *College and Research Libraries News* 79, 6 (2018), 296.
[19] Nathaniel Payne. 2019. An intelligent class: The development of a novel context capturing framework supporting the functional auto-classification of records. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 3136–3145.
[20] Cyndi Shein and Emily Lapworth. 2016. Say Yes to Digital Surrogates: Strengthening the Archival Record in the Postcustodial Era. *Journal of Western Archives* 7, 1 (2016), 9.
[21] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3645–3650.
[22] Xinyue Wang and Zhiwu Xie. 2020. Web archive analysis using hive and SparkSQL. In *Proceedings of the 18th Joint Conference on Digital Libraries* (Champaign, Illinois) *(JCDL '19)*. IEEE Press, 424–425.
[23] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3, 1 (2016), 1–9.
[24] Zhiwu Xie, Yinlin Chen, Julie Speer, and Tyler Walters. 2016. Evaluating Cost of Cloud Execution in a Data Repository. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries* (Newark, New Jersey, USA) *(JCDL '16)*. Association for Computing Machinery, New York, NY, USA, 247–248.
[25] Zhanyuan Yin, Lizhou Fan, Huizi Yu, and Anne J Gilliland. 2020. Using a three-step social media similarity (TSMS) mapping method to analyze controversial speech relating to COVID-19 in Twitter collections. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 1949–1953.
[26] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. 2019. Spatial data management in apache spark: the geospark perspective and beyond. *GeoInformatica* 23 (2019), 37–78.

A APPENDIX

Here we provide additional details about the queries we tested, our pre-processing steps, mapping and schema for the databases, the experiment results, and parameters for fuzzy search.

Type	Description	PostgreSQL Query Example	Elasticsearch Query Example
i	Match 1 Hashtag	SELECT * FROM post WHERE hashtags @> (ARRAY['tokyo2020']);	'bool': 'filter': ['term': 'hashtags': 'tokyo2020']
ii	Match Multiple Hashtags with OR relationship	SELECT * FROM post WHERE hashtags && (ARRAY['tokyo2020', 'tokyoolympics']);	'bool': 'should': ['term': 'hashtags': 'tokyo2020', 'term': 'hashtags': 'tokyoolympics']
iii	Match Multiple Hashtags with AND relationship	SELECT * FROM post WHERE hashtags @> (ARRAY['tokyo2020', 'tokyoolympics']);	'bool': 'filter': ['term': 'hashtags': 'tokyo2020', 'term': 'hashtags': 'tokyoolympics']
iv	Match Author Name	SELECT * FROM post WHERE author_name = 'Tokyo2020';	'bool': 'filter': ['term': 'author_name': 'Tokyo2020']
v	Match Author Name and Text	SELECT * FROM post WHERE author_name = 'Tokyo2020' and text_searchable @@ to_tsquery('swimming');	'bool': 'filter': 'term': 'author_name': 'Tokyo2020', 'must': 'match': 'text': 'swimming'
vi	Match Text	SELECT * FROM post WHERE text_searchable @@ to_tsquery('Olympic Tokyo');	'match': 'text': 'query': 'Olympic Tokyo', 'operator': 'and'

Table 2. Types of Queries in the Benchmark Tests. Includes examples of the queries in both PostgreSQL and Elasticsearch syntax.

		Preprocess Operation	
Column Name	Decahose Field	Schema 1	Schema 2
id	id	None	None
author_name	screen_name	lower	lower
text	full_text or text	use full_text unless not exist	use full_text unless not exist
created_at	created_at	None	None
hashtags	hashtags and extended_hashtags	union	union & lower
text_searchable	None	None	None

Table 3. Pre-processing steps taken on Decahose data

Column Name	Data Type	Index Type
id	TEXT	btree
author_name	TEXT	btree
text	TEXT	N/A
created_at	TIMESTAMP	btree
hashtags	TEXT	GIN
text_searchable	tsvector GENERATED ALWAYS AS (to_tsvector('english', text)) STORED	GIN

Table 4. PostgreSQL Table Schema

Field Name	Mapping
id	'type': 'keyword'
author_name	'type': 'keyword'
text	'type': 'text', 'analyzer': 'english'
created_at	'type': 'date'
hashtags	'type': 'keyword'

Table 5. Elasticsearch field mapping for the second mapping strategy (i.e., manual mapping)

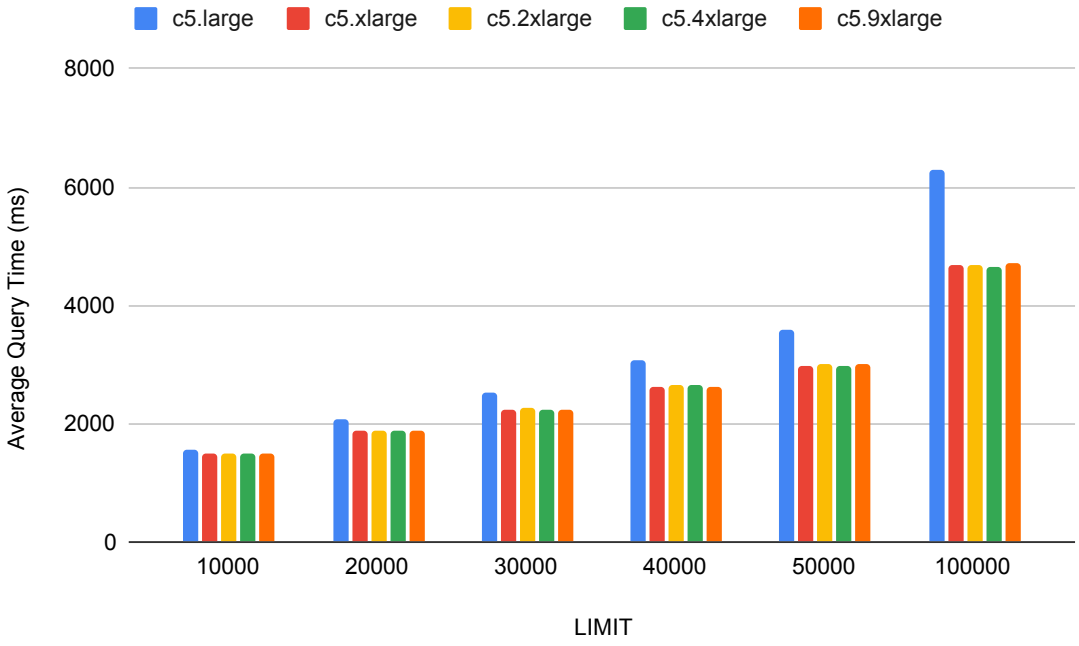

Fig. 3. Query times in PostgreSQL for full text search (query type vi). LIMIT indicates the number of results per query.

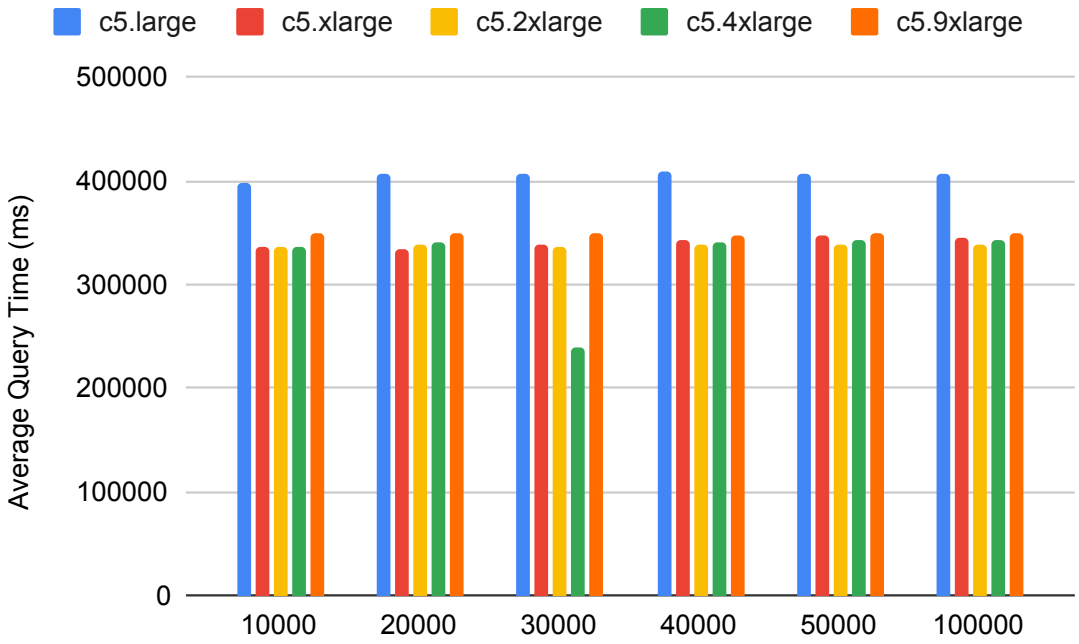

Fig. 4. Query times in Elasticsearch (Mapping 1) for full text search (query type vi). LIMIT indicates the number of results per query.

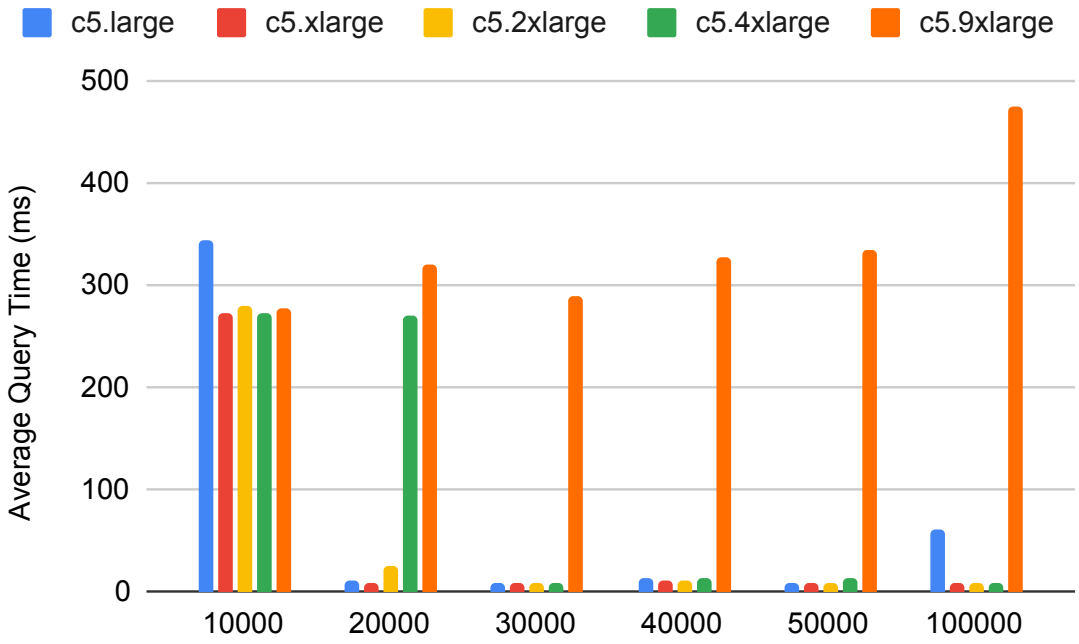

Fig. 5. Query times for Elasticsearch (mapping 2) for hashtag and author (query type iii). LIMIT indicates the number of results per query.

Query Type	Database	Schema or Mapping	EC2 instance type	Query time (ms)	1K Query CO_2 cost	1K Query $ cost
Match 1 hashtag	PostgreSQL	Schema 1	c5.large	235	0.46	0.01
			c5.xlarge	233	0.91	0.01
			c5.2xlarge	234	1.82	0.02
			c5.4xlarge	221	3.44	0.04
			c5.9xlarge	222	7.78	0.09
Match 1 hashtag	PostgreSQL	Schema 2	c5.large	659	1.28	0.02
			c5.xlarge	455	1.78	0.03
			c5.2xlarge	451	3.51	0.04
			c5.4xlarge	437	6.81	0.08
			c5.9xlarge	434	15.21	0.18
Match 1 hashtag	Elasticsearch	Mapping 1	c5.large	8849	17.21	0.21
			c5.xlarge	3682	14.42	0.42
			c5.2xlarge	8198	63.76	0.77
			c5.4xlarge	2701	42.09	0.51
			c5.9xlarge	8005	280.62	3.40
Match 1 hashtag	Elasticsearch	Mapping 2	c5.large	661	1.29	0.02
			c5.xlarge	296	1.16	0.03
			c5.2xlarge	771	6.00	0.07
			c5.4xlarge	678	10.57	0.13
			c5.9xlarge	796	27.90	0.34
Match multiple hashtags with OR	PostgreSQL	Schema 1	c5.large	372	0.72	0.01
			c5.xlarge	342	1.34	0.02
			c5.2xlarge	345	2.68	0.03
			c5.4xlarge	332	5.17	0.06
			c5.9xlarge	335	11.74	0.14
Match multiple hashtags with OR	PostgreSQL	Schema 2	c5.large	828	1.61	0.02
			c5.xlarge	778	3.05	0.04
			c5.2xlarge	757	5.89	0.07
			c5.4xlarge	752	11.72	0.14
			c5.9xlarge	751	26.33	0.32
Match multiple hashtags with OR	Elasticsearch	Mapping 1	c5.large	6981	13.57	0.16
			c5.xlarge	5003	19.60	0.33
			c5.2xlarge	4768	37.08	0.45
			c5.4xlarge	4622	72.03	0.87
			c5.9xlarge	4752	166.58	2.02
Match multiple hashtags with OR	Elasticsearch	Mapping 2	c5.large	623	1.21	0.01
			c5.xlarge	293	1.15	0.03
			c5.2xlarge	609	4.74	0.06
			c5.4xlarge	509	7.93	0.10
			c5.9xlarge	640	22.44	0.27
Match multiple hashtags with AND	PostgreSQL	Schema 1	c5.large	6	0.01	0.00
			c5.xlarge	6	0.02	0.00
			c5.2xlarge	6	0.05	0.00
			c5.4xlarge	6	0.09	0.00
			c5.9xlarge	6	0.21	0.00
Match multiple hashtags with AND	PostgreSQL	Schema 2	c5.large	137	0.27	0.00
			c5.xlarge	82	0.32	0.01
			c5.2xlarge	81	0.63	0.01
			c5.4xlarge	83	1.29	0.02
			c5.9xlarge	81	2.84	0.03
Match multiple hashtags with AND	Elasticsearch	Mapping 1	c5.large	1267	2.46	0.03
			c5.xlarge	887	3.47	0.06
			c5.2xlarge	937	7.29	0.09
			c5.4xlarge	1313	20.46	0.25
			c5.9xlarge	1068	37.44	0.45
Match multiple hashtags with AND	Elasticsearch	Mapping 2	c5.large	74	0.14	0.00
			c5.xlarge	66	0.26	0.00
			c5.2xlarge	69	0.54	0.01
			c5.4xlarge	65	1.01	0.01
			c5.9xlarge	66	2.31	0.03

Table 6. Estimated costs to run benchmark queries in terms of CO_2 emissions (grams of carbon dioxide equivalent) and cloud compute fees (USD). For each query, we limit results to 10,000 records. To make the costs easier to read, we provide the estimates per 1000 queries.

Query Type	Database	Schema or Mapping	EC2 instance type	Query time (ms)	1K Query CO_2 cost	1K Query $ cost
Match author	PostgreSQL	Schema 1	c5.large	51	0.10	0.00
			c5.xlarge	51	0.20	0.00
			c5.2xlarge	51	0.40	0.00
			c5.4xlarge	50	0.78	0.01
			c5.9xlarge	51	1.79	0.02
Match author	PostgreSQL	Schema 2	c5.large	849	1.65	0.02
			c5.xlarge	73	0.29	0.04
			c5.2xlarge	73	0.57	0.01
			c5.4xlarge	73	1.14	0.01
			c5.9xlarge	73	2.56	0.03
Match author	Elasticsearch	Mapping 1	c5.large	1600	3.11	0.04
			c5.xlarge	1103	4.32	0.08
			c5.2xlarge	1046	8.14	0.10
			c5.4xlarge	933	14.54	0.18
			c5.9xlarge	928	32.53	0.39
Match author	Elasticsearch	Mapping 2	c5.large	267	0.52	0.01
			c5.xlarge	216	0.85	0.01
			c5.2xlarge	156	1.21	0.01
			c5.4xlarge	154	2.40	0.03
			c5.9xlarge	174	6.10	0.07
Match author and search full text	PostgreSQL	Schema 1	c5.large	954	1.86	0.02
			c5.xlarge	970	3.80	0.05
			c5.2xlarge	968	7.53	0.09
			c5.4xlarge	967	15.07	0.18
			c5.9xlarge	968	33.93	0.41
Match author and search full text	PostgreSQL	Schema 2	c5.large	1617	3.14	0.04
			c5.xlarge	996	3.90	0.08
			c5.2xlarge	999	7.77	0.09
			c5.4xlarge	1000	15.58	0.19
			c5.9xlarge	1019	35.72	0.43
Match author and search full text	Elasticsearch	Mapping 1	c5.large	2016	3.92	0.05
			c5.xlarge	1809	7.09	0.10
			c5.2xlarge	1976	15.37	0.19
			c5.4xlarge	1825	28.44	0.34
			c5.9xlarge	1947	68.25	0.83
Match author and search full text	Elasticsearch	Mapping 2	c5.large	47	0.09	0.00
			c5.xlarge	37	0.14	0.00
			c5.2xlarge	15	0.12	0.00
			c5.4xlarge	38	0.59	0.01
			c5.9xlarge	292	10.24	0.12
Full text search	PostgreSQL	Schema 1	c5.large	1568	3.05	0.04
			c5.xlarge	1501	5.88	0.07
			c5.2xlarge	1498	11.65	0.14
			c5.4xlarge	1486	23.16	0.28
			c5.9xlarge	1492	52.30	0.63
Full text search	PostgreSQL	Schema 2	c5.large	1960	3.81	0.05
			c5.xlarge	1583	6.20	0.09
			c5.2xlarge	1582	12.30	0.15
			c5.4xlarge	1591	24.79	0.30
			c5.9xlarge	1614	56.58	0.69
Full text search	Elasticsearch	Mapping 1	c5.large	407022	791.43	9.61
			c5.xlarge	327517	1,282.77	19.22
			c5.2xlarge	334060	2,598.24	31.55
			c5.4xlarge	338256	5,271.16	63.89
			c5.9xlarge	342438	12,004.35	145.54
Full text search	Elasticsearch	Mapping 2	c5.large	474	0.92	0.01
			c5.xlarge	192	0.75	0.02
			c5.2xlarge	178	1.38	0.02
			c5.4xlarge	183	2.85	0.03
			c5.9xlarge	1577	55.28	0.67

Table 7. Estimated costs to run benchmark queries in terms of CO_2 emissions (grams of carbon dioxide equivalent) and cloud compute fees (USD). For each query, we limit results to 10,000 records. To make the costs easier to read, we provide the estimates per 1000 queries.

Fuzziness Type	Example
Stemming and Lemmatizing	‘cats‘ (‘ran‘) and ‘cat‘ (‘run‘)
Edit Distance	‘mistakn‘ and ‘mistaken‘
Semantic Similarity	‘coronavirus‘ and ‘covid‘

Table 8. Types of Fuzziness Support for Text Search

	PostgreSQL		Elasticsearch	
Fuzziness Type	Default	Adds-on	Default	Adds-on
Stemming and Lemmatizing	N	Y	Y	Y
Edit Distance	N	Y	Y	Y
Semantic Similarity	N	N	N	Y

Table 9. Types of text fuzziness that each system supports. Default means the function ability comes with installation. Adds-on means there need installing a module, package or system modification to realize the function.