On the Performance of Cloud-based Spreadsheets as a Backend for View-only Web Applications

Andrea Schwertner Charão, Felipe Marin, João Carlos D. Lima, Cristiano Cortez da Rocha, Luiz Angelo Steffenel

On the Performance of Cloud-based Spreadsheets as a Backend for View-only Web Applications

Andrea Schwertner Charão[1], Felipe Marin[1], João Carlos D. Lima[1], Cristiano Cortez da Rocha[3] and Luiz Angelo Steffenel[2]

[1] *Universidade Federal de Santa Maria (UFSM), Santa Maria, RS, Brazil*

[2] *Université de Reims Champagne-Ardenne (URCA), Reims, France*

[3] *Centro de Informática e Automação do Estado de Santa Catarina (CIASC), Florianópolis, SC, Brazil*

Keywords: Cloud Computing, Data Backend, Spreadsheet, Performance Evaluation.

Abstract: The wide offering of cloud-based services brings alternatives to traditional approaches for developing modern information systems. In this work, we examine cloud spreadsheet services as an alternative for data backend in small scale, view-only web applications. We review 6 cloud-based spreadsheets offering data access APIs to third-party applications, then we present a set of performance tests over spreadsheets hosted on Google Sheets. Preliminary findings show a performance penalty for transferring JSON-formatted data and an expressive failed request rate for many simultaneous accesses.

1 INTRODUCTION

Cloud computing is not only changing the way we use software, but also the way we build it. As more and more services migrate to the cloud, traditional components in software architecture may be provided by cloud-based services. Since the early days of cloud computing, the range of services has grown and spanned from personal to corporate applications. Examples of cloud services include word processors, spreadsheets, database managers and a lot more (Miller, 2008).

Databases are usually a core component of large information systems as well as of smaller web and mobile applications. Relational Database Management Systems (RDBMS) are well known for their efficiency in querying and filtering data or processing transactions. As so, they are the first choice for data backend in a wide range of web applications.

Modern web applications can vary greatly in their purposes and architectures. Some are highly interactive websites or mobile applications which perform frequent data read and write operations. Others are "*applications that use backend data, supporting the searching, sorting, filtering and visualizing of the data based on the user input*" (Chang and Myers, 2014). These are view-only web applications, in the sense that the end user only reads data that have been previously registered in the storage backend.

For such class of web applications, some developers are reportedly using cloud-based spreadsheets instead of relational databases as an alternative for data backend (Fisher, 2014; Hankinson, 2015), as many cloud providers offer APIs for reading or writing data onto the online spreadsheets. Arguments in favor of this approach usually mention its simplicity for developers, which do not need to create administrative interfaces for simple operations, as the spreadsheet interface already meets this need. Against this approach, arguments emphasize that spreadsheets and databases are very distinct components and this solution might a prototyping-only alternative.

In technical articles, arguments in favor or against this approach are not based on evidence from experiments and analysis in a given situation, but rather on general knowledge about the matter. Also, real applications relying on spreadsheets provide no public data on their performance.

In this work, we aim to further the discussion around this approach by focusing on performance issues. We begin by reviewing the current offering of cloud-based spreadsheets and examine their service plans and their endpoints for developers. We focus mainly on free services, so as to characterize what is the current entry-level performance. We chose a popular service (Google Sheets) to be the target of a set of performance tests with varying data sizes and number of accesses.

Charão, A., Marin, F., Lima, J., Rocha, C. and Steffenel, L.
On the Performance of Cloud-based Spreadsheets as a Backend for View-only Web Applications.
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018) - Volume 1*, pages 642-647
ISBN: 978-989-758-298-1

The rest of the paper is organized as follows: Section 2 presents a background to this work while reviewing 6 cloud-based spreadsheets offering data access APIs to third-party web applications. Section 3 briefly discusses related work. Sections 4 and 5 respectively describe our methodology and results, while Section 6 draws some conclusions and presents future work.

2 BACKGROUND

2.1 Cloud-based Spreadsheets

Cloud-based spreadsheets have been around as a service for roughly a decade. In (Miller, 2008), some web-based spreadsheets are presented, but few of them remain in the market nowadays. In the next paragraphs, we review some current offerings for this kind of cloud-based service.

Google Sheets[1] is a web and cloud-based spreadsheet from Google Inc., which is the first large provider of a SaaS spreadsheet and other office applications. Google launched its first release on 2006 and its capabilities are evolving since then. Among its features are multi-user concurrent editing, cloud-based availability and storage, multiple export/import options and a REST API for developers. It is available as a free service but is also offer some paid plans.

Zoho Sheet[2] is part of Zoho Docs, a cloud-based office suite from Zoho Corporation. It was launched in 2006 and is similar to Google Sheets, but its pricing plans only offer a free trial version which expires after 15 days. It does offer a publishing capability to export data, but it is not available in the free trial version. Alternatively, there is Zoho Creator, which is targeted to rapid developed of web applications. Zoho Creator allows to import a spreadsheet as data source and to export it through a REST API.

Excel Online[3] is a service from Microsoft Office Online, which offer a web-based office suite. It is a lightweight version of Excel on premises, offering collaboration features as in other cloud-based services. It was launched in 2010, but its REST API was only announced in 2016.

Airtable[4] is a cloud service which extends the typical spreadsheet features by incorporating some facilities for building simple web-based applications (for example, customized cell views, types and controls). It was launched around 2013 and its API was made available in 2015.

Fieldbook[5] is a spreadsheet-database hybrid service in the cloud launched around 2013. Its interface resembles other cloud-based spreadsheets, but it lets one to link sheets so to view and edit related items, as in database powered applications.

Rowshare[6] provide a cloud-based service for managing online collaborative tables. It provides fine-grained capabilities (for example, sharing a row instead of an entire table) and customizable interfaces for table viewing and editing. It was launched in 2015.

To develop web applications using the aforementioned cloud-based spreadsheets, one must rely on their provider's API. To illustrate this point, Table 1 presents example API requests for each of the services. Most of them provide an Uniform Resource Identifier (URI) to address an entire spreadsheet table. Depending on the provider, authentication may be required even for reading data, as for example in Excel Online. Some APIs require more than a single request to get the data, as it may be necessary to query a spreadsheet id which may not be known in advance. This is the case with Rowshare and Excel Online. Also, there may be different ways to get the spreadsheet data, as in the case of Google Sheets, which offers a comprehensive API for data access.

Each provider may have multiple service plans, which define prices and limits for data storage and API usage. Table 2 summarizes service plans for each cloud-based spreadsheet. Most providers offer free plans with lower limits than paid plans.

In Table 3, we present resource limits for the 6 cloud-based spreadsheet services, focusing on their free plans. Google Sheets stand out for its relatively high limits for its free plan. Also, this is the service of choice of some real-world examples using spreadsheets as a data backend, so we chose Google Sheets for the experiments in this work.

2.2 Spreadsheets as Data Backends

Using cloud-based spreadsheets as a data backend for web applications may sound surprising and controversial, as traditional approaches rely on databases. Even so, there is evidence that such alternative has been explored in some cases, as for example in the news and media domain (Fisher, 2014) and even in game development (Hankinson, 2015).

For developers, there are libraries exploring this

[1] https://www.google.com/sheets/about/
[2] https://www.zoho.com
[3] https://office.live.com/start/Excel.aspx
[4] https://airtable.com
[5] https://fieldbook.com
[6] https://rowshare.com

Table 1: Example API Request.

Service	Request Format
Google Sheets	https://spreadsheets.google.com/feeds/ list/¡ID¿/¡SHEET¿/public/values?alt=json
Zoho Creator	https://creator.zoho.com/api/json/¡ID¿/view/ ¡SHEET¿?authtoken=¡TOKEN¿&zc_ownername=¡USER¿&scope=creatorapi
Excel Online	https://graph.microsoft.com/v1.0/me/drive/ items/¡ID¿/workbook/worksheets(¡SHEET¿)/range(address=¡RANGE¿)
Airtable	https://api.airtable.com/v0/¡ID¿/¡SHEET¿?api_key=¡KEY¿
Fieldbook	https://api.fieldbook.com/v1/¡ID¿/¡SHEET¿
Rowshare	https://www.rowshare.com/api/row/loadforparent/¡KEY¿

Table 2: Service Plans.

Service	Plans
Google Sheets	Free, Basic, Business, Enterprise
Zoho Creator	Per user: Free, Basic, Premium. Unlimited users: Express, Express Plus, Ultimate
Excel Online	Free
Airtable	Free, Plus, Pro, Enterprise
Fieldbook	Free Trial (=Business, 7 days), Pro, Business, Enterprise
Rowshare	Free, Business

alternative, as for example Sheetsee.js [7] and Tabletop.js[8]. Both are Javascript libraries targeted to developing web applications that store data on Google Sheets. Another resource for developers is Tarbell[9], which uses Google Sheets as a content manager for building websites. A similar approach is taken by Tabledo[10], which is a service based on Fielbook spreasheets.

Using a cloud-based spreadsheet in place of a database may be advantageous because it brings with it a simple yet effective, well-known web interface for data management. The web-based spreadsheet user interface support insert, update and delete operations, which would otherwise need to be implemented for managing data records. Also, as they are built for collaboration, cloud-based spreadsheets usually offer easy access management features for granting read-write permissions.

A drawback of this approach is related to the limits imposed by the cloud service provider. This affects even paid service plans, so scalability is usually compromised. Also, performance may be a problem, as data transfers may take long and there is generally no guarantees in this regard.

[7]http://jlord.us/sheetsee.js/index.html
[8]https://github.com/jsoma/tabletop
[9]http://www.tarbell.io/
[10]http://tabledo.com

3 RELATED WORK

Many technical articles present cloud-based spreadsheets as a simple yet effective solution for data backend in web or mobile applications (Noble, 2015; Stanoev, 2015; Doubintchik, 2016; Dayagi, 2017). They highlight the ease of use of this approach and mostly recommend it for simple applications and prototyping.

From a different perspective, some authors have conducted a thorough investigation on cloud-based spreadsheets as statistical software (McCullough and Yalta, 2013). They point out many limitations, but do not focus on performance.

Focusing on performance, some authors evaluate traditional relational database management systems in cloud environments (Bini et al., 2014; Litchfeild et al., 2017). Findings indicate inefficiencies related to the cloud environment.

To the best of our knowledge, no previous work has addressed the performance of cloud-based spreadsheets as a replacement for relational databases in small-scale, view-only web applications.

4 METHODOLOGY

We conducted all experiments on a machine with an Intel Xeon E5620 2.4 GHz quad-core HT processor, 12 GB of DDR3 RAM and one 512 GB SATA disk.

Table 3: Limits for cloud-based spreadsheet services.

Service	Size limits	Request limits
Google Spreadsheets	2,000,000 cells / spreadsheet	20 / second
Zoho Creator	1,000 rows / spreadsheet	250 / day
Excel Online	10 MB / spreadsheet	N/A
Airtable	1,200 rows / spreadhseet	5 / second (after exceeding this rate, clients have to wait 30 s before subsequent requests will succeed)
Fieldbook (Free Trial = Business)	25,000 rows / spreadsheet 500 columns / sheet	25,000 / month
Rowshare	1,500 rows / spreadsheet	N/A

The server runs Debian 6 (Squeeze) operating system and is connected to a 1 Gigabit Ethernet switch.

The server was fully dedicated to this work and all experiments were carried out in a time frame of low internet traffic. However, the machine is located in a large institutional network with more than a thousand installations, so the internet link was naturally shared and subjected to uncontrollable traffic variations.

We prepared a set of 8 tables with varying number of rows (1000 to 20000) and columns (10 to 100). All cells are text fields of the same size (100 characters). These table properties take into account the service limits presented in Table 3.

Using a free account of Google Sheets, we imported those tables as spreadsheets and adjusted sharing/publishing permissions as to make them available as JSON-formatted data using REST API requests.

For the testbed, we used Apache JMeter to perform two test plans for simulating: (i) a single user requesting each one of the spreadsheets and (ii) multiple users requesting a single spreadsheet. All tests used a ramp-up period of 1 second. The first test plan allowed us to examine the impact of different table sizes on the service response and, consequently, on the elapsed response time perceived by the user. The second test plan was conceived to investigate how the service responds to different request rates.

To favor the replication of the experiments in other environments, all test scripts are available online at: https://github.com/AndreaInfUFSM/iceis2018-cloudspreadsheet.

5 RESULTS

In this Section, we describe the results we obtained from our testbed.

5.1 Single User Requests

In Table 4, we present response times for a single user requesting spreadsheets of varied sizes from Google Sheets, in JSON format. The transferred size for each requested spreadsheet comprises all table data plus JSON-formatted metadata. This size grows as we add rows or cols, and different table dimensions (rowsXcols) lead to different transferred sizes, even for a constant number of cells (see, for example, dimensions 1000x100 and 10000x10, both with 100000 cells).

As we can observe, the average response time grows with the transferred size, and even a relatively small table takes more than a second to be transferred. Also, minimum and maximum values for response times may vary significantly. This may be attributed to both the cloud service provider and the network traffic.

5.2 Multiple User Requests

In Table 5, we present response times for simultaneous requests over the smallest table (1000x10). As we increase the number of (virtual) users, we can observe the response times grow. This may be due to the cloud service delaying the response, but also due to a client overload. In either case, a web application running on a single machine will notice such response times when serving multiple users.

As we exceed the API request limits, some requests may fail and return an error code instead of the actual data. In this experiment, there are no failed requests for up to 200 users. Starting from 250 users, error rates grow and can reach high percentages. Even so, we may expect some requests to be successful if they are sent just after a failed batch, as Google Sheets do not impose a waiting time before subsequent requests succeeds.

Table 4: Response times for a single user requesting tables of varied sizes from Google Sheets.

		Response Time (ms)			
Rows x Cols	Size (KB)	Average	Min.	Max.	Std. Dev.
1000x10	4582	1996	1613	3867	658
1000x100	42094	5763	5460	6727	423
2000x10	9167	2435	2198	3285	333
2000x100	84224	10592	9905	12169	709
10000x10	45849	6389	5752	7239	586
10000x100	421307	48068	44059	65799	6241
20000x10	91701	11357	10641	13032	798
20000x100	842655	102179	91169	134397	11666

Table 5: Response times for simultaneous user requests over a 1000x10 table from Google Sheets.

Virtual	Response Time (ms)			Std.	Error
Users	Average	Min.	Max.	Dev.	(%)
10	4001	2014	8949	1933	0.0
20	7219	1920	21751	4748	0.0
50	13856	2434	26447	6058	0.0
150	39491	2489	64130	18258	0.0
200	54417	2355	84309	25146	0.0
250	57372	2387	93441	26789	14.8
300	58753	2982	95241	28746	29.6
500	61061	5921	105279	26723	56.4
700	62294	3026	102273	27711	67.8
900	61111	6254	103616	28993	75.1
1000	65230	3306	108314	29855	76.1

6 CONCLUSIONS

In this work, we conducted a preliminary investigation on the performance of a prominent cloud-based spreadsheet service, Google Sheets, which is used as a backend of some existing view-only web applications. We used the provider's API that returns data in JSON format, as mentioned in technical articles from developers adopting this approach.

The results indicate a significant penalty in getting JSON-formatted data from the cloud-based service, as the provider includes numerous metadata. This may not be a problem for the smallest cases, but may become a performance bottleneck as we add rows and columns to a table.

Our experiments also show the response times are significant and highly variable. This may be expected from a cloud-based service which does not mention this metric in its pricing plans. This may also occur in other non-cloud approaches where data is hosted far from the application server, where network traffic has great influence. Equally importantly, our experiments show to what extent the service supports concurrent requests.

Taking all this into account, it is important that developers made an informed decision when considering this approach. Also, when adopting such data backend, it is necessary to employ techniques to counterbalance the drawbacks of this approach, as for example using asynchronous tasks for data transferring.

Although this study is preliminary and has only covered a free service, our test scripts may be reused as-is to analyze the performance of paid plans from the same provider. It should be noted, however, that the testbed has limitations, as it issues concurrent requests from a single machine. This simulates a scenario where the web application is hosted on a single server, which consequently is the only source of all requests sent by the application to the cloud-based service. While this is a real-world scenario, the results are influenced by the concurrency on both the requester and the provider.

As future work, the testbed may be extended to other cloud-based spreadsheet services and include experiments with fine-grained operations offered by some APIs. Also, a comparison with other approaches for data backend may be useful as a reference to developers focusing on small-scale, view-only web applications.

REFERENCES

Bini, T. A., Sunyé, M. S., and Lange, A. (2014). Cloud computing - an evaluation of rules of thumb for tuning RDBMSs. In *ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems, Volume 1, Lisbon, Portugal, 27-30 April, 2014*, pages 187–192.

Chang, K. S.-P. and Myers, B. A. (2014). Creating interactive web data applications with spreadsheets. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 87–96, New York, NY, USA. ACM.

Dayagi, G. (2017). Get sheet done – Using Google Spreadsheets as your data backend. Available at: http://crwd.fr/2wzmJee.

Doubintchik, M. (2016). Retrieve Google Spreadsheets using JSON feed & output to HTML. Available at: https://allurewebsolutions.com/google-spreadsheets-json.

Fisher, T. (2014). How we built Borderland out of a spreadsheet. Available at: http://blog.apps.npr.org/2014/04/23/how-we-built-borderland-out-of-a-spreadsheet.html.

Hankinson, W. (2015). Turning data into enemies: How we used google spreadsheets as a CMS for Unity in Defend the Dam. Available at: https://www.gamasutra.com/blogs/WillHankinson/20150323/239489/.

Litchfeild, A., Althwab, A., and Sharma, C. (2017). Distributed relational database performance in cloud computing: an investigative study. In *AMCIS 2017 - Proceedings of the 23rd Americas Conference on Information Systems*.

McCullough, B. D. and Yalta, A. T. (2013). Spreadsheets in the cloud - not ready yet. *Journal of Statistical Software*, 52(7).

Miller, M. (2008). *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*. Que Publishing Company, 1 edition.

Noble, C. (2015). Quantified anything: Turning Google Sheets into a backend database. Available at: https://nobleintentstudio.com/blog/google-docs-as-a-backend/.

Stanoev, K. (2015). Using Google spreadsheets as data source in your Android app. Available at: https://www.telerik.com/blogs/google-spreadsheet-as-data-source-android.