Descriptive Modeling of the Web Mapping Systems Users' Behavior

Vinícius G. Braga[1], **Welder Oliveira**[1],
Vagner Sacramento[1], **Kleber V. Cardoso**[1]

[1]Federal University of Goiás (UFG)
Institute of Informatics (INF)
Goiânia – Brazil

{viniciusgoncalves, kleber, vagner}@inf.ufg.br

welder.oliveira@gogeo.io

Abstract. *Web mapping systems and Geographic Information Systems (GIS) in general are widely used nowadays. However, there is a lack of models that describe how users interact with this kind of application and the workload imposed on servers. These models are important for performance evaluation and to direct the focus of the development to the points that will affect the most used operations and the most users of the system. In this paper, we propose a descriptive model of the web mapping systems users' behavior. The model can be used as a starting point for the creation of workload generators to make performance evaluations in this kind of application. As a case study we created a workload generator and we applied it in a web tile server. We show that adding user features can significantly change the workload imposed on the server.*

1. Introduction

Web mapping systems, such as Google Maps and Bing Maps, are increasingly becoming part of people's lives and are used for different purposes. Besides these, there are several other applications that use Location Intelligence and Location Based Services in logistics, Business Intelligence and CRM systems. They are developed using either commercial (e.g. ArcGIS) or open source GIS platforms (e.g. PostGIS/GeoServer). A critical problem in GIS applications development is to ensure performance and scalability as the number of users and the data volume increase. The main factors that affect the performance of a system are the design, the implementation and the workload to which the system is subjected [Feitelson 2014]. The first two factors are well known by developers, but the last one is sometimes neglected.

Usually, performance of systems are evaluated using stress benchmarks [MAPLARGE 2014, OSGeo Wiki 2011], which are proper to help to detect bottlenecks and to compare a system with others. However, to analyze the performance of a system only using stress benchmarks is not a good choice. This kind of benchmark is very different from the workload of a system in production. They do not show the features of the system that will be most used in production and do not give a clear idea of what to priorize in the system's development.

Knowledge of the real workload allows a better resources allocation and a better cost-benefit in systems development. The developers can direct the improvement focus to

the points that will affect the most used operations and the most users. Without the correct focus, a lot of effort can be applied to improve functionalities that will be underused and vice versa. To know the workload is also important to investigate better caching strategies [Romoser et al. 2012].

Despite the great importance of knowing the real workload, there is a lack of studies in the literature on GIS user behavior and their impact on the workload imposed on servers. Some works try to simulate a real GIS workload and its effects on geographical databases [Ray et al. 2011, Simion et al. 2012]. Some works use the history to the server to improve the server cache strategies [García et al. 2012, Quinn and Gahegan 2010]. Romoser et al. (2012) created a workload characterization of the USGS EROS system, a system that allows its users navigate and download images of the Earth. However, we did not find any studies that characterize the typical user behavior and that model the workload imposed to web mapping applications.

In this paper, we carry out an investigation about Google Maps users behavior and we propose a descriptive model of the typical user behavior. The model describes several user behavior aspects, such as the distribution of the intervals between actions, the zoom levels frequency distribution, the distribution of session duration, and the frequency of the most used operations. The model also describes how users pan on a map and how the number of tile requests are intrinsically correlated with the screen resolution. In order to evaluate the impact of the model in a real system, we have created a synthetic workload and we compared it with a workload used by industry. The main contribution of this work is the description of a model that can be used as a starting point to create a workload generator to simulate real user behavior on web mapping applications.

This work is organized as follows. In Section 2, we present a brief description of web mapping systems and we talk about related works. In Section 3, we describe how the data were collected and analysed. In Section 4, we discuss results from data analysis and we present a descriptive model, which characterizes the users in several aspects. In Section 5, we show a case study with a workload generator based on the results of the data analysis and applied it to a web tile server. In Section 6, we present the final comments and we introduce some perspectives for future work.

2. Background and Related Work

Web mapping systems renderize the map in a set of fixed scales and divide it in images of the same size, called tiles. The number of tiles grows exponentially as the zoom level (scale) increases, following the 4^{zoom} rule. Google Maps, for example, works with tiles of 256x256 pixels. The number of tiles at zoom level 0 is 4^0, that is, just one tile. At zoom level 1, it is 4^1, that is, 4 tiles and so on, as can be seen in Figure 1, which shows the number of tiles increasing up to the zoom level 2. The highest zoom level that the application permits is level 21, in which the map has a resolution of 256x256 x 4^{21} pixels. This model has become a standard for web maps. It allows users request only the images related to the map part they want to see.

Despite the large use of web mapping systems, few research works seek to create a real workload model to this kind of system. Romoser et al. (2012) analyzed the logs of the USGS EROS system, a system that let users navigate and download images of the Earth, seeking to study the workload imposed to this system. They investigated the

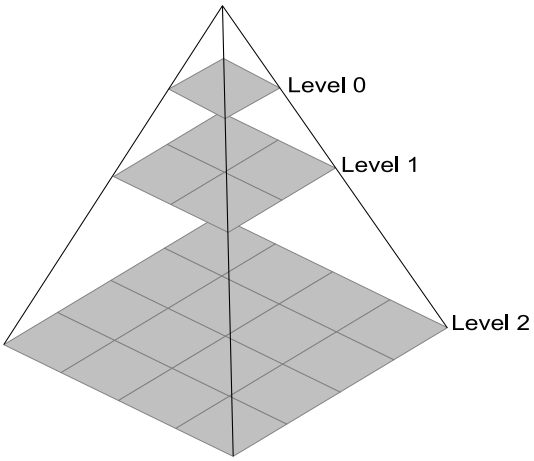


Figure 1. Tile pyramid with three levels.

accesses to the system and characterized them by user, by images and by requests. They found the most requests came from a little set of users, as well as these requests are made to a little set of popular scenes (specific places of the globe). Romoser et al. (2012) found a pattern of access to images in big disasters dates, such as earthquakes and tsunamis. These patterns were used to improve the cache and prefetching techniques, looking for improvements on system performance.

Kang et al. (2001) created an algorithm to perform prefetching of the most probable tiles to be requested based on the updated global access pattern. Furthermore, they created an algorithm to substitute the tiles in cache based on the same probabilities. The authors did not present any study of real data access on web GIS systems to prove the effectiveness of their algorithms. Other authors have implemented a prefetching strategy based on the history of previous accesses, identifying the most accessed areas and storing them in cache [Kefaloukos et al. 2012, García et al. 2012].

Ray et al. (2011) present Jackpine, a tool to benchmark spatial databases. The authors modeled several common access scenarios to these databases to simulate a real workload. Although they address several spatial database operations, the workload generator was created without taking into account several real GIS access aspects, such as think time, that is the time users take to analyze the map and perform the next action. In a later work [Simion et al. 2012], Jackpine was used to produce concurrent accesses to categorize different microbenchmarks based on CPU and disk usage. However, the workloads used by them were defined with no statistical analysis of real data, they used only good sense and the authors' experience in the industry. Whithout a statistical analysis there is no way to say that their workloads are close enough to real GIS workloads.

Zhang et al. (2007) analyzed traces of heterogeneous applications to characterize the workload imposed to the servers. They modeled the CPU usage statistically and created strategies of anomaly detection and capacity planning. Cibulka D. (2013) developed a study about the influence of the latitude, longitude and scale in the response time to recover a tile in web mapping systems. The results showed that the response time is bigger in regions with a greater density of objects, although he did not mention the cache effects.

3. Conceptualization and Methodology

Geographic Information Systems let users reference information with their locations on a map and visualize them in different types of maps. GIS allow the overlay of different georeferenced data layers that can be useful for many studies. An important use of this technology is in the creation of web mapping systems. These systems are popular and so simple to use that users do not need to have GIS knowledge to work on them.

Google Maps is one of the most popular web mapping systems of the world. Some of its operations, such as zoom and pan, are common to any web mapping system. The knowledge of the use of these operations can be useful in the characterization and modeling of the typical workload imposed to this kind of system. The latest Google Maps version, available for all users since February 2014, presents several useful information in its URL. It is possible, for example, to extract the central coordinate, the zoom level, the searches, routes, among other information. Google Maps changes the URL as a result of user actions. For example, if user moves the map, the central coordinate value is changed. This allows a reconstruction of the users' actions from the set of URLs of a Google Maps' navigation session. These points were crucial for the choice of this system in this work.

We have done a collect of a large set of accesses to Google Maps, collecting URLs and other information. With these data we have made several analysis and we characterized the users on several aspects. This section presents basic concepts about Google Maps and its URLs pattern and describes the collect and data analysis methodologies.

3.1. Google Maps

Google Maps is a popular map system and has a web and a mobile version. In this study, we focused on the web version. This version updates the URL as a result of user actions and the URLs have a lot of useful information that describe these actions. This allows the user actions to be reconstructed from the set of URLs of a navigation session. The two most basic information in the URLs are the geographic coordinate and the zoom level. Below is an example of a URL with both this information highlighted in bold. The first two numbers represent the latitude and longitude and the latter, with the "z", is the zoom level, which ranges from 0 to 21 on the road map.

- https://www.google.com/maps/@**37.3075744,-95.4133961,4z**

In addition to information about the coordinate and zoom level, it is possible to extract information about searches, routes, StreetView visualization, among others. Due to space restrictions, we will not present all the patterns, but all the cited information can be easily extracted with the right regular expressions.

3.2. Data Collection Methodology

First, we contacted companies and public agencies that work with web mapping systems and requested their systems access log. However, either they did not have this information or were unable to provide it for security reasons. In view of this, we decided to develop our own solution to collect data. We observed that the new Google Maps presents several useful information in its URL that can be used to identify user behavior features.

We developed a Google Chrome extension to collect these URLs and other information. The extension collects the Google Maps URLs and some mouse movements

inside the page, all with its respective timestamps. Additionally, it collects the page resolution in pixels, which is equivalent to the width and height of the part of the map that appears on the screen. The page resolution is collected for every user action, because the page can be resized during the session. The extension was published on Chrome Web Store and with about 120 users we have collected more than 60,000 URLs, together with other access information.

We defined a user navigation session as the time the user spends on Google Maps. This time starts when the user accesses the Google Maps page and ends when he finishes his work, either by closing the tab/window or accessing another address in the same tab. During a navigation session on Google Maps, the extension collects data on user actions. Once the user logs out, the information is assembled and sent to a server.

3.3. Data Analysis Methodology

Initially, we analyzed the timestamps of each user action and from them we identified the intervals between actions. To calculate them we created an algorithm that iterates on a set of URLs from a user session and performs the subtraction $T_{i+1} - T_i$, where T_{i+1} and T_i are the timestamps of URL_{i+1} and URL_i, respectively. The intervals represent the user think time, in other words, the time it takes to analyze the map and perform the next action. Figure 2 shows the collection time of the timestamps and what we consider to be a think time. The time T_1 was collected at the end of User Action 1, the time T_2 at the end of User Action 2 and so on. To obtain the Interval 1, we just have to subtract T_1 from T_2.

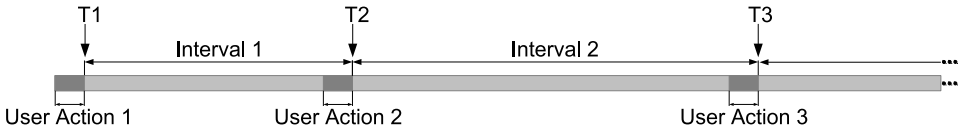


Figure 2. Interval's calculation.

Using the URLs we extracted some important information. First, we extracted the zoom level on the road map, which is the default map. The zoom level ranges from 0 to 21, and a greater zoom level means a greater scale. With this information, we were able to find the frequency of each zoom level, as shown in Figure 3(a). On the hybrid map (the map with satellite images), the zoom level is replaced with a distance information. This information varies a lot, and because of this, it were not used in the zoom level analysis.

We also analyzed the URLs in pairs to identify the action performed by the user. For that, we developed an algorithm that iterates over each set of URLs, use the Algorithm 1 to identify the actions and create an ordered list with the actions of a navigation session. Algorithm 1 analyzes two URLs in sequence and checks the change between URL_i and URL_{i+1} to define what kind of operation was performed by the user. The analysis of the first URL of a navigation session is a special case, because this URL has no predecessor, so the $urlI$ parameter is passed with a $NULL$ value and the operation is identified as $BEGIN$ type. To illustrate, suppose URLs 1 and 2 represent two sequential accesses of the same user. As can be seen, there was a change in the zoom level, passing from 4z to 5z. This change is then recorded as a zoom operation.

1. www.google.com/maps/@37.0625,-95.677068,**4z**
2. www.google.com/maps/@37.0625,-95.677068,**5z**

In many cases the URL changes in more than one part. However, different changes occur as a result of just one action. For example, let URLs 3 and 4 be two sequential accesses of the same user. As is highlighted, the URL has changed both the zoom level and central coordinate. Nevertheless, the action is recorded as a zoom, because the change in the central coordinate was due to the zoom, which is directed by the mouse pointer position when the *mousewheel* event occurs, not because of the pan. In Algorithm 1, one can see that the evaluation of zoom level change (line 8) occurs before the evaluation of the change of the central geographic coordinate (line 10), which means that the counting is done correctly. During the search and route operations, the central coordinate and the zoom level can also be changed. For this reason, Algorithm 1 evaluates changes in the route and search at first, detecting the performed operation correctly.

3. www.google.com/maps/@**37.0625,-95.677068,4z**
4. www.google.com/maps/@**38.2971386,-65.7063659,5z**

As shown above, search and route operation are recorded when a change occurs in the URL part that refers to them. For example, let URLs 5, 6 and 7 be sequential accesses of the same user. When the user searches for **drugstore**, the URL 5 changes to include the search information, as can be seen in bold in URL 6. This change causes the action to be accounted as a search operation. If the user performs a pan, the URL will continue with the search information, as can be seen in URL 7. However, as there was no change in the search part, the verification in line 4 of Algorithm 1 will return *false* and the operation will be correctly recorded as a pan.

5. .../maps/@37.0625,-95.677068,4z
6. .../maps/**search/drugstore/**@37.0625,-95.677068,4z/...
7. .../maps/**search/drugstore/@40.2194479,-80.7356618**,4z

Algorithm 1 Checks what changed between URL_i ($urlI$) and URL_{i+1} ($urlIPlus1$) and identifies the operation based on this change.

Input: $urlI$ and $urlIPlus1$
Output: The $operation$ the user carried out.

1: $operation$;
2: **if** $urlI = NULL$ **then**
3: $operation \leftarrow new\ Operation(Type.BEGIN)$;
4: **else if** $searchChanged(urlI, urlIPlus1)$ **then**
5: $operation \leftarrow new\ Operation(Type.SEARCH)$;
6: **else if** $routeChanged(urlI, urlIPlus1)$ **then**
7: $operation \leftarrow new\ Operation(Type.ROUTE)$;
8: **else if** $zoomChanged(urlI, urlIPlus1)$ **then**
9: $operation \leftarrow new\ Operation(Type.ZOOM)$;
10: **else if** $coordinateChanged(urlI, urlIPlus1)$ **then**
11: $operation \leftarrow new\ Operation(Type.PAN)$;
12: **else**
13: $operation \leftarrow new\ Operation(Type.ANOTHER)$;
14: **end if**
15: **return** $operation$;

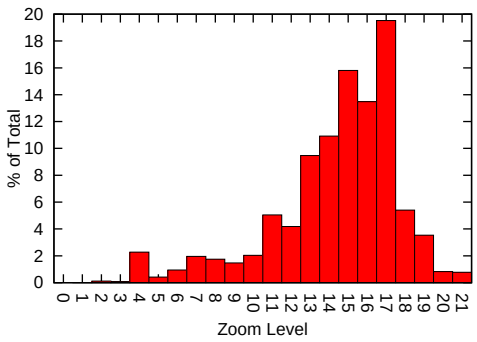
By using a coordinate and a zoom level it is possible to calculate the pixel and the corresponding tile on the map, as can be seen in the example of Google Maps API [Google Maps API 2014]. We implemented this algorithm to identify the tiles visualized on the users' screen. Each URL of the road map has the central coordinate and zoom level of the visualized map. With both these information, we calculated the central pixel position and using the page resolution we found the Bounding Box of the user screen.

We also collected the mouse positions during drag movements so that we could compute the pans sizes, but due to inertia effect on map, this information could not be used with confidence. Hence, we used the central geographic coordinate and the zoom level information of the road map's URLs to make this calculation. In every identified pan operation we computed the size of the central pixel coordinate movement using the Google Maps algorithm [Google Maps API 2014]. The distance the central pixel moves is the same distance that all map moves, which means the computed information is the pan size.

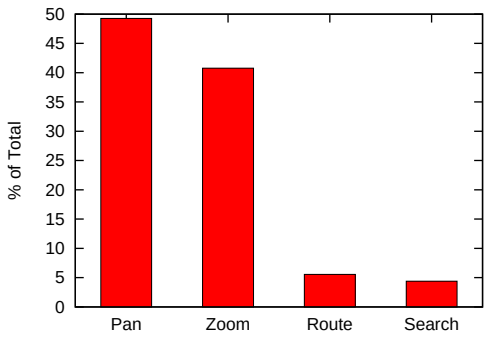
4. Descriptive Modeling

In this section, we present the results of the data analysis. All the results reveal some important user behavior characteristic, which can be used to create models and workload generators that are closer to the normal user behavior. The value of this kind of tool to systems development is described very well in Dror G. Feitelson's book [Feitelson 2014].

Figure 3(a) shows a histogram with the frequency of use of each zoom level. As can be seen, some zoom levels are used more than others and there is a greater concentration aroud zoom levels 15, 16 and 17. Zoom level 17 is the most widely used because Google Maps redirects the map automatically to this level as a result of a specific search at city level and zoom 17 allows a good navigation with a high level of details. Zoom levels greater than 17 are very close to the Earth and make the navigation difficult, which explains its low use. Very small zoom levels have a low level of details and do not bring value to most searches, that are for information and places at city level. Zoom levels closer to, but lower than, zoom 17 have a good level of details too and allow a good navigation, which explains their high use. A similar behavior was found by Garcia et al. (2012) when they analyzed traces of map servers.



(a) Histogram of the use of zoom levels.

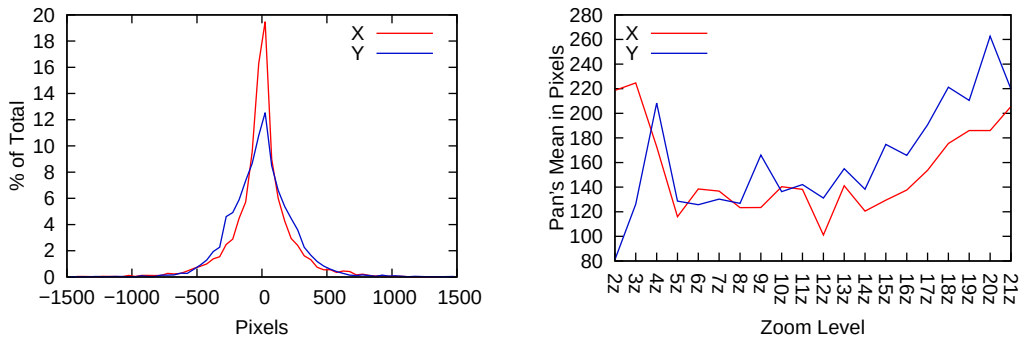


(b) Frequency of pan, zoom, route and search operations.

Figure 3. Evaluation of use of zoom levels and of the frequency of the four most widely used operations (b).

We made a comparison between the frequency of the most common operations performed on Google Maps. We considered the four most common operations: pan, zoom, search and route. Figure 3(b) shows the frequency of each one. Each stage of the route was treated as a route operation, since it entails spending resources on the server. As expected, pan and zoom occur with a higher frequency and a workload generator must take these frequencies into account.

However, it is not enough just to know that pans and zooms are the most frequently performed operations. We have to know how these operation are carried out, as well as, how users normally pan and zoom on a map. In view of this, we decided to conduct other analyzes. Figure 4(a) shows a histogram of the pans sizes, in pixels. The negative values represent the pans to the left and to the top on the x and y-axes, respectively, and the positive values to the right (x-axis) and to the bottom (y-axis). The number of pans are the same in both axes. As can be seen, most pans are small and, in general, users tend to make larger moves on the y-axis. This might be due to the widescreen format, which provides less information on the y-axis and results in a need for larger moves in this direction. As the widescreen format is currently the most used [StatCounter Global Stats 2014], most users move the map in this way. Figure 4(b) shows the average of the moves on the x and y-axes per zoom level and illustrates more clearly that the moves on y-axis are larger.



(a) Histogram of the pans size, in pixels.

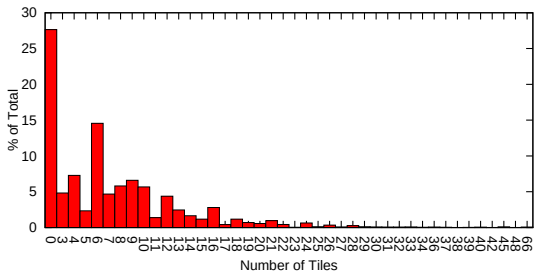
(b) Average size of pans, in pixels, per zoom level.

Figure 4. Evaluation of pan operation.

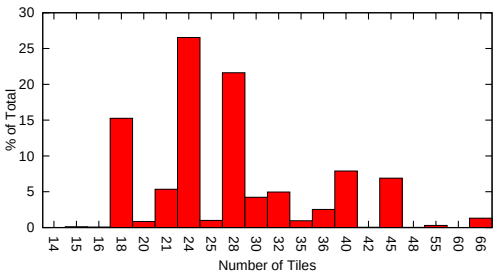
Pans and zooms were analyzed in terms of number of tiles requested by each operation and the results are shown below. Figure 5(a) shows a histogram of the number of tiles requested per pan operation. In most cases, the number of tiles is zero and this happens because of the large number of small pans that just show parts of the tiles that were already loaded and did not appear on the screen. Owing to the screen resolutions, that support at least three tiles on each axis, no pan generates a requisition of just 1 or 2 tiles. Currently, the most widely used screen resolution are 1366x768 [StatCounter Global Stats 2014], which allows a visualization of 6 tiles on the x-axis. This might explain the large number of pans that request 6 tiles, since the largest pans are made on the y-axis.

Figure 5(b) shows the histogram of the tiles requested per zoom operation. In this case, the number of tiles is directly proportional to the screen resolution. As can be seen, the top most number of tiles requested are 18, 24 and 28 and this can be justified by the fact that the most commonly used screen resolution is 1366x768. Depending on how tiles

are arranged, this resolution can fit 6 or 7 tiles on the x-axis and 3 or 4 tiles on the y-axis. If we multiply these values, we have the values 18, 24, 28 and 21, which is one of the highest values too. Frequencies of 40 and 45 tiles requested are greater than 21 and this is because that values are equivalent to the number of tiles requested on the second most widely used resolution, 1920x1080. In this resolution, the clients can request 32, 36, 40 or 45 tiles per zoom.



(a) Histogram of the number of tiles requested per pan operation.



(b) Histogram of the number of tiles requested per zoom operation.

Figure 5. Number of tiles requested per pan (a) and per zoom (b) operation.

So far, we have discused how users perform zoom and pan operations, but there is other important information which is when and for how long. Figure 6(a) shows the PDF of the intervals between user actions and Table 1 displays the mean and some important percentiles of these intervals. Although there are long intervals, some longer than 1 minute, they are not representative. As can be seen in Table 1, more than 95% of the intervals are shorter than 15 seconds. This information is very important to simulate the think time of the users and it was employed in our case study, outlined in Section 5. Figure 6(b) shows the CDF of the durations of the sessions. We made two analyzes: one directly with the collected data (Normal) and another where we only took account of sessions where there was no interval between actions greater than 1 minute (1 Min Interval's Cut). We carried out this second analysis to remove sessions in which the user leaves Google Maps open and starts to do other things. Although this kind of session has a long duration, it does not have an engaged user and there are no requests for long periods. Sessions with intervals shorter than one minute are smaller in average, but have more engaged users that are responsible for the most load on the servers.

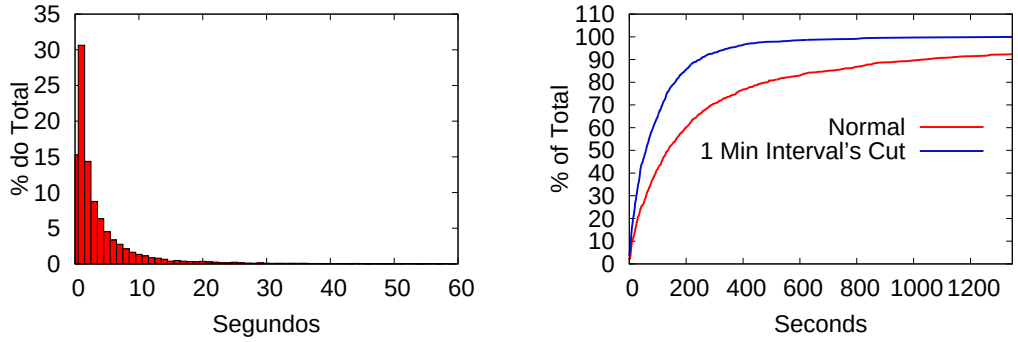
Mean	25th Percentil	50th Percentil (Median)	75th Percentil	95th Percentil
4297 ms	1302 ms	2320 ms	4608 ms	14347.1 ms

Table 1. Mean and percentiles of the intervals between user actions.

All the operations shown here are common to any web mapping system. Although the distributions might be different in other systems, the characteristics are common and the model can be extended to any web mapping system. This model can be used as a starting point to create a workload generator and simulate basic real user features, such as pans and zooms operations, the think time, among others.

5. Case Study With goGeo

Stress benchmarks are good to compare the performance between systems and to understand their behavior in stress situations. However, they do not give a clear idea of how



(a) Histogram of the intervals between actions.

(b) CDF of the user sessions durations.

Figure 6. Evaluation of the intervals (a) and of the sessions durations (b).

real users will interact with the system. Without the right workload developers of a system cannot know how many real users the system can support. For this reason, there is a need to know the real workload. Since the real workload is unknown until the system goes into production, use a synthetic workload closer to the real is the best option. In Session 4, we described some characteristics of the behavior of web mapping systems users and we argued about the importance of knowing these characteristics to create workload generators. For this session, we used one of that characteristics, the intervals between actions, to create a workload closer to the real and to compare it with a stress workload. Adding the intervals, we have a workload that simulates the users' think time and that gives us a better idea of how frequently users send requests to the server.

We used the goGeo's[3] infrastructure and its web tile server to execute the tests. All tests were done with the same machines and using a 151.3 MB database of over 500,000 companies in Brazil. The set of requested tiles covers the entire Brazilian territory. The server cache was disabled to avoid interfering in the results, so that all requests require a new tile rendering. Each test ran for a period of 10 minutes and we measured the response times of each request to compare the results of the workloads.

First, we executed a stress workload based on the MapLarge performance tests [MAPLARGE 2014]. We set 25 external clients to send requests for random tiles as quickly as possible, without intervals between subsequent calls. We computed the average response time of this workload, which was 105 ms. After that, we executed some workloads with the users' think time, starting with 25 clients and increasing this number until it reached a similar average response time of the stress workload. We used an empirical distribution, based on the distribution shown in Figure 6(a), to simulate the users' think time. With 800 clients the average response time was even smaller (71 ms), but with 1,000 clients it was greater (158 ms). This means that, with a similar average response time, the system can support a number of clients between 800 and 1,000 when the intervals are taken into account.

The difference between the stress workload and the user based workload was not just in the number of clients. When we added the intervals we observed a significant increase in the autocorrelation of the response times. Figure 7 shows the autocorrelation

[3]www.gogeo.io

function (ACF) of the workloads[4]. As can be seen, there is a reasonable difference between them. While the stress workload shows autocorrelation until a time lag of 6×10^1, the user based workload reach a lag of 2.5×10^2 and 1.8×10^3 for 800 and 1,000 clients, repectively. A workload that achieves an average response time similar to the stress workload would reach a time lag between these last two values. This means that, although the average response times are the same, the time lag of the user based workload is about one order of magnitude greater than the lag of the stress workload. In this case, the higher lag in response times is related to a higher lag in the users' requests, once the set of called tiles are the same in all the tests.

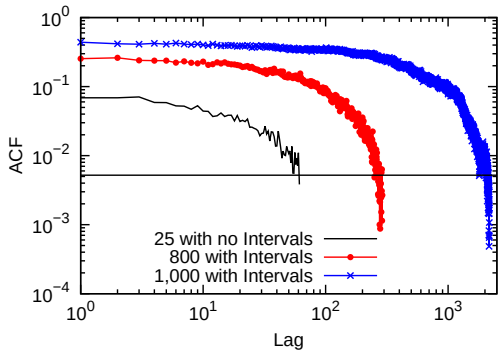


Figure 7. Evaluation of the autocorrelation.

Although the user based workload generator implemented here does not have all the characteristics of real users, it demonstrates what we want to show: adding real user features can significantly change the workload imposed on the server.

6. Conclusion and Future Work

The use of geographic information systems has increased in recent years, but there is still a lack of models for describing user behavior in this kind of application. The lack of such models makes it difficult to carry out a performance analysis and to estimate the capacity of new systems. Without a suitable workload, developers can end up by spending too much effort on system features that are rarely used and vice versa. In this study, we proposed a descriptive model of the behavior of web mapping applications users and we implemented a workload generator taking into account one of the users characteristics, the think time, to compare the results with the results of a stress benchmark. We showed that adding user features can significantly change the workload imposed on the server. In future work, we intend to improve the model to include the relationships between user actions and improve the workload generator to make the load closer to the real. Finally, we hope we have succeeded in drawing the academic community's attention to the importance of workload modeling for mapping and GIS applications in general.

References

Cibulka, D. (2013). Performance testing of web map services tn three dimensions–x, y, scale. *Slovak Journal of Civil Engineering*.

[4]The ACF is presented with confidence bounds at a significance level of 0.05, shown as a straight line parallel to the x-axis.

Feitelson, D. G. (2014). Workload modeling for computer systems performance evaluation. http://www.cs.huji.ac.il/~feit/wlmod/.

García, R., de Castro, J. P., Verdú, E., Verdú, M. J., and Regueras, L. M. (2012). Web Map Tile Services for Spatial Data Infrastructures: Management and Optimization. *Cartography - A Tool for Spatial Analysis*.

Google Maps API (2014). Showing pixel and tile coordinates. https://developers.google.com/maps/documentation/javascript/examples/map-coordinates. [Online; accessed 16-July-2014].

Kang, Y.-K., Kim, K.-C., and Kim, Y.-S. (2001). Probability-Based Tile Pre-fetching and Cache Replacement Algorithms for Web Geographical Information Systems. In *Proceedings of the 5th East European Conference on Advances in Databases and Information Systems*.

Kefaloukos, P. K., Vaz Salles, M., and Zachariasen, M. (2012). TileHeat: A framework for tile selection. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*.

MAPLARGE (2014). Map server performance. http://maplarge.com/mapserverperformance. [Online; accessed 16-July-2014].

OSGeo Wiki (2011). Benchmarking 2011. http://wiki.osgeo.org/wiki/Benchmarking_2011. [Último acesso: 31-Julho-2014].

Quinn, S. and Gahegan, M. (2010). A predictive model for frequently viewed tiles in a web map. *Transactions in GIS*.

Ray, S., Simion, B., and Brown, A. D. (2011). Jackpine: A benchmark to evaluate spatial database performance. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*.

Romoser, B., Fares, R., Janovics, P., Ruan, X., Qin, X., and Zong, Z. (2012). Global workload characterization of a large scale satellite image distribution system. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*.

Simion, B., Ray, S., and Brown, A. D. (2012). Surveying the Landscape: An In-depth Analysis of Spatial Database Workloads. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*.

StatCounter Global Stats (2014). Top 10 Desktop Screen Resolutions on July 2014 StatCounter Global Stats. http://gs.statcounter.com/#desktop-resolution-ww-monthly-201407-201407-bar. [Online; accessed 16-July-2014].

Zhang, Q., Cherkasova, L., Mathews, G., Greene, W., and Smirni, E. (2007). R-Capriccio: A Capacity Planning and Anomaly Detection Tool for Enterprise Services with Live Workloads. In *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*.