GeoStream - Exploiting User-Generated Geospatial Content Streams

FP7-SME-2012-315631

Web Computing Evaluation Report D3.2

Deliverable lead contractor: TALENT

Main contributors:

George Lamprianidis glampr@imis.athena-innovation.gr
Dimitris Skoutas dskoutas@imis.athena-innovation.gr
Manolis Koutlis koutlis@talent.gr

Due data: 30.09.2014

Abstract

This deliverable presents an evaluation report on the web computing framework that has been designed and implemented in the project, and which was reported in Deliverable 3.1. We describe the sets of experiments that have been conducted to evaluate the performance of the developed approach and we report the results of the evaluation.

Research Center "ATHENA," Greece
FU BERLIN, Germany
Fraunhofer, Germany
Michael Mueller Verlag, Germany
TALENT, Greece
WIGEOGIS, Austria

Table of Contents

List of Tables

Table of Figures

1 Introduction

The constantly increasing participation of users in social networking sites and in efforts and initiatives for open and collaborative data publishing has led to an explosion of crowdsourced geospatial data on the Web. This new wealth of sources and content opens up new opportunities for improving, enriching and enhancing services in the geospatial domain, such as location-based services, trip planning, etc. However, in its raw form, this content exists in very disparate sources and heterogeneous formats, often being incomplete and/or inaccurate. Thus, several challenges need to be tackled to allow for such data to be harvested, processed, analysed, and used in applications.

In GeoStream, we develop methods and tools to address these challenges, focusing in particular on (a) harvesting, integrating and mining user-generated geospatial content from the Web, (b) developing a Web and mobile computing framework for managing such content and supporting applications, (c) designing and implementing rich authoring tools to further facilitate and encourage users in providing such content, in ways that make it also easier to process it.

This deliverable is complementary to the deliverable D3.1. In D3.1, we have presented a novel framework that utilizes browser-based computing for the compilation of user-generated geospatial content. More specifically, we have designed and implemented an application of the map-reduce computation concept [2], in conjunction with browser-based computing, to accomplish mining and integration of user-generated content by means of Web-site visitors that contribute their "expertise" and computing power (browser) to perform this task.

This report presents an experimental evaluation of this framework. More specifically, we have conducted a series of experiments to measure the performance of our approach in different scenarios, covering both the stage of collecting crowdsourced geospatial data from Web sources as well as aggregating the data via clustering to compute regions of interest. The experiments also cover different sources and areas, in order to verify that the observations hold for different data distributions and characteristics.

In the following, we first present our experimental setup in Section 2 and then we report the results in Section 3. Section 4 concludes the report.

2 Experimental Setting

In our experiments we test how our architecture scales up and better handles the work load, when the number of available browser workers increases.

For this purpose, we executed two series of experiments: one to test and evaluate the data collection process and another one to test and evaluate the process of clustering Points of Interest (POIs). In both series, the experiments were executed using three different machines as clients:

- 1 physical machine, running Mac OS X 10.9.5, with a 2.7GHz Intel Core i5 CPU (4 cores) and 8GB of RAM, having up to two browsers open (Google Chrome version 37 and Safari version 7).
- 2 virtual machines, running Ubuntu Desktop 14.04, with a 2GHz QEMU Virtual CPU with 2 cores and 4GB of RAM each, having an instance of Firefox 29 open each.

The server is a virtual machine, running Ubuntu Desktop 12.04, with a 2GHz QEMU Virtual CPU with 2 cores and 8GB of RAM. All machines are connected to an at least 100Mbps network, and communicate over the public internet.

The order of use of the available browsers is shown in the figure below:

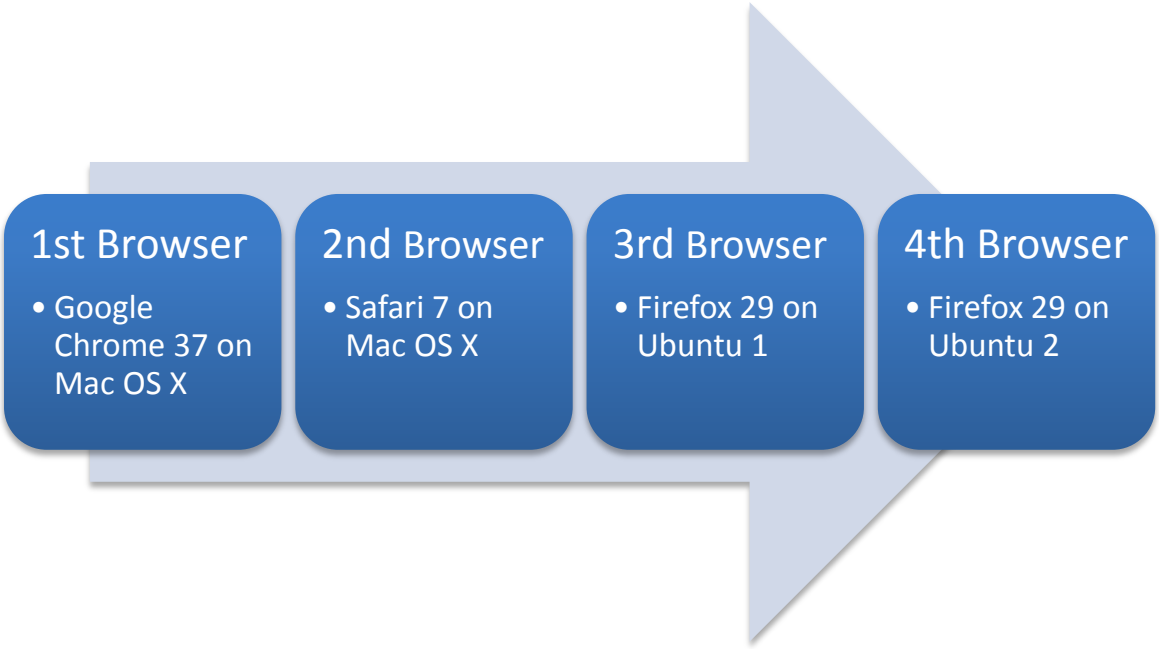

Figure 1: Order of browsers used for the experiments.

For the experiment regarding data collection, we have applied the process repeatedly to collect photos from both Flickr and Panoramio, and we have measured the time required to complete the task in each scenario. More specifically, we measured the time required to progressively download 5000, 10000, 15000 and 20000 photos for each source, using a number of browsers ranging from 1 to 4. This experiment was executed twice, once for the city of Los Angeles and once for the city of Paris.

For the clustering experiment, we have applied the process to cluster two POI categories, namely "Travel & Transport" and "Food", for two different areas: London and Athens. The clustering process takes a priori calculated cells that have been produced by splitting the given area into quadrants recursively, based

on the maximum number of POIs included in each cell, referred to as the cell size. This experiment was conduct 3 times for each setup, using cells that contained at most 500, 2000 or 5000 POIs at a time. The parameters of the DBSCAN algorithm [1] are always the same, having the following values:

$$\text{minPoints}=20 \text{ and } \varepsilon=0.001$$

where minPoints specifies the minimum number of points that should exist in a point's neighbourhood in order for it to be characterized as "dense" and ε specifies the radius of a point's neighborhood.

Each experiment was repeated 4 times, starting with one browser and at each subsequent repetition adding a new browser instance to work in parallel with the previous ones.

The results are reported in the next section.

3 Evaluation Results

3.1 Using browsers for data collection

In the first set of experiments, we have applied the browser-based computation approach to collect geocoded photos from Flickr and Panoramio for the areas of Los Angeles and Paris. We have conducted the experiment for two different cities in order to test if there are any significant variations in the observations made; moreover, we have selected these specific ones since they are areas that have a large number of photos to facilitate the experiments. The results are shown in the figures below.

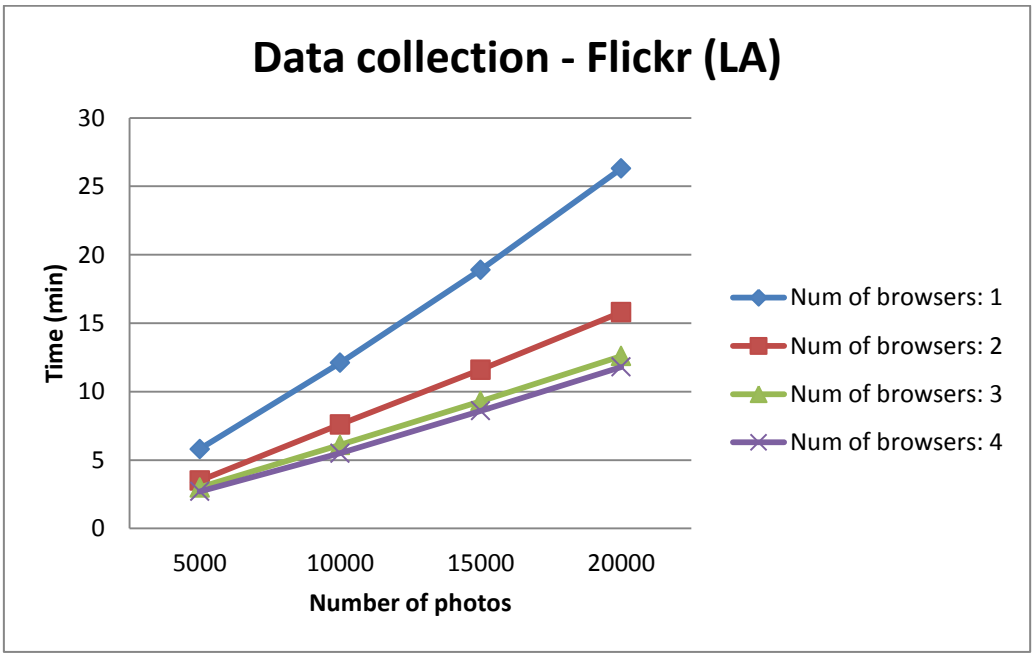

Figure 2: Elapsed time for collecting Flickr photos for Los Angeles.

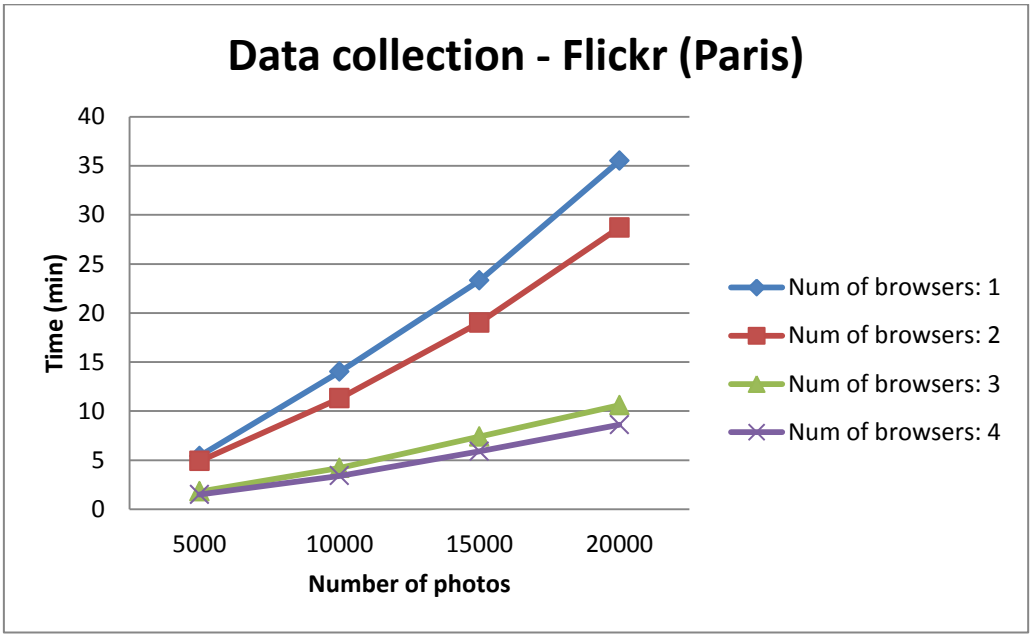

Figure 3: Elapsed time for collecting Flickr photos for Paris.

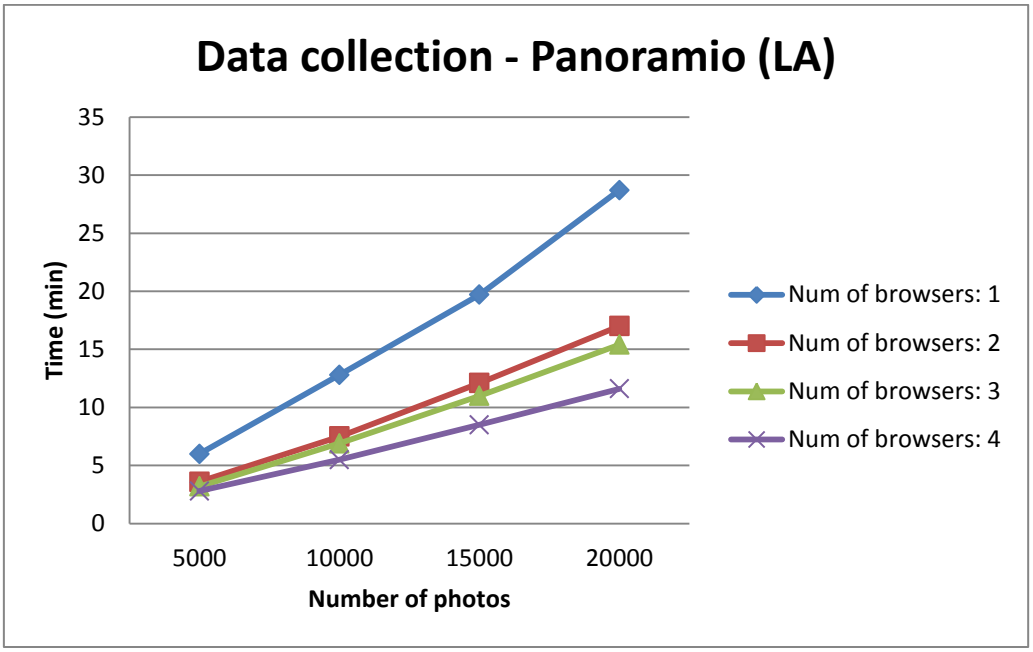

Figure 4: Elapsed time for collecting Panoramio photos for Los Angeles.

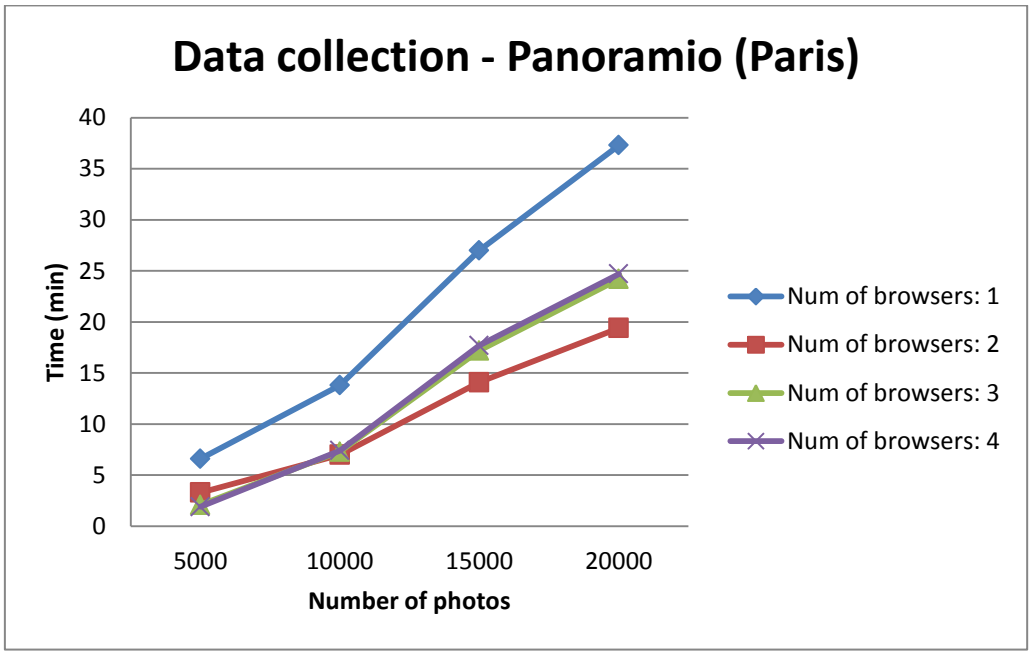

Figure 5: Elapsed time for collecting Panoramio photos for Paris.

3.2 Using browsers for data clustering

In the second set of experiments, we have applied the browser-based computation approach to find POI clusters in the areas of London and Athens for the categories "Travel & Transport" and "Food". As previously, we have conducted the experiments for two different areas and two different categories to examine whether similar conclusions hold.

Table 1 summarizes the total number of POIs that are given as input in each case.

Table 1: Number of POIs used for each experiment.

	Travel&Transport	Food
London	45,666	80,791
Athens	15,869	19,176

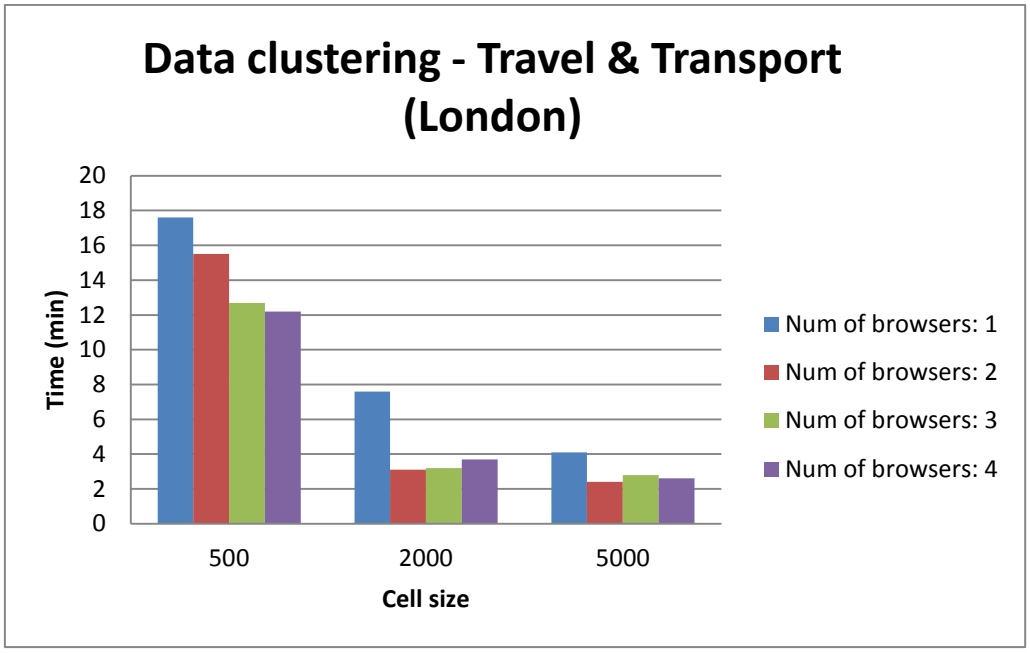

Figure 6: Execution time for clustering POIs in London for category "Travel&Transport".

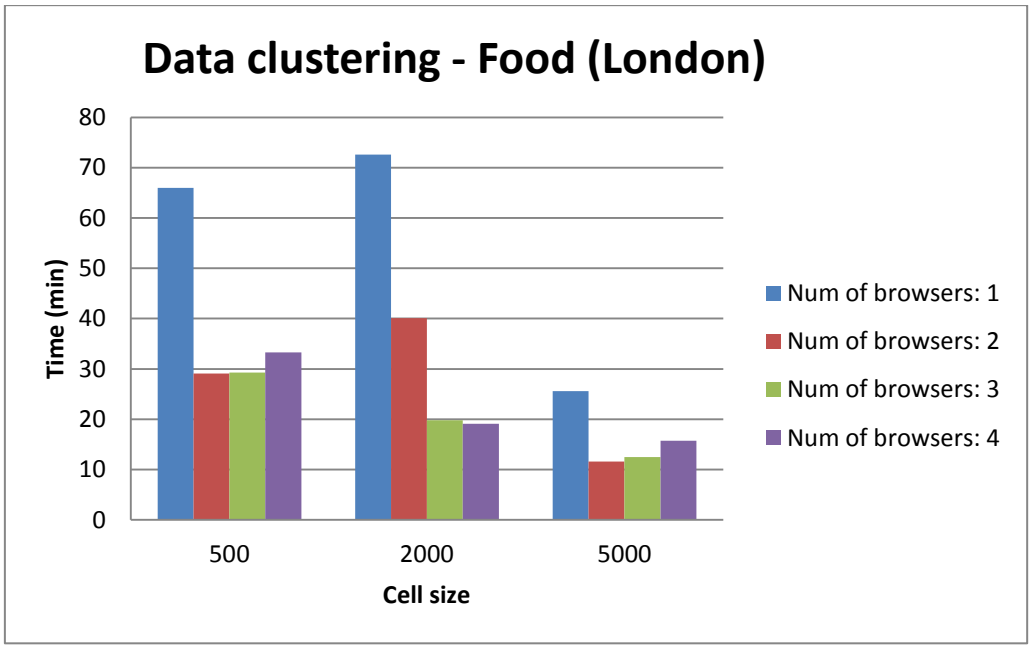

Figure 7: Execution time for clustering POIs in London for category "Food".

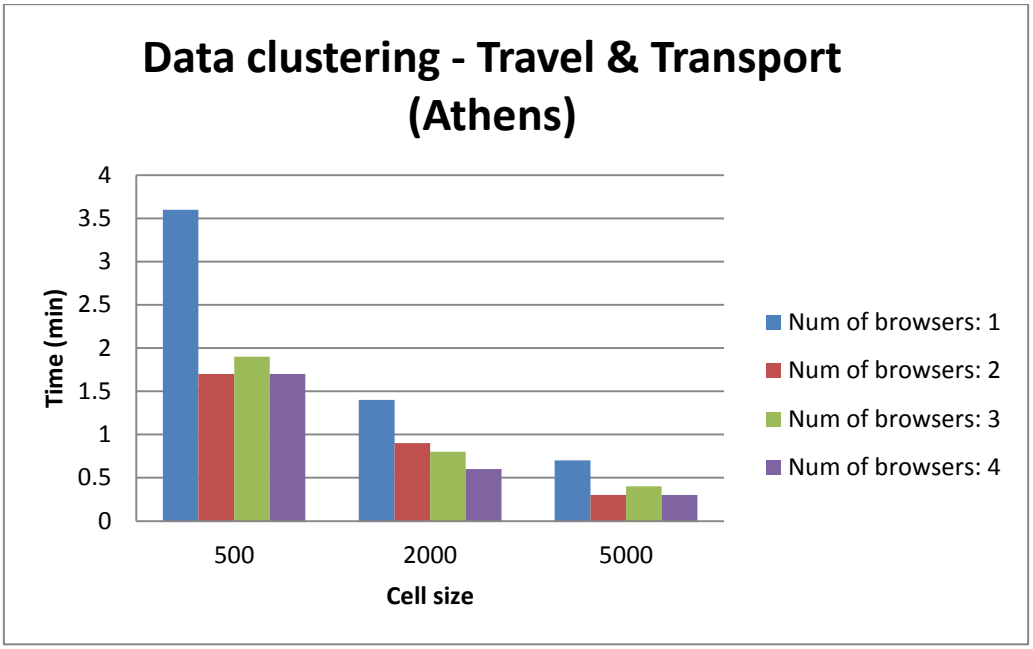

Figure 8: Execution time for clustering POIs in Athens for category "Travel&Transport".

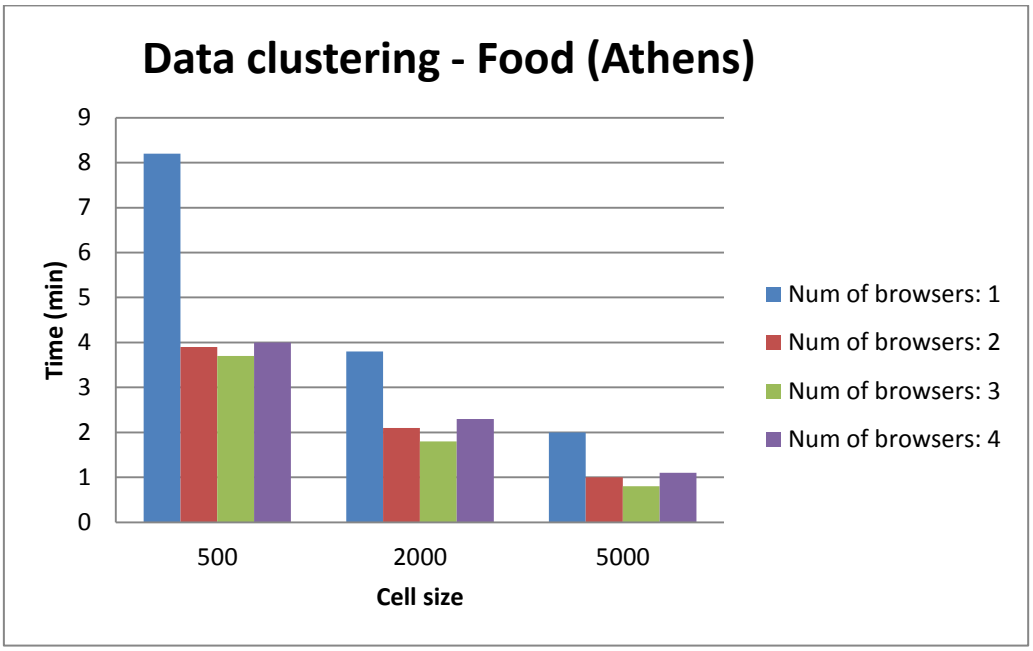

Figure 9: Execution time for clustering POIs in Athens for category "Food".

3.3 Observations

As can been seen from the results of the data collection process (Figures 2-5), the time required to complete the collection of up to 20,000 photos in the various scenarios using only one browser was around half an hour. This time was significantly reduced when more than one browsers where available, as a result of the fact that the task was carried out in parallel, assigning different cells to different browsers, which then retrieved data simultaneously.

For example, in the experiment of collecting photos from Flickr for the area of Los Angeles, when a second browser was used, the total time was reduced from 26 minutes to 15. This observation was also valid for other experiments; e.g. for the case of Panoramio and the area of Paris, the time needed when a second browser was used was reduced from 37 minutes to 19. Furthermore, additional reduction can be achieved by introducing a third or fourth browser. However, the reduction ratio drops. This can be attributed to two facts, as explained in the following.

First, it could be that the cells assigned to each browser for processing are not equally dense. Consider, for instance, two cells C_1 and C_2, with C_1 containing more photos than C_2, but both of them being below the split threshold. Assume also that C_1 is assigned to a browser B_1 and C_2 is assigned to a browser B_2. Naturally, the latter will finish earlier, but the overall execution time for the task is measured as the time needed for both to complete, hence it is dominated by the time needed by B_1.

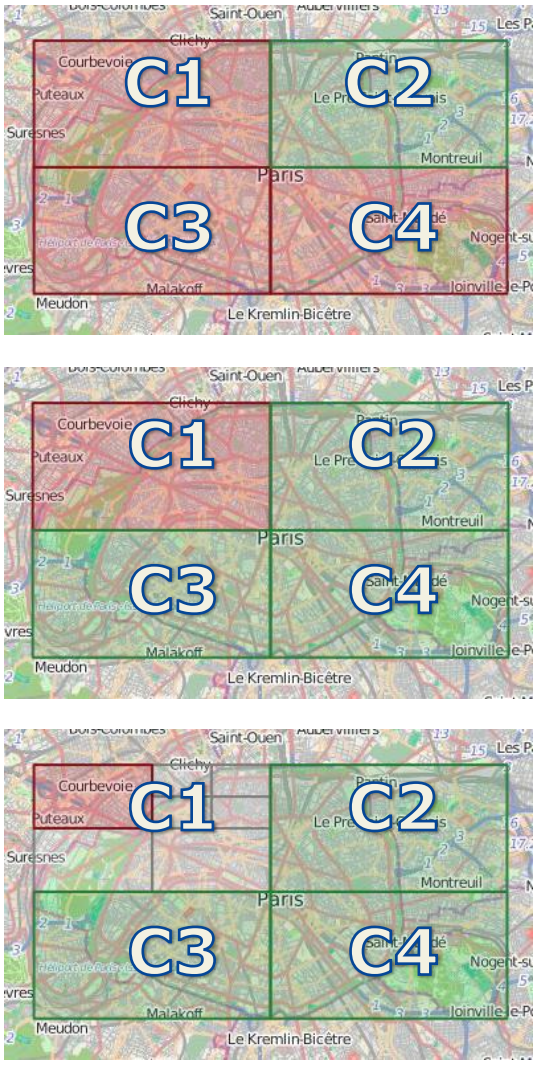

Figure 10. Collection progress.
(a) Originally 3 or 4 browsers are working.
(b) While B_1 is still working on C_1, the other browsers finish their respective tasks and terminate.
(c) Then the rest of the tasks inside C1 are assigned only to B_1.

Furthermore, the availability of tasks during the collection process is not known beforehand, but is rather determined as the collection progresses, since the amount of data each cell gathers cannot be known a priori. Because of this, the following scenario, which was observed in Panoramio photo collection for Paris (Figure 10) can occur: for example Browser 1 may finish its respective tasks and not finding any remaining jobs eventually terminate. However new tasks can be created by Browser 2, only after B_1 has terminated, hence all new jobs are assigned only to B_2 for the rest of the collection process. So the execution time does not decrease as expected, since some browsers may prematurely terminate. This affects our experiments because the scenarios we tested initiates all the browsers workers only at the beginning of the process, but may not affect real world usage, since users may visit the application at any time.

Note however that in a setting where data collection over a large number of areas is required, there would be also much more room (and need) for such parallelism. In addition, having the browsers perform the task completely alleviates this workload from the server, which is the first and foremost motivation for this approach.

Similar observations can be made for the operation of data clustering. Again, splitting the task to more than one browser makes it possible to exploit this

parallelism to reduce computation time. This is especially apparent for example in Figure 7. However, as before, increasing the number of browsers working together may not contribute to further decreasing the computation time. In addition to the first explanation provided above, there is one more factor here. In contrast to the operation of data collection, where the data retrieved by each client is simply collected by the server without any further processing, in the case of clustering the server needs to merge clusters computed by clients working on neighbouring cells into potentially larger clusters. The cost of this extra step is not trivial, and can be a potential bottleneck. Nevertheless, as mentioned above, when several different areas need to be covered, it is possible to assign browsers to different areas where they can work independently, rather than assigning many browsers to the same area.

Overall, the browser-based computation approach that has been developed achieves two goals, as shown also by the experiments:

- it lifts the burden from the server in performing otherwise long-running tasks, such as data collection or clustering over multiple, potentially large areas;
- it makes it possible to split an area into smaller cells and assign them for parallel processing by different browsers, thus further reducing the computation time.

4 Conclusions

In this deliverable, we have presented the results of an experimental evaluation that was conducted to test and validate the performance of the browser-based computation approach that was developed and described in Deliverable D3.1. The conducted experiments covered both operations of data collection and data clustering, as well as different sources and areas. The aim of the experiments was to measure the execution time for completing these tasks by assigning them to one or more browsers instead of performing the computation on the server. The experimental results validated the approach, showing that the computation was successfully completed by the browsers, noticing also a reduction of the computation time when more than one browser was used to execute the task in parallel.

References

[1] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, pp. 226-231).

[2] Jeffrey Dean, Sanjay Ghemawat: MapReduce: simplified data processing on large clusters. Commun. ACM 51(1): 107-113 (2008)