Grant agreement no. 607379

SPA.2013.2.1-01 - Analysis of Mars Multi-Resolution Images using Auto-Coregistration, Data Mining and Crowd Source Techniques

- Collaborative project -

D5.3
Stress-test of web-GIS

WP5 - iMars web-GIS Software

Due date of deliverable:	month 35 – November 2016	
Actual submission date:	14/12/16*	*(*) EC approval pending*
Start date of project:	January 1st 2014	Duration: 36 months

Lead beneficiary for this deliverable: Freie Universitaet Berlin (FUB)

Last editor: Sebastian Walter (FUB)

Contributors: UCL, DLR, EPFL

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

History table

Version	Date	Released by	Comments
1.0	4/11/2016	SW	Initial Document
1.1	24/11/2016	SW	Submitted to Coordinator
1.2	01/12/2016	JRK	Review
1.3	02/12/2016	JPM	Review
2.0	13/12/2016	SW	Final revisions included, submitted to Coordinator

Executive Summary

This document describes the activities of testing the performance of the i-Mars web-GIS client developed at FUB. It describes in detail both the technical details of the used performance collection software as well as the methods for putting load on the server. The server load was tested through concurrent access of the server from ten simultaneous users performing real-time use of the web application. After the load testing, it became apparent that the CPU speed and the number of the CPUs/cores are the most significant limiting factors with terms of performance. Even under heavy load conditions, the system appeared as responsive and stable. From these results, it is expected that the software setup ported to the target machines at MSSL will be performant for at least ten concurrent users.

Table of contents

Key word list

web-GIS, MapServer, load testing, benchmark, WMS, WFS, Mapcache, Apache

Definitions and acronyms

Acronyms	Definitions
GIS	Geographic Information System
web-GIS	A software service providing access to GIS functionalities over the web using standard protocols and browsers.
WMS	Web Map Service
WMS-T	WMS with time support
WFS	Web Feature Service
RAM	Random Access Memory
DDR	Double Data Rate
GB	Gigabyte
SAS	Serial Attached SCSI
CPU	Central Processing Unit
SQL	Structured Query Language
TIF	Tagged Image File Format
Apache	Apache HTTP Server, the most popular web server on the internet
ACRO	Automatically Co-Registered and Orthorectified images
Mapserver	Open-source platform for publishing spatial data and interactive mapping
MapCache	Stand-alone server that implements tile caching to speed up access to WMS layers (part of the Mapserver project).

1 Introduction

The performance and the reliability of the physical server are fundamental aspects of a web-based GIS platform. It determines the speed and efficiency of the user interaction with the system. If the application is experienced as being slow and interrupted, the user will lose interest in the application and leave the web page. As the performance of a single server hardware is limited by available technology and its price, the data structure, data storage and retrieval has to be highly optimised for specific applications. This document reports on the stress testing activities which have taken place at FU Berlin to optimise the web-GIS performance. It is focused on the access of the data structures on the server side (raster data, vector data, and attributes), not the possible software optimisations of the (JavaScript-based) client software, which is highly dependent on the browser employed by a user and general performance of the client computer.

1.1 Hardware setup

The load testing was carried out on the developer hardware housed in the server room of the Institute of Geological Sciences at Freie Universität Berlin. It consists of a Dell PowerEdge M520 blade unit server enclosed in an M1000 fully-redundant blade chassis. The blade server contains two Intel Xeon E5-2403v2 CPUs running at 1.8GHz each (8 total cores, no HyperThreading, no turbo mode). The CPUs were the lowest specification available processors for this system configuration - as the main target for the server was the development and it was not intended to host the final application which is planned to be performed at MSSL. The system's total memory is 48GB of dual-rank DDR-3 RAM running at 1333 MHz The system has a built-in disk array of two small-but-fast (300GB) SAS-based hard drives where the operating system is located. Further on, it is connected (via 8 GBit/s) to the Fibre-Channel storage network of the planetary sciences group, where higher volumes of mass memory are dynamically available (currently there are 8TB assigned). The throughput of the storage should reach well above 400MB/sec but, as it is based on SATA disks, it is expected that it is not well suited for high access times operations such as database access. The network connection is a single gigabit Ethernet adapter connected to a gigabit switch with direct connection to the German Science backbone.

1.2 Software setup

To be able to use different versions of operating systems and setups, the server system is virtualised. CentOS version 6.7 serves as the KVM-based hypervisor and a QEMU-based virtual machine under CentOS 7.2 hosts the actual system. The components of the system include Apache httpd 2.4.6 (stock CentOS), MapServer 7.0.0 (compilation), GDAL 2.1 (custom compilation) and Mapcache 1.5 (compilation). The data is stored in tiled (big-) TIF format images (raster data) and in stock-CentOS PostgreSQL version 9.2.15 databases (vector data).

The application processes vector data from the PostgreSQL database (footprint catalogues and additional information as e.g. nomenclature), raster data from TIFs stored on disks (base-maps, DTMs, images) and serves these as raster (WMS and WMS-T) and vector data (WFS) via Apache and the MapServer CGI to the client. The data is mainly served in tiles so that the client machine caches the files in the browser and the user perceives improved performance while panning and zooming.
All georeferenced files are stored in the same projection they are delivered in (to minimise "on-the-fly" projection). Where possible, the files are pre-rendered into tiles using Mapcache to improve the performance of the distribution of the requested data. Besides static layers like base-maps and coverages, the application serves single images (ACRO'd products and DTMs/ORIs) in a dynamic way, where a Python/MapScript-based CGI script creates a WMS layer for the single image on the fly (for a comparison between the data flows of a regular WMS process, the cached process using Mapcache and the dynamically loaded images using MapScript, see Figure 1). For these products it is by design not possible to create tiles in advance. Therefore some amount of lag and performance slowdown has to be expected when significant numbers of dynamic layers are shown simultaneously.

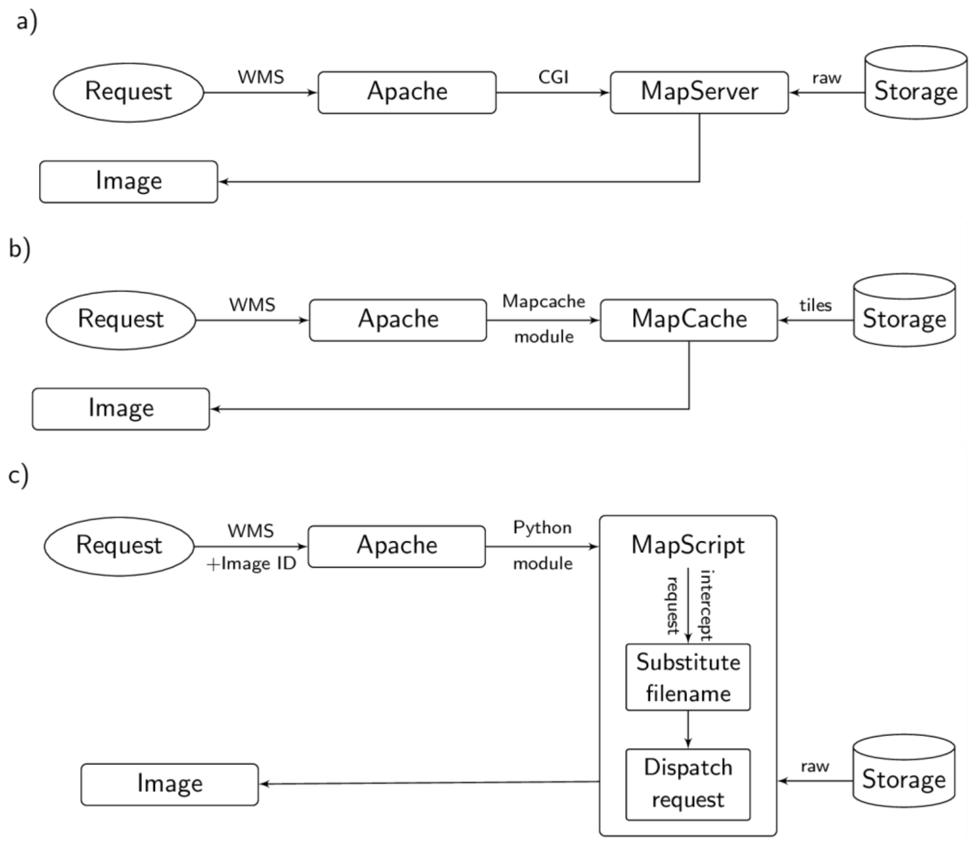
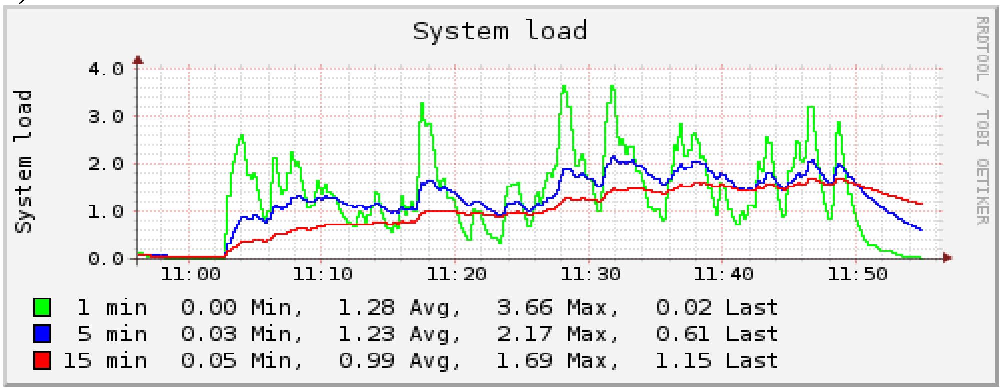


Figure 1: Data flow of the different WMS request setups. a) The regular WMS request is submitted to apache, which calls the MapServer binary via the common gateway interface (CGI). b) Static layers are cached on the server where possible. Instead of sending the request to MapServer, it is sent to the MapCache binary which loads tiles, pre-defined in several zoom levels, from its internal cache. c) In the case of dynamic images, an additional request parameter is added, the image ID. The request is called against a MapScript python instance, where the filename of the image is substituted with the image ID.

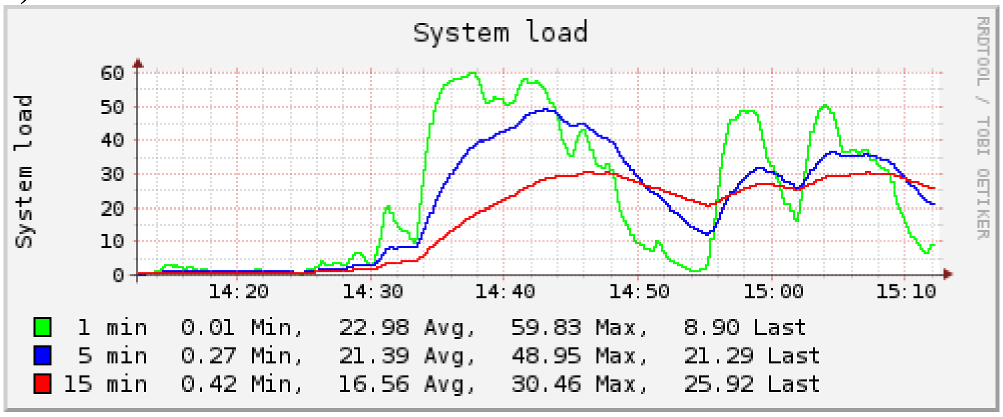
1.3 Objectives of the Stress Test

It is the scope of the current document to analyse the performance and general usability of the system during both minimally as well as heavily loaded conditions. The stress testing activities should demonstrate whether the proposed final production server setup located at UCL will be capable of serving multiple concurrent connections and where the architecture can be optimised before the transfer and implementation of the software and data.
We have installed a performance metering software collection system to gather statistics on the server (collectd, https://collectd.org) and have then performed two stages of manual load placement on the server. In the first round, the ten volunteer participating testers used the system exclusively by themselves and therefore under optimal conditions. They provided qualitative feedback about the usability and performance of the system and recorded measurements on absolute loading times for certain analysis tasks. In a second round, all ten testers accessed the server at the same time, to place some significant load on the server. A plot of the comparison of the system load of both setups is shown in Figure 2.
a)



```
🟩  1 min   0.00 Min,   1.28 Avg,   3.66 Max,   0.02 Last
🟦  5 min   0.03 Min,   1.23 Avg,   2.17 Max,   0.61 Last
🟥 15 min   0.05 Min,   0.99 Avg,   1.69 Max,   1.15 Last
```

b)



```
🟩  1 min   0.01 Min,   22.98 Avg,   59.83 Max,   8.90 Last
🟦  5 min   0.27 Min,   21.39 Avg,   48.95 Max,   21.29 Last
🟥 15 min   0.42 Min,   16.56 Avg,   30.46 Max,   25.92 Last
```

Figure 2: comparison of the two stages of the load testing. a) Single user access shows a maximum system load of 3 to 4, while b) concurrent multi user access shows a load higher by a factor of 20.

Both campaigns were recorded by collectd and can now be compared. Using this approach we expect to find the main drivers for the slowdown of the server to improve both stability and performance.

2 Methods

2.1 System Statistics Collection

The core measurement software used in this stress test is a system statistics collection daemon called "*collectd"* (http://collectd.org). *It* gathers metrics from various sources, e.g. the operating system, applications, log files and external devices, and stores this information in databases. These statistics can be used to find performance bottlenecks and to predict future system load. To secure the performance and the portability, it is written in C, and includes the functions to handle thousands of metrics based on a plugin architecture together with further optimisations. In addition, it also delivers a small collection of scripts generating concise web graphs based on the collectd data.

For the stress test, we used the following plugins of collectd:

1. Apache. The Apache plugin queries the status module of the Apache web server and submits the number of bytes transferred, the number of requests received, and the number of processes in the various states of the scoreboard (https://collectd.org/wiki/index.php/Apache).
2. CPU. The CPU plugin collects the amount of time spent by the CPU in various states, most notably executing user code, executing system code, waiting for IO-operations and being idle. It does not collect percentages, it collects "jiffies", the units of scheduling (http://collectd.org/wiki/index.php/Cpu).
3. Disk. The Disk plugin collects performance statistics on the hard-disks and, where supported, partitions. It reports disk traffic, disk operations, and the number of operations that could be merged into other, already queued operations, i.e. one physical disk access served two or more logical operations (the higher that number, the better); the average time an I/O-operation took to complete; the io_time - time spent doing I/Os (ms), which represents the device load percentage (value of 1 sec time spent matches 100% of load); the weighted_io_time, which measures both I/O completion time and the backlog that may be accumulating; the pending_operations shows the queue size of pending I/O operations (https://collectd.org/wiki/index.php/Disk).
4. Interface. The Interface plugin collects information about the traffic (octets per second), packets per second and errors of interfaces (number of errors during one second) (https://collectd.org/wiki/index.php/Interface).
5. Load. The Load plugin collects the system load. These numbers give a rough overview over the utilization of a machine, though their meaning is mostly overrated. The system load is defined as the number of runnable tasks in the run-queue and is provided by many operating systems as a one, five or fifteen minute average (https://collectd.org/wiki/index.php/Load).
6. Memory. The Memory plugin collects physical memory utilization. The values are reported by their use by the operating system. Under Linux, the categories are Used, Buffered, Cached and Free (https://collectd.org/wiki/index.php/Memory).
7. Processes. The Processes plugin collects the number of processes, grouped by their state (e. g. running, sleeping, zombies, etc.).
8. (https://collectd.org/wiki/index.php/Processes).

9. RRD. The RRDtool plugin writes values to RRD-files using librrd (https://collectd.org/wiki/index.php/RRD).

The collectd software has been set up on the development system in a way that the metrics provided by the different plugins are accessible via contributed CGI scripts (cf. Figure 2).

The measurement graphs appear as curved lines over a time axis. In the case of limiting factors such as limited amount of processes, limited bandwidth or limited CPUs, the lines become flat at an upper limit.

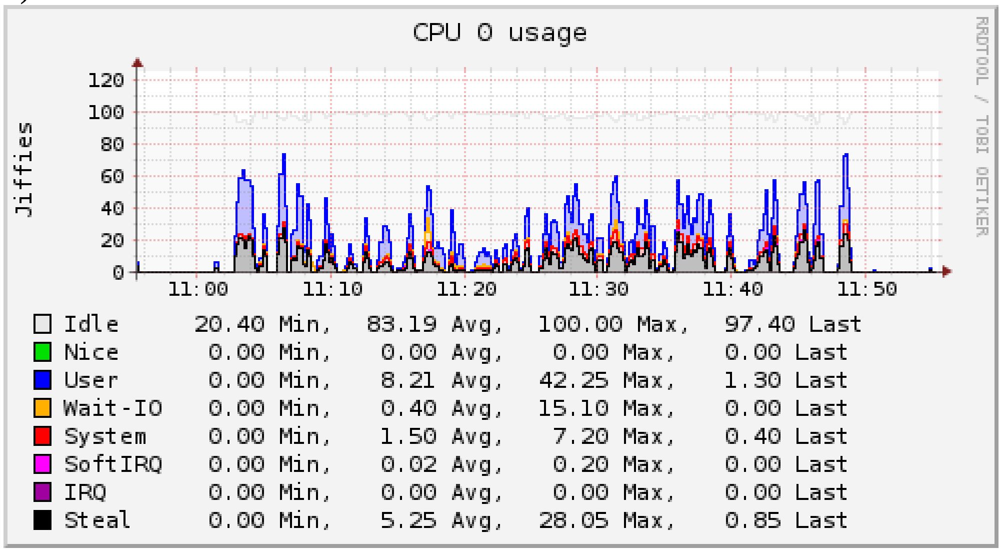
2.2 Single User Load Placement

During the first phase of the Stress Test, the objective was to find the relevant collectd metrics which reflect the current degree of capacity utilisation and system load. It was expected (and to be approved) that the server in its current dimension of hardware layout would be able to serve one exclusive user, without any additional load. To allow all participants of the stress test exclusive access to the server, a timetable has been created, where every participant booked and used the server on its own without concurrent access by others. During every single test, the collectd metrics were observed and then saved for later assessment after the test.

In preparation for the test, some instructions were distributed to the users. To rule out network bandwidth limitations between the tester and the server, a connection speed test (downloaded from http://speedtest.net) was executed before each test. After the speed test, the stress test on the web-GIS was performed. It consisted of several stages:
(1) The user should just move around in the map, zoom and pan, without any additional layers activated. These operations should be mainly served by the Mapcache daemon on the server, as all the initially activated layers were cached on the server. In this stage, the main impact was expected on the file system, not the CPU.
(2)Then the user should zoom into the highest possible images, the HiRISE DTMs, which are not applicable to caching. Here, a larger impact on the disk measurements was expected.
(3) After that, the user should activate vector layers with footprints loaded from databases.
(4) Additionally, filtering of the databases employing the time filter functionality so called WMS-T should make sure that the database tables were not cached and the real database speed would be tested.
(5) Directly after, the query layers should have been loaded and then example points with multiple coverage of ACRO images were to be queried. These selections were then shown as dynamic images. It is expected that this task places the most significant load on the server, as the large raster images – due to their dynamic nature – can't be cached and have to be loaded as individual WMS layers (to be able to flicker through them after loading). This could lead to concurrent loading of tens to hundreds of WMS raster layers. It was expected that for this last task, the server should reach its performance limit already during exclusive single user access (disk and CPU-intensive).

The users were asked to write down their experiences, measurements, and general performance impressions during the test. After each of the sessions of single user access, the collectd graphs have been saved for inspection. This way the key metrics of the server performance (with regard to the target web-GIS application) could be determined.

In total, ten users performed the test during different time slots. In terms of results returned by the collectd system, all the tests showed similar outputs, which are summarised here. The Apache metrics showed traffic between 0.5 and 2.0 MB/s, around 7 to 12 simultaneous connections and between 10 and up to 35 requests per second. The CPU usage was between 20 and 70 jiffies with no sign of processor overload (cf. Fig. 3).



```
□ Idle      20.40 Min,  83.19 Avg,  100.00 Max,  97.40 Last
🟩 Nice       0.00 Min,   0.00 Avg,    0.00 Max,   0.00 Last
🟦 User       0.00 Min,   8.21 Avg,   42.25 Max,   1.30 Last
🟧 Wait-IO    0.00 Min,   0.40 Avg,   15.10 Max,   0.00 Last
🟥 System     0.00 Min,   1.50 Avg,    7.20 Max,   0.40 Last
🟪 SoftIRQ    0.00 Min,   0.02 Avg,    0.20 Max,   0.00 Last
🟪 IRQ        0.00 Min,   0.00 Avg,    0.00 Max,   0.00 Last
■ Steal      0.00 Min,   5.25 Avg,   28.05 Max,   0.85 Last
```

Figure 3: Typical CPU measurement during the single user test (one CPU out of 8 – the diagrams of the others look similar). For an explanation of the cpu states, see e.g. *top man page*[1].

In this particular test, the disk metric shows a sign of higher load in terms of a peak in the disk IO measurements at around 11:17 in Fig. 4, when the weighted_io_time reached values of approx. 2,500. Also the disk traffic measurements show a peak at that time with values of up to 60 MB/s, which should still be well below the possibilities of the disk system.

[1] http://man7.org/linux/man-pages/man1/top.1.html

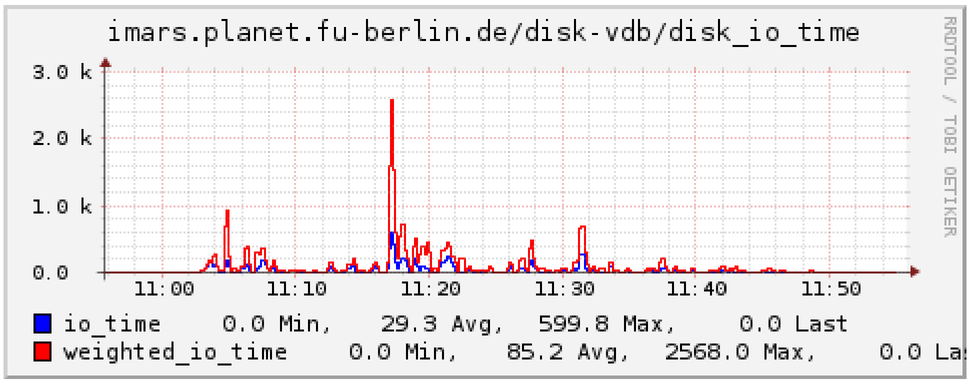


Figure 4: disk IO measurements during the single user test. The graph shows a clear peak at 11:17 which is a marker for higher load on the server.

The network traffic reached nearly 10 MBit/s, well below the capacity of the server. There were no significant memory consumption during the measurements which makes clear that the available memory is not relevant to the performance of the web-GIS. (cf. Fig. 5). Therefore, the available memory could be used for caching other data sources which receive a larger slowdown due to the disk architecture.

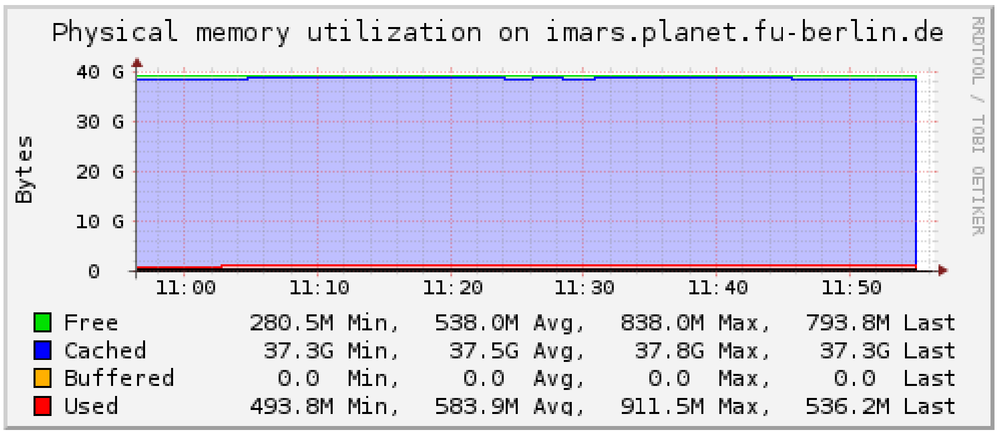


Figure 5: the measurement of used memory does not show any significant memory usage during the test.

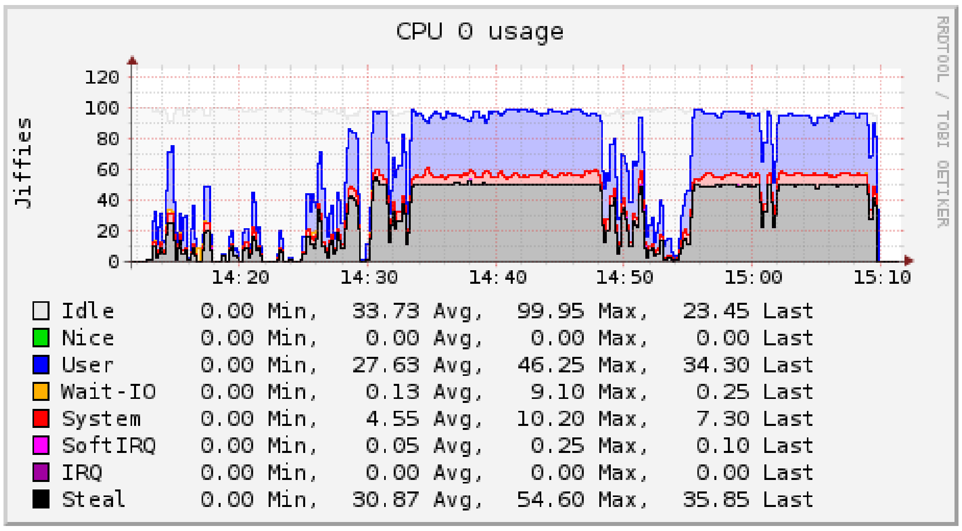
All participants reported that the web-GIS application was responsive during the whole test, with the single exception of the loading of the dynamic images. In many cases, it took not more than 20 seconds to load a specific map setup – a loading time which was experienced as adequate by the users. The more images that were loaded at once, the more delay was experienced. This behaviour of the server was expected though, and in fact turned out better than previously thought. The first stage of the test also served as an occasion for the testers to use the web-GIS application in an extensive way, get familiar with the system and its capabilities and limitations. Various feedbacks were provided to improve the user interface and the stability.

2.3 Multi User Load Placement

To simulate the expected concurrent access to the system, a second stage of the stress test was carried out. Ten users were asked to access the server simultaneously to bring a heavy load to the server and to try to reach the limits of the hardware. To synchronise the effort and to record exactly which actions lead to eventual slowdown of the application, a web conferencing system was used during this concurrent manual load placement. Because of the real time control with the help of the web conference tool, certain actions could be performed in a very coordinated fashion and the corresponding time could be recorded – which enabled us to track certain activities in the performance statistics. The meeting for the stress test started at around 2:00PM on 3 November. After some general discussions and introduction into the system and the stress test, three major blocks of different activity were conducted.

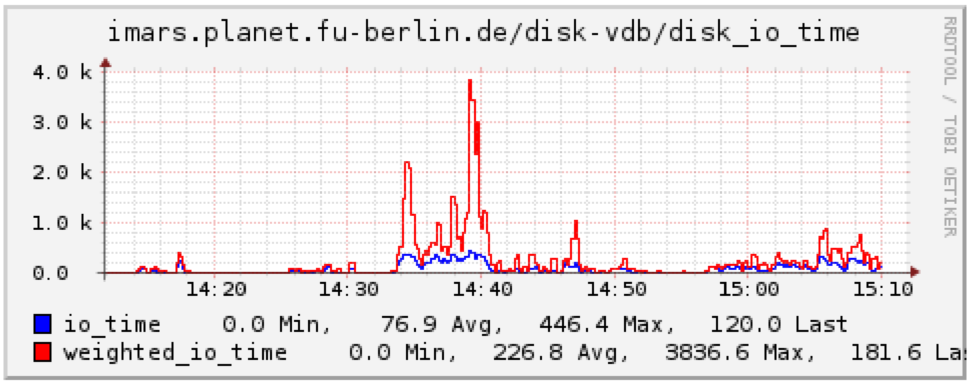
1) The first part test, Map Zoom in:
   A. Time line: Beginning at 2:35PM and end up at 3:15PM
   B. Action: all the users started zooming around the map, without enabling any additional layers. The task for all the ten users was then to zoom into a high-resolution HiRISE DTM (which was not cached). Also it was required during the zooming and panning, that the same area would not been covered several times, as the browser would then reload the tiles from its internal cache.
   C. The results: a significant slowdown of the server, a subset of one HiRISE scene in full resolution took up to 50 seconds to completely render. Despite the long loading time, the feedback from the users was that the system was still being regarded as responsive, as one could see the loading of the tiles and the system was not stuck.
2) The second part test, database layer activation
   A. Time line: Beginning at 2:46 PM
   B. Action: Activation of database layers, by all users concurrently.
   C. Results: It was experienced by some users that the system slowed down significantly. As the slowdown was experienced only by some users, while the loading of the tiles continued for others, it is assumed that the performance impact was induced by the limited number of cores and the weak processing power of the CPUs. Also, the database storage on the server is not optimised properly, as it is located on slow-but-large disk storage. It was already expected that at the time of server acquisition that the database speed would be a problem with such hardware configuration. However the cost for performance-leveraging high capacity SSD disks was too high to be considered as a replacement. With regards to the large amount of unused RAM memory, there are certainly additional methods for performance tuning of the PostgreSQL database, e.g. setting higher cache values (shared_buffers option). The final tuning for the database will be highly dependent on the target system at MSSL equipped with better processors with more cores and faster disk space.

In Figures 5 to 7, the relevant graphs from the statistics collect daemon are shown. It demonstrated that the processing power of the CPUs is the limiting variable on the server (Fig. 5 shows only one of eight graphs for the available CPU cores, but all look very similar to each other). During all active phases of the test, not only during the most demanding tasks, the CPUs reached their possible work limit. High disk access is visible during the loading of the high-resolution HiRISE DTMs at around 2:40PM. The other processes, including database access, do not seem to have large impact on the disk. Depending on the disk system at UCL, there is probably not much we can do about it to improve it. The caching of pre-rendered tiles at the best resolution level would most likely go beyond the scope of available disk memory at UCL as well.. The available RAM has not changed during the whole session; the maximum used value was 2.3 GB, which represents 5% or the total available memory.



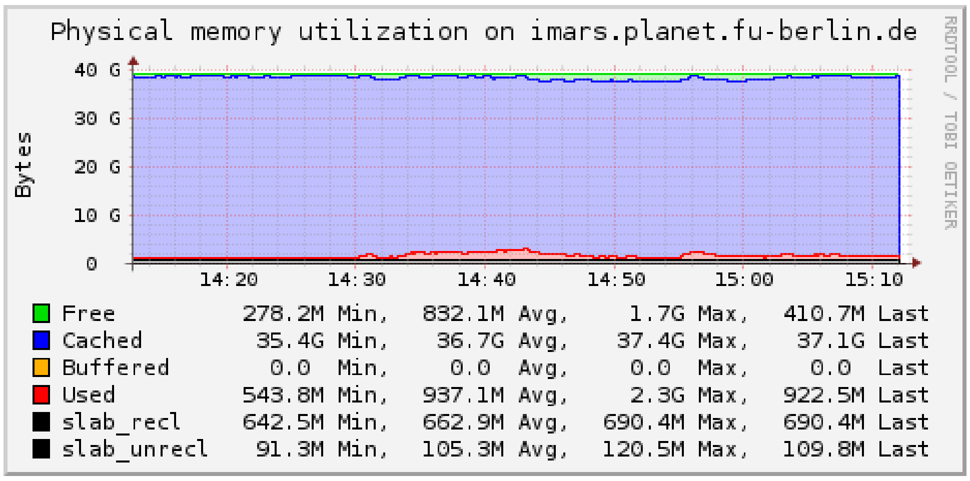
```
⬜ Idle       0.00 Min,  33.73 Avg,  99.95 Max,  23.45 Last
🟩 Nice       0.00 Min,   0.00 Avg,   0.00 Max,   0.00 Last
🟦 User       0.00 Min,  27.63 Avg,  46.25 Max,  34.30 Last
🟧 Wait-IO    0.00 Min,   0.13 Avg,   9.10 Max,   0.25 Last
🟥 System     0.00 Min,   4.55 Avg,  10.20 Max,   7.30 Last
🟪 SoftIRQ    0.00 Min,   0.05 Avg,   0.25 Max,   0.10 Last
🟪 IRQ        0.00 Min,   0.00 Avg,   0.00 Max,   0.00 Last
⬛ Steal      0.00 Min,  30.87 Avg,  54.60 Max,  35.85 Last
```

Figure 5: Graph of the CPU usage during the concurrent multi-user server access test. The CPUs are at their limits most of the time even during simple operations.



```
🟦 io_time        0.0 Min,    76.9 Avg,   446.4 Max,   120.0 Last
🟥 weighted_io_time    0.0 Min,   226.8 Avg,  3836.6 Max,   181.6 La
```

Figure 6: Graph of the disk IO time of the server during the concurrent multi-user server access test. The detailed view of non-cached high resolution layers had the most impact on the server performance.



```
🟩 Free          278.2M Min,  832.1M Avg,    1.7G Max,  410.7M Last
🟦 Cached         35.4G Min,   36.7G Avg,   37.4G Max,   37.1G Last
🟧 Buffered        0.0  Min,    0.0  Avg,    0.0  Max,    0.0  Last
🟥 Used          543.8M Min,  937.1M Avg,    2.3G Max,  922.5M Last
⬛ slab_recl     642.5M Min,  662.9M Avg,  690.4M Max,  690.4M Last
⬛ slab_unrecl    91.3M Min,  105.3M Avg,  120.5M Max,  109.8M Last
```

Figure 7: Graph of the physical memory utilisation during the concurrent multi-user server access test. During the test, only about 5% of the available 48GB RAM were in use.

3 Summary of activities and research findings

To summarise, the following key factors were found to be responsible for the performance of the web-GIS application:

- CPU: according to the measured statistics, the available processing power was the biggest limiting factor on the performance. The fact that all eight cores were loaded with an equal amount of load shows that the application can use all available processor cores efficiently.
- Disk: at some parts of the test, the disk access played a significant role (during the panning and zooming of non-cached images).

The physical RAM on the machine appeared to only play a minor role. On the other hand, the performance of database-related processes could not be measured with the available metrics.

4 Conclusions and future steps

Given the performed stress test results as presented here, we were able to successfully find the key factors for critical performance impacts on the server. It turned out during the test, that the processor speed and the amount of available cores per processor were fully loaded during regular multi-user access. Disk access became partly important at size-critical images which could not be pre-tiled beforehand. It turned out that RAM was not part of the limiting factors of the system.

The tested software will be transferred to the MSSL imaging cluster with better hardware with regard to CPU and storage, where better performance is expected. The recommendation here is upgraded CPUs with multiple cores. Storage design should aim for short access times which are achievable with equipment such as SSD drives or fast-spinning SAS drives. Large available disk space enables to pre-render tiles for a large amount of zoom levels for the static layers. For example, the tile cache of one half-quadrangle at zoom level 10 (20m/pixel) consumes 3 GB of disk space and it is quadrupled with every level. Fine-tuning of performance parameters of the database management system adapted to the target system should improve database access significantly.

Even on the limited developer hardware, the system was usable with ten concurrent users and highly demanding tasks. Therefore, it will operate stably for a limited number of science users though this depends on the number of concurrent users. Our experience in the past has shown that press releases with linked content to the server can lead to high access by many users. On the other hand, the application appears as an expert tool with many sophisticated functions which demand high performance. The target group of planetary science expert users are expected to take the time to view the "show-and-tell" YouTube videos and read help pages before or during the use of the software. It should be mentioned in the documentation that the user has to expect long delay times for certain functions. For example, the time of loading many dynamic images would last tens of seconds even on a very fast hardware, but the users are "rewarded" with the time-based flickering function afterwards. As long as the end users are well informed with proper documentation, we can expect that they would expect and accept delay times while working with the application.