ownCloud Enterprise Edition on IBM Elastic Storage Server

A performance and sizing study for large user number scenarios

A technical report

Udayasuryan Kodoly
IBM Systems - ISV Enablement Solution Architect

Dr. Oliver Oberst
IBM Systems - Industry Solution Architect

Chris Burnard
ownCloud - Senior Solutions Architect

March 2016

@IBMSystemsISVs

Table of contents

Abstract **1**
Introduction **1**
Scope 2
Intended audience 2
Building materials of lab proof of concept **2**
IBM Elastic Storage Server 2
ownCloud Enterprise Edition 3
ownCloud Enterprise and IBM storage infrastructure **4**
Proof of concept setup 5
Solution performance testing methods 7
Performance optimizations 10
Apache tuning 11
Large enterprise user account number systems 13
Proof of concept performance measurement 13
Sizing assumptions and considerations – the 100,000-user scenario 15
Enterpriseclass data availability feature of IBM Elastic Storage 17
Summary **18**
Appendix A: Test environment **19**
Appendix B: BeanShell PreProcessor and BeanShell PostProcessor scripts **20**
Appendix C: Lab proof of concept for a load of 1000 users **21**
Appendix D: Resources **24**
About the authors **25**
Trademarks and special notices **26**

Abstract

The on-premises file synchronization and sharing (file sync and share) solution build using ownCloud and IBM Elastic Storage Server (ESS) allows IT to regain control of sensitive data and gives users universal file access to all of their data seamlessly. The main goal of this technical report is to provide specific details on measuring the performance capabilities and sizing assumptions of a large user number of scenarios of the ownCloud application and database server and IBM Elastic Storage Server backend.

Introduction

Today's enterprise users want a simple, easy way to share files in their business environment that allows them to access their files seamlessly on all devices and collaborate with colleagues, customers, and business partners.

Enterprise IT, on the other hand, is concerned about the lack of control over corporate data, and the security and compliance risk it presents. Recent breaches of public consumer cloud services have proven those concerns to be justified, and the best course of action is to create a secure, easy-to-use alternative.

Together, ownCloud and IBM® are helping enterprises across industries overcome file sharing challenges, enabling agile, scalable and highly secure file sync and share with ownCloud Enterprise Edition and IBM storage solutions. ownCloud Enterprise Edition is powered by ownCloud’s open, modular architecture to deliver convenient end-user file access and ongoing IT data control. Backed by IBM storage technologies, such as IBM Spectrum Scale™ and IBM Elastic Storage™ Server, the solution can handle an organization’s most-demanding storage needs.

Successfully designing large-scale IT solutions require realistic system-load estimation, dependent on software usage patterns and user behavior, leading to a suitably sized system.

The purpose of this paper is to give an overview of the general ownCloud sizing considerations and the appropriate assumptions. In the lab proof of concept by ownCloud and IBM, measurements were conducted on high-end IBM Elastic Storage Server and owncloud workload staged on the x86-based compute infrastructure using high availability (HA) configuration with emphasis on large user number scenarios.

Specifically, the performance of ownCloud Enterprise Edition 8.2.1 was measured running on high-end x86 based nodes as database and application server infrastructure connected to IBM Elastic Storage Server (ESS-GL6) system as the storage back end. The lab proof of concept solution is configured with load balancing and maximum possible amount of authentication setup combinations. The proof of concept was built for measuring the performance capabilities and sizing assumptions of a large number of user scenarios of the ownCloud application and database server and the storage back end.

An important factor considered by the test team for specific system sizing is: *What is the number of intended users and their usage requirement with the system?* The sizing consideration is directly related to the size and number of files stored in the system, the number of application plug-ins enabled on the system, and the nature of users, devices, and bandwidth used to connect to the system. For example, higher bandwidth connections use less memory, but stress disk performance as more files are uploaded in a given time frame.

The important points considered by the test team are about users, devices, behaviors, and files. More specifically:

- How many concurrent users are to be active on the system each day
- How many devices will they be connecting to
- What type and number of devices (for example, mobile phone, tablet, or desktop notebook) will be used
- On an average, how many files are syncing and sharing per user

There are many other considerations, but these tend to be the most important.

Scope

This technical report:

- Discuss the approximate performance measurements and system sizing assumptions of the combined ownCloud Enterprise Edition 8.2.1 and IBM Elastic Storage Server file sync and share solution configurations.

This technical report does not:

- Discuss the installation and basic configuration of ownCloud Enterprise Edition 8.2.1.
- Discuss the installation and basic configuration of IBM Elastic Storage Server.
- Replace any already available document that is related to ownCloud, and IBM Elastic Storage Server system.

Intended audience

This technical report is intended for:

- Customers and prospects looking to implement a combined on-premises file sync and share solution using ownCloud Enterprise Edition 8.2.1 and IBM Elastic Storage Server.
- Users and management seeking information to implement a combined on-premises file sync and share solution using ownCloud Enterprise Edition 8.2.1 and IBM Elastic Storage Server.

Building materials of lab proof of concept

This section briefly describes the essential components used in the proof of concept.

IBM Elastic Storage Server

IBM Elastic Storage Server is a modern implementation of software-defined storage, combining IBM Spectrum Scale software with IBM POWER8® processor-based servers and storage enclosures. IBM Spectrum Scale, is a parallel file system that is at the heart of IBM Elastic Storage Servers. IBM Spectrum Scale scales system throughput with each new server while still providing a single name space to the clients. This ability eliminates data silos and simplifies storage management. By consolidating storage requirements across your organization into IBM Elastic Storage Server, you can reduce inefficiency and acquisition costs while simplifying management and improving data protection.

The capabilities of IBM Elastic Storage Server include:

- Software RAID: IBM Spectrum Scale Redundant Array of Independent Disks (RAID) runs IBM disks in a dual-ported storage enclosure that does not require external RAID storage controllers or other custom hardware RAID acceleration.
- Declustering: IBM Spectrum Scale RAID distributes client data, redundancy information, and spare space uniformly across all disks of just a bunch of disks (JBOD). This distribution reduces the rebuild or disk failure recovery process compared to conventional RAID. Critical rebuilds of failed multi-terabyte drives with full of data can be accomplished in minutes, rather than hours or even days when using traditional RAID technology.
- Data redundancy: IBM Spectrum Scale RAID supports highly reliable 2-fault-tolerant and 3-fault-tolerant Reed-Solomon-based parity codes (erasure coding) as well as 3-way and 4-way replication.
- Large cache: Using a combination of internal and external flash devices along with large memory cache in the IBM Power® server, IBM Elastic Storage Server is better able to mask the inefficiencies and long latency times of nearline SAS drives, while still using the high density of the drives themselves.
- Graphical user interface (GUI): The intuitive GUI allows management and monitoring of the system, both locally and remotely.
- Superior streaming performance: The system can deliver over 25 GB per second of sustained performance.
- Scalability: As server configurations are added to an installed configuration, the capacity, bandwidth, performance and single name space all grow. This means installations can start small and grow as data needs to expand.

ownCloud Enterprise Edition

ownCloud is a self-hosted file sync and share server. It provides access to enterprise data through a web interface, PC sync clients, Web-based Distributed Authoring and Versioning (WebDAV) or Android / IOS applications, while providing a platform to view, sync, and share across devices easily. ownCloud's server enables IT to protect and manage files within the ownCloud environment—from file storage to user provisioning and data processing. The server provides a secure web interface through which administrators control all of ownCloud's resources, allowing authorized users to enable and disable features, set policies, create backups, and manage users. Advanced features for enterprise directory integration and file “firewalls” give admins exceptional flexibility and control. The server also manages and secures application programming interface (API) access to ownCloud, while providing the internal processing engine needed to deliver high-performance file-sharing services.

ownCloud also delivers the consumer grade experience users expect on desktops, notebooks, tablets, and mobile phones. Intuitive interfaces guide users through a wide range of file sharing activities, and administrator efficiency is aided through wizards, management tools, and monitoring and logging capabilities. ownCloud also provides the ability for standard WebDAV clients to access ownCloud files, enabling users to continue to use standards-based productivity tools to interoperate seamlessly with ownCloud.

ownCloud Enterprise and IBM storage infrastructure

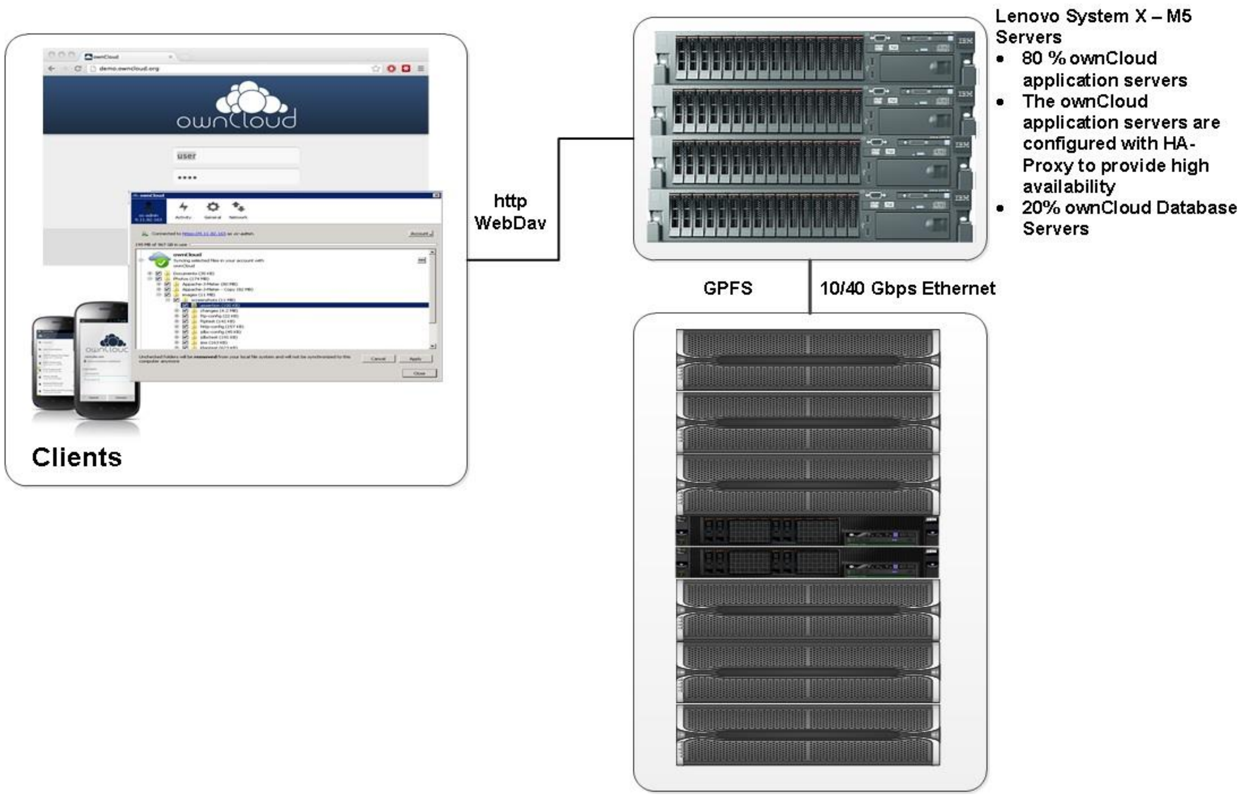

Lenovo System X – M5 Servers
• 80 % ownCloud application servers
• The ownCloud application servers are configured with HA-Proxy to provide high availability
• 20% ownCloud Database Servers
http WebDav
GPFS
10/40 Gbps Ethernet
Clients

Figure 1: Blueprint of IBM infrastructure running ownCloud Enterprise Edition. Application and database servers access ESS directly through GPFS

Figure 1 illustrates the lab file sync and share solution proof of concept of ownCloud Enterprise and IBM Elastic Storage Server. The lab proof of concept setup consists of System x86 servers' nodes running with ownCloud Enterprise application and database servers on RHEL Server release 7.2. The ownCloud application servers are configured behind a HAProxy load balancer to provide high availability and load balancing. The high-end Elastic Storage Server model GL6 is used for the lab proof of concept setup. The ownCloud application and database servers access Elastic Storage Server directly using IBM Spectrum Scale file system (formerly known as IBM GPFS™).

Today's exponential growth of data, transactions, and digitally-aware devices is demanding larger and larger amounts of unstructured data storage and management. An enterprise-class file sync and share solution is capable of scaling up to very large user numbers and requires a powerful and reliable storage back end as well as a suitably sized server infrastructure. The IBM Elastic Storage Server is designed to meet today's unprecedented data storage scaling to very large number of user requirements without compromising on performance.

IBM Spectrum Scale is the main core essence of IBM Elastic Storage Server. IBM Spectrum Scale enables the unification of virtualization, analytics, file and object use cases into a single scale-out storage solution. IBM Spectrum Scale can provide a single namespace for all of this data, offering a single point of management with an intuitive graphical user interface. Using storage policies transparent to users, data can be tiered, compressed, or migrated to help lower latency, improve performance, or cut costs. You can provide access around the globe using active file management (AFM) to help ensure that data is always available in the right place at the right time.

Proof of concept setup

In the lab, the team staged tests to estimate the amount of the different user access types, expected for typical usage scenarios within large organizations. The ownCloud environment was installed and setup by ownCloud Inc. on the System x86 server nodes (virtual machines) using a standard IBM-supported Linux® environment.

OS	Number of virtual processors	RAM	GPFS (Size)
Red Hat Enterprise Linux 7 (64 bit)	3	8 GB	4 TB (depending on the number of users, load should be increased accordingly)

Table 1: ownCloud application, database server, and client node (virtual machine) configuration

In the lab, file sync and share proof of concept setup, the system x86 nodes are attached to the IBM Elastic Storage Server on a 10GbE network infrastructure. The server node hardware details of the ownCloud application and database server are listed in Table 1.

On the back-end storage infrastructure side, the IBM Elastic Storage Server is configured with IBM Spectrum Scale declustered RAID array of eight data and three parity stripes (8+3p). The high-end IBM Elastic Storage Server GL6 model is installed with 2 TB hard disk drives (HDDs), which has a total of 403.1 TB of net space on the fully configured system.

In the lab proof of concept setup to test performance of ownCloud file sync and share solution, the ownCloud application and MariaDB database servers have been configured behind a HAProxy load balancer to provide high availability and distributed load balancing for performance and a *large user number* scenario testing.

Software – The detailed software setup is divided into three categories: the server system software, the ownCloud application configuration, and the benchmarking tools.

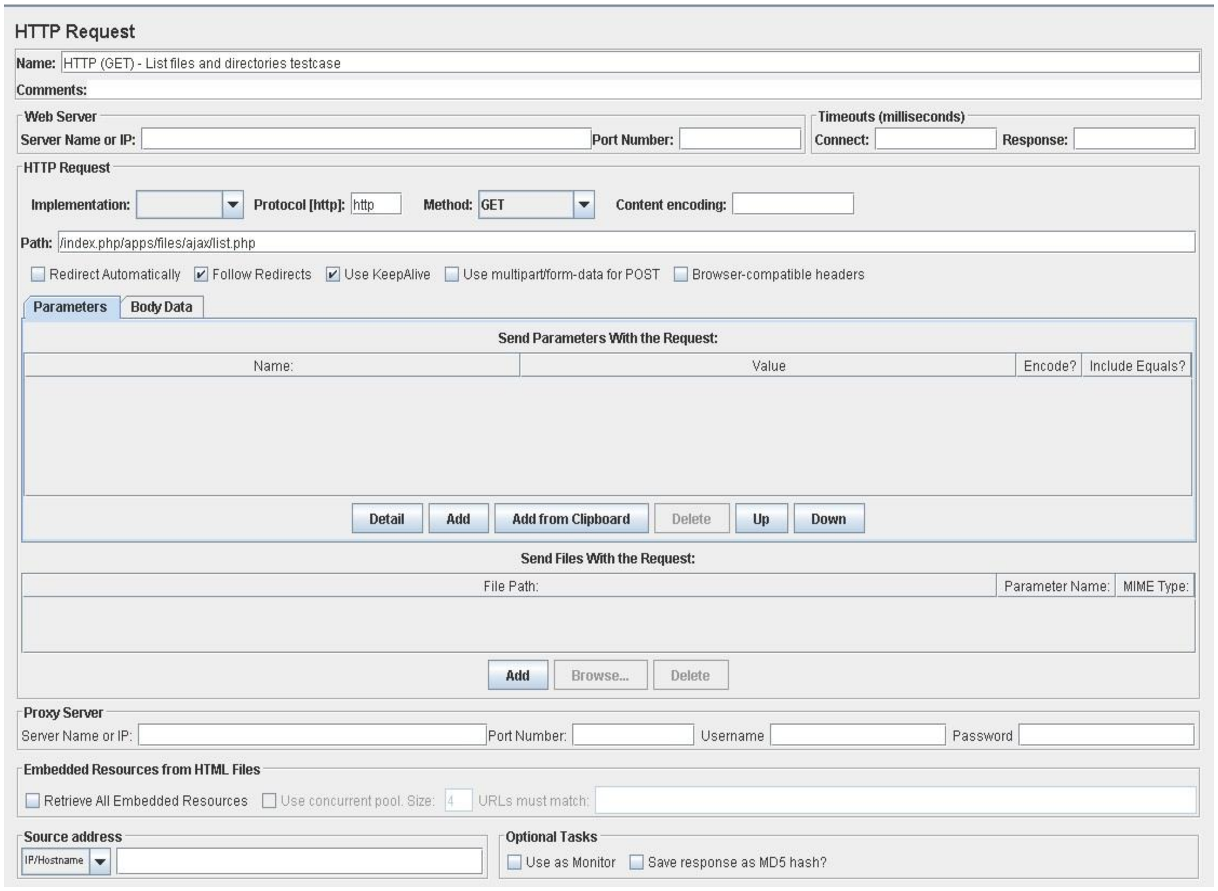
- System software:
  - Operating system on all servers: Red Hat Enterprise Linux Server release 7.2
  - Web server: Apache server version 2.4.6
  - Database: MariaDB Server 5.5.44
- ownCloud environment:
  - ownCloud Enterprise Edition 8.2.1
- Performance bench marking tool:
  - Apache JMeter (refer http://jmeter.apache.org)

Solution performance testing methods

The testing methods consist of a performance tuning after the installation and the final measurements of the maximum performance. A set of four measurements, using concurrent requests are run with Apache JMeter load and performance validation tool.

The following four test methods are concurrently run with Apache JMeter load and performance validation tool.

1. **Listing files and** directories **using HTTP_GET**: Configured Apache JMeter with ownCloud HAProxy load balance server and protocol set as **HTTP,** and **GET** method is selected on Apache JMeter and the test is run using the ownCloud script (/index.php/apps/files/ajax/list.php). Find the Apache JMeter test case definition as shown in Figure 2.


HTTP Request

Name: HTTP (GET) - List files and directories testcase

Comments:

Web Server — Server Name or IP: — Port Number:

Timeouts (milliseconds) — Connect: — Response:

HTTP Request — Implementation: — Protocol [http]: http — Method: GET — Content encoding:

Path: /index.php/apps/files/ajax/list.php

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Parameters | Body Data

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?

Send Files With the Request:

File Path:	Parameter Name:	MIME Type:

Proxy Server — Server Name or IP: — Port Number: — Username — Password

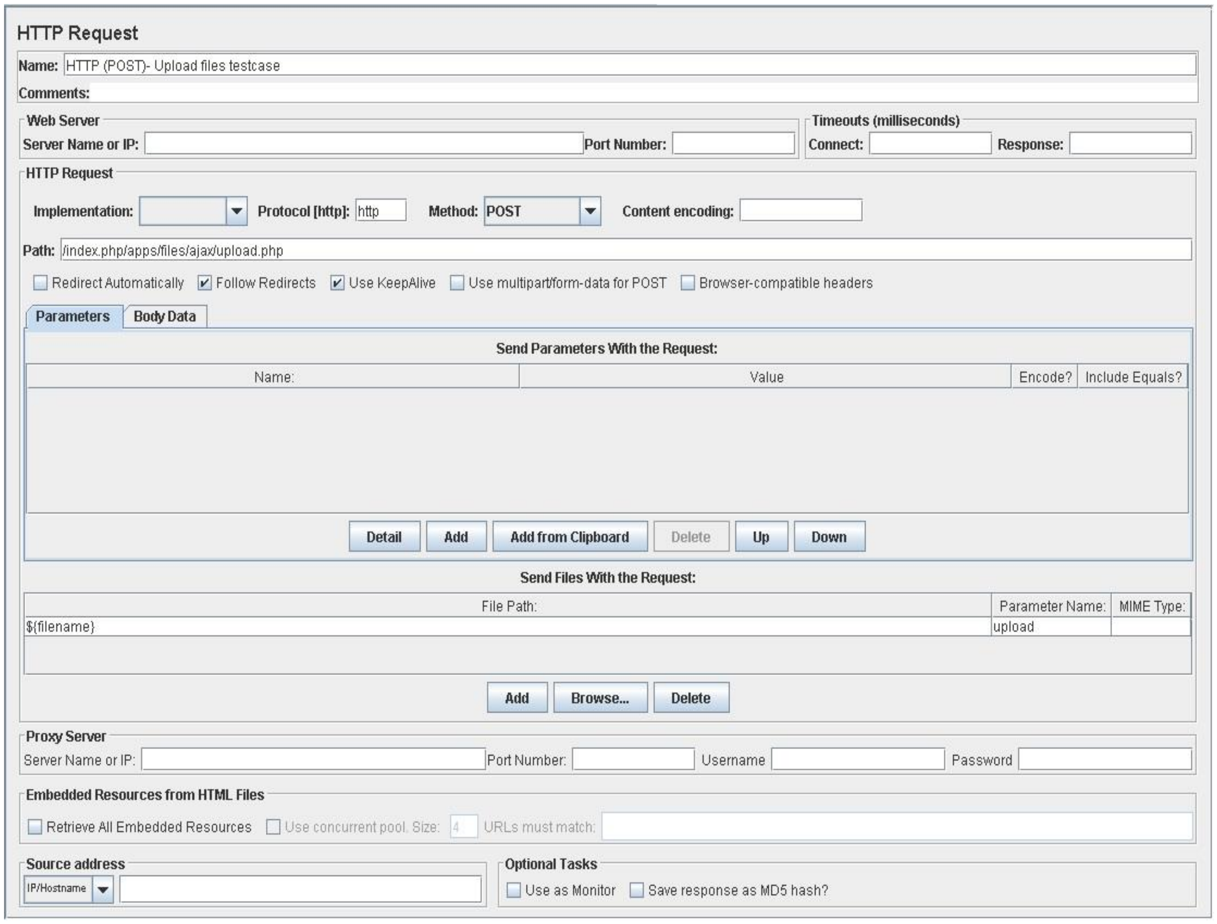
Embedded Resources from HTML Files — ☐ Retrieve All Embedded Resources ☐ Use concurrent pool. Size: 4 — URLs must match:

Source address — IP/Hostname

Optional Tasks — ☐ Use as Monitor ☐ Save response as MD5 hash?

Figure 2: HTTP_GET listing files and directories with Apache JMeter

2. **Random file upload test**: For this test, Apache JMeter has been configured with the **HTTP** protocol and the **POST** method. Test is run using the ownCloud upload script (/index.php/apps/files/ajax/upload.php). Find the test case definition on Apache JMeter as shown in Figure 3.


HTTP Request

Name: HTTP (POST)- Upload files testcase

Comments:

Web Server — Server Name or IP: — Port Number:

Timeouts (milliseconds) — Connect: — Response:

HTTP Request — Implementation: — Protocol [http]: http — Method: POST — Content encoding:

Path: /index.php/apps/files/ajax/upload.php

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Send Files With the Request:

File Path:	Parameter Name:	MIME Type:
${filename}	upload	

Embedded Resources from HTML Files — ☐ Retrieve All Embedded Resources ☐ Use concurrent pool. Size: 4

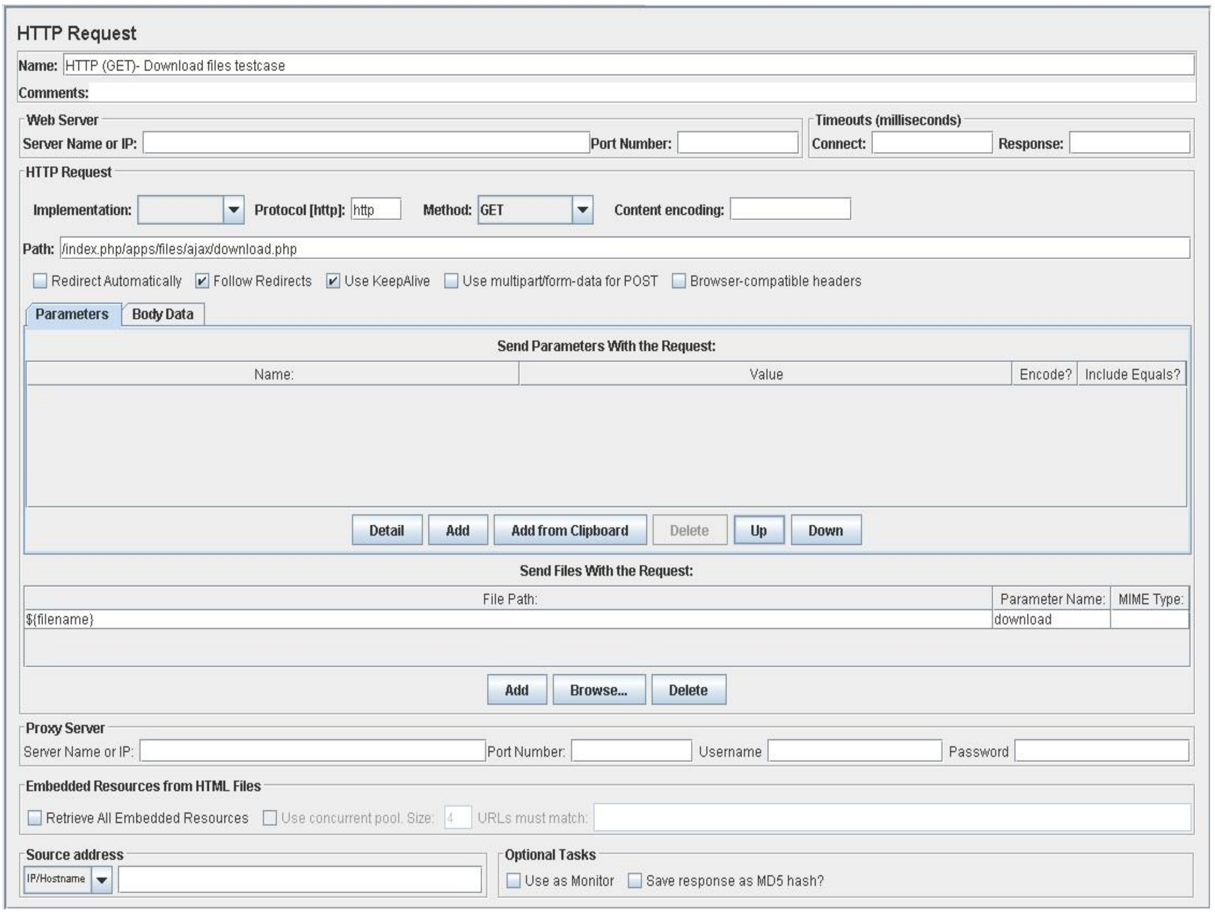
Source address — IP/Hostname

Optional Tasks — ☐ Use as Monitor ☐ Save response as MD5 hash?

Figure 3: Random files upload test case definition on Apache JMeter

Note: Also, the Bean Shell PreProcessor script is run to create a random file to upload, and a BeanShell PostProcessor script is run to delete the randomly created file as part of this test case. Refer to “Appendix B: BeanShell PreProcessor and BeanShell PostProcessor scripts”

3. **Download files test**: For this test, the Apache JMeter has been configured with using the **HTTP** protocol and the **GET** method. Test is run using the ownCloud download script (/index.php/apps/files/ajax/download.php). Find the test case definition on Apache JMeter as shown in Figure 4.



HTTP Request

Name: HTTP (GET)- Download files testcase

Comments:

Web Server
Server Name or IP: | Port Number:

Timeouts (milliseconds)
Connect: | Response:

HTTP Request
Implementation: | Protocol [http]: http | Method: GET | Content encoding:

Path: /index.php/apps/files/ajax/download.php

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?

Send Files With the Request:

File Path:	Parameter Name:	MIME Type:
${filename}	download	

Proxy Server
Server Name or IP: | Port Number: | Username | Password

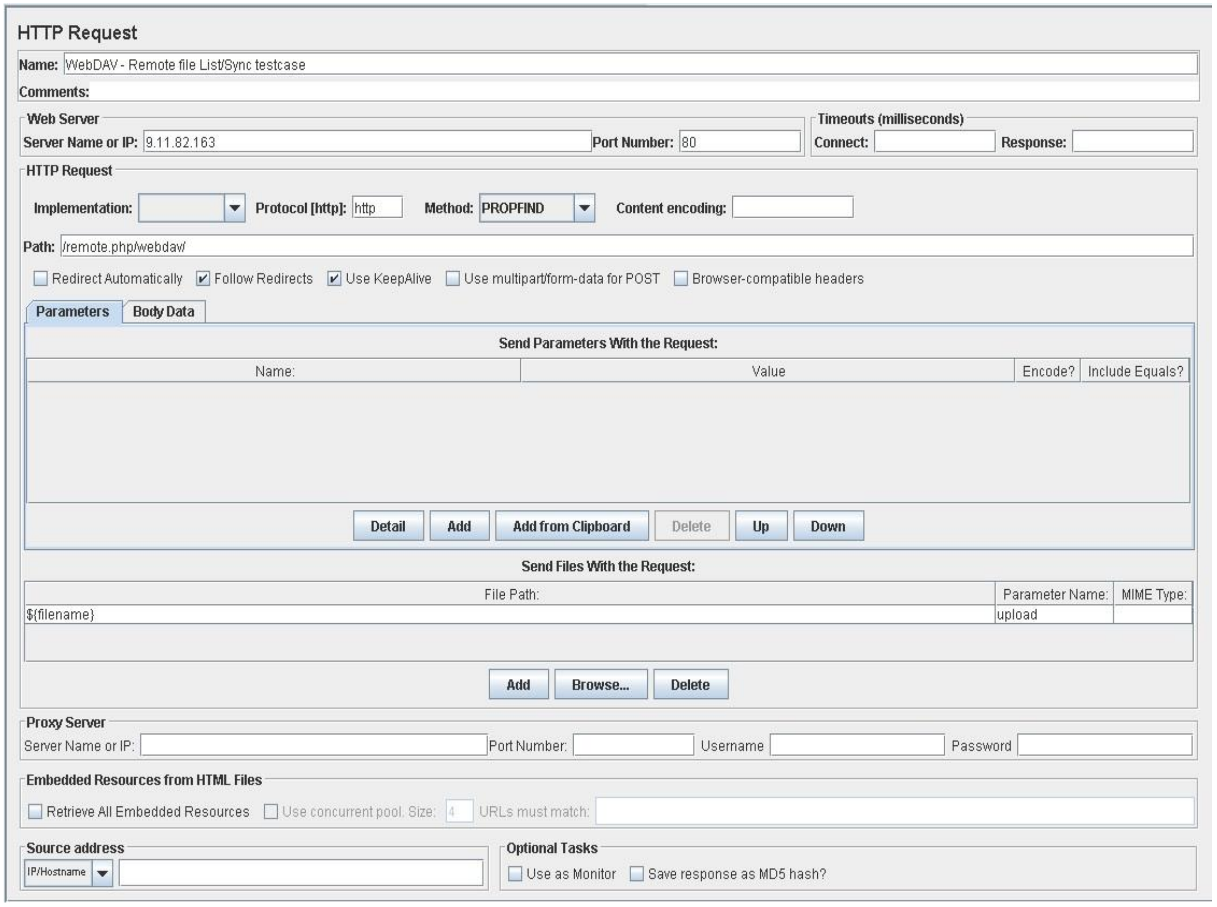
Embedded Resources from HTML Files
☐ Retrieve All Embedded Resources ☐ Use concurrent pool. Size: 4 | URLs must match:

Source address
IP/Hostname

Optional Tasks
☐ Use as Monitor ☐ Save response as MD5 hash?

Figure 4: Files download test definition on Apache JMeter

4. **WebDAV remote files list and sync test**: For this test, Apache JMeter is configured with using the **HTTP** protocol and the **PROPFIND** method. The ownCloud server configuration test is run using the ownCloud remote.php script (/remote.php/webdav/). Find the Apache JMeter test case definition, as shown in Figure 5.


HTTP Request

Name: WebDAV - Remote file List/Sync testcase

Comments:

Web Server — Server Name or IP: 9.11.82.163 — Port Number: 80

Timeouts (milliseconds) — Connect: — Response:

HTTP Request — Implementation: — Protocol [http]: http — Method: PROPFIND — Content encoding:

Path: /remote.php/webdav/

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?

Send Files With the Request:

File Path:	Parameter Name:	MIME Type:
${filename}	upload	

Proxy Server — Server Name or IP: — Port Number: — Username — Password

Embedded Resources from HTML Files — ☐ Retrieve All Embedded Resources ☐ Use concurrent pool. Size: 4 — URLs must match:

Source address — IP/Hostname

Optional Tasks — ☐ Use as Monitor ☐ Save response as MD5 hash?

Figure 5: Remote file sync (WebDAV) test case definition on Apache JMeter

Performance optimizations

There are number of options to calibrate the ownCloud installation and enable higher level of performance.

Recommendation: The recommended number of live Apache connections is 1000 or more. Similarly, the number of allowed MariaDB (RHEL) / MySQL connections also has to be increased for optimum performance. By using a tuned Alternative PHP Cache (APC) drastically increases the performance of the application server. Any given ownCloud deployment will have additional environment and policy specific configurations that also need to be revisited for gaining best performance.

The initial set of optimizations is mainly focused on the Apache connection settings. The recommended settings for the number of *MaxKeepAliveRequests* parameter should be to set within 10 to 200. In the lab proof of concept setup, APC is installed and the temporary space of MariaDB (RHEL) / MySQL is placed into the RAM disk. The second tuning interval consisted of pure Linux system parameter changes. Several IPv4related parameters were tuned, mostly related to the TCP protocol.

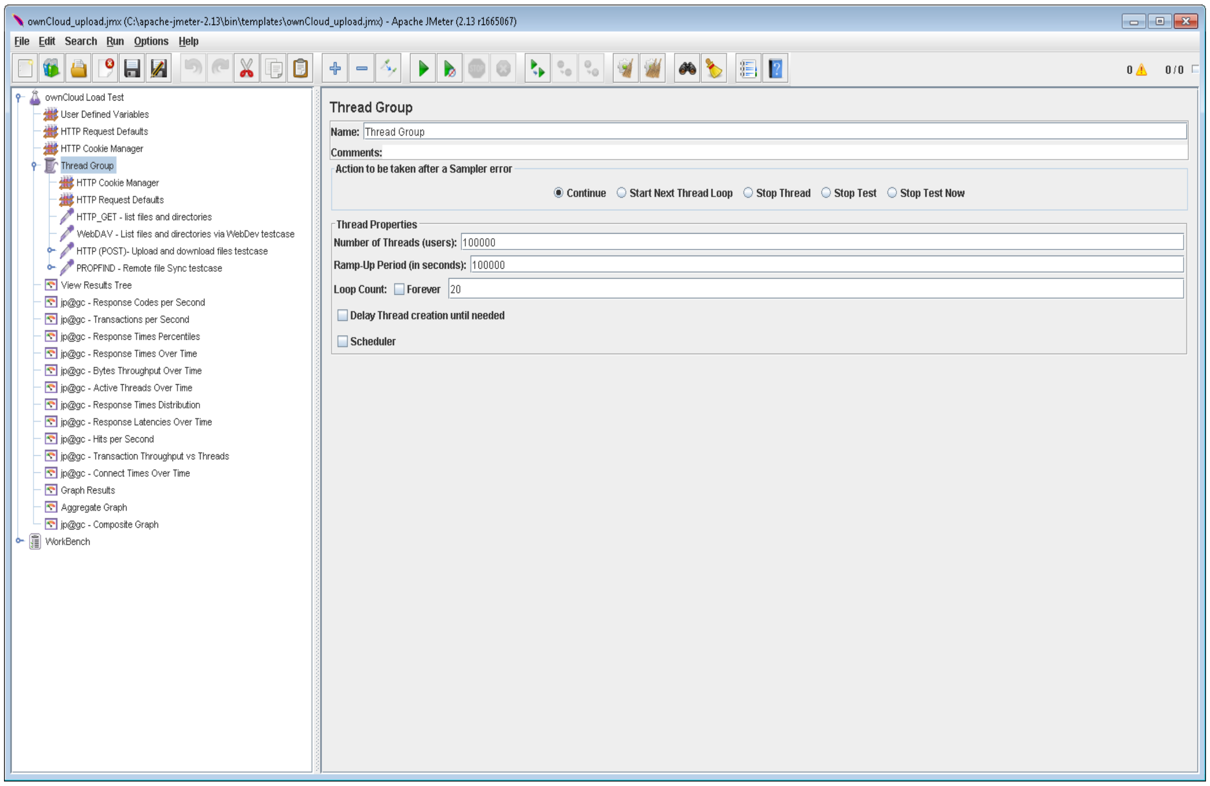
Apache tuning

An Apache process uses around 12MB of RAM. Apache should be configured so that the maximum number of HTTPD processes times 12MB is lower than the amount of RAM. Otherwise the system begins to swap and the performance goes down.

The Apache server configured in the lab is set to a maximum number of 6000.

For more detailed ownCloud installation and optimization, refer: https://doc.owncloud.org/server/8.2/admin_manual/configuration_server/performance_tuning.html

The simulation of ownCloud installation for 100000 users on Apache JMeter is shown in Figure 6. The recorded results of byte received and sent rate after tuning are presented in Figure 7 shows the results of the bytes received and sent per second before optimizing ownCloud and Figure 8 shows the results of the bytes received and sent per second after optimizing ownCloud.


ownCloud_upload.jmx (C:\apache-jmeter-2.13\bin\templates\ownCloud_upload.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
0 0 / 0

- ownCloud Load Test
  - User Defined Variables
  - HTTP Request Defaults
  - HTTP Cookie Manager
  - Thread Group
    - HTTP Cookie Manager
    - HTTP Request Defaults
    - HTTP_GET - list files and directories
    - WebDAV - List files and directories via WebDev testcase
    - HTTP (POST)- Upload and download files testcase
    - PROPFIND - Remote file Sync testcase
  - View Results Tree
  - jp@gc - Response Codes per Second
  - jp@gc - Transactions per Second
  - jp@gc - Response Times Percentiles
  - jp@gc - Response Times Over Time
  - jp@gc - Bytes Throughput Over Time
  - jp@gc - Active Threads Over Time
  - jp@gc - Response Times Distribution
  - jp@gc - Response Latencies Over Time
  - jp@gc - Hits per Second
  - jp@gc - Transaction Throughput vs Threads
  - jp@gc - Connect Times Over Time
  - Graph Results
  - Aggregate Graph
  - jp@gc - Composite Graph
- WorkBench

Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 100000

Ramp-Up Period (in seconds): 100000

Loop Count: ☐ Forever 20

☐ Delay Thread creation until needed

☐ Scheduler

Figure 6: Running the 100000-user simulation (thread group) with Apache JMeter

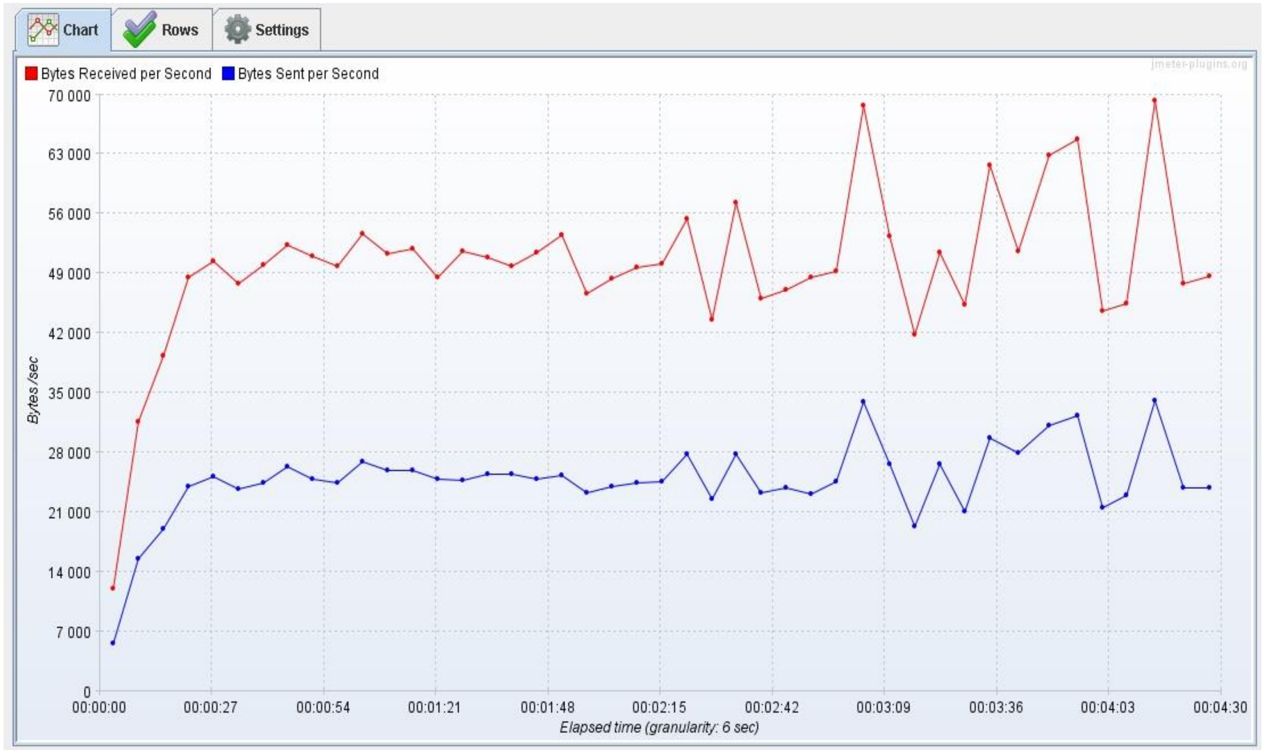


*Figure 7: Results of the bytes received per second and bytes sent per second **before** optimizing ownCloud*

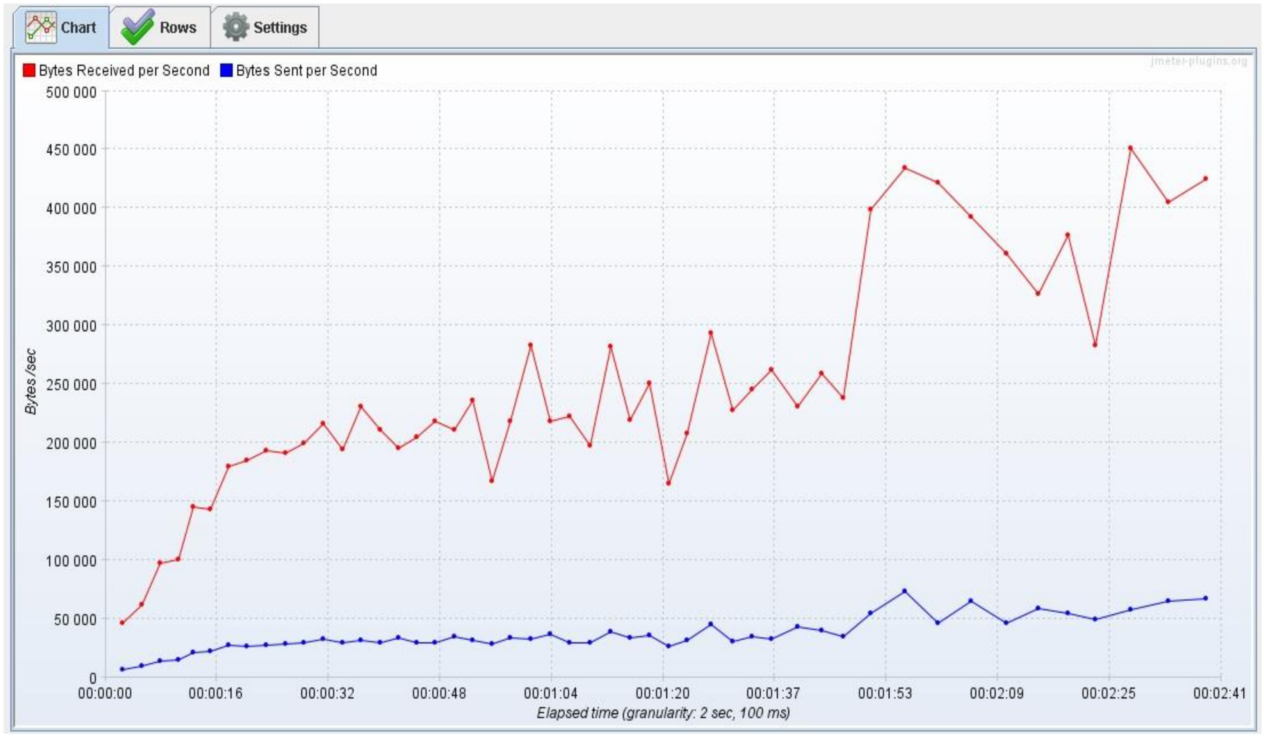


*Figure 8: Results of the bytes received per second and bytes sent per second **after** optimizing ownCloud*

The graphs in Figure 7 and Figure 8 show the ownCloud installation load testing by Apache JMeter. These graphs depicts the amount of bytes that ownCloud can receive and the number of bytes that Apache JMeter can send before and after the tuning of ownCloud installation.

Figure 7 illustrates that the ownCloud was able to receive approximately 430,000 to 45,000 bytes per second and Apache JMeter was able to send approximately 32,000 to 17,000 bytes per second before tuning the ownCloud installation for 100000 simultaneous thread group or user-load simulation.

Figure 8 illustrates that ownCloud was able to receive approximately up to 430,000 bytes per second and Apache JMeter can send approximately more than 50,000 bytes per second after tuning the ownCloud installation for 100000 simultaneous thread group or user-load simulation.

Note: The Apache JMeter run all the four listed test cases (explained in this section) in the proof of concept lab setup, in order to get the required results.

Large enterprise user account number systems

For large enterprise installations, with a large number of user accounts, the ownCloud Enterprise Edition features the provisioning API to enable a comfortable way to provision and de-provision accounts using the Representational State Transfer (REST) interface. In order to test the scalability concerning user account numbers, 100000 accounts were created with the lab proof of concept test environment and the performance tests were repeated. No measurable impact on the server performance could be observed.

You can find more details about ownCloud Enterprise Edition provisioning APIat: https://doc.owncloud.org/server/8.2/admin_manual/configuration_user/user_provisioning_api.html

Proof of concept performance measurement

In the lab proof of concept test environment, the following web service features of ownCloud were tested and the measurements were recorded using Apache JMeter.

- List files and directories: HTTP (GET)
- Upload files: HTTP (POST)
- Download files: HTTP (GET)
- Remote listing files and directories and sync through WebDAV: PROPFIND and (/remote.php/WebDAV) script

The detailed proof of concept solution lab test cases using Apache JMeter are listed in the section “Solution performance testing methods”.

For correctly recording test results using Apache JMeter, the lab proof of concept test setup is carefully staged with a load simulation of 1000, 10,000 and extreme 100,000 thread group or users.

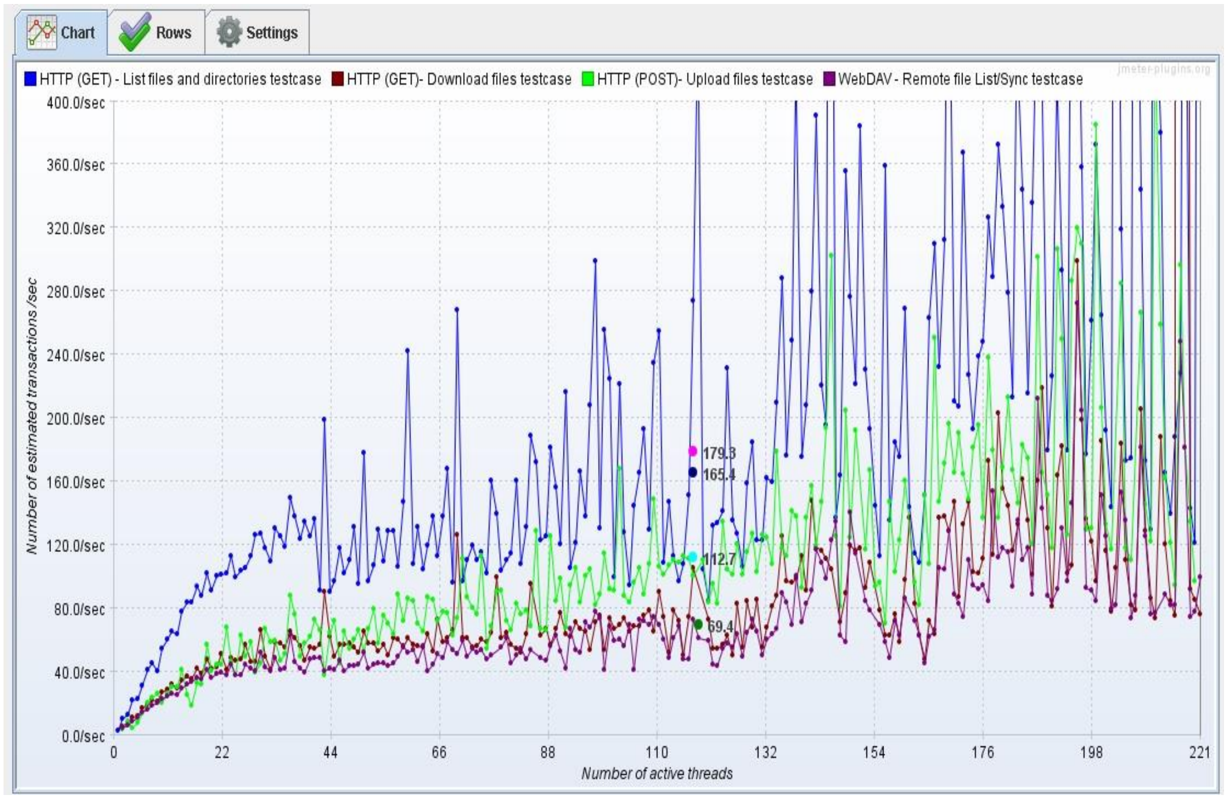


Figure 9: Transaction throughput versus threads illustrating the average of each test case recorded in real time for more than 125 threads

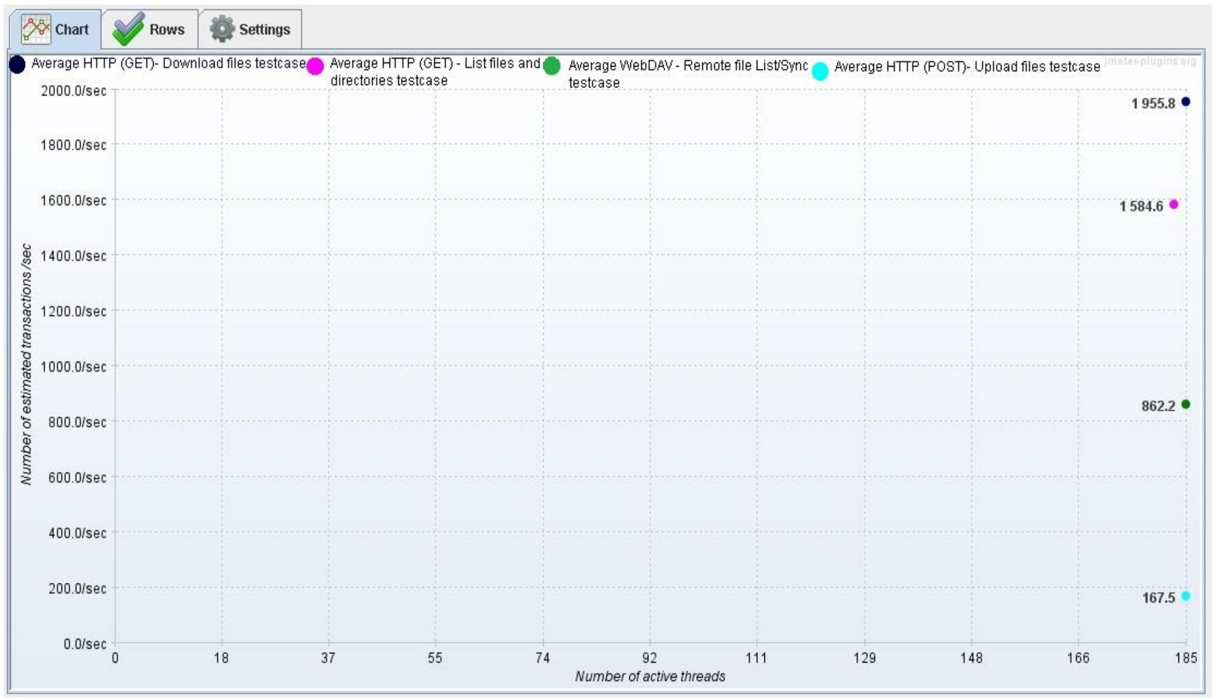


Figure 10: Transaction throughput versus threads illustrating the average of each test case for 185 threads

Figure 9 shows the recorded transaction throughput for threads. In this chart, for approximately more than 125 threads (user simulation), the following number of transactions are recorded per second for the test cases.

- HTTP (GET) - List files and directories; test case recorded 179.3 transactions per second
- HTTP (GET)- Download files; test case recorded 165.4 transactions per second
- HTTP (POST)- Upload files; test case recorded 112.7 transactions per second
- WebDAV - Remote file list / sync; test case recorded 69.4 transactions per second

Figure 10 shows the recorded transaction throughput for each test case with 185 threads, which is a level much higher than the expected load of average transactions.

- HTTP (GET) - List files and directories; test case recorded 1586.6 transactions per second
- HTTP (GET) - Download files; test case recorded 1955.8 transactions per second
- HTTP (POST) - Upload files; test case recorded 167.5 transactions per second
- WebDAV - Remote file list / sync; test case recorded 862.2 transactions per second

In the lab, proof of concept load test results are recorded for a load of 10,000 user threads. The system can handle each load test case without any issues.

The upload and download test cases using HTTP PUT and GET methods are performed after creating approximately 1 MB and 1 GB sized random files.

Apart from these test case results, separate stand-alone sync client tests also staged for more than 4500 files and mostly large size files (above 1 GB) are used during stand-alone desktop sync client tests.

The Apache JMeter uses the following formula for calculating the total server transaction throughput.

[**<active threads> * 1 second (1000 ms) / <1 thread response time>**]

Example: If 10 threads send requests to the ownCloud server and the ownCloud server responds after 100 ms, the transaction throughput is calculated as 10 threads * 1000 ms / 100 ms = 100 transactions per second.

Approximately, more than 3500 transactions per second was recorded during stand-alone sync client tests using PROPFINDs with 100,000 users load simulation test.

WebDAV interface-based transactions were approximately within 15% to 20% behind the HTTP GET and PROPFIND transactions. Concurrent upload of files tests with HTTP POST were also recorded behind the HTTP GET and PROPFIND transactions.

Refer to "Appendix C: Lab proof of concept for a load of 1000 users " of the lab proof of concept performance bench mark results.

Sizing assumptions and considerations – the 100,000-user scenario

In order to estimate the required infrastructure size of a comparable system like the one used in the lab proof of concept setup, a general assumption on the average user behavior and therefore the distribution of their access patterns has to be taken into account. Furthermore, to give a quantitative impression within the following section, the test team extrapolated the combination of assumed usage patterns and measured performance to a count of 100,000 active users and translated the results into hardware sizing numbers.

When using ownCloud within an enterprise grade environment using the file sync and share system as a central data hub within the daily work of each user, the desktop sync clients are of major importance. They are running permanently to keep their files in sync with the server. The default syncing rate of the desktop clients triggers PROFIND requests for every 30 seconds to check the synchronization status and to take action, if needed. 100,000 active desktop sync clients using the default setup lead to an average number of approximately up to 3600 PROFINDs each second. Using the benchmark results, this translates into a required reference application server node number of slightly less than six application servers that should be able to cope up with the load generated by all desktop syncing clients. Note that the synchronization frequency is adjustable. It is possible to intercept peak load times by automatically extending the synchronization interval.

Assuming that each user additionally uses a mobile device to connect to the ownCloud instance with an average access rate of one mobile access per hour, this adds up to 30 requests per second for each request type, PROPFIND and HTTP_GET. According to lab proof of concept benchmarks, the performance of about a quarter of the reference application server is needed additionally.

The web portal usage is assumed to be of the same order as mobile access. However, you can exchange the PROPFIND requests with additional HTTP_GET calls of the same number, resulting in 60 (2 * 30) HTTP_GET requests per second on the whole system. This translates to about a 0.20 seconds.

Finally, file upload and download have to be taken into account. You can anticipate average rates of one upload and two downloads per hour. The benchmark results lead to a required number of about 0.75 nodes for downloads and 0.1 nodes for uploads. You can round that up to one node for the file transfers.

Overall, for the 100,000-user scenario (including a buffer of about 20% for the interception of peak load times, the five hosts for the desktop sync clients, one and a half host for web and mobile access and file transfers, and finally two database servers) a rounded number of ten ownCloud workload server nodes and an IBM Elastic Storage Server (GL-6) are assumed to cope with the expected load.

Enterpriseclass data availability feature of IBM Elastic Storage

IBM Elastic Storage delivers a number of features that along with a high availability infrastructure help ensure a reliable enterprise-class storage solution. Robust clustering features and support for synchronous and asynchronous data replication make IBM Elastic Storage fault-tolerant and can help provide continued data access even if cluster nodes or storage systems fail.

IBM Elastic Storage includes the infrastructure to handle data consistency and availability, and does not rely on external applications for cluster operations such as node failover. In an IBM Elastic Storage cluster, all nodes see all data, and all cluster operations can be conducted through any node in the cluster. All nodes are capable of performing all the tasks and are not limited by who owns the data or is connected to the disks. The tasks that a node can perform are determined by the license type and the cluster configuration.

As part of the product’s built-in availability tools, IBM Elastic Storage continuously monitors the health of the file system components. When failures are detected, IBM Elastic Storage attempts automatic recovery. Extensive journaling and recovery capabilities help maintain metadata consistency when a node holding locks or performing administrative services fails.

Finally, IBM Elastic Storage supports snapshots, enabling enterprises to protect the file system’s contents against user error. Snapshots can be used as an online backup capability that allows files to be recovered easily from common problems, such as accidental file deletion.

Summary

In order to estimate the required IBM hardware infrastructure size for a large scale ownCloud Enterprise setup for six digit user numbers, the proof-of-concept described within this paper successfully delivered a basis for the sizing of future joint ownCloud and IBM solutions.

The derived numbers for the 100,000-user scenario has proven to be of great performance and leads to the conclusion that using a single IBM Elastic Storage Server and ownCloud Enterprise Edition, you can deliver an enterprise-class file sync and share environment for more than 100,000 users.

The test team did not reach the performance limit of the IBM Elastic Storage Server at all, whereas, it turned out that the sizing of the application and database servers is crucial.

The IBM Elastic Storage Server used for staging the ownCloud file sync and share lab proof of concept solution for performance and sizing information was simultaneously used for additional backup / restore solution performance benchmark purpose. This underlines the fact that IBM Elastic Storage can be used for hosting several industry solutions for storing critical data simultaneously.

The successful cooperation of ownCloud Inc. and IBM enables the delivery of perfectly customized solutions for your clients. ownCloud provides enterprises with a highly scalable on-premises alternative to consumer-grade and cloud-based applications. By installing ownCloud on-premises, fully integrated into existing identity management, security, governance, backup, and disaster recovery tools, organizations can be in full control of their sensitive data. Furthermore, organizations have the flexibility to choose among on-premises storage, cloud, and a hybrid model to find the model that perfectly matches their requirements.

You can find more information about implementing enterprise-class on-premises file sync and share solution using onwCloud Enterprise Edition on object storage with IBM Spectrum Scale or ESS at:

https://owncloud.com/wp-content/uploads/2015/07/WP-On-premise-File-Sync-Share-IBM-Spectrum-Scale-ownCloud.pdf

You can also find additional information at https://owncloud.com/ibm

Appendix A: Test environment

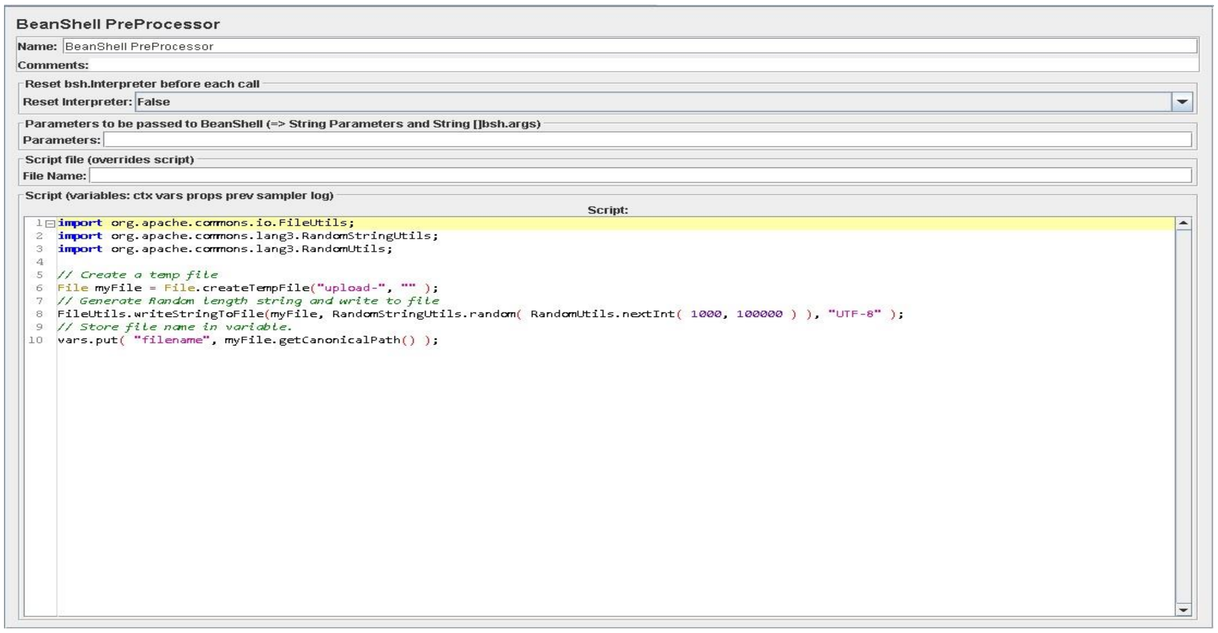
The following information provides details about the environment used for testing the solution.

Sr. No	Description	Model / Version
1	IBM Elastic Storage Server	GL-6
2	IBM Spectrum Scale	4.1.1
3	ownCloud Enterprise Edition	8.2.1

Table 2: Test environment

Appendix B: BeanShell PreProcessor and BeanShell PostProcessor scripts

Bean Shell PreProcessor script to randomly create a file



BeanShell PreProcessor

Name: BeanShell PreProcessor

Comments:

Reset bsh.Interpreter before each call

Reset Interpreter: False

Parameters to be passed to BeanShell (=> String Parameters and String []bsh.args)

Parameters:

Script file (overrides script)

File Name:

Script (variables: ctx vars props prev sampler log)

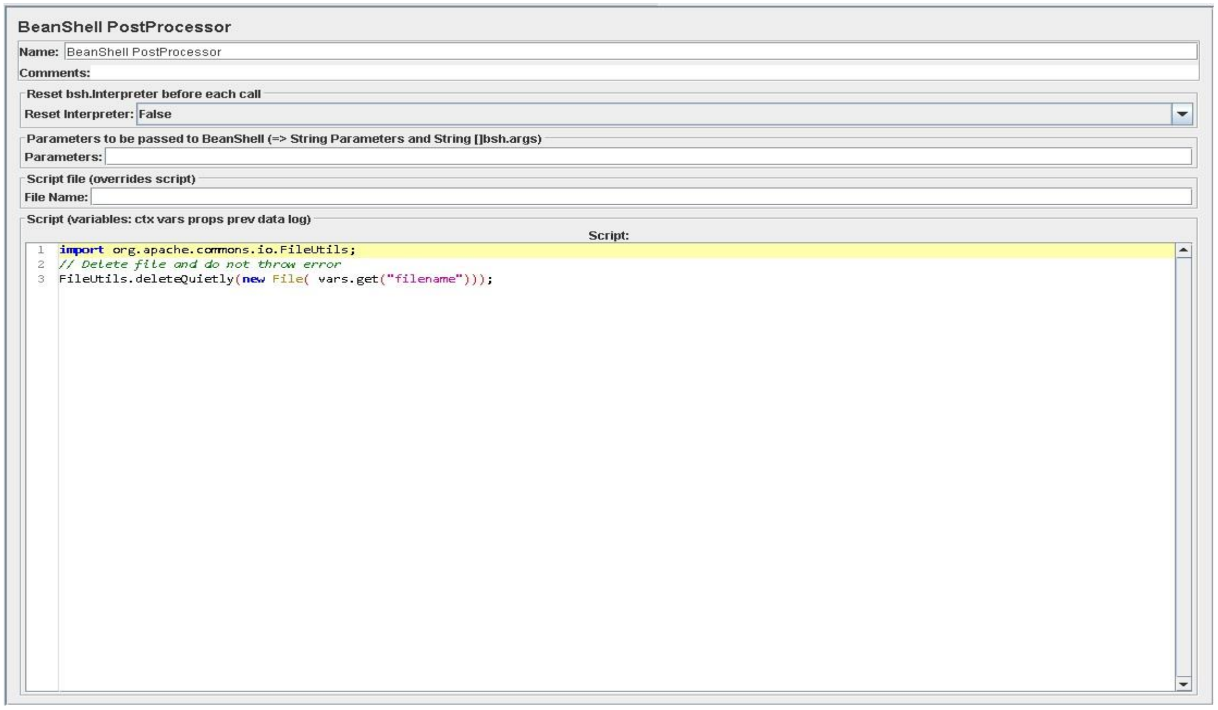
Script:

```
import org.apache.commons.io.FileUtils;
import org.apache.commons.lang3.RandomStringUtils;
import org.apache.commons.lang3.RandomUtils;

// Create a temp file
File myFile = File.createTempFile("upload-", "" );
// Generate Random length string and write to file
FileUtils.writeStringToFile(myFile, RandomStringUtils.random( RandomUtils.nextInt( 1000, 100000 ) ), "UTF-8" );
// Store file name in variable.
vars.put( "filename", myFile.getCanonicalPath() );
```

Figure 11: BeanShell PreProcessor script

BeanShell PostProcessor script to delete the randomly created file



BeanShell PostProcessor

Name: BeanShell PostProcessor

Comments:

Reset bsh.Interpreter before each call

Reset Interpreter: False

Parameters to be passed to BeanShell (=> String Parameters and String []bsh.args)

Parameters:

Script file (overrides script)

File Name:

Script (variables: ctx vars props prev data log)

Script:

```
import org.apache.commons.io.FileUtils;
// Delete file and do not throw error
FileUtils.deleteQuietly(new File( vars.get("filename")));
```

Figure 12: BeanShell PostProcessor script

Appendix C: Lab proof of concept for a load of 1000 users

The following figures show the load simulation results generated using Apache JMeter.

ownCloud load test result –proof of concept lab response time distribution graph

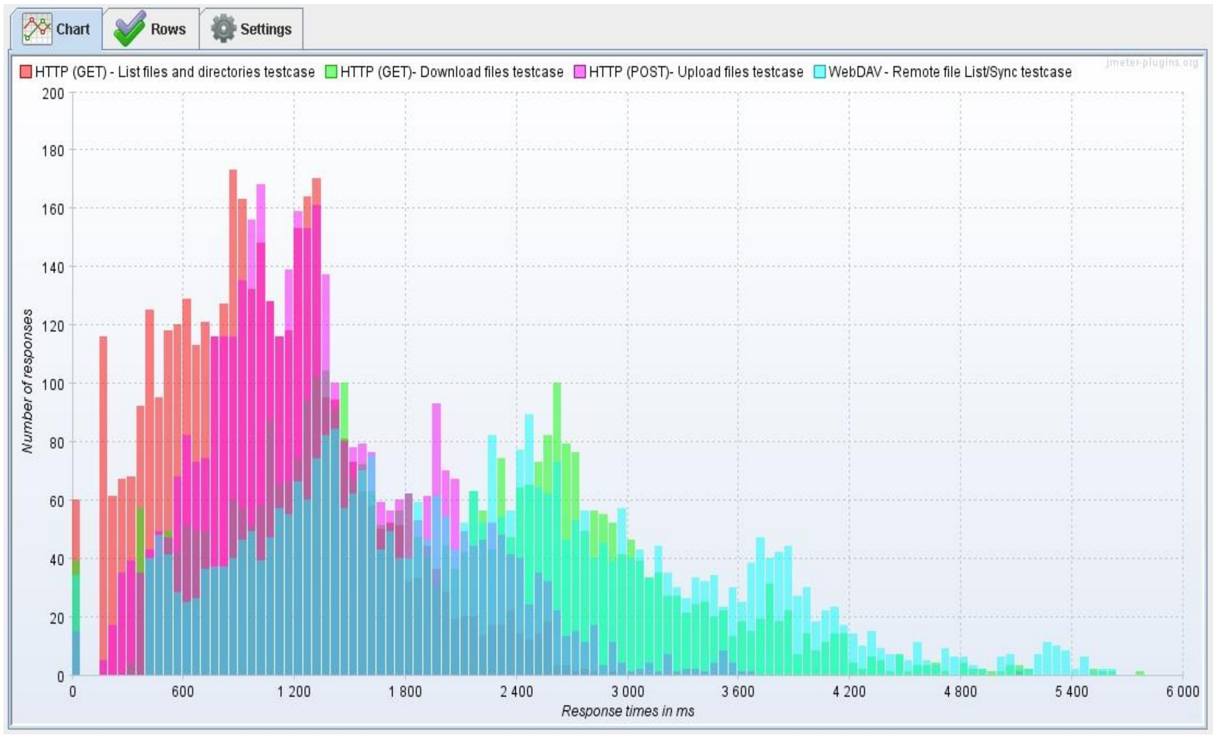


Figure 13: Proof of concept lab response time distribution for all four test cases

Overall response time graph – lab proof of concept of ownCloud load test result

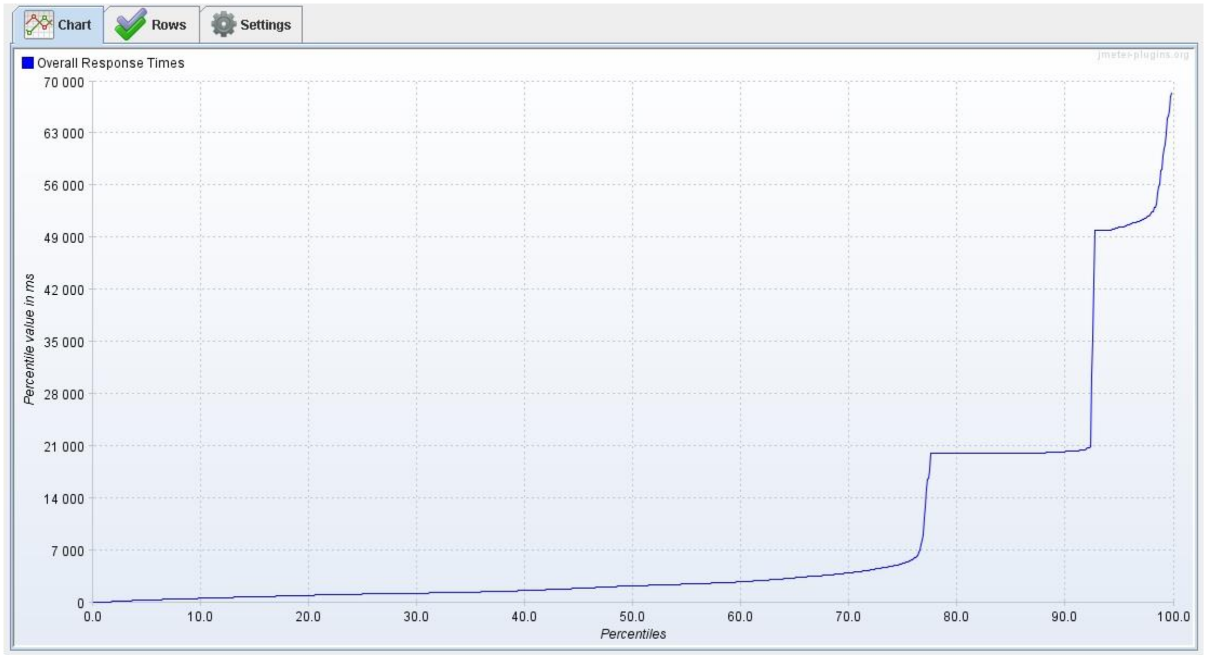


Figure 14: Apache JMeter load test result – overall response time graph

Response time over time graph – lab proof of concept of ownCloud load test result

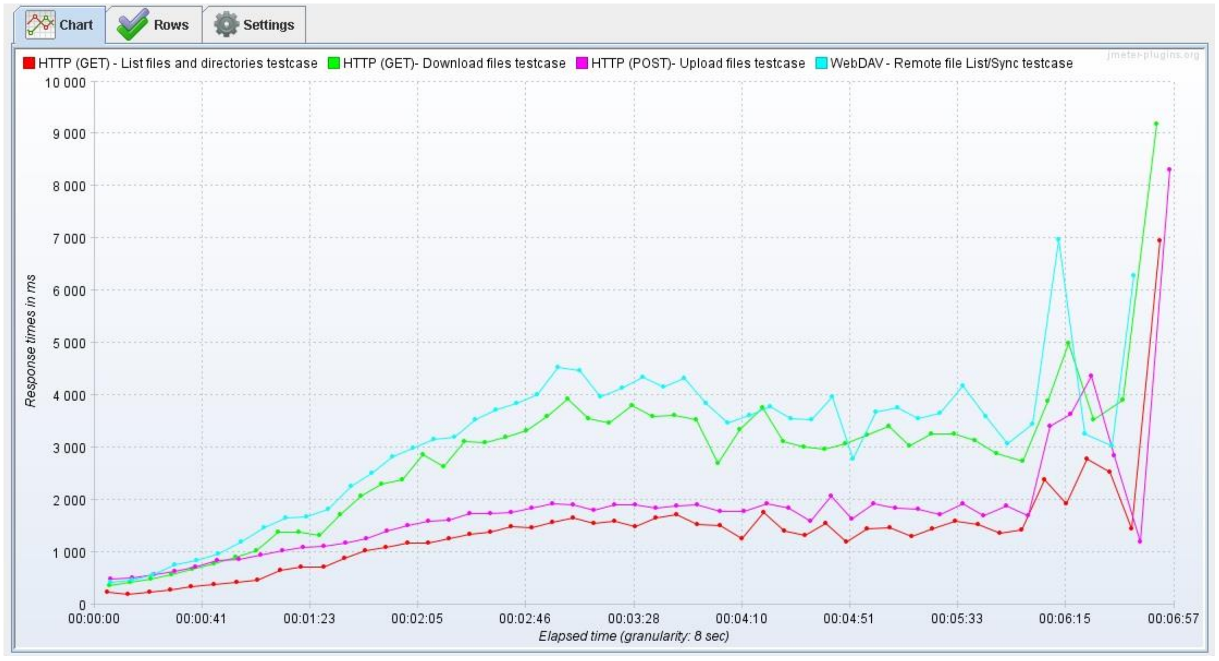


Figure 15: Apache JMeter load test result – response time over time graph

Additional composite graph

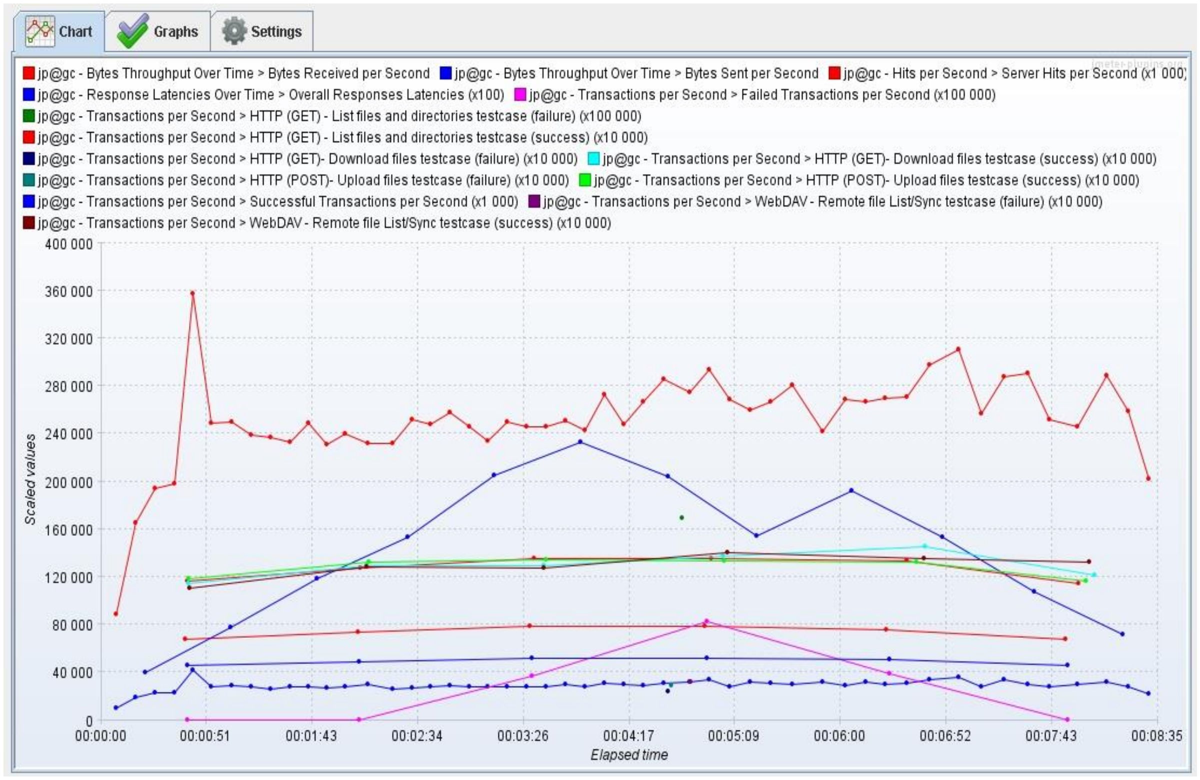


Figure 16: Apache JMeter composite graph

Appendix D: Resources

The following websites provide useful references to supplement the information contained in this paper:

- IBM Systems on PartnerWorld®
  ibm.com/partnerworld/systems

- IBM Redbooks®
  ibm.com/redbooks

- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US

- IBM System Storage Interoperation Center (SSIC)
  ibm.com/systems/support/storage/config/ssic/displayesssearchwithoutjs.wss?start_over=yes

- IBM Spectrum Scale
  ibm.com/systems/storage/spectrum/scale

- IBM Techdocs Library
  ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs

- ownCloud 8.2 User Manual
  https://doc.owncloud.org/server/8.2/ownCloud_User_Manual.pdf

- ownCloud 8.2 Administration Manual
  https://doc.owncloud.org/server/8.2/ownCloud_Server_Administration_Manual.pdf

- ownCloud 8.2 Developer Manual
  https://doc.owncloud.org/server/8.2/ownCloudDeveloperManual.pdf

- Apache JMeter User Manual
  http://jmeter.apache.org/usermanual/index.html

- Apache JMeter Best Practices
  http://jmeter.apache.org/usermanual/best-practices.html

About the authors

Udayasuryan Kodoly (Uday) is a storage technology solution architect in the IBM Systems ISV Enablement organization. Uday has extensive experience in designing, architecting storage solutions, and developing solution best practices for enterprise data centers and cloud infrastructures, and building and sustaining customer and partner relationships. Presently, Uday is engaged in planning, designing, and developing IBM Spectrum Scale, software-defined storage and cloud infrastructure solutions integrating with various ISVs.

You can reach Uday at uakodoly@us.ibm.com.

Dr. Oliver Oberst is an IBM industry solution architect, mainly responsible for clients in research and higher education in central Europe. Oliver started as a researcher in particle physics dealing with data management and data analysis of huge amounts of particle accelerator data. Today, he is working within the IBM Sales and Distribution organization and designs client solutions by incorporating all aspects of the IBM software, hardware, and cloud portfolio. Oliver was the lead architect in one of the large scale ownCloud on IBM Spectrum Scale solutions.

You can reach Dr. Oliver Oberst at oliver.oberst@de.ibm.com

Chris Burnard is a senior solutions architect at ownCloud, Chris has over 25 years of experience in large scale enterprise deployments and technologies, specializing in high availability solutions using storage, sever, network, and data center technologies. Currently, Chris works with ownCloud Enterprise customers, designing and deploying ownCloud enterprise file sync and share solutions.

You can reach Chris Burnard at cburnard@owncloud.com

@IBMSystemsISVs

Trademarks and special notices