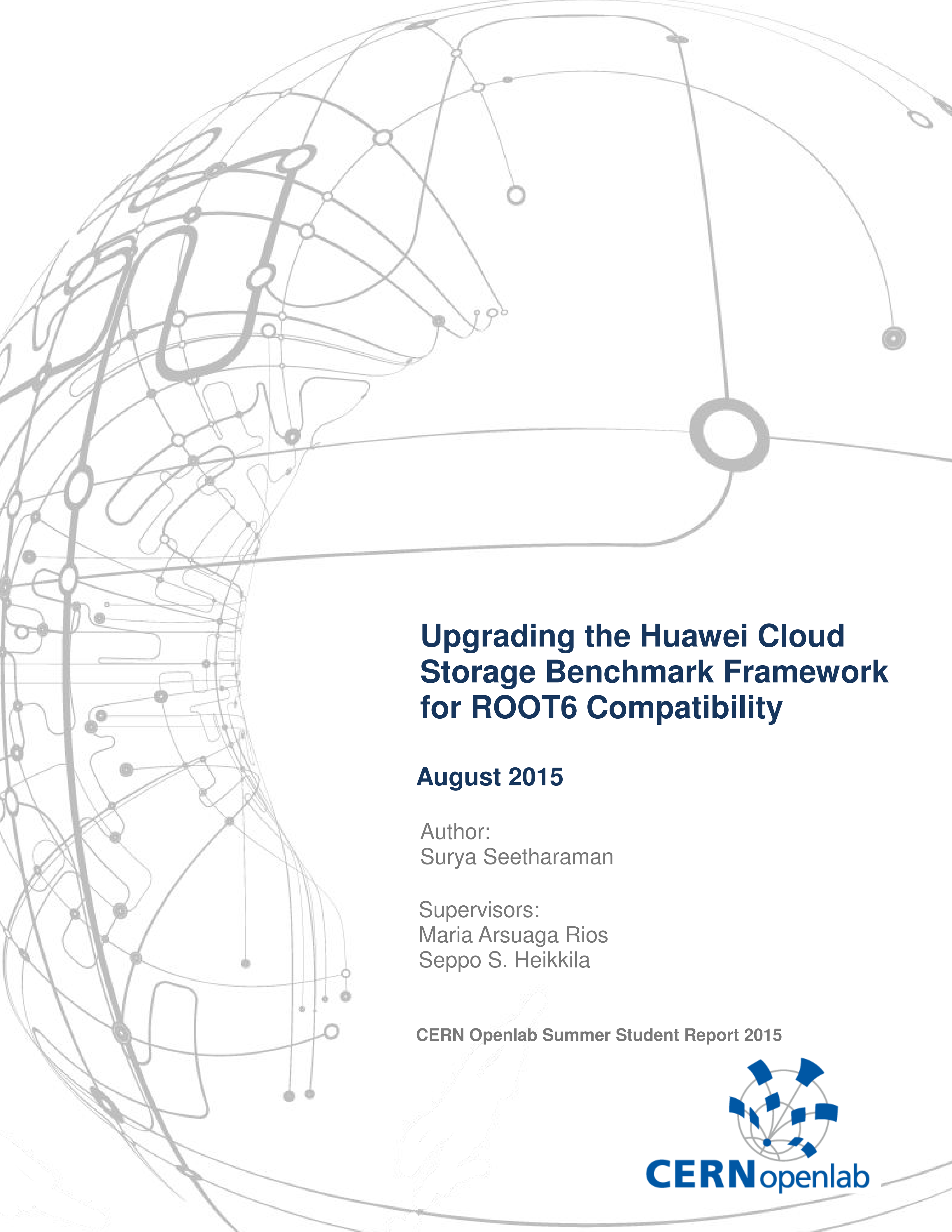


Upgrading the Huawei Cloud Storage Benchmark Framework for ROOT6 Compatibility

August 2015

Author:
Surya Seetharaman

Supervisors:
Maria Arsuaga Rios
Seppo S. Heikkila

CERN Openlab Summer Student Report 2015

Project Specification and Motivation

The multi-client benchmark framework of the Huawei Cloud Storage system at CERN was using ROOT5 to provide an analysis about the collection of data retrieved by the clients. Recently a new version of ROOT (ROOT6) was released with many improvements. Hence, to use the benchmark framework more effectively, it had to be upgraded so that it could run using ROOT6. Also this benchmark framework is generic in the sense that this upgraded benchmark framework can be used for measuring the scalability of other storage technologies at CERN like Kinetic Drives or EOS.

As a part of the Data and Storage Services group of the IT department (IT-DSS) at CERN, the task was to upgrade the cloud storage benchmark framework and run the existing tests on the upgraded benchmark.

Abstract

The European Organisation for Nuclear Research (CERN) is the largest research center in the world in the field of particle physics. Huge amounts of data, in the order of Peta Bytes, are being generated from the High Energy Physics experiments going on at CERN. The massive data growth prompts CERN to evaluate new storage technologies, to store and analyse the data that is being generated from the experiments. The scalability of the storage system is important for CERN as the laboratory faces the ever increasing demands of its physics users.

Huawei, a CERN collaboration member, focuses on investigating the applicability of new storage techniques and architectures to the storing of high energy physics data from the LHC experiments. In early 2012, Huawei's cloud storage system was delivered to the CERN site, and in three months, the installation and benchmark performance evaluation were completed. Huawei's cloud storage proved to show horizontal scalability in writing and reading performance and capability to automatically recover from storage node failures.

One of the main objectives of the Huawei - CERN partnership is to investigate the performance and scalability of a new cloud storage system and compare the results with more traditional approaches used at CERN. To perform this quantitative study, a multi-client benchmark has been developed using ROOT to allow statistical data analysis. Currently, a new version of ROOT has been released with many improvements. In order to allow for more efficient testing and rapid deployment of new benchmark workloads, the existing multi-client benchmark had to be upgraded to be compatible with ROOT 6.

This report documents and justifies that the benchmark framework was upgraded and verified to be able to produce the same results as the previous benchmark framework before the upgrade.

Table of Contents

1 Introduction

1.1 Huawei Universal Distributed Storage (UDS)

The Huawei UDS is a massive data storage system and two of its setup is physically located at the CERN Data Computing Center. There are two cloud storage generations deployed at CERN – UDS1 (Figure 1) with a storage capacity of 768TB and UDS2 (Figure1) with a storage capacity of 1.2PB. They are designed for handling large amounts of data with two main functional components – control (user frontends) and storage nodes (responsible for storing data in the hard disks). The connectivity between these components is provided with three switches and the system is accessed via two 10Gb network connections from a group of CERN-based client nodes.

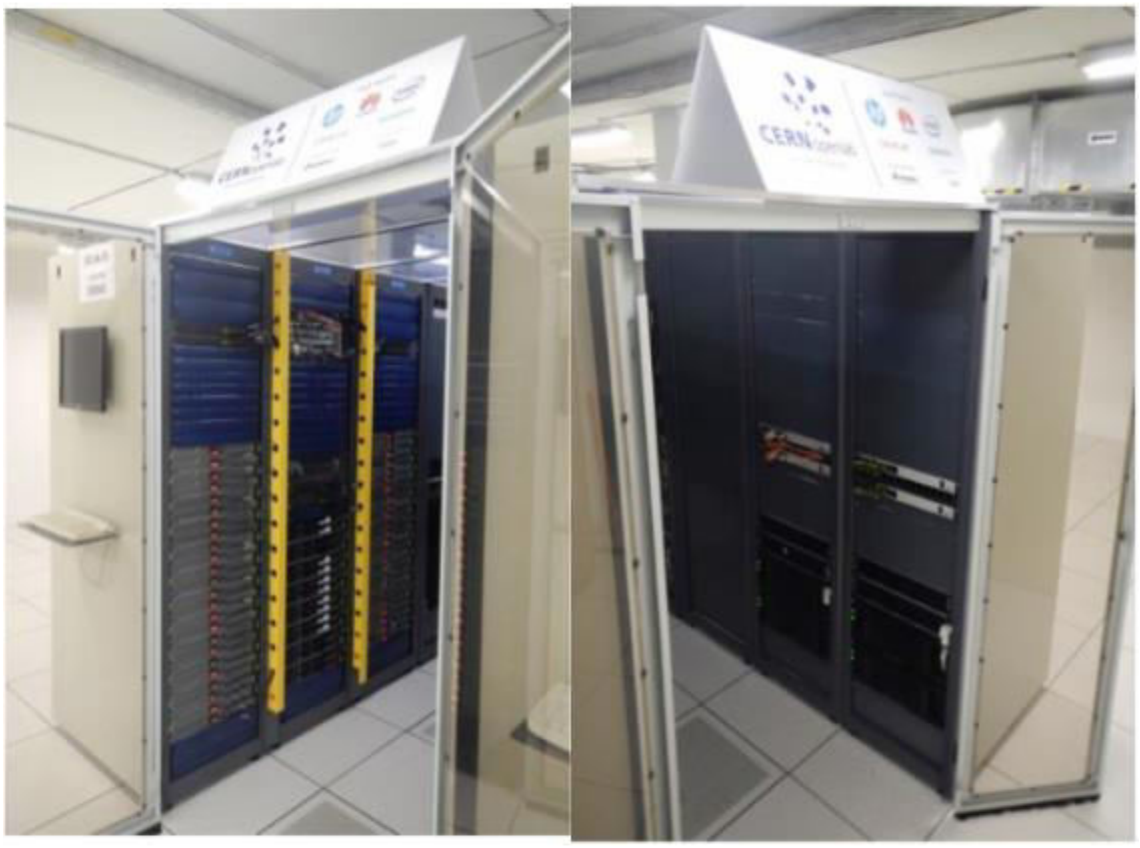


Figure 1 : Huawei UDS first (left) and second (right) generations

Huawei UDS	Storage Capacity	Control Nodes (Front Ends)	Storage Nodes (SODs)	Disk Capacity per SOD
First Generation	768TB	7	384	2TB
Second Generation	1.2PB	4	300	4TB

Table 1 : Hardware Specifications for the Huawei UDS generations

The main hardware features for both the Huawei UDS generations are shown in Table 1. Both the Huawei UDS generations use the Amazon S3 (Simple Storage Service) protocol, which is a widely adopted web API for scalable cloud storage that could also fulfil storage requirements of the high-energy physics community. S3 provides an API for making requests to the system by using HTTP methods such as GET, PUT or LIST. The S3 API of the second UDS generation supports latest S3 features such as multipart uploads, which is useful when operating with large HEP data files. The UDS2 generation has been used throughout this project.

1.2 Cloud Storage Benchmark

1.2.1 Benchmark Framework and Design

The cloud storage performance is evaluated with a S3 benchmark which was developed for this purpose at CERN. The benchmark framework is implemented in C++ and utilises the ROOT data analysis framework which is a widely used scientific data analysis package. A benchmark master process deploys and monitors many parallel client processes via ssh connections on a pool of dedicated client machines. The Amazon AWS Python library has been used for the clients to connect to the S3 server. Using the above packages a large number of benchmarks are run in different configurations to measure the aggregated throughput and rate of metadata operations. The Benchmark Framework is depicted in Figure 2.

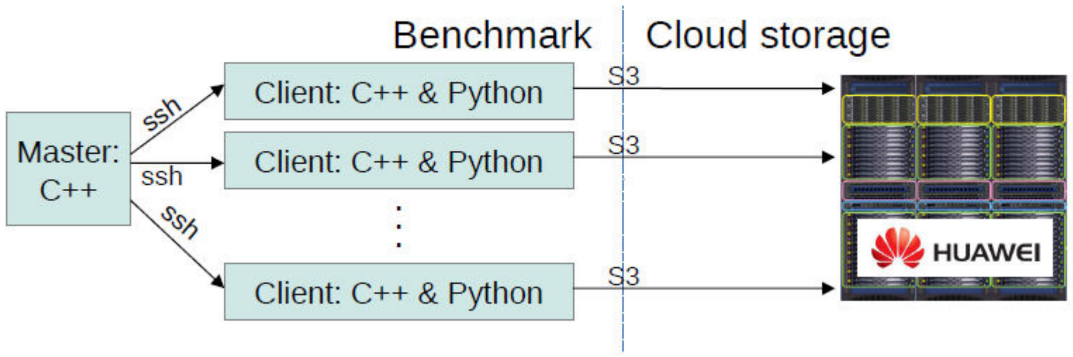


Figure 2: Software Architecture of the cloud storage benchmark framework.

1.2.2 Benchmark Deployment

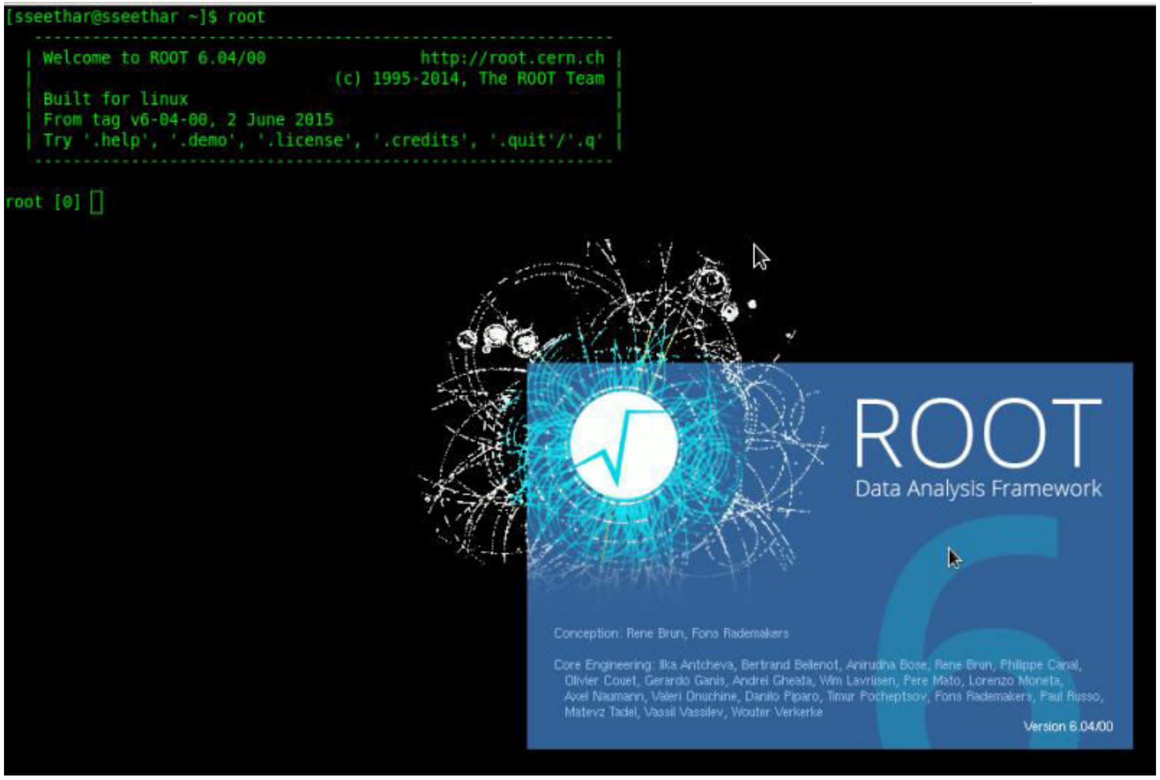
The benchmark software was hosted and distributed among all involved machines via the AFS, which is a distributed file system used at CERN. Up to 21 client machines (running Scientific Linux6) are connected via a 1Gb network link, matching the 20Gb network connectivity (two 10 Gb links) connecting to the UDS server. Each client machine is equipped with 48GB of RAM memory and 24 Intel Xeon 2.27 GHz cores.

2 Installing ROOT6

ROOT is a software that is used for data analytics by the physicists at CERN. ROOT is used mainly to provide a better analysis about the collection of data retrieved by the clients, in the used cloud benchmark framework.

A new version of ROOT (6.04.00) was released recently with new features which are interesting also for the cloud storage benchmark. The core of ROOT, which was cint before, was now replaced with cling which is a radical change to the root core. Cling is based on LLVM and CLANG libraries and provides full support for C++11/14. The Input/Output functionality is fully backward and forward compatible and ROOT6 allows support for more architectures. Cmake has been made the main build system now and a new plugin based on DAVIX is the default plugin for HTTP file access. ROOT6 also provides other new features like multi-threading, scheduling interface and easier/faster analysis. Thus on a whole, ROOT(6) has an increased overall performance, user friendliness, clarity, leverage parallelism and standardization over its previous versions.

ROOT6 was installed from source using the steps mentioned in the ROOT Website [1]. The exact commands used for installing ROOT6 are specified in Appendix1.



```
[sseethar@sseethar ~]$ root
   ------------------------------------------------------------
  | Welcome to ROOT 6.04/00                http://root.cern.ch |
  |                               (c) 1995-2014, The ROOT Team |
  | Built for linux                                            |
  | From tag v6-04-00, 2 June 2015                             |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
   ------------------------------------------------------------

root [0] 
```

ROOT
Data Analysis Framework
Conception: Rene Brun, Fons Rademakers
Core Engineering: Ilka Antcheva, Bertrand Bellenot, Anirudha Bose, Rene Brun, Philippe Canal, Olivier Couet, Gerardo Ganis, Andrei Gheata, Wim Lavrijsen, Pere Mato, Lorenzo Moneta, Axel Naumann, Valeri Onuchine, Danilo Piparo, Timur Pocheptsov, Fons Rademakers, Paul Russo, Matevz Tadel, Vassil Vassilev, Wouter Verkerke
Version 6.04/00

Figure 3 : Invoking ROOT software

3 Upgrading the Benchmark Framework

When upgrading the benchmark framework the important point was to use the right versions of the various software tools and libraries because ROOT6 required higher versions of the tools than the standard versions existing on the SCL6 installed machine. This meant that errors appeared when running the benchmark framework directly with ROOT6. This was due to the version difference and compatibility. It has been described below on how to configure the benchmark correctly on Scientific Linux 6 installed CERN machines and what versions of tools to be used.

[1] https://root.cern.ch/drupal/

The main dependencies are :

1. GCC version >=4.8 :

   ROOT6 will not compile with gcc 4.4.7 provided by the repositories. The easiest way to obtain a recent version of gcc is through the CERN's Developer Toolset repositories [1]. Here gcc was upgraded from 4.4.7 to 4.9.1 (refer Appendix 1).

2. Python version >=2.7 :

   Both Python versions (2.6.6 and 2.7.6) need to co - exist in the system. The 2.7.6 was not available in the package manager so it had to be installed from the source code. Even after installing Python2.7.6, the system continues to use python2.6 by default. So to activate Python2.7, the path to the executable of Python2.7 was used as an alias to Python (refer Appendix 1).

3. Boost version >=1.49 :

   The standard Scientific Linux machine at CERN has Boost1.41.0 . Since gcc version had to be upgraded to 4.9.1 from 4.4.7 to enable ROOT6 installation, boost also had to be upgraded because Boost1.41.0 and gcc4.9.1 were not compatible.

   Both yum and the Developer Toolsets provide an outdated version of boost, 1.41. ROOT6 requires at least 1.49. Most of the active developers use 1.55 (the default in Ubuntu 14.04. Here Boost1.55 was used. (refer Appendix 1).

4. ROOT6 :

   Basically the installation of the above software tools was important. So once they were installed and proper, ROOT6 was recompiled and installed again. The ROOT6 was configured to use Davix and xRootd libraries which help in the linking of ROOT6 with the cloud benchmark was also done. So it is better to do the ROOT6 installation towards the end after you get gcc, Python, Boost and the other dependencies in the right versions.

   The steps to install and upgrade the above software tools in the right versions are mentioned in Appendix 1 along with the results.

5. Changes in the Benchmark Framework

   The only real change that was needed to be done in the benchmark framework was to include ${PYTHON_LIBRARIES} in the cmake configuration file (CMakeLists.txt). Refer Appendix 2 for the exact line of code.

[1] http:/linux.web.cern.ch/linux/devtoolset/

Once the changes are made; the benchmark compiles and is ready to be run. Refer to Appendix 2 to get the steps used for installing the benchmark and building it along with the results.

The upgrade of the benchmark framework was successfully performed on both 32-bit and 64-bit machines. The old (Figure 4) and the new upgraded benchmark (Figure 5) setup is shown below.

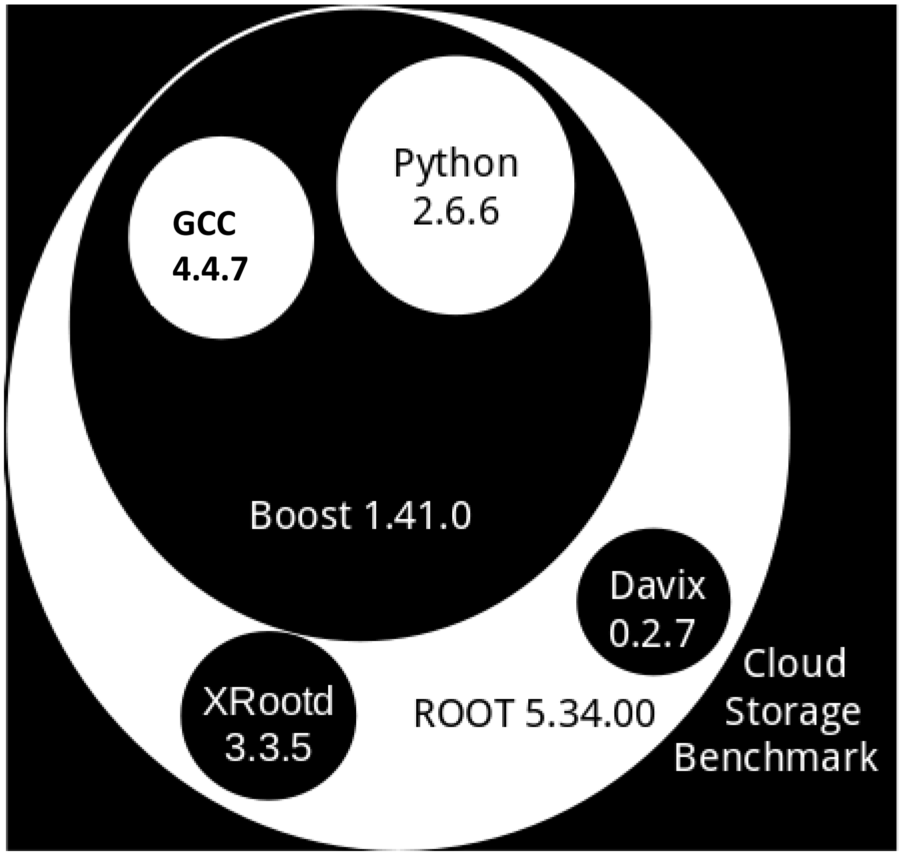


Figure 4 : Old benchmark setup

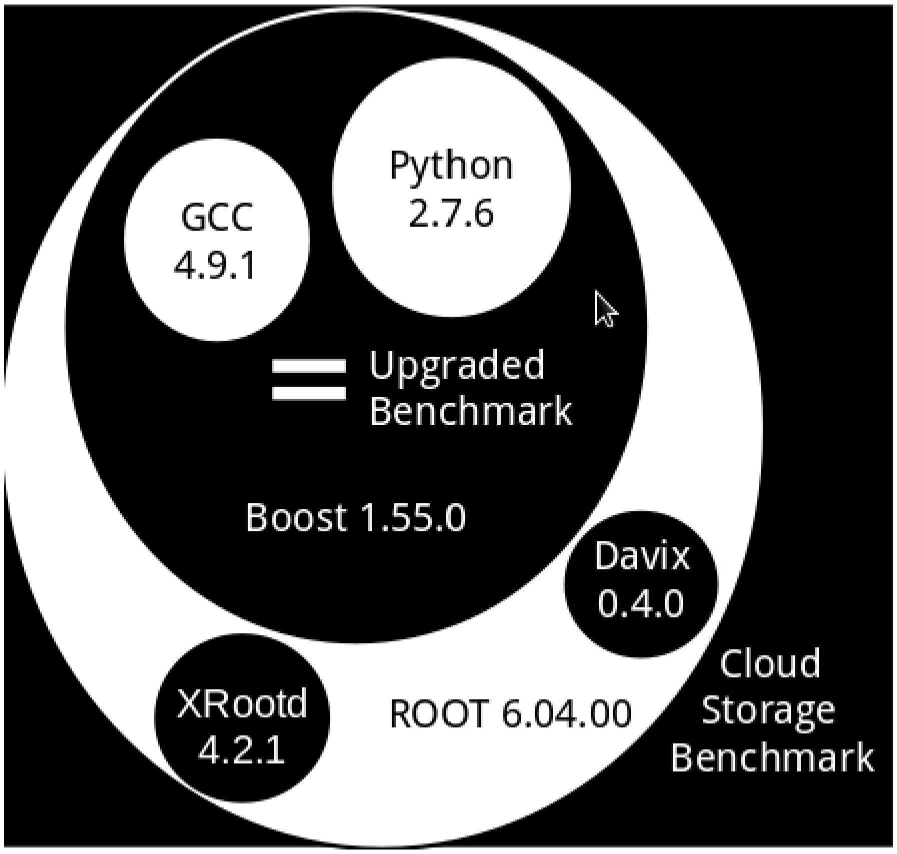


Figure 5: New benchmark setup

4 Testing on the Upgraded Benchmark

After upgrading the cloud benchmark framework, existing tests were performed on the upgraded framework.

4.1 Raw data performance tests

Raw data Performance comparison between the old benchmark and the upgraded benchmark were performed for metadata download and throughput download; to study their scalability. These tests are based on the Zotes' et. al. work, hosted and distributed among all involved machines via the AFS.

4.1.1 Metadata Download Performance

The metadata performance is evaluated using 4kB files. The environment used included 21 client machines, with maximum of 4 frontends (minimum 1), and a maximum of 81 threads (minimum 11 with stepping of 10 threads) with 3 times repetition. The result obtained included 96 points which were plotted as seen in Figure 7. It shows a speed of

3500 files per second by each frontend. The graphs obtained on testing with the old benchmark (Figure 6) and the new upgraded benchmark (Figure 7) are almost identical.

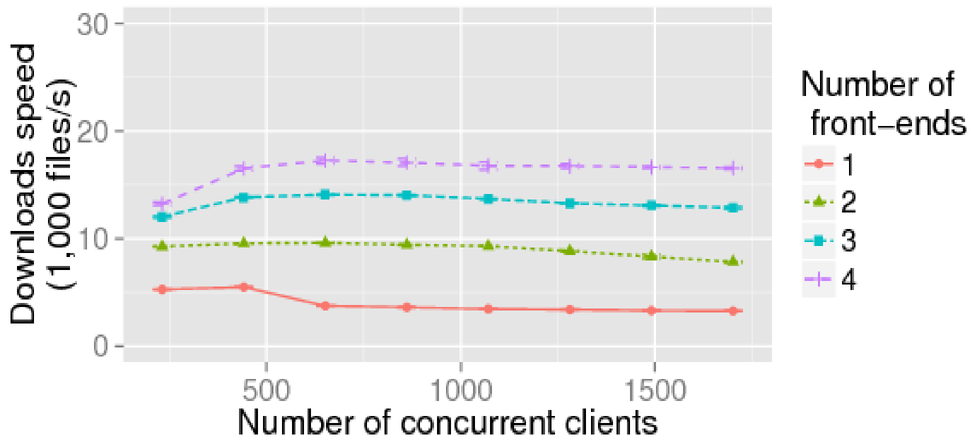


Figure 6 : Metadata performance on the old benchmark

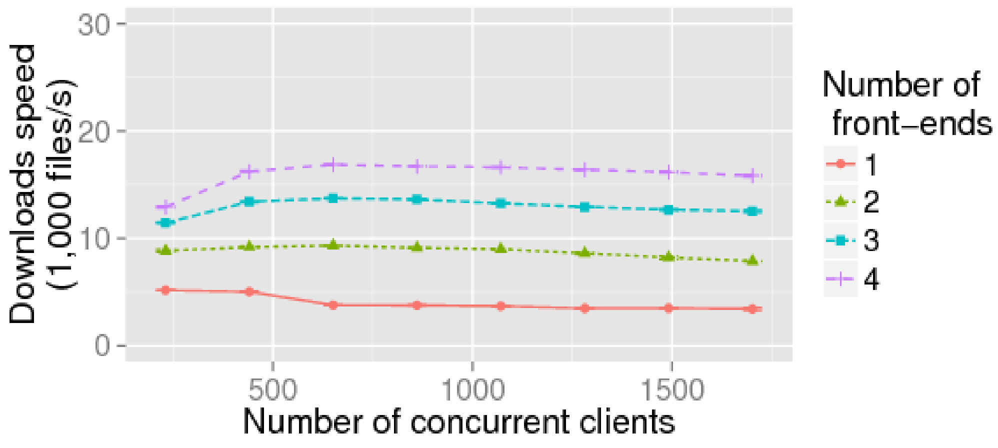


Figure 7 : Metadata performance on the new upgraded benchmark

4.1.2 Throughput Download Performance

The throughput performance is evaluated using 100MB files. The environment used included 21 client machines, with maximum of 4 frontends (minimum 1), and a maximum of 10 threads (minimum 1 with stepping of 2 threads) with 3 times repetition. The result obtained included 60 points which were plotted as seen in Figure 9. It shows a speed of almost 1GB files per second by each frontend. The graphs obtained on testing with the old benchmark (Figure 8) and the new upgraded benchmark (Figure 9) are almost identical.

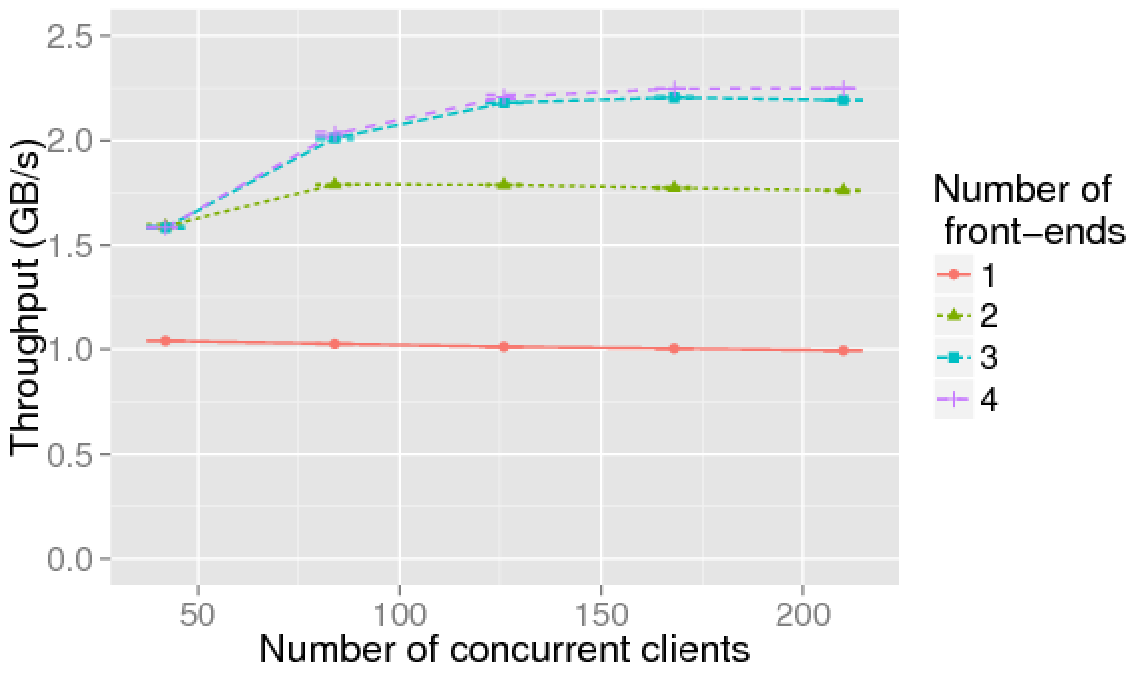


Figure 8 : Throughput performance on the old benchmark

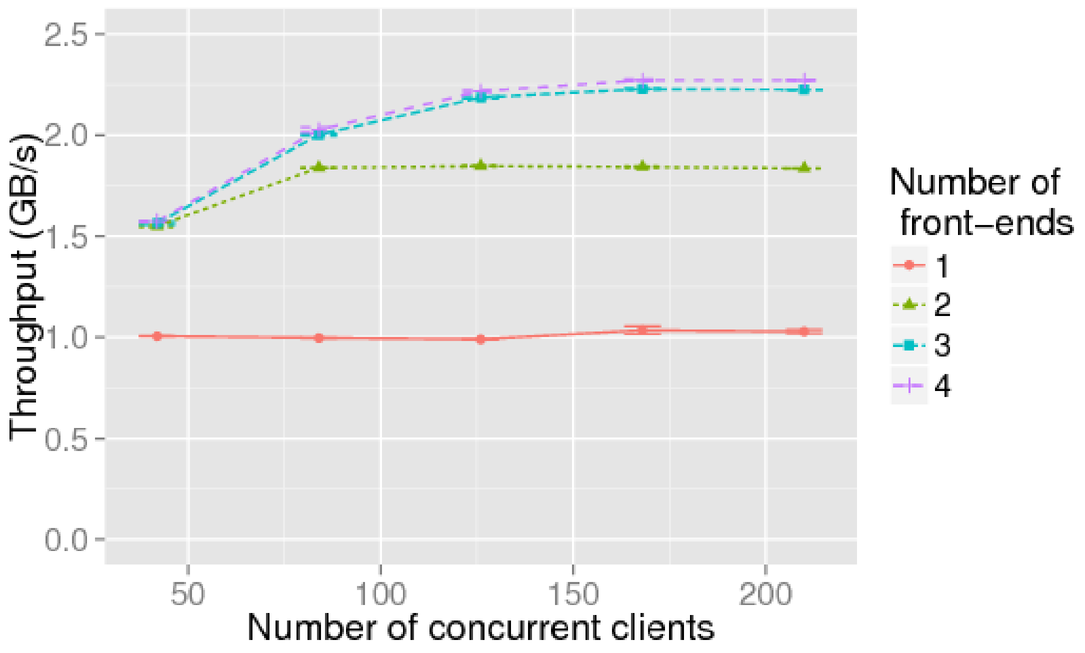


Figure 9: Throughput performance for the new benchmark

5 Conclusion and Future Work:

The CERN Openlab Huawei multi - client cloud storage benchmark has been successfully installed, configured, upgraded and made compatible with ROOT6. This allows to use the new features that are present in the latest versions of the various software tools that were used during the upgrade of the benchmark framework. Moreover the existing raw data performance tests have also been performed successfully on the new upgraded benchmark. The similar plots for the old benchmark and the upgraded benchmark verify the stability of the benchmark framework results. In future the ROOT Analysis test also needs to be performed on the upgraded benchmark which is the last among the existing tests that are needed to be done. Also this benchmark framework that was upgraded is generic in nature and hence this framework can be used to test the scalability of other storage technologies at CERN like the kinetic drives or EOS.

6 References

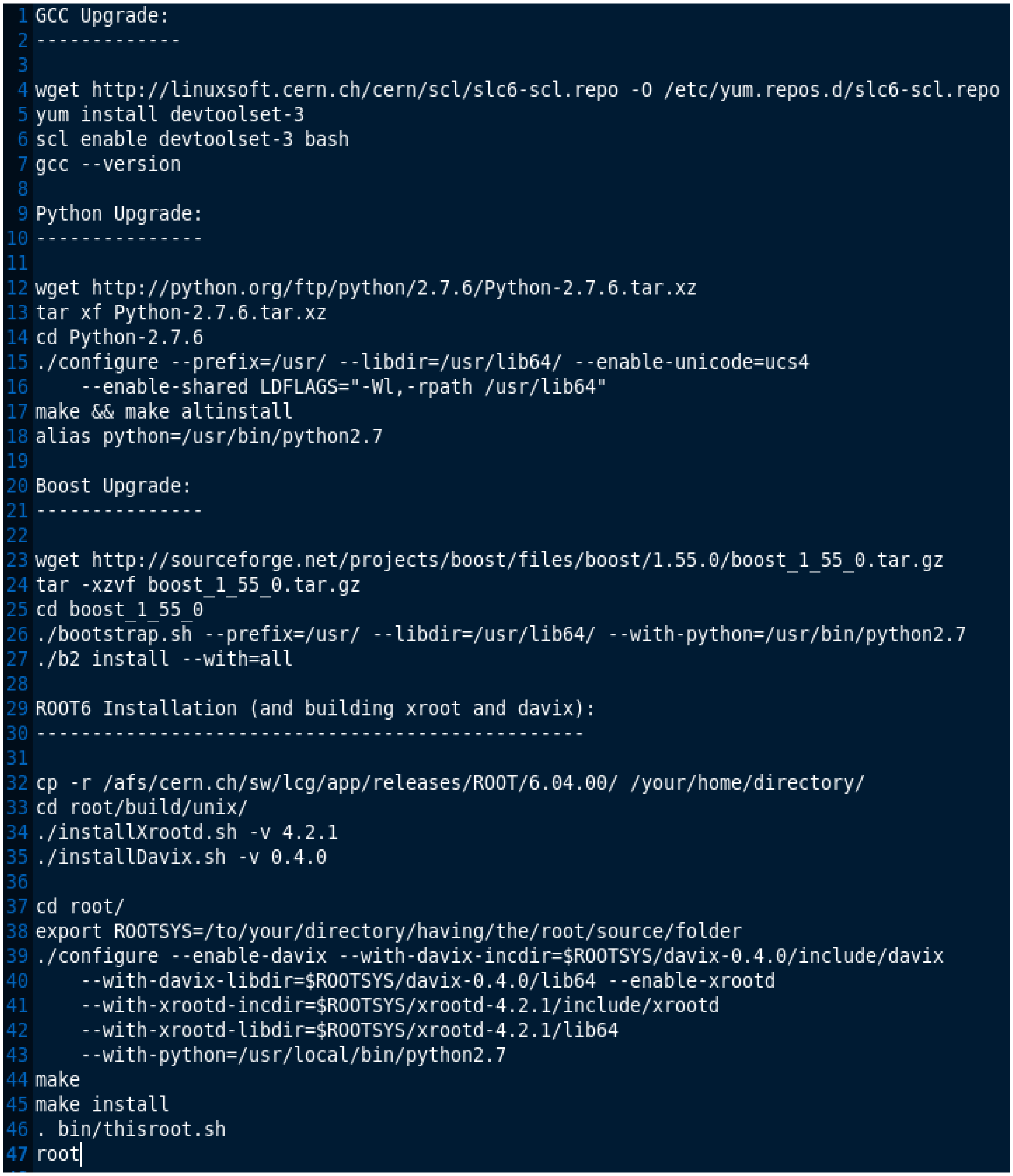
[1] Rios Arsuaga M., Heikkila S S., Duellman D., Meusel R., Blomer J., Couturier B. ‘‘Using S3 Cloud Storage with ROOT and CvmFS’’ , CHEP 2015, Japan.

[2] Zotes Resines M., Heikkila S S., Duellman D., Adde G., Toebbicke R., Huges J. & Wang L. ‘‘ Evaluation of the Huawei UDS cloud storage system for CERN specific data’’, Journal of Physics : Conference Series Vol.513(4), 2014.

[3] Bahaa V., ‘‘Development of Monitoring and Analysis Tools for the Huawei Cloud Storage ’’ CERN Openlab Summer Student Report, 2014.

7 Appendix 1

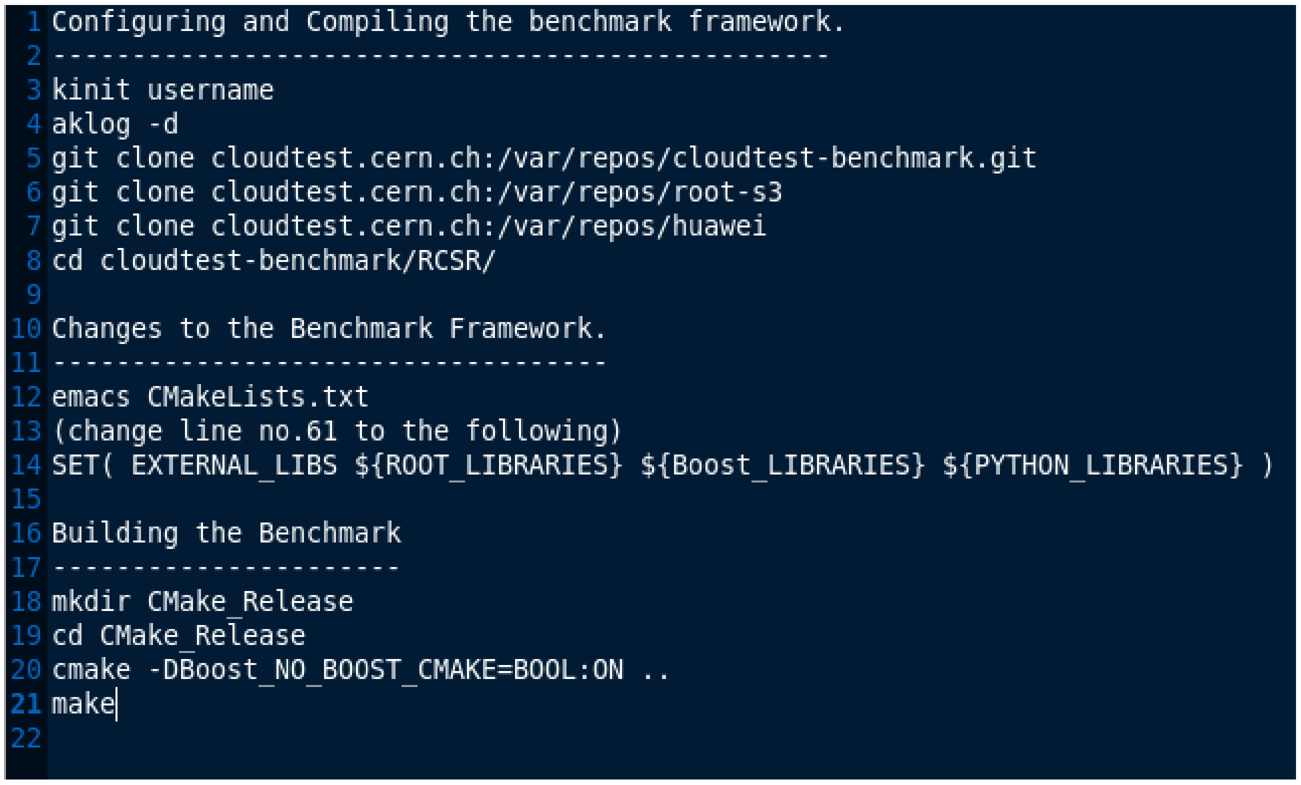
This appendix specifies the command line code for upgrading all the tools required to upgrade the cloud storage benchmark framework.



```
GCC Upgrade:
-------------

wget http://linuxsoft.cern.ch/cern/scl/slc6-scl.repo -O /etc/yum.repos.d/slc6-scl.repo
yum install devtoolset-3
scl enable devtoolset-3 bash
gcc --version

Python Upgrade:
---------------

wget http://python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
tar xf Python-2.7.6.tar.xz
cd Python-2.7.6
./configure --prefix=/usr/ --libdir=/usr/lib64/ --enable-unicode=ucs4
    --enable-shared LDFLAGS="-Wl,-rpath /usr/lib64"
make && make altinstall
alias python=/usr/bin/python2.7

Boost Upgrade:
---------------

wget http://sourceforge.net/projects/boost/files/boost/1.55.0/boost_1_55_0.tar.gz
tar -xzvf boost_1_55_0.tar.gz
cd boost_1_55_0
./bootstrap.sh --prefix=/usr/ --libdir=/usr/lib64/ --with-python=/usr/bin/python2.7
./b2 install --with=all

ROOT6 Installation (and building xroot and davix):
--------------------------------------------------

cp -r /afs/cern.ch/sw/lcg/app/releases/ROOT/6.04.00/ /your/home/directory/
cd root/build/unix/
./installXrootd.sh -v 4.2.1
./installDavix.sh -v 0.4.0

cd root/
export ROOTSYS=/to/your/directory/having/the/root/source/folder
./configure --enable-davix --with-davix-incdir=$ROOTSYS/davix-0.4.0/include/davix
    --with-davix-libdir=$ROOTSYS/davix-0.4.0/lib64 --enable-xrootd
    --with-xrootd-incdir=$ROOTSYS/xrootd-4.2.1/include/xrootd
    --with-xrootd-libdir=$ROOTSYS/xrootd-4.2.1/lib64
    --with-python=/usr/local/bin/python2.7
make
make install
. bin/thisroot.sh
root
```

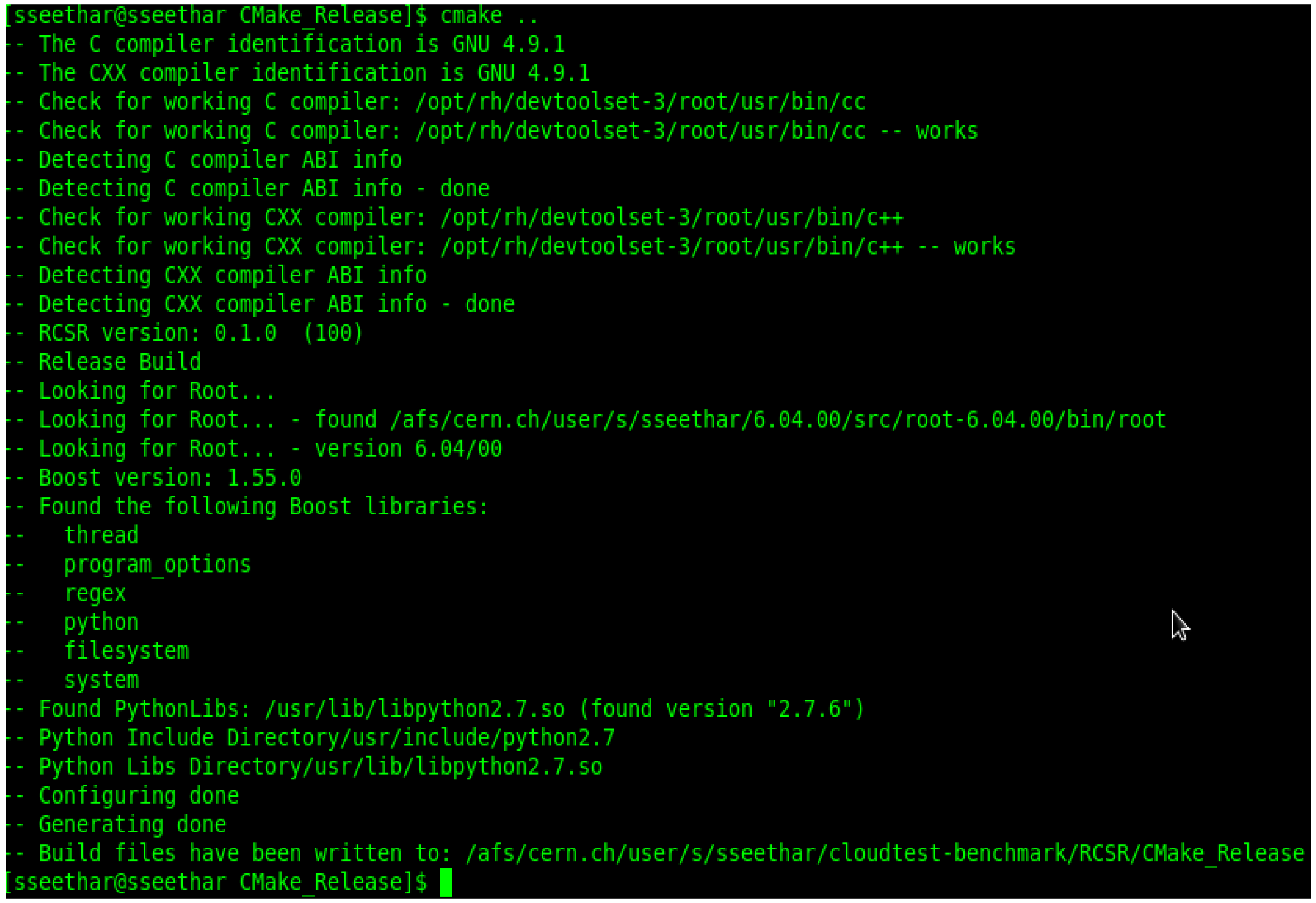
8 Appendix 2

This appendix specifies the command line code for compiling and upgrading the benchmark as well as the change that has to made in the benchmark framework.



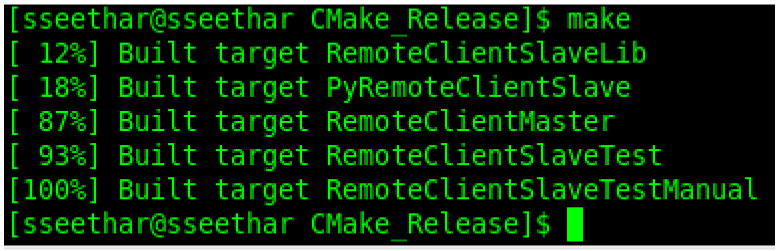
```
Configuring and Compiling the benchmark framework.
-------------------------------------------------
kinit username
aklog -d
git clone cloudtest.cern.ch:/var/repos/cloudtest-benchmark.git
git clone cloudtest.cern.ch:/var/repos/root-s3
git clone cloudtest.cern.ch:/var/repos/huawei
cd cloudtest-benchmark/RCSR/

Changes to the Benchmark Framework.
-----------------------------------
emacs CMakeLists.txt
(change line no.61 to the following)
SET( EXTERNAL_LIBS ${ROOT_LIBRARIES} ${Boost_LIBRARIES} ${PYTHON_LIBRARIES} )

Building the Benchmark
----------------------
mkdir CMake_Release
cd CMake_Release
cmake -DBoost_NO_BOOST_CMAKE=BOOL:ON ..
make
```

The following shows the output of the compilation and building of the benchmark framework.



```
[sseethar@sseethar CMake_Release]$ cmake ..
-- The C compiler identification is GNU 4.9.1
-- The CXX compiler identification is GNU 4.9.1
-- Check for working C compiler: /opt/rh/devtoolset-3/root/usr/bin/cc
-- Check for working C compiler: /opt/rh/devtoolset-3/root/usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /opt/rh/devtoolset-3/root/usr/bin/c++
-- Check for working CXX compiler: /opt/rh/devtoolset-3/root/usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- RCSR version: 0.1.0  (100)
-- Release Build
-- Looking for Root...
-- Looking for Root... - found /afs/cern.ch/user/s/sseethar/6.04.00/src/root-6.04.00/bin/root
-- Looking for Root... - version 6.04/00
-- Boost version: 1.55.0
-- Found the following Boost libraries:
--   thread
--   program_options
--   regex
--   python
--   filesystem
--   system
-- Found PythonLibs: /usr/lib/libpython2.7.so (found version "2.7.6")
-- Python Include Directory/usr/include/python2.7
-- Python Libs Directory/usr/lib/libpython2.7.so
-- Configuring done
-- Generating done
-- Build files have been written to: /afs/cern.ch/user/s/sseethar/cloudtest-benchmark/RCSR/CMake_Release
[sseethar@sseethar CMake_Release]$
```

Appendix 2.1 : Benchmark Compilation.



```
[sseethar@sseethar CMake_Release]$ make
[ 12%] Built target RemoteClientSlaveLib
[ 18%] Built target PyRemoteClientSlave
[ 87%] Built target RemoteClientMaster
[ 93%] Built target RemoteClientSlaveTest
[100%] Built target RemoteClientSlaveTestManual
[sseethar@sseethar CMake_Release]$
```

Appendix 2.2 : Benchmark Building