IMPLEMENTATION AND ANALYSIS OF XCACHE ON SOUTHWEST TIER 2 CLOUD CLUSTER FOR LARGE HADRON COLLIDER ATLAS EXPERIMENT

by

PRIYAM BANERJEE

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington

in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

MAY 2019

Acknowledgements

I would like to extend my heartfelt thanks to my supervisor, Professor David Levine, who inspired and motivated me to work on this thesis. This thesis would not have been possible without his encouragement and support. I extend my sincere gratitude to Prof. Levine for his invaluable support and supervision.

A big thanks to Mr. Patrick McGuigan from the Dept. of Physics, University of Texas at Arlington. Patrick has been an exceptional guide who left no stone unturned to explain me the concepts of High Throughput Computing. In short, this research could not have seen the light of the day without Patrick's constant support.

Sincerest gratitude to my committee member Prof. Ramez Elmasri, and the Department of Computer Science and Engineering, University of Texas Arlington, for their invaluable cooperation and support.

Special thanks to Dr. Kaushik De (Dept. of Physics, University of Texas at Arlington), Mr. Andrew Hanushevsky and Mr. Wei Yang (SLAC National Accelerator Laboratory).

Last but not the least, I would like to thank my Grandma (Dinda) and Late Grandpa (Dadu) who have always encouraged me to strive for excellence and perfection.

March 5, 2019

To Maa and Moni...

Abstract

IMPLEMENTATION AND ANALYSIS OF XCACHE ON SOUTHWEST TIER 2 CLOUD CLUSTER FOR LARGE HADRON COLLIDER ATLAS EXPERIMENT

Priyam Banerjee, MS

The University of Texas at Arlington, 2019

Supervising Professor : David Levine
Committee Members : Ramez Elmasri, Patrick McGuigan (Dept. of Physics)

The ATLAS Experiment is one of the four major particle-detector experiments at the Large Hadron Collider at CERN (birthplace of the World Wide Web). The ATLAS was one of the LHC experiments that successfully demonstrated the discovery of the Higgs-Boson in July 2012. At the end of 2018, CERN data archiving on tape-based drives reached 330 PB. Through the Worldwide LHC Computing Grid (WLCG), a distributed computing infrastructure, the calibrated data out of the particle accelerator is split in chunks and distributed all around the world for analysis. The WLCG runs more than two million jobs per day. At peak rate of data capturing, 10 GB of data is transferred per second and might require immediate storage.
The workflow management system known as PanDA (for Production and Distributed Analysis) handles the data analysis jobs for LHC's ATLAS Experiment. The University of Texas Arlington hosts two compute and storage data-centers together known as the SouthWest Tier II to process the ATLAS

data. SouthWest Tier II (SWT2) is a consortium between The University of Texas at Arlington and Oklahoma University. This thesis focuses on finding an efficient way to compensate and optimize the available hardware specification(s) with a caching mechanism.

The Caching mechanism (called Xcache), which uses XROOTD system and ROOT Protocol is installed at one of the machines of the cluster. The machine acts as a File Caching Proxy Server (with respect to the network) which redirects incoming client requests for data files over to the Redirector at CPB (Chemistry-Physics-Building, UT Arlington), thereby, acting as a direct mode proxy. In this process, the machine caches data files into its storage space (106 TB), and can be reused by Caching (Disk Caching Storage). This research focuses on the adoption of Xcache into the cluster and finding the network dependencies, performance parameters of the cache (Cache Rotation using High and Low Watermark, Bytes Input and Bytes Output for monitoring the network). Therefore, a proxy caching mechanism (Xcache) used to address bandwidth and access-latency (reduced network traffic) is also used to optimize the storage servers.

Table of Contents

List of Illustrations

Chapter 1 : Introduction

About CERN

The European Organization for Nuclear Research, known as CERN is a European Research Organization that has built and maintains the largest particle physics laboratory in the world. The CERN convention was signed in 1953 by 12 founding states, and became effective from September 29, 1954. As of today, CERN has 22 Member States and 6 Organizations having Observer Status. The United States of America has currently Observer Status. Over 600 Institutes and Universities contribute to CERN, and funding from both Member and Non-member States goes into the construction, operation, maintenance and development of the CERN.

CERN operates the Large Hadron Collider (LHC) [2] and the experiments associated with it. The LHC is a particle accelerator located about 175 meters underground in a tunnel with about 17 miles (about 27 kms) in circumference, and is responsible for colliding protons or heavy ions. LHC is the world’s largest and highest-energy particle accelerator. Seven particle detectors (CMS, ATLAS, LHCb, MoEDAL, TOTEM, LHC-forward and ALICE) have been constructed to study the particle collisions from different aspects using different technologies.

Detector	Description
ATLAS [10]	One of two general-purpose detectors. ATLAS studies the Higgs Boson [6] and looks for signs of new physics, including the origins of mass and extra dimensions.
CMS [10]	The other general-purpose detector, like ATLAS, studies the Higgs boson [6] and look for clues of new physics.
ALICE [10]	ALICE is studying a "fluid" form of matter called quark-gluon plasma [8] that existed shortly after the Big Bang.
LHCb [10]	Equal amounts of matter and antimatter [9] were created in the Big Bang. LHCb investigates what happened to the "missing" antimatter [9]

Scientific Achievements at CERN

- 1983 : The discovery of W and Z bosons in the UA1 and UA2 [3] Experiments
- 1995 : The first creation of antihydrogen [4] atoms in the PS210 Experiment
- 1999 : The discovery of direct CP (Charge-Parity) violation in the neutral kaon system [5] with NA48 Experiment [5]
- 2010 : The isolation of 38 atoms of antihydrogen [4]
- 2011 : Maintaining antihydrogen [4] for over 15 minutes
- 2012 : Higgs Boson [6] experimentally demonstrated

Nobel Prize and CERN : A close association

- 1984 Nobel Prize for Physics : Awarded to Carlo Rubbia and Simon van der Meer for the discovery of W and Z bosons [7]
- 1992 Nobel Prize for Physics : Awarded to Georges Charpak for his contribution to *Multiwire Proportional Chamber*. [7]
- 2013 Nobel Prize for Physics : Awarded to Peter Higgs and Francois Englert for their theoretical description of Higgs mechanism [7]

Storage and Computing at CERN

The distributed storage system for LHC exceeded 200 PB of raw storage with 600 million files. The system is a disk-based open-source developed by CERN for High-Energy Physics computing requirements. By the end of 2018, the LHC reached a proton-proton collision of 13 TeV (highest ever reached by a particle accelerator up until now), thereby completing it’s Run 2. [11]

During Run 2, from 2015 to 2018, LHC experiments produced colossal amount of data which meant extensive use of computing. The CERN Advanced Storage System (CASTOR) [12] archived permanent data of 330 PB on tape-based drive, which is equivalent to 2000 years of 24x7 HD video storing. [11]

The data center at CERN processes 1 PB of data everyday, where the LHC data is gathered and initial processing takes place. The data center hosts around 10,000 servers, with 90,000 processor cores, and 6000 changes are performed in the database every second [13]. Additional computing power is provided by a remote data center extension at Wigner Research Center for Physics at Hungary. Disks (both SSD and HDD) are used for short-term storage, while magnetic tapes retrieved by robots are used for archiving and long-term storage.

Two high speed links (100 Gbps) links CERN data center to its extension in Hungary. CERN data center is at the heart of Worldwide LHC Computing Grid (WLCG), enabling thousands of physicists to collaborate and share their results.

The Large Hadron Collider

The Large Hadron Collider (LHC) is the most powerful particle accelerator ever built. Located about 100 meters underground on the Franco-Swiss border near Geneva, Switzerland. ‘Large’ because it’s about 27 kms in circumference, ‘Hadron’ because it accelerates protons or ions belonging to the group of particles called ‘Hadrons’, and ‘Collider’ because the particles forming two beams are made to collide at four points around the machine.

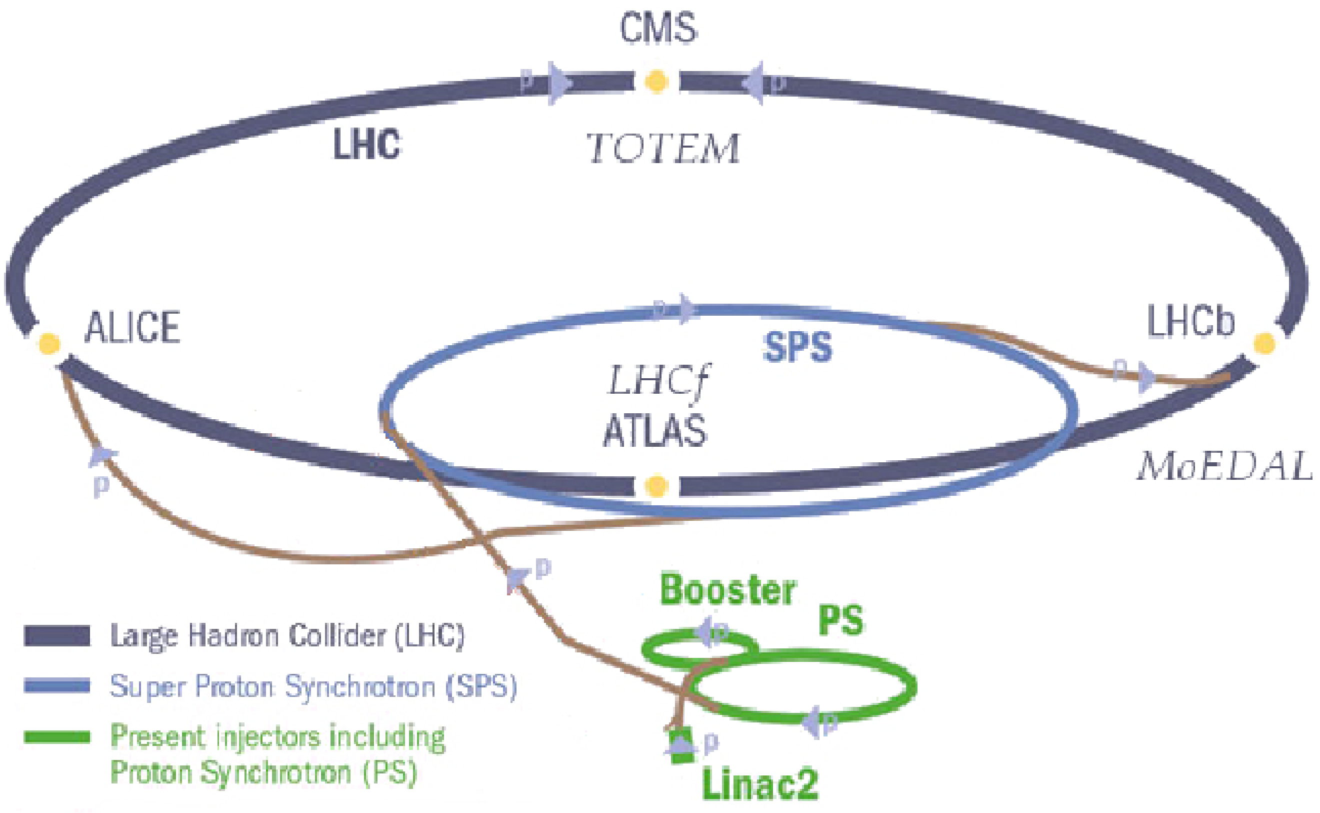


Illustration 1: High Energy Experiments at CERN [27]

Working of the Large Hadron Collider

The accelerator complex is a series of machines with increasing energy. Each machine accelerates the beam of particles to a particular energy before injecting it to the next machine in the chain, and so on. LHC is the last element in this chain where the beams reach their highest energies. The beams travel in opposite directions in respective beam pipes, kept at ultrahigh vacuum [15]. The two particle beams travel at the speed of light before they are made to collide.

Superconducting electromagnets maintaining strong magnetic field guides the particle beams around the accelerator ring. The electromagnets at LHC are cooled to -271.3 degree Celsius [14] (1.9 K) (superconducting state offering no resistance to passage of current). A vast quantity of liquid Helium keeps the magnets cool down to 1.9 K (a temperature cooler than outer space).

In 2012, running at 4 TeV, LHC consumed a maximum power of about 650 GWh. Total CERN energy consumption is about 1.3 TWh per year.

The first run (Run 1) lead to the discovery of the Higgs-Boson in 2012. Run 2 started in 2015 with proton-proton collision reaching energy of 13 TeV – the highest ever reached by a particle accelerator. By 2015 end, the beam intensity increased to about 2240 proton-bunch being collided per beam. Run 2 saw more than one billion collisions per second in ATLAS and CMS Experiments. Run 2 also produced twice the data as compared to Run 1, exceeding the anticipated 50 PB per year.

Calorimeters [16] are used to measure the energy loss of particles as they pass through. Calorimeter stops or absorbs most of the particles coming from collision, thereby causing them to deposit their energy in the detector. Tracking devices record electrical signals that particles trigger as they move through the device. Special tracking devices for detecting muons are placed at the outermost layer of the detector, because muons travel through meters of dense materials before it can be stopped. Collating the data from different parts of the detectors, physicists analyze the snapshot at the time of collision, trying to find out any unusual particle(s), or trend that doesn't follow the current theories.

Achievements of LHC so far :

[14]

- September 10, 2008 : LHC first beam
- November 23, 2009 : LHC first collisions
- March 2010 : First high energy collisions at 7 TeV
- April 5, 2012 : First collisions at 8 TeV
- July 4, 2012 : Announcement of Higgs-Boson experimentally discovered

Computing and storage upgrades during Long Shutdown 1 for Run 2

CERN private cloud [17] was built to provide a remote efficient management system to monitor the increased computing capacity for Run 2 (2015-2018). In order to use a scalable cloud operating system, CERN IT started using

OpenStack. Multiple OpenStack clouds allow successful simulation and analysis run for the user community. The compute capacity of CERN private cloud has reached about 2 million job slots. The control applications of LHC migrated from file-based archiver to centralized infrastructure using Oracle databases.

In order to optimize computing and storage resources, Run 2 has adopted new computing models. Moving away from the WLCG [ ] model, LHC has adopted a peer site model, making more use of the compute of its peer sites. The LHC Optical Private Network (LHCOPN) [18] and LHC Open Network Environment (LHCONE) [18] serves the networking requirements for the new computing model. The bandwidth connecting Tier 0 and Tier 1 [ ] sites have increased from 10Gbps to 20 and 40 Gbps [17]. Trans-atlantic connections improved with ESnet setting up three 100 Gbps links to CERN replacing the five 10 Gbps links [17].

During the second run (2015-2018), LHC successfully performed 16 million billion proton-proton collisions at energy of 13 TeV and lead-lead collisions at about 5.02 TeV. LHC is undergoing improvements and upgrades, and the proton beams will resume in the third run in Spring 2021.

High Luminosity Large Hadron Collider (HL-LHC)

The High Luminosity Large Hadron Collider project aims to rev up the performance of LHC after 2025. The objective is to increase LHC’s luminosity by a factor of 10 beyond its design value. Luminosity is proportional to the amount of collisions happening in a specific time period. The higher luminosity would cause higher data for analysis to observe rare processes. HL-LHC is expected to produce at least 15 million Higgs-bosons per year, compared to 3 million produced by the end of 2017 [19]. The project is lead by CERN with International collaborators from 29 institutions in 13 countries.

The Proton Synchrotron (PS) (CERN's first synchrotron) [20] responsible for accelerating protons to about 26 GeV before sending them to Super Proton Synchrotron (SPS) [21] and also to Antiproton Decelerator (AD) [22], will undergo a major overhaul for the higher beam intensity and injection for LHC's Run 3. PS has a total of 100 main magnets within (for bending and focusing the beam), out of which 43 magnets are scheduled to get refurbished in 2019 as part of HL-LHC upgrade. Proton Synchrotron 2 (PS2) is being upgraded to facilitate acceleration of beam from 5 GeV at injection to 50 GeV at extraction. [23]

HL-LHC requires technology upgrade for the magnets for the innovative crab cavities which steers the beam to maximize rate of collision. The new magnets would be using niobium-tin wire, instead of the present niobium-titanium wire, thereby able to withstand higher magnetic field. For the current transfer lines carrying current into the HL-LHC magnets, magnesium diboride superconductors are going to be used. Magnesium diboride is currently under test for HL-LHC suitability.

Plans for ATLAS in the next two years during Long Shutdown 2

Maintenance and upgrade of the detector is underway for the next two years of technical break, called Long Shutdown 2 (LS2). The HL-LHC project is set to run from 2026. A major improvement is the installation of two wheel-shaped muon particle-detectors, and spectrometers. These new wheels are expected to track higher muon rates expected from HL-LHC. Each wheel consists of sectors with detection chambers called micromegas (MM) and small strip thin gas chambers (sTGC). Both the devices have excellent precision tracking capabilities (up to 100 micrometers) and better response time. [24]

Sixteen new stations are linked to improve the ATLAS muon detection capability in the region between the barrel and the endcaps. The stations contain gas-filled small monitored drift tubes (SMDT) and resistive plate chambers (RPC). Physicists would study the trail of electrically charged particles caused by muons passing through the gas. [24]

Replacement of the front-end electronics of Liquid Argon Calorimeter (LAr) is another major upgrade taking place. The triggering and data-acquisition system in undergoing an upgrade too. After LS2, LHC restarts in 2021 [24].

Chapter 2 : The ATLAS Experiment

ATLAS is one of the four major particle detector experiments constructed at the Large Hadron Collider (LHC), a particle accelerator at CERN, Switzerland. It is a general-purpose particle physics experiment run by an International collaboration,together with CMS [10], and is designed to uncover the full potential of the physics opportunities that the LHC provides. ATLAS can be over-simply termed as ‘A Toroidal LHC Apparatus’. ATLAS was one of the two LHC experiments involved in the discovery of Higgs Boson in July 2012.

ATLAS’ scientific exploration is seeking the fundamental questions of the building blocks of matter, the fundamental forces of nature, and the underlying symmetry to our universe.

ATLAS physicists test the predictions of the Standard Model [25], which currently explains our understanding of the building blocks of matter and how they interact with one another. These tests have established discoveries (like the Higgs-boson) and the physics beyond the Standard Model. It can also lead to new theories for better describing our universe.

ATLAS comprises about 3000 scientific authors from about 183 Institutions around the world, representing 38 countries. Around 1200 doctoral students are involved in detector development, data collection and analysis.

ATLAS Detector

ATLAS detector is the largest volume detector ever made, having the dimensions of a cylinder, 46 m long and 25 m in diameter, and is placed 100 m below the ground. The detector weighs about 7000 tonnes (roughly the weight of Eiffel tower). It is a multi-layered instrument designed to detect the tiniest sub-atomic particles. It consists of six different detecting sub-systems around the collision point wrapping concentrically in layers. The purpose of the detector sub-systems are to find the trajectory, momentum and energy of particles, allowing them to be individually identified and measured. A huge torroidal magnet bends the path of the charged particles, so as to allow their momenta to be measured accurately.

Beams of particles traveling at 7 TeV from the LHC collides at the center of the detector, thereby leaving collision debris of new sub-particles which fly out in all directions. From over a billion particle collisons happening in the detector every second, only one in a million are flagged as potentially interesting and recorded as snapshot for further study.

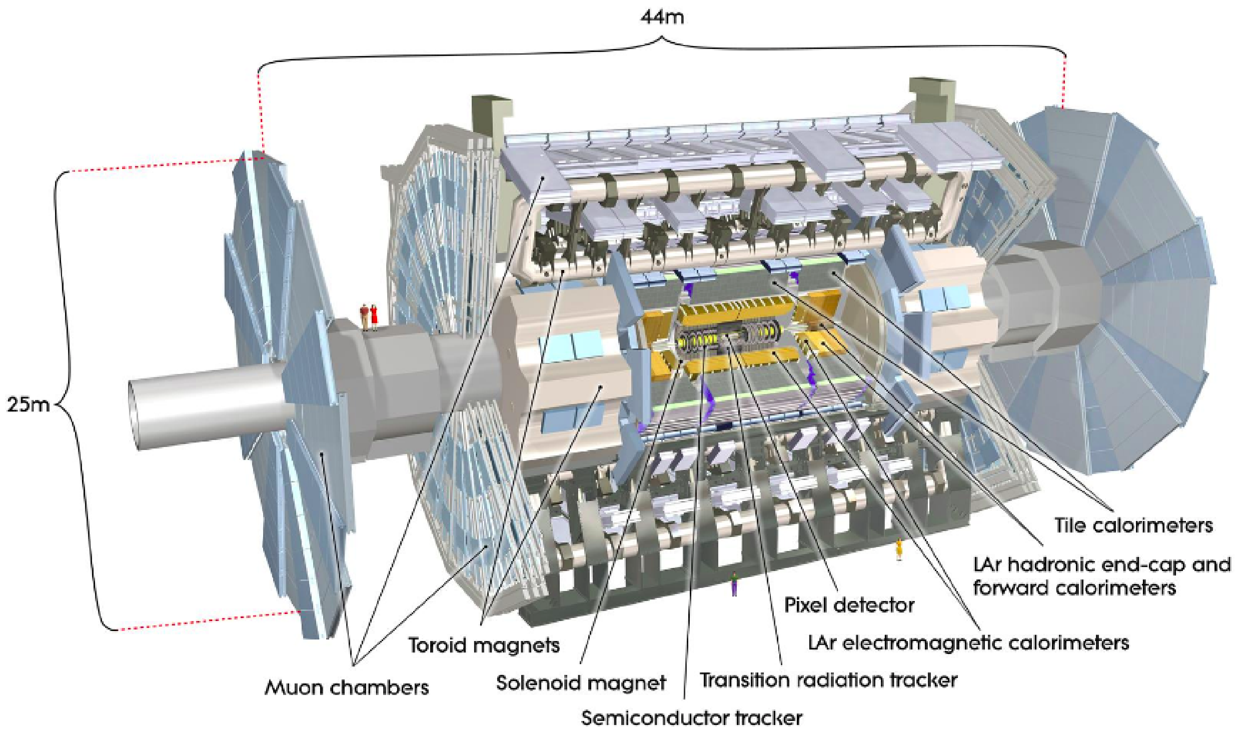

44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

Illustration 2: Atlas Detector [26]

The detector tracks and identifies particles ranging from the study of Higgs-boson, to quarks and the study of extra dimensions and particles that could potentially make up dark matter. The four major components of the ATLAS Detector are :

1. Inner Detector
2. Calorimeter
3. Muon Spectrometer
4. Magnet System

Other components include :

1. Tile Calorimeters (Intermediate Tile Calorimeter constructed at University of Texas at Arlington)

2. LA-r Hadronic end-cap and forward calorimeters
3. Toroidal and Solenoid Magnets
4. Muon Chambers
5. Trackers include Semiconductor tracker and Radiation tracker

Inner Detector

The inner detector is the first highly compact and sensitive part of ATLAS to observe the decay of the collisions. It consists of three different types of sensors in a magnetic field parallel to the beam axis. This detector's purpose is to measure the direction, momentum and charge of electrically-charged particles produced by proton-proton collision.

The main components include :

1. Pixel Detector
2. Semiconductor Tracker (SCT)
3. Transition Radiation Tracker (TRT)

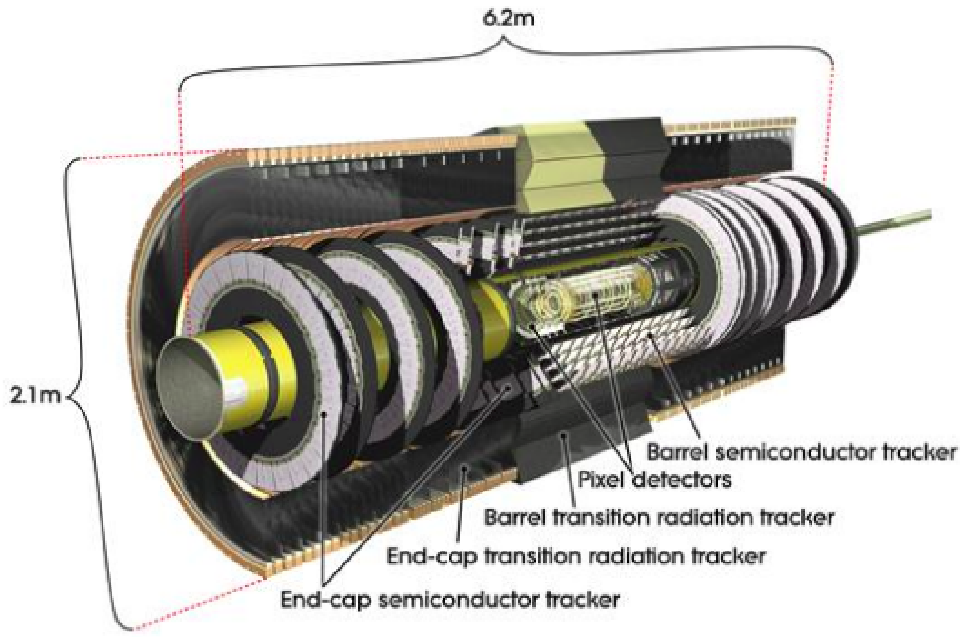

6.2m
2.1m
Barrel semiconductor tracker
Pixel detectors
Barrel transition radiation tracker
End-cap transition radiation tracker
End-cap semiconductor tracker

Illustration 3: Model of the ATLAS Inner Detector [28]

Pixel Detector Specifications [28]

- 80 million pixels (80 million channels). Area 1.7m2. 15 kW power consumption
- Barrel has 1,744 modules (10cm2) with 46,080 readout channels per module
- Pixel size 50 x 400µm2 . Resolution 14 x 115µm2
- Three Pixel disks (in each endcap) have 6.6 million channels
- 3 barrel layers: 1,456 modules (i.e. 67M)
- 3 disks in each end-cap: 288 modules (i.e. 13M)

Semiconductor Tracker Specifications [28]

- A silicon microstrip tracker consisting of 4,088 two-sided modules and over 6 million implanted readout strips (6 million channels)
- 60m2 of silicon distributed over 4 cylindrical barrel layers and 18 planar endcap discs

- Readout strips every 80µm on the silicon, allowing the positions of charged particles to be recorded to an accuracy of 17µm per layer (in the direction transverse to the strips)

Transition Radiation Tracker Specifications [28]

- 350,000 read-out channels
- Volume 12m3
- Basic detector element: straw tube with 4mm diameter, in the centre a 0.03mm diameter gold-plated tungsten wire
- 50,000 straws in Barrel, each straw 144 cm long. The ends of a straw are read out separately
- 250,000 straws in both endcaps, each straw 39 cm long
- Precision measurement of 0.17 mm (particle track to wire)
- Provides additional information on the particle type that flew through the detector, i.e. if it is an electron or pion

Calorimeter

A calorimeter is used to measure the energy a particle loses as it passes through the detector. It is used to stop entirely or absorb most of the particles coming out of a collision, forcing them to deposit all their energy within the detector.

Calorimeters are made up of ‘passive’ or ‘absorbing’ high-density materials, like lead, along with layers of lead-glass or liquid argon acting as ‘active medium’. Electromagnetic calorimeters measure the energy of electrons and protons, while Hadronic calorimeters sample the energy of hadrons (hadrons are the particles containing quarks – like proton and neutron). Calorimeters are known to stop most particles except for muons and neutrinos.

In broad sense, there are two types of calorimeters used in ATLAS Experiment :
1. Liquid Argon (LAr) Calorimeter
2. Tile Hadronic Calorimeter

Tile Calorimeter (TileCal) Specifications [29]

• Central barrel made of 64 wedges, each 5.6m long and weighing 20,000 kg
• Two extended barrels each with 64 wedges, each 2.6m long and weighing 9,600 kg
• 500,000 plastic scintillator tiles

Liquid Argon (LAr) Calorimeter Specifications [30]

- Barrel 6.4m long, 53cm thick, 110,000 channels.
- Works with Liquid Argon at -183ºC
- LAr endcap consists of the forward calorimeter, electromagnetic (EM) and hadronic endcaps
- EM endcaps each have thickness 0.632m and radius 2,077m
- Hadronic endcaps consist of two wheels of thickness 0.8m and 1.0m with radius 2.09m
- Forward calorimeter has three modules of radius 0.455m and thickness 0.450m each

Muon Spectrometer

Muons are the particles which pass through the Inner Detector and Calorimeter undetected. The Muon Spectrometer, made up of 4000 individual muon chambers use different technologies from 48 institutions (manufactured in 23 production sites) from around the world, measures the momenta of muons.

Following are the sub-sections of the Muon Spectrometer system :

1. Thin Gap Chambers
2. Resistive Plate Chambers
3. Monitored Drift Tubes
4. Cathode Strip Chambers

Magnet System

The magnet system of ATLAS helps in bending the particles towards various layers of the detector systems, thus making it easier to track particles. The main sections of the magnet system are :

- Barrel Toroid
- End-cap Toroid
- Central Solenoid Magnet

Barrel Toroid Specifications [31]

- 25.3 m length
- 20.1 m outer diameter
- 8 separate coils

- 1.08 GJ stored energy
- 370 tonnes cold mass
- 830 tonnes weight
- 4 T magnetic field on superconductor
- 56 km Al/NbTi/Cu conductor
- 20.5 kA nominal current
- 4.7 K working point temperature
- 100 km superconducting wire

End-cap Toroid Specifications [31]

- 5.0 m axial length
- 10.7 m outer diameter
- 8 coils 8 coils in a common cryostat each
- 0.25 GJ stored energy in each
- 160 tonnes cold mass each
- 240 tonnes weight each
- 4 T magnetic field on superconductor
- 13 km Al/NbTi/Cu conductor each
- 20.5 kA nominal current
- 4.7 K working point temperature

Central Solenoid Magnet Specifications [31]

- Bends charged particles for momentum measurement
- 5.3 long, 2.4 m diameter, 4.5 cm thick
- 5 tonne weight
- 2 tesla (T) magnetic field with a stored energy of 38 megajoules (MJ)
- 9 km of superconducting wire
- Nominal current: 7.73 kiloampere (kA)

Trigger and Data Acquisition System

About 1.7 billion collisions per second happen in the ATLAS detector, with a combined volume of more than 60 million Mbs per second. However, all the collisions are not worth studying. To reduce the flow of data to manageable levels, ATLAS uses a specialized two-level event selection system, which stays online. The system is called a 'Trigger System' which selects the events with distinguishing characteristics for analysis by physicists. [32]

The Trigger System is known to choose/select about 1000 out of 1.7 billion collisions occurring per second at the ATLAS detector. The Data Acquisition System is responsible for channeling the data from detector to the data center and subsequently to storage.

The **ATLAS Trigger System** carries out its responsibility in two stages :

- **Level 1 Hardware Trigger** – Constructed with custom-made electronic hardware, works on a subset of information from the calorimeter and the muon detectors. The decision to keep an event data of a collision is done within microseconds of the actual event, and retrieved through pipelined storage buffers. Level 1 Trigger can save up to 100,000 events per second for the High-Level Trigger (HLT) [32].
- **High Level Trigger (HLT)** – It is a software based trigger (a large farm of CPUs) which refines the analysis of Level 1 Hardware Trigger. About 1000 events per second are selected by HLT analysis and fully assembled into a

single event record. These events are passed on to data storage systems for offline analysis [32].

ATLAS Computing

Data from ATLAS detector is calibrated, reconstructed and automatically distributed all around the world by the ATLAS Data Management System (DDM). The Production Systems filter these events and selects the type needed for analysis. This makes the data manageable when put to analyze from a terminal or end system.

The Worldwide LHC Computing Grid (WLCG) is a Grid Framework allowing all member institutions equal access to ATLAS data. ATLAS computing infrastructure and software are continuously on the road to improvement. ATLAS has over 130 Computing Centers (or Compute Clusters) all around the world.

Worldwide LHC Computing Grid

The Worldwide LHC Computing Grid (WLCG) is a global collaboration of computer centers. WLCG is the world's largest computing grid. It was setup to provide a resource to store, distribute and analyze petabytes of data generated every year at LHC.

The laboratories and universities collaborating together had set up the computing service – called Grid. It now links over 170 compute and storage computing center across 41 countries [33]. The computer centers are arranged in Tiers, serving around 8000 physicists now all across the globe. The Grid facilitates the power to run these compute resources, analyze the data and provide storage for them too.

There are two main grids and two regional grids :

- European Grid Infrastructure (Main Grid)
- Open Science Grid (Main Grid – For United States)
- TWGrid (Regional Grid – In Taiwan)
- EU-IndiaGrid (Regional Grid – For supporting infrastructure across Europe)

More than 8000 physicists all across the world have access to real-time data from four major LHC Experiments – ALICE, ATLAS, CMS, LHCb. Tier 0 of the grid at CERN runs around a million job per day, and the peak data transfer rate can reach up to 10 GB per second. The rate of data generation is expected to be around 500 PB/year during the HL-LHC starting 2024 [33].

Atlas Distributed Data Management

Distributed Data Management (DDM) handles all ATLAS Experiment data on the grid, thereby helping the collaboration to store, manage and process LHC data. DDM Service is developed for data transfer between ATLAS sites and for data cataloging.

Requirements for DDM to function are as follows :

1. Identifying data files
2. Transferring data to and from sites
3. Deletion of data from sites

4. Ensuring data consistency on sites
5. Focusing on ATLAS Computation

ATLAS Computing Model (Tier-based Architecture)

WLCG constitutes of 4 Tiers (or Sites), namely Tier 0, 1, 2 and 3.

Tier 0 : CERN Resources

The CERN Data Center, located at Geneva, Switzerland and Wigner Research Center for Physics in Budapest, Hungary constitutes Tier 0 together. These two sites are connected by three dedicated 100 Gbps data channels. All data out of LHC passes through CERN hub. However, CERN accounts for less than 20% of the total Grid compute capacity.

Tier 0 is responsible for safeguarding the first copy of the raw data, first pass calibration reconstruction, distribution of raw data with reconstructed output to the Tier 1s. Tier 0 reprocesses some of the data during LHC downtime or partial shut-downs.

Tier 1 : National Resources (Based on participating countries)

13 large computing and storage facilities, each known as Tier 1 center with sufficient storage accounts for the continuous support of the Grid. Tier 1s are responsible for keeping proportinal amount of raw and reconstructed data received from Tier 0. They are also responsible for large-scale re-processing and safeguarding both the raw and reconstructed data.

Tier 1 is responsible for distributing data to participating Tier 2s and safekeeping a part of the simulated data processed out of Tier 2s. Tier 1s mostly comprise of National Laboratories.

In United States, Brookhaven National Laboratory (BNL) is Tier 1 for ATLAS Experiment.

Tier 2 : Regional Resources (smaller in storage and compute capacity than Tier 1)

Tier 2s are specifically educational institutions or universities having their dedicated clusters for CERN-specific computation. They can store adequate data and provide sufficient compute power for specific analysis and simulation tasks. They handle analysis requirements and simulate event production and reconstruction.

There are around 160 Tier 2 centers around the globe. In the United States, there are four Tier 2 Collaborations :

- Northeast
- Midwest
- Great Lakes
- Southwest

We'll be considering Southwest Tier 2 (SWT2) for this thesis. SWT2 is a Consortium between Physics Departments of University of Texas at Arlington, Oklahoma University and Langston University.

The SWT2 is funded via the National Science Foundation (NSF) as a subcontract to the U.S. ATLAS Operations program (PHY-119200).

At University of Texas at Arlington (UTA), there are two cloud clusters :

- UTA_SWT2 (Also known as ARDC) [Names would be used interchangeably]
- SWT2_CPB

Tier 3 : Local Resources

Scientists or research scholars will access to local computing resources (called Tier 3) for their individual analysis or simulation tests, on a local cluster or end client of the respective University, with no official affiliation to CERN resources. There's no formal understanding between WLCG and Tier 3 resources.

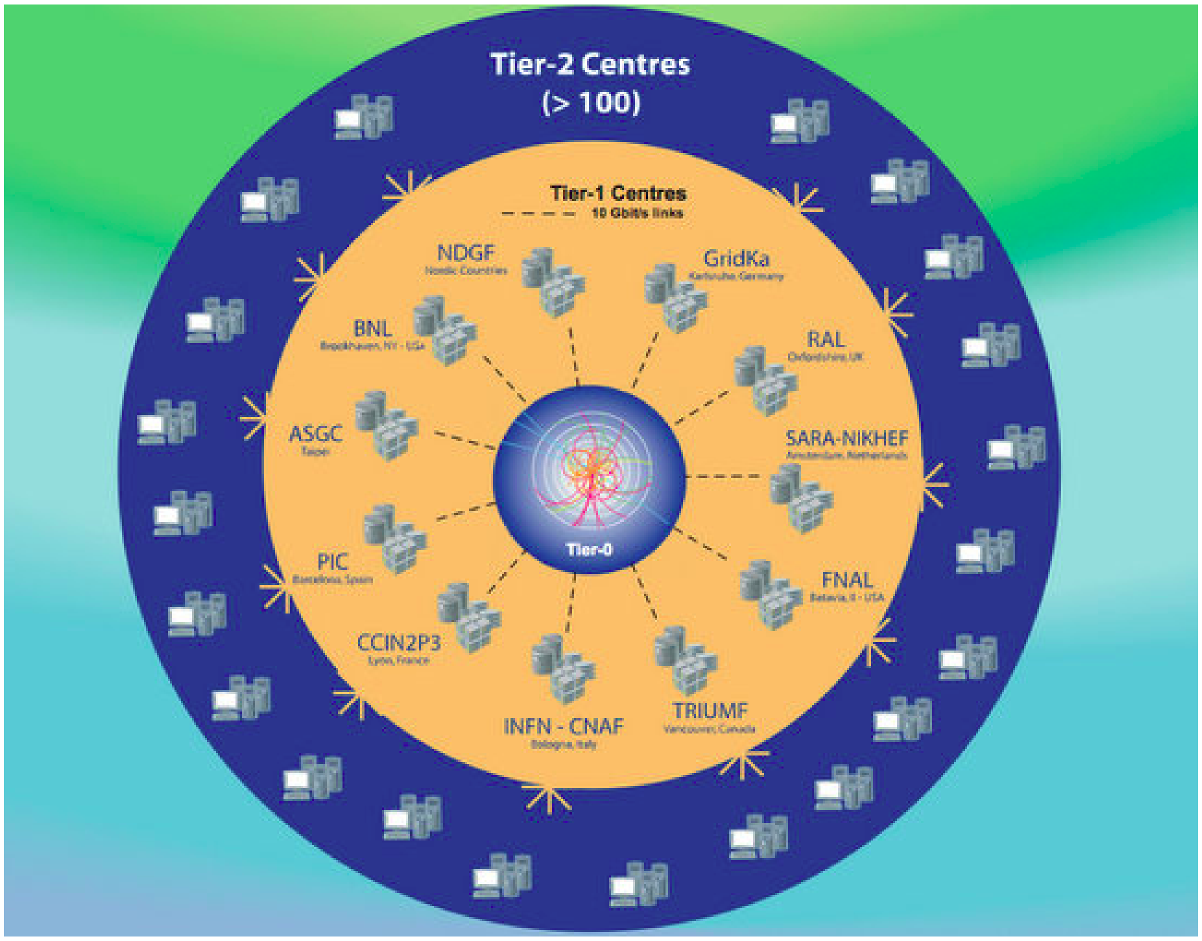


Illustration 4: WLCG Tiers (Compute and Data Centers) [34]

GRID FTP (For File Transfer)

Grid FTP is a file transfer protocol (FTP) used in Grid Computing. It's defined with GridFTP working group of the Open Grid Forum (Global Grid Forum). GridFTP is a high performance, reliable, secure data transfer protocol used for high-bandwidth wide-area network.

The protocol has the following features :-

1. It supports parallel TCP stream (multiple simultaneous TCP streams) and transfers in multi-node in order to achieve high performance. Striped and interleaved transfers, be it from different sources or the same source, causes increase in speed.

2. GridFTP is a fault tolerant implementation of File Transfer Protocol to address problems like network unavailability and server issues. Transfers are automatically restarted if a problem occurs. [36]

3. It requires the server to send checkpoints (markers) to the client.

4. Control channel is encrypted by default. Data channel is authenticated by default with optional integrity protection and encryption.

5. Third-party transfers are allowed, which is the transfer of data between two end hosts, meditated by a third host.

Globus Toolkit is the most common form of implementation of Grid FTP protocol. Globus Toolkit comprises of :-

1. Server implementation called *globus-gridftp-server*

2. Script command line client called *globus-url-copy*

3. Set of development libraries for custom clients

Globus GridFTP Framework supports both Grid Security Infrastructure (GSI) and SSH for securing the data transfer. SSH based GridFTP supports security features like authentication only, authentication and integrity protection, fully encrypted. While FTP doesn't support the transmission of only a certain portion of a file, GridFTP on the other hand, allows a subset of the file to be sent (Useful when only small sections of a large data file are used for processing) [36].

Globus online is the common interface to move data in and out of GridFTP Servers. Globus Online has a Web GUI, Command Line Interface (CLI) and supports REST API service.

The client *globus-url-copy* can be used as follows :

```
globus-url-copy  gsiftp://remote.host.edu/path/to/file  file:///path/on/local/host
```

The above command transfers a file from the remote host to local accessible file path.

Globus File Sharing

Globus file sharing allows individual researchers to simply and securely share data with each other, freeing HPC system administrators from routine requests for access. Globus was developed by computer scientists at University of Chicago and Argonne National Laboratory [35].

Globus is a software-as-a-service (SaaS) enabled by the cloud, but not a cloud storage service (hence not payable). Globus enables movement and sharing of data to and from storage servers across various endpoints (for example, a campus computing cluster, a supercomputer, a laboratory server, etc).

Globus Endpoint

Endpoints are the locations where data can be moved to or from using Globus transfer and sharing system. Endpoints can be on a single system (for e.g a user’s terminal) or for multiuser (located on a server for multiple user access).

In order to move data from cluster to local computer, a Global Connect Server is required to be set up on the cluster and Global Connect Personal on the local machine, thereby creating two endpoints for data sharing.

Globus terminology [35] :

- **Endpoint** : A logical address for a GridFTP server. Data is transferred between Globus endpoints.
- **Globus Connect Server** : A linux package for multiuser environments that sets up GridFTP server for use with Globus. Serves as the Server Endpoint.
- **Globus Connect Personal** : A client for communicating with other GridFTP servers through Globus. It is for individual user environment (for example a researcher’s terminal). Installation of Globus Connect Personal creates a local endpoint used to transfer data to and from the local computer.

Installing GridFTP using ssh support only

GridFTP is very useful for transferring large files across high-speed WANs.

Below are the steps to install GridFTP (on both the client and server hosts) with ssh support only (No X-509 certificate required) [37]:

1. For installing the Globus Repository :

rpm -hUv http://www.globus.org/ftppub/gt6/installers/repo/globus-toolkit-repo-latest.noarch.rpm

2. yum install yum-plugin-priorities [Globus installer needs these plugins]

3. yum install globus-data-management-client globus-data-management-server

4. For enabling sshftp, the command below is needed :
/etc/init.d/globus-gridftp-sshftp reconfigure

The GridFTP Server is automatically launched through sshd. A few sample commands are shown below [37] :

- For listing directories : *globus-url-copy -list sshftp://gridhost.uta.edu/tmp/*
- For copying a file named ‘bigdata’ :
  globus-url-copy sshftp://gridhost.uta.edu/etc/group file:/tmp/bigdata
- For testing the network throughput :
  globus-url-copy -vb -p 4 sshftp://gridhost.foo.gov/dev/zerofile:///dev/null

In order to make troubleshooting and performance analysis easier, the following commands can be added to */etc/gridftp.conf* [37] :

```
log_level ERROR,WARN,INFO
log_single /var/log/gridFTP/gridftp-auth.log
log_transfer /var/log/gridFTP/gridftp.log
log_module stdio_ng
```

CernVM File System (CVMFS)

CVMFS was developed to assist High Energy Physics (HEP) to deploy software on the worldwide distributed computing infrastructure for processing applications. It is scalable, reliable and low on maintenance software distribution service. Files and directories are hosted on standard web servers and mounted in universal namespace */cvmfs*.

CVMFS is implemented as POSIX read-only file system in user space.

CVMFS uses content-addressable storage and Merkle tree to maintain file data and meta-data. CVMFS uses only the outgoing HTTP, hence avoiding most of the firewall issues. It transfers data and meta-data on demand and verifies data integrity by cryptographic hashes. It uses aggressive caching to reduce latency. It consists of many small files that are frequently opened and read. Software uses frequent look-ups for files in multiple directories when search paths are examined.

CVMFS replaces package manager and shared software areas on cluster file system to distribute software to process experiment data.

CernVMFS is actively used by small and large High Energy Physics collaborations. For experiments at the LHC, CernVM-FS hosts several hundred million files and directories which are capable of distributing to hundred thousand client computers.

CVMFS Architecture

The CernVM-FS is a read-only file system designed to deliver scientific software onto virtual machines and physical worker nodes in a fast, scalable and reliable way. CVMFS uses a standard HTTP transport for distribution of files using a few web caches and some content delivery networks. CVMFS ensures data integrity and authenticity over these connections.

Running and compiling software is a use case which general distributed file systems are not designed for. By general purpose file system we mean NFS or AFS. Software installed in CVMFS does not need to be further packaged in contrast to Docker images/VM images. The CVMFS software comprises client-side software to mount ‘CVMFS Repositories’, and server-side toolkit to create distributable CVMFS repositories [38].

A dedicated machine called *Release-Manager Machine [38]* is used to create and update a CVMFS Repository. A CVMFS Repository is mounted on a Release Manager Machine in read/write mode by a union file system. The union file system overlays the CVMFS read-only mount point by a writable scratch area. CernVM-FS Server Toolkit merges changes made to the scratch area into CVMFS Repository. A CVMFS Repository is analogous to a version control system repository.

On the client side, only the data and meta-data are downloaded and cached.

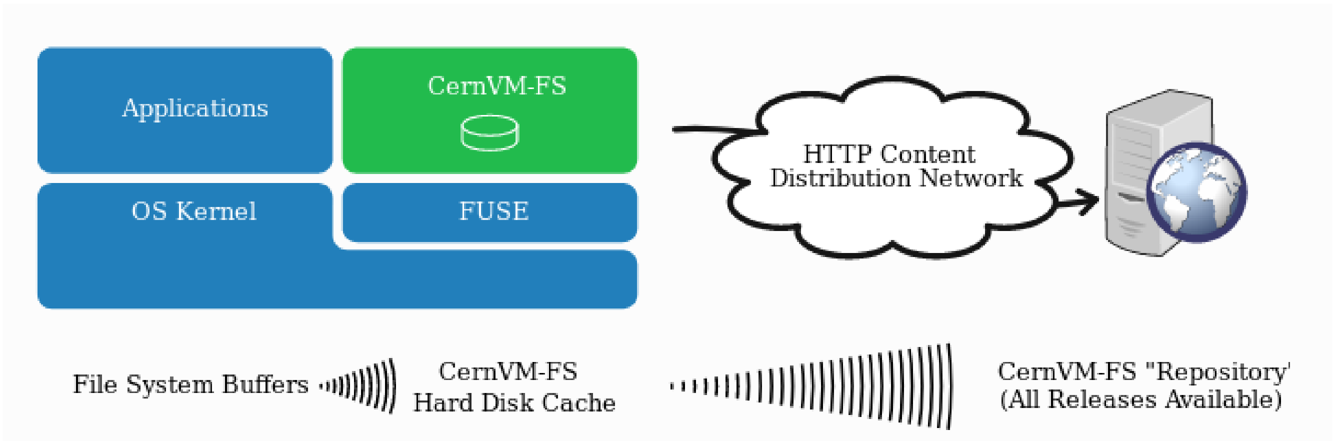
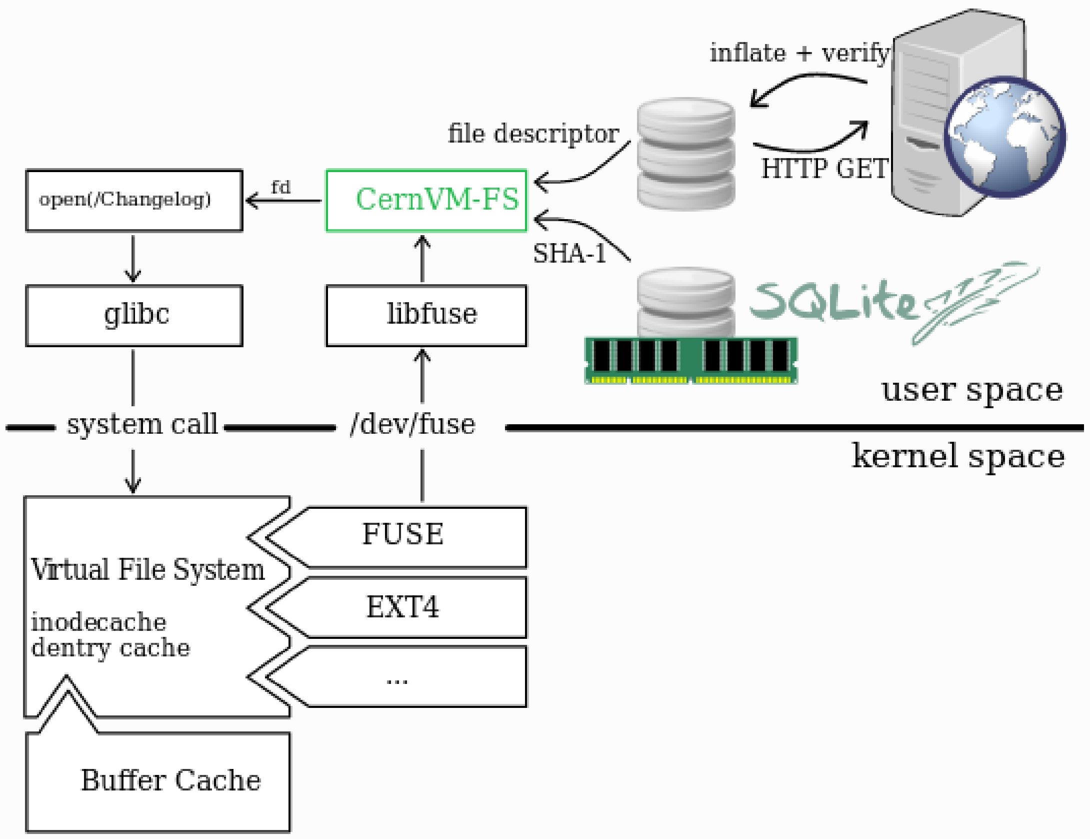

Applications
CernVM-FS
OS Kernel
FUSE
HTTP Content Distribution Network
File System Buffers
CernVM-FS Hard Disk Cache
CernVM-FS "Repository" (All Releases Available)

Illustration 5: CVM-FS in a Web Server [38]

As per *Illustration 5*, all the releases of software package are hosted on CVMFS Repository on a Web Server. A CVM-FS client system provides a virtual file system that loads data only on access.



Opening a file on CernVM-FS. CernVM-FS resolves the name by means of an SQLite catalog. Downloaded files are verified against the cryptographic hash of the corresponding catalog entry. The `stat()` system call can be entirely served from the in-kernel file system buffers.

Illustration 6: Opening a file on CVMFS [38]

RUCIO

The scientific data management software service that manages the ATLAS servers is known as Rucio. It is a project that provides services and associated libraries for allowing scientific collaborations for managing large volumes of data spread across different institutions and organizations. It is highly scalable and modular. An account is the unit of assigning privileges in Rucio. It can represent individual users, a group of users or centralized production activity like service accounts or workflow management system [40].

A Rucio user is authenticated by credentials, such as X509 certificates, username and password, SSH or tokens. Credentials can map to several accounts.

Rucio talks to GRIDFTP door, which in turn requests data from XROOTD Storage subsystems. Files are the smallest operational unit of data in Rucio. It allows users to identify and access any arbitrary set of files [40].

Rucio Storage Element (RSE)

It is the logical abstraction of a storage system for physical files. It is the smallest unit of storage space addressable in Rucio. It consists of abstraction for storage end-point, and can be grouped in different ways with tags [39].

It has a unique identifier and a set of meta-data attributes describing properties such as supported protocols, quality of service, storage type, geographical location/zone, host/port address and physical space (used space, available space).

RSE deterministically maps logical and physical file name to remove external file catalogue lookups.

E.g of RSE : Tier 1 RSE, Tier 2 RSE, US RSE.

Rucio Cache RSE

In Rucio, a cache is volatile and the controlling of cache is usually handled by an external process or applications (Workflow Management Systems) and not by Rucio [39].

The application populating the cache must register or unregister the file replicas in Rucio. Replicas registered on volatile RSEs are excluded from the Rucio Replica Management System.

Chapter 3 : PanDA and XRootD

PanDA

PanDA stands for Production ANd Distributed Analysis system, developed by ATLAS as data-driven workflow management for production and distributed analysis processing operation at LHC data processing sites. PanDA is a Grid Scheduling System, for processing and scheduling ATLAS jobs.

Jobs are submitted to PanDA using a client interface and the users define the job sets (production/analysis), the associated datasets and the input files with them. Job specifications are sent to PanDA via secure http (via grid certificate proxy), and submission information is returned to the client. PanDA server receives work from client interfaces (also called ‘front end’) and keep them in a global job queue based on job type, job priority, input data and locality, available CPU slots, etc.

A pre-condition for job execution is data pre-placement to the compute node(s). Jobs are not released for processing until the input data is completed accessible from the processing site (worker nodes). When data dispatch completes, jobs are made available to job dispatcher.

An independent subsystem manages the delivery of pilot jobs to worker nodes, using local job submission methods. A pilot when launched on a worker node contacts the dispatcher and receives available job to the site. The pilot isolates the PanDA system, so that at PanDA level the grids used by PanDA appears homogeneous.

PanDA Architecture

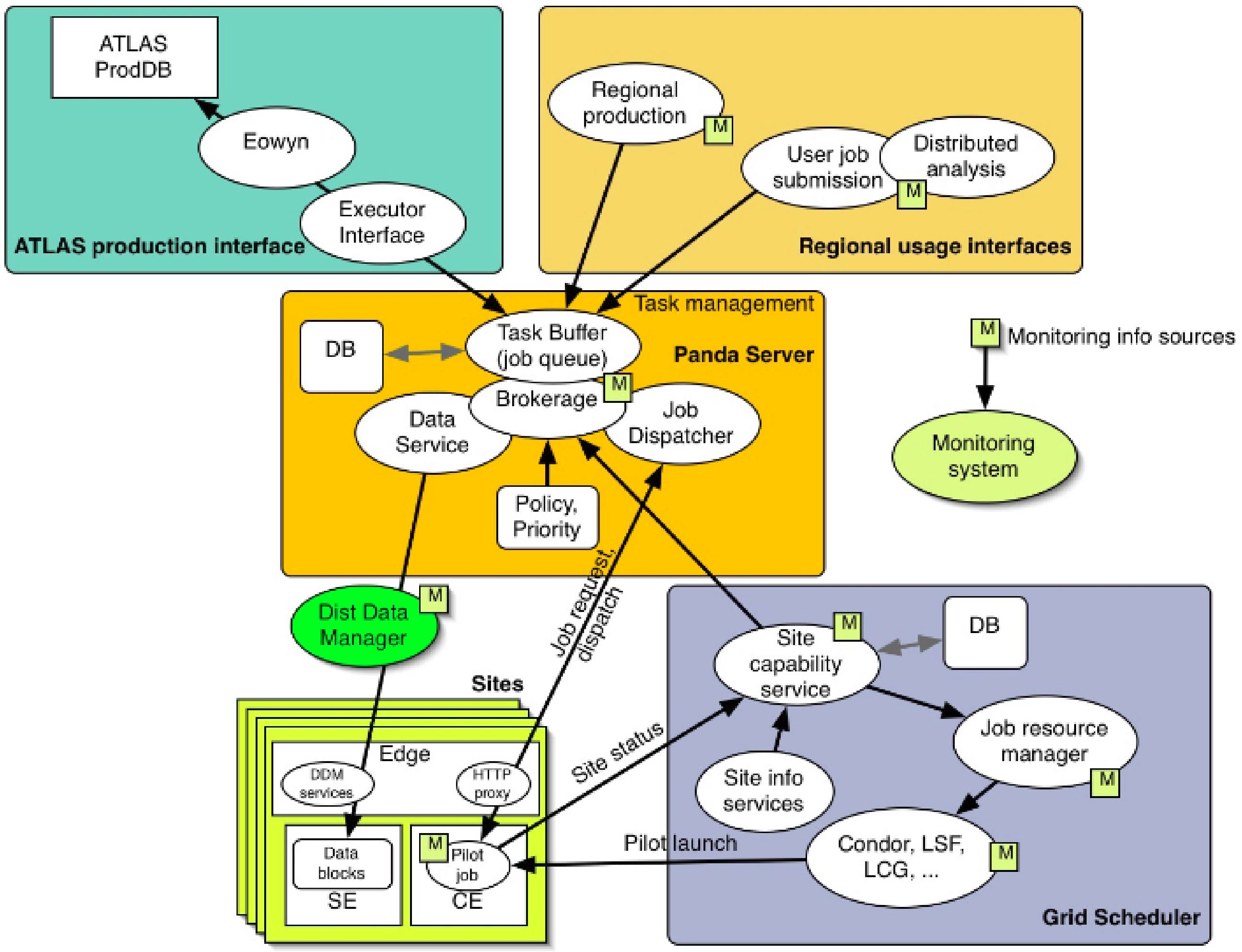


Illustration 7: PanDA Architecture [42]

A PanDA server receives jobs from the interfaces and keeps them in a global job queue. A brokerage module assigns priority to the work based on the job type, input data, locality, priority and available CPU resources. Job sets are allocated to the local processing site, followed by the allocation of corresponding input datasets to those sites.

After the input data is dispatched, jobs are made available to a job dispatcher. A subsystem manages the delivery of pilot jobs to worker nodes. A pilot launched on a worker node contacts the dispatcher and receives a job for the site. In case there's no job available, the pilot exits immediately.

The pilot dispatch mechanism bypasses any latency in the scheduling system for submitting by launching the pilot itself. A workload job is assigned only when a pilot is successfully launched at the worker node. The architecture allows isolation of pilot to subsequently allow the integration of different pilot system implementations with PanDA.

PanDA Job Workflow

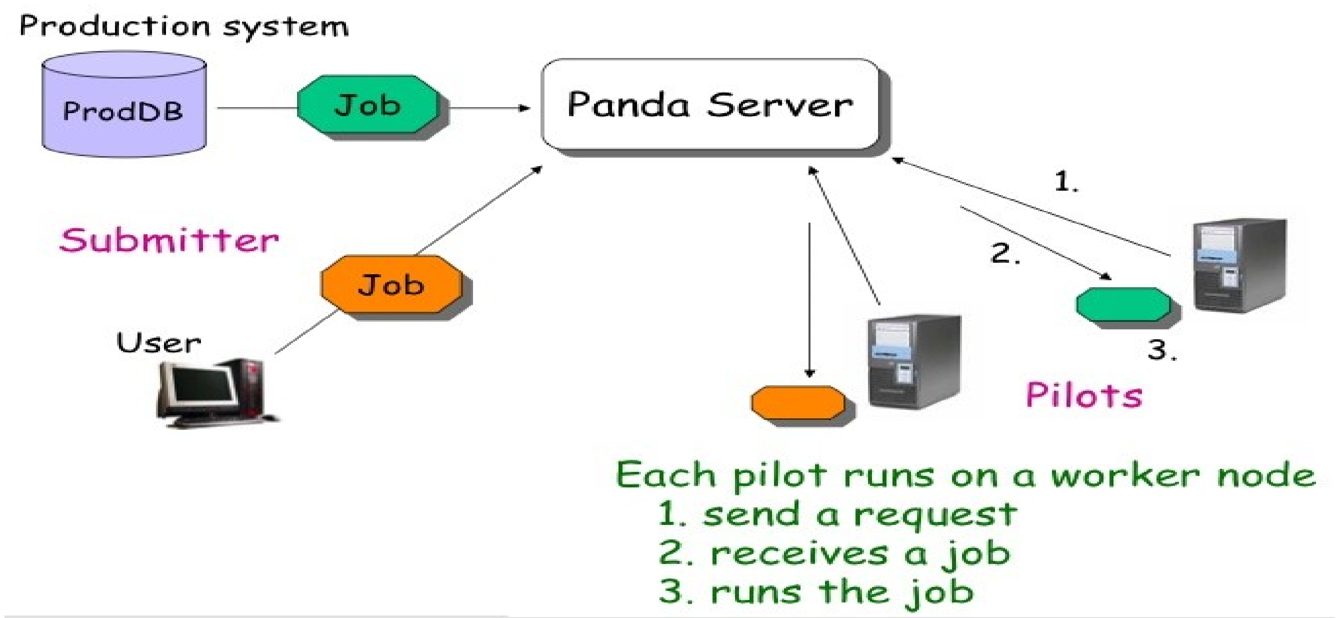


Illustration 8: PanDA Server acting as scheduling system

PanDA Components

PanDA Server : It is the central hub where the core PanDA operations take place [41]. It consists of the following :

1. PanDA Task Buffer : It is a job queue manager keeping track of all the active jobs in the system.
2. PanDA Brokerage : Manages the dispatch of input data to processing sites.
3. PanDA Job Dispatcher : Receives requests for jobs from pilots and dispatches job payloads.
4. PanDA Data Service : Data Management Services implemented with ATLAS DDM system.

PanDA Client : PanDA Client module is a python interface used for submitting jobs to PanDA. It is also used for querying the job status, and controlling jobs in the system. It uses HTTP as Communication Protocol, and a grid certificate is necessary to submit jobs.

PanDA Pilot : It is an environment used to prepare the execution in the computing element, request a production or analysis job, execute it and clean up the environment after execution. PanDA Pilot supports High Performance Computing (for e.g : NERSC, Titan using Backfill, etc) [41].

PanDA Harvester : Harvester is a resource-facing service between the PanDA server and collection of pilots for resource provisioning and workload shaping. It is a lightweight stateless service running on a VObox or an edge node of High Performance Computing centers to provide a uniform view for various resources [41].

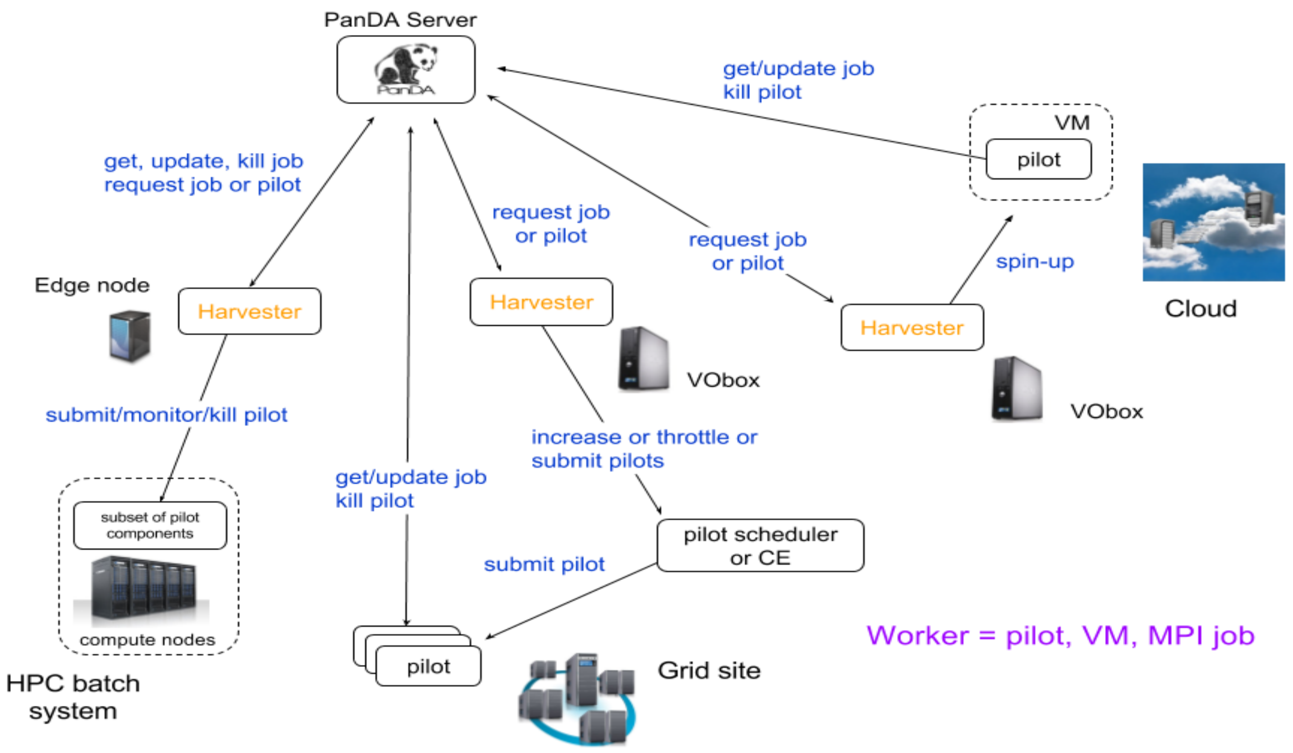


Illustration 9: Interoperability between PanDA Server, Harvester and Pilot [41]

Certificate Authority

The certificate authorities (CAs) provide the trust roots for the public key infrastructure Open Science Grid uses to maintain integrity of its sites and services. The certificate revocation lists(CRLs) on the OSG hosts needs to be up-to-date to find out which certificates to trust and the ones revoked (discarded) forever.

There are basically 3 ways of installing CA Certificates having different control levels :

1. Installing an RPM (Red Hat Package Manager) for a specific set of CA certificates (default)

2. Installing *osg-ca-scripts*, a set of scripts that provide fine-grained CA management

3. Installing an RPM that doesn't install any CAs. This is useful to manage CAs while satisfying RPM dependencies.

Host Certificates

Host certificates are X.509 certificates that are used to securely identify servers and to establish encrypted connections between service and client. Some grid resources (for example XRootD, GridFTP) require X.509 certificates [43].

Host certificates can be requested from one of the following certificate authority services [43] :

- InCommon : An IGTF-accredited (Interoperable Global Trust Federation) certificate authority for services intercating with WLCG.
- Let's Encrypt : OSG has included a non-IGTF CA in its distribution bundle.

User Certificates

A User certificate is required to interact with the OSG resources and infrastructure. The common user certificate formats used in OSG are PKCS12 and PEM. In PEM format, the user certificate is stored in two separate files, one for the certificate and the other for private key [44].

A PKCS12 format stores the certificate and the private key along with the certificate chain (optional) in a single file.

XRootD

XRootD is a hierarchical storage system that can be used in a variety of ways to access data, typically distributed among actual storage resources. XRootD can refer to many data resources at a single site, also it can refer to many storage systems, most likely distributed among sites. An XRootD system includes a Redirector, which accepts requests for data and finds a storage repository — locally or otherwise — that can provide the data to the requestor. An XRootD is not a typical directorial storage system, but a hierarchical namespace.

Scaling XRootD

XRootD can cluster servers, clustering done by the clustering manager called cmsd. cmsd can cluster any type of server. The cluster manager keeps track of where files are, redirects to the proper server, thereby avoiding duplicate caching of files [45].

XRootD is used for high performance, scalable fault tolerant access to different types of data repositories. The typical usage is to give access to file-based ones. It is based on a scalable architecture, a communication protocol, and a set of plugins and tools based on those. The freedom to configure it and to make it scale (for size and performance) allows the deployment of data access clusters of virtually any size, which can include sophisticated features, like authentication/authorization, integrations with other systems, WAN data distribution, etc [45].

XRootD is a software framework for fast, low latency, and scalable data access, which can serve natively any kind of data, organized as a hierarchical filesystem-like namespace, based on the concept of directory.

Requirements for Installing XRootD [45]:

- UserID : The installation will create a linux user ID XRootD.
- Service Certificate : The XRootD service uses a host certificate at */etc/grid-security/host*.pem*
- Port : XRootD service uses Port Number 1094 by default
- Obtaining root access to the host
- Preparing Yum Repositories
- Installing Certificate Authority Certificates

Installing an XRootD Server

Installation of XRootD server consists of the server itself and its dependencies. Installing using yum (Open source command line package management utility for computers running GNU/Linux OS using RPM Package Manager) :

```
root@host # yum install xrootd
```

Configuring and Testing an XRootD Server

A new installation of XRootD is already configured to run a standalone server that serves files from */tmp* on the local file system [45].

Following configuration demonstrates basic connectivity between client and server. Copying a file such as */bin/sh* to */tmp* using *xrdcp* command [45] :

```
root@host # yum install xrootd-client
root@host # xrdcp /bin/sh root://localhost:1094//tmp/first_test
[xrootd] Total 0.76 MB  [====================] 100.00 % [inf MB/s]
root@host # ls -l /tmp/first_test
-rw-r--r-- 1 xrootd xrootd 801512 Apr 11 10:48 /tmp/first_test
```

XRootD Cluster

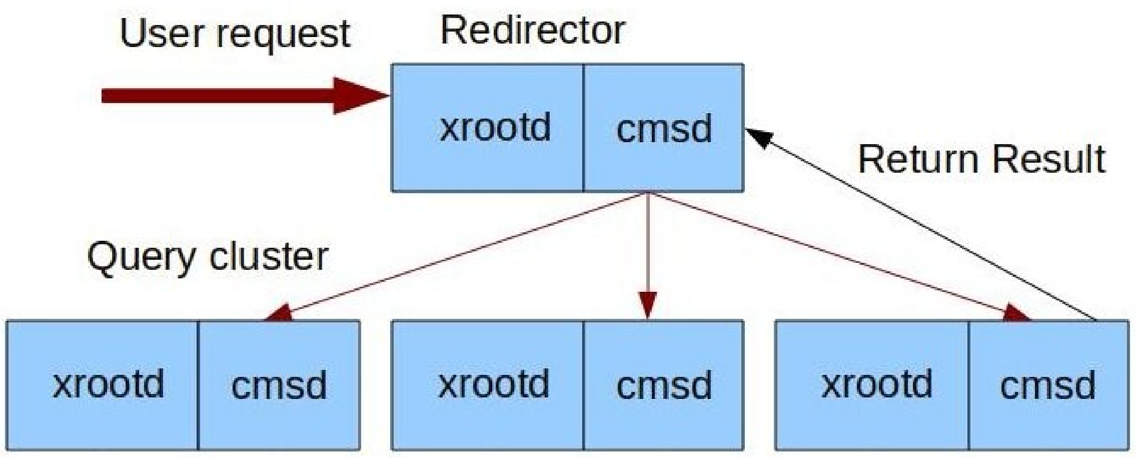


Illustration 10: Interaction of daemons xrootd and cmsd in XRootD Cluster [45]

If storage is spread across different storage servers or multiple hosts, there’s a need to set up XRootD cluster. The cluster uses a ‘Redirector’ node which directs the user requests to the appropriate storage server containing the data user requests.

So each node has two daemon processes running :

- xrootd : Extended Root Daemon controls the file access and storage
- cmsd : Cluster Management Services Daemon controls communication between nodes

Apart from the two daemon processes, a node also has a disk subsystem associated with it, which physically hosts the data.

A ‘Redirector’ serves at the most 64 storage server nodes. For multi-level hierarchy, many site-level Redirectors may communicate with a Regional/Global Redirector.

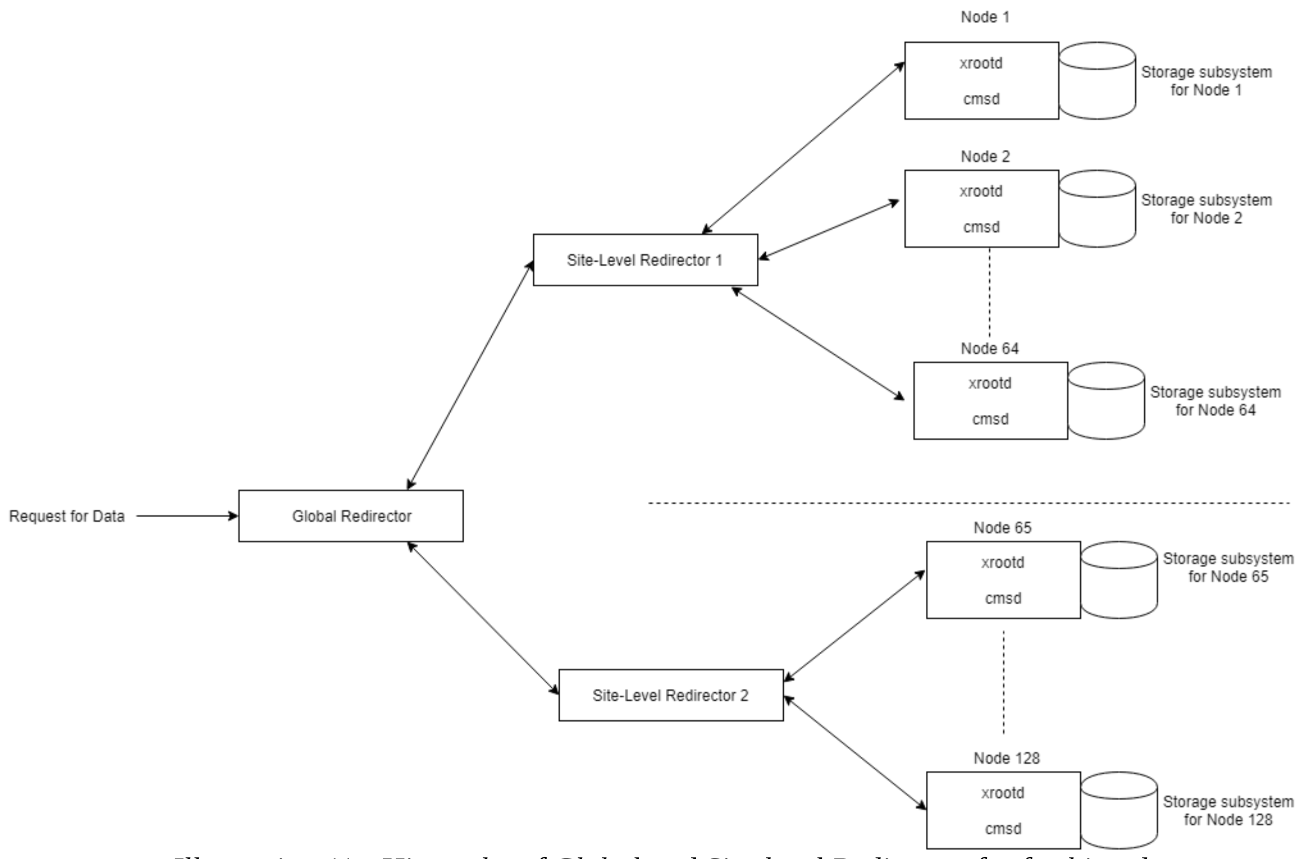


Illustration 11: Hierarchy of Global and Site-level Redirector for fetching data

Features of SWT2_CPB Cloud Cluster

Resource Overview [47]

- 20 Gbps external network connection
- 451 compute nodes
  - 4928 cores (9856 job slots) (Two computing threads per core)
  - At least 2 GB RAM per job slot
- 32 storage servers
  - 5.5 PB of usable storage
- 1 NFS Server
- 1 Admin Server
- 14 Service nodes
- 2048 switches

Compute Resource [47]

- Dell R630
  - CPU : 2 x Xeon E5-2640 v4 2.4 GHz Broadwell 10 cores, 20 threads/CPU
  - RAM 128 GB, 2.133 GHz, DDR4
  - Storage : 2 x 800 GB SSD Raid 0
  - Network :
    - 2 x 10 Gbps (SFP+)
    - 2 x 1 Gbps (Ethernet)

Storage [47]

- MD 3460 Shelf
  - 60 x 8 GB SATA disk
  - 4 x RAID 6 Disk Groups

    - 4 x 12 Gbps SAS connections
- Dell R730
    - CPU : 2 x Xeon E5-2680 v4 2.8 GHz
    - RAM : 256 GB, 2.133 GHz, DDR4
    - Network : 4 x 10 Gbps (SFP+), 2 x 1 Gbps (Ethernet)

Storage is federated through XRootD.

Networking [47]

- Core : 2 x Dell S6000 (LAG)
    - 32 x 40 Gbps (QFSP+)
- TOR : Dell N4032F
    - 24 x 10 Gbps (SFP+)
    - 2 x 40 Gbps (QFSP+)
- Legacy : Dell N2048
    - 48 x 1Gbps (Ethernet)
    - 2 x 10 Gbps (SFP+)

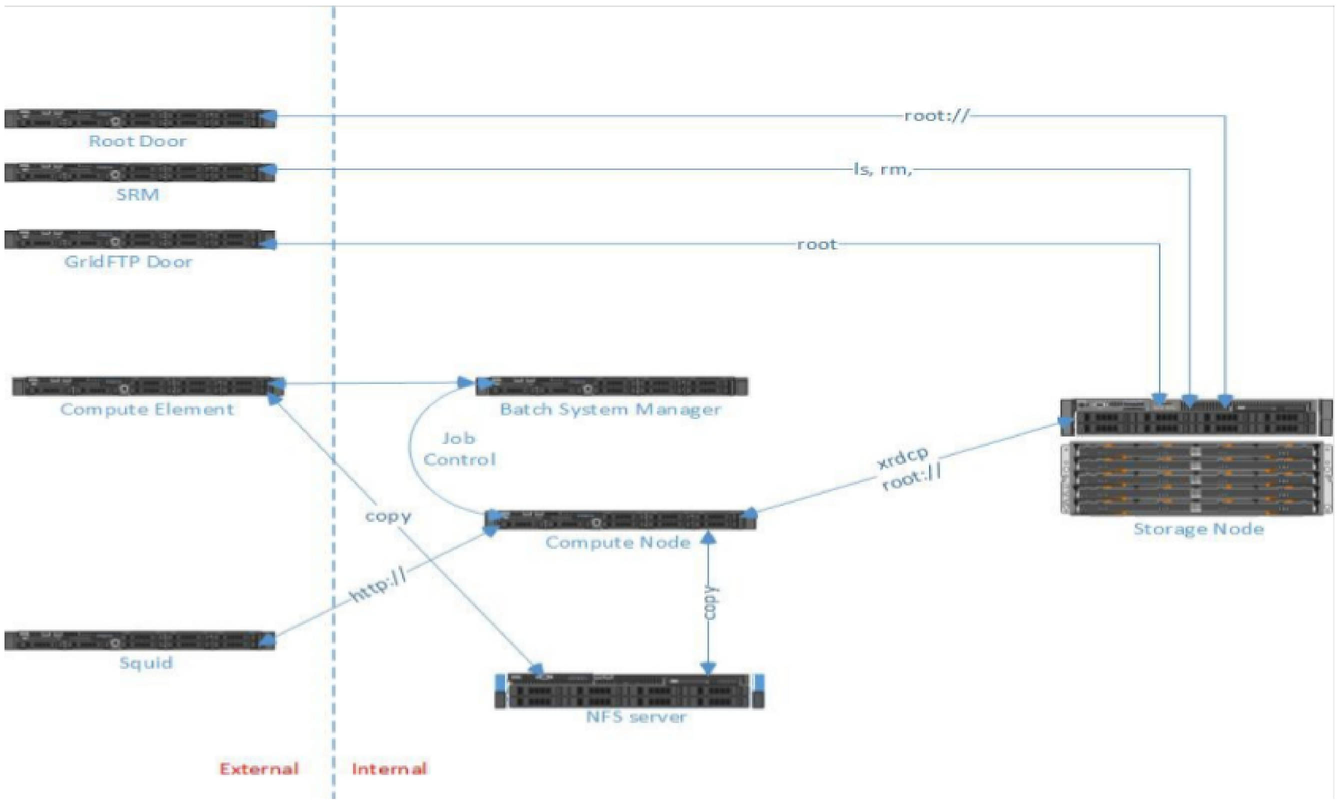


Illustration 12: Network topology of cloud cluster [47]

Chapter 4 : Xcache as File Caching Proxy Server

Xcache

Xcache is a file block caching service. It is horizontally scalable, and configurable for different workflows. Xcache is self manageable in terms of disk utilization. It is used mainly for regional/local use, and accessible by HTTPS or ROOT protocol.

Xcache primarily uses ‘root’ Protocol. It is a multi-threaded file caching application that can asynchronously fetch and cache file segments or the entire file(s). The designed use case for Xcache is to cache static scientific data files of any format be it small or large in size. Size of file can range from 1 MB to 32 MB. Xcache is built upon XrootD and is customizable through configuration or plugins.

An Xcache instance can deployed using a container or CVMFS, but in our case we’ll consider implementation of Xcache in a cloud cluster (data-center). Xcache works with RUCIO to facilitate an improvement in cache hit rate and allowing location independent data access via global logical file name [46].

Another important use case, is the access of Xcache by client(s) through HTTP protocol, which is being explored to replace Squid cache in CVMFS distribution chain.

Importance of Xcache

Xcache when implemented near computational resources would :

1. Reduce access latency
2. Keeps localized data at the localized spot, causing lesser transport of data
3. Helps in accessing replicated data from worldwide without any change to applications

Xcache enhances the Ceph random I/O performance where Ceph is deployed. ‘Ceph’ is a storage technology based on object storage. Provides cache service compatible with CVMFS.

Xcache Implementation

XRootD POSIX plug-in caches files or file blocks on local disk. POSIX library is used by a proxy plug-in, and XRootD can use the plug-in for storage access. An XRootD server accesses the files from a remote location, and Xcache helps in local caching. So the resulting system/service is an XRootD caching proxy server, and all the data access goes through this service.

Before understanding the Xcache architecture, let’s try to comprehend the clusters at ARDC (UTA_SWT2) and CPB (Chemistry-Physics-Building).

Southwest Tier 2 Resources [47]

1. The University of Texas at Arlington
   - **UTA_SWT2** :
     - Dedicated to simulations
     - 2360 job slots, 420 TB of storage
   - **SWT2_CPB** :
     - Performs all Tier 2 processing jobs
     - 10,000 job slots, 5.5 PB of storage

2. The University of Oklahoma
   - **OU_OCHEP** :
     - Performs all Tier 2 processing jobs
     - 844 job slots
   - **OU_OSCER** :
     - Campus cluster provides additional simulation resources
     - 2000 job slots, 700 TB of storage

ATLAS Tier 2 Roles [47]

- Provides CPU cycles for :
  - User Analysis of collision events
  - Monte Carlo simulations of events

    - Reprocessing exercises
- Providing storage for :
    - High Energy Physics data :
        - From recorded collisions
        - From simulated collisions
    - User Analysis results

Xcache System Architecture in Cloud Cluster

The compute node(s) at UTA_SWT2 (ARDC) cloud cluster, requests for a data file to the Xcache system being configured as a direct mode proxy. The Xcache (Redirector at ARDC) system through Root protocol asks for a file through the proxy door from the Redirector at CPB. The compute nodes and the storage subsystems at both clusters CPB and ARDC are in respective private networks.

The Proxy doors at both end of the private network are open to the Public network (the Internet). The two cloud clusters are connected by a 1 Gbps dedicated fiber-optic channel.

RUCIO the data management system requests for data through the GridFTP Servers (GridFTP Door). The XrootD door accesses the storage at CPB cluster through an xrd request via ROOT Protocol.

The communication protocol between Xcache system at ARDC and the Proxy Door at CPB is also ROOT. The compute nodes at ARDC request for data files to start running local jobs allocated by PanDA.

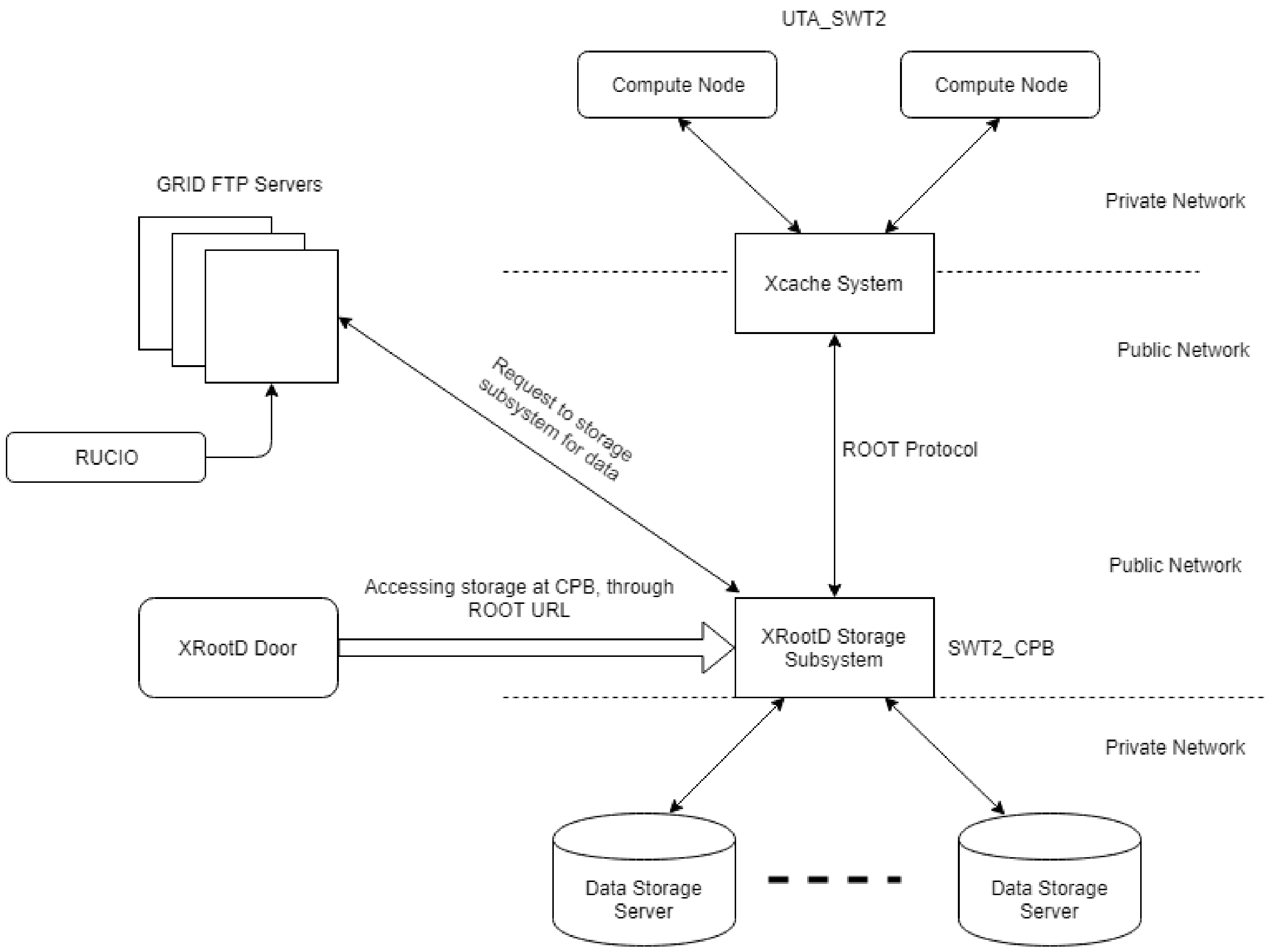


Illustration 13: Xcache System Architecture with respect to a Grid Infrastructure

Requirements for Xcache setup

Software/Protocol Requirements

1. CVMFS (CernVM File System)
2. ROOT Protocol (Protocol for transferring data files)
3. LSM and LSM-Cache (Local Site Mover)
4. Storage Resource Manager (SRM) – For coordinating storage location, streaming data between sites and enforcing secure interface to storage systems
5. X.509 Host Certificates (for authentication)

Machine Requirements

Atleast 2 machines and a number of clients to access the second machine :

1. **Machine 1** : Must have CVMFS (File System) and CVMFS Server Tools, and XRootD Daemon (xrd)
2. **Machine 2** : XRootD and Xcache installed. This machine would be connected to the client terminals (also known as the worker nodes).

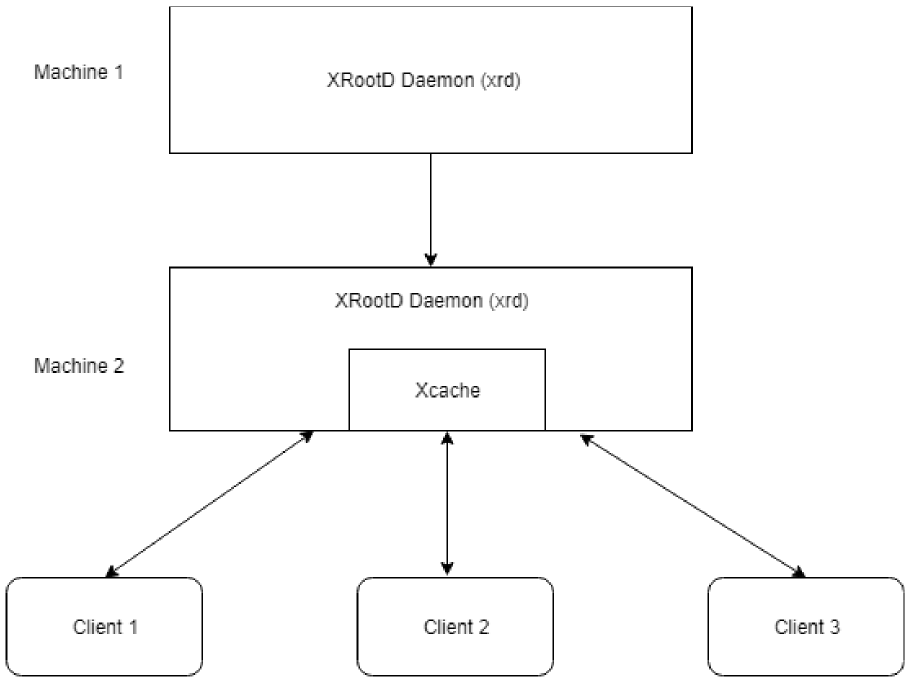


Illustration 14: Block Diagram of Xcache and XRootD as two machine system

The XRootD structure requires all components to read directives from the same configuration file. This configuration file is started when *service xrootd start* command is issued to start xrootd service. The protocol driver (xrd) runs the xroot/root protocol which utilizes the file system plug-in that relies on the proxy system plug-in. Collectively all the components together make up *xrootd.*

Components of XRootD system

The four components of xrootd system that needs to be configured are [48] :

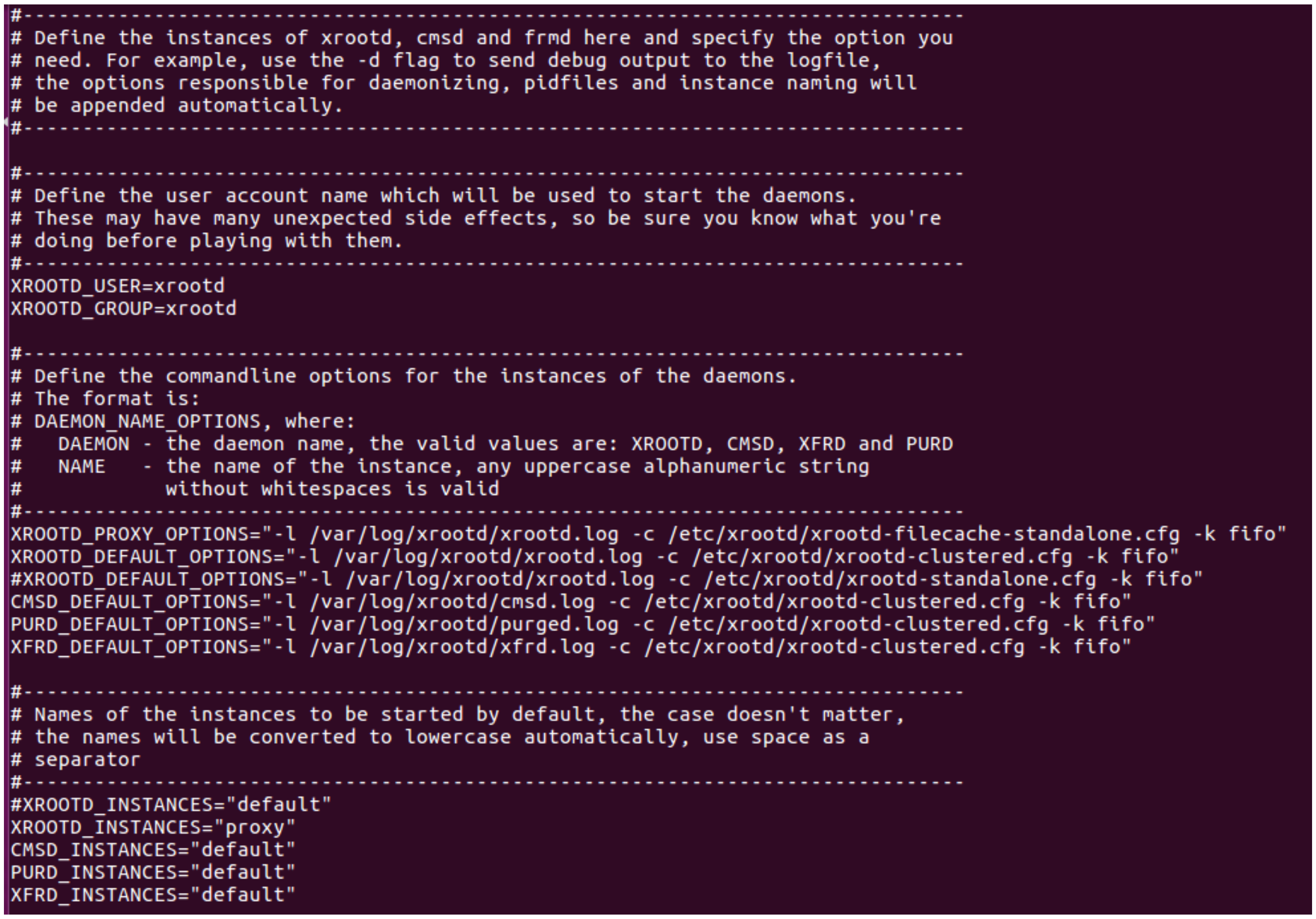
- File system plug-in (*ofs*) : Open File System coordinating acc, cms, oss and proxy storage plug-ins (*pss*).
- Proxy file cache / Specialized proxy plug-in (*pfc*).

- Proxy storage system service/ proxy storage plug-in (*pss*) : Specialized *oss* plug-in.
- *all* component applies the directive to ofs, pfc and pss components.

The ofs.osslib loads the shared library implementing a specialized pss component.

XRootD Configuration for ARDC Cluster

Below is the example of the *xrootd* configuration for our cluster environment for defining the xrootd users, xrootd groups, the default and proxy configuration settings :

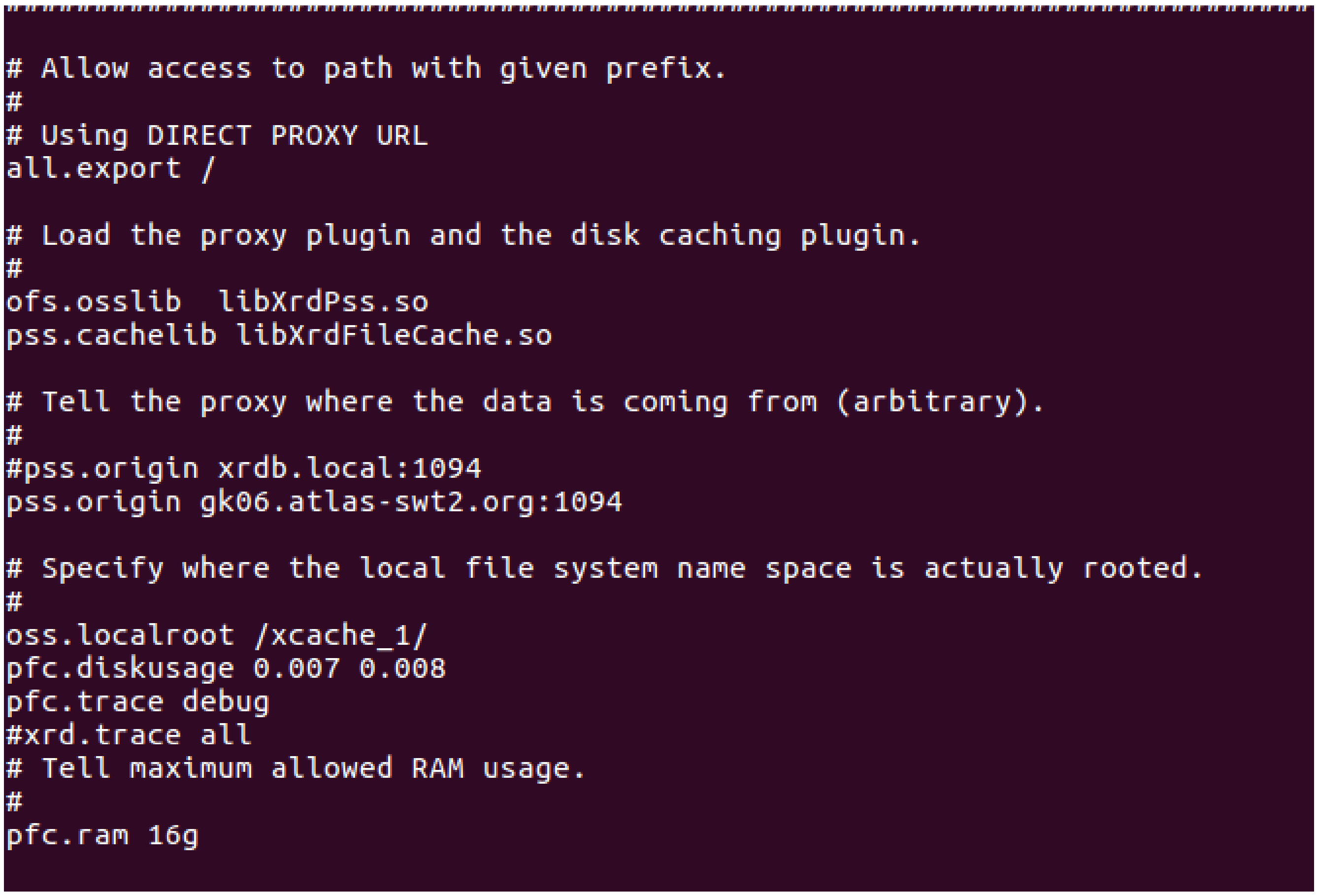


```
#---------------------------------------------------------------
# Define the instances of xrootd, cmsd and frmd here and specify the option you
# need. For example, use the -d flag to send debug output to the logfile,
# the options responsible for daemonizing, pidfiles and instance naming will
# be appended automatically.
#---------------------------------------------------------------

#---------------------------------------------------------------
# Define the user account name which will be used to start the daemons.
# These may have many unexpected side effects, so be sure you know what you're
# doing before playing with them.
#---------------------------------------------------------------
XROOTD_USER=xrootd
XROOTD_GROUP=xrootd

#---------------------------------------------------------------
# Define the commandline options for the instances of the daemons.
# The format is:
# DAEMON_NAME_OPTIONS, where:
#   DAEMON - the daemon name, the valid values are: XROOTD, CMSD, XFRD and PURD
#   NAME   - the name of the instance, any uppercase alphanumeric string
#            without whitespaces is valid
#---------------------------------------------------------------
XROOTD_PROXY_OPTIONS="-l /var/log/xrootd/xrootd.log -c /etc/xrootd/xrootd-filecache-standalone.cfg -k fifo"
XROOTD_DEFAULT_OPTIONS="-l /var/log/xrootd/xrootd.log -c /etc/xrootd/xrootd-clustered.cfg -k fifo"
#XROOTD_DEFAULT_OPTIONS="-l /var/log/xrootd/xrootd.log -c /etc/xrootd/xrootd-standalone.cfg -k fifo"
CMSD_DEFAULT_OPTIONS="-l /var/log/xrootd/cmsd.log -c /etc/xrootd/xrootd-clustered.cfg -k fifo"
PURD_DEFAULT_OPTIONS="-l /var/log/xrootd/purged.log -c /etc/xrootd/xrootd-clustered.cfg -k fifo"
XFRD_DEFAULT_OPTIONS="-l /var/log/xrootd/xfrd.log -c /etc/xrootd/xrootd-clustered.cfg -k fifo"

#---------------------------------------------------------------
# Names of the instances to be started by default, the case doesn't matter,
# the names will be converted to lowercase automatically, use space as a
# separator
#---------------------------------------------------------------
#XROOTD_INSTANCES="default"
XROOTD_INSTANCES="proxy"
CMSD_INSTANCES="default"
PURD_INSTANCES="default"
XFRD_INSTANCES="default"
```

Illustration 15: xrootd system configuration file

As per *Illustration 15*, the XROOTD_USER and XROOTD_GROUP are defined as xrootd, so the daemons start up with the name ‘xrootd’.

The proxy options for xrootd system is defined in the file *xrootd-filecache-standalone.cfg* :



```
# Allow access to path with given prefix.
#
# Using DIRECT PROXY URL
all.export /

# Load the proxy plugin and the disk caching plugin.
#
ofs.osslib  libXrdPss.so
pss.cachelib libXrdFileCache.so

# Tell the proxy where the data is coming from (arbitrary).
#
#pss.origin xrdb.local:1094
pss.origin gk06.atlas-swt2.org:1094

# Specify where the local file system name space is actually rooted.
#
oss.localroot /xcache_1/
pfc.diskusage 0.007 0.008
pfc.trace debug
#xrd.trace all
# Tell maximum allowed RAM usage.
#
pfc.ram 16g
```

Illustration 16: Configuration file for starting xrootd file caching proxy server

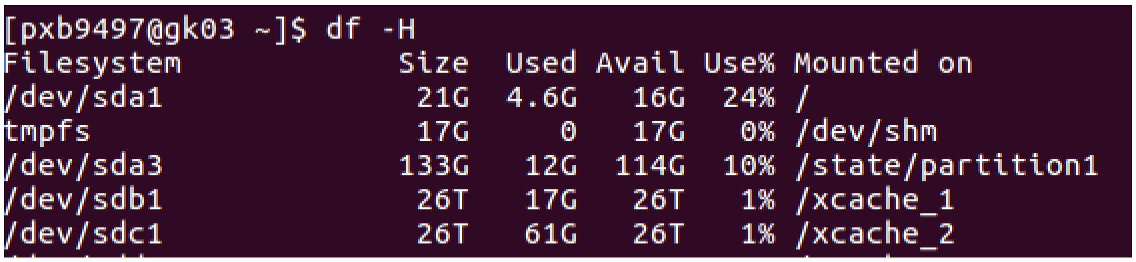
The *libXrdFileCache.so* is a proxy server cache plug-in shared library for caching data into local files.

The *libXrdPss.so* is the plug-in where *ofs.osslib* directive orders the *ofs* layer to change a regular *xrootd* server into a proxy server.

The *pss.origin* directive directs the proxy server to the hostname and port to connect to get the data files. In our case the Redirector at ARDC (Machine *gk03.local*) configured as proxy server gets its data files from machine *gk06.atlas-swt2.org:1094* at CPB.

The *oss.localroot* directive followed by a path directs the proxy to store the data files in that path upon request. In our case we have chosen */xcache_1/* as our local path for file storage.

The pfc.diskusage directive is the physical storage management properties. p*fc.diskusage* is followed by two numerical values sets the Low and High Watermark respectively for the storage area path. In our test environment, we have chosen Low and High Watermark as 0.007 and 0.008 respectively. The storage space of */xcache_1/* is 26 TB (Refer to *Illustration 17*) :



```
[pxb9497@gk03 ~]$ df -H
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1              21G  4.6G   16G  24% /
tmpfs                  17G     0   17G   0% /dev/shm
/dev/sda3             133G   12G  114G  10% /state/partition1
/dev/sdb1              26T   17G   26T   1% /xcache_1
/dev/sdc1              26T   61G   26T   1% /xcache_2
```

Illustration 17: Storage space of /xcache_1 (Area configured to store Xcached files)

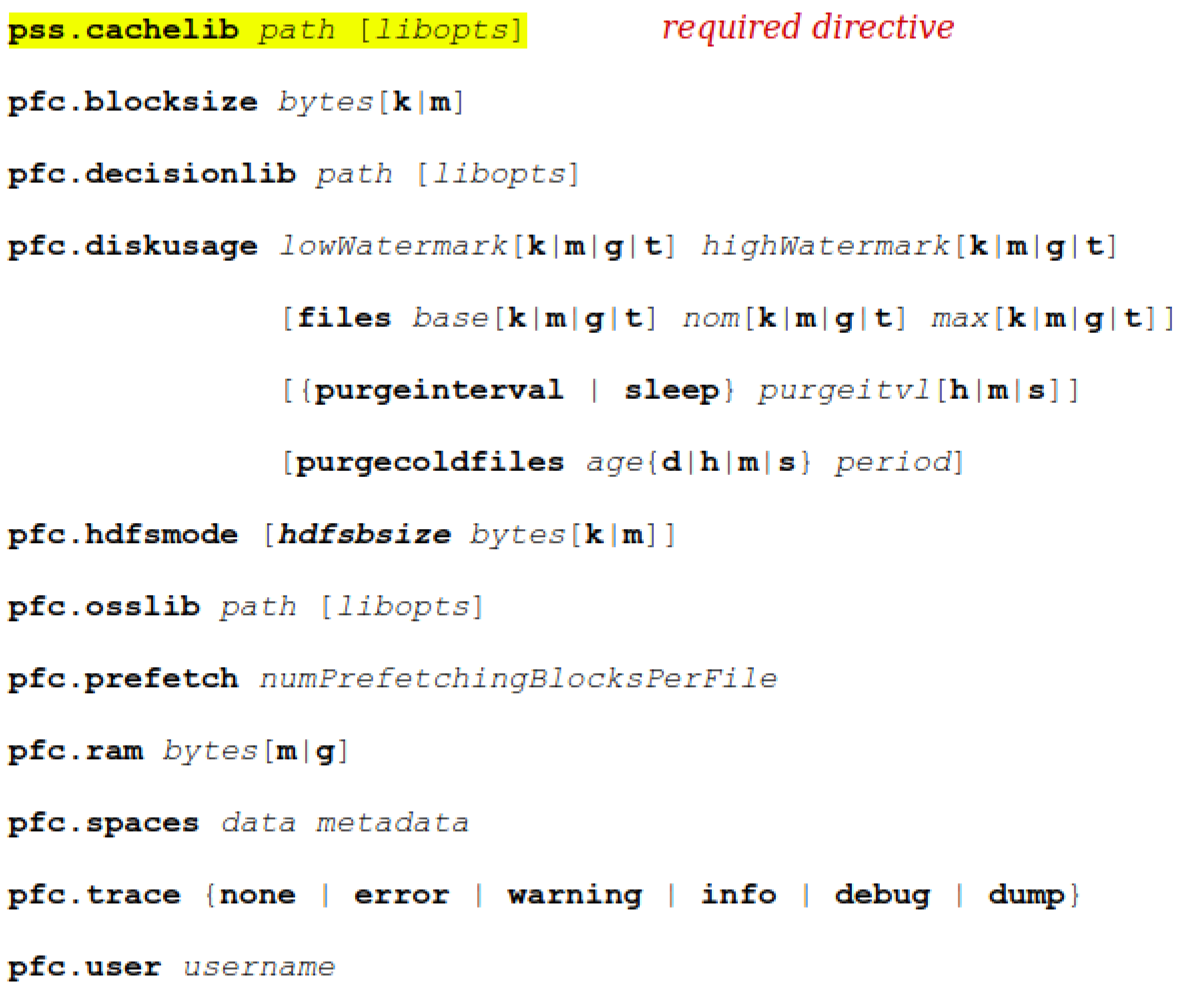
Low Watermark for Test Environment = 0.007 x 26 TB = 182 GB
High Watermark for Test Environment = 0.008 x 26 TB = 208 GB

Another reason to set High and Low Watermarks are to study the Cache Purging/Cache Rotation phenomenon.

The *pfc.trace* directive when set to *debug* shows more logging than the default value of *warning*.

The *pfc.prefetch* directive specifies the maximum number of pre-fetching blocks per file (Refer to *Illustration 18*). The default prefetch blocks is 10. 0 is set to disable pre-fetching.

The *pfc.ram* directive defines the maximum allowed RAM usage for caching proxy buffers which needs to be written to disk. Default is 256 MB.



```
pss.cachelib path [libopts]          required directive

pfc.blocksize bytes[k|m]

pfc.decisionlib path [libopts]

pfc.diskusage lowWatermark[k|m|g|t] highWatermark[k|m|g|t]

              [files base[k|m|g|t] nom[k|m|g|t] max[k|m|g|t]]

              [{purgeinterval | sleep} purgeitvl[h|m|s]]

              [purgecoldfiles age{d|h|m|s} period]

pfc.hdfsmode [hdfsbsize bytes[k|m]]

pfc.osslib path [libopts]

pfc.prefetch numPrefetchingBlocksPerFile

pfc.ram bytes[m|g]

pfc.spaces data metadata

pfc.trace {none | error | warning | info | debug | dump}

pfc.user username
```

Illustration 18: xrootd configuration file directives [48]

Types of Proxies

Direct Mode Proxy

Direct proxy setup is suitable to place before XRootD cluster. The *pss.origin* directive specifies the initial point of contact.

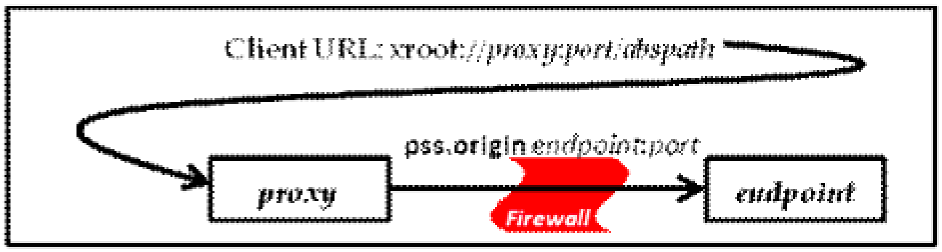


Illustration 19: Direct Mode Proxy Setup [48]

As per *Illustration 19*, the Client URI identifies the proxy cluster/server. The proxy then goes on to identify the data origin and the endpoint for the client, and goes on to perform the client request. The endpoint can either be a Server or a Cluster's Redirector. The Direct Mode Proxy system accesses files or data objects based on *all.export* directive of XRootD configuration.

We are using Direct Mode Proxy for our Xcache proxy setup (Refer *Illustration 16*).

Forwarding Mode Proxy

This type of setup requires the client to specify the actual endpoint to be used for request. This use-case is suitable wherein the client decides the actual endpoint that must be used.

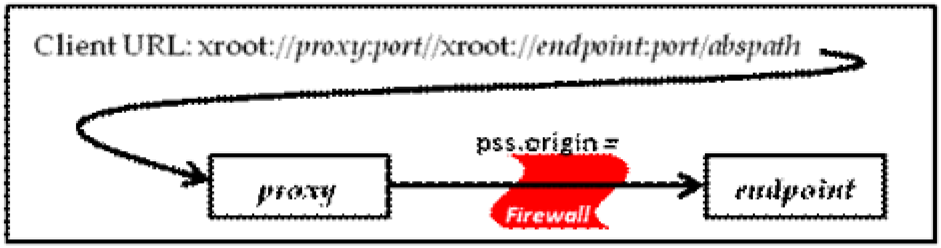

Client URL: xroot://proxy:port//xroot://endpoint:port/abspath
pss.origin =
proxy
Firewall
endpoint

Illustration 20: Forwarding Mode Proxy Setup [48]

As per *Illustration 20*, the client prefixes the destination URI with the proxy location. The proxy server's role is to contact the endpoint on behalf of the client. The client can specify any valid endpoint.

Forwarding File Path

The configuration of Forwarding Proxy is done in a way such that the URL suffix supplied by the client is taken as the actual path [48]. This denotes :

- Exporting some portions of the URL path
- Able to perform authorization operations on the specified URL

Exporting hostname : *all.export /xroot:/helloworld:1094/* or,

all.export /root:/helloworld:1094/

The client's destination should match the export specification.

The *pss.permit* directive can be used to restrict the set of target destinations.

Exporting Object IDs can be done by : *all.export ** or,

*all.export *?*

Xcache as a Reverse Proxy

In order to configure an HTTP reverse proxy layer for CVMFS on Xcache system, we need Machine A to server the contents of CVMFS repository (should also have CVMFS server tools installed); Machine B acting as Reverse Proxy (only XRootD installed here); and CVMFS client for mounting repository.

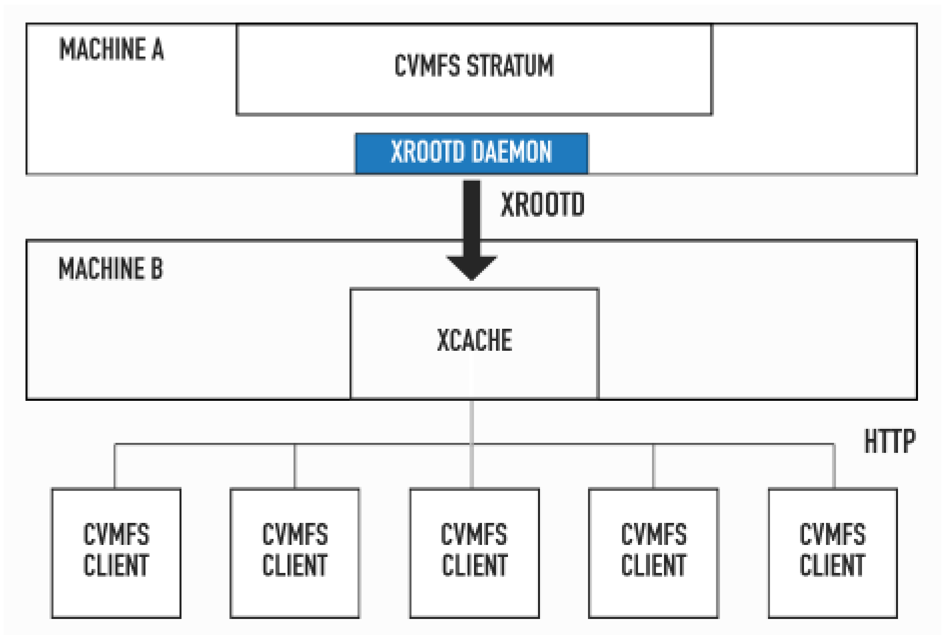
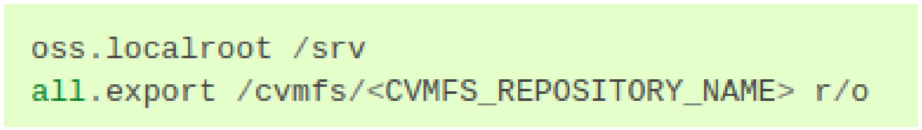


Illustration 21: Xcache system as Cache Layer between CVMFS and clients [49]

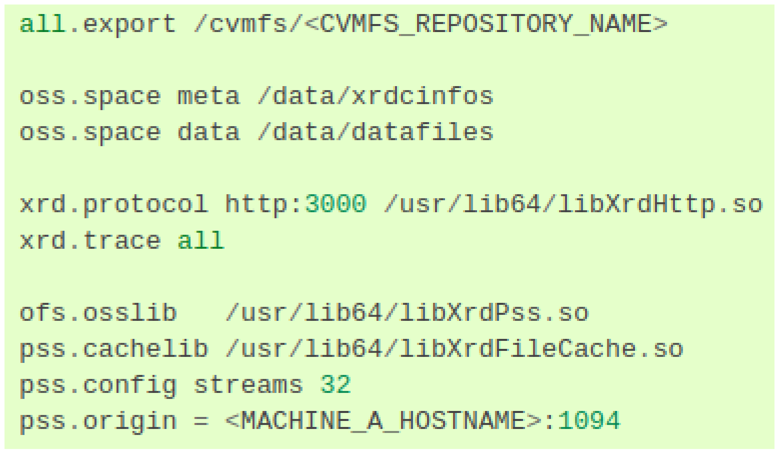
Configuration change in XRootD configuration file :



```
oss.localroot /srv
all.export /cvmfs/<CVMFS_REPOSITORY_NAME> r/o
```

Illustration 22: Configuration change for Reverse Proxy [49]

The Xcache instance running on Machine B can be pointed to *xrd* daemon pointing to the hostname of Machine A (Refer to *Illustration 23*).



```
all.export /cvmfs/<CVMFS_REPOSITORY_NAME>

oss.space meta /data/xrdcinfos
oss.space data /data/datafiles

xrd.protocol http:3000 /usr/lib64/libXrdHttp.so
xrd.trace all

ofs.osslib   /usr/lib64/libXrdPss.so
pss.cachelib /usr/lib64/libXrdFileCache.so
pss.config streams 32
pss.origin = <MACHINE_A_HOSTNAME>:1094
```

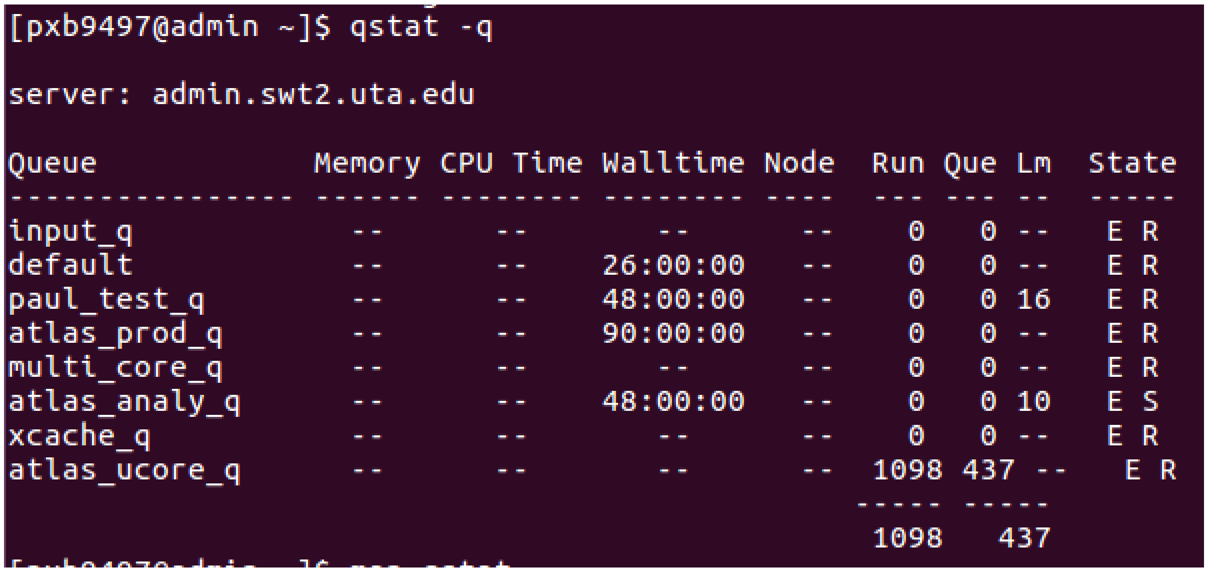
Illustration 23: For Reverse Proxy Xcache of Machine B points to host A [49]

The CVMFS client can use the Xcache Service Layer by altering the CVMFS Server URL as follows :

CVMFS_SERVER_URL=http://<Hostname-of-Machine-B>:3000/cvmfs/<CVMFS-REPO>

Chapter 5 : Analysis of Xcache

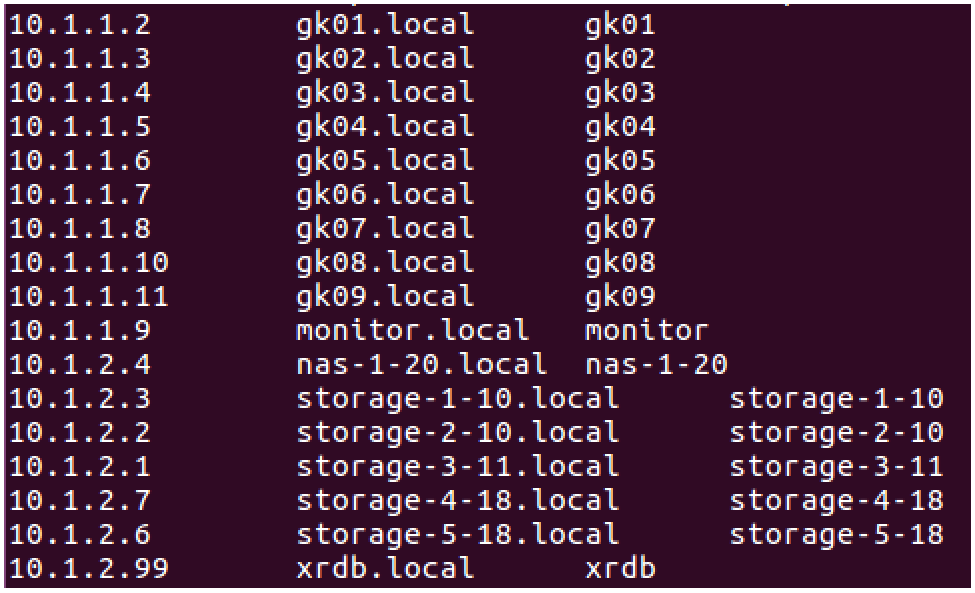
On the ARDC cluster ‘swt2.uta.edu’, the status of jobs, queues or batch servers can be determined as shown in *Illustration 24* :



```
[pxb9497@admin ~]$ qstat -q

server: admin.swt2.uta.edu

Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
input_q            --      --       --     --     0   0 --   E R
default            --      --    26:00:00  --     0   0 --   E R
paul_test_q        --      --    48:00:00  --     0   0 16   E R
atlas_prod_q       --      --    90:00:00  --     0   0 --   E R
multi_core_q       --      --       --     --     0   0 --   E R
atlas_analy_q      --      --    48:00:00  --     0   0 10   E S
xcache_q           --      --       --     --     0   0 --   E R
atlas_ucore_q      --      --       --     --  1098 437 --    E R
                                               ----- -----
                                                1098   437
```

Illustration 24: Jobs and Queues running in ARDC cluster

ARDC Cluster contains 110 compute nodes, 6 storage nodes and 1 Redirector. About 430 TB of storage is divided into 6 storage nodes.

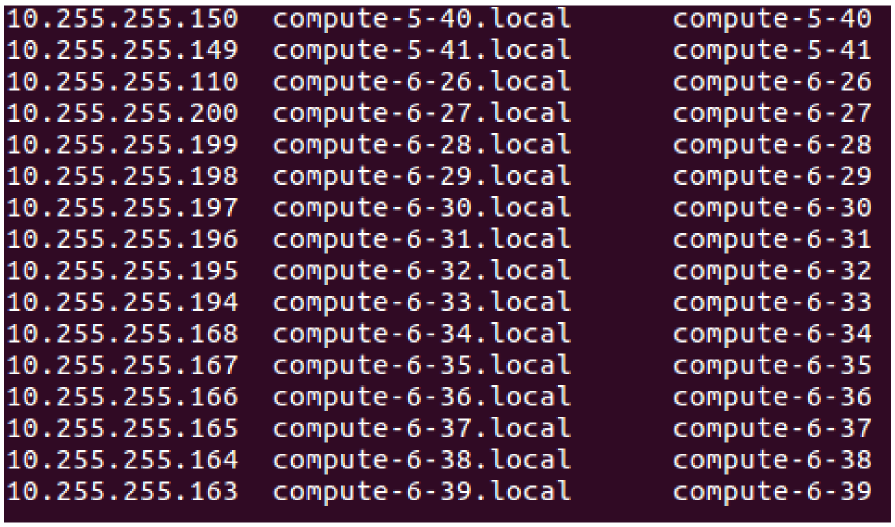


```
10.1.1.2        gk01.local      gk01
10.1.1.3        gk02.local      gk02
10.1.1.4        gk03.local      gk03
10.1.1.5        gk04.local      gk04
10.1.1.6        gk05.local      gk05
10.1.1.7        gk06.local      gk06
10.1.1.8        gk07.local      gk07
10.1.1.10       gk08.local      gk08
10.1.1.11       gk09.local      gk09
10.1.1.9        monitor.local   monitor
10.1.2.4        nas-1-20.local  nas-1-20
10.1.2.3        storage-1-10.local      storage-1-10
10.1.2.2        storage-2-10.local      storage-2-10
10.1.2.1        storage-3-11.local      storage-3-11
10.1.2.7        storage-4-18.local      storage-4-18
10.1.2.6        storage-5-18.local      storage-5-18
10.1.2.99       xrdb.local      xrdb
```

Illustration 25: Storage nodes, Machines and Redirector at ARDC

From *Illustration 25*, the machines are from *gk01* to *gk09*. *nas-1-20* is the NAS storage node. *Storage-1-10* to *storage-5-18* are the actual storage nodes. The Redirector of ARDC is *xrdb.local*.

The compute nodes are shown in *Illustration 26* as followed :



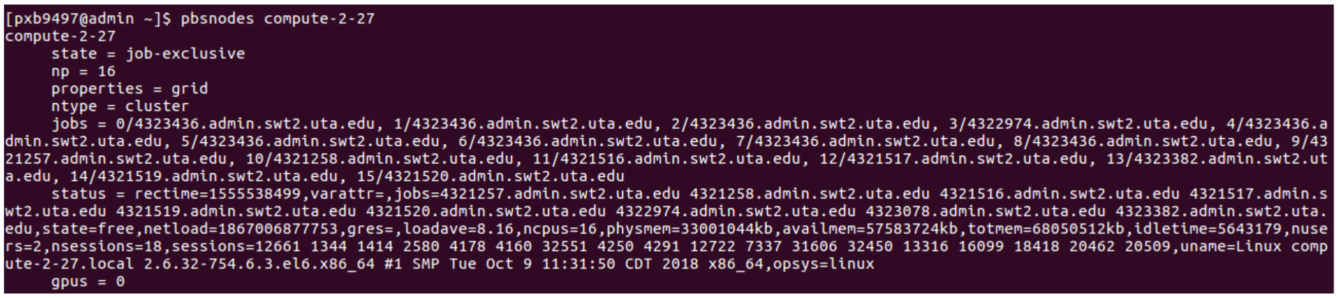
```
10.255.255.150  compute-5-40.local      compute-5-40
10.255.255.149  compute-5-41.local      compute-5-41
10.255.255.110  compute-6-26.local      compute-6-26
10.255.255.200  compute-6-27.local      compute-6-27
10.255.255.199  compute-6-28.local      compute-6-28
10.255.255.198  compute-6-29.local      compute-6-29
10.255.255.197  compute-6-30.local      compute-6-30
10.255.255.196  compute-6-31.local      compute-6-31
10.255.255.195  compute-6-32.local      compute-6-32
10.255.255.194  compute-6-33.local      compute-6-33
10.255.255.168  compute-6-34.local      compute-6-34
10.255.255.167  compute-6-35.local      compute-6-35
10.255.255.166  compute-6-36.local      compute-6-36
10.255.255.165  compute-6-37.local      compute-6-37
10.255.255.164  compute-6-38.local      compute-6-38
10.255.255.163  compute-6-39.local      compute-6-39
```

Illustration 26: Compute nodes at ARDC Cluster

The IP addresses with 129.107.***.*** belongs to the public network. For example Admin machine : *admin.swt2.uta.edu* and machines from *gk01.swt2.uta.edu* to *gk09.swt2.uta.edu* belong to the public network.

The IP addresses with 10.***.***.*** belong to the private network. For example the storage servers belong to the private network. The *.local* suffix of the resources also indicate that the resource is a part of private network.

The state of a node (either running or idle) is shown in *Illustration 27* :



```
[pxb9497@admin ~]$ pbsnodes compute-2-27
compute-2-27
     state = job-exclusive
     np = 16
     properties = grid
     ntype = cluster
     jobs = 0/4323436.admin.swt2.uta.edu, 1/4323436.admin.swt2.uta.edu, 2/4323436.admin.swt2.uta.edu, 3/4322974.admin.swt2.uta.edu, 4/4323436.a
dmin.swt2.uta.edu, 5/4323436.admin.swt2.uta.edu, 6/4323436.admin.swt2.uta.edu, 7/4323436.admin.swt2.uta.edu, 8/4323436.admin.swt2.uta.edu, 9/43
21257.admin.swt2.uta.edu, 10/4321258.admin.swt2.uta.edu, 11/4321516.admin.swt2.uta.edu, 12/4321517.admin.swt2.uta.edu, 13/4323382.admin.swt2.ut
a.edu, 14/4321519.admin.swt2.uta.edu, 15/4321520.admin.swt2.uta.edu
     status = rectime=1555538499,varattr=,jobs=4321257.admin.swt2.uta.edu 4321258.admin.swt2.uta.edu 4321516.admin.swt2.uta.edu 4321517.admin.s
wt2.uta.edu 4321519.admin.swt2.uta.edu 4321520.admin.swt2.uta.edu 4322974.admin.swt2.uta.edu 4323078.admin.swt2.uta.edu 4323382.admin.swt2.uta.
edu,state=free,netload=1867006877753,gres=,loadave=8.16,ncpus=16,physmem=33001044kb,availmem=57583724kb,totmem=68050512kb,idletime=5643179,nuse
rs=2,nsessions=18,sessions=12661 1344 1414 2580 4178 4160 32551 4250 4291 12722 7337 31606 32450 13316 16099 18418 20462 20509,uname=Linux comp
ute-2-27.local 2.6.32-754.6.3.el6.x86_64 #1 SMP Tue Oct 9 11:31:50 CDT 2018 x86_64,opsys=linux
     gpus = 0
```

Illustration 27: Status of a job running on a node

As per *Illustration 27*, an exclusive job is shown to be running on *compute-2-27* with 16 job slots. *Sessions* directive indicates the number of jobs running on the system.

In general every machine has 2 CPUs, and every CPU has varying number of cores. The newest CPUs have 10 core/CPU and the machines are configured to run hyperthreading. In Hyperthreading, there's two computing threads per core.

Xcache Architecture for Experimental Setup

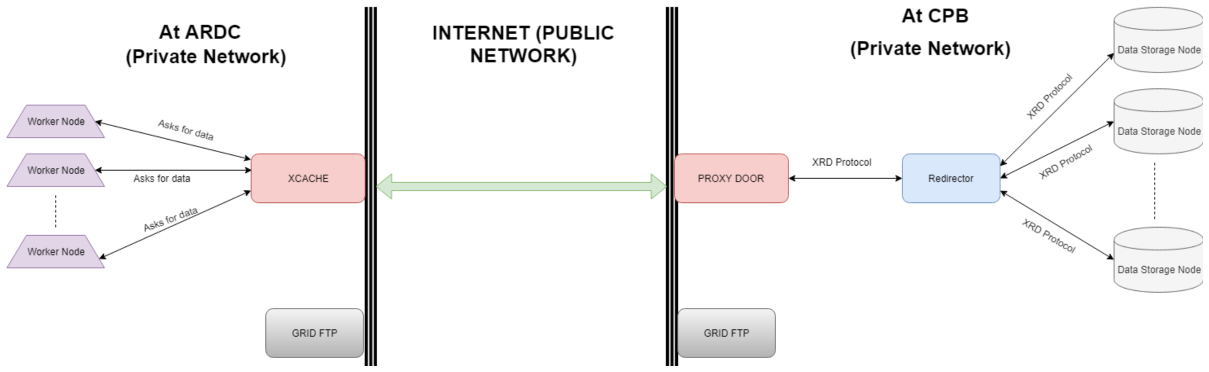


Illustration 28: Xcache Architecture with respect to Private-Public Network

As per our use case of configuring Xcache as a file-caching proxy server, we have chosen machine *gk03.swt2.uta.edu* in UTA_SWT2 (ARDC) cluster which would serve as Xcache.

The redirector at CPB has been chosen as machine *gk06.local*.

In ARDC cluster, Machine *gk04.local* is the workflow management system. It takes jobs from PanDA and loads them to the local scheduler.

Machines gk08 and gk09 are the Squid Caches.

At CPB,

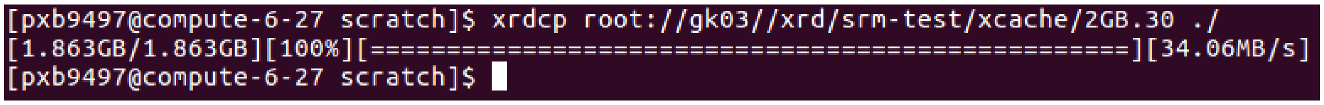
gk02 is the Torque Node (Local Scheduler).

gk04 and *gk05* are the Data Tranfer Nodes (DTN).

gk06 is the XRootD Node.

Xrdcp Command

The *xrdcp* utility copies one or more files from one location to another. The data source may be a local or remote file. Location of Xcache in ARDC is /xcache_1/xrd/srm-test/xcache/. For downloading a file through Xcache setup the command used is : xrdcp <file-source> <file-destination>

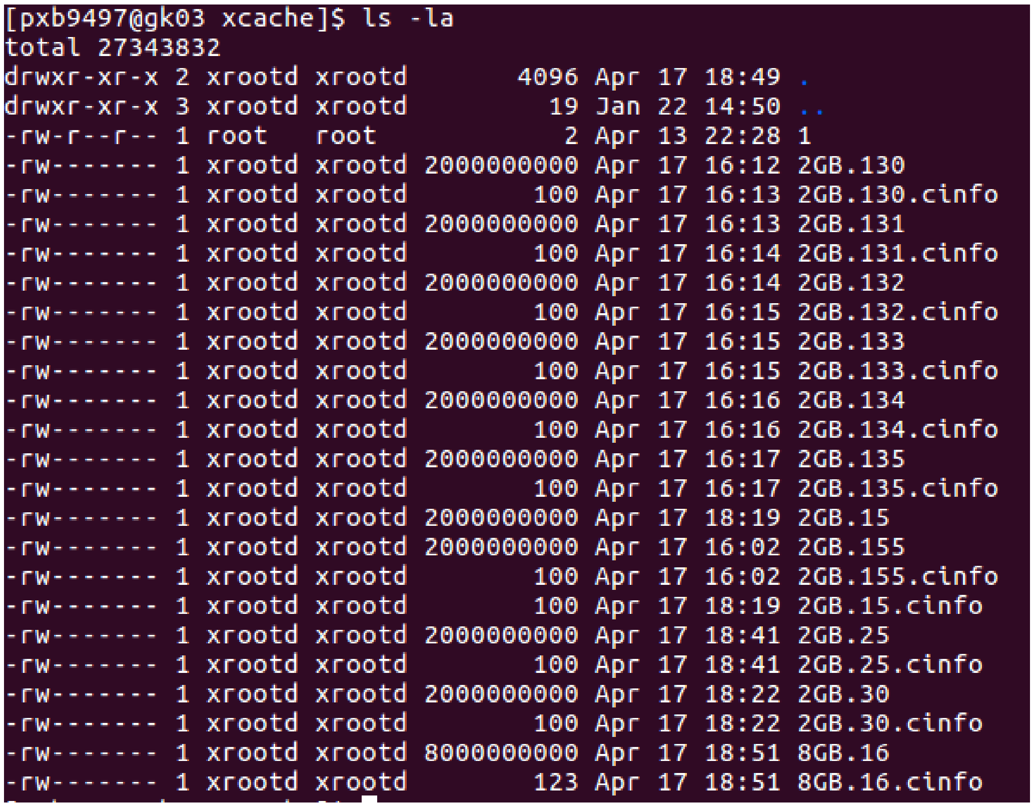


```
[pxb9497@compute-6-27 scratch]$ xrdcp root://gk03//xrd/srm-test/xcache/2GB.30 ./
[1.863GB/1.863GB][100%][==================================================][34.06MB/s]
[pxb9497@compute-6-27 scratch]$
```

Illustration 29: *xrdcp command to copy/download files*

Before starting download a file called <requested-filename>.cinfo is created at the Xcache path which contains the caching information encrypted in it.

Refer *Illustration 30* for the contents of the Xcache after files being cached :



```
[pxb9497@gk03 xcache]$ ls -la
total 27343832
drwxr-xr-x 2 xrootd xrootd       4096 Apr 17 18:49 .
drwxr-xr-x 3 xrootd xrootd         19 Jan 22 14:50 ..
-rw-r--r-- 1 root   root            2 Apr 13 22:28 1
-rw------- 1 xrootd xrootd 2000000000 Apr 17 16:12 2GB.130
-rw------- 1 xrootd xrootd        100 Apr 17 16:13 2GB.130.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 16:13 2GB.131
-rw------- 1 xrootd xrootd        100 Apr 17 16:14 2GB.131.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 16:14 2GB.132
-rw------- 1 xrootd xrootd        100 Apr 17 16:15 2GB.132.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 16:15 2GB.133
-rw------- 1 xrootd xrootd        100 Apr 17 16:15 2GB.133.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 16:16 2GB.134
-rw------- 1 xrootd xrootd        100 Apr 17 16:16 2GB.134.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 16:17 2GB.135
-rw------- 1 xrootd xrootd        100 Apr 17 16:17 2GB.135.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 18:19 2GB.15
-rw------- 1 xrootd xrootd 2000000000 Apr 17 16:02 2GB.155
-rw------- 1 xrootd xrootd        100 Apr 17 16:02 2GB.155.cinfo
-rw------- 1 xrootd xrootd        100 Apr 17 18:19 2GB.15.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 18:41 2GB.25
-rw------- 1 xrootd xrootd        100 Apr 17 18:41 2GB.25.cinfo
-rw------- 1 xrootd xrootd 2000000000 Apr 17 18:22 2GB.30
-rw------- 1 xrootd xrootd        100 Apr 17 18:22 2GB.30.cinfo
-rw------- 1 xrootd xrootd 8000000000 Apr 17 18:51 8GB.16
-rw------- 1 xrootd xrootd        123 Apr 17 18:51 8GB.16.cinfo
```

Illustration 30: *Contents of the Xcache at ARDC*

Every time a file gets cached in the Xcache system, an entry is added to the .cinfo file. A .cinfo file is created when the file is cached for the first time, and subsequently everytime the file gets served out of the Xcache, an entry gets added to the .cinfo file.

Xcache Storage Experimental Setup

According to *Illustration 16,* the directive *pfc.diskusage* followed by the Low and High Watermark values for our Xcache storage experiment has been set to :

Low Watermark = 120 GB
High Watermark = 164 GB

We automated the cache fill-up process to study the Cache Purging or Rollover and observed in *Illustration 31* for the Cache Rollover. Files of size 1GB, 2GB, 4GB and 8 GB were downloaded through Xcache over 3.5 hours to observe the storage behavior, and High and Low Watermarks graphically.

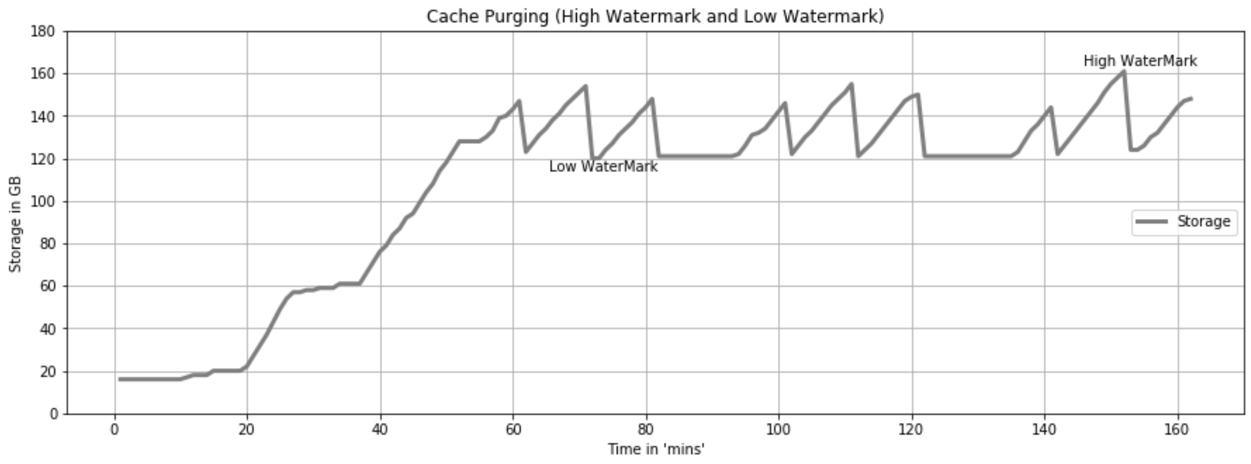
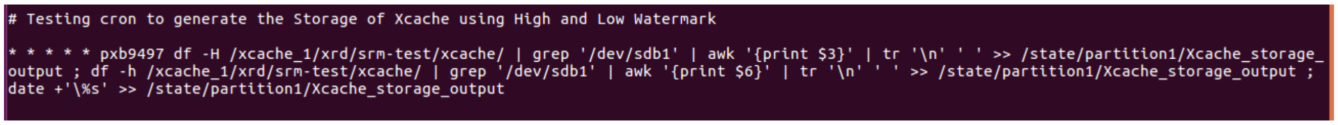


Illustration 31: Xcache storage behavior depicting High and Low Watermark

Cron job used to monitor the Xcache storage every second under load (files requested to be downloaded) is shown in *Illustration 32*.

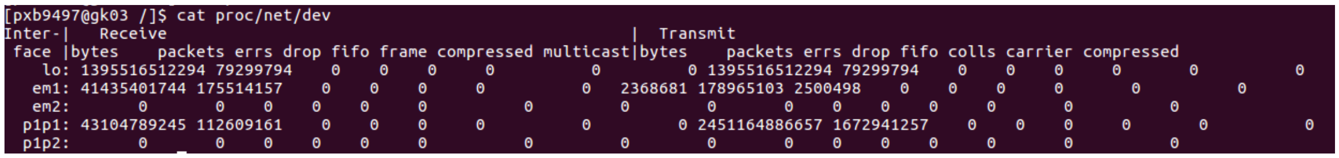


```
# Testing cron to generate the Storage of Xcache using High and Low Watermark

* * * * * pxb9497 df -H /xcache_1/xrd/srm-test/xcache/ | grep '/dev/sdb1' | awk '{print $3}' | tr '\n' ' ' >> /state/partition1/Xcache_storage_
output ; df -h /xcache_1/xrd/srm-test/xcache/ | grep '/dev/sdb1' | awk '{print $6}' | tr '\n' ' ' >> /state/partition1/Xcache_storage_output ;
date +'\%s' >> /state/partition1/Xcache_storage_output
```

Illustration 32: Cron job to monitor the Xcache storage

Since the Xcache system is a proxy file caching system caching the file blocks (at the file-block level), we have decided to monitor the network traffic flowing in and out of the system. The download of a data file takes place in contiguous file-blocks streaming mechanism.

So in order to monitor the network traffic in and out of Xcache system, the linux command *cat /proc/net/dev* provides the bytes received and transmitted out of Machine *gk03.local* at a given point of time.

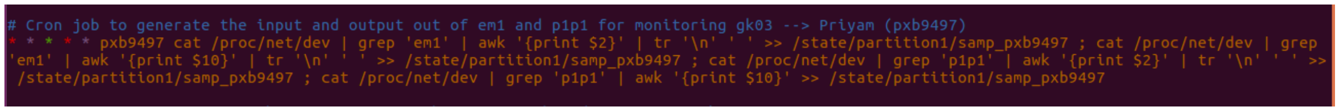


```
[pxb9497@gk03 /]$ cat proc/net/dev
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo colls carrier compressed
    lo: 1395516512294 79299794    0    0    0     0          0         0 1395516512294 79299794    0    0    0     0       0          0
   em1: 41435401744 175514157    0    0    0     0          0   2368681 178965103 2500498    0    0    0     0       0          0
   em2:        0        0    0    0    0     0          0         0        0        0    0    0    0     0       0          0
  p1p1: 43104789245 112609161    0    0    0     0          0         0 2451164886657 1672941257    0    0    0     0       0          0
  p1p2:        0        0    0    0    0     0          0         0        0        0    0    0    0     0       0          0
```

Illustration 33: Bytes In and Bytes Out of Xcache System at a given point

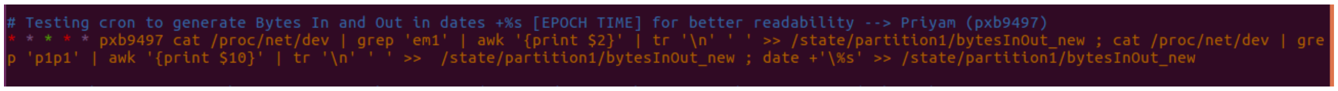
From *Illustration 33*, the Received Bytes at Network Interface Card (NIC) *em1* is the input (received) bytes at Machine *gk03*(Xcache system), and Transmitted Bytes at NIC *p1p1* is the output (transmitted) bytes out of Machine *gk03*.

The cron job shown in *Illustration 34*, is responsible for parsing the *dev* file and capturing the values of input bytes to em1 and output out of p1p1 every minute.



```
# Cron job to generate the input and output out of em1 and p1p1 for monitoring gk03 --> Priyam (pxb9497)
* * * * * pxb9497 cat /proc/net/dev | grep 'em1' | awk '{print $2}' | tr '\n' ' ' >> /state/partition1/samp_pxb9497 ; cat /proc/net/dev | grep
'em1' | awk '{print $10}' | tr '\n' ' ' >> /state/partition1/samp_pxb9497 ; cat /proc/net/dev | grep 'p1p1' | awk '{print $2}' | tr '\n' ' ' >>
 /state/partition1/samp_pxb9497 ; cat /proc/net/dev | grep 'p1p1' | awk '{print $10}' >> /state/partition1/samp_pxb9497
```

Illustration 34: Cron job running every minute to parse and capture Bytes In and Bytes Out of Machine gk03



```
# Testing cron to generate Bytes In and Out in dates +%s [EPOCH TIME] for better readability --> Priyam (pxb9497)
* * * * * pxb9497 cat /proc/net/dev | grep 'em1' | awk '{print $2}' | tr '\n' ' ' >> /state/partition1/bytesInOut_new ; cat /proc/net/dev | gre
p 'p1p1' | awk '{print $10}' | tr '\n' ' ' >>  /state/partition1/bytesInOut_new ; date +'\%s' >> /state/partition1/bytesInOut_new
```

Illustration 35: Cron job with the same functionality as Illustration 34 but with Time in Epoch for ease of parsing

Experiment for ‘Caching with Xcache’ VS ‘No Caching’

The aim of this experiment was to observe the network bandwidth of Xcache system (bytes/sec converted to GB/min in graphs as seen in Illustrations 36 and 37) with and without caching. The objective was also to find how faster would the files be downloaded if served straight out of Xcache rather than getting fetched out of storage servers from CPB, in a near-hypothetical situation of 100% caching.

No Caching Scenario

For the first setup to test for ‘No caching scenario’, the cache was kept empty, and 30 files 4GB each were downloaded into */state/partition1/scratch* from Machine *compute-6-27*. The waveform is shown in *Illustration 36* as follows :

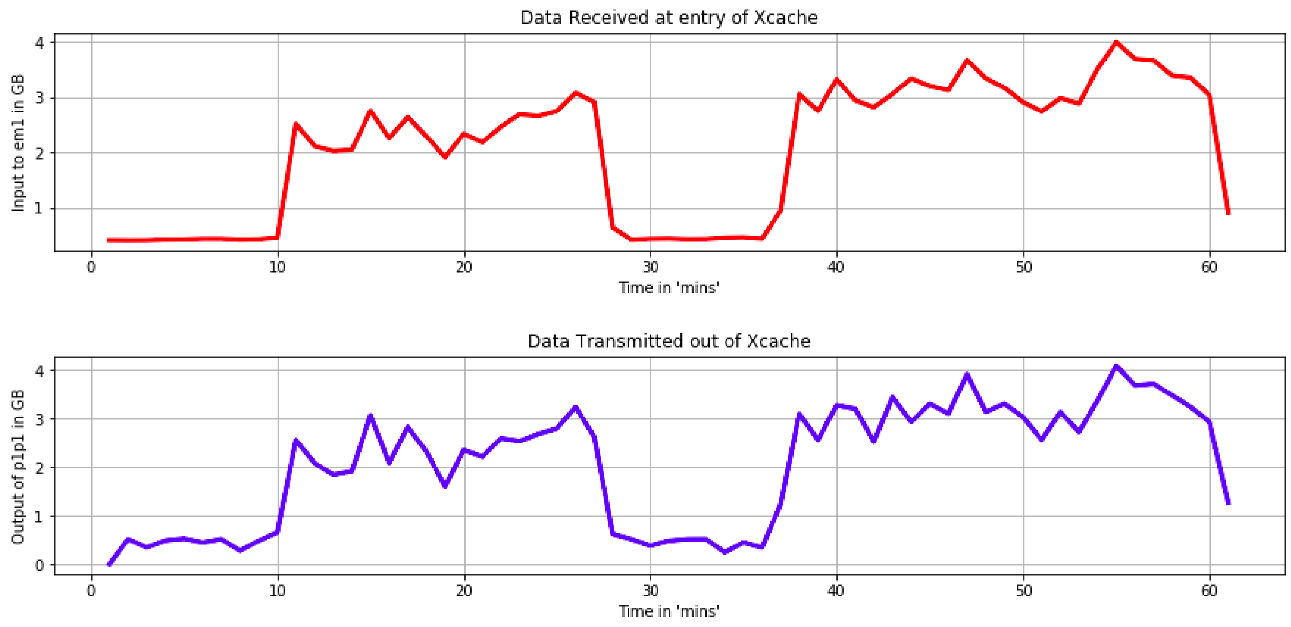


Illustration 36: Input and Output Bandwidth of Xcache system without Caching enabled

Caching Scenario

A near-hypothetical scenario of all files being served out of Xcache is being tested to observe the Bandwidth and download performance (time in ‘minutes’) of Xcache system.

The waveform for this case is shown in *Illustration 37* as follows :

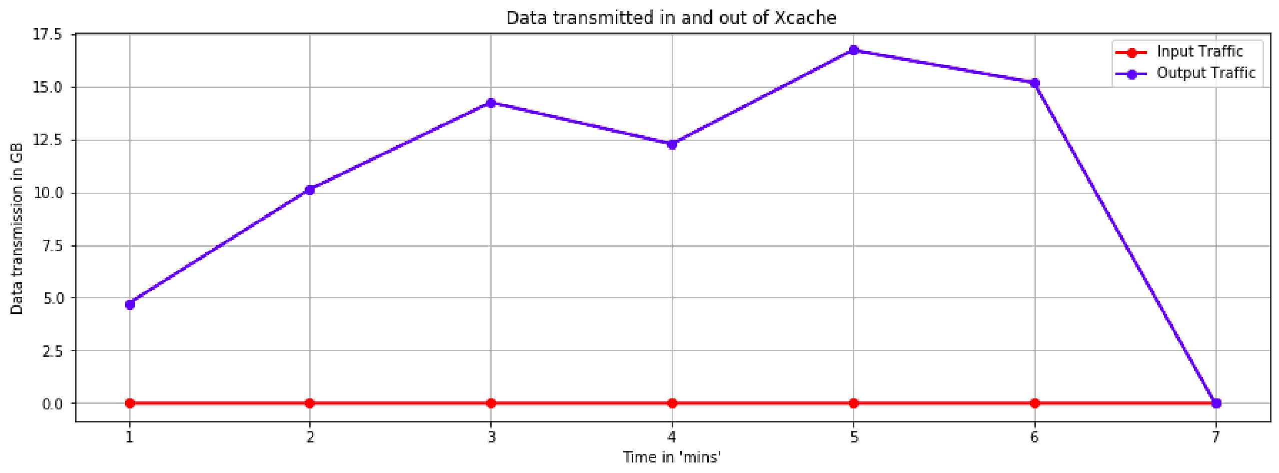


Illustration 37: Input and Output Bandwidth of Xcache system with Caching

Comparison of Caching VS No Caching

The bandwidth observed for Caching enabled peaks at around 17 GBps compared to the peak of around 4 GB/sec in case of No Caching situation. Moreover, the time taken to download same amount of files during Caching takes around 7 minutes, while No Caching takes around 60 minutes.

So, for this test with 30 files 4GB each :

- Peak Bandwidth for Caching is more than 4 times of No Caching.
- Total Time taken to download files from Cache is about 88.3% faster

Statistically Sampled Real-time Network Bandwidth

SCENARIO 1 : 300 files of sizes 1GB, 2GB and 4GB randomly pulled

For this experimental setup, 300 random files of sizes 1GB, 2GB and 4GB have been randomly asked from the storage servers at CPB.

The storage space of Xcache was capped at 208 GB (High Watermark), and the test was started with empty Xcache. Frequency of the input and output data collection is per minute.

The test went on for about 5 hours and considerable Bandwidth saving was observed for the files already residing in Xcache. Refer to Illustration 38 for the Bandwidth monitoring of the test :

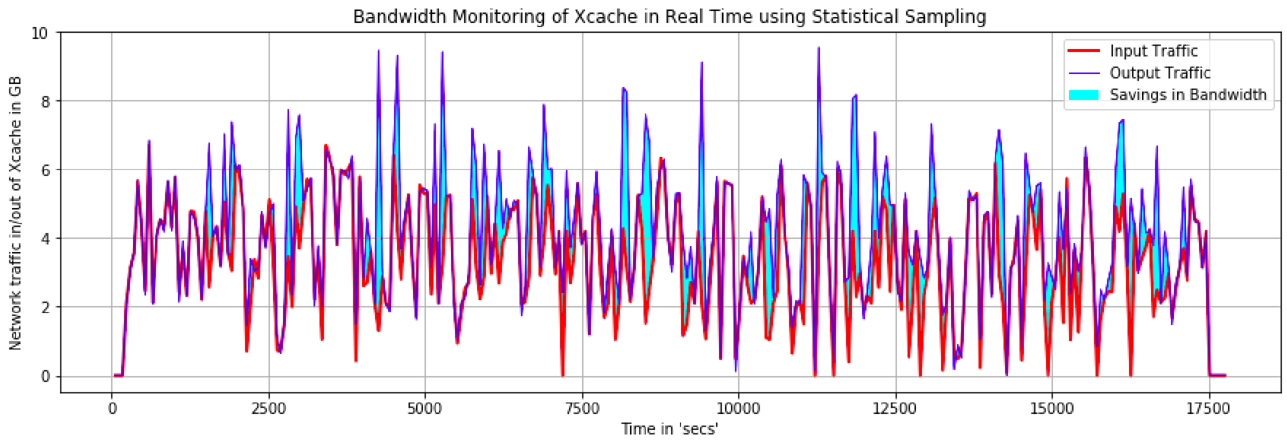


Illustration 38: Scenario 1 : Statistically sampled data files showing savings in Bandwidth

The shaded area (shown in ‘cyan’ color) between red and blue line-marker shows the scenarios where a file got served out of Xcache rather than from a CPB storage server.

So over a period of about 5 hours, as we can see from *Illustration 38*, considerable caching has been observed.

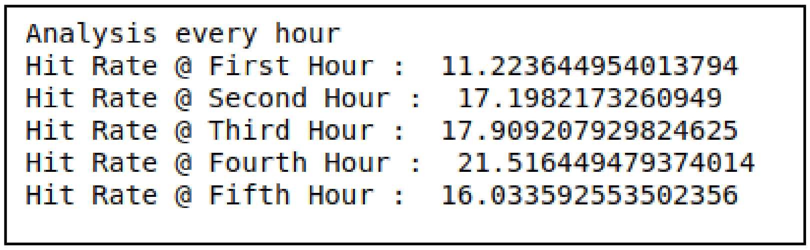
Observation

The Hit Rate is defined as the number of cache hits over the total number of hits and misses. In our scenario, we're measuring the bytes input and output out of the Xcache.

Hit Rate = $(\text{Bytes}_{\text{OutputXcache}} – \text{Bytes}_{\text{InputXcache}}) / (\text{Bytes}_{\text{InputXcache}})$

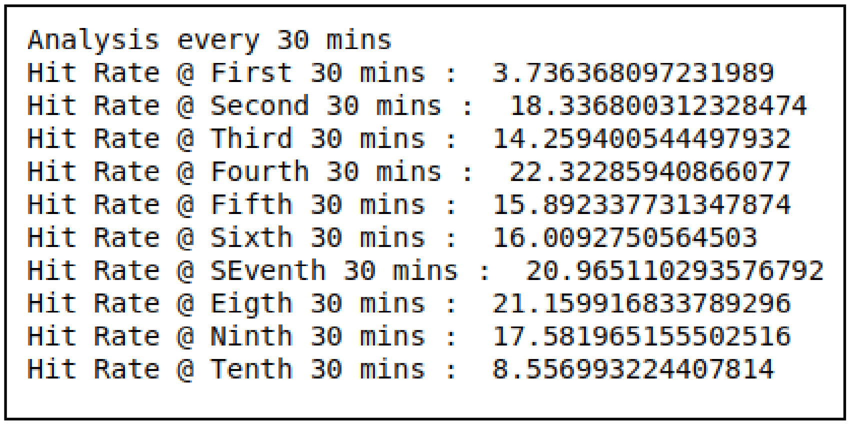
As per *Illustration 38*, the experiment ran for 296 minutes, with caching shown in the 'cyan' shaded area has shown a Average Hit Rate of about 16%. In other words, over the period of about 5 hours downloading 300 random files, the number of file bytes (in terms of network traffic) served out of Xcache is 16%.

An analysis of Hit Rate per hour and every 30 minutes are shown as follows :



```
Analysis every hour
Hit Rate @ First Hour :  11.223644954013794
Hit Rate @ Second Hour :  17.1982173260949
Hit Rate @ Third Hour :  17.909207929824625
Hit Rate @ Fourth Hour :  21.516449479374014
Hit Rate @ Fifth Hour :  16.033592553502356
```

Illustration 39: Finding Hit Rate metric per hour for the Xcache system



```
Analysis every 30 mins
Hit Rate @ First 30 mins :  3.736368097231989
Hit Rate @ Second 30 mins :  18.336800312328474
Hit Rate @ Third 30 mins :  14.259400544497932
Hit Rate @ Fourth 30 mins :  22.32285940866077
Hit Rate @ Fifth 30 mins :  15.892337731347874
Hit Rate @ Sixth 30 mins :  16.0092750564503
Hit Rate @ SEventh 30 mins :  20.965110293576792
Hit Rate @ Eigth 30 mins :  21.159916833789296
Hit Rate @ Ninth 30 mins :  17.581965155502516
Hit Rate @ Tenth 30 mins :  8.556993224407814
```

Illustration 40: Finding Hit Rate metric every 30 minutes for Xcache system

The idea is to observe the Hit Rate over time.

SCENARIO 2 : 500 files of sizes 1GB, 2GB and 4GB randomly pulled

For this experimental setup, 500 random files of sizes 1GB, 2GB and 4GB have been randomly asked from the storage servers at CPB.

The storage space of Xcache was capped at 208 GB (High Watermark), and the test was started with empty Xcache. Frequency of the input and output data collection is per minute.

The test went on for about 5 hours and considerable Bandwidth saving was observed for the files already residing in Xcache. Refer to Illustration 41 for the Bandwidth monitoring of the test :

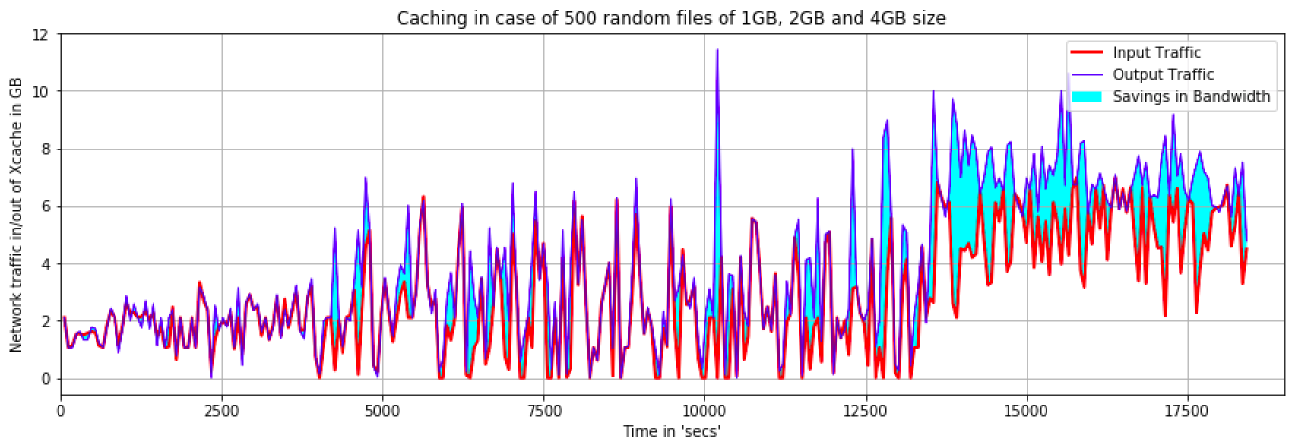


Illustration 41: Scenario 2 : Statistically sampled data files showing savings in Bandwidth

The shaded area (shown in ‘cyan’ color) between red and blue line-marker shows the scenarios where a file got served out of Xcache rather than from a CPB storage server.

So over a period of about 5 hours, as we can see from *Illustration 41*, considerable caching has been observed. Caching (Savings in Bandwidth) is observed more towards the end of the test, especially at the last 1 hour, after the cache is well warmed up.

Bash Script To Automate File Download

```
#! /bin/bash
declare -i rand_file
declare -i rand_name
for i in {1..100} # depends on the number of files to be downloaded
do
arr=(1 2 4)
file_size=${arr[$((RANDOM%3))]} # randomly selects 1,2 or 4
rand_file=1+RANDOM%100    # generates random number between 1 and 100
rand_name=22+RANDOM%10000
echo "Iter : $i.. Copying $file_size""GB.$rand_file"
xrdcp root://gk03//xrd/srm-test/xcache/$file_size"GB.$rand_file"
/state/partition1/scratch/$file_size"GB.$rand_file.$rand_name"
date +%s              # Prints out the Epoch Date-time in terminal
done
```

Observation

The Hit Rate is defined as the number of cache hits over the total number of hits and misses. In our scenario, we're measuring the bytes input and output out of the Xcache. So, the formula becomes :

Hit Rate = $(\text{Bytes}_{\text{OutputXcache}} - \text{Bytes}_{\text{InputXcache}}) / (\text{Bytes}_{\text{InputXcache}})$

Average Hit Rate = $\sum \mathit{Calculated\ Hit\ Rate\ for\ all\ data\ points} / \mathit{Total\ number\ of\ data\ points}$

As per *Illustration 41*, the experiment ran for 292 minutes, with caching shown in the 'cyan' shaded area has shown a Average Hit Rate of about 17%. In other words, over the period of about 5 hours downloading 500 random files, the number of file bytes (in terms of network traffic) served out of Xcache is 17%.

Conclusion

Both tests for Scenarios 1 and 2 ran for around 300 minutes, even though the number of files were 300 and 500 respectively. The same run-time attributing to the fact that the tests ran under different bandwidth conditions.

For both scenarios, the Average Hit Rate is around 16%. This is due to the fact that a steady state of Avg. Hit Rate is obtained while downloading files above a certain number. In here, the steady state of Hit Rate is already obtained while downloading 300 files.

Experiment to compare the Avg. Hit Rates while downloading different number of files (Size of each file 2GB)

Test Setup 1.1

The aim is to download 20 random files from the storage servers of CPB. Each file is 2GB in size and the network input-output data is collected once every minute. The disk-cache size of Xcache system is set to 208 GB, and the download starts by keeping the cache empty.

Test Setup 1.2

The aim is to download 50 random files from the storage servers of CPB. Each file is 2GB in size and the network input-output data is collected once every minute. The disk-cache size of Xcache system is set to 208 GB, and the download starts by keeping the cache empty.

Test Setup 1.3

The aim is to download 100 random files from the storage servers of CPB. Each file is 2GB in size and the network input-output data is collected once every minute. The disk-cache size of Xcache system is set to 208 GB, and the download starts by keeping the cache empty.

Study of Bandwidth and Caching Behavior

For Test Setup 1.1, the bandwidth and caching behavior is shown in Illustration 42 :

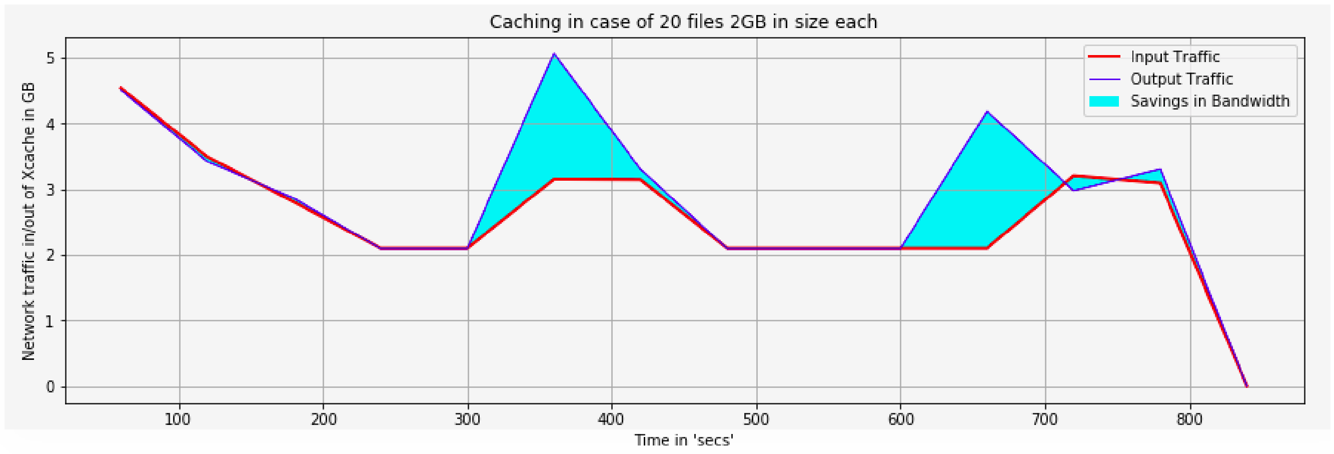


Illustration 42: Savings in bandwidth with respect to Input and Output Network traffic for downloading 20 random files 2GB each

For Test Setup 1.2, the bandwidth and caching behavior is shown in Illustration 43 :

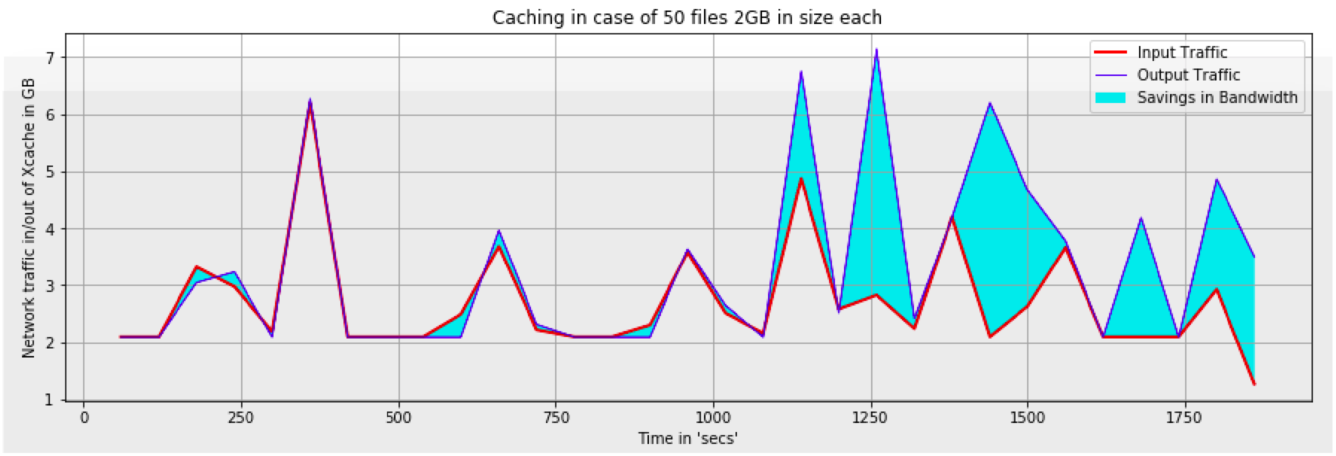


Illustration 43: Savings in bandwidth with respect to Input and Output Network traffic for downloading 50 random files 2GB each

For Test Setup 1.3, the bandwidth and caching behavior is shown in Illustration 44 :

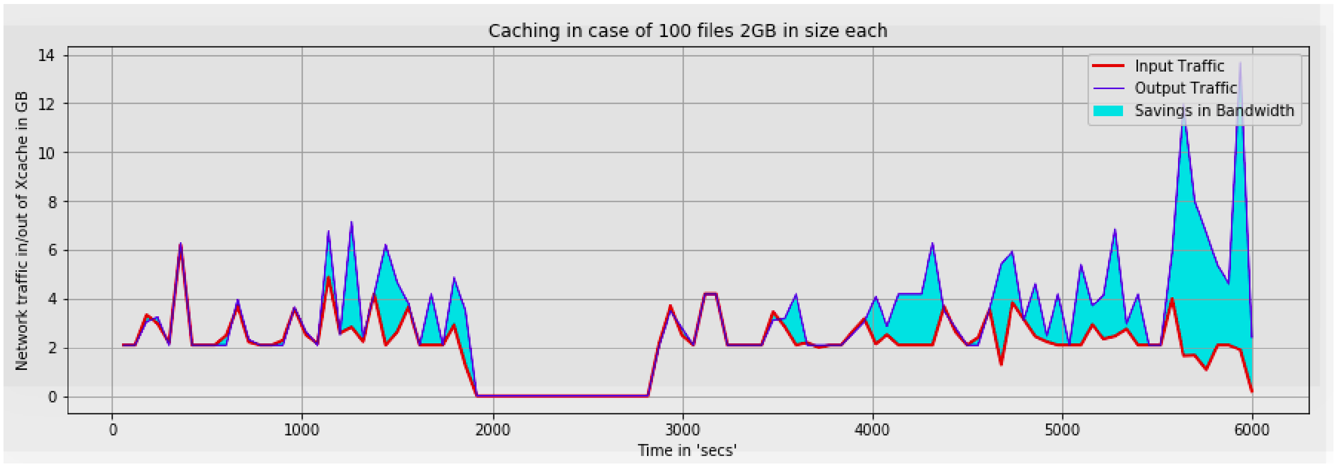


Illustration 44: Savings in bandwidth with respect to Input and Output Network traffic for downloading 100 random files 2GB each

Observation and Conclusion

The Average Hit Rates have been observed to increase in ascending order for scenarios 1.1, 1.2 and 1.3 respectively, i.e. while downloading 20, 50 and 100 files respectively.

$\text{Hit Rate}_{20\text{files}} = 4\ \% < \text{Hit Rate}_{50\text{files}} = 12\% < \text{Hit Rate}_{100\text{files}} = 17\%$

Experiment to compare the Avg. Hit Rates while downloading different number of files (Size of each file 8GB)

Test Setup 2.1

The aim is to download 20 random files from the storage servers of CPB. Each file is 8GB in size and the network input-output data is collected once every minute. The disk-cache size of Xcache system is set to 208 GB, and the download starts by keeping the cache empty.

Test Setup 2.2

The aim is to download 50 random files from the storage servers of CPB. Each file is 8GB in size and the network input-output data is collected once every minute. The disk-cache size of Xcache system is set to 208 GB, and the download starts by keeping the cache empty.

Test Setup 2.3

The aim is to download 100 random files from the storage servers of CPB. Each file is 8GB in size and the network input-output data is collected once every minute. The disk-cache size of Xcache system is set to 208 GB, and the download starts by keeping the cache empty.

Study of Bandwidth and Caching Behavior

For Test Setup 2.1, the bandwidth and caching behavior is shown in Illustration 45 :

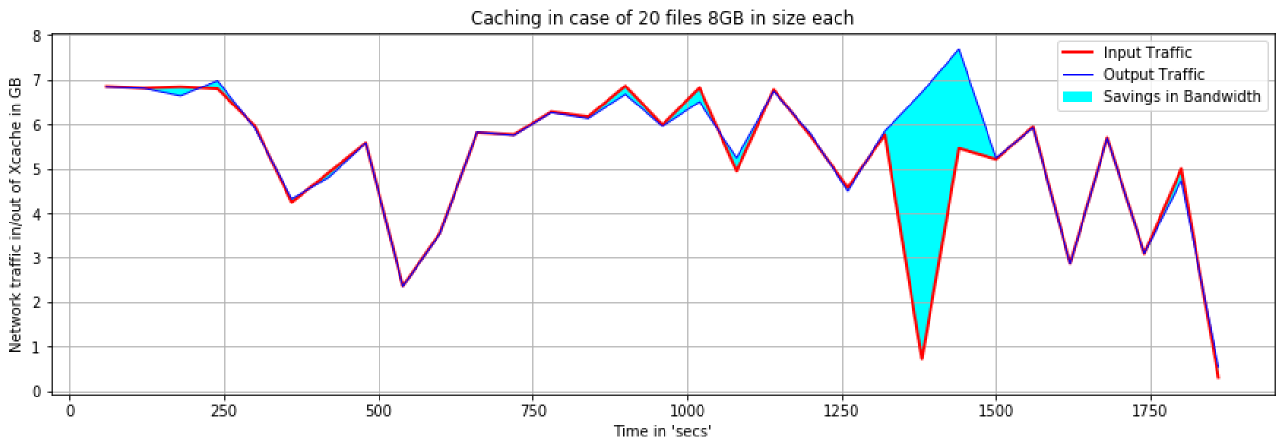


Illustration 45: Savings in bandwidth with respect to Input and Output Network traffic for downloading 20 random files 8GB each

For Test Setup 2.2, the bandwidth and caching behavior is shown in Illustration 46 :

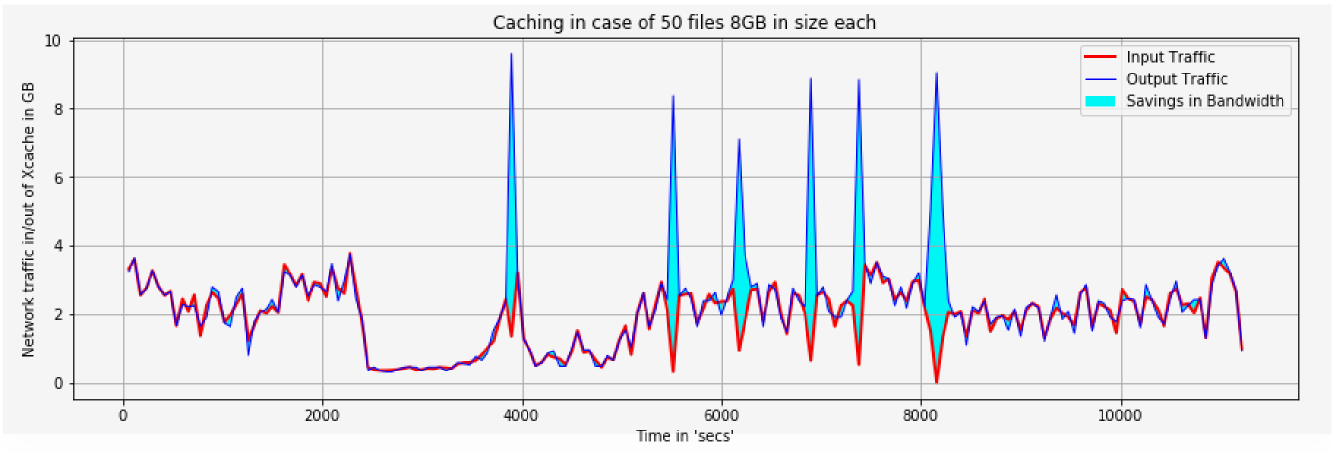


Illustration 46: Savings in bandwidth with respect to Input and Output Network traffic for downloading 50 random files 8GB each

For Test Setup 2.3, the bandwidth and caching behavior is shown in Illustration 47 :

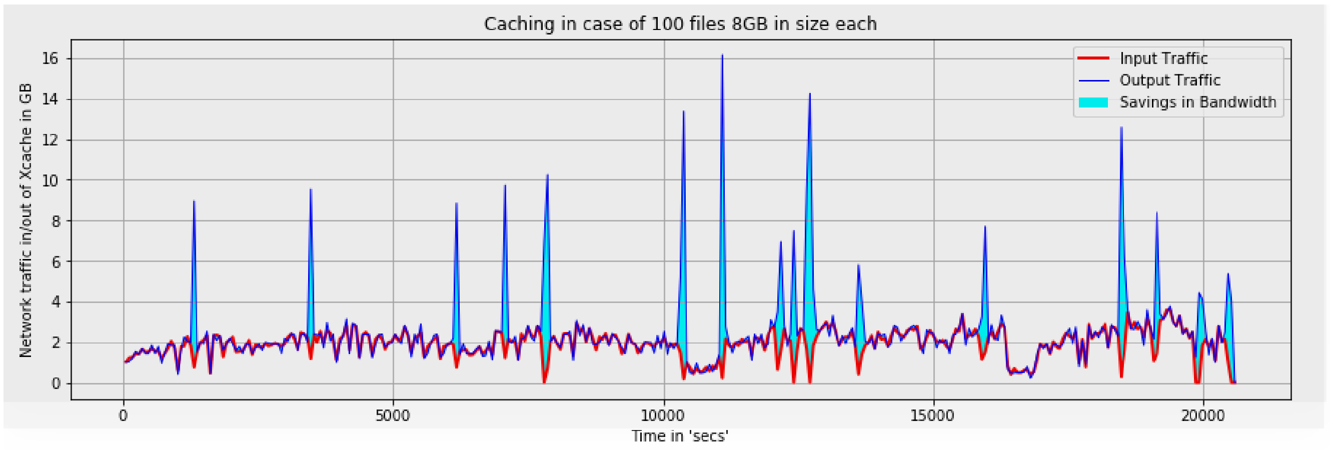


Illustration 47: Savings in bandwidth with respect to Input and Output Network traffic for downloading 100 random files 8GB each

Observation and Conclusion

While we observed a trend in Caching through the Avg. Hit Rate in cases 1.1, 1.2 and 1.3, there's no trend observed for scenarios 2.1, 2.2 and 2.3.

Avg. Hit Rate$_{20files}$ = around 4%

Avg. Hit Rate$_{50files}$ = around 2%

Avg. Hit Rate$_{100files}$ = around 4%

Sensitivity Test

In order to increase the accuracy of caching results, instead of collecting the Network Input-Output data every minute, now the frequency of data collection is every second.

A few parameters which are assumed to govern the Avg. Hit Rate are ‘Number of files downloaded’, ‘Size of each file being downloaded’ and ‘Size of the Cache’. A ‘Sensitivity Test’ has been performed to measure the Avg. Hit Rate for the varying parameters of ‘Number of files being downloaded’, ‘Size of each file being downloaded’ and ‘Size of the Cache’, thereby trying to establish a relation between the parameters and the Hit Rate.

Test Setup 3.1

Number of files being downloaded : 100

Size of each file : 1GB

Cache Size : 200 GB

Test Setup 3.2

Number of files being downloaded : 100

Size of each file : 1GB

Cache Size : 100 GB

Test Setup 3.3

Number of files being downloaded : 100

Size of each file : 2GB

Cache Size : 200 GB

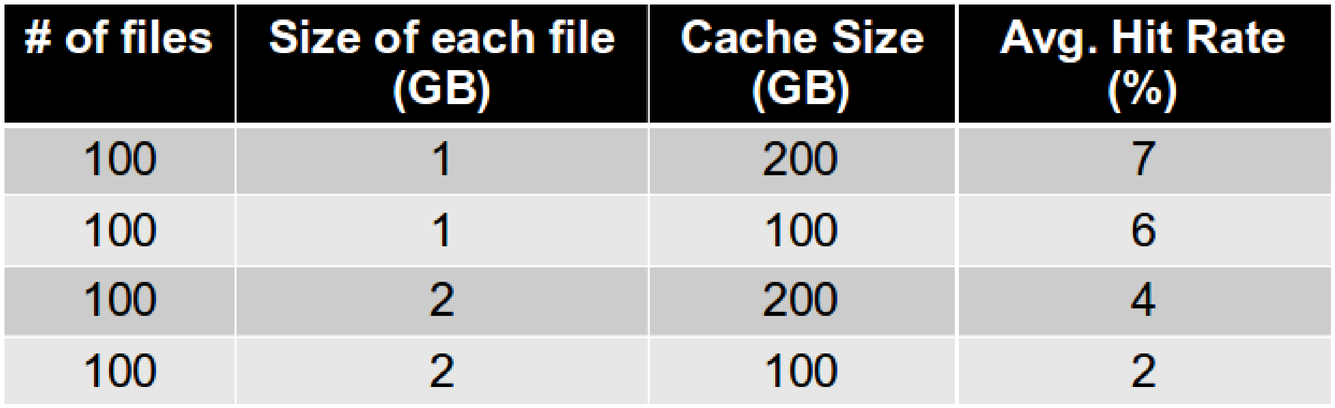
Test Setup 3.4

Number of files being downloaded : 100

Size of each file : 2GB

Cache Size : 100 GB

The results of the sensitivity test for Scenario # 3 is shown in Illustration 48 :



# of files	Size of each file (GB)	Cache Size (GB)	Avg. Hit Rate (%)
100	1	200	7
100	1	100	6
100	2	200	4
100	2	100	2

Illustration 48: Sensitivity Test results for Scenario # 3

Observation for Scenario # 3

A trend is observed for decreasing Avg. Hit Rate with the increase in the file size and subsequently decrease in Cache Size. Test Setup 4 would evidently show how the Avg. Hit Rate behaves while downloading 200 files.

Test Setup 4.1

Number of files being downloaded : 200

Size of each file : 1GB

Cache Size : 200 GB

Test Setup 4.2

Number of files being downloaded : 200

Size of each file : 1GB

Cache Size : 100 GB

Test Setup 4.3

Number of files being downloaded : 200

Size of each file : 2GB

Cache Size : 200 GB

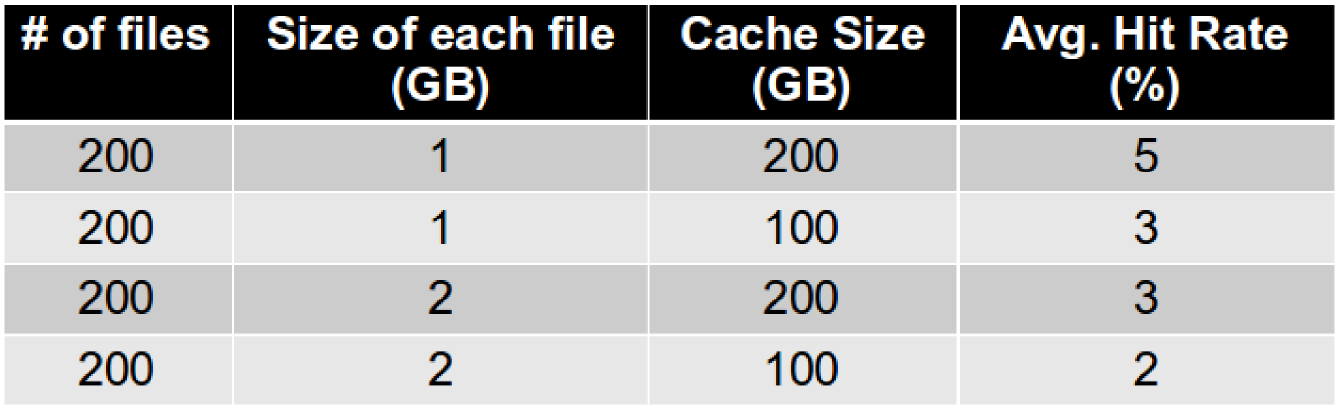
Test Setup 4.4

Number of files being downloaded : 200

Size of each file : 2GB

Cache Size : 100 GB

The results of the sensitivity test for Scenario # 4 is shown in Illustration 49 :



# of files	Size of each file (GB)	Cache Size (GB)	Avg. Hit Rate (%)
200	1	200	5
200	1	100	3
200	2	200	3
200	2	100	2

Illustration 49: Sensitivity Test results for Scenario # 4

Observation for Scenario # 4

A trend is observed for decreasing Avg. Hit Rate with the increase in the file size and subsequently decrease in Cache Size. Test Setup 5.1 focuses on finding the Avg. Hit Rate for more files each of size 1GB.

Test Setup 5.1

Number of files being downloaded : 300

Size of each file : 1GB

Cache Size : 200 GB

Result of Test 5.1

Avg. Hit Rate : 2%

Observation for 100, 200 and 300 files of 1GB each

Hit Rate$_{100files}$ > Hit Rate$_{200files}$ > Hit Rate$_{300files}$, for 200GB Cache in all three cases

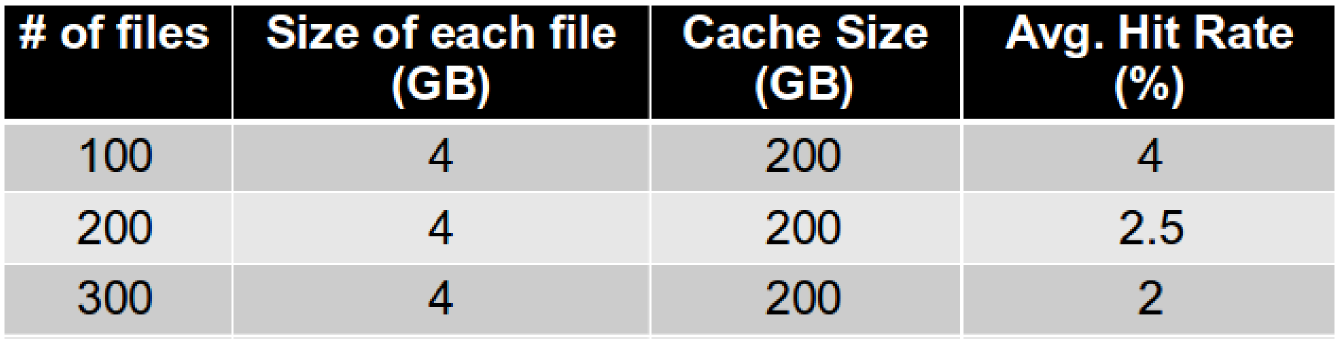
Test Setup 6

Number of files being downloaded : 100, 200 and 300 respectively

Size of each file : 4GB

Cache Size : 200 GB

The results of the sensitivity test for Scenario # 6 is shown in Illustration 50 :



# of files	Size of each file (GB)	Cache Size (GB)	Avg. Hit Rate (%)
100	4	200	4
200	4	200	2.5
300	4	200	2

Illustration 50: Sensitivity Test results for Scenario # 6

Conclusion

The Avg. Hit Rate is dependent on factors like ‘Number of files being downloaded’, ‘Size of each file’ and the ‘Cache size’. The Avg. Hit Rate is found to be decreasing with the increase in the number of files. However, no fixed percent decrease in Hit Rate can be established with increase in the number of files. Xcache is a very suitable file-caching mechanism which can also be configured as a proxy server. The disk space for caching is now 26 TB, but going forward it will be set at 104 TB (4 x 26).

Future Work

- Rolling out Xcache in to ARDC Production environment.
- Studying the behavior of Xcache while connected to multiple clients (worker nodes).
- Extending the Caching mechanism to address the latency factor between The University of Texas at Arlington and Oklahoma University.

References

[1] “CERN Accelerating Science” [Online]

Available : https://home.cern/about/who-we-are/our-governance/member-states

[2] “Large Hadron Collider” in Wikipedia, Wikimedia Foundation [Online]

Available : https://en.wikipedia.org/wiki/Large_Hadron_Collider

[3] “About CERN” [Online]

Available : https://home.cern/about

[4] “Antihydrogen” [Online]

Available : https://home.cern/tags/antihydrogen

[5] “Fifty years CP Violation” [Online]

Available : https://home.cern/news/news/physics/fifty-years-cp-violation

[6] “The Higgs Boson” [Online]

Available : https://home.cern/science/physics/higgs-boson

[7] “CERN” in Wikipedia, Wikimedia Foundation [Online]

Available : https://en.wikipedia.org/wiki/CERN#cite_note-wz-13

[8] “Heavy Ions and Quark Gluon Plasma” [Online]

Available : https://home.cern/science/physics/heavy-ions-and-quark-gluon-plasma

[9] “Antimatter” [Online]

Available : https://home.cern/science/physics/antimatter

[10] “LHC Experiments” [Online]

Available : https://home.cern/science/experiments

[11] “LHC : Pushing computing to its limits” [Online]

Available : https://home.cern/news/news/computing/lhc-pushing-computing-limits

[12] “CASTOR : CERN Advanced Storage Manager” [Online]

Available : http://castor.web.cern.ch/

[13] “CERN IT in 8 minutes” [Online]

Available : https://home.cern/resources/video/computing/cern-it-8-minutes

[14] “LHC Season 2 : Facts and Figures” [Online]

Available : https://home.cern/sites/home.web.cern.ch/files/2018-07/factsandfigures-en_0.pdf

[15] “Proton beams are back in the LHC” [Online]

Available : https://home.cern/news/press-release/cern/proton-beams-are-back-lhc

[16] “How a detector works : CERN Accelerating Science” [Online]

Available : https://home.cern/science/experiments/how-detector-works

[17] “CERN’s IT gears up to face the challenges of LHC Run 2” [Online]

Available : https://cerncourier.com/cerns-it-gears-up-to-face-the-challenges-of-lhc-run-2/

[18] “LHC Open Network Environment” [Online]

Available : http://lhcone.web.cern.ch/

[19] “High-Luminosity LHC : CERN Accelerating Science” [Online]

Available : https://home.cern/science/accelerators/high-luminosity-lhc

[20] “The Proton Synchrotron : CERN Accelerating Science” [Online]

Available : https://home.cern/science/accelerators/proton-synchrotron

[21] “The Super Proton Synchrotron : CERN Accelerating Science” [Online]

Available : https://home.cern/science/accelerators/super-proton-synchrotron

[22] “The Antiproton Decelerator : CERN Accelerating Science” [Online]

Available : https://home.cern/science/accelerators/antiproton-decelerator

[23] “High Luminosity Large Hadron Collider” in Wikipedia, Wikimedia Corporation [Online]

Available : https://en.wikipedia.org/wiki/High_Luminosity_Large_Hadron_Collider

[24] “Wheels in motion : what’s planned for ATLAS in next two years” [Online]

Available : https://home.cern/news/news/experiments/wheels-motion-whats-planned-atlas-next-two-years

[25] “The Standard Model” [Online]

Available : https://home.cern/science/physics/standard-model

[26] “ATLAS : Detector and Technology” [Online]

Available : https://atlas.cern/discover/detector

[27] “LHC Filling Accelerator chain and experiments location” [Online]

Available : https://www.researchgate.net/figure/LHC-filling-accelerator-chain-and-experiments-location-Figure-based-on-4_fig1_254468943

[28] “ATLAS Inner Detector” [Online]

Available : https://atlas.cern/discover/detector/inner-detector

[29] “ATLAS Calorimeter” [Online]

Available : https://atlas.cern/discover/detector/calorimeter

[30] “Performance of ATLAS Liquid Argon Calorimeter” [Online Article]

Available : https://iopscience.iop.org/article/10.1088/1748-0221/9/03/C03049/pdf

[31] “ATLAS Magnet System” [CERN Online]

Available : https://atlas.cern/discover/detector/magnet-system

[32] “ATLAS : Trigger and Data Acquisition System” [Online]

Available : https://atlas.cern/discover/detector/trigger-daq

[33] “Worldwide LHC Computing Grid” [Online]

Available : http://wlcg-public.web.cern.ch/about

[34] “The LHC and Largest Computing Grid Ever” [Online]

Available : https://www.theskepticsguide.org/the-lhc-and-the-largest-computing-grid-ever

[35] “Globus : How it works” [Online]

Available : https://www.globus.org/how-it-works

[36] “Features of GridFTP” in Wikipedia, Wikimedia Foundation [Online]

Available : https://en.wikipedia.org/wiki/GridFTP

[37] “Energy Sciences Network : GridFTP Quick Start Guide” [Online]

Available : https://fasterdata.es.net/data-transfer-tools/gridftp

[38] “CernVM-FS : Overview” [Online]

Available : https://cvmfs.readthedocs.io/en/stable/cpt-overview.html

[39] “Rucio Documentation” [Online]

Available : https://rucio.readthedocs.io/en/latest

[40] “Rucio : Scientific Data Management” [Online]

Available : https://rucio.cern.ch

[41] “PanDA Harvester” in GitHub [Online]

Available : https://github.com/PanDAWMS/panda-harvester/wiki

[42] “The PanDA Production and Distributed Analysis System” [Online]

Available : https://twiki.cern.ch/twiki/bin/view/PanDA/PanDA

[43] “Open Science Grid Documentation : Security” [Online]

Available : https://opensciencegrid.org/docs/security/host-certs

[44] "Open Science Grid Documentation : User Certificate" [Online]

Available : https://opensciencegrid.org/docs/security/user-certs

[45] "OSG : Installing and Maintaining XRootD" [Online]

Available : https://bbockelm.github.io/docs/data/xrootd/install-storage-element

[46] "Xcache in ATLAS Distributed Computing" [Online]

Available : http://cds.cern.ch/record/2648892/files/ATL-SOFT-PROC-2018-031.2.pdf

[47] "ATLAS Computing Infrastructure at UTA" by Mr. Patrick McGuigan [Online]

Available : https://indico.cern.ch/event/773606

[48] "Proxy Storage Configuration : XRootD Documentation" [Online]

Available : http://xrootd.org/doc/dev49/pss_config.htm

[49] "Setting up an Xcache Reverse Proxy" [Online]

Available : https://cvmfs.readthedocs.io/en/stable/cpt-xcache.html

Appendix A

Network Traffic Data Files

- 100files1GBEach100GBCache
- 100files1GBEach200GBCache
- 100files2GBEach100GBCacheSize_SENSITIVITY
- 100files2GBEach200GBCacheSize_SENSITIVITY
- 100files4GBEach200GBCache
- 200files1GBEach100GBCache
- 200files1GBEach200GBCache
- 200files2GBEach100GBCache
- 200files2GBEach200GBCache
- 200files4GBEach200GBCache
- 2GB100filesCacheEnabled
- 2GB20filesCacheEnabled
- 2GB50filesCacheEnabled
- 300files1GBEach200GBCache
- 8GB100filesCacheEnabled
- 8GB20filesCacheEnabled
- 8GB50filesCacheEnabled
- mixedBag500files
- NIC_InputOutput
- NIC_InputOutput2
- Output_CacheFilled

Appendix B

ipynb files used for Hit Rate Calculation and Graph Plots

- Storage_HighLowWatermark
- InputOutputXcache_StatisticalSampling-500Files
- InputOutputXcache_StatisticalSampling_300files
- 50files8GBEach
- 100files2GBEach
- 100files8GBEach
- 20files2GBEach
- 20files8GBEach
- 50files2GBEach
- 100files1GBEach200GBCacheSize_sense1
- 100files1GBEach100GBCacheSize_sense2
- 100files2GBEach200GBCacheSize_sense3
- 100files2GBEach100GBCacheSize_sense4
- 200files2GBEach200GBCacheSize_sense5
- 200files2GBEach100GBCacheSize_sense6
- 200files1GBEach100GBCacheSize_sense7
- 200files1GBEach200GBCacheSize_sense8
- 300files1GBEach200GBCacheSize_sense9
- 100files4GBEach200GBCache_sense10
- 200files4GBEach200GBCache_sense11

Biographical Information

Priyam Banerjee received his Bachelor's Degree in Electrical Engineering from West Bengal University of Technology, Kolkata, India in 2014. He worked as a Software Developer and Tester with Infosys Limited in Mysore from August 2014 to July 2017. He took a keen interest in the intricacies of software development process and decided to pursue Master's Degree in Computer Science and Engineering.

During his time as a Graduate student at The University of Texas at Arlington, Priyam grew interested in Cloud Computing, High Throughput Computing, High Performance Computing, Data Mining and Machine Learning. He started his research in Cluster Computing and High Performance Computing on CERN's High Energy Physics ATLAS Experiment Project under the guidance of Professor David Levine. He worked closely with Dept. of Physics Research Faculty member Mr. Patrick McGuigan for his Thesis work on Cloud/Cluster Computing. Priyam's interest lies in areas of Cloud Computing, High Throughput Computing, Data Mining, Machine Learning and Software Engineering.

In his spare time, Priyam loves reading books and watching basketball matches. He loves traveling to new places and photographing them.