University POLITEHNICA of Bucharest
Faculty of Automatic Control and Computer Science

Summary of the PhD Thesis

Optimizări și soluții de stocare distribuită

Distributed Storage Solutions and Optimizations

Autor: Ing. Sorin-Andrei Piștirică

COMISIE DOCTORAT

Preşedinte	Prof. Dr. Ing. Adina Magda Florea	Universitatea POLITEHNICA din București
Coordonator științific	Prof. Dr. Ing. Florica Moldoveanu	Universitatea POLITEHNICA din București
Referent	Prof. Dr. Ing. Nicolae Ţăpuș	Universitatea POLITEHNICA din București
Referent	Prof. PhD. Eng. Alexandru Soceanu	Munich University of Applied Sciences
Referent	Prof. Dr. Ing. Ștefan Pentiuc	Universitatea „Ştefan cel Mare” din Suceava

București, 2015

TABLE OF CONTENTS

1 INTRODUCTION

During the last ten years internet traffic volume grew more than 300 times and the pressure on technologies used for network infrastructure increased significantly. Besides internet, there are systems such as clouds or data centers, where the pressure on I/O intensive systems, like storage systems, influenced the evolution of network infrastructure concerning the capital costs without performance penalty and pushed the design of the software towards distribution.

Considering the above, I studied various features of storage systems and propose an infrastructure for eLearning technologies. Since storage systems are very I/O intensive, I propose a solution for increasing their performances based on dedicated hardware for packet processing. Furthermore, considering the trend of converged networks emerged from the need to ease the management and reduce the capital costs, I propose several improvements to Quantized Congestion Notification protocol as well as an algorithm for dynamic balancing of traffic flows: Quantized Congestion Notification – Weighted Flow Queue Ranking.

In context of distributed storage systems, the scaling-out methods based on file system layers are very important. Each layer requires different type of network, thus generates different types of distributed storage systems. It is also important to see how they are characterized based on classifications and what are the main deficiencies of the existing implementations. Therefore, I propose a simple, yet comprehensive classification based on four main characteristics: locality, sharing, distribution level and semantics of concurrent access. In terms of low level organization of data there are several ways, but all follow the same constituent elements, so I propose a suitable generalized data layout regardless the distribution status of the system. Within proposed layout, data is distributed in different units (i.e. files, objects, blocks or data segments) analyzed further, with an increasing trend towards objects, due to autonomous characteristic of object devices. Afterwards, I studied five different architectures of distributed storage systems (Andrew File System, Google File system, General Parallel File System, Lustre File System and Ceph) with more focus on Ceph's particularities, used later as case study and performance measurements for the optimizations proposals. Afterwards, I studied the way that everything is linked together and using graph theory I present several key characteristics that influence properties such as throughput, capital cost, fault tolerance or high availability, followed by studies of protocol stacks used to identify different alternatives suited for converged networks.

I propose several enhancements to solve issues related to system overheads due to processing of flows at multi-gigabit rates. The enhancements are based on integrated hardware systems of dedicated packet processors with general purpose cores. Besides system overhead issues, another challenge is the enhancement of the network hardware performance to make optimal use of the available network bandwidth. These problems are overcome by combining multi-core processors with parallel processing and multi-function network hardware capabilities. Thus, I propose a generic hardware packet processor emphasizing its components along with processing stages for each frame. I also designed two alternatives for coexistence of different applications with different traffic demands: Per Core Cluster Node and SMP Cluster Node. Using Ceph as case study, I propose two different optimizations: accelerations methods of each node using the proposed models listed above (i.e. A-MDS, A-MON and A-OSD) and a method of decreasing latency of sensitive flows based on queue weighting (A-RADOS). The proposed solutions are supported by several micro-benchmark results based on P2041 QorIQ^{TM} as integrated packet processor.

Furthermore, in the context of network convergence in data centers, I/O protocols (such as SCSI) do not have contention or retransmission support and they require a lossless transmission environment, such as Fibre Channel – a high speed, low latency and lossless network by design. Ethernet by design is a best effort communication environment and with IP protocol it provides an end-to-end network for reliable transport protocols, such as TCP. In absence of reliable protocols, Ethernet has been enriched with a set of protocols which enabled a lossless medium: Priority Flow Control (PFC), Enhanced Transmission Selection (ETS), Data Center Bridging Capabilities exchange (DCBx) and Quantized Congestion Notification (QCN). Despite the main purpose of these enhancements, there are other uses cases, for example "TCP Incast" that can be improved using QCN.

Basically, QCN provides congestion information to the source to avoid packet loss, but it doesn't solve the congestion fairness within the entire network. For this, I propose QCN Weighted Flow Queue Ranking (QCN-WFQR), an algorithm based on Quantized Congestion Notification for dynamic workload balancing. The algorithm can be used in traditional computer networks or in new Software Defined Networks. It computes a series of congestion indicatives relative to each flow per each congested point and a series of system wide congestion indicatives. These indicatives can be used cooperatively (by all elements in the cluster) or automatically (by a system profiler or SDN controller) to increase the overall system performance. For exemplification, I propose two different applications for the use of these congestion indicatives. Former application proposes a method of choosing replicas in distributed and parallel file systems to achieve a less congested network. The latter application propose a method of distributing traffic workload in the network by migrating already established flows to alternate less congested paths, thus reducing the need to slow down the traffic at the source.

1.1 Scientific publications in connection with this thesis

Patents

[App. #20150023172]. **Sorin A. Pistirica,** Dan A. Calavrezo, Casimer M. DeCusatis, Keshav G. Kamble, "**Congestion Profiling of Computer Network Devices**", Patent Pending, USPTO: http://patents.justia.com/patent/20150023172, Jul 16 – 2013

[Patent #8891376]. **Sorin A. Pistirica**, Dan A. Calavrezo, Keshav G. Kamble, Mihail-Liviu Manolachi, "**Quantized Congestion Notification—defense mode choice extension for the alternate priority of congestion points**", USPTO: http://patents.justia.com/patent/8891376, Oct 07 – 2013

Articles

[PIST, 2013] Pistirica Sorin Andrei, Caraman Mihai Claudiu, Moldoveanu Florica, Moldoveanu Alin, Asavei Victor, "**Hardware acceleration in CEPH Distributed File System**", ISPDC: IEEE 12th International Symposium on Parallel and Distributed Computing, Bucharest, June 2013, pp. 209-215, **IEEE Indexed**

[PIST, 2014/1] Pistirica Sorin Andrei, Victor Asavei, Horia Geanta , Florica Moldoveanu , Alin Moldoveanu , Catalin Negru , Mariana Mocanu, "**Evolution Towards Distributed Storage in a Nutshell**", HPCC: The 16th IEEE International Conference on High Performance Computing and Communications, August 2014, Paris, pp. 1267-1274, **IEEE Indexed**

[PIST, 2014/2] Pistirica Sorin Andrei, Asavei Victor, Egner Alexandru, Poncea Ovidiu Mihai, "**Impact of Distributed File Systems and Computer Network Technologies in eLearning environments**", eLSE: Proceedings of the 10th International Scientific Conference "eLearning and Software for Education", Bucharest, April 2014, Volume 1, pp. 85-92, **ISI Indexed**

[PIST, 2015] Pistirica Sorin Andrei, Poncea Ovidiu, Caraman Mihai, "**QCN based dynamically load balancing: QCN Weighted Flow Queue Ranking**", CSCS: The 20th International Conference on Control Systems and Computer Science, Bucharest, May 2015, Volume 1, pp. 197-205, **ISI Indexed**

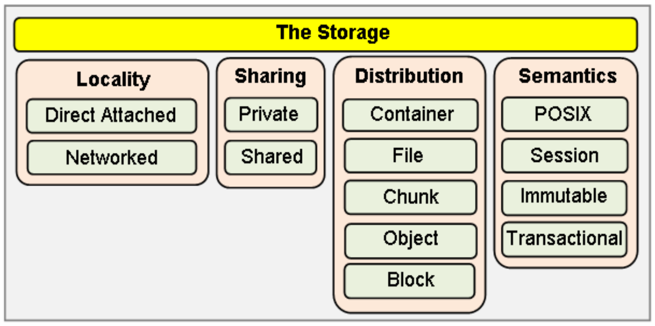
2 STORAGE SYSTEMS: STATE-OF-THE-ART

In general, any **storage system** comprises a set of devices and a collection of software modules for managing data units, such as blocks, objects or files. Distributed storages are special cases, where storage elements are distributed over a set of nodes within a computer network.

2.1 Storages classification

Storage systems classifications are very important, mostly because it drives the intrinsic characteristics of them and points out the deficiencies of existing implementations. Therefore, I propose a classification of storage systems based on four main characteristics **[PIST, 2014/1]**.

- **Locality**;
- **Sharing capability**;
- **Distribution level:** data units used for distribution;
- **Semantics**: sharing semantics of concurrent access.



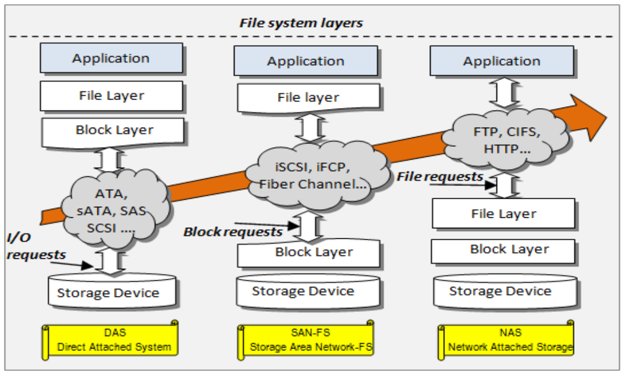
2.2 Requirements for distributed storages

In the early 70'es at the beginning of computer networks, the main purpose of distributed storage systems was just to share small pieces of data between nodes. Later on, with the advent of cloud computing, HPC domains or internet applications such as Google or Facebook, the need for storage space increased exponentially, therefore I propose the following list of basic requirements based on articles published in the field: **data sharing, storage scalability, storage elasticity, transparency (access, location, failure, replication, migration), high availability, fault tolerance, system recovery, data balanced distribution, workload balance, data migration, concurrent and consistent data access, snapshot, archive, network infrastructure performance and data security**.

2.3 Storage systems scaling methods

To meet nowadays demands of storage space, which is very high reaching at petascale or even exascale size, the number of storage devices is very large (i.e. scale-out) within the cluster **[PIST, 2014/1]**. This can be achieved by distribute data at different file system levels:

- hardware layer (storage devices);
- block or object layer (sequence of bits stored on storage devices);
- file layer (typed data managed by the underling operating system);
- application layer.



Depending on the layer at which the system is decoupled and computer networks are introduced, different types of systems are built:

- SAN based distributed storage systems – at block I/O level (e.g. SCSI over Fibre Channel);
- NAS based distributed storage system – at file level (e.g CIFS or NFS are used to carry requests);

- Hybrid systems can be built by mixing these SAN and NAS types;
- Furthermore, *distributed and parallel file systems* (e.g. GFS, Lustre or Ceph) are built by abstracting storage devices.

Mass-storage systems

Mass-storage systems are built by using Redundant Arrays of Independent Disks (**RAID).** These systems are based on two main ideas: to improve performance by accessing multiple disks in parallel and to improve high availability and fault tolerance by duplicating data over multiple devices. EMC[1] introduced **Storage Pools** and **RAID based Storage Pools –** collections of logical/physical disks or RAIDs managed together, used to simplify space management. Fine grained management space is achieved with virtualization of storage pools into **Virtual LUNs** or **Virtual Volumes**, where the mapping between Virtual LUNs and RAID LUNs are made by a **Virtualization Controller**.

From architectural point of view there are three main approaches of mass-storage virtualization: host-based (e.g. *Linux Logical Volume Management*: LVM) [4], device-based (e.g. disk arrays: RAIDS) and network-based (e.g. SAN systems) [5] [6].

2.4 Sharing semantics

One of the most important characteristics of the distributed storage systems is **sharing semantics** (i.e. semantics of concurrent access of two or more tenants). Andrew S. Tanenbaum in his book "Distributed Operating Systems" [7] proposed a simple and comprehensive set of four different types of file sharing semantics: POSIX, session, immutable files and transactional. Among them, the POSIX semantics is the hardest model to implement, because it requires complicated synchronization mechanisms.

2.5 Low level data organization

2.5.1 Data organization at device level

In the context of data organization at device level, I consider three main organizational types: structured, log-based and tree-based. Briefly, the main difference between structured and log-based organization is that the former type has distinct areas for inodes and data blocks, while the later stores them in continuous form. Tree-based organization (e.g. B-tree File System: BTRFS) uses a copy-on-write (COW) model, where data is organized in a binary tree structure. Oracle released in 2007 a file system that uses this type or organization (Binary Tree File System [1]), included in Linux kernel in 2009. Comparing with UFS, BTRFS scales better and also showed performance improvements.

2.5.2 Networked storage data organization

Generally speaking, in any distributed storage system, data is divided in different types of units (e.g. files, blocks or objects) distributed across the nodes of a network using different techniques, such as mapping tables or hash functions. As in the case of local file systems, in distributed file systems as well, the organization of data segments is specified in a type of metadata structure, distributed as any regular file or distributed in a private cluster or stored in a centralized way.

Usually, a distributed storage system does not have the concept of partition, but sometimes there are implemented mechanisms that can be considered as similar concepts, for example volumes for Andrew File System or the multiple systems managed by MGS in Lustre.

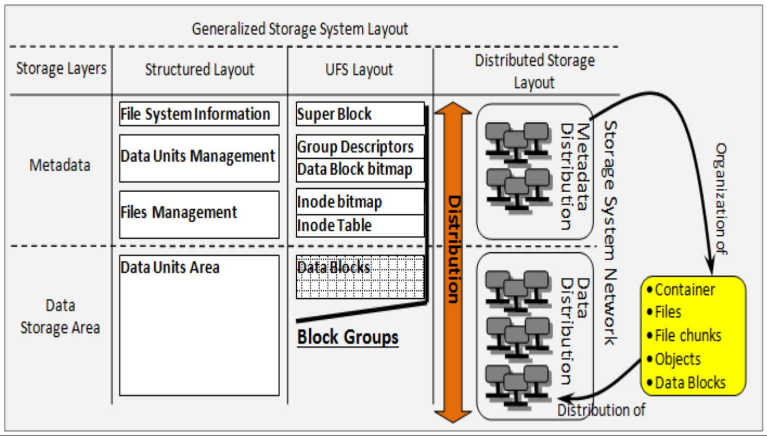
[1] EMC: corporation that provides IT storage hardware solutions.

SAN and **NAS** are two of the most known networked file systems. In terms of architecture, SAN based file systems usually use storage nodes with RAID disks, while NAS based file systems are often used along with SAN environments, were file servers are constructed over NAS and data is stored in a SAN medium (i.e. hybrid file system).

2.5.3 Generalization of data organization

Regardless of data semantics or distribution status, any file system has two main layers, as shown below:

- **Metadata area**: specifications of storage organization;
- **Data area**: a flat address space of data units.



2.5.4 Trends towards object based storage systems

Stored data may be distributed using different types of units (as shown above): containers, files, segments, blocks or objects. Currently, the tendency is to use file segments to distribute data, rather the entire file, because for example, this way a higher degree of availability and a better fault tolerance is achieved.

Although there are several types of data representation used for distribution, objects seems to gain in popularity, since many researchers claim that OSDs take advantages from DAS, SAN and NAS architectures grouping them into a single device [12]. Basically, it addresses issues related to **security**, **scalability** and **management**. First, security is improved by adding credentials to each operation, while in contrast a block device has per LUN based security. Second, scalability and management is improved by moving space management from metadata servers to storage controllers. And third, one management advantage that emerges from OSDs characteristics is that object based storage systems simplify data layout by replacing large blocks lists with small objects lists and distribute the low-level block allocation problem. Although, it has many advantages object based distribution does not solve the fundamental issue of distributing data among a very large number of devices.

2.6 Distributed storages architectures

Among many distributed storage systems with different architectures, I have chosen to study five representative systems to cover an important set of topics: low level organization, overall system design and sharing semantics: **AFS** (Andrew File System) [13] [14], **GPFS** (General Parallel File System) [17], **GFS** (Google File System) [15] (and the open source version **Hadoop** (HDFS) [16]), **Lustre** [18] and **Ceph** [19]**, [PIST, 2014/1]**. The main focus was on Ceph, being used also as case study for my optimizations proposals.

2.6.1 Ceph File System

Ceph [19] is a new distributed storage system that assembles many advantages from previously existing implementations. It distributes everything: namespaces, monitoring, security and data, therefore it is very scalable, but very complex being considered "*the new dream distributed file system*". One of the main characteristics of Ceph is that it completely isolates metadata management from data storage (RADOS – Reliable, Autonomic Distributed Object Store) [20]. Ceph stripes files into objects similar with RAID model, to improve parallel I/O and furthermore RADOS distribute these objects over a cluster of OSDs using *controlled hash function* (CRUSH – Controlled Replication Under Scalable Hashing) [21].

RADOS is a system composed of a large OSD cluster and a small cluster of monitors used as supervisors. Briefly, files are striped into objects, further mapped to *placement groups* (used to control the degree of replication declustering). The OSD cluster layout is described by a *cluster map*. By applying sets of *placement policies* on this map, different failure domains can be configured. Furthermore, a deterministic list of OSD is obtained using CRUSH algorithm based on placement group id and a set of placement rules. The main messages exchanged (used for case study presented in section 4.4) are listed in the following table:

#	RADOS messages	Description
1	**Leader monitor election**	• Provides consistent cluster-map to all system parties, • Monitors leader election based on simplified PAXOS and quorum for serialization of map updates [22] pp. 127.
2	**Leases**	• Grants short-term leases to active monitors to give them the rights to distribute copies of cluster map to all system parties, [22] pp. 127.
3	**Cluster-map distribution**	• Responses of active monitors (which have granted a lease) to map queries from MSDs, clients or new added OSDs. Since an OSD cluster may have a very large number of devices [22] pp. 129, the monitors don't broadcast map updates, but actually the map updates are exchanged between OSDs from the same placement group.
4	**Heartbeat messages**	• To prevent accessing inconsistent data, timely heartbeat messages are exchanged between OSDs [22] pp. 129.
5	**I/O traffic**	• Objects read/write and data replication (i.e. primary-copy, chain and splay) [22] pp. 131.
6	**Recovery**	• Failure recovery is based on cluster-map epoch and the set of active OSDs in each placement group and it uses a peering algorithm to assure a consistent view of the placement group of all OSDs within [22] pp. 137.

The **namespace management** is distributed across a metadata cluster, which uses a technique named *adaptive workload distribution* (i.e. dynamic sub-tree partition to achieve a scalable performance). Ceph migrates or replicates pieces of files tree at directory fragment level based on popularity degree, which becomes costly when the number of nodes is very large.

2.6.1.1 Decentralized data distribution

The main purpose of decentralized data distribution algorithms is to replace the lookup servers. In this subsection I briefly present two such algorithms: Replication Under Scalable Hashing (**RUSH**) [23] and Controlled Replication Under Scalable Hashing (**CRUSH)** [21].

RUSH algorithm has several flavors for data placement, such as: using prime numbers, support for removal and a tree-based approach [24]. It has two main principles: identifying which object must be moved to maintain a balanced cluster and decide the object destination based on hash functions. Also it implies that devices are replaced in groups (e.g. per shelf basis), rather than one-by-one.

CRUSH [21] introduce the idea of **controlled distribution** per different failure domains by means of placement policies. Basically, it produces a deterministic ordered list of storage devices for an object based on *pseudo-random hash function*, *cluster map* and *placement rules*.

2.6.2 Comparison

- Location of data in the distribution space can be handled in two ways: by using information maps (similar with inode map from UFS) or by decentralized distribution algorithms such as CRUSH or RUSH. The drawback of mapping against decentralized distribution algorithms is the increased pressure on some dedicated lookup servers.
- Distributing metadata management has pros and cons: in essence, a distributed management comes with a better I/O performance, but a more complex architecture. Regarding the above systems, Ceph and GPFS handle metadata in a distributed way, while GFS/HDFS use only shadow servers for high availability, but the actual activities are handled only by one active server.
- The distribution level refers to the data unit used for distribution, such as: data segments, files or objects. There is a global tendency towards OSD usage, due to several strength, including: better data security (per object), flexible size (therefore it can be adjusted to increase I/O performance), standard API and a local space management. In terms of studied systems, Lustre and Ceph uses OSDs and the future leads to OSDs for AFS (OpenAFS) and GPFS as well.
- GPFS is sensitive to devices failures, because usually it uses a declustered RAID approach. In terms of RAIDS, there is also a trend to use Object RAID [27] where PanFS is leading.
- In distributed storages is a trend to add support for commodity hardware, where failures are a common behavior rather than an exception. To overcome the occurring issues, the systems must implement different mechanisms to achieve a specific degree of fault tolerance and high availability.
- The access interface is an important characteristic of storage systems, therefore the distributed storages may be divided in two main categories: systems that have support for conventional API (i.e. Berkeley socket) or using a custom API. A conventional API has the advantage of transparency, but a custom API has the fine tune functionality to increase performance. Ceph mixes the two methods and uses an enriched conventional API with additional features for manipulating different system characteristics.
- Systems that uses pieces of a files to distribute data is less sensitive to failures and have a better degree of high availability compared to AFS, for example, which distributes entire files (volume cloning).
- The file sharing semantics is handled as follows:
  - AFS: Session Semantics (weak);
  - GPFS, Lustre and Ceph: POSIX Semantics;

- GFS/HDFS: mimic POSIX Semantics (more relaxed semantics).

2.7 Network technologies for distributed storages

The network infrastructure has a great influence in the performance of distributed storage systems **[PIST, 2014/1]** and is divided in two main topics: network topologies and protocol stacks.

2.7.1 The topology

The topologies used in distributed systems are defined by graphs used to create paths between storage devices and edges [28]. There are intrinsic similarities between graphs used to connect general purpose cores in multi-core systems and distributed systems in general, since they aim relatively same characteristics: a perfect interconnection topology should have **smallest latency** possible (round trip time close to zero), **maximized throughput** (line rate using any packet size), **minimum capital cost** and a **perfect fault tolerance** and **high availability** [29]. In data centers (and distributed storage systems) graph diameter influences latency and contention (characteristic more relevant for SAN based systems). Also, a smaller node degree decreases the capital cost of the network's equipment, but it increases the latency since the topology will have more hops, a smaller diameter lowers the latency and round trip time and increases the throughput, a smaller bisection width facilitates the left-to-right traffic (important for data migration and data distribution among storage nodes) and a high number of edges influences the deployment.

Analysis of topologies:

- The complete-graph has the perfect **diameter** (value of 1), hypercube, cube-connected-cycles, binary-fat-trees and hypertrees has a logarithmic diameter (it is worth to mention that hypertrees has the advantage of using constant speed for links at all levels, while fat-trees must provide *fatter* links toward the tree root).
- The **node degree** is constant for torus, binary-fat-trees and hypertrees. Nevertheless, hypertrees have the ability to scale also horizontally changing the k-ary dimension. The node degree of the hypercube grows faster by logarithmic function, regarding the number of nodes in the topology.
- In terms of **fault-tolerance**, except for fat-trees, all other topologies offer alternative routing paths. Hypertrees preserve diameter in case of faulty nodes, a very important advantage. Cube style topologies have a better fault-tolerance, because they have many alternative routes between any two nodes.

When designing a topology, the chosen graph depends on traffic requirements: east-west/south-north traffic profile, low latency or high throughput and of course the balance between costs and performance.

2.7.2 Protocol stack

In terms of protocol stack used, distributed storages are divided into systems that require lossless mediums and systems that can use best effort mediums. Regarding OSI model, the reliability problem can be solved at network layer 2 (data link) or layer 4 (transport).

Ethernet is the most common data link protocol used, but is a best effort by design, while for example Fibre channel is a lossless medium. Basically, fibre channel protocol (FCP) runs over fibre channel infrastructure, but there is a solution where FCP frames are embedded into Ethernet frames: Fibre Channel over Ethernet. At layer 4, reliability can be handled by TCP, but with a relatively performance penalty degree: iFCP, iSCSI, FCIP.

Ethernet has been enriched with a group of protocols named DCB (Data Center Bridging) by IEEE group, [42] for traffic control to avoid packet dropping and therefore Ethernet has become a lossless environment and an alternative to Fibre Channel.

Distributed storages built at file level are less demanding and usually run over TCP/IP/Ethernet stack (classic LAN) using protocols such as: NFS, CIFS or SMB.

3 CASE STUDY: INFRASTRUCTURE FOR ELEARNING ENVIRONMENTS

3.1 Motivation

Usually, high education is expensive, therefore eLearning platforms provide a way of affordable education with an effective cost of investments. In the last decade, cloud computing research and adoption increased greatly due to its many advantages including economic benefits, ease of management, power saving and so on. In essence, it provides the means to organize and deliver a wide variety of software services including **eLearning environments**. Along with cloud computing, file systems solutions were improved to meet requirements imposed by the distribution characteristics of clouds. The distribution level of file systems, data management, data seek methods and many other features of distributed file systems, influence performance of eLearning environments. Also, clouds are constructed over computer networks topologies. Further, I propose a high level architecture of cloud platforms adequate for eLearning software by outlining several advantages that overcome issues related to distributed eLearning using Ceph as a data storage environment and several domestic network topologies **[PIST, 2014/2]**.

Why Clouds and eLearning?

eLearnig environments have several issues and some of them may be better handled by clouds systems. One of the main issues is the infrastructure. Usually, eLearning infrastructure requires huge investments, thus clouds being by definition an infrastructure provider may be used as lay foundation for eLearning applications. Clouds scale dynamically (by demand), and offer a collaborative environment as well [56] – an important feature for eLearning services. Basically, eLearning environments have huge databases of learning objects that can be stored in cloud's storage systems. Therefore, the way that data is handled in cloud influences different aspects of eLearning mechanism including searching and content delivery of learning objects. There is a wide variety of storage systems implementations for clouds, encompassing many advantages. Other than the storage system, the way that everything is linked together influences eLearning content delivering performance. There are many network topologies suitable for clouds, including fat-tree, hyper-tree, cube and hyper-cube. I took into consideration some characteristics that influence I/O performance: scalability, latency and costs.

3.2 Clouds storage and network enhanced for eLearning environments

Clouds aims to meet several characteristics that impose a set of main requirements to storage systems, including: sharing, scalability, transparency, high availability, fault tolerance, concurrent and consistent data access and security (section 2.2). The design of storage systems may influence the delivery, maintenance and management of learning objects, while network architecture influences the system I/O performance.

3.2.1 Data distribution for eLearning environments

eLearning environments usually have tremendous database of learning objects, which are organized per category and therefore a hierarchical organization won't help – an alternative could be semantic aware storages. There are researchers that support the idea that the hierarchical file systems will be replaced by semantic file systems [57], because other than categorization issues they also improve searching capabilities [58].

I propose a parallel and distributed file system architecture for a cloud storage system based on two layers: first layer stores learning objects in flat address space (for example RADOS can be used) and the second layer a semantic aware metadata for categorization and searching capabilities.

Another characteristic of eLearning systems is that the learners (users) are geographically spread, using different network paths to access the cloud's storage. To balance traffic load, the system must take into account the learners profile distribution and split the data cluster into distribution domains, where basically each domain is accessed from a specific gateway. The object distribution has to be replicated in each of these domains, thus the data access performance may be improved.

There is an alternative solution based on delegation points, which is a cacheable point of learning objects that in the end are distributed to a group of users. There are few drawbacks for this method including: security delegation, local storage for cached learning objects, lease method and so on. Users may access the learning system from their home using just a web page, so even if the internet provider would have such method in place there won't be any visible gain, the user could still see a big latency. The consequence of this proposal would be that the data has to be replicated on many storage devices – can be considered a good idea since the price per GB storage is usually low.

One more improvement could be to use the network links at optimal capacity and balance the load on the gateways if the network equipment supports Quantized Congestion Notification. There is a genuine method that balances the traffic load between network gateways, which uses a red-black tree to sort the available gateways based on congestion distance [59] or QCN-WFQR (section 6).

3.2.2 Clouds network

A cloud can span across a single or multiple data centers (public or private). The current data center networking follows three layers architecture: **core**, **aggregation** and **access**. The main drawback is that this architecture limits the allowed topologies to those that support a multi layered organization

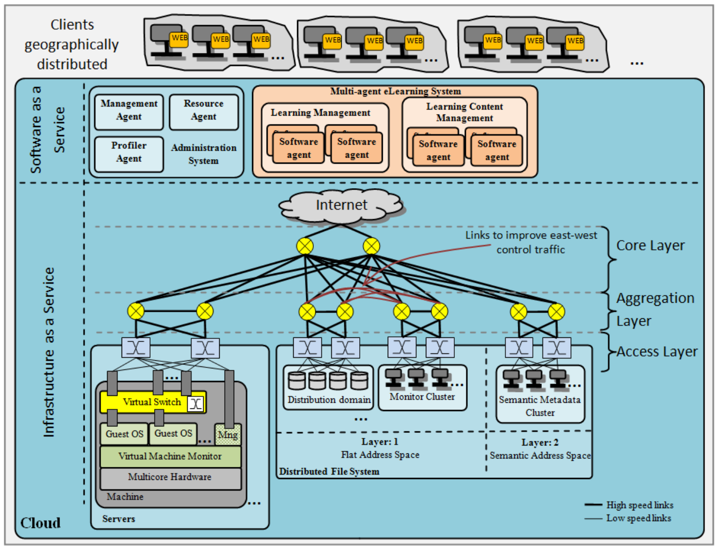
I propose a solution mostly based on fat-tree topology with redundancy. Basically, it provides a 1:1 subscription but is limited in height due to the necessity of providing an uplink capacity equal to the sum of all the downlinks. Due to this disadvantage, it is not used in a *pure* form; usually along the way oversubscription increases. Also, this topology is better suited when most of the traffic is north-south but it is not recommended for use in configurations where east-west traffic is predominant due to the fact that distant nodes cannot communicate directly. In an eLearning environment most of the data transfer is south-north, therefore this solution is a good choice.

3.3 Clouds and eLearning environment: proposed architecture

I propose a cloud architecture emphasizing different aspects related to storage system and network topology that may influence eLearning environments, from architectural and I/O performance point of view.

The architecture is composed of three main infrastructure components: Storage System, Servers and Network and two main software components: Administration System and Multi-agent eLearning System:

Administration System manages server resources (creates, migrate and destroys virtual machines and virtual switches) and eLearning agents creates, migrate and destroys agents, based on profile statistics.



3.3.1 The storage system

In terms of storage, I propose an alternative to OGSA-GFS [60]: a two layered storage system based on Ceph components: (a flat address space (RADOS) and a semantic aware metadata. There are several methods to implement such file organization: property-based, content-based or context-based 0 and several architecture directions: integrated, augmented and independent (native). I propose a native implementation with content-based semantics, because a hierarchical file organization brings no advantage and the content-based architecture categorizes files and improves searching.

Using Ceph/CRUSH, I propose a data distribution based on domains and therefore I define a placement policy and cluster map to replicate each object three times in each domain for high availability and fault tolerance.

3.3.2 The network profile

Read only and high throughput traffic is mostly south-north from system to learners, therefore it will be small amount of replicated or rebalanced data (low east-west throughput demands). Nevertheless, the monitors from RADOS which manages OSDs cluster by keeping storage data synchronized and collect the state of each OSD should have low latency.

A hierarchical topology facilitates south-north high throughput traffic along with a new link from monitors group to each OSDs group to decrease the latency between monitors and OSDs. Also, is worth to mention that if the system grows, aggregation layers may be added, so in a hierarchical topology the east-west latency is growing.

4 OPTIMIZATIONS OF DISTRIBUTED STORAGE SYSTEMS BASED ON HARDWARE ENGINES

4.1 Motivation

Due to the advent of high speed Ethernet (40Gbps, 100Gbps or even 800Gbps), the hardware systems along with software architectures must adapt to newly increased traffic. The capacity of servers to handle such high traffic volume was exceeded. Thus, increasing port speed and CPU computation power along with leveraging different cache levels were the first technologies used to improve throughput and latency at server side. But, in the recent years the servers failed to meet even higher traffic demands due to many factors, such as networks convergence or high volume of real time traffic (e.g. voice or video) [31]. One improvement would be to increase the degree of parallelism (i.e. increase the number of nodes in clusters and split tasks at finer granularity), but this will lead to an increased complexity, power consumption and maintenance costs. Other improvement would be to increase performance of each node by offloading the CPU cores and moving specific tasks to dedicated hardware engines. Hardware engines have double impact: first, being dedicated modules perform their role faster than software which runs on general purpose cores and second, reduce significantly CPU utilization, thus increase the system performance and free CPU processing time.

Intel investigations revealed several I/O bottlenecks at receiver side divided into three overhead categories: system overhead, TCP/IP processing and memory access [31]. To meet these overheads, Intel developed *Accelerated High-Speed Networking* technology (I/OAT) – a set of features to reduce the receiver side packet processing overhead (e.g. split headers, DMA copy engine and multi-queue usage to receive frames). Intel's solution showed improvements for about 38% in CPU utilization, the number of transactions processed increased by 14% and throughput by 12%, [32].

I followed a similar idea to increase the performance of each node from a cluster by offloading the general purpose cores of packet processing at receiver side **[PIST, 2013]**. Offloading comes with flows classifications and multi-queues distributions and by using Accelerated Receive Flow Steering [33] is leveraged the parallelism of multicore systems. In other words, the hardware engines are instructed to classify and distribute flows considering the SMP characteristic of multicore systems and increase performance of parallel I/O applications running on the same machine. Also, by adding queues weights adjustments I proposed a method of reducing the latency of sensitive flows, thus making the system much responsive.

4.2 Integrated packet processing engines

In general, any System on Chip (SoC) integrates a set of dedicated hardware modules along with general purpose cores into a single chip. In this thesis, I refer to a packet processing **SoC** as a chip that integrates multiple hardware engines dedicated to network specific tasks, usually handled by general purpose cores. And I refer to term **offload** (or *hardware offload*) to the migration of particular tasks from general purpose cores to dedicated engines freeing processing time.

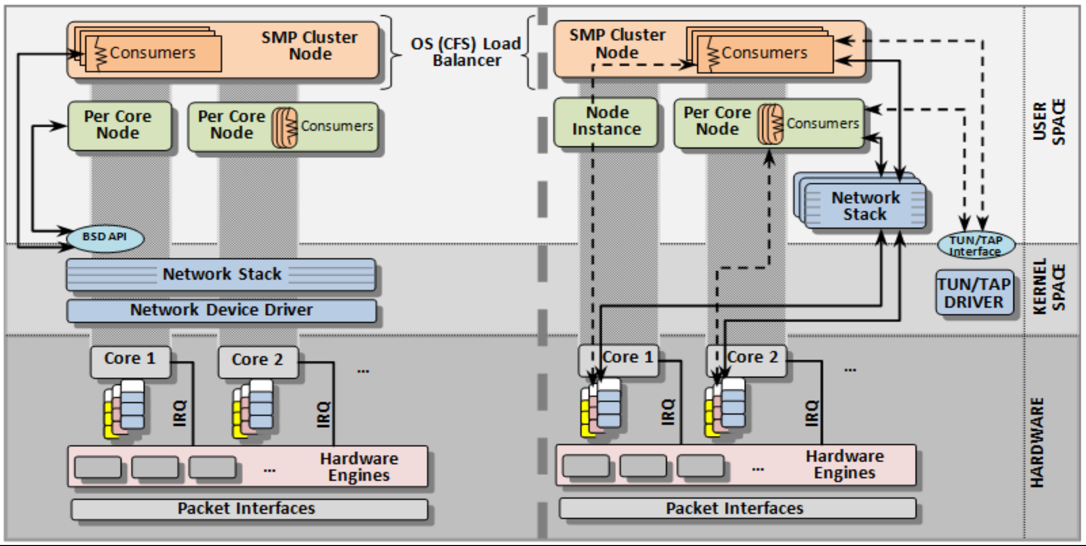
A packet processing SoC may have many programmable flows, therefore each transmitted or received frame is distributed to a specific flow, where each flow can be seen as a path through a graph with hardware engines as nodes and queues as edges.

4.3 Hardware accelerated cluster nodes

OS is the usual solution for coexistence of different applications on the same hardware system. If the applications are operating systems, then virtualization (i.e. hypervisors such as KVM or

XEN) is the solution or if groups of applications (such as independent network stacks) needs a specific isolation degree, then Linux containers (LXC) is the solution.

Given the above motivations, I propose two less restrictive models as alternatives for coexistence of applications with high and low traffic demands, using hardware acceleration and parallel processing of flows: **Per Core Cluster Node** and **SMP Cluster Node** and two implementation directions: in **kernel** and **user space**.



(a). Kernel space implementation (b). User space implementation

Per Core Cluster Node model is characterized by the fact that each application instance is bound to a core, thus the scalability is limited to the number of CPU cores. The modularity is achieved by division of tasks at a smaller granularity by using dedicated threads (i.e. consumers) without being scheduled on a different core. Each consumer is bound to a specific hardware queue or group of queues that provides only specific frames to the consumer thread. The hardware engines must be instructed to classify and distribute flows to match the consumers.

SMP Cluster Node model is similar with former model with the difference that the workload is distributed over a set of cores using capabilities of hardware engines for flows classifications. One solution would be to let hardware engines to distribute workload by pushing frames on cores in "round-robin" fashion (using hash based functions), or to leave the OS scheduler to flatten the workload by scheduling consumers among the core set and instruct hardware engines to consider where the specific consumers are scheduled and push frames accordingly.

By configuring the hardware engines in concordance with software architecture and scheduler affinity, multiple instances of these models belonging to different applications in a converged system may coexist with an increased performance. It is obvious that SMP Cluster Node is befitting for high traffic volume applications with multiple consumers and also is a generic form that can be configured to behave as a Per Core Cluster Node as well if the core affinity is set to only one core.

To leverage the parallelism of multicore systems, I propose the usage of **Accelerated Receive Flow Steering (ARFS)** [33]: the frames distributions have to consider where the specific consumers were scheduled to run. This requirement imposes two things: first, each possible core where a consumer could be scheduled must have a specific queue or group of queues where the hardware engine is instructed to push the frames and second, the hardware engines must be able to steer and push the frames accordingly. In another words, the hardware engines collaborate with OS scheduler for workload balance while achieving a better performance degree.

Also, I propose two different implementations of these models: one uses the drivers in **kernel-space** (more efficient) and the other one uses drivers in **user-space** (small performance penalty, while assuring rapid development and flexible license policy).

An issue for the user-space model is the access method to a TCP/IP stack. By default Linux has the network stack implemented in kernel, thus in case of user-space implementation the applications have to deal with raw frames and lack of connection-oriented protocols. There are few solutions for adding TCP/IP support, such as: network tunneling/tap interfaces or off-the-shelf user-space TCP/IP stack [34]. Another issue for the user-space model is the method of signaling the arrival of frames with few solutions: polling or process signals.

To decrease the latency of sensitive traffic, I propose a solution based on queue weighting: the queues associated with most sensitive flows have higher weights and are scheduled first, thus the sensitive flows are handled first. How flows sensitiveness is established depends on applications purposes, thus for exemplification I propose a solution for Ceph by classifying Monitors and OSDs traffic in three different classes, each associated with a different group of queues with specific weights.

4.4 Case study: Hardware accelerated Ceph with QorIQ™

As case study, I have used QorIQ^{TM} P2041 packet processor to decrease latency of sensitive flows and increase performance of Ceph's nodes: OSDs, monitors and metadata servers.

4.4.1 QorIQ™ packet processors

QorIQ combines multi-general purpose cores with various hardware engines providing a very flexible infrastructure (**Data Path Acceleration Architecture** – **DPAA**) for processing high volume of network traffic at high speeds. Among multiple hardware engines within DPAA, I have focused on following three components: Queue Manager (**QMan** [35], pp. 6-1/299), Buffer Manager (**BMan** [35], pp. 7-1/531) and Frame Manager (**FMan** [35], pp. 8-1/594). QMan is the means by which data is passing between hardware modules, BMan role is to reduce overhead of software for memory management and FMan has three main functions (**PCD**): **P**arse fames to check packet integrity and identify protocols within frame headers, **C**lassify frames by means of keys generated (KeyGen [35], pp. 8-395/1557) based on parsing results and **D**istribute frames to queues following different distribution profiles.

Apart from standard protocols (i.e. hard-wired Parser capabilities), FMan can be configured to parse and detect up to ***three** proprietary protocol headers or application defined fields* by means of **Software Parser** ([35], pp. 8-391/982) feature. It uses NetPDL (an XML-based language for describing packet headers [37]) to define non-standard fields configured by Frame Manager Configuration Tool.

In DPAA context, the unit of transmitting data is a *frame* (detailed by a frame descriptor), stored in *frame queues*. Frame queues granularity and creation depends on software architecture and applications purposes. Furthermore, frame queues are stored in *work queues* grouped in *channels* of 8 prioritized items, where each channel identifies one hardware entity.

4.4.2 Accelerated Ceph's nodes

Combining hardware engines capabilities with software architecture, the performance of each node in a cluster can be increased, therefore for the same performance is required a lower number of nodes which leads to smaller capital costs. Also, with the advent of converged systems, each cluster node (following models presented in Section 4.3) may perform different roles in the same time.

One key for increasing performance of each node is the granularity of traffic division of each Ceph service in classes handled by specific consumers – following models presented in section 4.3.

Basically, high traffic volume needs to be divided among multiple consumers, queues and related set of cores.

In RADOS context, OSDs nodes have the highest traffic volume (I/O mostly), while MONs need low latency traffic, since they handle OSDs failures. I propose **A-OSD** (as Accelerated OSD), **A-MDS** (as Accelerated MDS) and **A-MON** (as Accelerated Monitor). For A-OSD I propose a **SMP Cluster Node** model to be able to handle high I/O traffic, and same for A-MDS to be able to handle journals and metadata (stored by RADOS) and queries from clients. A-MON doesn't need high processing power, neither ports that supports high traffic volume, so in the context of converged systems, monitors may share the node with different other applications, therefore I propose a **Per Core Cluster Node** model.

Accelerated nodes can be implemented using QorIQ^{TM} packet processors with FMan PCD capabilities and QMan scheduling features. Instructing FMan to classify and distribute frames according to each class particularities (e.g. see A-RADOS classification), a performance increase is achieved. I propose for A-OSD and A-MSD a uniform distribution on queues, distributed further uniform on cores (in case where ARFS is not used), this way clients are distributed fair across the system rather than facilitating traffic of a single client. In case of ARFS usage, the queues are uniform distributed on the cores, but when a particular frame is pushed to a core's queue, the hardware engine consider where the consumer was scheduled to run. For system traffic (i.e. system consistency messages, journaling and metadata) classification I propose to use user defined flows recognition facilitated by Software Parser. A-MON has only user defined flows and therefore the only solution is the usage of Software Parser capabilities.

4.4.3 Accelerated RADOS

RADOS is composed by two distinct clusters: one of monitors and one of OSDs, where monitors are responsible for managing a large number of OSDs. Two important keys of data clusters (including RADOS) are data reliability and availability ([22], pp. 119). Usually, to make consistent data available to all parties, these must be handled in timely fashion.

For the above reasons, I propose an accelerated RADOS (**A-RADOS**) implementation based on packet processing engines. By classification and prioritization of RADOS messages I aim to decrease latency of sensitive operations (section 2.6.1) and improve overall cluster responsiveness, thus improve data availability.

In monitor context, I consider that the most important message types are the ones that assure a consistent view of the system and are sensitives for cluster stability, followed by map distribution to monitors, responsible in its turn to distribute it further in the whole cluster: Elections and leases messages (class 1), cluster-map updates distribution to Monitors (class 2), cluster-map distribution to all other system parties (class 3).

In OSDs cluster context, I consider that the messages required for system consistency are the most essential, followed by I/O traffic (i.e. read/writes and replication strategies) and storage recovery is the least sensitive operation, so I propose the following classification: heartbeats and cluster-map update propagation (class 1), data I/O traffic (e.g. reads/writes) and data replication (class 2), OSDs recovery (class 3).

Prioritization of traffic classes is achieved by configuring weights to flow queues, where each flow queue deals with a specific class. Thus, considering the above classifications, the first traffic classes have the highest weight and their flow queues will be scheduled first, while the last traffic classes have the lowest weight and their queues will be scheduled last, and therefore the latency of sensitive traffic is decreased while achieving a better system responsiveness.

Using a QorIQ packet processor, the prioritization of three different traffic classes is achieved by splitting work queues in three groups and configuring weights to reflect the traffic prioritizations listed above.

4.4.4 Micro-benchmark results

I consider that the number of clients served per seconds by a MDS reflects its performance. For testing purposes, I have measured how much time a particular MDS request takes (in core ticks) and simulate using Per Cluster Node model, 4 different nodes that handles requests in parallel. Using this setup, I did three different tests along with kernel adjustments: without hardware acceleration (i.e. without frame classification and distribution) and using Receive Flow Steering (software implementation only) with and without core affinity.

As hardware platform I have used a QorIQ P2041 machine with 4 general purpose cores and hardware acceleration for flow steering, flow distribution and queue management (DPAA). The platform supports network connections of 1Gbps and 10Gbps, but for compatibility purposes with lab systems I've used only 1Gbps connections. The topology was very simple, composed by one PC with 1Gbps Ethernet card as client machine and P2041 as A-MDS.

For each test, I have counted the number of handled requests per second (i.e. transactions). From the tests results listed in, the improvement in terms of transactions was improved by the test with RFS support and no affinity with about 3.5% and comparable CPU utilization and with affinity the number of transactions was improved with about 12%:

Model	Requests per second					CPU util.
	node 1	node 2	node 3	node 4	Avg.	
no acc.	6867	6877	6860	6882	6871	25%
RFS	7109	7117	7117	7119	7115	25%
RFS affinity	7672	7692	7717	7725	7701	27%

For A-RADOS I did two kinds of tests (results shown below): by running concurrent requests of all three class types and by running only class III. Former tests type emphasizes that the most priority class is handled before the other classes if hardware queues prioritization support is used, while the latter shows that if only class III traffic is running the number of handled requests are comparable.

Model	Requests per second			
	Class I	Class II	Class III	Total
No prioritization	2851	2902	2959	8712
Queue prioritization	6556	1118	348	8022

Model	Requests per second			
	Class I	Class II	Class III	Total
No prioritization	0	0	6957	6957
Queue prioritization	0	0	6858	6858

5 NETWORKING CHARACTERISTICS IN CONVERGED INFRASTRUCUTRE

The main reasons of **Converged Networks** (also known as **Unified Networks Infrastructure**) are economic in nature, but also technological [38], [39], [40]: simplicity, cost saving, elasticity, requirements imposed by virtualization, better usage of servers resources, simplified management and so on.

A defining characteristic of I/O protocols for storage devices (such as SCSI) is that they do not handle lost data in timely fashion and therefore a lossless communication environment is befit. Usually SANs use Fibre Channel (used by approximately 80% of data center storage market [40]) which by design is a low latency and lossless environmental high speed. While Ethernet by design is a best effort communication environment and along with IP protocol it provides an end-to-end network for reliable transport protocols, such as TCP.

A converged support for LANs and SANs imposes a set of Ethernet enhancements (i.e. Data Center Bridging group of protocols) to enable a lossless medium. Also, Ethernet became a viable solution due to its advantages against FC networks, such as supported high speeds (up to 10Gbps, 100Gbps, 400Gbps or even new Intel's 800Gbps, while FC supports up to 2, 4, 8, 16 or 32Gbps just arriving) or lower capital-costs.

Nevertheless, there are several options that can be used with vanilla Ethernet too (e.g. iSCSI, iFCP or FCIP), but reliable protocols underneath to solve contention and congestion issues (e.g. TCP) are required. Basically, these approaches are at a lower cost, but bring a significant performance penalty due to extra encapsulation. Usually, these are used for small and medium business along with low cost Ethernet environments [41].

Fibre Channel over Ethernet (FCoE) protocol enables the transmission of FC frames over Ethernet and it relies exclusively on DCB [42] extensions, which today comprise the following protocols: **Priority Flow Control** (PFC), **Enhanced Transmission Selection** (ETS), **Data Center Bridging Exchange** (DCBx), **Quantized Congestion Notification** (QCN).

However, these extensions can be used for any loss-sensitive application that does not have contention or congestion control mechanism and requires only L2 protocols. There are even more ingenious ideas, such us improving TCP incast communication problem (i.e. multiple servers simultaneously transmit TCP data to a single aggregator: TCP performance is degraded as consequence of retransmission timeouts as result of packet loss due to overwhelmed queues at network bridges level) [43].

The main focus was on QCN (a viable alternative to Fibre Channel), used further for QCN-WFQR algorithm.

5.1 Quantized Congestion Notification

Quantized Congestion Notification [48] ([49], pp. 1071) enables peer-to-peer congestion management by dynamically adjusting throughput due to changing bottlenecks in absence management of an upper layer protocol.

It is composed by three main components: congestion, reaction and reflection points. Congestion Points (CPs) samples incoming frames and notifies sources to adjust their rates in order to keep its queue to an equilibrium value:

$\boldsymbol{F_b = -\left(Q_{offset} + \omega \cdot Q_\delta\right)}$ **,** where

$Q_{offset} = Q - Q_{equilibrium},$ (1)

$Q_\delta = Q - Q_{old}$ and ω is rate weight (taken to be 2 for baseline simulations).

Reaction point’s *rate limiters* (RLs) adjust *current rates* (CRs) to reach at *targeted rates* (TRs) using a *byte counter* in phases, as follows:

- RP receives feedback, RL sets TR as CR and CR is decreased with not more than half:

$$\boldsymbol{TR = CR}$$
$$\boldsymbol{CR = CR \cdot (1 - G_d \cdot |F_b|)}\text{, where} \qquad (2)$$

- RP enters in rate increase composed by three phases: Fast Recovery (FR), Active Increase (AI) and Hyper-Active Increase (HAI). In FR phase the reaction point tries to recover the lost bandwidth, while in the next phases it checks for extra bandwidth:

$$\boldsymbol{TR = TR + R}$$
$$\boldsymbol{CR = \frac{1}{2}(CR + TR)}\text{, where} \qquad (3)$$

R is 0 in *fast recovery*, 5Mbps for *active increase* and *i*50Mbps for and *hyper active increase* (where *i* is the *i*th smaller cycle counted by BC and timer in AI) for 10Gbps line rate. The rate increases phases have 5 cycles each. At the end of each group of 5 cycles, if no F_b has been received, RL enters in next phase (where in HAI phase, the RL throughput can reach at line rate). In cases when CRs are very small and the BC measured in time is very large, the phases are controlled by a timer. The timer is similar with the BC and counts 5 cycles of Tms (e.g. of 10ms for a 10Gbps line rate) in FR and $\frac{T}{2}$ms in AI.

Regarding both byte counter and timer, RL is in FR phase if both BC and timer are and CR is updated when at least one completes a cycle. The RL is in AI phase if at least one completes a group of 5 cycles and eventually the RL is in HAI if both BC and timer are.

5.2 QCN weaknesses and improvements

Due to existence of different types of devices within a network, the coexistence of congestion aware and unaware segments is a challenge. According to standard, this situation is solved using priority regeneration tables, therefore when frames are received from congestion unaware segments with a priority that is congestion controlled, it is remapped to different priority left unused, so this implies that at least one priority to be kept for this scenario (i.e. alternate non-CNPV priority ([49], pp. 1071).

The weakness is revealed when automatic configuration is enabled and alternate priorities are automatically chosen, and according to standard, priority chosen is the lowest unused in CNDs. This may affect QoS policies used for other traffic types and lead to unwanted traffic injections for that QoS domain from edge ports.

One solution would be to check each bridge within topology and verify all QoS requirements and priorities. Impractical, because a topology may have high number of network devices and it won’t be automatic anymore. The proposed solution contains a hybrid defense mode choice that allows both manual and automatic settings to use the same alternate priority value, therefore the administrator is able to choose the alternate priority used by automatic configuration and avoid QoS disruptions **[Patent #8891376]**.

Another weakness of QCN is the rate unfairness of multiple flows sharing the same bottleneck link. This comes from the fact that when a congestion point computes the feedbacks it doesn’t take into consideration the rate of the reaction points. This issue has been solved by two proposed algorithms: Approximate Fairness with QCN (AF-QCN [51]) and Fair Quantized Congestion Notification (FQCN [52]).

5.2.1 The Fair QCN Solution

The main idea of FQCN is that every flow has a share rate with respect to the congested queue and the feedback computed per flow which is proportional with the exceeded rate. Basically, the FQCN algorithm aims to things: first it identifies the overrated flows and second the feedbacks are individual to each source using per flow monitoring.

In my opinion, there are few implementation drawbacks: first, the byte counter per flow must be supported by hardware, because the software implementations base on interrupts or polling may have problems at high rates and second, the flow tables can be very large, in which case the management is hampered.

6 QCN BASED DYNAMICALLY LOAD BALANCING ALGORITHM

6.1 Motivation

It is almost impossible to predict the network load profile at a given time or to build an optimal topology or configuration to ensure perfect congestion domains, therefore decision-making based on profiling may lead to better performances.

I propose **QCN Weighted Flow Queue Ranking** (QCN-WFQR), an algorithm based on Quantized Congestion Notification protocol **[PIST, 2015]**. The main idea of the algorithm is to compute a series of congestion indicatives used by different entities (e.g. clients, Network Profilers or SDN controllers) to balance the traffic and serve better the purposes of applications. The algorithm is generic, so that it can be used in old and well known computer networks and in the novel software defined networks paradigm.

6.2 Alternative solutions

In congestion aware networks (i.e. QCN based), each reaction point receives feedback messages from multiple congestion points and adjust rate accordingly. To achieve a better congestion profile in the network, one solution would be that each congestion source (i.e. reaction point) to create a data base with congestion indicatives per each congestion point from which it receives feedbacks, thus each reaction point in the system is associated with indicatives received from each congestion point (CPID: congestion point ID embedded in each congestion notification message): t (time of occurrence), p (priority of sampled frame), $Q_{nz}F_b$ (quantized value of feedback), Q_{off} (queue size excess) and Q_δ (rate excess) **[App. #20150023172]**. Furthermore, reaction points may use these databases for performance analysis, based on which system administrators can change topology configuration and improve the system performance. Or reaction points may use them to adapt and influence different aspects of the network devices and obtain a better congestion profile in the system.

I chose to follow a different path, where instead of databases created by each reaction point, each congestion point creates profiles for each reaction point it servers. The main idea of the algorithm is to compute different congestion indicatives, local (by each congestion point) and system wide. Based on these indicatives the congestion profilers or SDN controllers are instructed to dynamically move flows between congestion points for a better congestion profile at system level. Also, targets can be instructed to choose a device that has the lowest contribution to system congestion.

6.3 QCN Weighted Flow Queue Ranking

The main idea is that each node keeps a local information database that reflects the congestion status of a portion of the network from its point of view. Further, all local information databases are gathered by a central entity and compute a database related to the entire system. Depending on purposes of the applications, system wide or local congestion information may be used to achieve a better usage of the network and improve the system performance.

The local congestion indicatives are **flow shares**, **queues weights** and **queue ranks**, while system congestion indicatives are **flow weights** and **reaction point (or device) weights**.

Furthermore, I propose a method of QCN-WFQR usage in distributed and parallel file systems, where by using system wide congestion indicatives, a particular data clone is chosen to achieve a better balanced traffic load within the network.

Also, I propose a method of using QCN-WFQR for distributing traffic workload between multiple servers hosting the same application and migrating already established flows to alternate, less congested paths, thus reducing the need to slow down the traffic at the source.

6.3.1 QCN-WFQR indicatives

6.3.1.1 Flows

A computing node (e.g. server) in distributed systems offers services by means of flows. A service is identified by a service id which can be encapsulated into congestion notification tags ([49], pp.1096).

6.3.1.2 Ranking

A **queue rank** (4) within a port is defined as the number of flows within that queue (***no multicast support***):

$$\boldsymbol{R_t^{q_k} = COUNT_t\langle f_i \rangle, f_i \in Q^{f_i}} \qquad (4)$$

6.3.1.3 Flow shares

A **flow share** (Eq. 6, 12) is defined as a congestion measure from each topology's node point of view for a specific flow related to a port's queue (i.e. congestion point).

6.3.1.3.1 Flow shares based on standard QCN

The flow share has to contain information to be able to compare flows that shares the same congestion points and also flows with different routes and different congestion points. Therefore, the share is a combination of three indicatives: feedback average within a time frame, the frequency of feedbacks transmitted to a flow and queue rank within the same time frame. The feedback is a measure of congestion related to the queue regardless the rate of source, so I assume that higher rates imply higher probabilities of receiving feedbacks for a flow. The rank of a queue reflects the number of flows affected in case of congestion (i.e. the main idea is to affect the smallest number of flows in case of congestion).

To reflect the frequency of transmitted feedbacks for a flow, when a feedback is sent to a specific reaction point for a specific flow, all the other flows within the queue recalculates their feedbacks averages by considering a null feedback sample, therefore flows with smaller rates are more likely to have a larger number of null samples than flows with higher rates and for implementation purposes I have used exponential moving average function.

$$\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right) = \begin{cases} \alpha_t \cdot \Psi_t^{q_k}\left(Fb_t^{q_k}\right) + (1-\alpha_t) \cdot \Psi_{t-1}^{q_k}\left(Fb_{t-1}^{q_k}, f_i\right), q_k\ sample f_{i,} \\ (1-\alpha_t) \cdot \Psi_{t-1}^{q_k}\left(Fb_{t-1}^{q_k}, f_i\right), otherwise \\ \underset{f_j \in \mathcal{F}^{q_k}}{\boldsymbol{MIN}} \Psi_t^{q_k}\left(Fb_t^{q_k}, f_j\right), initial\ value \end{cases} \qquad (5)$$

Where $\Psi_t^{q_k}\left(Fb_t^{q_k}\right)$ is quantized feedback at moment *t* relative to queue *k*.

And $\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right)$ is quantized feedback for f_i at moment *t* relative to queue k.

So, flow share for flow f_i computed by queue q_k at moment *t* is:

$$\boldsymbol{Share_t^{q_k}(f_i) = \Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right) \times R_t^{q_k}} \tag{6}$$

6.3.1.3.2 Flow shares based on FQCN

If FQCN algorithm is used, the feedback is computed by considering the bytes received in a specific time frame *T*. Most network devices have the ability to provide average throughput per queue basis, but it is needed per flow basis, which it may be considered a drawback since it is very hard to implement this feature at hardware level. Considering the above motivation, there is a possibility to approximate throughput per flow by using the same method of exponential moving average, as shown below (throughput at moment *t* for flow f_i related to queue *k* is):

$$Rate_t^{q_k}(f_i) = \alpha_t \cdot Bytes(frame \in f_i) + (1 - \alpha_t) \cdot Rate_{t-1}^{q_k}(f_i)\ , \quad \text{Where } \alpha_t = 1 - e^{-\frac{\Delta t}{T}} \tag{7}$$

Furthermore, to compute the bytes received in time frame *T* for flow f_i related to queue *k* is easy:

$$B_t^{q_k}(f_i) = T \cdot Rate_t^{q_k}(f_i) \tag{8}$$

The FQCN shares and FQCN fine grained shares are computed as follows:

$$M_t^{q_k}(f_i) = \frac{w_i^{q_k}}{\sum w_i^{q_k}} \cdot \sum B_t^{q_k}(f_i) \tag{9}$$

$$MH_t^{q_k}(f_i) = \frac{w_i^{q_k}}{\sum_{S^H} w_i^{q_k}} \cdot \sum B_t^{q_k}(f_i) \tag{10}$$

And the FQCN feedback for each culprit is computed as shown below:

$$\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right) = \frac{\frac{B_t^{q_k}(f_i)}{w_i^{q_k}}}{\sum_{S^R} \frac{B_t^{q_k}(f_i)}{w_i^{q_k}}} \cdot \Psi_t^{q_k}\left(Fb_t^{q_k}\right) \tag{11}$$

Where $\Psi_t^{q_k}\left(Fb_t^{q_k}\right)$ is the feedback quantized at 64 bit and $\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right)$ is quantized feedback for flow f_i and S^R is the fine grained culprit list

Each event time, the QCN-WFQR flow share is updated, as shown below:

$$\boldsymbol{Share_t^{q_k}(f_i) = \Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right) \times R_t^{q_k}} \tag{12}$$

The difference between QCN-WFQR based on standard QCN and QCN-WFQR based on FQCN is that FQCN consider the flow rates when computes feedback, therefore the flow share contains enough information to be able to compare flows that shares the same congestion points and also flows with different routes and different congestion points.

6.3.1.4 Weighting

A **flow weight** (13) is defined as a congestion measure from *system point of view* for a specific flow, while a **reaction point weight** (14) is defined as a congestion measure from *system point of view* for a specific reaction point. Therefore, a flow weight at moment *t* for flow f_k is computed as the sum of flow shares received from all nodes (i.e. Q^{f_k}):

$$\boldsymbol{Weight_t(f_k)} = \sum_{q_i \in Q^{f_k}} \boldsymbol{Share_t^{q_i}(f_k)} \tag{13}$$

And a reaction point weight at moment *t* for R_k is computed as the sum of all flow weights for flows whose rates are adjusted by it (i.e. $\mathcal{F}^{R_k}$):

$$\boldsymbol{Weight_t(R_k)} = \sum_{f_i \in \mathcal{F}^{R_k}} \boldsymbol{Weight_t(f_i)} \tag{14}$$

A **queue weight** (15) at moment t is defined as a local congestion measure related to queue q_k and computed as sum of all feedback averages of flows controlled by the queue (i.e. $\mathcal{F}^{q_k}$) and its rank ($R_t^{q_k}$), as follows:

$$\boldsymbol{Weight_t(q_k)} = \left[\sum_{f_i \in \mathcal{F}^{q_k}} \boldsymbol{\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right)}\right] \times \boldsymbol{R_t^{q_k}} \tag{15}$$

This weight enables the comparison of queues based on which a specific flow may be routed or migrated achieving a better balanced traffic within the network.

6.3.2 QCN-WFQR algorithm based on standard QCN

Local congestion indicatives calculus: <u>flow shares</u>, <u>queue ranks</u> and <u>queue weights</u>

```
procedure Q-DO-ENQUEUE(q, p)                          º add received packet p to sampled queue q
    Fb ← QCN-FEEDBACK(q, p)                           º compute feedback Fb for received packet p
    if Fb < 0 then                                    º if computed Fb is negative then do QCN-WFQR algorithm
        call QCN-WFQR-UPDATE-FLOW-TABLE(q, p)         º update flow table is packet p belongs to new flow
        call QCN-WFQR-UPDATE-QUEUE-RANK(q)            º update queue rank (size of flow table)
        call QCN-WFQR-UPDATE-INDICATIVES(q, p, Fb)    º update QCN-WFQR congestion indicatives
    end if
    add (q, p)                                        º add packet p to queue q
end procedure
```

```
procedure QCN-WFQR-UPDATE-FLOW-TABLE(q, p)            º update flow table regarding packet p (if new flow)
    if flow(p) ∉ flow_table(q) then                   º if packet p belongs to a new flow, then add flow to table
        add (flow(p), table(q))
    end if
    call QCN-WFQR-AGE-FLOW_TABLE(q)                   º remove from table older flows slipped from time frame
end procedure
```

```
procedure QCN-WFQR-UPDATE-INDICATIVES(q, p, Fb)     ° local congestion indicatives: flow shares, q rank and weight
    c_time ← current time                           ° current time (for EMA calculus)
    min_ema ← QCN-WFQR-GET-MIN-EMA(q)               ° minimum EMA from all flows within flow table
    q_weight ← 0
    for “each entry e” ∈ flow_table(q) do           ° for each flow from table related to queue q
        if e.ema_old = 0 then                       ° initialize OLD EMA with minimum if first
            e.ema_old ← min_ema
        end if
        e.ema ← EMA (p, Fb, c_time, e);             ° new EMA for flow related to entry e
        e.flow_share ← e.ema * rank(q)              ° compute flow share for entry e
        q_weight ← q_weight + e.flow_share          ° update queue weight
    end for
end procedure
```

System related congestion indicatives calculus: <u>flows weights</u> and <u>reaction points weights</u>

```
procedure QCN-WFQR-QUERY-INDICATIVES(cpid)          ° query each CPID for changes in flow table
    for “each flow f” ∈ CPID do                     ° for each flow received
        call QCN-WFQR-UPDATE-SYS-TABLE(flow)        ° if indicatives changed update sys table
    end for
end procedure

procedure QCN-WFQR-UPDATE-SYS-TABLE(flow)           ° update sys table
    call QCN-WFQR-UPDATE-SHARE(flow)                ° update flow’s share
    call QCN-WFQR-UPDATE-SHARE-WEIGHT(flow)         ° update flow’s weight
    call QCN-WFQR-UPDATE-RP-WEIGHT(flow)            ° update reaction point weight
end procedure
```

6.3.3 Algorithm analysis

6.3.3.1 Flows shares analysis

The main idea of flow share is that it has to reflect the reaction point rate and in the same time to reflect the feedback values. I assumed that the reaction point rate is reflected by the frequency with which the feedback is sent to a specific flow, while the distance until congestion is outlined by the feedback. Basically, the former situation enables comparison of flows that shares the same queue but with different rates, while the later situation enables comparison of flows having same rate but on different routes and different queues with different congestion conditions (i.e. different feedback values). Based on synthetic simulations was shown that the flow share responds for both rate and feedback variation.

6.3.3.2 Congestion indicatives analysis

At a particular moment a flow cannot have more than one congestion point, because a congestion point implies the smallest throughput within a flow path. But, if a different point is congested due to profile traffic changes, then the throughput must decrease even more, therefore the old congestion point will be released.

In terms of flow comparison, I distinguish three different situations as follows:

1. **Solitary congestion islands**: the compared flows shares the same queue (congest the same point);
2. **Disconnected congestion islands:** the compared flows congest different queues (different congestion points);

3. **Hybrid congestion islands:** the compared flows are in solitary congestion island, but occasionally different flows may cause other congestion points and move flows into disconnect congestion islands or vice-versa.

6.3.3.3 QCN-WFQR simulator

For simulations was used an open-source simulator: NS-3 (a discrete-event network simulator), [53]. The simulator of QCN-WFQR implementation has the following main objects: reaction point, congestion point and profiler or controller in case of SDN.

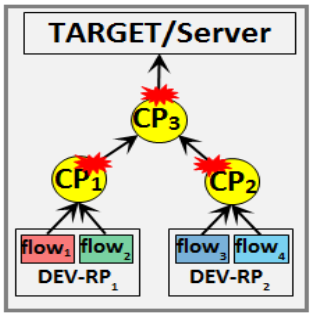
6.3.3.4 QCN-WFQR simulation analysis

6.3.3.4.1 Flows contributions to system congestion profile

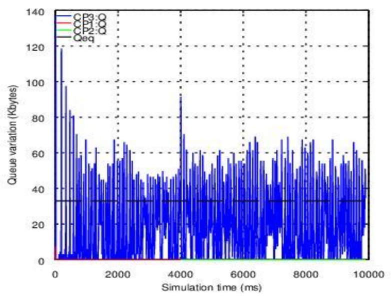
This first test aims to emphasize the flows contributions to system congestion profile: flow shares, flow weights and reaction point weights. In another words, reaction points with low flow rates implies lower weights, therefore tenants can cooperate and decide upon different services locations and achieve a fair congestion profile and a better system performance.

For simulation purposes I have used a binary tree topology with three congestion controlled nodes. Each child node has a device with a reaction point and two flows each, while the parent node has a device with a server (as target). From QCN stand point, each congestion controlled node has a transmission queue used for feedback calculus. From QCN-WFQR stand point, flow share calculus uses a window of 2 seconds, but the flow table has an age function of 250ms.

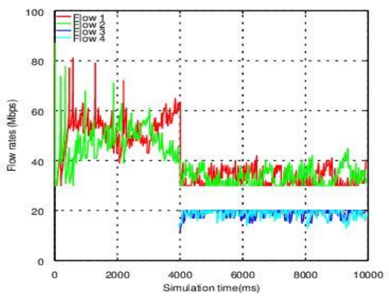
It can be seen from the simulations results, shown below, that the CP$_3$'s local share variation, the flow weights and reaction points weights follows the flows rates profile:



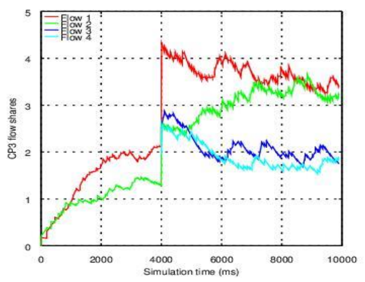
(a) Simulation topology



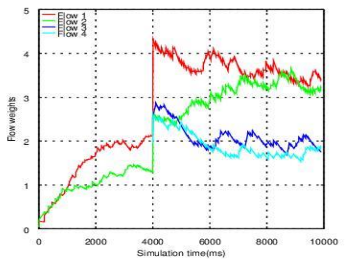
(b) CPs queue variation



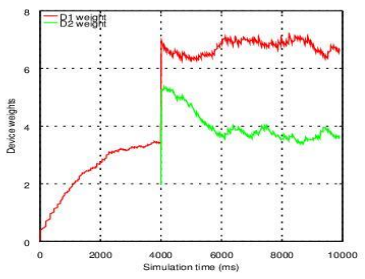
(c) Flows rates variation



(d) CP3 flow shares variation



(e) Flow weights variation



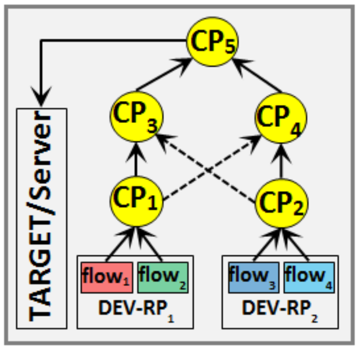
(f) Reaction points (devices) weights variation

QCN-WFQR (NS3): simulation results for variation of flow related indicatives

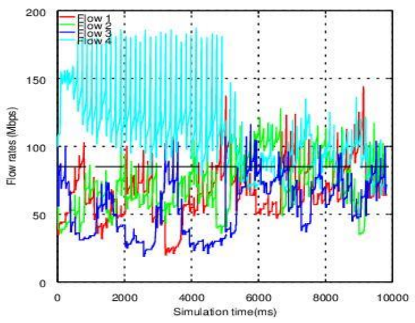
6.3.3.4.2 Queues weights variations

The second simulation aims to emphasize that flows rates variations reflects queues weights variations, thus migration decisions may be taken to balance traffic and achieve a fair congestion profile among system congestion points.

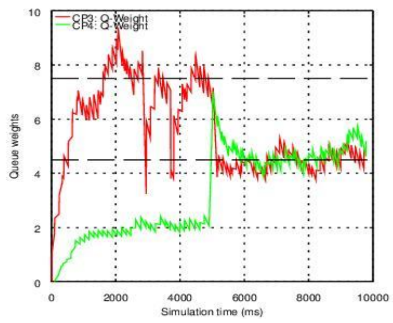
Same as for the former simulation, a tree based topology was used with alternative paths, so each of the four device's flows has 2 alternative paths. In the first 5 seconds of simulation time, the congestion point 3 forwards and samples 3 flows (flow 1, 2 and 3), while congestion point 4 forwards and samples one flow (flow 4). It can be seen from the results depicted below that the CP3's weight queue is significantly higher compared to CP4, which servers only 1 flow and also, from device weights stand point, RP_1 flows ($flow_{1\&2}$) has higher throughput capabilities, then it has a higher probability to receive feedbacks, so RP_1 weight is higher than RP_2 weight:



(a) Simulation topology



(b) Flow rates variations



(c) CP3, CP4 queues weight variation

QCN-WFQR (NS3): simulation results for variation of queue related indicatives

6.3.4 QCN-WFQR applications

6.3.4.1 Elections in distributed systems with homogenous nodes

In general, considering a distributed system (or cloud) with multiple classes of services, where each class has many homogenous nodes. By using QCN-WFQR, I propose an election method to maximize bandwidth usage while minimize congestion in the network and improve overall system response time.

If the number of clients varies and their needs are unknown, then the network congestion pattern is basically nondeterministic. When a particular service is required, I propose that the choice of a node to be based on congestion indicatives provided by QCN-WFQR system profiler.

For exemplification, in distributed and parallel file systems, a file is divided in data chunks (or objects) replicated on multiple storage devices, so a data segment can be transferred from any replica. Most distributed file systems uses a static distribution (e.g. CRUSH – Controlled Replica Under Scalable Hashing), but with advent of converged networks a static distribution may not be the best choice. Therefore, using QCN-WFQR the replica may be chosen dynamically and improve system responsiveness by trying to avoid network bottlenecks. In this case is worth to mention that the reaction points are the storage devices.

The heart of this method is that the local congestion indicatives communicated to system profiler and the congestion indicatives communicated to the client must reflect the current system congestion status. So, if the information is communicated too often, then the system profiler may be overwhelmed or if it is too seldom, then the information will not reflect the current congestion status of the system. This means that the algorithm is very sensitive to wild congestion profile variations, so I propose to use larger data chunks.

Besides congestion indicatives communication, the performance and implementation of the system profiler greatly influence the system behavior: distributed implementation and logically centralized or completely centralized implementation. Plus, the profiler must have a very low latency connection with each congestion point and each client, so that the congestion indicatives to be exchanged in timely fashion.

6.3.4.2 Dynamically balanced flows in computer networks

The algorithm performance is directly influenced by the latency with which the Network Profiler or SDN controller gets the relevant congestion indicatives, decides and move flows while achieving a better balanced traffic within the system topology.

First, the profiler/controller design follows either a centralized or distributed model [54]. Each method has its own pros and cons. For example, a centralized design has scaling limitation, bigger bottleneck probability, single point of failure, but it has a simpler complexity (basically it follows a multi-threaded design over SMP systems) and it has strong semantic consistency, while a decentralized design scales up easy and meet performance requirements, handle better data plane resilience and scalability, fault tolerant, but it has a weak consistency semantics (and it is worth to mention that a strong consistency implies complicated implementation of synchronization algorithms) and it has much more complex implementation [54].

Second, how congestion indicatives are communicated is strategic to relevance of the decisions based on QCN-WFQR algorithm. Basically, it has to either transmit information at short intervals overloading the controller or to filter and condense the information at the source before transmitting it to the controller.

7 CONCLUSIONS

This thesis begins with a general analysis of the following distributed storage systems: AFS, GFS, GPFS, Lustre and Ceph, studding different aspects such as taxonomies, architecture details and network characteristics. The analysis is followed by a proposal for eLearning infrastructures, as a case study. Afterwards, several optimizations to distributed storage systems are proposed and a novel algorithm for dynamic load balancing in congested aware networks: **QCN-WFQR**.

7.1 The original contributions of the thesis

Contributions to the analysis of distributed storage systems (chapter 2)

I proposed a simple and comprehensive taxonomy for storage systems considering four important characteristics: locality, sharing, distribution level and semantics (section 2.1). For characterization of distributed storage systems I proposed a relevant set of requirements based upon existing documentation in the field (section 2.2). Also, I highlighted the scale-out methods at different layers of file systems, thus building different types of storage systems, such as SAN or NAS (section 2.3). And to emphasize the similarities between distributed and centralized storage systems I proposed a generalized layout (section 2.5.3).

The analysis of distributed storages and related contributions were published in **[PIST, 2014/1]**.

Contributions to the infrastructure for eLearning environments (chapter 3)

In this chapter I proposed a parallel and distributed file system adapted for eLearning environments, in turn hosted by cloud systems. The proposed storage system can be considered an alternative solution to OGSA-GFS. It has two main components: a flat address space based on RADOS (Ceph's object store) and a module for managing the semantic aware metadata. I proposed a native implementation of the semantic aware metadata with content-based semantics to improve the searches of data in the system. In terms of network, I proposed a hierarchical topology that facilitates south-north throughput with links added to decrease the latency between RADOS's nodes.

These contributions were published in **[PIST, 2014/2]**.

Optimizations based on packet processing engines (chapter 4)

The main contribution in this chapter is the proposal of several solutions to optimize traffic intensive clusters using integrated systems of multi-general purpose cores with packet processing engines (section 4.2 and 4.3). While Ceph addresses different optimizations at system level, my contributions focus on improving each node performance which in the end reflects the overall system performance (section 4.4).

First, the performance of each node is improved by classifying and distributing requests among cores, facilitating the parallelization of traffic consumers. Furthermore, the OS scheduler decisions are leveraged by using Accelerated Receive Flow Steering mechanisms (section 4.3). Therefore, I proposed two models and two different implementations for each model: one in kernel targeting efficiency and one in user-space facilitating a faster development and flexible license policy. As case study, I created a model using QorIQ^{TM} platforms (section 4.4.1) for Ceph's nodes. Each model for each node type is designed based on related flow types and traffic volume (section 4.4.2).

Second, the latency of sensitive operations is decreased and the overall cluster responsiveness and availability is improved by adjusting the weights of hardware queues. Accelerated RADOS (A-RADOS) divides the traffic of monitors and OSDs in three different classes with different priorities based on importance related to Ceph (section 4.4.3).

Performance measurements that showed improved number of transactions per second and the results of the RADOS's messages processed based on priorities were presented in section 4.4.4. Besides the presented improvements there are inherent advantages of using multi-core integrated systems such as low power consumption and appealing price per performance.

The proposed optimizations based on packet processing engines and the micro-benchmark results were published in **[PIST, 2013]**.

Solutions for issues occurred in converged infrastructure for data centers (chapter 5)

A group of protocols were added to Ethernet in context of converged infrastructure for data centers (i.e. collapsing tiers and unification of general purpose networks with networks for distributed storage systems): PFC, ETS, DCBx and QCN. The main focus was on QCN, which has several weaknesses, such as unwanted traffic injections for QoS domains from edge ports and rate unfairness of flows that shares the same bottleneck.

I was part of a team that proposed a patented solution for automatic configurations of QCN parameters through LLDP in a hybrid network, where segments with and without QCN coexists considering different QoS requirements. The solution implies a new hybrid defense mode choice that allows both manual and automatic settings to use the same alternate priority value, thus the injection of unwanted traffic for a specific QoS domain is avoided (section 5.2).

The patent is public and can be inspected at **[Patent #8891376]**.

Algorithm for dynamic load balancing: QCN-WFQR (chapter 6)

QCN protocol alleviate congestion in the network, but it does not address the congestion problem as a whole, where a uniform distribution of congestion domains will lead to a better utilization of bandwidth. Therefore, I proposed a partial solution that tries to solve this issue. The proposal was based on the choice of the gateway with the biggest distance until congestion and later on, it led to a more general solution that was patented, where the congestion sources (i.e. reaction points) are building databases for each congested point. This database can be used for performance analysis or can be used to influence different aspects of the network devices (section 6.2).

I have followed a different path and I proposed a **novel** and more **complete** algorithm (**Quantized Congestion Notification – Weighted Flow Queue Ranking)** that tries to solve this problem. The algorithm **QCN-WFQR** computes multiple congestion indicatives that are measures of the network load generated by flows in different points in the network, based on which different cooperative or automatic decisions may be taken to balance the workload and achieve a less congested network (section 6.3.1).

The algorithm is capable to compute the local contribution of each flow in different network points ($\boldsymbol{Share_t^{q_k}(f_i)}$) and also the system wide contribution of each flow ($\boldsymbol{Weight_t(f_k)}$). Besides flow contribution, the system is also capable to compute the contribution of each congestion source ($\boldsymbol{Weight_t(R_k)}$) to the network load. Based on these congestion indicatives any component within the system can determine the best appropriate *service* (congestion source/reaction point) location in order to cooperatively balance the workload in the network. Furthermore, I present the analysis of *flows shares* and the system wide congestion geometry (i.e. congestion islands) related to each flow (sections 6.3.3.1 and 6.3.3.2). It is worth to mention, that the QCN-WFQR algorithm makes use of the exponential moving average to store flows shares in order to minimize the memory usage on network nodes (section 6.3.1.1).

Along with the congestion indicatives listed above, the algorithm is capable to compute the importance of each queue ($\boldsymbol{R_t^{q_k}}$) related to the congestion in the system in terms of the number of handled flows and the congestion degree of each queue ($\boldsymbol{Weight_t(q_k)}$) related to all handled flows. Using these congestion indicatives, a network can be reprogrammed to optimally balance the workload between different congestion points.

For simulations purpose I implemented QCN-WFQR in NS3 (an open source discrete-event network simulator), where besides algorithm modules, I implemented also the following QCN modules: rate limiter, reaction point and congestion point (section 6.3.3.3). I did two different simulations aiming flows contributions to the congestion profile and variations of queues weights using tree based topologies (section 6.3.3.4). The former simulation revealed that the traffic of congestion sources is reflected in the congestion indicatives of flows and reaction points (section 6.3.3.4.1), while the later simulation revealed the reflection in the congestion indicatives of queues (section 6.3.3.4.2).

I proposed two applications of QCN-WFQR: elections of replicas in distributed storage systems based on congestion indicatives in order to achieve a better balanced congestion profile while achieving a much responsive system and a dynamically load balancer, more appropriate for SDN networks.

The patented solution can be inspected at **[App. #20150023172]**, while the QCN-WFQR algorithm with the simulations results were published in **[PIST, 2015]**.

7.2 Future work

Since, researches to optimize nodes were conducted only at ingress side, now researches are focused on egress side. The QCN-WFQR algorithm has a sound approach with promising impact in the networking field, especially in Software Defined Networking. Therefore, now the research focuses on implementation of different algorithms, such as Dijkstra's algorithm, using different congestion indicatives as path costs proposed in the present thesis. Since the distributed systems has several characteristics related to scaling methods, investigations are directed towards implementation of logically decentralized controllers with QCN-WFQR support and the relevance of decisions based on latency and architectures of network topologies.

8 AUTHOR'S PUBLICATIONS

8.1 Patents

[App. #20150023172]. **Sorin A. Pistirica,** Dan A. Calavrezo, Casimer M. DeCusatis, Keshav G. Kamble, "**Congestion Profiling of Computer Network Devices**", Patent Pending, USPTO: http://patents.justia.com/patent/20150023172, Jul 16 – 2013

[Patent #8891376]. **Sorin A. Pistirica**, Dan A. Calavrezo, Keshav G. Kamble, Mihail-Liviu Manolachi, "**Quantized Congestion Notification—defense mode choice extension for the alternate priority of congestion points**", USPTO: http://patents.justia.com/patent/8891376, Oct 07 – 2013

8.2 Articles

[PIST, 2013] Pistirica Sorin Andrei, Caraman Mihai Claudiu, Moldoveanu Florica, Moldoveanu Alin, Asavei Victor, "**Hardware acceleration in CEPH Distributed File System**", ISPDC: IEEE 12th International Symposium on Parallel and Distributed Computing, Bucharest, June 2013, pp. 209-215, **IEEE Indexed**

[PIST, 2014/1] Pistirica Sorin Andrei, Victor Asavei, Horia Geanta , Florica Moldoveanu , Alin Moldoveanu , Catalin Negru , Mariana Mocanu, "**Evolution Towards Distributed Storage in a Nutshell**", HPCC: The 16th IEEE International Conference on High Performance Computing and Communications, August 2014, Paris, pp. 1267-1274, **IEEE Indexed**

[PIST, 2014/2] Pistirica Sorin Andrei, Asavei Victor, Egner Alexandru, Poncea Ovidiu Mihai, "**Impact of Distributed File Systems and Computer Network Technologies in eLearning environments**", eLSE: Proceedings of the 10th International Scientific Conference "eLearning and Software for Education", Bucharest, April 2014, Volume 1, pp. 85-92, **ISI Indexed**

[PIST, 2015] Pistirica Sorin Andrei, Poncea Ovidiu, Caraman Mihai, "**QCN based dynamically load balancing: QCN Weighted Flow Queue Ranking**", CSCS: The 20th International Conference on Control Systems and Computer Science, Bucharest, May 2015, Volume 1, pp. 197-205, **ISI Indexed**

[ASAV, 2014] Asavei Victor, Moldoveanu Alin, Moldoveanu Florica, **Pistirica Sorin Andrei**, "**Lightweight 3D MMO Framework with High GPU Offloading**", ICSTCC: 18th International Conference On System Theory, Control and Computing, October 2014, Sinaia, pp. 708-714, **ISI Indexed**

[SIMI, 2015] Simion Andrei, Asavei Victor, **Pistirica Sorin Andrei**, Poncea Ovidiu, "**Practical GPU and voxel-based indirect illumination for real time computer games**", CSCS: The 20th International Conference on Control Systems and Computer Science, May 2015, Bucharest, Volume 1, pp. 379-384, **ISI Indexed**

[GRAD, 2015] Alexandru Grădinaru, Alin Moldoveanu, Victor Asavei, **Sorin Andrei Pistirica**, "**Case Study - OpenSimulator for 3D MMO Education**", eLSE: Proceedings of the 11th International Scientific Conference "eLearning and Software for Education", Bucharest, April 2015, **ISI indexed**

[ASAV, 2015] Victor Asavei, Alexandru Gradinaru, Alin Moldoveanu, Sorin Andrei Pistirica, Ovidiu Poncea, Alexandru Butean, "**Massively Multiplayer Online virtual spaces-classification, technologies and trends**", U.P.B. Sci. Bull., 2015, (in press, accepted for publication)

9 BIBLIOGRAPHY

[1] B-tree File system, https://btrfs.wiki.kernel.org/index.php/Btrfs_design

[2] Extended File System, http://e2fsprogs.sourceforge.net/ext2intro.html

[3] EMC Storage Pool Deep Dive: Design Considerations & Caveats, http://vjswami.com/2011/03/05/emc-storage-pool-deep-dive-design-considerations-caveats/, 2011

[4] Linux Volume Management, http://tldp.org/HOWTO/LVM-HOWTO/

[5] G.C.Foxy, K.A.Hawick, A.B.White, "Characteristics of HPC Scientific and Engineering Applications", January 1996

[6] Chang-Soo Kim, Gyoung-Bae Kim, Bum-Joo Shin, "Volume Management in SAN Environment", Parallel and Distributed Systems ICPADS, 2001, pp. 500-505

[7] Andrew S. Tanenbaum, "Distributed Operating Systems", August 25th 1994 by Prentice Hall

[8] Christian Bandulet, "The Evolution of File Systems", http://www.snia-europe.org/objects_store/Christian_Bandulet_SNIATutorial%20Basics_EvolutionFileSystems.pdf, Storage Networking Industry Association, 2012

[9] Michael Factor, Kalman Meth, Dalit Naor, Julian Satran, Ohad Rodeh, "Object Storage: The Future Building Block for Storage Systems", Local to Global Data Interoperability - Challenges and Technologies. IEEE Computer Society. pp. 119–123, 2005

[10] Information Technology - SCSI Object Based Storage Device Commands (OSD), Revision 3, 1 October 2000

[11] Information Technology - SCSI Object Based Storage Device Commands -2 (OSD-2), Revision 4, 24 July 2008

[12] J. Satran A., Teperman, "Object Store Based SAN File Systems", International Symposium of Santa Caterina on Challenges in the Internet and Interdisciplinary Research, 2004

[13] OpenAFS, Open source implementation of the Andrew Distributed File System, http://www.openafs.org/

[14] ARLA, Free AFS implementation from KTH, http://www.stacken.kth.se/project/arla/html/arla.html

[15] Sanjay Ghemawat, Howard Gobioff and Shun-Tak Leung, The Google File System , 19th ACM Symposium on Operating Systems principles, pp. 29-43, December 2003

[16] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, "The Hadoop Distributed File System", 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, pp. 1-10

[17] Frank Schmuck and Roger Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters", Proceedings of the Conference on File and Storage Technologies, pp. 231–244, January 2002

[18] Peter J. Braam et al, The Lustre Storage Architecture, available at www.lustre.org, 2004.

[19] Sage A. Weil, Scott A. Brandt, Ethan L. Miller and Darrell D. E. Long, —Ceph: A Scalable, High-performance Distributed File System , 7th Conference on Operating Systems Design and Implementation, November 2006

[20] Sage A. Weil, Andrew W. Leung, Scott A. Brandt and Carlos Maltzahn, —RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters, Petascale Data Storage Workshop, November 2007

[21] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Carlos Maltzahn, "CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data", Proceedings of the 2006 ACM/IEEE conference on Supercomputing, 2006

[22] Sage A. Weil, Reliable, Scalable and High-Performance Distributed Storage, PhD thesis, 2007

[23] Honicky, R.J. ,Miller, E.L., "RUSH: Balanced, Decentralized Distribution for Replicated Data in Scalable Storage Clusters", Proceedings of the 20th IEEE - 11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003, pages 146–156

[24] R. J. Honicky, Ethan L. Miller, "Replication Under Scalable Hashing: A Family of Algorithms for Scalable Decentralized Data Distribution", Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004

[25] Andrew W. Leung, Ethan L. Miller, Stephanie Jones, "Scalable Security for Petascale Parallel File Systems", Proceedings of the 2007 ACM/IEEE conference on Supercomputing, 2007

[26] Yan Li, Nakul Sanjay Dhotre, Yasuhiro Ohara, Thomas M. Kroeger, Ethan L. Miller, Darrell D. E. Long, "Horus: Fine-Grained Encryption-Based Security for Large-Scale Storage", Proceedings of the sixth workshop on Parallel Data Storage, 2013, pp. 19-24

[27] PanFS, Object RAID, https://www.panasas.com/products/panfs/object-raid

[28] Goodman, J.R.,Sequin," Hypertree: A Multiprocessor Interconnection Topology", IEEE Transactions on Computers, 1981, pp. 923-933

[29] Direct interconnection networks, http://pages.cs.wisc.edu/~tvrdik/5/html/Section5.html

[30] Andy D. Hospodor, Ethan L. Miller, "Interconnection Architectures for Petabyte-Scale High-Performance Storage Systems", 12th NASA Goddard Conference on Mass Storage Systems and Technologies, April 2004

[31] White Paper, Accelerating High-Speed Networking with Intel® I/O Acceleration Technology

[32] Karthikeyan Vaidyanathan, Dhabaleswar K. Panda, "Benefits of I/O Acceleration Technology (I/OAT) in Clusters", International Symposium on Performance Analysis of Systems & Software, pp. 220-229, 2007

[33] Scaling in the Linux Networking Stack, https://github.com/torvalds/linux/blob/master/Documentation/networking/scaling.txt, December 2011

[34] Luigi Rizzo, "netmap: a novel framework for fast packet I/O", Proceedings of the 2012 USENIX Annual Technical Conference, pp. 101-112, June 2012

[35] Freescale Semiconductor, Inc., —QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual , 2011

[36] Freescale Semiconductor, Inc., —Frame Manager Configuration Tool Example Configuration and Policy, 2013

[37] Fulvio Risso, and Mario Baldi, NetPDL: An Extensible XML-based Language for Packet Header Description, Computer Networks (COMPUT NETW), vol. 50, no. 5, pp. 688-706, 2006

[38] Sangam Racherla, Silvio Erdenberger, Harish Rajagopal, Kai Ruth, "IBM Red Book, Storage and Network Convergence Using FCoE and iSCSI", International Technical Support Organization: Storage and Network Convergence Using FCoE and iSCSI, January 2014

[39] White Paper, Emulex, Top-5 Reasons for Deploying Network Convergence, 2009

[40] White Paper, Forrester Consulting, "Benefits Of SAN/LAN Convergence. Evaluating Interest In And Readiness For Unified Fabric", 2009

[41] White Paper, “Fabric convergence with lossless Ethernet and Fibre Channel over Ethernet”, 2008

[42] White Paper, Ethernet Alliance, Data Center Bridging, 2010

[43] Devkota P., Reddy A.L.N., "Performance of Quantized Congestion Notification in TCP Incast Scenarios of Data Centers", Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2010

[44] IEEE Standard 802.3x PAUSE, 1997

[45] 802.1Qbb PFC – Priority-based Flow Control, 802.1Qbb Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment 17, 2010

[46] 802.1Qaz ETS – Enhanced Transmissions Protocol, 802.1az Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment 18,2011

[47] Manoj Wadekar (Qlogic), et al, DCB Capability Exchange Protocol Base Specification Rev 1.01

[48] 802.1Qau QCN – Quantized Congestion Notification, , 802.1Qau Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment 13,2010

[49] IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks, 31 August 2011

[50] 802.1AB Station and Media Access Control Connectivity Discovery, IEEE Standard for Local and metropolitan area networks, 2009

[51] Abdul Kabbani, Mohammad Alizadeh, Masato Yasuda, Rong Panz and Balaji Prabhakar, AF-QCN: Approximate Fairness with Quantized Congestion Notification for Multi-tenanted Data Centers, Proceedings of the 18th IEEE Symposium on High Performance Interconnects, pp. 58-65, 2010

[52] Yan Zhang, Nirwan Ansari, Fair Quantized Congestion Notification in Data Center Networks, IEEE Transactions on Communications, pp. 4690 - 4699, 28 November 2013

[53] Discrete Network Simulator, http://www.nsnam.org/

[54] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky and Steve Uhlig, Software-Defined Networking: A Comprehensive Survey, Proceedings of the IEEE Vol. 103, No. 1, January 2015

[55] Fibre Channel Backbone – 5 (FC-BB-5), INCITS working draft proposed American National Standard for Information Technology, Rev. 2.0, June 4, 2009

[56] Shudayfat Eman Ahmad, Moldoveanu Alin, Moldoveanu Florica, Gradinaru Alexandru. Virtual Reality-based Biology Learning Module, 9th International Conference eLearning and Software for Education. Carol I Natl Defence Univ Publishing House, vol. 2, ISSN 2066-026X, pp. 621 --626, April 2013

[57] Venu Vasudevan, Paul Pazandak, Semantic File System Survey, http://www.objs.com/survey/OFSExt.htm, 1996

[58] Margo Seltzer, Nicholas Murphy, “Hierarchical File Systems are Dead”, USENIX HOTOS, May 2009

[59] Ip.com, IPCOM000230977D, A Method and System for Storage Server Selection based on a Network Congestion Status, September 20, 2013

[60] SungHo Chin, JongHyuk Lee, HwaMin Lee, DaeWon Lee, HeonChang Yu, Pillwoo Lee, “OGSA-GFS : A OGSA based Grid File System”, Proceedings of the First International Conference on Semantics, Knowledge, and Grid (SKG 2005), 2006

OGSA-DAI (Open Grid Services Architecture-Data Access and IntegrationHung Ba Ngo, Christian Bac, Frederique Silber-Chaussumier, Thang Quyet Le, “Towards Ontology-based Semantic File Systems”, 2007 IEEE International Conference on Research, Innovation and Vision for the Future, pp. 8-13