University POLITEHNICA of Bucharest
Faculty of Automatic Control and Computer Science

REZUMAT TEZĂ DE DOCTORAT

Optimizări și soluții de stocare distribuită

Distributed Storage Solutions and Optimizations

Autor: Ing. Sorin-Andrei Pîștirică

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

CUPRINS

1 INTRODUCERE

In timpul ultimilor zece ani volumul de trafic pe internet a crescut cu mai mult de 300 de ori, iar presiunea pe tehnologiile utilizate pentru infrastructura retelei este considerabila. In afara de internet exista sisteme, precum cloud-urile si centrele de date, unde sistemele de stocare au influentat evolutia infrastructurilor de retea in vederea scaderii costurilor fara a afecta performanta.

Considerand motivele de mai sus, am studiat caracteristicile sistemelor de stocare si am propus o infrastructura pentru sisteme eLearning. Si din moment ce sistemele de stocare sunt sisteme intensive I/E, am propus o solutie pentru cresterea performantei bazata pe motoare hardware dedicate pentru procesare de pachete. Apoi, luand in considerare trendul retelelor convergente aparute din nevoia simplificarii administrarii si a reducerii costurilor, am propus cateva imbunatatiri ale protocolului Quantized Congestion Notification si un algoritm pentru balansarea dinamica a fluxurilor de date: QCN – Weighted Flow Queue Ranking.

In contextul sistemelor distribuite de stocare, metodele de scalare bazate pe nivelele sistemelor de fisiere sunt foarte importante. Fiecare nivel necesita un tip diferit de retea, astfel generand tipuri diferite de sisteme de stocare distribuite. Este de asemenea important sa se observe cum aceste sisteme sunt caracterizate si care sunt deficientele principale ale implementarilor existente. Prin urmare, am propus o clasificare simpla, dar cuprinzatoare bazata pe patru caracteristici principale: localizare, partajarea, nivelul distributiei și semantica de acces concurent. In ceea ce priveste organizarea datelor la nivel scazut exista mai multe moduri, dar toate urmeaza aceleasi elemente constitutive, astfel am propus o schema generalizata potrivita care nu depinde de caracteristica distributiei. In cadrul schemei propuse, datele sunt distribuite in unitati diferite (de exemplu: fisiere, obiecte, blocuri sau segmente de date) analizate in continuare. Am studiat cinci arhitecturi diferite de sisteme de stocare distribuite (Andrew File System, Google File System, General Parallel File System, Lustre si Ceph) cu un accent mai mare pe particularitatile Ceph-ului, utilizat ulterior ca studiu de caz si in masuratori de performanta pentru optimizarile propuse. Am studiat trasaturile retelelor si folosind teoria grafurilor, am prezentat mai multe caracteristici cheie ce influenteaza proprietati, precum rata de transfer, costurile de capital, toleranta la caderi sau disponibilitatea sistemului, urmate de studii ale stivelor de protocoale folosite pentru a identifica diferitele alternative folosite in retelele convergente.

De asemenea, am propus o serie de imbunatatiri pentru a rezolva probleme legate de supraincarcarea sistemului din cauza prelucrarilor fluxurilor de date la rate multi-gigabit. Imbunatatirile sunt bazate pe sisteme hardware integrate de procesare de pachete dedicate cu nuclee de uz general. Pe langa problemele de supraincarcare, o alta provocare este sporirea performantei dispozitivelor de retea pentru a utiliza in mod optim banda retelei. Prin urmare, aceste probleme pot fi depasite prin combinarea procesoarelor multi-nucleu cu dispozitive de retea cu capabilitati multi-functie. Astfel, in primul rand am propus un procesor generic dedicat procesarii de pachete subliniind componentele sale hardware si etapele de prelucrare ale pachetelor. De asemenea, am propus doua alternative pentru coexistenta diferitelor aplicatii cu cerinte diferite de trafic: Per Core Cluster Node si SMP Cluster Node. Folosind sistemul Ceph ca studiu de caz, am propus doua optimizari diferite, si anume: metode de accelerare a fiecarui nod folosind modelele de mai sus si o metoda de scadere a letentei fluxurilor sensibile bazata pe prioritizarea cozilor (A-RADOS). Solutiile propuse sunt sustinute de mai multe masuratori bazate pe platforma P2041 QorIQ™.

In contextul retelelor convergente, protocoalele de tip I/E (cum ar fi SCSI) nu au mecanisme de control al congestiei si astfel necesita medii de transmisie fara pierderi, de exemplu Fibre Channel –

protocol cu viteze mari de transmisie, latente mici si fara pierderi. Ethernet-ul este un sistem de tip best-effort, care impreuna cu protocolul IP ofera o retea de tip end-to-end pentru protocoale orientate conexiune, cum ar fi protocolul TCP. In lipsa acestor protocoale, Ethernet-ul a fost imbogatit cu o serie de protocoale suport pentru a-l transforma intr-un mediu fara pierderi, dupa cum urmeaza: Priority Flow Control (PFC), Enhanced Transmission Selection (ETS), Data Center Bridging Capabilities exchange (DCBx) si Quantized Congestion Notification (QCN). In afara de scopul principal al acestor protocoale suport, mai exista si alte cazuri in care pot fi folosite, precum “TCP Incast”.

In principiu, QCN-ul ofera informatie de congestie surselor pentru a evita perderile de pachete, dar nu rezolva distributia echitabila a segmentelor de congestie din sistem. Astfel, am propus algoritmul **Quantized Congestion Notification – Weighted Flow Queue Ranking** (QCN-WFQR) bazat pe QCN, pentru balansarea dinamica a incarcarii retelei. Algoritmul poate fi folosit in retele traditionale dar si in retelele de tip SDN. In cateva cuvinte, QCN-WFQR calculeaza o serie de indicative de congestie ce pot fi folosite cooperativ (de catre toate entitatile sistemului) sau automat (de catre profiler sau controlerl SDN) pentru a creste performanta sistemului. Pentru exemplificare, am propus doua metode diferite: o metoda de alegere a replicilor intr-un sistem de fisiere distribuit si paralel si o metoda de distributie a traficului in vederea balansarii incarcarii retelei.

1.1 Lucrari stiintifice in care au fost publicate contributiile cuprinse in aceasta teza

Patente

[Apg. #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

Articole

[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, pg. 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, pg. 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, pg. 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, pg. 197-205, **ISI Indexed**

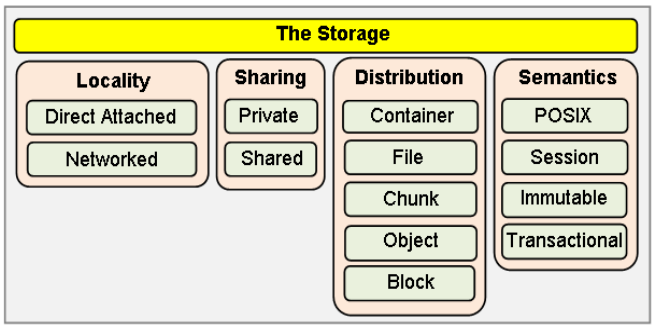
2 SISTEME DE STOCARE

In general, **sistemele de stocare** sunt formate dintr-un set de dispozitive si o colectie de module software pentru administrarea unitatilor de date (blocuri, obiecte sau fisiere). In cazul sistemelor de stocare distribuite, atat dispozitivele cat si modulele software sunt distribuite pe o retea de calculatoare.

2.1 Clasificarea sistemelor de stocare

Clasificarile sunt foarte importante, pentru ca evidentiaza atat caracteristicile intrinseci ale sistemelor de stocare precum si deficientele implementarilor existente. Astfel, propun o clasificare bazata pe patru caracteristici principale **[PIST, 2014/1]**, menita sa cuprinda orice sistem de stocare indiferent de gradul de distributie:

- **Locatie**;
- **Capabilitati de partajare**;
- **Nivel de distributie**: unitatile de date folosite pentru distributie;
- **Semantica**: semantica accesului concurent.



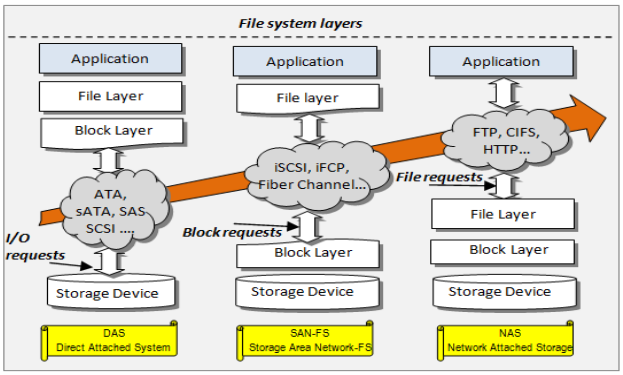
2.2 Cerinte pentru sistemele de stocare distribuite

La inceputul anilor 70, sistemele de stocare distribuite erau folosite doar pentru a partaja date intre noduri de retea. Apoi, cu aparitia *cloud*-ului, a domenilor HPC sau a aplicatiilor de internet precum Google sau Facebook, nevoia spatiului de stocare a crescut exponential. Astfel, propun o lista de cerinte principale pentru sistemele de stocare distribuite: **partajare de date, scalabilitatea spatiului de stocare, elasticitatea spatiului de stocare, transparenta (acceselor, localizarilor, esecurilor, replicilor, migratiilor), valabilitate mare a datelor, toleranta la caderi, recuperarea sistemului, distributia balansata a datelor, balansarea incarcarii, migrarea datelor, accesul concurent si consistet al datelor, arhivarea datelor, performanta infrastructurii de retea si securitatea datelor**.

2.3 Metode de scalare a sistemelor de stocare distribuite

Pentru a preintampina nevoia tot mai mare a spatiului de stocare (*petascale* sau chiar *exascale*), in prezent numarul dispozitivelor de stocare a devenit foarte mare (*scale-out*) **[PIST, 2014/1]**. Acesta scalare se poate obtine prin decuplarea sistemului la diferite nivelu stfel obtinandu se diferite sisteme de stocare distribuite:

- Nivelul hardware (dispozitive de stocare);
- Nivelul bloc sau obiect (secvente de biti inregistrate pe dispozitivele de stocare);
- Nivelul fisier (tipuri de date administrate de catre sistemul de operare);
- Nivelul de aplicatie.



Prin decuplarea sistemului la diferite niveluri si introducerea retelelor de calculatoare, se obtin diferite sisteme de stocare distribuite, dupa cum urmeaza:

- **Sisteme de tip SAN**: decuplarea sistemului la nivelul blocurilor de date (ex. SCSI sau SATA peste Fibre Channel);
- **Sisteme de tip NAS**: decuplarea sistemului la nivelul fisier (ex. CIFS or NFS);
- **Sisteme hibrid**: construite prin combinarea sistemelor de tip SAN si NAS;
- **Sisteme de fisiere distribuite si paralele,** construite prin abstractizarea dispozitivelor de stocare.

Sisteme mass-storage

Sistemele *mass-storage* sunt construite folosind Redundant Arrays of Independent Disks (**RAID).** Ideea de baza este imbunatatirea performantelor prin accesarea unui set de discuri in paralel si pentru a imbunatatii nivelul de disponibiltate si rezistenta la caderi prin duplicarea datelor. EMC[1] introduce notiunile de **Storage Pools** si **RAID based Storage Pools** – o colectie de discuri logice sau fizice sau RAID-uri administrate impreuna. Administrarea spatiului la o granularitate mai mica se obtine prin virtualizarea *storage pool*-urilor in virtual LUNs sau Virtual Volumes, unde maparea dintre Virtual LUNs si RAID LUNs se face de catre un controler.

Din punct de vedere arhitectural, sunt trei abordari diferite: bazata pe gazda (e.g. *Linux Logical Volume Management*: LVM) [4], bazata pe dispozitiv (e.g. disk arrays: RAIDS) si bazata pe retea (e.g. SAN) [5] [6].

2.4 Semantica partajarii

Metoda de accesare concurenta a datelor (semantica de partajare) este foarte importanta, pentru ca stabileste conditiile in care datele sunt accesate in mod consistent. Andrew S. Tanenbaum in cartea sa “Distributed Operating Systems” [7] propune o taxonomie simpla de patru tipuri distincte, astfel: semantica POSIX, semantica sesiune, fisiere imuabile si semantica tranzactionala. Merita mentionat ca in cazul sistemelor distribuite semantica de tip POSIX este cel mai greu de obtinut, pentru ca necesita mecanisme de sincronizare complicate.

2.5 Organizarea datelor la nivel scazut

2.5.1 Organizarea datelor la nivelul dispozitivelor de stocare

In contextul organizarii datelor la nivel scazut, consider trei tipuri principale de organizare a datelor: structurat, jurnal şi arbore. Principala diferenţă între organizarea structurata şi organizarea jurnal este ca prima foloseste zone distincte pentru stocarea blocurilor de date şi inode-urilor, in timp ce cea de-a doua le stocheaza in forma continua. Organizarea de tip arbore (ex. B-arbore: BTRFS) foloseste metoda copy-on-write (COW). Oracle a lansat o astfel de orgnizare in 2007, fiind introdusa in Linux in 2009: Binary Tree File System [1]. Merita mentionat faptul ca sistemele de tip BTRFS confera imbunatatiri de performanta comparativ cu sistemele de tip Unix File System.

2.5.2 Organizarea datelor in sistemele de stocare distribuite

In general, in orice sistem distribuit de stocare, datele sunt impartite in diferite tipuri de segmente distribuite pe nodurile unei retele de calculatoare folsind diferite tehnici. La fel ca in sistemele centralizate si in sistemele distribuite metoda de organizarea a datelor este specificata intr-o structura de tip *metadata*, distribuita ca orice alt fisier, fiind distribuita intr-un cluster specializat, fie stocata centralizat.

[1] EMC: corporatie IT ce ofera solutii de stocare de date.

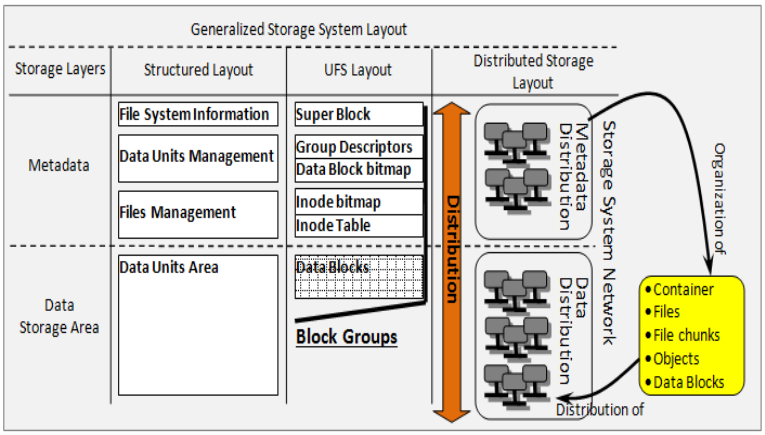
De obicei in sistemele de stocare distribuite nu exista conceptul de partitie, dar unele implementari contin mecanisme similare, de exemplu in cazul sistemului AFS volumele pot fi considerate concepte similare sau sistemul Lustre poate administra multiple sisteme de fisiere administrate de servere MGS ce pot fi de asemenea considerate partitii.

SAN si NAS sunt unele dintre cele mai cunoscute sisteme de stocare pe retele de calculatoare. Din punct de vedere al arhitecturii, sistemele de fisiere de tip SAN folosesc disc-uri organizate in grupuri RAID, iar sistemele de fisiere de tip NAS sunt folosite de obicei impreuna cu sisteme SAN, unde serverele NAS stocheaza datele in medii de tip SAN (i.e. sisteme hibride).

2.5.3 Generalizarea organizarii datelor

Fara a tine cont de semantica datelor sau de tipul de distributie, orice sistem de fisiere este compus din doua niveluri principale:

- **Nivelul de metadate**: specifica organizarea datelor;
- **Nievelul de stocare al datelor**: un spatiu de adresare liniar pentru stocarea datelor



2.5.4 Tendinte catre sisteme de stocare bazate pe obiecte

Tendinta actuala a folosi segmente de fisiere in detrimentul intregului fisier pentru distributia datelor; astfel se obtine un grad mai mare de disponibilitate a datelor si o mai buna toleranta la caderi.

Sistemele de stocare bazate pe obiecte castiga in popularitate, pentru ca grupeaza avantajele DAS-urilor, SAN-urilor si NAS-urilor intr-un sigur dispozitiv [12], idee sustinuta de multi cercetatori. In principiu, OSD-urile adreseaza probleme legate de **securitate**, **scalabilitate** and **administrare**. In primul rand, securitatea este imbunatatita prin adaugarea de drepturi per operatie, in contrast cu dispozitivele bloc ce asigura securitate la nivel LUN. In al doilea rand, scalabilitatea si administrarea este imbunatatita prin mutarea lor de la nivelul serverelor de metadate la nivelul controlerelor dispozitivelor. Iar in al treilea rand, listele mari de blocuri sunt inlocuite cu liste restranse de obiecte, iar problema alocarii/eliberarii de blocuri este distribuita.

Desi OSD-urile au numeroase avantaje, nu rezolva problema fundamentala a distributiei datelor pe un numar foarte mare de dispozitive.

2.6 Arhitectura sistemelor de stocare distribuite

Din multitudinea de sisteme de stocare distribuite cu arhitecturi diferite, am ales sa studiez cinci sisteme reprezentative pentru a acoperi un set important de aspecte, cum ar fi organizarea datelor la nivel scazut, arhitectura sistemelor sau semantica partajarii: **AFS** (Andrew File System) [13]

[14], **GPFS** (General Parallel File System) [17], **GFS** (Google File System) [15] (si versiunea open-source **Hadoop** (HDFS) [16]), **Lustre** [18] si **Ceph** [19]**, [PIST, 2014/1]**.

Sistemul Ceph a fost studiat in amanunt, fiind utilizat ca studiu de caz in optimizarile propuse in sectiunea 4.

2.6.1 Ceph File System

Ceph [19], este un sistem de fisiere nou ce incorporeaza multe avantaje de la implementari anterioare. Distribuie totul: spatiile de nume, monitorizarea, securitatea si datele. Astfel, este foarte scalabil, dar si foarte complex fiind considerat "*the new dream distributed file system*". Unul dintre caracteristicile importante este faptul ca izoleaza complet administrarea metadatelor de distributia datelor (RADOS – Reliable, Autonomic Distributed Object Store) [20]. Pe scurt, Ceph imparte fisierele in obiecte folosind o metoda similara cu metoda RAID-urilor, apoi RAODS-ul distribuie aceaste obiecte pe un cluster de OSD-uri folosind o functie hash (CRUSH – Controlled Replication Under Scalable Hashing) [21].

RADOS este un sistem compus dintr-un cluster foarte mare de OSD-uri (Object Storage Device) si un cluster cu numar redus de noduri de monitoare pentru supervizare. Pe scurt, obiectele sunt mapate peste grupuri de plasament (folosite pentru controlul replicarii). Schema OSD-urilor este descrisa de o harta a cluster-ului. Prin aplicarea de politici de plasament pe aceasta harta, pot fi configurate diferite domenii de esec. Mai departe, o lista determinista de OSD-uri se obtine prin folosirea algoritmului CRUSH bazat pe grupurile si regulile de plasament. Lista de mesaje principale in cadrul sistemului RADOS (folosite in studiul de caz din sectiunea 4.4) sunt listate in tabelul de mai jos:

#	Mesaje RADOS	Descriere
1	**Algerea monitorului lider**	• Ofera o harta consistenta tuturor partilor sistemului • Lider-ul este ales folosind un algoritm simplificat PAXOS si un cvorum pentru serializarea actualizarilor hartii, [22] pg. 127.
2	**Chirii**	• Acorda chirii pe termen scurt monitoarelor active pentru a avea dreptul sa distribuie copii ale hartii tuturor partilor sistemului, [22] pg. 127.
3	**Distributia hartii**	• Raspunsurile monitoarelor active la cerintele hartii de catre MSD-uri, clienti sau noile OSD-uri. Din moment ce un cluster de OSD-uri poate avea un numar foarte mare de dispozitive [22] pg. 129, monitoarele nu difuzeaza actualizarile, ci actualizarile sunt schimbate intre OSD-urile din acelas grup de plasament.
4	**Mesaje de tip heartbeat**	• Pentru a preveni accesarea de date inconsistente, mesaje de tip heartbeat sunt interschimbate intre OSD-uri [22] pg. 129.
5	**Trafic I/E**	• Scrierea/Citirea obiectelor si replicarea datelor (i.e. primary-copy, chain si splay) [22] pg. 131.
6	**Recuperare**	• Recuperarea dupa caderi este bazata pe epoca hartii si pe setul de OSD-uri active din fiecare grup de plasament. Se foloseste un algoritm de cercetare pentru a asigura o imagine consistenta a tuturor OSD-urilor din grupul de plasament, [22] pg. 137.

Administrarea spatiilor de nume este distribuita pe un cluster de metadate, ce foloseste o tehnica denumita *adaptive workload distribution* (i.e. partitionarea activa a sub-arborilor pentru a obtine o performanta scalabila). Ceph migreaza sau replica bucati ale arborelui de fisiere la nivel de

fragmente de directoare folosind o metrica bazata pe popularitate, mecanism ce are un cost ridicat daca numarul de noduri este foarte mare.

2.6.1.1 Distribuirea datelor descentralizat

Scopul principal al distributiei descentralizate a datelor este de a inlocui serverele de tip lookup. In aceasta sectiune am prezentat 2 algoritmi: Replication Under Scalable Hashing (**RUSH**) [23] si Controlled Replication Under Scalable Hashing (**CRUSH)** [21].

Algoritmul RUSH are cateva versiuni pentru plasamentul datelor, cum ar fi: folosirea numerelor prime, support pentru eliminari precum si o abordare folosind arbori [24]. Are doua principii de baza: identificarea obiectelor ce trebuie migrate pentru a obtine un cluster balansat si decizia destinatiilor bazat pe functii hash. De asemenea, dispozitivele sunt inlocuite in grupuri (ex. per sertar).

Algoritmul CRUSH [21] introduce ideea de distributie controlata per domenii de esec prin folosirea politicilor de plasament. Pe scurt, produce o lista ordonata si determinista de dispozitive de stocare pentru un obiect bazata pe o functie pseudo-random, harta clusterului de OSD-uri si politicile de plasament.

2.6.2 Comparatii

- Localizarea datelor in spatiul de distributie poate fi facuta in doua moduri: prin folosirea hartilor (similar cu hartile de inoduri) sau prin algoritmi decentralizati cum ar fi RUSH sau CRUSH. Folosirea hartilor de date impune o presiune mare pe serverele de lookup, fiind considerat un dezavantaj in comparatie cu folosirea algoritmilor descentraliati.
- Distributia administrarii metadatelor are si avantaje si dezavantaje: in esenta, administrarea distribuita vine cu o performanta de I/E mai buna, dar cu o arhitectura mai complexa. De exemplu, Ceph si GPFS distribuie administrarea metadatelor, pe cand GFS/HDFS foloses doar servere *umbra* pentru rezistenta la caderi.
- Nivelul de distributie se refera la unitatile de date folosite in distributie, cum ar fi: segmente de date, fisiere sau obiecte. Exista o tendinta in a folosi OSD-uri datorita avantajelor lor, precum: securitatea datelor per obiect, dimensionare flexibila, API standard si administrare locala a spatiului. Astfel, Lustre si Ceph folosesc OSD-uri si exista o tendinta in a folosi OSD-uri in AFS (OpenAFS) si GPFS.
- Sistemul GPFS este sensibil la caderi, deoarece foloseste RAID-uri ne-clusterizate. De asemenea exista un trend in a folosi obiecte si RAID-uri [27] – PanFS.
- Exista un trend in sistemele de stocare distribuite pentru a folosi dispozitive ieftine, unde caderile sunt comune. Pentru rezolvarea acestei probleme sistemele implementeaza diferite mecanisme pentru a asigura un anumit grad de rezistenta la caderi.
- Interfata de acces este foarte importanta in sistemele distribuite, astfel ele pot fi impartite in doua categorii principale: sisteme ce suporta interfete conventionale si sisteme ce au interfete personalizate. Interfetele conventionale au avantajul transparentei, pe cand cele personalizate pot oferi metode de ajustare a diferitelor caracterisiti ale sistemului. Dar, de exemplu Ceph amesteca cele 2 tipuri de interfete, imbogatind astfel interfata conventionala cu cateva metode de configurare.
- Semantica accesului concurent la date este implementata astfel:
  - AFS: semantica la nivel de sesiune;
  - GPFS, Lustre si Ceph: semantica POSIX;

- GFS/HDFS: semantica POSIX aproximata.

2.7 Tehnologii de retea folosite in sisteme de stocare distribuite

Infrastructura de retea are o mare influenta in performanta sistemelor distribuite de stocare **[PIST, 2014/1]** si poate fi impartita in 2 subiecte mari: topologii si stive de protocoale.

2.7.1 Topologii

Topologiile din sistemele de stocare distribuite sunt definite prin grafuri, folosite pentru a crea cai intre dispozitive de stocare si porturile de la marginile retelei (OSD – portaluri/gateway) [28]. Exista similitudini intrinseci intre grafurile folosite in conectarea nucleelor din sistemele multi-nucleu si conectarea elementelor sistemelor distribuite, din moment ce urmaresc aproximativ aceleasi caracteristici: astfel, o topologie perfecta trebuie sa aiba cea mai mica latenta posibila, transfer maxim, costuri minime si o rezistenta la caderi perfecta [29]. In centrele de date (si sistemele de stocare distribuite) diametrul grafurilor influenteaza latenta si accesul concurent (caracterisitici relevante pentru sistemele SAN). De asemenea, un grad mic al nodurilor micsoreaza costurile echipamentelor, dar creste latenta din moment ce topolgia va avea mai multe *hop*-uri, un diametru mic micsoreaza latenta si timpul de raspuns si creste rata de transfer, o bisectoare mica faciliteaza traficul est-vest (important pentru migrarea si distributia datelor) iar un numar mare de arce influenteaza implementarea.

Analiza topologiilor:

- Grafurile complete au un diametru perfect (de valoare 1), hiper-cuburile, cuburile-ciclu-conectate, arborii-binari-grasi si hiper-arborii au un diametru ce creste logaritmic cu numarul de noduri (trebuie mentionat ca hiper-arborii au avantajul ca folosesc arce de viteza constanta la toate nivelurile, pe cand arborii-grasi trebuie sa ofere conexiuni mai grase (de viteze mai mari) pe masura ce se apropie de radacina).
- Gradul nodului este constant pentru torus, arbori-binari-grasi si hiper-arbori. Hiper-arborii au abilitatea de a scala orizontal prin schimbarea dimensiunii k-ary. Iar, gradul nodurilor pentru hiper-cuburi creste logaritmic (mai repede).
- In afara de arborii-grasi, toate celelalte topologii ofera rute alternative, astfel ofera o toleranta la caderi. Hiper-arborii pastraza constant diametrul in cazul caderilor nodurilor, topologiile de tip cub au o toleranta la caderi mai buna pentru ca ofera rute alternative intre oricare 2 noduri.

In proiectarea topologiilor se tine cont de cerintele de trafic, rutele traficului intens, latenta si rata si evident balanta intre performanta si cost.

2.7.2 Stive de protocoale

In ceea ce priveste stivele folosite, sistemele de stocare distribuite se impart in 2 categorii: cele care necesita medii de transmisie fara pierderi si cele care pot functiona folosind si medii de transmisie de tip *best effort*. Fiabilitatea se poate implementa la nivelul 2 (legaturilor de date) sau la nivelul 4 (transport) – OSI model.

Ethernet-ul este cel mai folosit protocol la nivelul legaturii de date, dar este un protocol de tip best effort pe cand, de exemplu, Fibre Channel-ul este un protocol ce ofera si un mediu fara pierderi. Fibre Channel-ul este folosit ca suport pentru protocolul FCP, dar exista solutii si pentru Ethernet. In cazul Ethernet-ului au fost implementate o serie de protocoale suport, cum ar fi iFCP, iSCSI sau FCIP.

Pentru ca Ethernet-ul sa fie folosit ca suport pentru protocoale gen FCP, a fost imbogatit cu o serie de protocoale suport numite DCB (Data Center Bridging) de catre IEEE group [42] pentru controlul traficului si pentru evitarea de pierderi de pachete, astfel devenind o alternativa la FC.

Sistemele distribuite de stocare construite la nivelul fisier nu sunt atat de pretentioase si de obicei functioneaza perfect peste stiva clasica TCP/IP/Ethernet peste care ruleaza protocoale percum: NFS, CIFS sau SMB.

3 STUDIU DE CAZ: INFRASTRUCTURA MEDIULUI ELEARNING

3.1 Motivare

Invatamantul de nivel inalt este, de cele mai multe ori, costisitor. Platformele de eLearning ofera o alternativa mai ieftina pentru educatie, facand o investitie rentabila. In ultimul deceniu cercetarea si adoptarea metodelor de cloud computing a crescut simtitor datorita multiplelor avantaje, incluzand beneficiile economice, usurinta de utilizare, economisirea de energie si altele. In esenta cloud computing-ul ofera metodele de organizare si livrare a unei largi game de servicii software, incluzand invatamantul electronic. In acelasi timp si solutiile de sisteme de fisiere au fost imbunatatite pentru a satisface cerintele impuse de caracteristicile distribuite ale cloud-ului. Nivelul de distribuire al sistemelor de fisiere, managementul de date, metodele de cautare a datelor si multe alte caracteristici ale sistemelor de fisiere distribuite influenteaza performanta sistemelor de eLearning. De asemenea, cloud-ul este construit peste topologiile retelelor de calculatoare. In continuare propun o arhitectura de nivel inalt de platforme cloud, potrivita pentru software-ul de eLearning, subliniind cateva avantaje care inving problemele legate de aceste sisteme distribuite, folosing Ceph ca mediu de stocare a datelor si cateva topologii de retea **[PIST, 2014/2]**.

De ce clouds si eLearning?

Mediile de eLearning au cateva probleme, unele dintre ele fiind mai usor de rezolvat in sistemele de cloud, una dintre principalele probleme fiind infrastructura. De obicei infrastructura pentru invatamantul electronic presupune investitii uriase, asa ca cloud-ul fiind deja considerat prin definitie un provider de infrastructura, poate fi folosit ca mediu de baza pentru aplicatiile de eLearning. Cloud-ul se poate redimensiona in mod dinamic daca este nevoie si in acelasi timp ofera un mediu colaborativ [56] – o caracteristica importanta pentru serviciile de eLearning. Mediile de eLearning au baze de date imense cu obiecte de invatare care pot fi stocate in sistemul de stocare al cloud-ului, deci modul in care datele sunt manevrate in cloud influenteaza diverse aspecte ale mecanismelor de invatamanat electronic, incluzand cautarea si livrarea continutului obiectelor de invatare. Exista o mare varietate de implementari ale sistemelor de stocare ale cloud-ului, cuprinzand mai multe avantaje. Pe langa sistemul de stocare, si modul in care totul este interconectat influenteaza performantele de livrare ale continutului din mediile de eLearning. Exista mai multe topologii de retele potrivite pentru cloud, cum ar fi: *arbore-gras, hiper-arbore, cub si hiper-cub*. Am luat in considerare si cateva caracteristici care influenteaza performantele de intrare/iesire, cum ar fi scalabilitatea, latenta si costul.

3.2 Sisteme de stocare si retele adaptate pentru medii eLearning

Cloud-ul isi propune sa satisfaca un set de cerinte pentru sistemele de stocare, cum ar fi: posibilitatea de a folosi aceleasi resurse de catre mai multe surse, redimensionare a capacitatii resurselor in functie de nevoi, transparenta, disponibilitatea resurselor oricand e nevoie de ele, toleranta la caderi, acces la date in mod sigur si in acelasi timp de catre mai multe surse, securitatea (sectiunea 2.2). Design-ul sistemelor de stocare poate influenta modalitatea de livrare, mentenanta si managementul obiectelor de invatat, in timp ce arhitectura retelei influenteaza performanta sistemului de intrare/iesire.

3.2.1 Distributia datelor pentru medii eLearning

Mediile eLearning au de obicei baze de date foarte mari, continand obiecte de invatare, organizate pe categorii. Din acesta cauza o organizare ierarhica nu va ajuta, o alternativa putand fi organizarea semantica. Unii cercetatori sustin ca sistemele de fisiere ierarhice vor fi inlocuite de

sisteme de fisiere semantice [57], deoarece pe langa problema de organizare ele imbunatatesc si metodele de cautare [58].

In cadrul tezei propun o arhitectura de sistem de fisire paralela si distribuita pentru un sistem de stocare al unui cloud, bazata pe doua nivele: primul nivel stocheaza obiecte de invatare intr-un spatiu de adrese liniar (poate fi folosit de exemplu RADOS) iar al doilea nivel stocheaza metadate organizate in asa fel incat sa faciliteze cautari semantice.

O alta caracteristica a sistemului de eLearning este ca cei care invata, utilizatorii sistemului, sunt imprastiati geographic si folosesc diverse cai de retea pentru a accesa sistemul de stocare al cloud-ului. Incarcarea traficului trebuie balansata, si pentru aceasta sistemul trebuie sa tina cont de profilele de distributie ale utilizatorilor si sa imparta cluster-ul de date in domenii de distributie, unde fiecare domeniu este accesat de catre un anume gateway. Distributia obiectelor trebuie sa fie reprodusa pentru fiecare domeniu astfel performanta de acces a datelor poate fi imbunatatita.

Exista o solutie alternativa bazata pe puncte de delegare, acestea fiind puncte de stocare a obiectelor de invatare care apoi sunt distribuite unui grup de utilizatori. Sunt si cateva dezavantaje pentru aceasta metoda, incluzand: delegarea de securitate, stocare locala pentru fiecare obiect de invatare, metoda de inchiriere a obiectelor, si altele. Utilizatorii pot accesa sistemul de eLearning din casa lor, folosind doar o pagina web, astfel incat, chiar daca furnizorul de internet ar avea o astfel de metoda, nu ar fi nici un castig vizibil, utilizatorul ar vedea in continuare o latenta mare. Consecinta acestei propuneri este ca datele trebuie sa fie reproduse in mai multe dispozitive de stocare - poate fi considerata o idee buna, deoarece pretul pe GB este de obicei scazut.

S-ar putea aduce o imbunatatire daca s-ar utiliza legaturile de retea la capacitate optima si s-ar echilibra incarcarea pe gateway-uri, daca echipamentul de retea are functia de notificare cuantificata a congestiei. Exista o metoda reala care echilibreaza incarcarea traficului intre gateway-uri de retea, care utilizeaza un *arbore rosu-negru* pentru a sorta portalurile disponibile bazate pe distanta congestionarii [59] sau QCN-WFQR (capitolul 6).

3.2.2 Reteaua cloud-ului

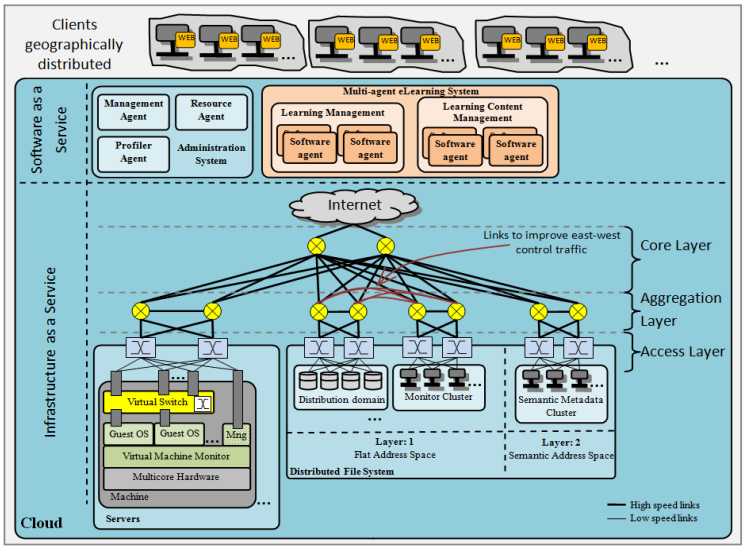
Un cloud se poate intinde peste unul sau mai multe centre de date (publice sau private). Centrele de date actuale au o retea cu trei straturi de arhitectura: **de baza, de agregare si de acces**. Principalul dezavantaj este ca aceasta arhitectura limiteaza topologii permise celor care au o organizare cu mai multe nivele.

In teza propun o solutie bazata in principiu pe topologia *arbore-gras* cu redundanta. Practic, acesta ofera subscriere 1:1 dar este limitata in inaltime, datorita necesitatii de a oferi o capacitate uplink egala cu suma tuturor downlinks. Datorita acestui dezavantaj, aceasta topologie nu este utilizata intr-o forma pura; de obicei pe parcurs creste suprasubscrierea. De asemenea, aceasta topologie este mai potrivita atunci cand cea mai mare parte a traficului este nord-sud, dar nu este recomandata pentru utilizare la configuratii in care traficul est-vest predomina datorita faptului ca nodurile la distanta nu pot comunica direct. Intr-un mediu de eLearning de cele mai multe ori transferul de date este sud-nord, prin urmare aceasta solutie este o alegere buna.

3.3 Cloud-ul si mediul de eLearning: arhitectura propusa

Propun o arhitectura de cloud subliniind diferite aspecte legate de sistemul de stocare si topologie de retea care pot influenta mediul de eLearning, din punct de vedere al arhitecturii si performantelor de intrare/iesire

Arhitectura este compusa din trei componente principale de infrastructura: sistemul de stocare, serverele si reteaua si doua componente software principale: sistem de administrare si sistemul de agenti de eLearning. Sistemul de administrare gestioneaza resurse de pe server (creeaza, migreaza si distruge masinile virtuale si switch-uri virtuale) si agentii de eLearning creeaza, migreaza si distrug agentii pe baza profilurilor statistice



3.3.1 Sistemul de stocare

In ceea ce priveste stocarea datelor, propun o alternativa la OGSA-GFS [60]. Un sistem cu doua niveluri de stocare bazat pe componentele Ceph: un spatiu de adrese liniar (RADOS) si metadate semantice. Exista mai multe metode de a pune in aplicare o astfel de organizare a fisierelor: pe baza de proprietati, pe baza de continut, sau pe baza de context si mai multe directii de arhitectura: integrata, augmentata si independenta (nativa). Eu propun o implementare nativa cu semantica bazata pe continut, pentru ca o organizare ierarhica a fisierelor nu aduce nici un avantaj si arhitectura bazata pe continut clasifica fisierele si imbunatateste cautarea.

Folosind Ceph/CRUSH, propun o distributie de date bazata pe domenii si, prin urmare, definesc o politica de plasare si harta cluster-elor pentru a reproduce fiecare obiect de trei ori in fiecare domeniu pentru disponibilitate ridicata si toleranta la caderi.

3.3.2 Profilul retelei

Traficul care este doar citit si care are o rata de transfer mare este in mare parte sud-nord, de la sistem la utilizatori, asa ca va fi o cantitate mica de date replicate sau reechilibrate (cererile est-vest sunt scazute). Cu toate acestea, monitoarele din RADOS care administreaza cluster-ele OSD prin pastrarea datelor de stocare sincronizate si colecteaza starea fiecarui OSD ar trebui sa aiba latenta scazuta. O topologie ierarhica faciliteaza traficul sud-nord cu rata de transfer mare impreuna cu o noua legatura de la grupul de control spre fiecare OSD, pentru a reduce latenta dintre monitoare si OSD-uri. De asemenea, merita mentionat ca odata cu cresterea sistemului se pot adauga nivele de agregare, asa ca intr-o topologie ierarhica, latenta est-vest va creste.

4 OPTIMIZARI IN SISTEMELE DE STOCARE DISTRIBUITE UTILIZAND MOTOARE HARDWARE

4.1 Motivare

Datorita vitezelor de transmisie foarte mari suportate de Ethernet (40Gbps, 100Gbps sau 800Gbps), capacitatea serverelor de a suporta un volum mare de trafic a fost depasita. Astfel, cresterea vitezelor porturilor de transmisie si cresterea puterii de calcul a unitatii centrale impreuna cu adaptarea multiplelor niveluri de cache au fost primele tehnologii folosite in imbunatatirea ratelor de transfer si micsorarea latentelor la nivelul serverelor. Dar recent, serverele au esuat in sustinerea unui volum si mai mare de trafic datorita mai multor factori, precum convergenta in cadrul centrelor de date sau volumul de trafic in timp real (ex. voce sau video) [31]. O imbunatatire ar putea fi obtinuta prin cresterea performantei ficarui nod prin degrevarea nucleelor CPU si migrarea de sarcini in motoare hardware dedicate.

Investigatiile din cadrul companiei Intel au aratat cateva strangulari la receptionarea pachetelor impartite in trei categorii dupa cum urmeaza: supraincarcarea sistemului, procesare TCP/IP si acesul la memorie [31]. Pentru a preintampina aceste supraincarcari Intel a dezvoltat o tehnologie numita *Accelerated High-Speed Networking* technology (I/OAT) – un set de functionalitati pentru a reduce supraincarcarea la receptionarea pachetelor. Aceasta tehnologie arata imbunatatiri de aproximativ 38% in utilizarea CPU-ului, numarul de tranzactii a fost crescut cu 14% si rata de transfer cu 12%, [32].

Am urmat o idee similara si am crescut performanta fiecarui nod din cadrul unui cluster prin degrevarea nucleelor CPU de sarcinile receptionarii pachetelor prin folosirea clasificarilor si distributiilor fluxurilor de date pe mai multe cozi si prin folosirea tehnologiei Accelerated Receive Flow Steering [33] am scos in evidenta avantajele sistemelor multi-nucleu **[PIST, 2013]**. De asemenea, prin adaugarea de ponderi la cozile de pachete am propus o metoda pentru reducerea latentei fluxurilor de date sensibile, astfel sistemul raspunde mai bine cerintelor aplicatiilor.

4.2 Motoare integrate de procesare de pachete

In general, orice sistem-on-chip (SoC) cuprinde un set de module hardware dedicate impreauna cu nuclee de procesare generala intr-un singur cip, unde motoarele dedicate sunt folosite pentru degrevarea nucleelor de procesare prin migrarea anumitor sarcini; astfel, se elibereaza timp de procesare pentru nuclee.

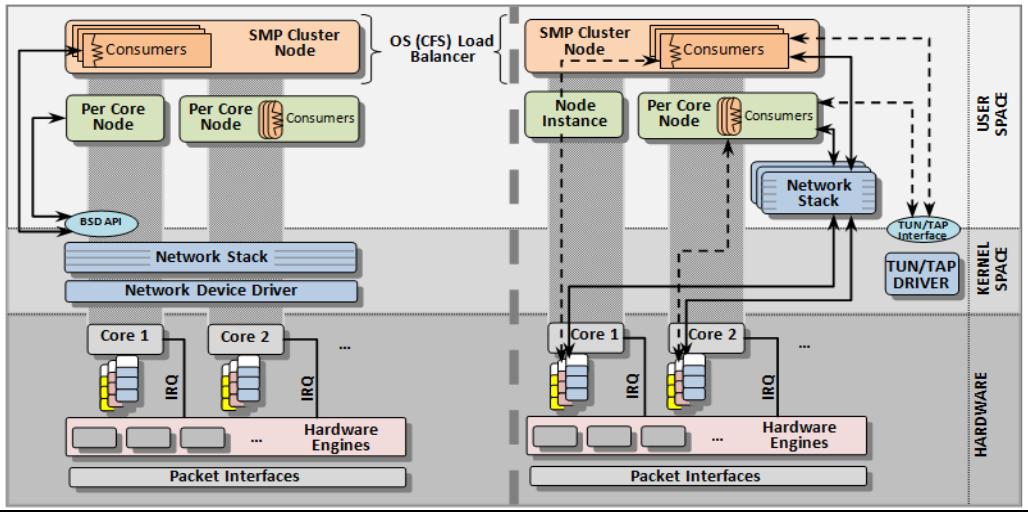
Un sistem de procesare de pachete (SoC) poate avea multiple fluxuri de date ce pot fi programate, astfel orice pachet trimis sau receptionat este distribuit unui anumit flux. Aceste fluxuri pot fi vazute ca si cai intr-un graf unde motoarele hardware sunt nodurile grafului iar cozile de procesare arcele lui.

4.3 Noduri accelerate hardware

Din punctul de vedere al convergentei, de obicei OS-ul este solutia pentru coexistenta mai multor aplicatii diferite pe acelasi sistem hardware. Daca aceste aplicatii sunt OS-uri, atunci virtualizarea este solutia adecvata, iar daca grupuri de aplicatii au nevoie de un anumit tip de izolare atunci solutia adecvata este folosirea containerelor Linux (LXC).

Datorita motivelor de mai sus, propun doua metode alternative mai putin restrictive pentru coexistenta consumatorilor de trafic cu volum mare si consumatori de trafic cu volum mic folosind o metoda de procesare a fluxurilor de date in paralel si metode de accelerare hardware: **Per Core**

Cluster Node si **SMP Cluster Node**, precum doua directii de implementare: in **spatiu nucleu** si in **spatiu utilizator**.



(a). Implementare in spatiu nucleu (b). Implementare in spatiu utilizator

Modelul **Per Core Cluster Node** este caracterizat prin faptul ca fiecare instanta a unei aplicatii este legata de un anumit nucleu, astfel scalabilitatea este limitata la numarul de nuclee. Modularitatea este obtinuta prin divizarea sarcinilor la o granularitate mai mica prin folosirea firelor de executie dedicate fara a fi re-programate pe un nucleu diferit. Fiecare consumator este legat de o anumita coada de procesare sau de un anumit grup de cozi ce ofera doar pachete specifice firelor de executie ale consumatorului. Motoarele hardware trebuie sa fie configurate sa clasifice si sa distribuie pachetele pe cozi conform cu firele de executie ale consumatorilor.

Modelul **SMP Cluster Node** este similar cu primul model, cu diferenta ca sarcinile sunt distribuite pe un set de nuclee folosind capabilitatile motoarelor de a clasifica pachetele. O solutie ar fi ca motoarele hardware sa distribuie incarcarea pe nuclee folosind tehnici gen Round-Robin, sau OS-ul sa distribuie uniform incarcarea pe nuclee, iar motoarele hardware sa tina cont de acest lucru.

Prin configurarea motoarelor hardware in concordanta cu arhitectura software si afinitatea planificatorului de fire de executie, mai multe instante ale acestor modele apartinand diferitelor aplicatii intr-un sistem convergent pot coexistata intr-un sistem cu performanta ridicata. Este evident ca primul model este aplicabil in cazul in care traficul este redus, pe cand cel de-al doilea este aplicabil in cazul traficului cu intensitate mare. Merita mentionat ca cel de-al doilea model poate fi considerat similar cu primul in cazul in care setul de nuclee este redus la 1.

Pentru a folosi paralelismul sistemelor multi-nucleu, propun utilizarea tehnologiei **Accelerated Receive Flow Steering (ARFS)** [33]: motoarele hardware sa distribuie pachetele tinand cont de nucleul unde destinatarul (firul de executie/consumatorul) a fost programat sa ruleze. Aceasta cerinta impune doua lucruri: primul, ficare nucleu posibil trebuie sa contina o coada de unde firul de executie sa poata prelua pachete si al doilea, motoarele hardware trebuie sa aiba capabilitatea sa inspecteze pachete si sa le distribuie corespunzator.

In teza, propun doua implementari pentru aceste modele: una in spatiul nucelu (mai eficient) si una in spatiul utilizator (penalizare de performanta mica, in timp ce asigura o dezvoltare rapida si o politica de licentiere flexibila).

In implementarea in spatiul utilizator exista o problema legata de metoda de acces la stiva TCP/IP. Implicit, Linux-ul foloseste o implementare in spatiul nucleu a stivei, astfel in spatiul utilizator

nu exista suport pentru protocoale orientate pe conexiune. O varianta poate fi utilizarea de stive proprietare in spatiul utilizator [34] sau folosirea de conexiuni de tip tun/tap. O alta problema este metoda de semnalare a receptionarii pachetelor in spatiul utilizator. Ca solutii la acesta problema poate fi folosirea de metode de interogare (polling) sau folosirea de semnale-proces.

Pentru micsorarea latentei traficului sensibil, propun o solutie bazata pe ponderarea cozilor: astfel, cozile ce servesc cele mai sensibile date vor avea ponderi mai mari si vor fi programate primele. Cum se stabileste sensibilitatea fluxurilor de date depinde de cerintele aplicatiei, astfel ca studiu de caz propun o solutie aplicata in sistemul Ceph pentru clasificarea traficului monitoarelor, OSD-urilor si MDS-urilor.

4.4 Studiu de caz: Ceph accelerat hardware folosing QorIQ™

Pentru acest studiu de caz, am folosit QorIQ™ P2041, procesor de pachete integrat pentru scaderea latentei fluxurilor sensibile si pentru cresterea performantei nodurilor din Ceph: OSDs, monitoare si serverele de metadate.

4.4.1 Procesare de pachete de tip QorIQ™

QorIQ combina multi-nuclee de scop general cu un set variat de motoare hardware, oferind o infrastructura flexibila (**Data Path Acceleration Architecture – DPAA**) pentru procesare de volum mare de trafic la rate mari. Printre multiplele motoare din cadrul DPAA, m-am axat pe urmatoare module: Queue Manager (**QMan** [35], pg. 6-1/299), Buffer Manager (**BMan** [35], pg. 7-1/531) si Frame Manager (**FMan** [35], pg. 8-1/594). QMan este modulul folosit pentru a transmite date intre motoarele sistemului, BMan are rolul de a reduce supraincarcarea softului de administrare a memoriei si FMan-ul are trei functii principale (**PCD**): **P**arsarea pachetelor cu verificarea integritatii si identificarea protocoalelor din antetul pachetului, **C**lasificarea pachetelor prin folosirea de chei generate (KeyGen [35], pg. 8-395/1557) bazate pe rezultatele parsarii si **D**istributia pachetelor pe cozi folosind diferite profiluri de distributie.

In afara de protocoalele standard, FMan-ul poate fi configurat sa parseze si sa detecteze pana la **trei** protocoale proprietare sau campuri predefinte din pachete prin folosirea capabilitatii **Software Parser** ([35], pg. 8-391/982). Acesta foloseste NetPDL (limbaj bazat pe XML pentru descrierea pachetelor [37]) pentru definirea campurilor non-standard configurate folosind instrumentul Frame Manager Configuration.

4.4.2 Noduri Ceph accelerate

Prin combinarea motoarelor hardware cu arhitectura software, performanta nodurilor din cluster poate fi imbunatatita, astfel pentru aceeasi performanta este nevoie de un numar mai mic de noduri si deci implica costuri mai mici. Cu aparitia sistemelor convergente, fiecare nod din cluster poate executa diferite roluri in acelasi timp.

In contextul Ceph, nodurile OSD au cel mai mare volum de trafic (in principal I/E), pe cand monitoarele necesita latenta mica pentru a administra caderile OSD-urilor. Astfel, propun **A-OSD** (Accelerated OSD), **A-MDS** (Accelerated MDS) si **A-MON** (Accelerated Monitor). Pentru A-OSD propun folosirea modelului **SMP Cluster Node** pentru a putea sustine volumul ridicat de trafic si la fel pentru A-MDS pentru a putea sustine jurnalele si metadatele stocate in RADOS precum si cerintele de la clienti. A-MON nu necesita porturi cu rate mari sau putere de procesare mare. Astfel, in contextul sistemelor convergente pot coexista cu alte tipuri de aplicatii astfel propun folosirea modelului **Per Core Cluster Node**.

Nodurile accelerate pot fi implementate folosind procesoarele QorIQ impreuna cu capabilitatile FMan-PCD si QMan. Propun pentru A-OSD si A-MDS o distributie uniforma, astfel clientii

sunt distribuiti uniform de-a lungul sistemului. Pentru mesajele sistemului (mesaje de mentinerea consistentei, jurnal si metadate) propun folosirea Software Parser-ului.

4.4.3 RADOS accelerat

RADOS-ul este compus din doua clustere distincte: un cluster foarte mare de OSD-uri si un cluster mic de monitoare folosit pentru administrarea OSD-urilor. Doua caracteristici importante ale RADOS-ului sunt fiabilitatea si disponibilitatea datelor ([22], pg. 119). Pentru a face disponibile datele intr-o maniera consistenta tuturor partilor, acestea trebuie sa fie gestionate in timp util.

Pentru motivele enuntate mai sus, propun un RADOS accelerat (**A-RADOS**) implementat folosind motoare de procesare hardware. Prin clasificarea si prioritizarea mesajelor in RADOS urmaresc scaderea latentei operatiilor sensibile listate in sectiunea 2.6.1 si imbunatatirea reactiei globale a cluster-ului si a disponibilitatii datelor.

In contextul monitoarelor, consider ca cele mai importante mesaje sunt mesajele ce asigura o vedere consistenta a sistemului si sunt sensibile pentru stabilitatea cluster-ului, urmate de distributia hartii, apoi distributia hartii in intregul sistem: alegerea liderului si chirii (clasa I), distributia actualizarilor hartii monitoarelor (clasa II), distributia hartii in intregul sistem (clasa III).

In contextul clusterului de OSD-uri, consider ca mesajele schimbate intre entitati pentru a asigura conistenta sistemului sunt cele mai importante, urmate de traficul I/E si de recuperarea sistemului. Astfel, propun urmatoarea clasificare: mesajele de tip heartbeat si propagarea actualizarilor hartii (clasa I), trafic de tip I/E si replicarea datelor (clasa II) si recuperarea OSD-urilor (clasa III).

Prioritizarea claselor se obtine prin configurarea ponderilor cozilor, unde fiecare coada trateaza o clasa specifica. Astfel considerand clasificarea de mai sus, clasa I va avea cea mai mare pondere si astfel va fi programata prima, pe cand clasa III va fi programata ultima si astfel latenta traficului sensibil va fi micsorata iar timpul de reactie al sistemului este imbunatatit.

Prin folosirea procesoarelor QorIQ, prioritizarea celor 3 clase diferite se obtine prin impartirea cozilor de lucru in 3 grupuri si prin configurarea de ponderi pentru a reflecta clasificarea facuta mai sus.

4.4.4 Rezultate masuratori (micro-benchmark)

Consider ca numarul de clienti serviti pe secunda de un MDS (metadata server) reflecta performanta sa. Pentru testare am masurat cat timp necesita o cerinta rezolvata de MDS (core ticks) si am simulat folosind modelul Per Cluster Node, 4 diferite noduri ce trateaza cerinte in paralel. Folosind acest setup, am facut 3 teste folosind cateva modificari ale nucleului Linux-ului: fara a folosi accelerare hardware si folosind tehnologia Receive Flow Steering (implementare in software) cu si fara afinitatea nuceleelor. Am folosit o masina QorIQ P2041 cu 3 nuclee si motoare hardware pentru inspectarea pachetelor si distributia lor folosind DPAA.

Pentru fiecare test, am contorizat numarul de cerinte tratate pe secunda (tranzactii). Din teste a rezultat ca modelul folosind RFS fara afinitate arata o imbunatature a numarului de tranzacii de aproximativ 3.5% si utilizarea CPU-ului este comparabila, iar testul folosind afinitatea nucleelor arata o imbunatatire a numarului tranzactiilor cu aproximativ 12%.

	Requests per second					CPU util.
Model	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%

Pentru A-RADOS am facut 2 tipuri de teste (cu rezultatele listate mai jos): rularea fluxurilor celor 3 clase concurent si rularea doar a fluxurilor de clasa 3 folosind cozi ponderate si cozi fara ponderi. Primul test arata ca cea mai prioritara clasa este tratata prima, pe cand cel de-al doilea arata ca rezultatele celor doua teste sunt comparabile.

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 INFRASTRUCTURA CONVERGENTA IN CENTRE DE DATE

Pricipalele motive ale **Retelelor Convergente** (cunoscute si sub numele de **Unified Networks Infrastructure**) sunt in primul rand economice, dar si tehnologice [38] [39] [40], cum ar fi simplitate, costuri reduse, elasticitate, cerinte impuse de virtualizare, o mai buna utilizare a resurselor, administrare mai simpla si asa mai departe.

O caracteristica definitorie a protocoalelor de access la dispozitivele de stocare (ex. SCSI) este aceea ca nu trateaza in timp util pachetele pierdute si astfel un mediu fara pierderi ar fi necesar. De obicei, sistemele SAN folosesc Fibre Channel (sisteme ce ocupa aproximativ 80% din piata centrelor de date [40]) – prin definitie fiind un mediu fara pierderi cu latenta mica si rate de transfer mari. Pe cand protocolul Ethernet este un sistem de comunicatie de tip best-effort care impreuna cu protocolul IP ofera o retea end-to-end pentru protocoalele orientate conexiune (ex. TCP).

O retea convergenta ce suporta retele LAN si SAN impune un set de imbunatatiri (ex. Data Center Bridging) pentru a permite un mediu fara pierderi. De asemenea, Ethernet-ul a devenit o solutie viabila datorita avantajelor sale in comparatie cu retelele de tip FC, cum ar fi ratele mari de transmisie (pana la 10Gbps, 100Gbps, 400Gbps sau chiar 800Gbps, pe cand retelele de tip FC suporta rate pana la 2, 4, 8, 16 sau 32Gbps) sau costurile mai reduse.

Cu toate acestea, sunt cateva optiuni care pot fi folosite cu Ethernet-ul standard (ex. iSCSI, iFCP sau FCIP), dar protocoalele orientate conexiune trebuie adugate pentru a rezolva problemele de congestie (ex. TCP). Aceste variante impun costuri reduse, dar si performante mai mici din cauza penalizarilor datorate incapsularilor suplimentare [41].

Protocolul **Fibre Channel over Ethernet** (FCoE) este folosit pentru a permite folosirea pachetelor FC peste Ethernet, bazandu-se exclusiv pe extensiile DCB [42]: **Priority Flow Control** (PFC), **Enhanced Transmission Selection** (ETS), **Data Center Bridging Exchange** (DCBx) si **Quantized Congestion Notification** (QCN).

Aceste extensii pot fi folosite pentru orice aplicatie care necesita un mediu fara pierderi si nu are un sistem de rezolvare a congestiei si in acelasi timp necesita doar protocoale L2. Sunt idei si mai ingenioase, cum ar fi imbunatatirea problemei “TCP incast” [43].

Accentul principal a fost pe QCN (o alternativa la Fibre Channel), folosit mai departe pentru algoritmul QCN-WFQR.

5.1 Quantized Congestion Notification

Protocolul Quantized Congestion Notification [48] ([49], pg. 1071) permite administrarea congestiei la nivel peer-to-peer prin ajustarea dinamica a ratei datorita schimbarii blocajelor in absenta administrarii la nivele superioare. Este alcatuit din trei componente principale: punctele de congestie, reactie si reflexie. Punctele de congestie (CPs) esantioneaza pachetele receptionate si notifica sursele pentru a-si ajusta ratele de transmisie, astfel incat punctele de congestie sa-si poata mentine cozile de transmisie la un anumit nivel de echilibru:

$$F_b = -\left(Q_{offset} + \omega \cdot Q_\delta\right), \text{ unde}$$
$$Q_{offset} = Q - Q_{equilibrium}, \quad (1)$$
$$Q_\delta = Q - Q_{old} \text{ iar } \omega \text{ este ponderea ratei (considerata 2 in simularile de referinta).}$$

Limitatoarele de rata (RL) ajusteaza ratele curente (CR) pentru a ajunge la ratele specificate (TR) folosind un numarator de octeti in faze, dupa cum urmeaza:

- RP receptioneaza mesajul feedback si TR devine CR, iar CR scade dar nu mai mult de jumatate:

$$\boldsymbol{TR = CR}$$
$$\boldsymbol{CR = CR \cdot (1 - G_d \cdot |F_b|)} \quad (2)$$

- RP intra in faza de crestere a ratei compusa din trei parti: Recuperare Rapida (FR), Crestere Activa (AI) si Crestere Hiper-Activa (HAI). In faza de recuperare rapida punctele de reactie incearca sa recupereze banda pierduta, pe cand in celelalte faze verifica existenta unei latimi de banda suplimentare:

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

R este 0 in *FR*, 5Mbps pentru AI si *i*50Mbps pentru HAI (unde *i* este al i^{lea} cel mai mic ciclu numarat de catre BC si Timer in AI) pentru o viteza de transfer a liniei de 10Gbps.

Fazele de crestere a ratei de transmisie au 5 cicli. La sfarsitul fiecarui grup de 5 cicli, daca nici un F_b nu a fost receptionat, RL intra in urmatoarea faza (unde in faza HAI, rata RL-ului poate ajunge la rata de transmisie a liniei). In cazurile in care CR-urile sunt foarte mici si BC masurat in timp este prea mare, fazele sunt controlate de Timer. Timer-ul este similar cu BC-ul si astfel numara cicli de Tms (ex. T = 10ms pentru o linie de 10Gbps) in FR si $\frac{T}{2}$ms in AI.

Cu privire la BC si Timer, RL este in faza FR daca BC si Timer sunt in FR si CR este actualizat cand cel putin unul completeaza un ciclu. RL este in faza AI daca cel putin unul completeaza un grup de 5 cicli si in cele din urma RL este in HAI daca ambele sunt.

5.2 Punctele slabe ale QCN-ului si imbunatatiri

Datorita existentei diferitelor tipuri de dispozitive in cadrul retelei, coexistenta dispozitivelor cu control al congestiei si a dispozitivelor fara controlul congestiei este o provocare. Potrivit standardului, aceasta situatie este rezolvata folosind tabele de regenerare a prioritatilor. Astfel, in momentul in care un pachet receptionat dintr-un segment fara control al congestiei este introdus intr-un segment controlat din punct de vedere al congestiei, prioritatea acestui pachet este re-mapata la o prioritate nefolosita in domeniul cu congestie controlata, acest lucru implica faptul ca cel putin o prioritate trebuie pastrata pentru astfel de scenarii (i.e. prioritati alternative non-CNPV ([49], pg. 1071).

Deficientele sunt dezvaluite cand configurarea automata este activata si prioritatile alternative sunt alese automat, astfel potrivit standardului, prioritatea aleasa este cea mai mica pastrata nefolosita in CND. Acest lucru poate afecta politicile QoS folosite pentru alte tipuri de trafic si duce la injectii de trafic nedorit pentru acel domeniu QoS de la porturile *edge* (porturi aflate la granita dintre domeniile cu si fara congestie controlata).

O solutie ar fi verificarea fiecarui nod din topologie si verificarea tuturor cerintelor de QoS si prioritatile folosite. Nepractic, pentru ca o topologie poate avea un numar foarte mare de dispozitive de retea si configurarea nu ar mai fi automata. Solutia propusa contine un mod de alegere hibrid ce permite atat configurarilor manuale cat si celor automate sa foloseasca aceleasi prioritati alternative, astfel administratorul poate alege prioritatile alternative folosite de configurarile automate fara a afecta segmentele de QoS **[Patent #8891376]**.

O alta deficienta a QCN-ului este incorectitudinea ratelor multiplelor fluxuri de date ce impart acelasi punct de congestie. Pe scurt, la calculul feedback-ului nu se ia in considerare rata punctelor de reactie. Aceasta problema a fost rezolvata prin 2 algoritmi propusi: Approximate Fairness with QCN (AF-QCN [51]) si Fair Quantized Congestion Notification (FQCN [52]).

5.2.1 Solutia FQCN

Ideea principala a algoritmului FQCN este ca fiecare flux de date are o rata ponderata in raport cu coada congestionata si feedback-ul calculat per fiecare flux de date proportional cu depasirea cotei ratei. Astfel, algoritmul urmareste doua lucruri: in primul rand, identifica fluxurile de date cu rate prea mari si in al doilea rand, feedback-urile sunt calculate individual per flux de date folosind tehnici de monitorizare per flux.

In opinia mea, sunt doua dezavantaje la implementarea acestei solutii: 1) trebuie sa exista un contor per fiecare flux de date implementat in hardware, deorece implementarea in software poate avea probleme la rate foarte ridicate; 2) tabelele per flux de date pot fi foarte mari, caz in care administrarea fluxurilor este ingreunata.

6 ALGORITM DE BALANSARE DINAMICA BAZATA PE QCN

6.1 Motivare

Daca este aproape imposibil sa se poata prezice incarcarea retelei la un moment dat sau este imposibila construirea unei topologii sau a unei configuratii pentru a asigura domenii de congestie uniforme, atunci decizii luate pe baza de *profiler* pot duce la performante mai bune.

Propun **QCN Weighted Flow Queue Ranking** (QCN-WFQR), un algoritm bazat pe protocolul Quantized Congestion Notification **[PIST, 2015]**. Principala idee a algoritmului este calcularea unor serii de indici de congestie folositi de diferite entitati pentru a balansa traficul si pentru a servi mai bine scopurilor aplicatiilor. Algoritmul este construit generic pentru a putea fi folosit atat in retelele clasice de calculatoare, precum si in paradigma noua denumita Software Defined Networks.

6.2 Solutii alternative

In retelele cu control al congestiei (bazate pe QCN), fiecare punct de reactie receptioneaza mesaje de feedback de la mai multe puncte de congestie si isi ajusteaza rata in concordanta cu aceste mesaje. Pentru a obtine un profil mai bun al congestiei in retea, o solutie ar putea fi ca fiecare sursa de congestie (RP) sa isi creeze o baza de date cu indicativele de congestie per punct de congestie cu parametrii mesajelor de feedback receptionate de la fiecare punct de congestie (CPID, t, p, $Q_{nz}F_b$, Q_{off} si Q_δ) **[Apg. #20150023172]**. Apoi, punctele de reactie pot folosi aceste baze de date pentru analiza performantelor, iar administratorul poate lua decizii de schimbare a topologiei pentru a obtine performante mai bune. Sau punctele de reactie pot folosi aceste baze de date pentru a influenta diferite aspecte ale dispozitivelor de retea si a obtine un profil mai bun al congestie in sistem.

Eu am ales o alta cale, unde in locul bazelor de date create de catre punctele de reactie, fiecare punct de congestie sa creeze per fiecare punct de reactie pe care il deserveste o baza de date cu indici de congestie. Principala idee a acestui algoritm este de a calcula diferiti indici de congestie locali si per sistem, care apoi sa fie folositi in decizii luate de catre *profilere* de retea sau controlere de SDN sa duca la o crestere in performante (ex. migrarea dinamica de fluxuri de date), sau sa fie folositi in alegerea de noduri cu cea mai mica contributie la congestie in sisteme omogene.

6.3 Algoritmul: QCN Weighted Flow Queue Ranking (QCN-WFQR)

Indicativele de congestie locale sunt: **pondere flux**, **sarcina coada** si **rang coada**, iar cele la nivel de sistem sunt **sarcina flux** si **sarcina punct de reactie** (sau **sarcina dispozitiv**). In teza, propun doua aplicatii ce folosesc acest algoritm: o aplicatie de alegere a replicilor intr-un sistem distribuit si paralel de stocare de date si o aplicatie de migrare de fluxuri de date intr-un sistem SDN pentru a reduce nevoia de micsorare a traficului.

6.3.1 Indicative QCN-WFQR

6.3.1.1 Fluxuri

Un nod de calcul (ex. server) intr-un sistem distribuit ofera servicii prin intermediul fluxurilor. Un serviciu este identificat printr-un ID ce poate fi incapsulat in tag-uri QCN ([49], pg.1096).

6.3.1.2 Ranguri

Un **rang coada** (4) in cadrul unui port este definit ca numarul de fluxuri din cadrul cozii apartinand acelui port:

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

6.3.1.3 Pondere flux

O **pondere flux** (Eq. 6, 12) este definita ca o masura a congestiei din punctul de vedere al fiecarui nod din retea pentru un anumit flux de date relativ la coada unui port (i.e. punct de congestie).

6.3.1.3.1 Ponderi flux bazate pe QCN

Indicativul pondere flux trebuie sa contina informatie astfel incat sa poata fi comparate doua fluxuri ce congestioneaza acelasi punct dar si fluxuri ce congestioneaza puncte diferite avand rute diferite. Astfel, acest indicativ este o combinatie de trei indici: feedback-ul mediu in unitatea de timp, frecventa cu care un feedback este transmis si rangul cozii. Feedback-ul este o masura a congestiei relativ la coada fara a tine cont de rata fluxului; astfel consider ca ratele mai mari implica o probabilitate mai mare de a receptiona feedback-uri. Rangul cozii reflecta numarul de fluxuri ce sunt afectate in cazul congestiei (i.e. ideea principala este ca in cazul congestiei sa fie afectate cat mai putine fluxuri).

Pentru a reflecta frecventa de transmitere a feedback-urilor, cand un feedback este trimis catre un anumit punct de reactie, toate celelalte fluxuri din cadrul cozii respective isi recalculeaza feedback-ul mediu considerand ca au primit un feedback de valoare zero; astfel fluxurile cu rate mai mici au o probabilitate mai mare sa primeasca un numar mare de valori zero. Pentru considerente de implementare am ales sa folosesc o functie exponentiala pentru calculul mediei (EMA – Exponential Moving Average), dupa cum se observa mai jos:

$$\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right) = \begin{cases} \alpha_t \cdot \Psi_t^{q_k}\left(Fb_t^{q_k}\right) + (1-\alpha_t) \cdot \Psi_{t-1}^{q_k}\left(Fb_{t-1}^{q_k}, f_i\right), q_k\ esantioneaza f_{i,} \\ (1-\alpha_t) \cdot \Psi_{t-1}^{q_k}\left(Fb_{t-1}^{q_k}, f_i\right), altfel \\ \boldsymbol{MIN}_{f_j \in \mathcal{F}^{q_k}} \Psi_t^{q_k}\left(Fb_t^{q_k}, f_j\right), valoare\ initiala \end{cases} \quad (5)$$

Unde $\Psi_t^{q_k}\left(Fb_t^{q_k}\right)$ feedback-ul cuantificat la momentul *t* relativ la coada *k*.

Si $\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right)$ feedback-ul cuantificat pentru fluxul f_i la momentult *t* relativ la coada k.

Astfel, ponderea flux pentru fluxul f_i calculat de coada q_k la momentul *t* este:

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

6.3.1.3.2 Ponderi flux bazate pe FQCN

Daca este folosit algoritmul FQCN, atunci feedback-ul este calculat tinand cont de numarul de octeti receptionat in unitatea de timp T. Cele mai multe dispozitive de retea au abilitatea de a calcula rata de transfer medie per coada, dar in acest caz este nevoie de rata de transfer medie per flux de date, ce poate fi considerat un dezavantaj din moment ce este foarte greu de implementat acest lucru la nivel hardware.

Tinand cont de motivul prezentat mai sus, exista o posibilitate de aproximare a ratei de transfer per flux de date folosind aceeasi metoda EMA; astfel, rata medie la momentul *t* pentru fluxul f_i relativ la coada *k* este:

$$Rate_t^{q_k}(f_i) = \alpha_t \cdot Bytes(frame \in f_i) + (1-\alpha_t) \cdot Rate_{t-1}^{q_k}(f_i)\ , \quad (7)$$

Unde $\alpha_t = 1 - e^{-\frac{\Delta t}{T}}$

Apoi, pentru a calcula numarul de octeti receptionati in unitatea de timp *T* pentru fluxul $f_i$relativ la coada *k* este:

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

Ponderile FQCN si ponderile de granulatie mica FQCN sunt calculate astfel:

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

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

Feedback-ul FQCN pentru fiecare flux *inculpat* (flux ce isi depaseste ponderea) se calculeaza astfel:

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

Unde $\Psi_t^{q_k}\left(Fb_t^{q_k}\right)$ feedback-ul cuantificat la 64 de biti pentru fluxul f_i este $\Psi_t^{q_k}\left(Fb_t^{q_k}, f_i\right)$
Si S^R lista cu *inculpati* de granulatie mica.

La fiecare eveniment, ponderea flux este calculata astfel:

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

Diferenta dintr QCN-WFQR si QCN standard este ca QCN-WFQR bazat pe FQCN tine cont de ratele fluxurilor in momentul in care calculeaza feedback-ul; astfel, ponderile flux contin destula informatie pentru a putea compara fluxuri de date ce impart acelasi punct de congestie si fluxuri cu diferite rute si diferite puncte de congestie.

6.3.1.4 Sarcini

O **sarcina flux** (13) este definita ca o masura de congestie din punct de vedere al sistemului pentru un anumit flux, pe cand **sarcina punct de reactie** (14) este definita ca o masura decongestie din punct de vedere al sistemului pentru un anumit punct de reactie.

Astfel, o sarcina flux la momentul *t* pentru fluxul f_k este calculata ca suma ponderilor flux receptionate de la toate nodurile (i.e. $\mathcal{Q}^{f_k}$):

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

Iar o sarcina punct de reactie la momentul *t* pentru R_k este calculata ca suma tuturor sarcinilor flux pentru fluxurile a caror rata de transmisie este ajustata de punctul de reactie in discutie (i.e. $\mathcal{F}^{R_k}$):

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

O **sarcina coada** (15) la momentul t este definita ca indice de congestie masurat relativ la coada q_k si calculat ca suma a tuturor feedback-urilor medii ale tuturor fluxurilor controlate de coada in discutie (i.e. $\mathcal{F}^{q_k}$) si rang-ul ($R_t^{q_k}$):

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

Aceasta sarcina permite compararea de cozi astfel incat fluxuri de date pot fi rutate sau migrate, astfel obtinandu-se o mai buna balansare a traficului in retea.

6.3.2 Algoritmul QCN-WFQR bazat pe QCN standard

Calculul indicativilor locali: ponedere flux, rang coada si sarcina coada

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

```
procedure QCN-WFQR-UPDATE-FLOW-TABLE(q, p)      ° update flow table regarding packet p (if new flow)
    if flow(p) ∉ flow_table(q) then             ° 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)             ° 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
```

Calculul indicativilor de congestie la nivel de sistem:: sarcina flux, si sarcina punct de reactie

```
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 Analiza algoritmului

6.3.3.1 Analiza ponderilor flux

Principala idee a ponderilor flux este ca aceste indicative trebuie sa reflecte ratele punctelor de reactie iar in acelasi timp sa reflecte si valoarea feedback-urilor. Consider ca rata punctelor de reactie este reflectata de frecventa cu care fiecare feedback este trimis unui anumit flux, pe cand distanta pana la congestie este subliniata de valoarea feedback-urilor. Astfel, prima situatie permite compararea de fluxuri ce impart aceeasi coada dar cu rate diferite, pe cand cea de-a doua situatie permite compararea de fluxuri avand aceeasi rata, dar rute diferite si diferite cozi cu diferite conditii de congestie (i.e. diferite valori ale feedback-ului). Rezultatele simularilor sintetice arata ca indicele pondere flux raspunde la variatiile celor doua elemente: rata si feedback.

6.3.3.2 Analiza indicativilor de congestie

La un moment dat un flux nu poate avea mai multe puncte de congestie, pentru ca un punct de congestie implica cea mai mica rata pentru drumul fluxului de date. Dar, daca un alt punct se congestioneaza datorita schimbarilor in profilul congestiei la nivel de sistem, atunci rata scade si mai mult, astfel punctul de congestie anterior este eliberat.

Din punct de vedere al comparatiei fluxurilor, se disting trei situatii diferite, dupa cum urmeaza:

1. **Insule de congestie singulare**: ponderile flux comparate apartin aceluiasi punct de congestie;
2. **Insule de congestie deconectate**: ponderile flux comparate apartin la puncte de congestie diferite;
3. **Insule de congestive hibride**: ponderile flux comparate se afla in insule de congestie singulare, dar ocazional datorita schimbarii profilului de congestie al sistemului ponderile flux se afla in insule de congestie deconectate sau vice-versa.

6.3.3.3 Simulatorul pentru QCN-WFQR

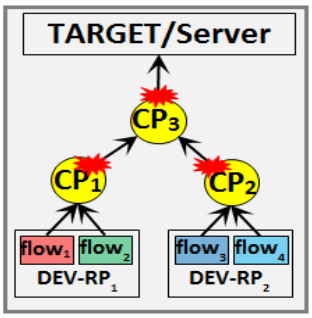
Pentru simulari a fost folosit un simulator open-source: NS-3 (simulator cu evenimente discrete) [53]. Au fost implementate urmatoarele module principale: puncte de reactie, puncte de congestie si profiler sau controler in cazul retelelor SDN.

6.3.3.4 Analiza simularilor QCN-WFQR

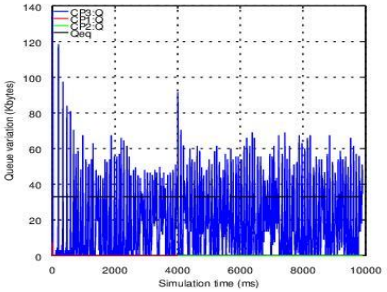
6.3.3.4.1 Contributia fluxurilor de date la profilul congestiei in sistem

Acest prim test urmareste sa sublinieze contributia fluxurilor de date la profilul congestiei: ponderi flux, sarcini flux si sarcini puncte de reactie. Cu alte cuvinte, punctele de reactie cu rate mai mici implica sarcini mai mici, astfel elementele sistemului pot decide asupra diferitelor locatii ale serviciilor pentru a obtine un profil al congestiei echitabil si o performanta mai buna. Pentru simulari am folosit o topologie sub forma de arbore binar, unde nodul radacina are un nod server. Din punctul de vedere al QCN-ului, fiecare nod cu control al congestiei are o coada de transmisie folosita pentru calculul feedback-urilor. Din punct de vedere al QCN-WFQR, ponderile flux folosesc ferestre de timp de 2 secunde, dar tabelele de flux au o functie de imbatranire de 250ms.

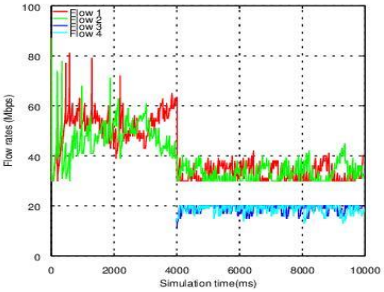
Se poate observa din rezultatele simularilor de mai jos, ca ponderile flux ale CP_3-ului, sarcinile flux si sarcinile puncte de reactie reflecta profilul ratelor fluxurilor de date:



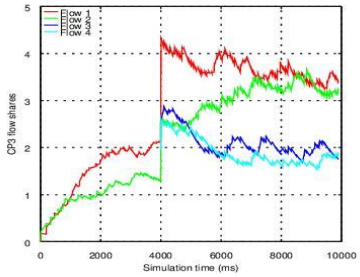
(a) Topologia



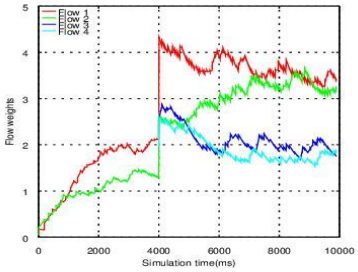
(b) Variatia cozilor CP-urilor



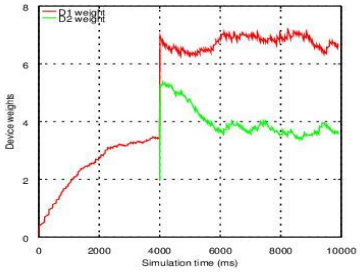
(c) Variatia ratelor fluxurilor



(d) Variatia ponderilor flux relativ la CP_3



(e) Variatia sarcinilor fluxurilor



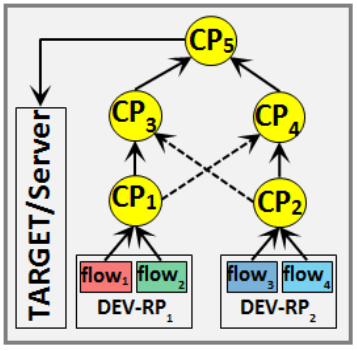
(f) Variatia sarcinilor punctelor de reactie

QCN-WFQR (NS3): rezultatele simularilor pentru variatia indicativelor relative la flux

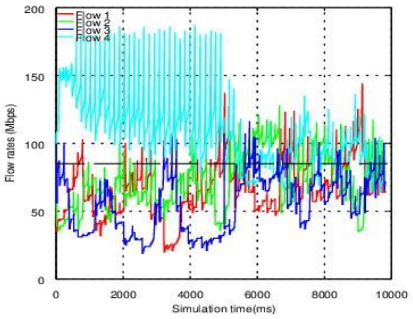
6.3.3.4.2 Variatia sarcinilor coada

Cea de-a doua simulare urmareste sa scoata in evidenta faptul ca variatia ratelor reflecta variatia sarcinilor cozi, astfel prin decizia de a migra fluxuri de date se obtine un trafic mai balansat obtinandu-se un profil al congestiei mai echitabil.

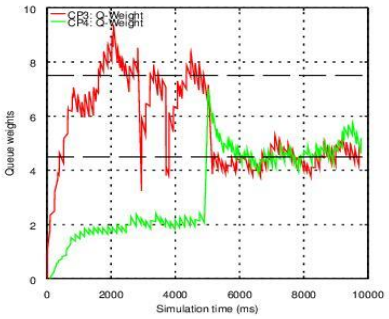
La fel ca la prima simulare, se foloseste o topologie arbore cu cai alternative, astfel fiecare dintre cele 4 fluxuri are 2 cai alternative. In primele 5 secunde al simularii, 3 fluxuri urmeaza aceeasi cale, apoi fluxul 4 se migreaza pe o cale alternativa si se obtine un trafic mai balansat in retea, dupa cum se observa in graficele urmatoare.



(a) Topologia



(b) Variatia ratelor fluxurilor



(c) Variatia sarcinilor cozilor CP_3, CP_4

QCN-WFQR (NS3): rezultatele simularilor pentru variatia indicilor de congestie relativa la cozi

6.3.4 Aplicatii pentru QCN-WFQR

6.3.4.1 Alegeri in sisteme distribuite cu noduri omogene

Intr-un sistem distribuit (sau cloud) cu multiple clase de servicii, unde fiecare clasa are mai multe noduri omogene, propun ca alegerea unui anumit serviciu sa se faca pe baza algoritmului QCN-WFQR. Aceasta metoda urmareste maximizarea utilizarea benzii in retea si imbunatatirea timpului de raspuns al sistemului.

Daca numarul de clienti variaza si nevoile lor sunt necunoscute, atunci tiparul congestiei retelei este non-deterministic. In momentul in care este cerut un serviciu, propun ca alegerea nodului sa fie facuta pe baza de indicative de congestie furnizate de profiler-ul QCN-WFQR.

Pentru exemplificare, in sistemele distribuite si parelele, un fisier este divizat in segmente de date sau obiecte replicate pe mai multe dispozitive, astfel ca un segment de date poate fi transferat de pe orice replica. Cele mai multe sisteme de stocare distribuite folosesc o distributie statica (ex. CRUSH), dar cu aparitia retelelor convergente o distributie statica nu mai este cea mai buna alegere. Astfel, prin folosirea algoritmului QCN-WFQR, o replica poate fi aleasa dinamic si astfel se poate imbunatatii timpul de raspuns al sistemului prin incercarea de a evita strangulari in retea. In acest caz merita mentionat ca punctele de reactie sunt dispozitivele de stocare.

Inima acestei metode tine de faptul ca indicativele de congestie comunicate profiler-ului si indicativele comunicate clientului trebuie sa reflecte starea de congestie reala a sistemului. Astfel, daca informatia este comunicata prea des, profiler-ul poate fi complesit sau daca este comunicata prea rar, aceasta informatie e posibil sa numai reflecte profilul real al congestiei sistemului. Acest lucru inseamna ca algoritmul este foarte sensibil la variatii violente ale profilului, astfel propun folosirea de segmente de date mari.

In afara de indicativele de congestie comunicate, performanta si implementarea profiler-ului influenteaza foarte mult comportamentul sistemului: o implementare distribuita dar centralizata din punct de vedere logic sau o implementare centralizata complet. Plus, profiler-ul trebuie sa aiba o latenta mica cu fiecare punct de congestie si cu fiecare client, astfel ca indicativele de congestie sa fie comunicate in timp util.

6.3.4.2 Fluxuri balansate dinamic in retele de calculatoare

Performanta algoritmului este influentata direct de latenta cu care profiler-ul de retea sau controlerul SDN primeste indicativele de congestie relevante, decide si muta fluxurile de date si realizeaza un trafic mai balansat in topologia retelei.

In primul rand, proiectarea profiler-ului/controler-ului urmareste fie un model centralizat fie unul distribuit [54]. Fiecare metoda are avantajele si dezavantajele sale. De exemplu, un model centralizat are limitari de scalabilitate, o probabilitate de gatuire mai mare, un punct singular de cadere, dar are o complexitate mai simpla, pe cand un model distribuit scaleaza mai usor indeplinind cerintele de performanta mai bine, este mai tolerant la caderi, dar semantica sincronizarii mai greoaie si o implementare mai complexa [54].

Iar in al doilea rand, cum sunt comunicate indicativele de congestie are o mare relevanta in deciziile algoritmului. Aceasta informatie poate fi transmisa la intervale scurte si astfel controlerul este supraincarcat sau informatia poate fi filtrata si condensata la sursa inainte de a fi transmisa la controler.

7 CONCLUZII

In prima parte a tezei se face o analiza generala a sistemelor distribuite de stocare, urmarind aspecte precum: clasificari, detalii de arhitectura sau aspecte ale retelei. Urmata de o propunere de infrastructura pentru sistemele eLearning, ca studiu de caz. In urmatoarele capitole se propun cateva optimizari pentru sistemele distribuite in general si pentru sistemele distribuite de stocare in special, urmate de un algoritm original pentru balansarea dinamica a traficului intr-o retea cu congestie controlata (QCN-WFQR).

7.1 Contributiile originale ale tezei

Contributii la analiza sistemelor de stocare distribuite (capitolul 2)

Am propus o taxonomie simpla si cuprinzatoare a sistemelor de stocare (sectiunea 2.1). Apoi, pentru caracterizarea sistemelor de stocare distribuite am propus un set relevant de cerinte (sectiunea 2.2) si am scos in evidenta metodele de scalare pe orizontala bazate pe diferite niveluri ale sistemelor de fisiere, astfel construind multiple tipuri de sisteme de stocare distribuite (sectiunea 2.3).

De asemenea, am propuns o schema generala a sistemelor de stocare scotand in evidenta similitudini intre sistemele distribuite si sistemele centralizate de stocare (sectiunea 2.5.3).

Analiza sistemelor de stocare distribuite si contributiile relevante au fost publicate in **[PIST, 2014/1]**.

Contributii la infrastructura mediilor eLearning (capitolul 3)

Am propus un sistem de fisiere paralel si distribuit adaptat pentru mediile eLearning gazduite de sisteme cloud. Sistemul de stocare propus poate fi considerat o alternativa la OGSA-GFS. Are doua componente principale: un spatiu de adresare liniara bazat pe RADOS si un modul pentru metadate semantice. Am propus o implementare nativa a metadatelor semantice cu semantica de tip continut pentru a imbunatati cautarile de date. Din punct de vedere al retelei, am propus o topologie ierarhica ce faciliteaza traficul de volum mare sud-nord cu legaturi adaugate pentru a micsora latenta intre nodurile RADOS-ului.

Aceste contributii au fost publicate in **[PIST, 2014/2]**.

Optimizari bazate pe motoare de procesare de pachete (capitolul 4)

Am propus cateva solutii de optimizare pentru clustere cu trafic intensiv folosind sisteme integrate de multi-nuclee cu motoare de procesare de pachete (sectiunile 4.2, 4.3). Apoi, in timp ce Ceph adreseaza diferite optimizari la nivelul sistemului, contributiile mele se concentreaza pe imbunatatirea performantei fiecarui nod (sectiunea 4.4).

Performanta fiecarui nod este imbunatatita prin clasificarea si distributia cerintelor printre nucleele sistemului, facilitand paralelizarea consumatorilor de trafic si arhitectura software si, mai departe deciziile programatorului sistemului de operare sunt facilitate la randul lor de folosirea mecanismului Accelerated Receive Flow Steering (sectiunea 4.3). De asemenea, am propus doua modele de accelerare hardware si doua implementari diferite pentru fiecare: una in spatiul nucleu, urmarind eficienta si una in spatiul utilizator, facilitand dezvoltarea rapida si politici de licentiere flexibile. Din punct de vedere al Ceph-ului, am propus un model pentru fiecare tip de nod bazat pe tipurile de fluxuri si volumul de trafic (sectiunea 4.4.2) folosind sistemul QorIQ (sectiunea 4.4.1).

Latenta operatiilor sensibile iar timpul de raspuns si disponibilitatea sistemului sunt imbunatatite prin acordarea de prioritati cozilor hardware. RADOS accelerat (A-RADOS) divide traficul

monitoarelor si OSD-urilor in trei clase diferite cu prioritati diferite bazate pe importanta impusa de sistemul Ceph (sectiunea 4.4.3).

Masuratorile de performanta ce au aratat imbuntatiri la numarul de tranzactii pe secunda si procesarea mesajelor RADOS-ului per prioritati au fost prezentate in sectiunea 4.4.4. In afara de imbunatatirile prezentate sunt si avantaje mostenite prin folosirea sistemelor integrate multi-nucleu cum ar fi consumul mic de energie sau pretul atractiv per performanta.

Optimizarile propuse pentru motoarele de procesare de pachete precum si rezultatele masuratorilor au fost publicate in **[PIST, 2013]**.

Solutii la problemele aparute in infrastructuri convergente ale centrelor de date (capitolul 5)

Printre protocoalele din grupul suport ce a fost adaugat Ethernet-ului in cotextul infrastructurilor convergente, eu m-am concentrat pe QCN, care are cateva slabiciuni, precum QoS-ul segmentelor cu si fara control al congestiei sau inechitabilitatea ratelor fluxurilor ce imart acelasi punct de congestie.

Am facut parte dintr-o echipa ce a propus o solutie patentata pentru configurarile automate ale parametrilor QCN-ului prin protocolul LLDP in retele hibride unde segmentele cu si fara QCN coexista si au diferite cerinte de QoS. Solutia implica un noua metoda ce permite atat setarilor manuale cat si celor automate sa foloseasca aceeasi prioritate alternativa, astfel injectiile de trafic nedorit pentru un domeniu QoS specific sunt evitate (sectiunea 5.2).

Patentul este public si se poate inspecta la **[Patent #8891376]**.

Algoritm pentru balansarea dinamica a incarcarii retelei: QCN-WFQR (capitolul 6)

Relativ la slabiciunile domeniilor de congestie ne-echitabile, am propus un algoritm original: **Quantized Congestion Notification – Weighted Flow Queue Ranking**.

QCN-ul nu rezolva existenta domeniilor de congestie ne-echitabile si astfel am propus o solutie partiala ce incearca sa rezolve aceasta problema. Propunerea se bazeaza pe alegerea *gateway*-ului cu cea mai mare distanta pana la congestie; aceasta solutie a condus catre o metoda mai generica ce a fost patentata, unde sursele de congestie creaza baze de date pentru fiecare punct de congestie. Aceste baze de date se pot folosi pentru analiza performantelor retelei sau pot fi folosite pentru a influenta diferite aspecte ale dispozitivelor din retea (sectiunea 6.2).

Algoritmul **QCN-WFQR** calculeaza multiple indicative de congestie ce sunt masuri ale incarcarii retelei generate de fluxuri de date in diferite puncte ale retelei, folosite apoi in balansarea incarcarii in retea (sectiunea 6.3.1).

Algoritmul este capabil sa calculeze contributia la congestie a fiecarui flux in diferite puncte de retea ($\boldsymbol{Share_t^{q_k}(f_i)}$), la nivel de sistem ($\boldsymbol{Weight_t(f_k)}$), precum si contributia fiecarui punct de reactie ($\boldsymbol{Weight_t(R_k)}$) la congestia sistemului. Bazat pe aceste indicative de congestie orice componenta din cadrul sistemului poate determina cea mai potrivita locatie a unui serviciu in scopul balansarii in mod cooperativ incarcarea retelei. Mai departe, prezint analiza ponderilor flux si a geometriei sistemului (i.e. insule de congestie) relative la fiecare flux (sectiunile 6.3.3.1 si 6.3.3.2). Merita mentionat faptul ca algoritmul QCN-WFQR foloseste o functie exponentiala (EMA) pentru a stoca ponderile flux in scopul de a micsora utilizarea memoriei (sectiunea 6.3.1.1).

Algoritmul este de asemenea capabil sa calculeze importanta fiecarei cozi ($\boldsymbol{R_t^{q_k}}$) relativ la congestia in sistem prin numarul de fluxuri tratate si gradul de congestie al fiecarei cozi ($\boldsymbol{Weight_t(q_k)}$) relativ la toate fluxurile tratate. Folosind aceste indicative, o retea poate fi reprogramata sa balanseze optim incarcarea intre diferite puncte de congestie.

Pentru simulare am implementat algoritmul QCN-WFQR in NS3, unde in afara de obiectele algoritmului, am implementat si obiectele QCN-ului, anume: limitatorul ratei (RL) precum si punctele de congestie si de reactie (CP si RP), sectiunea 6.3.3.3. Am facut doua simulari diferite urmarind contributia fluxurilor de date la profiul congestie si variatia indicativelor de congestie ale coziilor (sectiunea 6.3.3.4). Primul test arata relatia dintre indicativele de congestie relative la fluxurile de date si ratele fluxurilor (sectiunea 6.3.3.4.1), in timp ce cea de-a doua simulare arata relatia dintre in indicativele de congestie relative la cozi si ratele fluxurilor de date (sectiunea 6.3.3.4.2).

Am propus si doua aplicatii ale algoritmului: alegerea de replici in sistemele de stocare distribuite bazata pe indicativele de congestie pentru a obtine un profil de congestie mai balansat si o metoda de balansare dinamica ce se preteaza la retelele de tip SDN.

Solutia patentata poate fi inspectata la **[Apg. #20150023172]**, in timp ce algoritmul si rezultatele simularilor au fost publicate in **[PIST, 2015]**.

7.2 Activitati viitoare

In contextul folosirii moatarelor de procesare de pachete se urmareste studierea de metode de optimizare la transmiterea pachetelor, in timp ce in contextul QCN-WFQR se va extinde standardul Openflow pentru a incorpora indicative QCN-WFQR. Se vor efectua simulari cu numar foarte mare de noduri pentru a observa cum se comporta algoritmul in conditii de stres. Nu in ultimul rand, se va implementa un controler SDN si algoritmi de rutare (ex. Dijkstra) ce vor folosi diferite indicative de congestie. De asemenea, se vor studia si diferite tipuri de trafic si modul in care algoritmul va raspunde.

8 PUBLICATIILE AUTORULUI

8.1 Patente

[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 Articole

[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, pg. 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, pg. 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, pg. 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, pg. 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, pg. 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, pg. 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 BIBLIOGRAFIE

[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, pg. 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. pg. 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, pg. 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, pg. 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, pg. 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, pg. 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, pg. 923-933

[29] Direct interconnection networks, http://pages.cs.wisc.edu/~tvrdik/5/html/Sectiunea5.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, pg. 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, pg. 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, pg. 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, pg. 58-65, 2010

[52] Yan Zhang, Nirwan Ansari, Fair Quantized Congestion Notification in Data Center Networks, IEEE Transactions on Communications, pg. 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, pg. 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, pg. 8-13