UNIUNEA EUROPEANĂ

Instrumente Structurale 2014-2020

Programul Operațional Competitivitate

CeCBiD-EOSC

Centru Cloud și Big Data pentru participarea la Cloud-ul European pentru Știință Deschisă (CeCBiD-EOSC)

Raport Științific și Tehnic 2.3

Servicii și aplicații informatice pentru satisfacerea cerințelor IT din faza operațională inițială a proiectului ELI-NP

Editor: Mihnea Dulea
Autori: Alecsandru Vladimir Chiroșca
Tudor Mitran
Dragoș Ciobanu-Zabet
Ionuț Vasile
Versiunea: Finală (1.0)
Data: 31.07.2023
Distributie: Internă
Cod document: CBD_RST-2.3

Rezumat: Proiectul POC CeCBiD-EOSC a fost derulat de către echipa Departamentului Fizică Computațională și Tehnologia Informației din IFIN-HH în perioada 2020-2023. Acest document raportează rezultatele obținute în cadrul Subactivității 2.3 – *"Realizarea de servicii și aplicații informatice pentru satisfacerea cerințelor de calcul și de stocare de date din faza operațională inițială a proiectului ELI-NP"*. Sunt prezentate: a) implementarea în cadrul Centrului Cloud și Big Data (CCBD) a sistemului de arhivare a datelor generate la ELI-NP; b) realizarea de servicii și aplicații software oferite prin intermediul virtualizării pentru simulari numerice și analiza datelor experimentale. Partea a doua a studiului este consacrată dezvoltării de instrumente software noi ce utilizeaza tehnici Big Data pentru prelucrarea și analiza datelor de monitorizare a Sistemului Laser de Mare Putere, atat în regimul de testare cât și în cel de producție științifică. Software-ul dezvoltat a fost integrat într-o patformă dedicată, gazduită in infrastructura CCBD.

Lista reviziilor

Data	Versiune	Editor/Autor/Coautori	Sumarul modificărilor/completărilor
12.05.2023	0.0	M. Dulea	Structură document; Prefață; 1.2 Obiective
22.05.2023	0.1	M. Dulea	1.1 Context general și necesitate
29.05.2023	0.2	M. Dulea	2.1 Necesitate; 2.2 Implementarea sistemului …
06.06.2023	0.3	M. Dulea	4.1 Obiectiv și …; 4.3.1 Obiectivele analizei …
07.06.2023	0.4	T. Mitran	4.2.1 Analiza imaginilor …; 4.2.2 Date de …
19.06.2023	0.5	M. Dulea / D. Ciobanu, I. Vasile	3. Servicii si aplicații software pentru cercetare
14.07.2023	0.6	A. Chirșca	4.3 Analiza datelor în regimul de producție științifică
26.07.2023	0.7	A. Chiroșca	5. Algoritmi de învățare automată
31.07.2023	1.0	M. Dulea	Rezumat; 6. Concluzii

Prefață

Investițiile în infrastructura de Cloud computing și de Big Data în vederea maximizării potențialului de creștere a economiei digitale europene reprezintă una dintre direcțiile prioritare ale Strategiei Pieței Unice Digitale, care a fost stabilită în 2015 de către Comisia Europeană.[1]

Recunoscând capabilitațile de exploatare a fenomenului Big Data oferite de tehnologia Cloud, Comisia a lansat în aprilie 2016 Inițiativa Europeană Cloud[2], a carei implementare se bazeaza pe Cloud-ul European pentru Stiinta Deschisa (*European Open Science Cloud* – EOSC) si pe Infrastructura Europeana de Date (*European Data Infrastructure* – EDI).

In viziunea Comisiei Europene, EOSC trebuie să asigure pentru comunitatea științifică un mediu virtual sigur, deschis, capabil să ofere servicii de stocare, management, analiză, precum și de refolosire a datelor dincolo de frontiere și discipline stiintifice. În acest cadru, s-a recomandat infrastructurilor europene de cercetare (și în primul rând infrastructurilor ESFRI) sa promoveze reutilizarea datelor proprii pentru inovare și în scopuri educaționale prin sprijinirea conectării lor la EOSC.[3]

Parcursul de Implementare a EOSC[4] a stabilit liniile de acțiune pentru crearea unei federații pan-europene a infrastructurilor de date pentru cercetare, care să înlocuiască fragmentarea existentă cu soluții eficiente și ușor de utilizat pentru stocarea, găsirea, partajarea și refolosirea datelor. Direcțiile de acțiune propuse pentru implementarea modelului federalizat al EOSC privesc arhitectura sistemului, administrarea datelor, serviciile, accesul și interfețele de acces, regulile de participare, precum și guvernanța. Arhitectura EOSC cuprinde un nucleu federativ, care include resursele partajate ale EOSC, precum și multiple infrastructuri de date federate angajate în furnizarea de servicii către EOSC.

Începand din anul 2018, IFIN-HH a contribuit la implementarea infrastructurii EOSC prin intermediul Departamentului Fizică Computațională și Tehnologia Informației (DFCTI, https://cc.ifin.ro), care a participat, în calitate de asociat al coordonatorului, Fundația EGI[5], la proiectul H2020 EOSC-Hub[6] (2018-2020), destinat dezvoltării resurselor și serviciilor Cloud inițiale pentru EOSC. Sarcina DFCTI a fost de a furniza, prin intermediul centrului Cloud CLOUDIFIN[7], resurse pentru susținerea diferitelor comunități de utilizatori, precum și de a contribui cu servicii naționale la catalogul de servicii al EOSC, în conformitate cu regulile de angajare ale proiectului. În acest scop, EOSC-Hub a finanțat activitatea de management și operare a site-ului CLOUDIFIN, asigurând continuitatea furnizării acestor servicii.

Pentru a putea finanța realizarea masei critice de resurse necesară participării la EOSC, DFCTI a propus proiectul CeCBiD-EOSC in cadrul apelului POC 398/2018. Totodată, pentru continuarea implementării serviciilor specifice EOSC, DFCTI participă, incepînd din 2020, la proiectul H2020 EGI-ACE – *„Advanced Computing for EOSC*" (2020-2023), a cărui misiune este de a asigura servicii EOSC gratuite pentru cercetătorii din toate disciplinele științifice care necesită calcule intensive și Big Data. Astfel, proiectele EGI-ACE și CeCBiD-EOSC acționeaza complementar, la nivelul EU și, respectiv, național, pentru realizarea strategiei de integrare a infrastructurii de calcul și de date a IFIN-HH în EOSC.

Obiectivul general al proiectului CeCBiD-EOSC este creșterea capacității de cercetare în scopul ridicării nivelului de competitivitate științifică pe plan intern și internațional al IFIN-HH, prin modernizarea infrastructurii Cloud, extinderea infrastructurii masive de date și realizarea unui centru de date cu performanțe înalte, care să fie integrat în infrastructura Cloud Europeană

[1] "*A Digital Single Market Strategy for Europe*" - COM(2015) 192
[2] "*European Cloud Initiative – Building a competitive data and knowledge economy in Europe*" – COM (2016) 178
[3] "*Long-term sustainability of Research Infrastructures*" – SWD(2017) 323
[4] "*Implementation Roadmap for the European Open Science Cloud*" - SWD(2018) 83
[5] Fundatia EGI, https://www.egi.eu/about/egi-foundation/
[6] "*Integrating and managing services for the EOSC"*, https://www.eosc-hub.eu/
[7] Centrul de resurse Cloud al DFCTI, CLOUDIFIN, http://cloudifin.ifin.ro/

pentru Știința Deschisă.[8]

Totodata, proiectul propune o soluție tehnică pentru interconectarea la nivel național, în cadrul unui Cloud federalizat, a centrelor de tip Cloud privat dezvoltate în instituții aparținând sistemului de CDI, capabilă să ofere utilizatorilor acces printr-o interfață unică la resurse și servicii furnizate de aceste centre. Implementarea acestei soluții va eficientiza utilizarea resurselor Cloud de către grupurile de cercetători și va stimula semnificativ cooperarea între specialiștii în tehnologii informatice avansate.

Infrastructura realizată în cadrul proiectului va susține dezvoltarea unor activități de CDI în domeniile sistemelor de calcul paralel și distribuit, învațării automatizate, calculului științific și bioinformaticii, cu aplicații relevante pentru fizica materiei condensate, studiul interacției laser-materie, nanofizică și nanoelectronică.

Obiectivele specifice ale proiectului CeCBiD-EOSC au fost următoarele:

1. Realizarea unui centru performant de resurse Cloud și Big Data prin achiziționarea și instalarea de active corporale și necorporale necesare pentru derularea activităților de CDI prevăzute în proiect.

2. Dezvoltarea și diversificarea serviciilor furnizate la nivel european de catre centrul CLOUDIFIN în perspectiva integrării acestuia în EOSC.

3. Realizarea în cadrul centrului de resurse Cloud a unei soluții tehnice capabilă să interconecteze la nivel național, în cadrul unui sistem federalizat, centrele de tip Cloud privat dezvoltate în instituții aparținând sistemului de CDI, care să ofere utilizatorilor acces printr-o interfață unica la resurse și servicii furnizate de către aceste centre.

4. Asigurarea condițiilor de susținere informațională în tehnologie Cloud și Big Data a participării institutului la colaborări internaționale de anvergură, precum și a noilor opțiuni strategice privind angajarea în direcții de cercetare emergente din spațiul științific internațional, cu relevanță socio-economică deosebită.

5. Dezvoltarea și implementarea de servicii și aplicații informatice pentru administrarea și funcționarea centrului de resurse, care utilizeaza tehnologii Cloud și Big Data pentru: satisfacerea cerințelor IT din faza operațională a proiectului Extreme Light Infrastructure - Nuclear Physics (ELI-NP); modelarea și simularea la nivel molecular a nano- și biosistemelor complexe; analiza datelor de secvențiere de nouă generație.

6. Realizarea condițiilor tehnice și asigurarea suportului de specialitate pentru obținerea și/sau îmbunătățirea de către parteneri economici, în special din cadrul Clusterul Tehnologic Magurele (MHTC), a unor produse și servicii în domenii de specializare inteligentă, care vor conduce la creșterea competitivității acestora.

7. Formarea și perfecționarea personalului calificat, precum și transferul de cunoștințe în domeniile Cloud computing și Big Data către personalul științific și tehnic din alte entitati ale sistemului de CDI.

8. Diseminarea rezultatelor proiectului prin participarea la conferințe (inter)naționale și publicarea de articole științifice în parteneriat public-privat.

Prin obiectivele sale, proiectul conduce la extinderea capacității resurselor Cloud și Big Data, precum și la îmbunătățirea calitativă și diversificarea serviciilor de calcul și de analiza de date pe care Centrul de Calcul Avansat din IFIN-HH le va oferi comunității științifice naționale și internaționale, contribuind prin aceasta la dezvoltarea sistemului național de CDI și la creșterea vizibilității la nivel european.

Rezultatele prevăzute ale proiectului sunt prezentate mai jos.

[8] Cerere de Finantare, proiect CeCBiD-EOSC

Nr.	Rezultat prognozat	Documentul în care a fost raportat
1.	Documentatia de achizitie a serviciilor de consultanta pentru elaborarea documentatiei tehnice necesare echiparii centrului de resurse Cloud si Big Data	RP2
2.	Contract de furnizare a serviciilor de consultanta pentru elaborarea documentatiei tehnice necesare echiparii centrului de resurse Cloud si Big Data	RP2
3.	Documentatie tehnica privind echiparea centrului de resurse Cloud si Big Data	RP3
4.	Documentatia de achizitie a serviciilor de informare si publicitate	RP1
5.	Contract de achizitie servicii de informare si publicitate	RP1
6.	Contracte de achizitie active corporale pentru echiparea centrului de resurse Cloud si Big Data	-
7.	Un comunicat de presa publicat la lansarea proiectului	RP1
8.	Un comunicat de presa publicat la finalizarea proiectului	
9.	Pagina web a proiectului, publicata la adresa http://cecbid-eosc.nipne.ro	RP1
10.	Materiale de informare si publicitate (roll-up-uri, afise, rame afis, mape, banner).	RP1
11.	Sistem de procesare de date achizitionat si instalat. Raport tehnic	-
12.	Sistem de stocare de date achizitionat si instalat. Raport tehnic.	-
13.	Switch pentru interconectarea sistemelor hardware achizitionat si instalat. Raport tehnic.	-
14.	Instalatie de climatizare achizitionata si instalata. Raport tehnic.	-
15.	Sistem UPS achizitionat si instalat. Raport tehnic.	-
16.	Rack-uri pentru gzduirea echipamentelor IT achizitionate si instalate.	-
17.	Tablouri electrice si retea electrica achizitionate si instalate. Raport tehnic.	-
18.	Servicii si aplicatii informatice pentru administrarea si monitorizarea centrului CLOUDIFIN, precum si pentru asigurarea accesului utilizatorilor.	RP1-4
19.	Interfata de acces al utilizatorilor la resurse oferite de centre Cloud multiple. Manual de utilizare.	-
20.	Servicii si aplicatii informatice pentru satisfacerea cerintelor IT din faza operationala initiala a proiectului ELI-NP. Raport stiintific si tehnic	-
21.	Servicii si aplicatii informatice pentru suportul activitatii de modelare si simulare a nanostructurilor complexe (partial)	RP1-4
22.	Servicii si aplicatii informatice pentru suportul activitatii de analiza a datelor de secventiere de noua generatie (partial)	RP1-4

23.	Studiu privind performantele centrului Cloud si Big Data. Manual de management	-
24.	Lucrări și comunicări științifice	RP2-4
25.	Fișe de post	RP1-2
26.	Documente de raportare (rapoarte de activitate / de progres; procese verbale ale intalnirilor echipei de management a proiectului)	RP1-4
27.	Cereri de plată/rambursare	RP4
28.	Raport de audit final al proiectului.	-

Proiectul CeCBiD-EOSC a demarat la data semnării contractului de finațare de catre Ministerul Educației și Cercetării (19.05.2020) si a fost finalizat in luna iulie 2023.

Proiectul CeCBiD-EOSC este cofinanțat din Fondul European de Dezvoltare Regională (FEDR) în baza contractului de finanțare încheiat cu Ministerul Educației și Cercetării în calitate de Organism Intermediar, în numele și pentru Ministerul Fondurilor Europene în calitate de Autoritate de Management.

Cuprins

1. Introducere 11
1.1 CONTEXT GENERAL ȘI NECESITATE 11
1.1.1 Date furnizate de sisteme de monitorizare și control 12
1.1.2 Fluxul datelor științifice 13
1.2 OBIECTIVELE SUBACTIVITĂȚII 2.3 15
2. Stocarea pe bandă magnetică a datelor de la ELI-NP 16
2.1 NECESITATE 16
2.2 IMPLEMENTAREA SISTEMULUI DE ARHIVARE A DATELOR 16
3. Servicii și aplicații software pentru cercetare 17
3.1 MAȘINI VIRTUALE PENTRU SIMULĂRI NUMERICE ȘI ANALIZA DE DATE 17
3.2 DISTRIBUIREA IMAGINILOR DE VM-URI ÎN CENTRE CLOUD MULTIPLE 17
3.3 TESTAREA APLICAȚIILOR DE SIMULARE A INTERACTIEI LASER-MATERIE 17
4. Prelucrarea și analiza fluxului de date de monitorizare 19
4.1 OBIECTIV ȘI STRATEGIE 19
4.2 ANALIZA DATELOR ÎN REGIMUL DE TESTARE A HPLS 19
4.2.1 Analiza imaginilor furnizate de camerele CCD 19
4.2.2 Date de monitorizare a sistemelor de pompaj laser 27
4.3 ANALIZA DATELOR ÎN REGIMUL DE PRODUCȚIE ȘTIINȚIFICĂ 30
4.3.1 Obiectivele analizei datelor de monitorizare în regim de producție 30
4.3.2 Platforma de prelucrare și analiză a datelor 33
4.3.3 Analiza de performanță 38
5. Algoritmi de învățare automată 40
6. Concluzii 42
Mulțumiri 42
ANEXA 1 – Codul de calcul al distanţei Wasserstein secţionate 43
ANEXA 2 – Corespondența dintre denumirile fișierelor și poziționarea senzorilor 46
ANEXA 3 – Script de procesare date 47
ANEXA 4 – Codul de agregare a datelor 52
ANEXA 5 – Codul de validare 53
ANEXA 6 – DAG-ul de ingestie 58
ANEXA 7 – DAG-ul de agregare 63
ANEXA 8 – DAG-ul de procesare a datelor 67
ANEXA 9 – Generarea fișierului tfrecords folosit pentru antrenare 72
ANEXA 10 – Generarea și antrenarea modelului ML 77

Lista figurilor

FIG. 1.1: Cele 3 sisteme de amplificare a puterii fasciculului laser situate pe un braț al HPLS[18] 12
FIG. 1.2: Fluxul de date științifice în și între ELF și infrastructura CLOUDIFIN 14
FIG. 3.1: Interfata web a imaginilor de masini virtuale 17
FIG. 3.2: Teste de scalabilitate pe un server fizic si masina virtuala 18
FIG. 3.3: Teste de scalabilitate pe doua server fizice si masini virtuale 18
FIG. 4.1: (a) Suma intensităților și (b) energiile colectate pentru aceeași secvență de semnale 20
FIG. 4.2: Imaginea la momentele (a) 0, (b) 200 si (c) 2000 asociate secventei din Fig. 4.1. 20
FIG. 4.3: (a) Rezultatul interpolarii cu trei Gaussiene 2D a imaginii din dreapta (b) 21
FIG. 4.4: (a) Reprezentarea a 6x2000 de imagini; (b) intensitatea totala a pixelilor din (a). 22
FIG. 4.5: Imaginea #66541038, corespunzatoare pulsului usor decelabil din Fig. 4.4 22
FIG. 4.6: Captură de zgomot în absența fasciculului 23
FIG. 4.7: Efectul reducerii zgomotului de fundal la diferite intensități ale fasciculului laser 23
FIG. 4.8: Evaluarea calității modelului 24
FIG. 4.9: Exemplu de convergenta DWS în funcţie de numărul de secţiuni 26
FIG. 4.10: Proiecţii asociate celor 5 secţiuni ale distribuţiei 2D 26
FIG. 4.11: Matricea distanţelor pentru 50 de profile de fascicul 27
FIG. 4.12: Numărul și valorile datelor colectate, în ordine cronologică 28
FIG. 4.13: Întreruperile și intervalele de timp fără înregistrări 28
FIG. 4.14: Procentul de timp petrecut de laserii de pompaj într-o anumită stare 29
FIG. 4.15: Reducerea zgomotului din semnalul energimetrelor folosind un fitru cu profil gaussian 30
FIG. 4.16: Structura datelor din baza NOSQL 32
FIG. 4.17: Structura datelor după procesul de agregare 33
FIG. 4.18: Nodurile cluster-ului 34
FIG. 4.19: Servicii furnizate de cluster 35
FIG. 4.20: Caracteristicile setului de date furnizate de HPLS și utilizate în acest raport 37
FIG. 4.21: Timpii totali de procesare a fisierelor 39
FIG. 4.22: Imagine preluată de la sistemul de amplificare laser 41
FIG. 4.23: Antrenare și validare pentru rețea CAE 41

Rezumat

Scopul livrabilului

Scopul acestui raport este de a prezenta rezultatele obținute în cadrul Subactivității 2.3 a proiectului CeCBiD-EOSC, privind serviciile și aplicațiile informatice dezvoltate și implementate în infrastructura Centrului de Resurse Cloud și Big Data (CCBD) în vederea satisfacerii cerințelor de calcul, de stocare și de prelucrare / analiză Big Data din faza operațională inițială a proiectului ELI-NP[9].

Impact

RST2.3 este un livrabil cheie al proiectului, al cărui scop pricipal este descrierea realizării platformei de procesare și analiză a datelor de monitorizare a infrastructurii Sistemului Laser de Mare Putere (*High-Power Laser System* – HPLS), destinată utilizării de către personalul operator în vederea detecției și preîntampinării funcționării anormale a acesteia.

Conținutul raportului

În Introducere se face o scurtă prezentare a rolului pe care il are proiectul în suportul computațional al activității de cercetare desfășurată în cadrul facilității ELI-NP și la analiza datelor de monitorizare a HPLS. Totodata se trec în revista tipurile de senzori utilizați la HPLS și fluxurile de date științifice și de monitorizare.

Cap. 2 descrie implementarea în cadrul CCBD a sistemului de arhivare pe bandă magnetică a datelor achiziționate la ELI-NP.

În Cap. 3 sunt sumarizate rezultatele obținute privind dezvoltarea serviciilor și aplicațiilor software destinate suportului Cloud al simularilor numerice și analizei datelor experimentale.

Metodele, algoritmii și codurile dezvoltate pentru prelucrarea și analiza fluxului de date de monitorizare a infrastructurii HPLS sunt prezentate în Cap. 4 și în Anexe. În prima parte a capitolului se descriu metodele dezvoltate pentru analiza imaginilor profilului fasciculului laser și a datelor de monitorizare a sistemelor de pompaj. Partea a doua este dedicată realizării platformei software pentru prelucrarea și analiza datelor de monitorizare a HPLS în regimul de producție științifică.

Capitolul 5 prezintă implementarea unei metode distribuite de antrenament și evaluare a modelelor de învățare automată, care urmează să fie efectuate, folosind fluxurile create pentru ingestia datelor, în același timp cu încărcarea de noi date în infrastructură.

Concluziile raportului

Subactivitatea 2.3 se încheie cu realizarea rezultatului planificat nr. 20 al proiectului - ”*Servicii și aplicatii informatice pentru satisfacerea cerintelor IT din faza operationala initiala a proiectului ELI-NP*”.

Totodata, prin programarea aplicației de analiză a datelor de monitorizare bazată pe microservicii care este descrisă în Cap. 4.3 și 5, Subactivitatea 2.3 a contribuit la îndeplinirea indicatorului suplimentar de realizare “*Aplicații dezvoltate folosind tehnici pentru infrastructuri masive de date (Big Data)*” din Cererea de Finanțare a proiectului.

Platforma software de analiză a datelor de monitorizare implementată in cadrul CCBD va îmbunătăți precizia de detecție și viteza de reacție a personalului operator privind eventualele anomalii din funcționarea HPLS, mărind totodată capabilitatea de predicție a acestora.

[9] Extreme Linght Infrastructure – Nuclear Physics, https://www.eli-np.ro/

1. Introducere

1.1 Context general și necesitate

Infrastructura laser și experimentele cu fascicule luminoase intense de la *Extreme Light Infrastructure – Nuclear Physics* (ELI-NP, https://www.eli-np.ro/) utilizează un număr considerabil de echipamente care măsoară diverși parametri de control și experimentali: spectrometre, energimetre și powermetre, camere video de mare viteză, etc. Acestea generează un flux apreciabil și heterogen de date (Big Data) ce necesită filtrare, procesare, arhivare și stocare. Se estimează ca doar cele 94 de camere CCD de înaltă rezoluție pot genera în regim normal de exploatare maxime de 5 Gb/s și aproximativ 145 TB de date pe zi.

Necesitatea utilizării unor metode automatizate și eficiente de procesare a datelor achiziționate de la instalații de cercetare complexe a devenit evidentă în ultimii ani, odată cu dezvoltarea experimentelor de la Marele Accelerator de Hadroni[10] - CERN, când s-a început implementarea metodelor de învățare automată (*machine learning* - ML) pentru aliniere[11], filtrare, reconstrucție[12], clasificare[13] (inclusiv de imagini[14]) și analiza[15] de date.

Metode numerice similare pot fi implementate sau dezvoltate în cazul ELI-NP pentru a reduce timpul de calcul și pentru a acoperi o parte din necesarul de analiza de date.

Detecția și clasificarea de imagini din fluxul video pot fi abordate într-o primă etapă utilizând biblioteci software de ML și *computer vision* de tip open source, cum sunt TensorFlow[16] si OpenCV (*Open Source Computer Vision*[17]). Pe baza acestora, se pot implementa algoritmi de prelucrare și clasificare automatizată de semnal care au avantajul că se pot adapta, cu ajutorul invățării supervizate și cu un minim de programare suplimentară, la o clasă largă de evenimente ce se doresc a fi detectate și clasificate. În plus, dacă este folosită o arhitectura hardware de tip GPU, acești algoritmi de tip rețea neuronală convolutivă[18] permit o reducere substanțială a timpului de calcul.

Actualizarea și extinderea infrastructurii Cloud și Big Data a DFCTI care se realizează în cadrul proiectului CeCBiD-EOSC este menită să contribuie, printre altele, la asigurarea suportului computațional al activității de cercetare desfășurată în cadrul facilității ELI-NP. Acesta privește atât activitatea de cercetare propriu zisă derulată de către grupurile experimentale, cât și activitatea de monitorizare și control a componentelor infrastructurii laser, care generează cantități mari de date pentru înregistrarea, prelucrarea și analiza cărora sunt necesare capacități dedicate de procesare și de stocare.

În prezent principalele componentele active ale facilității ELI-NP sunt: a) Sistemul Laser de Mare Putere (*High-Power Laser System* – HPLS), care include două brațe laser capabile să genereze pulsuri de 10 PetaWatt (PW) fiecare[19]; b) Sistemul de Transport al Fasciculului Laser (*Laser Beam Transport System* – LBTS), inaugurat în 2020[20]; c) zone/laboratoare experimentale, utilizate de echipele de cercetare.

---

[10] *The Large Hadron Collider*, https://home.cern/science/accelerators/large-hadron-collider
[11] G Azzopardi et al (2017), CERN-ACC-NOTE-2018-0010, http://cds.cern.ch/record/2305798?ln=en
[12] S Farrell et al (2017) https://dl4physicalsciences.github.io/files/nips_dlps_2017_28.pdf
[13] S Farrell et al, EPJ Web of Conferences 150, 00003 (2017)
[14] A Schwartzman et al, 2016 J. Phys.: Conf. Ser. 762 012035
[15] A Alves, 2017, J. of Instrumentation 12 (05) T05005
[16] M Abadi et al, Procs. of the 12th USENIX Symposium on Operating Systems Design (2016)
[17] I Culjak et al, IEEE 2012 Procs of the 35th International Convention MIPRO, 1725-1730
[18] C Fernández Madrazo et al, EPJ Web of Conferences 214, 06017 (2019)
[19] F Lureau et al, High Power Laser Science and Engineering, (2020), Vol. 8, e43, sectiunea 2;
[20] "*Inaugural 10 PW Laser and Users Symposium: Moving into Uncharted Territories*", https://www.eli-np.ro/2020-symposium/

În secțiunea următoare se trec în revistă subsistemele de monitorizare și control de la ELI-NP generatoare ale fluxurilor Big Data care constituie subiectul acestui livrabil.

1.1.1 Date furnizate de sisteme de monitorizare și control

Fasciculul laser generat de sursa primară (un oscilator Ti:Sa de 6 fs) este amplificat, lărgit și apoi comprimat în cadrul unuia din cele două *Front End*-uri[19] (FE) ale HPLS. De aici străbate de-a lungul fiecărui braț sisteme de amplificare succesive, denumite în Fig. 1.1 Amp 1 (care conține 8 laseri de pompaj SAGA ce contribuie la obținerea pulsurilor de 100 TeraWatt (TW) și frecvența de 10 Hz), Amp 2 (cuprinzând 6 pompe laser GAIA care contribuie, împreună cu Amp 1, la obținerea pulsurilor de 1PW și frecvența de 1 Hz) și, respectiv, Amp 3 (care conține 8 pompe laser ATLAS și contribuie, împreună cu Amp 1 și Amp 2, la obținerea pulsurilor de 10PW și frecvența de 1/60 Hz). Săgețile verticale din figură reprezintă ieșirile fasciculului laser amplificat către compresoarele temporale necesare pentru obținerea pulsurilor de 100TW, 1PW sau 10PW, de unde fasciculul este apoi direcționat către LBTS.

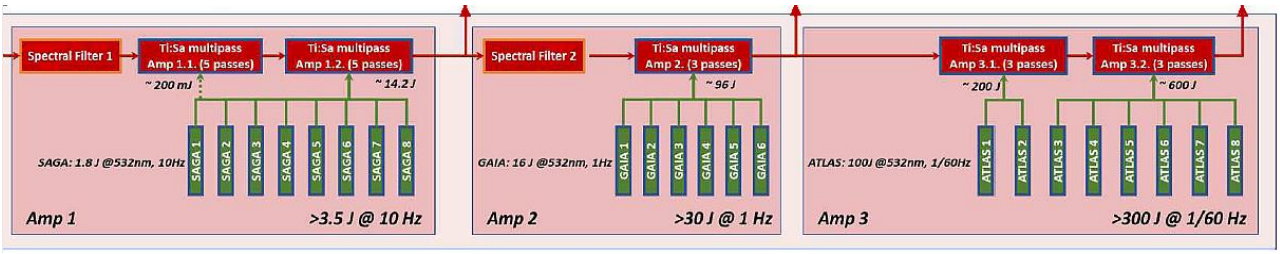


FIG. 1.1: Cele 3 sisteme de amplificare a puterii fasciculului laser situate pe un braț al HPLS[19]

Pentru o configurație dată (100TW, 1PW sau 10PW), fasciculul laser parcurge, în această ordine: a) unul dintre cele două FE-uri; b) amplificatoare de putere (Amp1 - în toate cele 3 configurații, Amp2 - în configurațiile de 1 și 10 PW, Amp3 - doar în configurația de 10PW); c) sisteme de compresie și sisteme de diagnosticare; d) LBTS; e) camere de interacție situate în zonele experimentale.[21]

Acuratețea rezultatelor experimentale depinde de calitatea fasciculului laser. În plus, o serie de componente ale HPLS și/sau LBTS pot fi deteriorate prin generarea unui fascicul laser de calitate necorespunzătoare. Astfel, scăderea semnificativă, în raport cu nivelul de referință, a lățimii de bandă spectrală poate duce la o creștere neprevăzută a puterii de vârf / densității de energie a pulsului, care să depășească pragul de distrugere (*laser induced damage threshold - LIDT*) pentru mai multe elemente optice. În mod similar, un profil spațial neuniform al fasciculului, în care apar mici zone ("puncte fierbinți") mult mai strălucitoare decât restul suprafeței de iluminare, poate provoca daune locale elementelor optice care sunt poziționate în și după punctul de producere a anomaliei respective.

Dintre elementele optice vulnerabile/consumabile se remarcă: a) oglinzi dure cu apertura de ordinul sutelor de mm din LBTS, destinate transportului în vid al pulsurilor laser de până la 10 PW de la compresoarele temporale către camerele de interacție; b) rețele de difracție cu dimensiuni de ordinul metrului din compresoarele temporale; c) oglinzi laser de dimensiuni mai mici, care operează în aer, situate în brațele de amplificare ale HPLS; oglinzi care direcționează fasciculele laser de pompaj; etc.

Design-ul sistemului laser a fost conceput astfel încât valoarile LIDT ale elementelor optice să fie suficient de mari față de densitatea de energie laser în regimul de lucru nominal, ceea ce face ca rata de defectare să fie foarte mică în condiții normale de funcționare. Cu toate acestea nu pot fi complet excluse incidentele nedorite, datorate de exemplu unor erori de operare și/sau de configurare a infrastructurii complexe, ceea ce duce în practică la o rată de defectare superioară celei teoretice.

[21] C Radier et al, High Power Laser Science and Engineering, (2022), Vol. 10, e21

Deterioriorarea componentelor optice are atât consecințe cu impact științific (prin întârzieri de ordinul săptămânilore ale programelor de cercetare), cât și financiare. Cheltuielile pentru achiziționarea unor elemente de tipul celor descrise la punctele a)-c) de mai sus sunt mari și există puține firme specializate în fabricarea echipamentelor de calitatea cerută (de exemplu, oglinzile pentru transportul in vid al pulsurilor laser cu durata de femtosecunde sunt produse prin tehnologii *ion-assisted deposition*, *electron beam deposition* sau *ion beam sputtering*).

Pentru a preveni producerea incidentelor descrise mai sus este necesară o supraveghere eficientă și detaliată a sistemului laser în timpul funcționării. În acest scop, pentru monitorizarea sistemului și verificarea calității fasciculului FE, componentele sistemelelor de amplificare (Amp1.1-2, Amp2, Amp3.1-2 din Fig. 1.1) și sistemele de diagnosticare ale fasciculului laser au atașate, printre altele, camere CCD, energimetre și spectrometre (denumite aici generic senzori). În plus, alte instrumente furnizează informații suplimentare privind starea diferitelor componente, poziția motoarelor, sistemul de sincronizare, frontul de unda, etc. În total sunt aproximativ 250 de snzori integrați într-un sistem de control de tip SCADA.

Informațiile furnizate de către senzori (imagini ale profilului spațial al fasciculului laser, energia/puterea, spectrul, etc.) sunt utilizate pentru verificarea/corectarea alinierii și detecterea eventualelor anomalii din sistem.

Rata de achiziție a datelor este de 10 Hz pentru majoritatea senzorilor (în FE și pe linia de 100TW), ceea ce conduce la aproximativ 500.000 de achiziții în 14 ore de funcționare zilnică. Frecvența de achiziție pentru dispozitivele montate pe linia de 1PW este de una pe secundă, rezultând mai mult de 50.000 de achiziții pentru aceeași perioadă de funcționare. Iar pentru linia de 10PW, care furnizează un impuls laser pe minut, pot fi obținute peste 800 de achiziții de date în cele 14 ore de fascicul. De la ansamblul dispozitivelor de monitorizare și control atașate echipamentelor din cele două brațe ale HPLS se acumulează într-o singurî zi de lucru 2x300 GB de date, adică aproximativ 150 TB / an.

Rata ridicată de acumulare de date impune prelucrarea lor automată, iar pentru detectarea și caracterizarea rapidă a anomaliilor din sistem se recomandă utilizarea metodelor de analiză specifice tehnologiei Big Data, bazate pe algoritmi *machine learning*.

1.1.2 Fluxul datelor științifice

Facilitatea ELI-NP este destinată cercetării de vârf, aici fiind prevăzută investigarea unei game largi de noi subiecte științifice, de la cele din domeniile fizicii fundamentale, fizicii nucleare și astrofizicii, până la aplicații în fizica materialelor, științele vieții și managementul materialelor nucleare.

Experimentele efectuate cu radiație laser, gamma, și cele cu radiație combinată laser-gamma necesită resurse de procesare, de stocare și de rețea atât pentru arhivarea și analiza datelor științifice achiziționate, cât și pentru simulări ale proceselor fizice de interes. Datele experimentale au un volum de ordinul PetaByte iar structura lor variază de la un experiment la altul, ceea ce reprezintă caracteristici definitorii ale Big Data.[22]

Analiza cerințelor preliminare de resurse de calcul științific ale experimentelor a demarat în 2015, iar rezultatele acesteia au fost sintetizate ulterior de către specialiști DFCTI într-un Raport de Design Tehnic (TDR)[23]. Design-ul arhitecturii sistemului IT s-a realizat conform recomandărilor dîn livrabilul D10 al proiectului H2020 ELITRANS[24] susținut de cei 3 parteneri (*pillars*) ai ELI[25].

[22] MO Cernaianu et al, Romanian Reports in Physics, Vol. 68, Supplement, P. S349–S443, 2016
[23] M Ciubancan et al, versiune preliminară "*ELI-NP Data and Computing Technical Design Report*", document intern ELI-NP.
[24] "*Enabling ELI's transformation from ERDF - funded distributed implementation towards ERIC - governed unified operation*", https://eli-trans.eu/
[25] G Beckett et al, *Report on ELI-wide Data Management Concept: Prediction of Expected Dataset Characteristics and Requirement Definition for an ELI-wide Data Management*, ELITRANS D10.1 Deliverable, 2017.

În proiectul de TDR s-a estimat că după intrarea în regim normal de producție științifică a HPLS datele primare și simulate vor necesita în primii doi ani de exploatare a facilității o capacitate de stocare totală de 3,5 PetaBytes (PB).

La actualizarea din 2018 a cerințelor grupurilor experimentale s-a considerat că, dacă se exclud experimentele cu radiație gamma, care vor începe după 2023, necesarul de capacitate de stocare din primii ani de exploatare va fi de 1,65 PB, iar numărul minim de nuclee de procesare (*CPU cores*) dedicate analizei de date va fi de 2.024. Aceste estimări nu includ cerințele de calcul de înaltă performanță (*High-Performance Computing* – HPC) pentru simulări, care sunt de ordinul 20.000 *cores*.

Acumularea masivă de date științifice necesită stocarea acestora pe termen mediu pe disc, după care datele de interes trebue să fie arhivate pe termen lung alături de datele experimentale. În livrabilul D10 al ELITRANS s-a recomandat ca fiecare dintre cei 3 parteneri ELI să își asigure într-o Facilitate Locală (*ELI Local Facility* – ELF) atât stocarea pe termen scurt până la lung a datelor experimentale, cât și calculul științific pe clustere dedicate.

În acord cu aceste recomandări, arhitectura ELF de la ELI-NP, care a fost propusă în proiectul de TDR și în D10, cuprinde componentele reprezentate pe fond verde în Fig. 1.2.

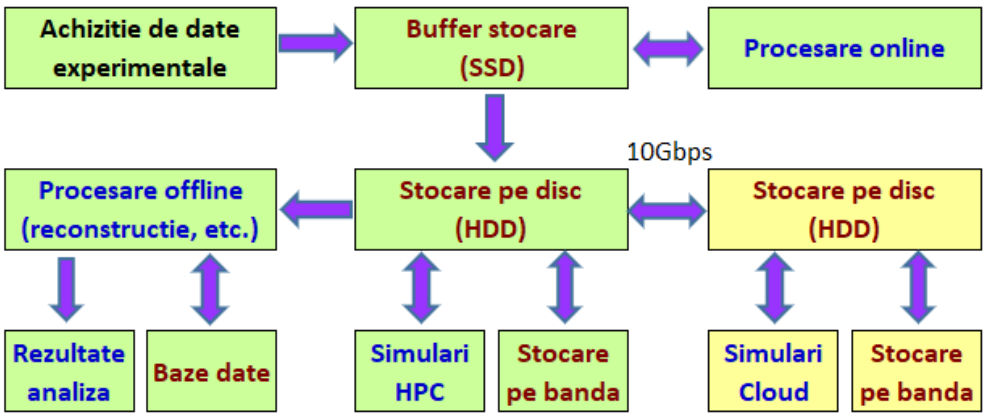


FIG. 1.2: Fluxul de date științifice în și între ELF și infrastructura CLOUDIFIN

Deoarece la data redactării propunerii CECBID-EOSC IFIN-HH nu dispunea de infrastructura IT necesară pentru ELF, iar în finanțarea facilității ELI-NP nu erau prevăzute fonduri suficiente dedicate realizării suportului de calcul avansat, în propunerea de proiect a fost inclus obiectivul de alocare către ELI-NP a unei părți din echipamentele achiziționate pentru stocarea de date rezultate din simulări și a copiilor de siguranță a datelor experimentale brute, precum și pentru calculul Cloud. Această inițiativă își demonstrează în prezent utilitatea întrucât, de exemplu, la data redactării acestui livrabil capacitatea CPU a ELF este doar de ordinul sutelor de *cores*, față de ~2.000 cât a fost estimat în propunerea de TDR pe baza cerințelor experimentelor.

Conform livrabilului D10[24], în ELF datele sunt stocate pe disc pe termen de maxim 2 ani și utilizate pentru analiza, reconstructia și simularea evenimentelor, reprocesare, etc. După finalizarea analizei de către grupurile experimentale, datele procesate sunt copiate pe bandă magnetică împreună cu datele brute, pentru arhivarea pe termen lung. Politica de utilizare a datelor arhivate impune ca acestea să fie accesibile local și la distanță pentru utilizatorii autorizați, în vederea efectuării unor analize ulterioare.

Contribuția Centrului Cloud și Big Data (CCBD) la suportului ELI-NP (reprezentată pe fond galben în Fig. 1.2) este o extensie a ELF ce cuprinde capacități suplimentare de stocare pe disc / bandă magnetică și sisteme de procesare în Cloud, destinate completării necesarului de resurse IT al experimentelor. Resursele respective sunt conectate la SCF prin itermediul unei conexiuni de fibră optică cu lațimea de bandă de 10 Gbps upgradabilă în funcție de cerințe.

1.2 Obiectivele Subactivității 2.3

Conform Cererii de Finanțare[8], unul dintre scopurile principale ale proiectului este *dezvoltarea și implementarea de servicii și aplicații informatice pentru administrarea și funcționarea centrului de resurse, care utilizează tehnologii Cloud și Big Data pentru satisfacerea cerințelor IT din faza operațională a proiectului ELI-NP*.

Pentru realizarea acestui obiectiv, în cadrul Subactivității 2.3 s-au propus urmatoarele:

A) Contribuții la asigurarea și la managementul resurselor Big Data necesare grupurilor experimentale, prin

1. Implementarea unui sistem de stocare pe bandă magnetică pe termen lung a datelor primare și simulate generate de experimentele din cadrul ELI-NP.

În cadrul acestei activități s-a preconizat, printre altele, instalarea pe infrastructura de stocare a centrului CLOUDIFIN a unei soluții software care sa permita transferul automat al fișierelor între resursele de stocare pe disc și, respectiv, pe bandă magnetică.

B) Punerea la dispoziția utilizatorilor, în cadrul infrastructurii Cloud, a unor instrumente software adecvate investigării tematicilor de cercetare abordate la ELI-NP, prin

2. Implementarea unei metode de accesare a aplicațiilor software utilizate în cadrul organizatiei virtuale (VO) eli-np.eu, cuprinzând:

a) generarea unor imagini de mașini virtuale (VM) care conțin software-ul de interes (cum sunt aplicatiile *epoch*, *root*, *geant*, *namd*, etc.);

b) dezvoltarea unui sistem software de distribuire a imaginilor de VM-uri în centrele Cloud care suportă organizatia virtuala *eli-np.eu*.

3. Instalarea și optimizarea mediului / bibliotecilor software necesare pentru utilizarea metodelor de învațare automată (ML - machine learning), inclusiv a suportului GPGPU.

Setul minimal de componente software care a fost prevăzut inițial include *Python 3.4+*, *pip*, *NumPy*, *SciPy*, *Virtualenv*, *Docker*, *CUDA Toolkit*, *cuDNN SDK*, etc.

4. Instalarea unor biblioteci de învățare automată și computer vision, cum sunt *Tensorflow*, *Keras*, *OpenCV*, *scikit-learn*, *LIBSVM*, *SVMlight*, etc.

C) Dezvoltarea de aplicații software pentru analiza fluxului de date generat de senzorii / dispozitivele SCADA ale infrastructurii laserului de mare putere de la ELI-NP, utilizând metode de învățare automată. Acest obiectiv cuprinde:

5. *Optimizarea interfeței de decodare a datelor pentru a fi procesate de algoritmii de ML*.

Datele brute de monitorizare (imagini, valori parametri, semnale) obținute de la camerele video, energimetre sau de la diverși alți senzori trebuie să fie preprocesate pentru a asigura consistența și calitatea necesare pentru utilizarea algoritmilor ML.

6. *Dezvoltarea de algoritmi de tip ML* pe bază de învațare automată sau semi-automată *pentru detectarea semnalelor de interes* în funcție de tiparele ascunse (*patterns*) și trăsăturile principale (*features*) din fluxul de date.

7. *Optimizarea arhitecturii algoritmilor de ML pentru creșterea vitezei de răspuns și a preciziei de detecție*.

2. Stocarea pe bandă magnetică a datelor de la ELI-NP

2.1 Necesitate

Sistemele actuale de arhivare pe bandă magnetică oferă o serie de avantaje față de cele care utilizează discuri mecanice: au un consum redus de energie electrică (deoarece nu este necesar pentru păstrarea datelor); benzile sunt mai fiabile și au timp de viață de ordinul decadelor; necesită capacități minime de climatizare; costul per TB stocat este mai mic; viteza de extragere a datelor este mai mare.

Găzduirea și operarea în CCBD a unei facilități de stocare pe bandă magnetică a datelor de la ELI-NP are motivații multiple, dintre care amintim:

- **Siguranța datelor**

  Replicarea în siguranță pe termen lung a datelor primare și simulate unice care provin de la experimentele efectuate la ELI-NP este obligatorie deoarece acestea constituie fundamentul și justificarea rezultatelor științifice publicate.

- **Replicarea datelor la distanță**

  Reglementările standard pentru arhivarea datelor pe termen lung impun ca echipamentul de stocare pe bandă să fie găzduit într-o locație diferită de cea a unității de stocare primară.

- **Avantajele replicării într-un cloud propriu**

  Arhivarea într-un cloud extern ar adauga costuri suplimentare și poate pune probleme privind accesibilitatea și controlul asupra datelor.

2.2 Implementarea sistemului de arhivare a datelor

Investigațiile preliminare efectuate la DFCTI[26] privind viteza de scriere și de citire pe bandă a fișierelor în regim de arhivare a Big Data specifice ELI-NP au evidențiat avantajele utilizării sistemului de fișiere LTFS (*Linear Tape File System*), care permite atingerea unor performanțe apropiate de cele realizate prin utilizarea discurilor mecanice. Rezultatele acestor teste și prospectarea pieței au determinat adoptarea in cadrul CCBD a unei soluții de arhivare care să utilizeze LTFS.

Pentru implementarea soluției de stocare a datelor pe termen lung s-au achizitionat prin procedură de licitație publică și s-au instalat o unitate de benzi magnetice Dell EMC ML3 cu suport LTFS și un server de acces și management DellEMC PowerEdge R650.

Serverul este echipat cu două controllere tip SAS HBA (*Host Bus Adapter*) compatibile cu unitatea de bandă și conectat printr-o legatură de minim 10Gbps full-duplex la infrastructura de comunicare de date a centrului.

Biblioteca de benzi are capacitatea de accesare a 32 de *cartridge*-uri minim LTO8 însumand 384 TB online, cu posibilitate de extindere cu 280 de sloturi prin module de expansiune până la minim 3.360 TB.

Sistemul a fost livrat cu 2 unități LTO Ultrium 8 și 5 cartuse de date LTO-8 ce însumează 60 TB brut (*raw*). Această capacitate poate fi ulterior mărită în funcție de cerințe și de resursele de finanțare.

[26] T Ivanoaica, M Ciubancan et al, "*Exploring long-term tape-storage solutions*," *IEEE EUROCON 2017 - 17th International Conference on Smart Technologies*, Ohrid, Macedonia, 2017, pp. 949-952.

3. Servicii și aplicații software pentru cercetare

3.1 Mașini virtuale pentru simulări numerice și analiza de date

Pentru suportul computational al activitatii desfasurate de utilizatorii din cadrul comunitatii de cercetare *eli-np.eu* au fost programate imagini de masini virtuale (virtual machines - VM) care includ aplicatii si biblioteci software necesare acestora.

Prima imagine de VM ofera utilizatorilor software machine learning (platforma TensorFlow, interfata de programare a aplicatiilor Keras) si computer vision (OpenCV), utile pentru prelucrarea si analiza datelor de monitorizare furnizate de senzorii infrastructurii laserului de mare putere de la ELI-NP.

Cea de-a doua imagine de VM contine aplicatiile EPOCH (pentru simularea numerica a interactiei laser cu materia prin metoda Particle-in-Cell), NAMD (dinamica moleculara) si SIESTA (pentru simularea proprietatilor materialelor prin metoda functionalei de densitate – DFT.

3.2 Distribuirea imaginilor de VM-uri în centre Cloud multiple

Pentru distribuirea imaginilor de masini virtuale care contin software-ul de interes s-a implementat un sistem (modul python) în interfata web. Imaginile de masini virtuale sunt stocate pe un server ftp din cadrul centrului de resurse principal (baza de date a aplicatiilor).

Sistemul de distributie implementat în interfata web permite utilizatorilor sa copieze aceste imagini pe centrul cloud propriu iar apoi sa le importe cu ajutorul serviciului glance (OpenStack) in site-ul cloud al organizatiei din care fac parte.

CLOUDIFIN - Manage Images

Image Name	Description	Project	Size	Action
CentOS7.9_ML_tensorflow_keras_opencv.qcow2	Image with ML Software	eli-np.eu	4.1GB	Get
CentOS7.9_epoch_root_geant_namd.qcow2	Image with Siesta, Epoch, Namd Software	eli-np.eu	4.6GB	Get

FIG. 3.1: Interfata web a imaginilor de masini virtuale

Copierea imaginii masinii virtuale de catre utilizator se face prin actionareaa butonului "Get" din interfata de management a imaginilor de VM-uri (Fig. 3.1).

Pentru copierea imaginii VM pe controller-ul cloud al site-ului partener din SCF, dupa apasarea butonului "Get" se initializeaza un mesaj mqtt de catre interfata web a ICM, iar agentul de pe controller-ul centrului cloud partener receptioneaza acest mesaj si executa comanda "wget" pentru descarcarea imaginii respective.

3.3 Testarea aplicațiilor de simulare a interactiei laser-materie

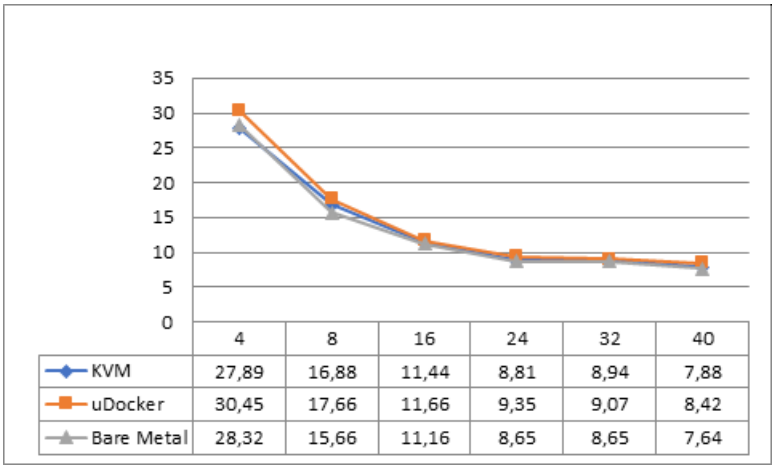
Performantele imaginilor de masini virtuale descrise in cap. 2 au fost testate in cadrul clusterului CLOUDIFIN. Spre exemplificare reproducem mai jos testele de scalabilitate ale programului EPOCH, paralelizat prin MPI.

Nodul de calcul utilizat a avut sistem de operare Linux Ubuntu 20.04, doua procesoare AMD Epyc a cate 24 de nuclee de procesare (core) fiecare si 3.2 GB RAM/core.

Rularea exemplului s-a facut intr-o prima etapa pe o singura masina virtuala, pe seturi de 4, 8, 16, 32, 40 de core iar scrierea datelor de iesire s-a facut local.

S-au utilizat OpenMPI 4.1.1, EPOCH 4.17.16, iar pentru executia containerelor s-a folosit uDocker 1.3.1, care nu necesita privilegii de root.

Testele aplicatiei EPOCH au fost rulate utilizand fisierul input.deck disponibil la adresa web https://github.com/numericalalgorithmsgroup/performance_profiling_examples

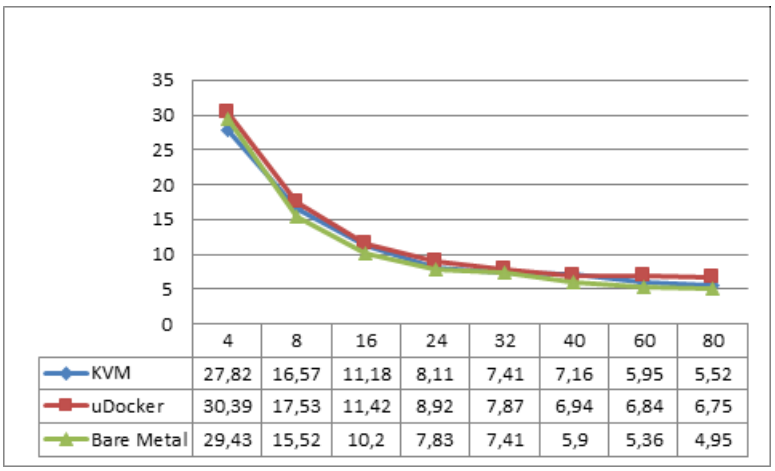


	4	8	16	24	32	40
KVM	27,89	16,88	11,44	8,81	8,94	7,88
uDocker	30,45	17,66	11,66	9,35	9,07	8,42
Bare Metal	28,32	15,66	11,16	8,65	8,65	7,64

FIG. 3.2: Teste de scalabilitate pe un server fizic si masina virtuala

In Fig. 3.2 s-a reprezentat pe abscisa numarul de nuclee de procesare utilizate, iar pe ordonata timpul de rulare (in secunde).

In a doua etapa a studiului s-a implementat solutia MPI dezvoltata de DFCTI in cadrul proiectului EGI-ACE, pentru a testa aplicatia pe un sistem de doua masini virtuale, datele fiind scrise prin intermediul NFS. Testele de scalabilitate pentru acest caz sunt reproduse in Fig. 3.3.



	4	8	16	24	32	40	60	80
KVM	27,82	16,57	11,18	8,11	7,41	7,16	5,95	5,52
uDocker	30,39	17,53	11,42	8,92	7,87	6,94	6,84	6,75
Bare Metal	29,43	15,52	10,2	7,83	7,41	5,9	5,36	4,95

FIG. 3.3: Teste de scalabilitate pe doua server fizice si masini virtuale

Rezultatele testelor, prezentate in figurile de mai sus, reflecta o buna scalabilitate atat pe servere fizice, cat si pentru solutiile de virtualizare KVM (Kernel-based Virtual Machine) si, respectiv, uDocker.

Datorita cresterii cantitatii datelor transferate in urma virtualizarii, timpul necesar rularii testului creste cand se trece de la serverul fizic la KVM si apoi la uDocker, dar diferenta nu este mai mare de 10%, ceea ce demonstreza utilitatea folosirii aplicatiei in cloud.

4. Prelucrarea și analiza fluxului de date de monitorizare

4.1 Obiectiv și strategie

Subactivitatea 2.3 prevede, printre altele, dezvoltarea de algoritmi pentru detectarea rapidă a semnalelor de interes din fluxul de date provenite de la HPLS, care necesită utilizarea unor metode de analiză specifice tehnologiei Big Data, bazate pe algoritmi de învățare (semi) automata.

Datele de monitorizare achiziționate de la senzorii descriși în Cap. 1.1.1 sunt stocate de către sistemul de control al HPLS în fișiere HDF5 (*Hierarchical Data Format version 5*). Ele trebuie să parcurgă mai întâi o etapă de preprocesare pentru a asigura coerența și calitatea necesară utilizării algoritmilor de învățare automată. În cazul imaginilor, de exemplu, aceasta poate implica acțiuni cum sunt normalizarea, redimensionarea sau eliminarea zgomotului.

Etapa următoare constă în extragerea caracteristicile relevante din datele preprocesate. În exemplul anterior, al imaginilor furnizate de camerele CCD, acest lucru se poate realiza aplicând metode utilizate în *computer vision* sau prin folosirea tehnicilor de *deep learning*, cum sunt rețelele neuronale convoluționale (CNN).

În prima parte a subactivității 2.3, prezentată în Cap. 4.2, s-au propus și testat metode de preprocesare și analiză a unui set de date de monitorizare achiziționate la HPLS în perioada premergătoare începerii experimentelor de funcționare (prima jumătate a anului 2021).

Capitolul 4.3 este dedicat descrierii metodelor dezvoltate pentru analiza datelor de monitorizare colectate de la senzorii HPLS în timpul primelor experimente de funcționare (2022).

Menționăm că pentru "curățarea" datelor de la HPLS s-a propus anterior o metodă alternativă.[27]

4.2 Analiza datelor în regimul de testare a HPLS

Au fost investigate semnalele înregistrate de o cameră CCD și energimetre în cadrul testelor de fascicul făcute de echipa Departamentului Sisteme Laser (*Lasers System Department* - LSD) din ELI-NP.

Datele colectate[28] au fost stocate in format HDF5, iar setul analizat în acest studiu a constat din 380.676 de imagini monocrome de 102x128 pixeli (ale caror intensitati sunt reprezentate prin valori intregi intre 0 si 4092), precum si datele inregistrate de energimetre.

4.2.1 Analiza imaginilor furnizate de camerele CCD

Unul din aspectele de bază al experimentelor desfăşurate în cadrul proiectului ELI-NP este necesitatea de a caracteriza forma şi de a determina calitatea profilului fasciculului laser, care poate fi influențată de o mulțime de factori, ca de exemplu dezalinieri de-a lungul traseului optic.

În acest scop, într-o primă abordare au fost analizate succesiunile secvențelor de semnale CCD, iar imaginile individuale au fost fitate pe baza unui model descris mai jos și reprezentate într-un spațiu cu dimensionalitate redusă (15 si respectiv 2 dimensiuni față de cele 13.056 date de intensitățile individuale ale pixelilor).

A) Evoluția profilului pulsului laser

In Fig. 4.1(a) este reprezentata suma intensitatilor pixelilor pentru primele 25.000 de imagini.

[27] G Kolliopoulos, G Prodan, B Boisdeffre, I Dâncuș, Romanian Reports in Physics 72, 409 (2020)
[28] Datele colectate sunt proprietatea © *Lasers System Department*, ELI-NP, IFIN-HH

După cum se observă comparând cu Fig. 4.1(b), semnalul sumei intensităților pixelilor este aparent in buna corelatie cu energiile colectate de energimetru, dar în Fig. 4.1(a) se pot decela mai bine fluctuatiile de intensitate care nu sunt vizibile in valorile energiilor din Fig. 4.1(b). Acesta este si motivul pentru care in analiza ulterioara s-a preferat folosirea valorilor de intensitate in locul celor citite de energimetru.

Formarea imaginii pulsului evolueaza gradual in primele sute de pasi de timp, dupa cum se constata in Fig. 4.2 (a) si (b).

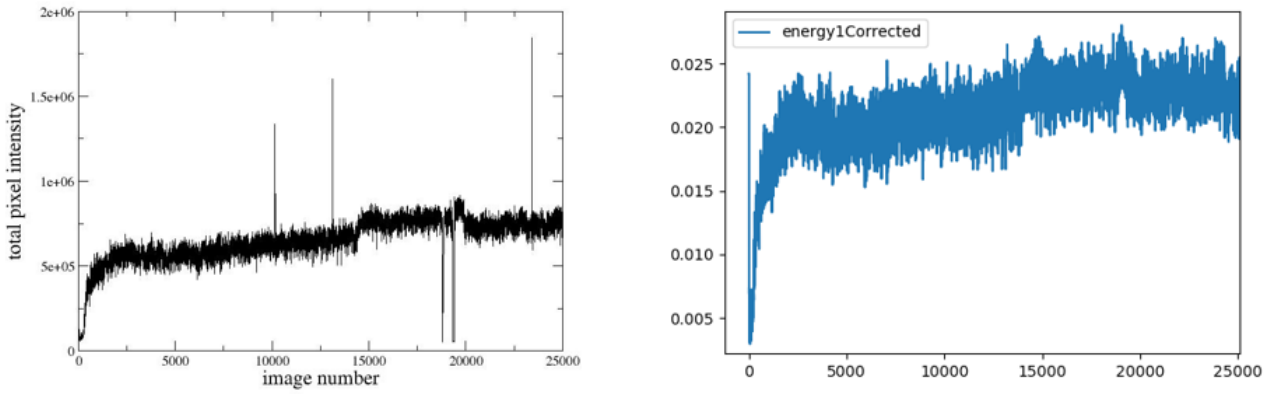


FIG. 4.1: (a) Suma intensităților și (b) energiile colectate pentru aceeași secvență de semnale

Ulterior conturul imaginii pulsului devine relativ constant, similar cu cel din in Fig. 4.2(c), însă intensitatea pixelilor variaza spațial.

In Fig. 4.2 se pot vedea si pixeli care dau semnale false (par sa fie "aprinsi" continuu chiar in lipsa unui fascicul luminos) si care trebuie reetalonati in imagini pentru a elimina aceasta sursa de zgomot.

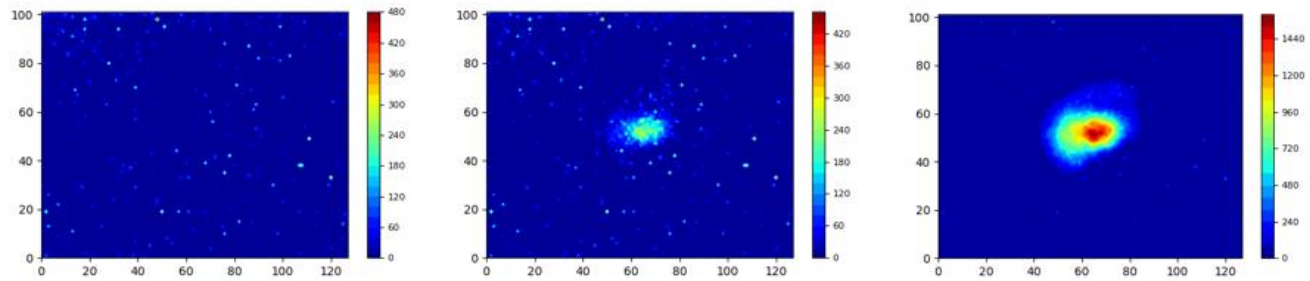


FIG. 4.2: Imaginea la momentele (a) 0, (b) 200 si (c) 2000 asociate secventei din Fig. 4.1.

B) Reducerea dimensionalității

Pentru a putea analiza eficient caracteristicile fasciculului laser (reprezentat in acest caz de proiectia lui pe senzorul CCD si imaginile captate astfel) este nevoie de o reducere a dimensionalitatii imaginilor – de la cele 13.056 de dimensiuni date de cei 102x128 pixeli.

Metoda propusă în acest studiu este reprezentarea imaginii fasciculului ca o suprapunere de trei Gaussiene bidimensionale

(4.1) $$f(x,y) = \sum_{i=1}^{3} A_i \, exp\left(-\left(\frac{(x-x_i)^2}{(2\sigma_{xi}^2)} + \frac{(y-y_i)^2}{(2\sigma_{yi}^2)}\right)\right)$$

unde se urmărește optimizarea urmatorilor 15 parametri:

- amplitudinile (Ai),
- coordonatele centrelor Gaussienelor pe cele doua axe, x_i si y_i,
- deviatiile pe axele x si y, σ_x si σ_y (adica latimea lor).

Pentru optimizare a fost folosită metoda celor mai mici patrate, mai exact algoritmul Levenberg–Marquardt[29] (LM), implementat în biblioteca SciPy[30].

In Fig. 4.3(a) se poate observa rezultatul interpolarii cu acest model pentru valorile de intensitate inregistrate in imaginea (b).

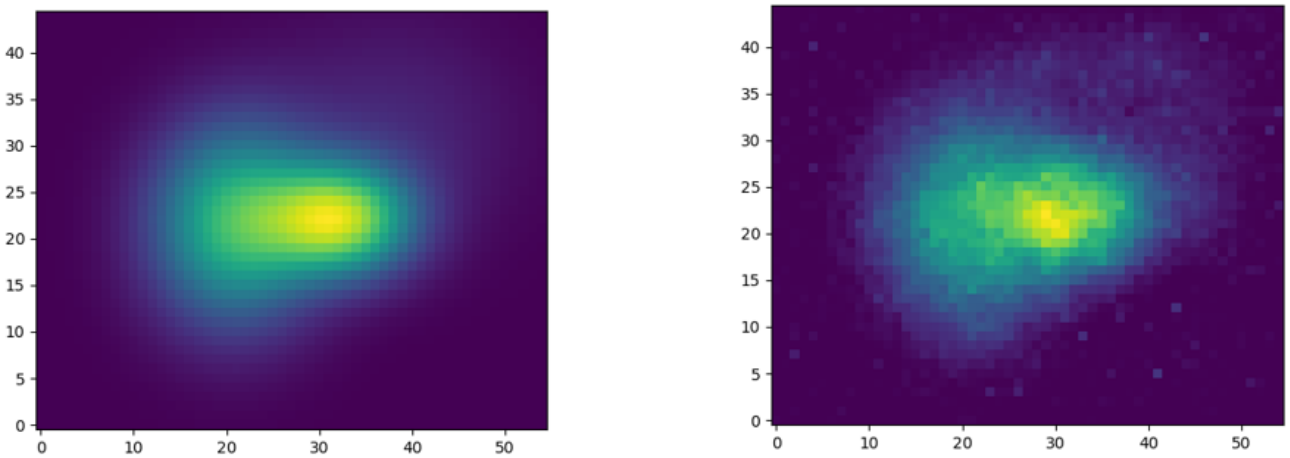


FIG. 4.3: (a) Rezultatul interpolarii cu trei Gaussiene 2D a imaginii din dreapta (b)

Folosind acest model de interpolare, au fost reprezentate intr-un spatiu 15-dimensional sase seturi, fiecare a cate 2000 de imagini, alese aleatoriu din tot setul de date. Ordinea cronologica a seturilor este pastrata dar acestea sunt separate unele de altele de alte zeci de mii de pasi care sunt omiși pentru a diminua efortul numeric.

De asemenea, în cadrul fiecarui set este pastrata succesiunea si ordinea imaginilor inregistrate in timpul experimentului.

Ulterior, pentru a putea vizualiza datele, a fost folosita si o procedura de analiza a componentelor principale (*Principal Component Analysis* - PCA) intr-un nou spatiu 2D. Procedura PCA se bazeaz[pe selectarea unor directii ortogonale dintr-un spatiu multi-dimensional care sa maximizeze varianta datelor proiectate.

Pentru aplicarea procedurii s-a utilizat implementarea numerica a PCA din biblioteca *scikit-learn*.[31]

In Fig. 4.4(a) sunt reprezentate prin culori diferite cele sase seturi de date in noua reprezentare a primelor doua componente principale, PC1 si PC2. Se observa ca seturile de date raman grupate, in special de-a lungul axei orizontale PC1.

Analiza vizuala partiala (adica a catorva dintre imaginile cu profilul fasciculului selectate din cele sase seturi) nu permite observarea unor particularitati clare, usor detectabile, si este necesar sa se analizeze trasaturile caracteristice pentru fiecare grup.

In Fig. 4.4(b) este reprezentata intensitatea insumata a pixelilor fiecarei imagini, ca în Fig. 4.1, dar nu există corelații vizibile ale acestui parametru cu modul in care se face diferentierea din analiza PCA, in special pe PC1, deoarece datele din toate seturile sunt distribuite in aproximativ acelasi interval energetic (sau, conform Fig. 4(b), aceeasi intensitate totala).

In Fig. 4.4 (a) si (b) se poate observa insa ca pulsul cu intensitatea maximă estimata din suma valorilor pixelilor este usor decelabil in ambele reprezentari, fiind proiectat la distanta considerabila pe PC2. Imaginea corespunzatoare (#66541038) este cea din Fig. 4.5, in centrul careia sunt inregistrate intensitati ale pixelilor care ating valoarea maxima masurata, adica 4092.

[29] K Levenberg, Quarterly of Applied Mathematics 2, 164-168 (1944)
[30] Biblioteca de algoritmi Python, https://scipy.org
[31] *Machine Learning in Python*, https://scikit-learn.org

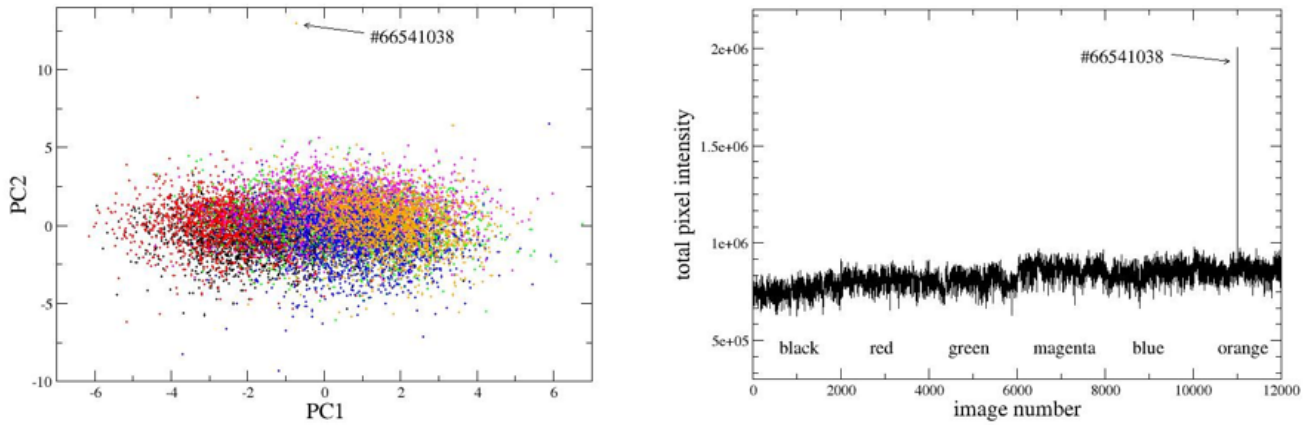


FIG. 4.4: (a) Reprezentarea a 6x2000 de imagini; (b) intensitatea totala a pixelilor din (a).

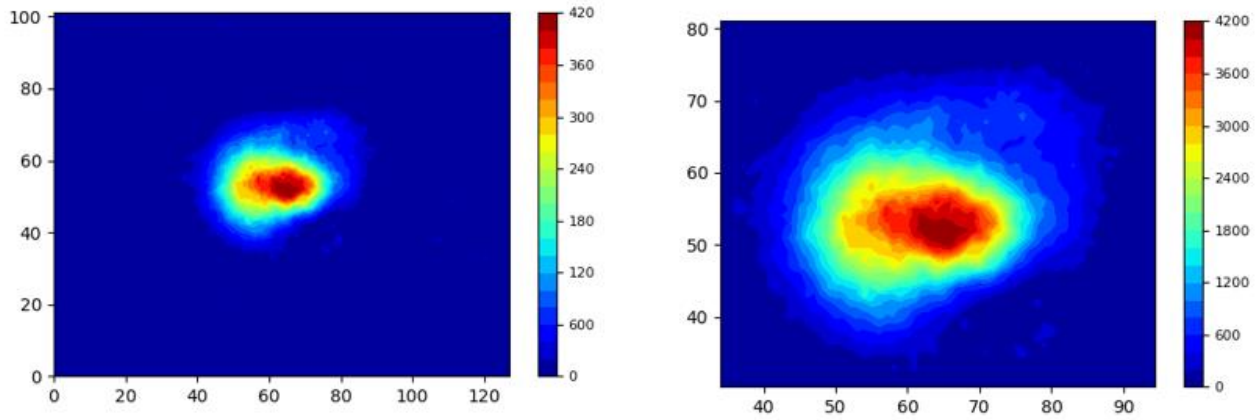


FIG. 4.5: Imaginea #66541038, corespunzatoare pulsului usor decelabil din Fig. 4.4

În concluzie, pentru analiza datelor captate de senzorul CCD s-a propus în această secțiune o metodă de interpolare a acestora cu un model de trei gaussiene bidimensionale. Parametrii modelului au fost folositi pentru a efectua o prima reducere a dimensionalitatii.

Pentru a putea vizualiza datele, a fost folosita o procedura de extragere a componentelor principale, reducand spatiul datelor la cele mai importante doua dimensiuni.

Proiectat pe primele două componente principale, setul de date relevă o corelație temporală, vizibilă în gruparea spațială a celor șase seturi de date reprezentate în Fig. 4.4(a).

C) Eliminarea zgomotului din imaginile CCD

Pentru îmbunătățirea calității imaginilor înregistrate de senzorul CCD a fost dezvoltat un algoritm de eliminare a zgomotului pe baza unui profil statistic asociat fiecărui pixel.

Algoritmul a fost testat pe un subset al datelor de monitorizare colectate în regimul de preproducție al HPLS și pus la dispoziție de grupul LSD de la ELI-NP.

Concret, a fost selectat un set de 50 de imagini monocrome cu rezolutia 102x128 capturate în absența fasciculului laser (un exemplu caracteristic poate fi observat în Fig. 4.6) pentru analiza zgomotului provenit de la fluctuații termice combinat cu parametrii contructivi specifici fiecarui pixel.

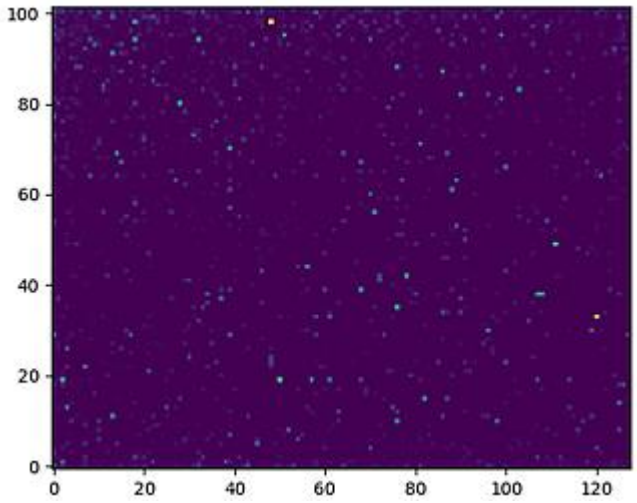
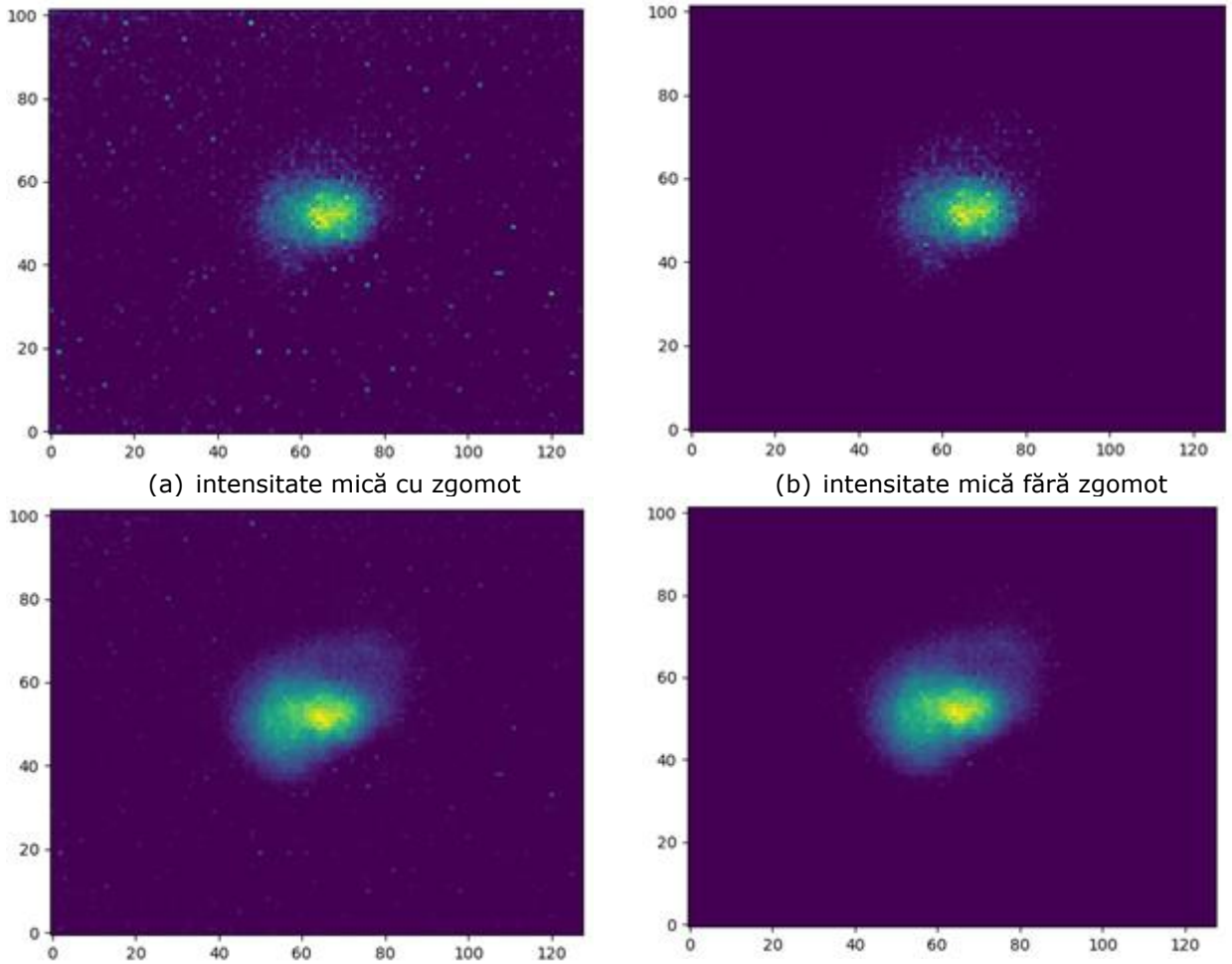


FIG. 4.6: Captură de zgomot în absența fasciculului

Pentru a modela statistica zgomotului per pixel, au fost fitate distribuții 1D normale (gaussiene) pe cele 50 de valori înregistrate. În urma fit-urilor au fost extrași parametrii caracteristici (valori medii și deviații standard, reprezentați în Fig. 4.8(d).



(a) intensitate mică cu zgomot (b) intensitate mică fără zgomot

(c) intensitate mare cu zgomot (d) intensitate mare fără zgomot

FIG. 4.7: Efectul reducerii zgomotului de fundal la diferite intensități ale fasciculului laser

Procedura de eliminare a zgomotului a constat în impunerea unui prag statistic de 10^{-6} pentru fiecare valoare înregistrată: dacă valoarea intensității pixelului se încadrează în distribuția de

zgomot cu probabilitate mai mare de 0.0001%, se consideră că acesta înregistrează doar zgomot de fond iar valoarea lui este redusă la zero. Alegerea unui prag mai mic/mare duce la reducerea mai accentuată/redusă a zgomotului.

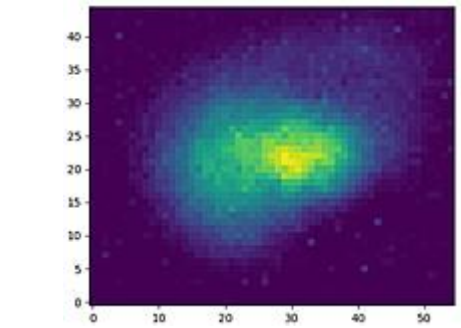
În Fig. 4.7 se reproduc imagini ale profilului laser înainte și după eliminarea zgomotului. În Fig. 4.7(c,d) intensitatea maximă înregistrată este de peste trei ori mai mare decât în Fig. 4.7(a,b) ceea ce duce la rescalarea automată a profilului cromatic și, implicit, la reducerea vizibilității zgomotului în imagine.

În continuare a fost comparată calitatea fitării analizând eroarea pătratică medie normalizată, înainte și după reducerea zgomotului, în modelul de fitare al celor trei Gaussiene bidimensionale prezentat mai sus.

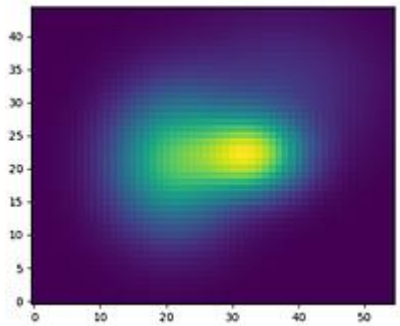
Procedura de fitare a fost aplicată doar pe o secțiune centrală de 45x55 de pixeli din cei 102x128, pentru reducerea timpului de calcul. Din acest motiv, efectul reducerii zgomotului este diminuat față de fitul realizabil pe întreaga imagine.

Calitatea modelului a fost evaluată pe baza distribuției rădăcinii pătrate a deviațiilor pătratice medii normate obținută din fitul pe 10.000 de imagini. Normarea erorilor s-a realizat prin împărțirea acestora la diferența dintre intensitatea maximă și cea minimă înregistrată pe probă.

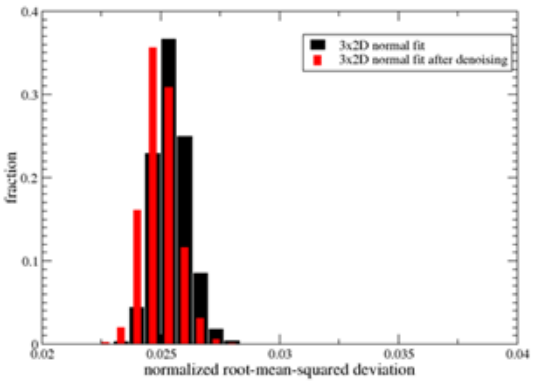
În Fig. 4.8 sunt reprezentate: (a) secțiunea centrală a imaginii care conține profilul fasciculului; (b) fit-ul obținut cu modelul de trei gaussiene 2D; (c) distribuția erorilor normalizate calculată pentru 10.000 de imagini, înainte (negru) și după eliminarea zgomotului (roșu); (d) parametrii distribuțiilor normale obținuți prin fit pentru fiecare pixel.



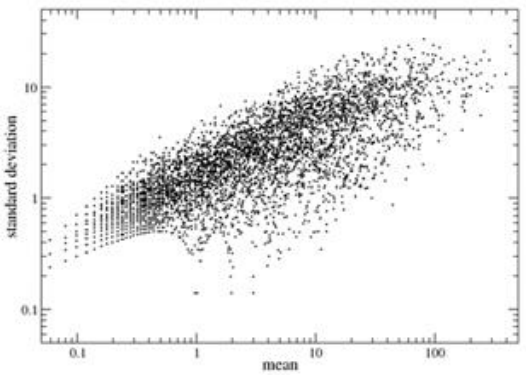
(a) Secțiunea centrală în imaginea inițială



(b) După modelarea prin cele 3 gaussiene



(c) Distribuțiile erorilor normalizate



(d) Parametrii distribuțiilor normale

FIG. 4.8: Evaluarea calității modelului

Se poate observa că eliminarea zgomotului determină o ușoară diminuare a erorilor de fit, ceea ce confirmă valabilitatea metodei și recomandă optimizarea și utilizarea ei ulterioară în etapa de preprocesare a datelor colectate la HPLS în regim de producție.

În concluzie, algoritmul propus de eliminare a zgomotului pe baza profilului statistic asociat pixelilor contribuie la îmbunătățirea calității imaginilor înregistrate de senzorii CCD din HPLS și crește calitatea de fitare prin modelul cu trei Gaussiene 2D.

D) Caracterizarea profilului fasciculului laser

Metodele de caraterizare bazate pe fitarea distribuţiilor luminoase din fiecare imagine cu cele obținute prin diferite modele plauzibile din punct de vedere fizic (ca cel din Secțiunea B) au dezavantajul restricționării analizei datelor doar la aspectele care pot fi reproduse în modelele respective, ignorându-se orice alte trăsături relevante. Din acest motiv, o abordare mai puţin restrictivă ar fi compararea directă a semnalelor captate de camerele CCD și măsurarea similarității dintre diferite distribuții ale intensității luminoase din fluxul de imagini.

Pentru aceasta s-a dezvoltat și implementat o nouă metodă de caracterizare a profilului fasciculului laser în vederea detectării corelațiilor între imaginile din flux, utilizând o distanță statistică corespunzătoare între distribuții.

În literatura de specialitate există numeroare exemple de distanţe statistice (care nu sunt neapărat metrice din punct de vedere matematic), cum ar fi: distanţa variaţională totală, Kullback–Leibler, Jensen–Shannon, Bhattacharyya, Mahalanobis sau Kolmogorov–Smirnov.

În acest studiu s-a optat pentru metrica (Kantorovich -) Wasserstein, care reprezintă intuitiv ”costul” necesar pentru a transforma o distribuţie în alta prin mutarea unei cantităţi (de probabilitate) pe o anumită distanţa.

Distanţa Wasserstein W_p cu moment *p* este definită matematic în Ec. 4.2, unde *μ* şi *ν* sunt două distribuţii de probabilitate cu momente finite de ordin *p* definite pe un spatiu euclidian *d*-dimensional cu normă || ||, iar Π(*μ*, *ν*) sunt toate distribuţiile definite pe $R^d \times R^d$ ale căror distribuţii marginale sunt date de *μ* şi *ν*.

(4.2)
$$\mathbf{W}_p^p(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^d\times\mathbb{R}^d} \|x-y\|^p \,\mathrm{d}\pi(x,y)$$

Aflarea distanţei Wasserstein (DW) constă în determinarea unei distributii comune (*joint distribution*) care minimizează costul de transport între două distribuţii univariate.

DW este uşor de calculat pentru distribuţii univariate, dar devine netratabilă în cazurile multidimensionale. O abordare pentru aceste situaţii este distanţa Wasserstein secţionată[32] (DWS), care se rezultă din proiecţia unei distribuţii multidimensionale pe diferite axe pentru a obţine multiple proiecţii unidimensionale (deci distribuţiile univariate $\theta_{\#}\mu$ şi $\theta_{\#}\nu$) care trebuie integrate pe toată sfera unitate *d-1* dimensională.

În aplicatii se foloseşte frecvent o abordare Monte Carlo cu un număr finit de secţiuni care se mediază pentru a obţine DW (Ec. 4.3).

(4.3)
$$\mathbf{SW}_{p,L}^p(\mu,\nu) = (1/L)\sum_{l=1}^{L} \mathbf{W}_p^p\left((\theta_l)_\sharp^\star\mu, (\theta_l)_\sharp^\star\nu\right)$$

Astfel, DWS reprezintă o metodă eficientă şi practică de a compara direct similaritatea între două distribuţii.

DWS a fost deja implemntată în librării software[33], dar implementările existente acceptă ca input doar date brute (sampling din distribuţii, nu distribuţii de probabilitate deja calculate). O posibilă utilizare a implementărilor deja existente ar consta în efectuarea unui sampling din distribuţiile

[32] Aa https://arxiv.org/abs/2106.15427; B https://arxiv.org/abs/1902.00434 ; C https://tel.archives-ouvertes.fr/tel-03533097/documenta
[33] aa https://pythonot.github.io/auto_examples/sliced-wasserstein/plot_variance.html

luminoase, dar aceasta ar presupune din start o pierdere de informaţie şi inducerea unei erori suplimentare în mod nenecesar.

Din acest motiv a fost reimplementat algoritmul DWS pornind doar de la calculul DW uni-dimensional din biblioteca Python POT[34] (*ot.wasserstein_1d*). A fost reimplementată astfel toată metoda de proiecţie şi secţionare pornind de la distribuţia de probabilitate (mai exact, a probabilităţii de masă pentru cazul discret) iar algoritmul de calcul pentru DWS a fost paralelizat pentru o eficienţă numerică sporită (codul este inclus în Anexa 1).

Pentru testarea metodei, pornind de la principiul că o imagine a profilului laser este o estimare a distribuţiei luminoase, s-a folosit DWS pentru a compara între ele 50 de imagini de fascicul folosind 3.000 de secţiuni pentru fiecare pereche.

Calculul celor 1.225 de distanţe a fost rulat în paralel pe 32 de nuclee de procesare (*cores*), ceea ce a necesitat un timp total de rulare aproximativ de 640 de ore (~20 de ore per core). Numărul de 3.000 de secţiuni a fost ales după analiza convergentei distributiei DWS (Fig. 1 de mai jos).

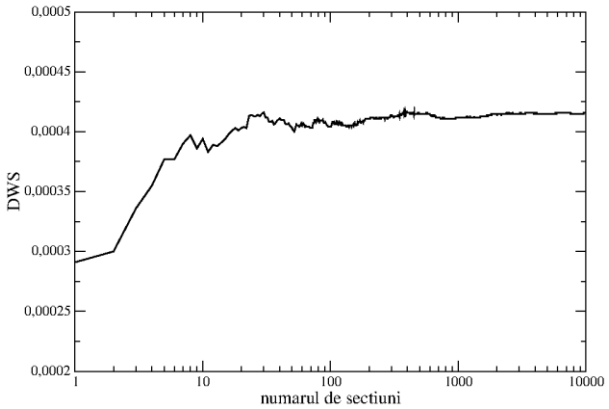


FIG. 4.9: Exemplu de convergenta DWS în funcţie de numărul de secţiuni

Exemple de proiecţii 1D sunt reproduse în în Fig. 4.10 (stânga) pentru 5 secţiuni diferite ale unei distribuţii 2D (dreapta).

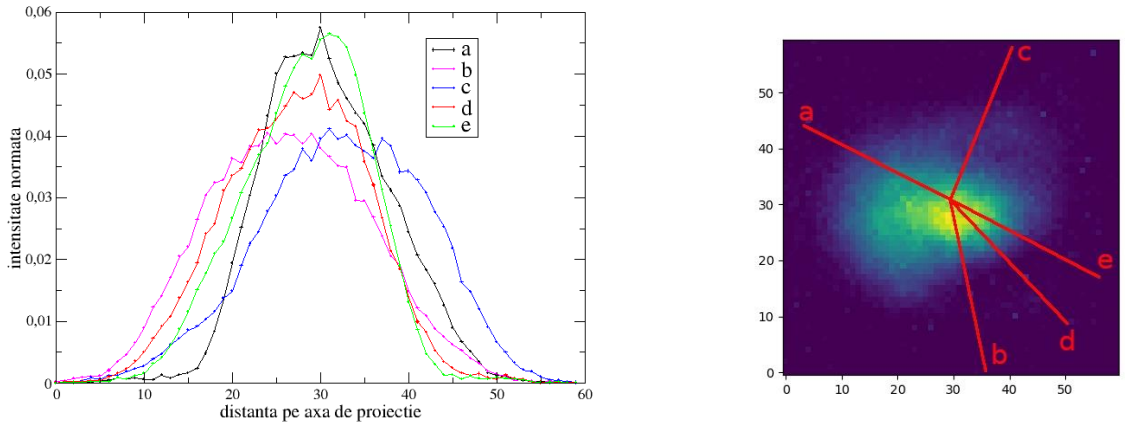


FIG. 4.10: Proiecţii asociate celor 5 secţiuni ale distribuţiei 2D

Folosind metoda descrisă mai sus, s-a obţinut o matrice de distanţe pentru primele 50 de profile de fascicul (în ordine cronologică) din setul DTG_66440739 (Fig. 4.11), simetrică față de diagonala principală.

[34] Pot (6) https://pythonot.github.io/all.html?highlight=wasserstein_1d#ot.wasserstein_1d

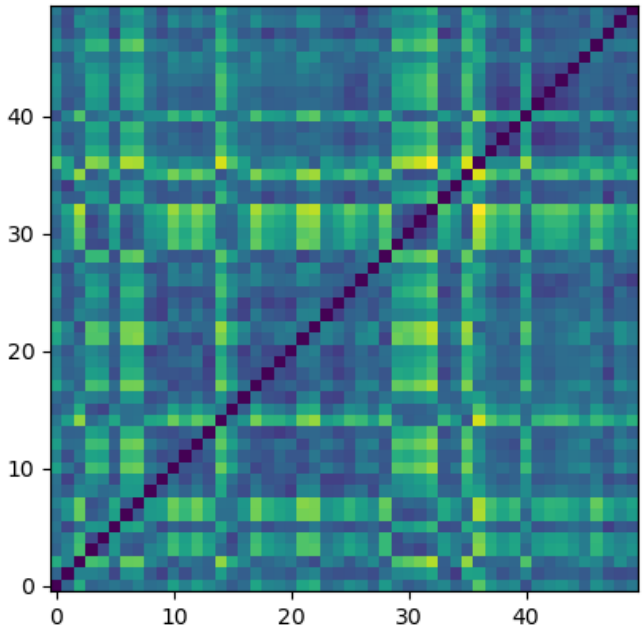


FIG. 4.11: Matricea distanţelor pentru 50 de profile de fascicul

La analiza rezultatelor se observă unele regularităţi care indică corelaţii (de exemplu, profilele 9-14 respectiv 16-28 par sa fie puternic corelate între ele). Zonele cu distanţe mici, deci cu posibilă corelatie ridicată (porţiunile rectangulare cu nuanţă albastru-închis aflate de-a lungul diagonalei), indică o posibilă periodicitate, care poate fi confirmată dacă se extinde calculul la un numar suficient de mare de profile.

În concluzie, analiza prin metoda propusă a unei secvenţe mai mari de profile poate evidenţia anumite corelaţii sau o periodicitate în modul de funcţionare al sistemului laser, care să indice eventuale particularităţi sau probleme tehnice ale ansamblului. În acest scop, acest studiu se poate extinde relativ ușor, dacă se dispune de suficient timp de calcul (care creşte pătratic cu numarul de profile investigate).

4.2.2 Date de monitorizare a sistemelor de pompaj laser

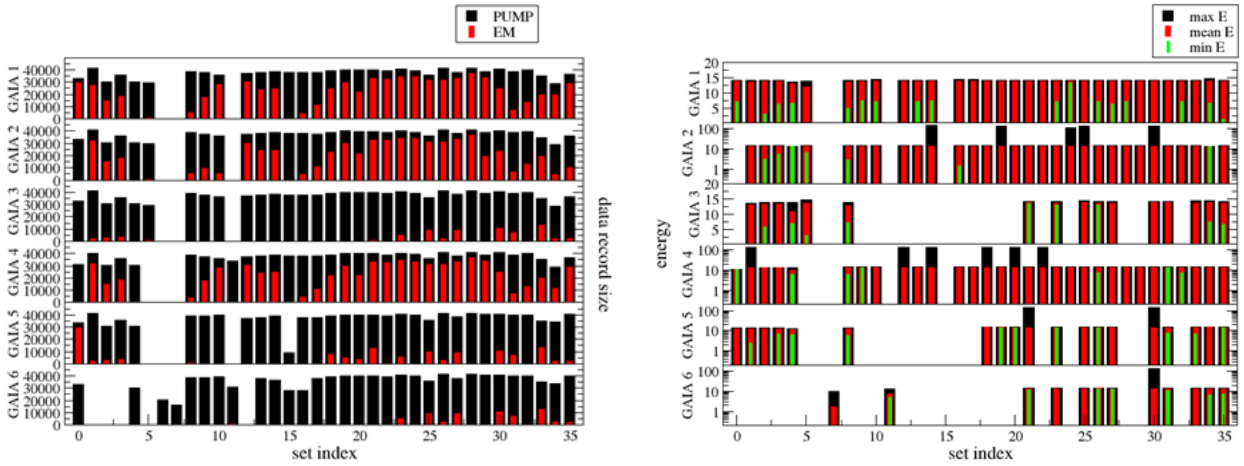
Partea a doua a studiului a fost dedicată investigării caracteristicilor datelor de monitorizare și diagnoză furnizate de sistemele de pompaj în perioada de test a HPLS. Datele investigate au provenit de la cei 6 laseri de pompaj GAIA HP din Fig. 1.1 și energimetre, în urma a 36 de perioade de testare (indicate mai jos prin indexul setului și considerate în ordine cronologică). Pentru fiecare set de date au fost analizate pana la 40.000 de înregistrări.

Ordinea indicilor (0-35) corespunde urmatoarei liste a seturilor din baza de date: '2021-06-30.h5', '2021-08-04.h5', '2021-08-05.h5', '2021-08-06.h5', '2021-08-19.h5', '2021-08-20.h5', '2021-08-26.h5', '2021-08-27.h5', '2021-08-31.h5', '2021-09-01.h5', '2021-09-02.h5', '2021-09-03.h5', '2021-09-06.h5', '2021-09-07.h5', '2021-09-08.h5', '2021-09-09.h5', '2021-09-10.h5', '2021-09-13.h5', '2021-09-14.h5', '2021-09-15.h5', '2021-09-16.h5', '2021-09-17.h5', '2021-09-20.h5', '2021-09-22.h5', '2021-09-23.h5', '2021-09-24.h5', '2021-09-27.h5', '2021-09-28.h5', '2021-09-29.h5', '2021-09-30.h5', '2021-10-01.h5', '2021-10-04.h5', '2021-10-05.h5', '2021-10-06.h5', '2021-10-07.h5', '2021-10-08.h5'.

A) Statistica datelor de monitorizare

În Fig. 4.12 sunt reprezentate: (a) numărul total de date colectate de energimetre (EM) și pompele laser (PUMP) pentru cele 36 de perioade de înregistrare, aranjate în ordine cronologică; (b) valorile minimă (verde), maximă (roșu) și medie (negru) înregistrate de cele șase energimetre.

Se poate observa că numărul total de date colectate diferă atât de la o linie de măsură la alta cât și în funcție de perioada de testare.



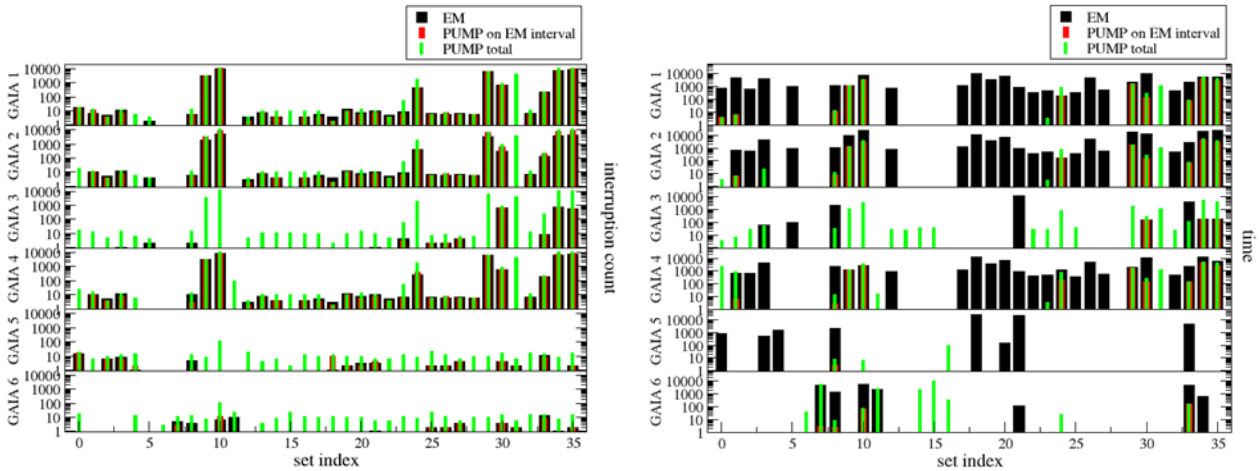
(a) Numărul total de date colectate (b) Valorile extreme și medii înregistrate de EM

FIG. 4.12: Numărul și valorile datelor colectate, în ordine cronologică

Analiza datelor de monitorizare a relevat discrepanțe între indicii de timp înregistrați de energimetre și sistemele de pompaj. Prin discrepanțe se înțelege fie absența datelor înregistrate la un anumit moment izolat de timp, fie un interval mai mare de 1000 de pași/unitati de timp (ut) intre doua inregistrari succesive.

În Fig. 4.13(a) sunt contorizate întreruperile (cu un interval diferit de 1000 ut între două înregistrări succesive), pe perioada totală de funcționare a pompelor laser (verde), pe perioada de funcționare a pompelor laser care se suprapune cu cea a energimetrelor (roșu) și pentru energimetre (negru).

În Fig. 4.13(b) sunt cumulate intervalele de timp pentru care nu sunt înregistrări (în unități de 1000 ut), pe perioada totală de funcționare a pompelor laser (verde), pe perioada de funcționare a pompelor laser care se suprapune cu cea a energimetrelor (roșu) și pentru energimetre (negru).



(a) Numărul de întreruperi (b) Intervale de timp fără înregistrări

FIG. 4.13: Întreruperile și intervalele de timp fără înregistrări

Analiza cumulată a statisticilor din Fig. 4.12 și 4.13 relevă un comportament relativ similar, posibil corelat, între unitățile GAIA 1, 2 și 4. Evidențierea acestor posibile corelatii între laserii de pompaj recomandă folosirea metodei în etapele ulterioare ale exploatării HPLS.

Suplimentar, au fost analizate și înregistrările de diagnoză pentru sistemele de pompare, iar timpul petrecut în diversele stări ale dispozitivului au fost reprezentat procentual în Fig. 8. Stările

înregistrate au fost ['ON', 'STANDBY', 'UNREACHABLE', 'RUNNING', 'FAULT', 'OFF']. Diferența până la 100 de procente sau absența oricărei coloane în Fig. 4.14 reprezintă starea 'ON'.

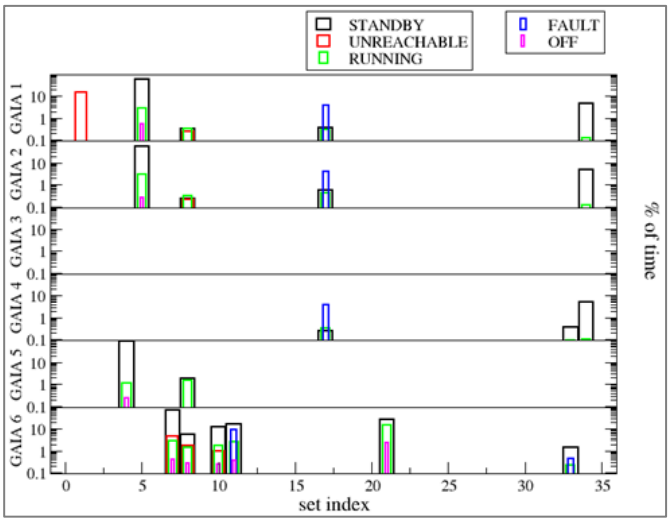


FIG. 4.14: Procentul de timp petrecut de laserii de pompaj într-o anumită stare

În concluzie, în ciuda discrepanțelor constatate în statistica datelor de monitorizare colectate în timpul testelor HPLS, s-a confirmat calitativ valabilitatea metodelor de analiză prezentate pentru evidențierea unor eventuale corelații între evenimentele ce denotă o funcționare anormală a componentelor infrastructurii. Eliminarea cauzelor acestor discrepanțe până la intrarea în faza de exploatare va permite elaborarea unei metode de predicție a eventualelor probleme în funcționarea HPLS.

B) Eliminarea zgomotului din semnalele colectate de energimetre

În mod similar cu necesitatea eliminării zgomotului din datele colectate de senzorii CCD pentru profilele de fascicul laser, descrisă în secțiunea C din Cap. 4.2, semnalele colectate de energimetre necesită de asemenea curăţarea de zgomotul de frecvență înaltă.

Valorile energiei fasciculului sunt colectate cu un pas de timp bine determinat, de către energimetre. Eliminarea zgomotului din aceste date poate ajuta la observarea fluctuaţiilor reale ale energiei fasciculelor şi se pot face corelaţii între acestea şi alţi parametrii înregistraţi.

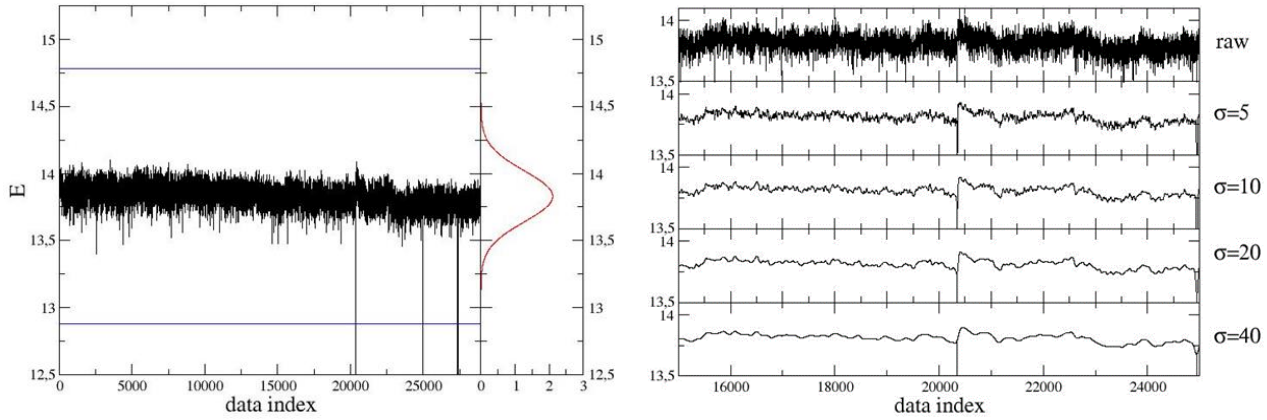
Pentru reducerea zgomotului s-a ales folosirea unui fitru cu profil gaussian. Acest tip de filtru este des întâlnit în prelucrarea semnalelor dependente de timp şi constă în modificarea semnalului măsurat prin aplicarea unei convoluţii cu o funcţie gaussiană. S-a folosit implementarea numerică din biblioteca Python SciPy pentru semnale unidimensionale (*scipy.ndimage.gaussian_filter1d*).

Metoda a fost testată pe un set de date înregistrate de energimetrul corespunzator sistemului GAIA 1, care este reprezentat în negru în Fig. 4.15(a) împreună cu profilul distributiei normale asociate datelor (în roşu).

Se observă că datele fluctuează în jurul unei valori medii (pentru setul prezentat în figură media este 13.83), dar există şi valori extreme, mult sub valoarea medie (vizibile ca linii verticale).

Pentru a nu elimina valorile extreme, s-a ales aplicarea filtrului Gaussian numai datelor care se află la cel mult 5 deviaţii standard *(5σ)* de media setului (barele orizontale albastre indică intervalul de valori filtrate). Valorile care se află în afara acestui interval (liniile verticale din figură) sunt copiate într-un set separat iar în setul original sunt înlocuite cu valoarea medie dintre cele doua puncte imediat învecinate (valoarea de la momentul t-1 şi cea de la t+1). Cele doua bare orizontale (albastru) indică intervalul (5σ) de valori care vor fi ulterior filtrate.

Toate valorile extreme eliminate în această primă etapă sunt readăugate ulterior, după aplicarea filtrului.



(a) Setul de date și distribuția normală asociată (b) Datele înainte și după aplicarea filtrului

FIG. 4.15: Reducerea zgomotului din semnalul energimetrelor folosind un fitru cu profil gaussian

În Fig. 4.15(b) sunt prezentate valorile energiei măsurate (*raw*) şi după aplicarea unui filtru gaussian cu diferite valori pentru deviaţia standard σ (echivalentul ferestrei de netezire). Se poate observa că filtrul elimina fluctuaţiile de frecvenţă înaltă şi permite observarea mult mai clară a variației profilului energetic în timp.

În concluzie, s-a realizat o procedură de filtrare a zgomotului din semnalele colectate de energimetre, care poate fi aplicată în etapa de preprocesare a datelor de monitorizare obținute în regimul de exploatare a HPLS.

4.3 Analiza datelor în regimul de producție științifică

4.3.1 Obiectivele analizei datelor de monitorizare în regim de producție

Conform descrierii HPLS din Cap. 1.1.1, în regim de producție etapele de propagare ale fasciculului laser relevante pentru acest studiu cuprind: unul dintre cele două Front End -uri, unul sau mai multe sisteme de amplifcare, sisteme de diagnosticare.

La trecerea din regimul de testare la cel de exploatare a HPLS echipa LSD a reorganizat baza de date de monitorizare astfel încât denumirile fișierelor HDF5 (reproduse în Anexa 2) să reflecte mai bine poziționarea senzorilor respectivi pe traseul optic.

Folosind notațiile din Fig. 1.1 și Anexa 2, o descriere mai detaliată a etapelor de propagare a fasciculului este sintetizată în tabelul următor:

Denumire etapă	Notație în Fig. 1.1	Notație în Anexa 2	Componente (Fig. 1.1)	Componente (Anexa 2)	Configurația în care este activ
Front End		F1 sau F2			100TW, 1PW, 10PW
Amplificator 1	Amp 1	A1	Amp 1.1, Amp 1.2	A11, A12	100TW, 1PW, 10PW
Amplificator 2	Amp 2	A2			1PW, 10PW
Amplificator 3	Amp 3	A3	Amp 3.1, Amp 3.2	A31, A32	10PW
Sisteme de diagnosticare		TW_DI 1P_DI XP_DI			100TW, 1PW, 10PW 1PW, 10PW 10PW

Fiecare din datele de monitorizare este înregistrată în baza de date împreună cu o marcă de timp (*time stamp*).

Primul pas al analizei constă în detectarea datelor lipsă pentru subseturi de senzori corelați care sunt activi la momentul respectiv (adică pentru care atributul STATE are valoarea ON).

De exemplu, în cazul unui subset de senzori activi compus dintr-un CCD, un energimetru și un spectrometru, se urmăresc atributele profilul fasciculului, energia, respectiv spectrul, pentru o anumită etapă de propagare, într-o anumita configurație și la o anumită marcă de timp. Se detectează o anomalie atunci când, pentru o marcă de timp se constată că lipsește cel puțin o valoare pentru cele 3 atribute de interes de mai sus.

În vederea realizării unui algoritm eficient trebuie mai întâi să se identifice soluția optima pentru procesarea datelor de intrare. Această procesare trebuie să ia in calcul speficitățile setului de date dar și dimensiunea ridicată a acestuia. Analiza preliminară a fost dezvoltată folosind scripturi Python (3.10) care să folosească în mod eficient resursele computaționale existente.

Setul de date privind monitorizarea funcționării și diagnosticul sistemului de laser de mare putere de la ELI-NP[35] este reprezentat sub forma unui set de fișiere de tip *h5* (formatul Hierarchical Data Format versiunea 5) ce conțin înregistrările preluate de la sistemul de control (SCADA) pentru fiecare dintre senzori în parte. Fiecare fișier conține date de la un singur sistem și necesită corelarea informațiilor cu date existente în alte fișiere, fapt ce este îngreunat de volumul ridicat de date înregistrat pentru fiecare zi de funcționare (aprox. $5.5*10^3$ înregistrări pentru fiecare oră de funcționare a sistemului HPLS de la ELI-NP). Fiecare astfel de fișier are dimensiunea medie de 1.85 Mb ceea ce ridică probleme specifice în lucrul cu seturile de date (de cele mai multe ori sistemele de operare nu pot indexa seturi atât de mari de date). Din acest motiv s-a recurs la o metodă de extragere a informațiilor (metadata) din fișiere într-o bază de date facilitând astfel regăsirea și procesarea informațiilor înregistrate.

Deoarece volumul de date este mare (aprox 10^6 înregistrări pentru fiecare săptămână), iar optimizarea algoritmilor de căutare se bazează pe indexarea bazei de date pentru cheile de căutare, utilizarea unei baze de date relaționale (RDBMS) nu poate asigura cerințele de scalabilitate iar reziliența datelor poate fi asigurata doar prin replicare. Acest lucru conduce la limita de funcționare a RDBMS (pentru RDBMS PostgreSQL exista posibilitatea de a partiționa înregistrările, dar gradul de complexitate introdus de acest tip de adresare îngreunează administrarea).

Arhitectura platformei de procesare necesită un pas intermediar de prototipare a proceselor (așa numitul *proof-of-concept*) care constă dintr-un set de scripturi Python care să proceseze setul de date. Pentru acest pas s-a dezvoltat scriptul din Anexa 3, acest script folosind o bază de date cu instanță singulară, pe VM-ul respectiv, pentru stocarea datelor. Deoarece acestă instanță este una accesibilă doar local, nu au fost anonimizate datele de conectare. Pentru procesarea fișierelor acest script le citește direct dintr-un folder mapat peste o partiției NFS (*Network File System*).

Din firul de execuție al scriptului de încărcare se poate observa faptul că acesta folosește un mod agresiv de accesare a resurselor computaționale. Acest mod nu este unul recomandat pentru platforma finală, dar a permis identificarea corectă a structurilor de date prezentate în fișierele de tip H5 preluate din sistemul SCADA al ELI-NP.

Acest pas introduce și un filtru pentru a nu lua în considerare alte fișiere decât cele de tip h5 (folder-ele încărcate în sistem conținând și alte tipuri de fișiere înafara celor de la sistemul de diagnoză).

Următorul pas după ce datele sunt încărcate în baza de date constă în identificarea tripleților (imagine, spectru, respectiv citirile de la energimetru pentru fiecare lanț din sistemul de formare a fasciculului laser (conform datelor înregistrate în 2022[36]).

Deoarece datele nu au o formă standardizată, iar în marea majoritate a cazurilor datele înregistrate de sistemul de diagnoză sunt împărțite între mai multe puncte de achiziție a datelor,

[35] D. Ursescu, G. Cheriaux, P. Audebert, M. Kalashnikov, T. Toncian, M. Cerchez, M. Kaluza, G. Paulous, G. Priebe, R. Dabu, M.O. Cernaianu, M. Dinescu, T. Asavei, I. Dancus et al., *Laser Beam Delivery at ELI-NP*, Romanian Reports in Physics, Vol. 68, Supplement, P. S11–S36, (2016)

[36] Radier, C., Chalus, O., Charbonneau, M., Thambirajah et al. *10 PW peak power femtosecond laser pulses at ELI-NP*, High Power Laser Science and Engineering, 10, E21. doi:10.1017/hpl.2022.11, (2022).

este necesară corelarea informațiilor din fișiere astfel încât pentru fiecare eveniment în parte să putem extrage informațiile de interes din mai multe astfel de fișiere de tip h5. Pentru a indentifica și corela informațiile din fișierele primite am folosit conceptul de *agregare* al bazelor de date NOSQL de tip MongoDB[37]. Astfel, am implementat codul disponibil în Anexa 4 pentru agregarea datelor.

Odată grupate informațiile privitoare la experimentele rulate, este necesară validarea acestora. Procesul de validare presupune deschiderea fiecărui fișier și verificarea stării sistemului respectiv (Camera, Spectrometer sau Energy Meter), iar dacă starea este ”ON”, se pot procesa informațiile înregistrate pentru a vedea dacă acestea sunt valide.

Validarea informațiilor presupune verificarea parametrilor imaginii înregistrate, verificarea datelor privitoare la spectrul de achiziție, respectiv validarea cantității de energie înregistrată de energimetre. Pentru procesarea acestor informații a fost dezvoltat scriptul din Anexa 5. Asemănător scriptului anterior și acesta din urmă folosește un mod agresiv de adresare a resurselor computaționale scopul acestuia fiind acela de a identifica rapid structurile de date ce conțin informații relevante pentru cercetători.

Scopul principal al proceselor de agregare este acela de a oferi cercetătorilor și inginerilor de infrastructură ce lucrează la ELI-NP posibilitatea de a găsi în mod facil informațiile de interes pentru asigurarea bunei funcționări a sistemului laser, respectiv pentru a furniza o formă inteligibilă a informațiilor distribuite în mai multe fișiere de rezultat obținute de la sistemul SCADA. Astfel, spre exemplificare, în figurile de mai jos se regăsește modalitatea de identificare a datelor înainte (Figura 4.16) și după procesul de agregare (Figura 4.17).


```
_id: ObjectId('64ac6d84eb8e7702aa1cbca3')
front: "LA"
arm: null
amp: "A3"
device: "Camera"
file: "s3://eli-np-data/2022-07-01/LA_CameraComputer01-Recorder_60s_snapshot_…"
interval: ""
date: "2022-07-01"
timestamp: "09-34-00-000"
timestamps: "09-34-00"
data_group: "LA_A3_32-Camera_1-Camera-1"
```


FIG. 4.16: Structura datelor din baza NOSQL

În acest caz s-a optat pentru agregarea în doi pași deoarece resursele utilizate de baza de date pot duce la blocarea sistemelor atunci când nu sunt folosite arhitecturi distribuite în rețea (clustere) și deoarece timpul de procesare pentru setul de date extins de la ELI-NP necesită timpi îndelungați de procesare.

Odată identificate structurile de bază pentru metadata asociată fișierelor de date ce conțin informațiile brute putem trece la definirea platformei prin care cercetătorii de la ELI-NP să poată procesa aceste date într-un mod interactiv.

[37] Akalanka Mailewa, Susan Mengel, Lisa Gittner, Hafiz Khan, Mechanisms and techniques to enhance the security of big data analytic framework with MongoDB and Linux Containers, Array, Vol. 15, doi 10.1016/j.array.2022.100236 (2022).

FIG. 4.17: Structura datelor după procesul de agregare

4.3.2 Platforma de prelucrare și analiză a datelor

Platforma de procesare a datelor științifice a fost implementată în conformitate cu cerințele și recomandările EOSC, precum și cu bunele practici menționate în ghidul[38] proiectului european *EOSC synergy*[39].

Implementarea se bazează pe un cluster Kubernetes[40] accesibil din interfata platformei CCBD, aflat în regim de Înaltă Disponibilitate (*High Availability* – HA). Cluster-ul cuprinde 3 noduri de tip Master (*Control Plane*) și 3 noduri Compute (*Worker nodes*), asigurând astfel disponibilitatea serviciilor.

Cluster-ul utilizează un bloc privat de adrese IP din cadrul DFCTI@IFIN-HH asigurând astfel balansarea și accesul prin intermediul unei instanțe de tip HA-PROXY (implementat extern printr-o instanță virtualizată a router-ului[41]). S-a ales această opțiune deoarece asigură posibilitatea de a folosi *Common Address Redundancy Protocol*[42], ceea ce facilitează implementarea unei proceduri de failover la nivel 2 al infrastructurii, dacă acest lucru este necesar.

---

[38] *EOSC Synergy, How to use the interface*, How to use the infrastructure - EOSC-SYNERGY Link List of WP2 last seen July 2023

[39] *EOSC Synergy, Extending EOSC coordination at the National level*, EOSC synergy – Building capacity, developing capability (eosc-synergy.eu) last seen July 2023

[40] The Kubernetes Authors, The Linux Foundation, Operators Pattern, https://kubernetes.io/docs/concepts/extend-kubernetes/operator/ (Accessed June 02, 2023)

[41] Camargo, J. C. B. (2022). OPNsense Beginner to Professional (1st ed.). Packt Publishing. Retrieved from https://www.perlego.com/book/3547283/opnsense-beginner-to-professional-pdf (Original work published 2022)

[42] Nur, R., et al, R. *Gateway Redundancy Using Common Address Redundancy Protocol (CARP)*. IJITEE (International Journal of Information Technology and Electrical Engineering), 2(3), 71-77 (2019). doi:10.22146/ijitee.43701

Deși în această etapă s-au utilizat servere fizice dedicate, este posibilă implementarea folosind OpenStack Octavia[43] ce folosește aceeași soluție ca și cea dezvoltată aici.

App.	Type	Count	CPU cores	RAM	Storage
Storage	bare-metal	1	20 (Xenon Gold 5115)	128 Gb	130 Tb
Master	virtualized	2	8 (Xenon Gold 5115)	16 Gb	35 Gb
Worker	baremetal	1	16 (E5-2650 v2)	64 Gb	465 Gb
Worker*	baremetal	2	32 (E5-2670)	64Gb	465 Gb

* One of the workers is also registered as a master node

FIG. 4.18: Nodurile cluster-ului

Control Plane conține nodurile ce funcționeaza în regim de master, adică asigură distribuirea workload-urilor la nivelul cluster-ului. Aceste noduri de regulă nu rulează procese publice. Scopul principal al acestora este de a asigura cvorumul etcd - serverul de management al configurărilor, al infrastructurii de networking și al daemon-ului (proces de fundal) care asigură controlul (la nivel de API) al resurselor clusterului. Din aceasta categorie fac parte în acest moment 2 noduri virtualizate și un nod fizic mixt (master și worker).

Pentru păstrarea flexibilității implementării în sensul că aceasta poate fi scalată vertical (prin creșterea sau scăderea resurselor disponibile în interiorul nodurilor) și orizontal (prin introducerea sau oprirea nodurilor), implementarea cluster-ului nu va folosi mai multe resurse decât îi sunt necesare pentru fiecare nod în parte.

Astfel, nodurile vor rula containerele asociate serviciilor pe care le deservesc (management sau aplicații) acestea având nevoie să poată fi stocate local. Din acest motiv HDD-ul nod-urilor are o capacitate mai mare decât în situația în care ar trebui să ruleze doar sistemul de operare. De asemenea, aplicațiile au definite în descrierea lor resursele necesare astfel încât planificatorul intern să le ruleze pe un nod ce poate pune la dispoziție aceste resurse. Sistemul de rulare a containerelor (*Container Runtime Implementation* - CRI) se bazează în cazul acestei implementări pe soluția **containerd**, deoarece implementarea folosind **docker / docker-shim** nu mai este suportată de clusterele de tip k8s începând cu versiunea 1.24.

Separarea infrastructurii de rețea a cluster-ului de infrastructura de uz general se realizează la nivel de switch. Această rețea este segmentată astfel încât să se asigure accesul securizat și îmbunătățirea experienței utilizatorilor, conform tabelului de mai jos. Astfel, întreg cluster-ul este organizat într-o clasă privată de tip B (172,19/16) în care există o serie de subrețele care asigură scopuri diferite de implementare. Astfel, blocul C 172.16.1/24 asigură comunicarea fizică în cadrul acestei infrastructuri în timp ce blocurile superioare (de ex. 172.19.2/20) asigură rețeaua de tip overlay specifică cluster-ului. Această împărțire a fost aleasă pentru a permite scalabilitatea ulterioară.

Scop	Bloc	IP-uri	Ocupare
infrastructură	172.19.1.0/24	254	TBD
rutare BGP [a]	172.19.1.33/27	32	TBD
Pod-uri [b]	172.19.2.0/20	4094	TBD
Servicii	172.19.16.0/20	4094	TBD

[a] Pentru anunțarea corectă a pachetelor BGP acesta trebuie să se suprapună cu rețeaua de bază

[b] Pod-ul conține o unitate de lucru cu unul sau mai multe containere

Implementarea sistemului se bazează pe o arhitectură HA clasică, având o serie de servicii disponibile extern (prin *load balancer*), respectiv disponibile doar intern, conform schemei reprezentate in Fig. 4.19.

[43] Introducing Octavia — octavia 12.1.0.dev72 documentation (openstack.org) last seen July 2023

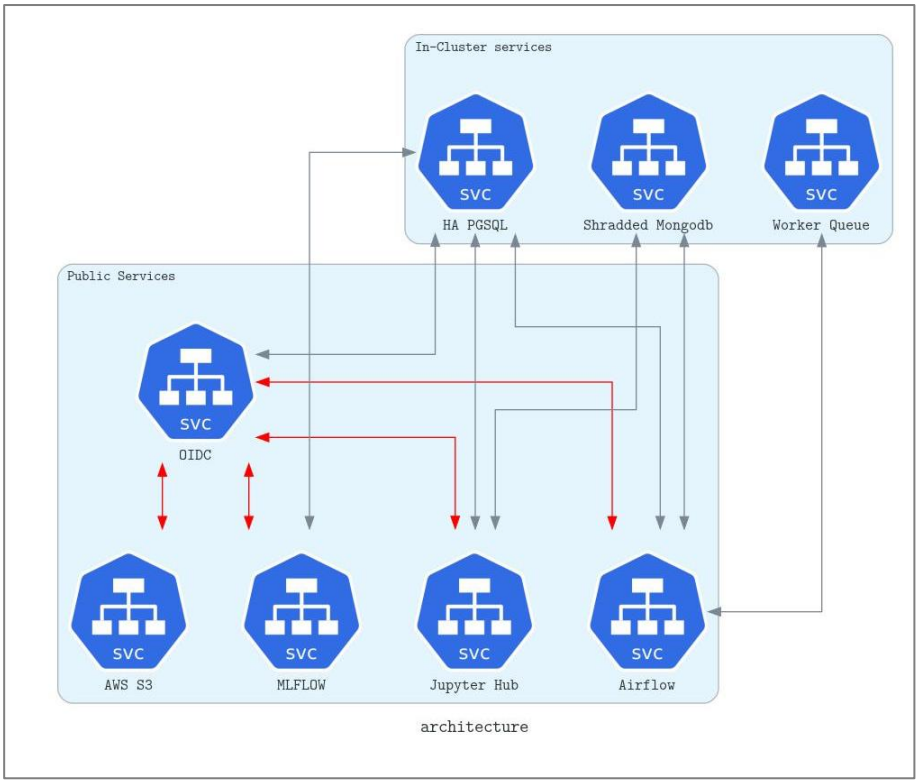


FIG. 4.19: Servicii furnizate de cluster

Servicii interne:

- **Baza de date relațională** este necesară pentru stocarea configurărilor și rularea celor mai multe aplicații din cluster. Inclusiv serverul de autentificare (OIDC) necesită o conexiune la o bază de date pentru stocarea configurației. Soluția aleasă pentru aceasta este PostgreSQL implementată în mod distribuit (3 noduri) care funcționează în regim de multi-master astfel încât pierderea unui nod să nu afecteze funcționarea platformei. Administrarea acesteia este realizată printr-o pagină dedicată (PGAdmin 4) ce nu este expusă publicului și poate fi accesată doar de la nivel local. Fiecare instanță din cele 3 va avea alocat un spațiun de stocare de 8Gb (ce poate fi redimensioat ulterior la nevoie).
- **Baza de metadata** este folosită pentru indexarea fișierelor ingerate de sistem pentru a facilita ulterior regăsirea informațiilor de interes pentru utilizatorul final. Această bază de date este implementată folosind o instanță de MongoDB ce funcționează în mod distribuit conform strategiei de *shrading* pentru a maximiza disponibilitatea la interogare a datelor.
- **Coada de procesare** este necesară pentru distribuirea corectă și procesarea consistentă a datelor ingerate în sistem, aceasta fiind o dependință internă a sistemului de ingestie al datelor, am ales implementarea unei soluții AMQP bazată pe RabbitMQ. Utilizarea acesteia permite distribuirea task-urilor la nivel global in cluster astfel încât să asigurăm funcționarea corectă a aplicațiilor ce interacționază cu utilizatorul îmbunătățind experiența acestuia pe platformă.
- **Redis cache database** Marea majoritate a aplicațiilor folosesc REDIS pentru a păstra cache-ul într-o formă de tipul key-value. Datorită acestui fapt, la nivelul namespace-ului database am instalat o instanță redis. Ea este inițial folosită de aplicațiile publice.

În cadrul implementării pachetele instalate în cluster sunt izolate în *namespace*-uri pentru a facilita astfel administrarea ulterioară. Bazele de date sunt instalate în namespace-ul *database* iar coada de procesare este instalată în namespace-ul *amqp*. Pentru a asigura flexibilitatea implementării serviciilor s-a ales Kubernetes Operator pentru coada de procesare, ceea ce înseamnă că resurse ulterioare pot fi implementate diferit (dar izolate la nivel de namespace)

folosind *Custom Resource Definition*-urile puse la dispoziție de furnizorul pachetului atunci când acestea sunt disponibile.

Servicii publice:

- **Single Sign On OIDC** este asigurat de o instanță de Keycloak[44] configurat pentru asigurarea accesului *multitenat* deoarece, în contrast cu metodologia de cloud privat care este, conform standardelor, una *single tenant*, implementarea CECBID asigură accesul federalizat la resurse. Limitarea drepturilor de acces si aplicațiile pentru care un utilizator are acces este realizată la nivelul acestui server folosind scopuri (*OIDC scopes*) specifice.
- **Prelucrarea datelor științifice** Accesul federalizat și partajarea resurselor folosind diverse tipuri de aplicații presupune implementarea unei soluții flexibile pentru acoperirea unui grad cât mai mare de date științifice ce trebuie prelucrate. Din fericire, în ultimii ani accentul a început să fie pus pe aplicații de analiză folosind Jupyter[45], în cazul nostru aceasta fiind identificată ca fiind platforma optimă pentru a satisface cât mai multe dintre cerințele utilizatorilor mai ales în contextul modurilor hibride de lucru specifice ultimilor ani.
- **Docker Registry Proxy / Private Registry** Docker Registry / Proxy cu Helm Repository - care ajuta la scaderea impactului asupra imaginilor descarcate din Docker HUB (au introdus o politica agresiva de *ratelimit*). De asemenea ne permite implementarea unui repository pentru imaginile Docker dezvoltate intern (imagini de machine learning, Jupyter notebook sau lab personalizate pentru aplicatiile noastre). Helm repository (Chart Museum) ne ajuta in implementarea infrastructurii si a solutiilor impachetate pentru kubernetes. Poti privi Helm chartul ca un apt pentru distributiile linux bazate pe Debian. Soluția implementată este Harbor[46].
- **Ingestia datelor din surse externe** implementeaza fluxul de ingestie in platforma a fisierelor h5 si automatizeaza procesele repetitive. El asigură funcționalitatea de monitorizare, scalare si logare a evenimentelor. Spre deosebire de *cron*, acesta poate genera fluxuri de procesare bazate pe evenimente (de ex putem implementa un hook ce foloseste *inotify* pentru a procesa un fisier imediat ce acesta este salvat pe disc). Soluția implementată este *Apache Airflow*[47].
- **Machine Learning Management server** implementeaza infrastructura pentru dezvoltarea de modele ML si monitorizarea / implementarea acestora. De asemenea poate rula experimente (de ex *Hyper Parameter Tunning*) pe baza modelelor existente. El se integreaza cu notebook-urile folosite, dezvoltate in limbaj Python, Julia, R, etc., si integrează majoritatea framework-urilor disponibile. Soluția implementată este *MLFlow*[48].
- **Data Lake** asigura un sistem de stocare AWS S3 compatibil cu OIDC. Este utilizat pentru stocarea datelor în mod distribuit și pentru stocarea modelele de machine learning și a datelor de training respectiv pentru stocarea modelelor prinse in implementare alaturi de statisticile asociate acestora. Soluția aleasă este *Minio*[49].

---

[44] RedHat inc. Keycloak documentation, https://www.keycloak.org/documentation.html (Accesed May 29, 2023)

[45] Thomas Kluyver et. al., Jupyter Notebooks - a publishing format for reproducible computational workflows, Positioning and Power in Academic Publishing: Players, Agents and Agendas, IOS Press, pp. 87–90 (2016), https://eprints.soton.ac.uk/403913/

[46] Harbor Authors 2023, Harbor, url: https://goharbor.io/ (Accesed June 02, 2023)

[47] The Apache Software Foundation, Apache Airflow documentation, url: https://airflow.apache.org/docs/ (Accessed June 02, 2023)

[48] Zaharia, M.A., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., and Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. IEEE Data Eng. Bull., 41, 39-45

[49] Intel Corporation, Implementation Guide for MinIO Storage-as-a-Service, url:https://min.io/resources/docs/CPG-MinIO-implementation-guide.pdf (Accessed June 02, 2023)

Soluția implementată pentru accesul la datele științifice și procesarea acestora dă posibilitatea de a utiliza o multitudine de limbaje de programare astfel încât cercetătorii să nu fie limitați în dezvoltarea codului lor la limbajele standard pentru web (Python, R, Julia) dar să poată să folosească orice limbaj de programare disponibil.

Implementarea bazată pe Jupyter Hub permite pe lângă utilizarea interfețelor de tip jupyter (notebook sau lab) și folosirea mediilor de lucru integrate (IDE) de tip VSCode prin utilizarea unui Code Server ce poate fi utilizat atât în regim de acces web cât și sub forma unei Portable Web Application (PWA) integrată cu sistemul de operare (Windows, Linux sau MacOS).

Utilizarea IDE a permis implementarea și a platformelor specifice C/C++ folosite atât pentru Machine Learning cât și pentru aplicații științifice.

În acest sens a fost implementată o demonstrație de utilizare a Geant4, alte pachete standardizate putând fi integrate la cerere (ROOT, pyROOT, Garfield etc.). De asemenea este posibilă integrarea unei platforme paralelizate folosind Message Passing Interface (MPI) care să permită programatorilor de aplicatii sau a dezvoltatorilor care folosesc G4MPI accesul la resursele din cluster. Dezvoltări ulterioare pot include pe lângă aceste librării și utilizarea unui planificator (SLURM conform recomandărilor EOSC-Synergy).

Seturile de date furnizate de sistemul SCADA de la HPLS sunt extinse, fiind furnizate cu o rată de 24,35 Gb/oră de funcționare a sistemului laser; aceasta în condițiile din prezent, in care sistemul furnizează doar imagini la rezoluție scăzută. Rata de transfer a datelor crește considerabil (aproximativ de 10 ori) dacă se folosesc imagini la rezoluție completă.

Având în vedere faptul că laserul funcționează aproximativ 11 ore pe zi și că atunci când sunt rulate experimente pe sistemul laser acesta nu se oprește în weekend, pentru o lună de funcționare a ELI-NP rezultă un set de 8169,43 Gb de date de diagnostic.

Freq [Hz]	Record count	Total [Gb]	Average [Mb]	Sample count*
0.17**	13255	1.961	0.148	578
1	120321	52.375	0.435	47374
10	928124	4620.846	4.979	85626
Total	1061700	4675.182	4.403	133578

* Samples taken from the fileset for statistical purposes

** due their reduced size only a small subset was considered for the statistics

FIG. 4.20: Caracteristicile setului de date furnizate de HPLS și utilizate în acest raport

In figura de mai sus sunt reprezentate pe ultima coloană datele folosite în analiza statistică a proceselor de ingestie și procesare a datelor. Se observă faptul că pentru această analiză au fost considerate cu o pondere considerabil mai mare datele de volum ridicat (cele achiziționate cu frecvența de 10Hz). Datele totale (record count, total și average) au fost calculate pentru setul complet de date ce conține înregistrările sistemului SCADA pentru 8 zile de experiment la ELI-NP.

Următorul pas în implementarea serviciilor este adaptarea codurilor prezentate în anexele 3-5 pentru utilizarea infrastructurii realizate. Această adaptare transformă scripturile de încărcare și procesare a datelor din simple scripturi Python în DAG-uri specifice Airflow. Sunt utilizate instanțe specifice pentru managementul proceselor de calcul prin implementarea unui sistem de trigger-are – pornire a task-urilor – respectiv a unui set de noduri de lucrucare să le ruleze. Deoarece în această implementare managementul resurselor este asigurat de instanța de Airflow (prin utilizarea planificatorului *inter* bazat pe *Celery* și *AMQP*, a trigger-ului, respectiv a nodurilor de lucru) configurările asociate resurselor computaționale disponibile nu mai trebuie implementate în DAG.

De asemenea, pentru a asigura scalabilitatea sistemului s-a folosit pentru managementul instanțelor de workeri Airflow un sistem de scalare orizontală automată[50] ce permite scalarea la 0 instanțe (spre deosebire de sistemul standard de scalare din clusterele kubernetes) astfel încât resursele să fie imediat eliberate după ce utilizarea acestora nu mai este necesară în scopul ingestiei documentelor.

Asemănător scripturilor prezentate anterior și în cazul DAG-urilor s-au folosit instanțe separate pentru ingestia, preprocesarea respectiv analiza datelor. În primă instanță, ingestia documentelor presupune încărcarea fiecărui fișier în *Data Lake* și verificarea conținutului acestuia.

DAG-ul este prezentat în figura alăturată. Acesta este cel mai complex pas din ingestie și folosește procesoare dinamice alocate pentru batch-uri de câte 500 de fișiere. În Anexa 6 este prezentată sursa acestui DAG.

4.3.3 Analiza de performanță

În urma procesării documentelor disponibile s-a constatat că doar 2,3% dintre acestea au fost transferate în mod eronat (fie fișierele aveau dimensiunea 0 bytes fie acestea nu erau interpretabile), ceea ce reprezintă o situație normală pentru date transferate manual între medii neomogene.

Retransferarea acestor date a făcut posibilă interpretarea informațiilor și din aceste fișiere.

DAG-urile sunt astfel implementate încât atunci când sunt procesate date deja existente în baza de date ce asigură metadata pentru acestea toate informațiile să fie suprascrise iar setul de date să fie assignat pentru reprocesare pe întreg lanțul de analiză.

Informațiile referitoare la conexiunile la bazele de date, respectiv la sistemul de stocare, sunt administrate intern de către instanța de Airflow folosind *secrets,* astfel încât aceste informații să nu se regăsească în interiorul codului, și deci să se poată minimaliza eventualele riscuri de securitate ce pot apărea atunci când sursele sunt disponibile înafara sistemului.

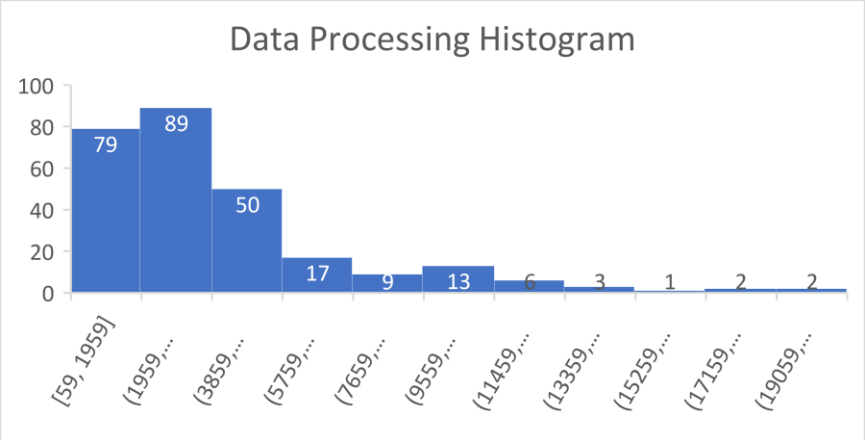
Step	Total [s]	Average [s]	Stdev [s]
List files within the ingestion bucket	349	2.08	1.02
Test if my type of file	$\ll 1$	$\ll 1$	$\ll 1$
Pair up source and destination	1044	6.21	0.86
Process one file	758130	4539.70	3630.93
Copy files to destination bucket	32230	192.99	234.52
Cleanup ingestion bucket	15362	91.99	158.74
Total	**821133**	**4916.94**	**3711.44**

[50] KEDA | Kubernetes Event-driven Autoscaling, last seen July 2023

Din analiza statistică prezentată în tabelul de mai sus se observă timpii de procesare pentru setul de date disponibil. Această analiză conține și optimizările bazelor de date motiv pentru care timpii de analiză sunt supraestimați.

Din această analiză se observă faptul că pentru a asigura o ingestie de tip real-time (sau *same-day*) este necesară alocarea unui set mult mai mare de workeri – în acest moment procesul de ingestie fiind limitat la numărul de procesoare disponibile pentru cluster-ul nostru.

De asemenea, se observă faptul că pentru optimizarea timpilor de procesare este necesară scalarea sistemului de stocare prin implementarea unui cluster de stocare (acest lucru este vizibil prin abaterea standard mult mai ridicată pentru cazurile în care sistemul face operații de intrare-ieșire (I/O). Anexele 7 și 8 conțin DAG-urile asociate proceselor de agregare.



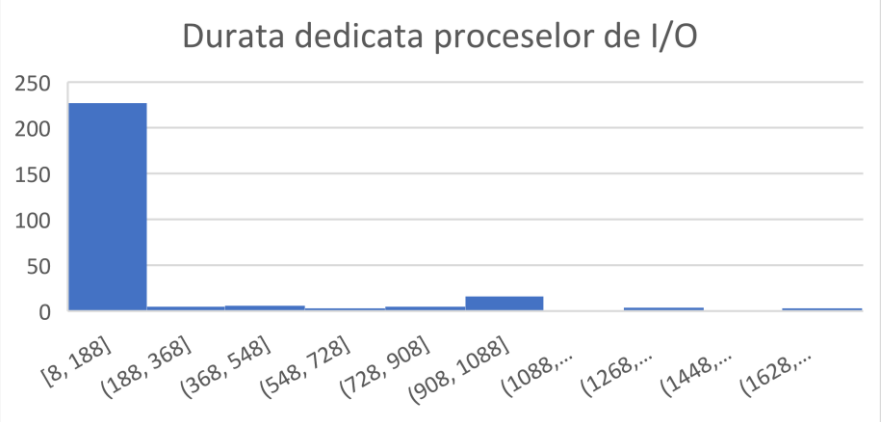


FIG. 4.21: Timpii totali de procesare a fisierelor

5. Algoritmi de învățare automată

Infrastructura de calcul dezvoltata in cadrul proiectului permite implementarea unei metode distribuite de antrenament al ML, evaluare si ajustare a hiperparametrilor (HP - *Hyper Parameter*). Pentru aceasta, datele de instruire și evaluare trebuie să fie disponibile în rețea. In acest scop se utilizeaza fisiere *tfrecords* care sunt stocate intr-un *data lake* compatibil cu AWS S3[51]. Acest lucru permite accesul secvential mai rapid (folosind *protobuf*) pentru infrastructura distribuita de calcul.

Pentru a realiza ingestia datelor intr-un fisier *tfrecords* trebuie stabilite caracteristicile de interes pentru modelul ML. Datorita faptului ca exista mici diferente intre imaginile inregistrate pe cele doua brate ale HPLS, amplificatorul si bratul trebuie se figureze ca trasaturi/caracteristici (*features*) ale modelului, pe langa *frontend*. Marca de timp este utilizata pentru identificarea inregistrarile din baza de metadate, ceea ce va permite evaluarea erorilor si a datelor de iesire asteptate pentru modelul ML.

Caracteristicile principale (*meta features* – informatiile privind identificarea datelor inregistrate) sunt preluate din campul "*_id*" al codului reprodus anterior. Constructia inregistrarilor *tf*, realizata conform documentatiei oficiale TFData[52], permite *streaming*-ul de la *data lake* catre toti clientii interesati de aplicatiile ML.

Această abordare permite automatizarea operațiunilor de invatare și evaluare a modelelor ML care urmează să fie efectuate folosind fluxurile create pentru ingestia datelor în timp ce noi date ajung în infrastructură.

Folosind informațiile ingerate în fluxul de ingerare a documentelor se poate trece la implementarea unui sistem de Machine Learning pentru identificarea anomaliilor în infrastructura HPLS de la ELI-NP, ajutând astfel operatorii infrastructurii să analizeze și să identifice anomaliile din funcționarea sistemului.

Astfel, s-a realizat unui sistem care să asigure analiza anomaliilor din imaginile înregistrate într-o formă cât mai simplă. Se folosește metodologia *Convolutional Auto Encoder* (CAE) pentru a reconstrui imaginea de intrare, iar metrica asociată probabilității de apariție a unei anomalii este dată de integrala diferenței dintre valorile pixelilor din imaginea originală față de imaginea reconstruită pentru fiecare dintre cele 3 canale de culoare în parte.

Ulterior acest model poate fi extins cu ajutorul cercetătorilor și a inginerilor specializați în laserii de mare putere, introducând ca parametri spectrul și energia înregistrate în sistemul de diagnostic.

Implementarea din cadrul proiectului CECBID-EOSC se bazează pe utilizarea framework-ului Tensorflow, motiv pentru care este necesară transformarea datelor din formatul specific ELI-NP utilizat în platforma noastră într-un format specific (*tf.data*) care să permită accesul în mod secvențial din *data lake*, reducând astfel impactul asupra benzii de internet disponibile. Acest lucru este posibil prin utilizarea fișierelor de tip *tfrecords* peste sistemul compatibil AWS S3 deja existent în sistem.

În Anexa 9 este prezentat scriptul ce parsează informațiile privitoare la fișiere și generează fișierul *tfrecords* folosit pentru antrenarea modelului de Machine Learning.

În Anexa 10 este prezentat modelul de bază și scriptul de antrenare al acestuia (folosind infrastructura implementată în cadrul proiectului pentru metrici și versionarea modelului în scopul implementării în producție).

Antrenarea modelului a fost realizată cu imagini de forma celei de mai jos.

[51] Amazon S3 - Cloud Object Storage, https://aws.amazon.com/s3/
[52] https://www.tensorflow.org/guide/data

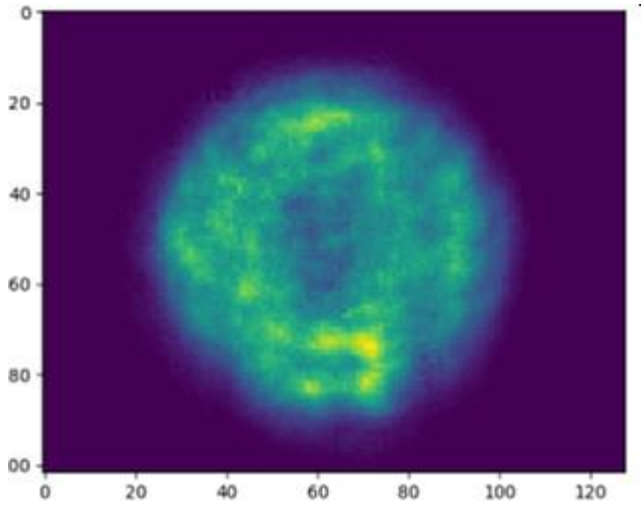


FIG. 4.22: Imagine preluată de la sistemul de amplificare laser

În ultima parte a studiului s-a implementat un model de Machine Learning de bază, folosind CAE și utilizând un model cu un singur fir de execuție în batch-uri de diferite dimensiuni.

Pentru batch-uri de 100 de imagini timpul de procesare este de 1.5 s, pentru batch-uri de 500 de imagini timpul crește la 7.5 s și ajunge la 71,9s pentru batch-uri de 1000 de imagini.

În figura de mai jos se poate observa faptul că modelul trece destul de repede în *overfitting* (pentru mai mult de 7 epoci pentru batch-uri de 100 de imagini).

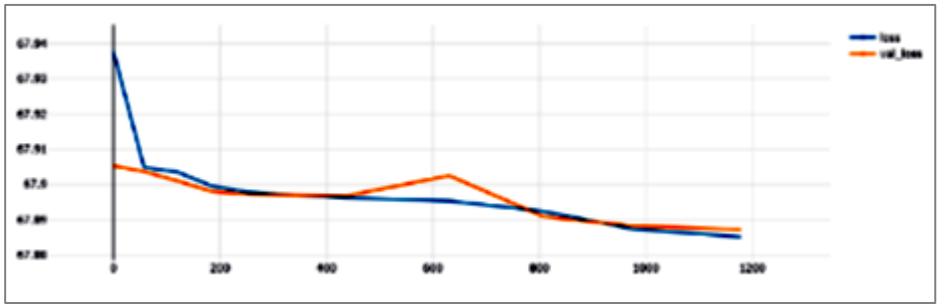


FIG. 4.23: Antrenare și validare pentru rețea CAE

Platforma dezvoltata este disponibilă la adresa https://bigdata.nipne.ro, accesul la aceasta folosind sistemul federat de autentificare bazat pe protocolul Open ID Connect (OIDC) – https://sso.bigdata.nipne.ro – respectiv managementul și rularea experimentelor de Machine Learning prin utilizarea *mlflow* – https://mlf.bigdata.nipne.ro.

Deocamdată accesul din exteriorul IFIN-HH la serviciile de ML, respectiv la sistemul de ingestie (https://air.bigdata.nipne.ro) este permis doar în regim de citire, urmând ca ulterior operatorii autentificați și care au dreptul să acceseze aceste platforme (prin intermediul scopurilor OIDC alocate) să o poată utiliza.

6. Concluzii

Principalul rezultat al acestui studiu este realizarea și testarea platformei software de analiză a datelor de monitorizare ale infrastructurii HPLS, care este gazduită în Centrul de Resurse Cloud și Big Data. Pentru aceasta s-a dezvoltat și implementat o metodă distribuită de antrenament și evaluare a modelelor de învățare automată, care funcționează simultan cu încărcarea de noi date în infrastructură.

Implementarea platformei de analiză a datelor HPLS într-o manieră compatibilă cu integrarea în EOSC permite aplicarea unor metode similare și în cazul altor experimente de fizică, de exemplu pentru simularea Monte-Carlo a transportului radiației folosind framework-ul GEANT4.

Mulțumiri

Echipa proiectului este recunoscătoare dlor Sorin Moldoveanu, Mihai Caragea și Bertrand de Boisdeffre, de la Departamentul Sisteme Laser al ELI-NP, care, sub coordonarea dlui Ioan Dăncuș, au furnizat seturile de date de monitorizare a infrastructurii HPLS utilizate pentru testarea aplicațiilor dezvoltate și au explicat structura bazei de date care le gazduiește.

ANEXA 1 – Codul de calcul al distanţei Wasserstein secţionate

Codul Python folosit pentru calculul DWS este reprodus mai jos.

```
import h5py
import numpy as np
import pylab as plt
import math
import random
import ot
import concurrent.futures
import multiprocessing
manager = multiprocessing.Manager()

random.seed(2)

def rand_sign():
    return 1. if random.random() < 0.5 else -1.

def rand_Lo2(L): #get value on circle with radius L/2 (L=side of cutout)
    x=random.uniform(-float(L)/2., float(L)/2.)
    y=rand_sign() * math.sqrt( (float(L)/2.)**2-x**2)
    return [x+float(L)/2.,y+float(L)/2.]

def project_on_line(a, b, p): #print(project_on_line([0.,0.],[2.,2.],[0.,2.]))
    a=np.array(a)
    b=np.array(b)
    p=np.array(p)
    ap = p-a
    ab = b-a
    result = a + np.dot(ap,ab)/np.dot(ab,ab) * ab
    return result

def dist_to_line_end_int(point, origin, end): #origin is [L/2,L/2]
    #print(point,origin,end)
    coord=project_on_line(origin, end, point)
    return math.floor(math.sqrt((coord[0]-end[0])**2+(coord[1]-end[1])**2 ))

def is_in_circle(L, point):
    if     math.sqrt((point[0]-float(L)/2.)**2+(point[1]-float(L)/2.)**2)     <=
float(L)/2.: x=True
    else: x=False
    return x

def get_slice(data_matrix, Lo2):
    L=len(data_matrix) #
    bins=[0 for i in range(L)] #only binns data from center cutout, ignores
corners
    for i in range(L):
        for j in range(L):
            #print(i,j)
            if is_in_circle(L,[i,j]):
                bins[dist_to_line_end_int([float(i),float(j)],
[float(L)/2.,float(L)/2], Lo2)]+=data_matrix[i][j]

    return bins

def normalize_slice(one_slice):
```

```
    S=sum(one_slice)
    O=[]
    for x in one_slice:
        O.append(float(x)/float(S))
    return O

f=h5py.File('../2020-09-04.h5','r')
G='DTG_66440739'
img_names=[]
for x in f['IMAGES'][G].keys():
    img_names.append(x)
norm_data=[]
no_images=50 #number of images that it compares (all possible pairs)
for i in range(no_images):
    norm_data.append([f['IMAGES'][G][img_names[i]][25:85,35:95]]) #coutout sides
mut be equal and an even number
n_slices=3000
randLo2list=[rand_Lo2(len(norm_data[0][0])) for i in range(n_slices)]
def thread_function(im_nos):
    print(im_nos)
    SW=0
    for i in range(n_slices):
        randLo2=randLo2list[i]
        one_slice=get_slice(norm_data[im_nos[0]][0], randLo2) #pass content of
array
        two_slice=get_slice(norm_data[im_nos[1]][0], randLo2) #pass content of
array

SW+=ot.wasserstein_1d(np.array(normalize_slice(one_slice)),np.array(normalize_sl
ice(two_slice)), p=1) #p=1->1-Wasserstein

    SW/=float(n_slices)
    final_list.append([im_nos[0],im_nos[1],SW])

final_list = manager.list()

my_range=[]
for i in range(no_images):
    for j in range(i+1,no_images):
        my_range.append([i,j])

with concurrent.futures.ProcessPoolExecutor() as executor:
    executor.map(thread_function, my_range)

outfile=open('SW_results_'+str(no_images)+'_'+str(n_slices)+'.dat','w')
for x in final_list:
   outfile.write('%i %i %.12f\n'%(x[0],x[1],x[2]))
outfile.close()
```


ANEXA 2 – Corespondența dintre denumirile fișierelor și poziționarea senzorilor

Beam	Config	Energy Meter Recorded - On	Diag 10PW	DIAG 1PW	DIAG 100TW	AMP3	AMP2	AMP1	Front End
B	10PW	LB_XP_DI-EM_1/EnergyMeter/1	LB_XP_DI-Camera_1/Camera/1 LB_XP_DI-Camera_2/Camera/2 AB_XP_DI-Spectro_1-2/SpectrometerExt/B			LB_A3_31-EM_1/EnergyMeter/1 LB_A3_32-EM_1/EnergyMeter/1 LB_A3_32-Camera_1/Camera/1	LB_A2-EM_1/EnergyMeter/1 LB_A2-Camera_1/Camera/1 LB_A2-Spectro_1-2/SpectrometerExt/1 LB_A2-Spectro_1-2/SpectrometerExt/2	LB_A1_11-EM_1/EnergyMeter/1 LB_A1_12-EM_1/EnergyMeter/1 LB_A1_12-Camera_1/Camera/1 LB_A1_11-Camera_1/Camera/1 LB_A1-Spectro_1-3/SpectrometerExt/3	LB_F2_S2-EM_1/EnergyMeter/1 LB_F2_S2-Camera_1/Camera/1 LB_F2-Spectro_5/SpectrometerExt/5
A	10PW	LA_XP_DI-EM_1/EnergyMeter/1	LA_XP_DI-Camera_1/Camera/1 LA_XP_DI-Camera_2/Camera/2 AB_XP_DI-Spectro_1-2/SpectrometerExt/A			LA_A3_31-EM_1/EnergyMeter/1 LA_A3_32-EM_1/EnergyMeter/1 LA_A3_32-Camera_1/Camera/1	LA_A2-EM_1/EnergyMeter/1 LA_A2-Camera_1/Camera/1 LA_A2-Spectro_1-2/SpectrometerExt/1 LA_A2-Spectro_1-2/SpectrometerExt/2	LA_A1_11-EM_1/EnergyMeter/1 LA_A1_12-EM_1/EnergyMeter/1 LA_A1_12-Camera_1/Camera/1 LA_A1_11-Camera_1/Camera/1 LA_A1-Spectro_1-3/SpectrometerExt/3	LB_F2_S2-EM_1/EnergyMeter/1 LB_F2_S2-Camera_1/Camera/1 LB_F2-Spectro_5/SpectrometerExt/5
B	1PW	LB_1P_DI-EM_1/EnergyMeter/1		LB_1P_DI-Camera_1/Camera/1 LB_1P_DI-Camera_2/Camera/2 AB_1P_DI-Spectro_1-2/SpectrometerExt/B			LB_A2-EM_1/EnergyMeter/1 LB_A2-Camera_1/Camera/1 LB_A2-Spectro_1-2/SpectrometerExt/1 LB_A2-Spectro_1-2/SpectrometerExt/2	LB_A1_11-EM_1/EnergyMeter/1 LB_A1_12-EM_1/EnergyMeter/1 LB_A1_12-Camera_1/Camera/1 LB_A1_11-Camera_1/Camera/1 LB_A1-Spectro_1-3/SpectrometerExt/3	LB_F2_S2-EM_1/EnergyMeter/1 LB_F2_S2-Camera_1/Camera/1 LB_F2-Spectro_5/SpectrometerExt/5
A	1PW	LA_1P_DI-EM_1/EnergyMeter/1		LA_1P_DI-Camera_1/Camera/1 LA_1P_DI-Camera_2/Camera/2 AB_1P_DI-Spectro_1-2/SpectrometerExt/A			LA_A2-EM_1/EnergyMeter/1 LA_A2-Camera_1/Camera/1 LA_A2-Spectro_1-2/SpectrometerExt/1 LA_A2-Spectro_1-2/SpectrometerExt/2	LA_A1_11-EM_1/EnergyMeter/1 LA_A1_12-EM_1/EnergyMeter/1 LA_A1_12-Camera_1/Camera/1 LA_A1_11-Camera_1/Camera/1 LA_A1-Spectro_1-3/SpectrometerExt/3	LB_F2_S2-EM_1/EnergyMeter/1 LB_F2_S2-Camera_1/Camera/1 LB_F2-Spectro_5/SpectrometerExt/5
B	100TW	LB_TW_DI-EM_1/EnergyMeter/1			LB_TW_DI-Camera_1/Camera/1 LB_TW_DI-Camera_2/Camera/2 AB_TW_DI-Spectro_1-2/SpectrometerExt/B			LB_A1_11-EM_1/EnergyMeter/1 LB_A1_12-EM_1/EnergyMeter/1 LB_A1_12-Camera_1/Camera/1 LB_A1_11-Camera_1/Camera/1 LB_A1-Spectro_1-3/SpectrometerExt/3	LB_F2_S2-EM_1/EnergyMeter/1 LB_F2_S2-Camera_1/Camera/1 LB_F2-Spectro_5/SpectrometerExt/5
A	100TW	LA_TW_DI-EM_1/EnergyMeter/1			LA_TW_DI-Camera_1/Camera/1 LA_TW_DI-Camera_2/Camera/2 AB_TW_DI-Spectro_1-2/SpectrometerExt/A			LA_A1_11-EM_1/EnergyMeter/1 LA_A1_12-EM_1/EnergyMeter/1 LA_A1_12-Camera_1/Camera/1 LA_A1_11-Camera_1/Camera/1 LA_A1-Spectro_1-3/SpectrometerExt/3	LB_F2_S2-EM_1/EnergyMeter/1 LB_F2_S2-Camera_1/Camera/1 LB_F2-Spectro_5/SpectrometerExt/5

ANEXA 3 – Script de procesare date

```
import h5py
import glob
from multiprocessing import Pool, cpu_count
import logging
import pymongo
import datetime

## 01-08 July 2022 : 1119520 h5 files (lowres) aprox 6 hours of parallel processing (1.21 Tb of data)



logging.basicConfig(filename="upload_datastructure.log",
                    format="%(asctime)s: %(message)s",
                    filemode="a",
                    level=logging.DEBUG)
logger = logging.getLogger("Threaded_processor")
logger.setLevel(logging.DEBUG)
logger.root.setLevel(logging.DEBUG)

myfiles = glob.glob("data/*/*/*.h5", recursive=True)

uri = "mongodb://mongo:eli2023np@127.0.0.1:27017/?authMechanism=DEFAULT"

client = pymongo.MongoClient(uri)
db = client["eli-np"]
filecol = db["h5-data"]

def is_group_state_valid(file, timestamp:str, group:str) -> bool:
    try:
        dta = file[timestamp][group]["STATE"][()].decode()
        return "ON" in dta.upper()
    except Exception as e:
        logging.error("Error processing state record %s" % e)
        return False

def is_spectrum_valid(file, timestamp: str, group:str) -> bool:
```

```
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False
    spec = file[timestamp][group]["intensityFitted"][:]
    f = 0.0
    for i in spec:
        f += i
    if f > 0.0:
        return True
    else:
        return False
def is_energy_metter_valid(file, timestamp:str, group:str) -> bool:
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False
    try:
        e1 = file[timestamp][group]["energy1Corrected"][0]
        e2 = file[timestamp][group]["energy2Corrected"][0]
        if e1 == 0.0 and e2 == 0.0:
            return False
        return True
    except Exception as e:
        logging.debug("Group not found in file exception: %s" % e)
        return False

def is_image_valid(file, timestamp:str, group:str, refshape=(102,128)) -> bool:
    if not is_group_state_valid(file,timestamp, group):
        # logging.info("Image state is not ON")
        return False
    try:
        myshape = file[timestamp][group]["ROIImageLow"].shape
        if not myshape == refshape:
            logging.info("Invalid shape recorded %s" % myshape)
            return False
        return True
    except Exception as e:
        logging.debug("Exception while processing group: %s" % e)
        return False
```

```
def get_arm_from_groupname(group:str) -> str:
    if ("A1" in group):
        if ("_11" in group):
            return "11"
        if ("_12" in group):
            return "12"
        return "00"
    if ("A2" in group):
        return "00"
    if ("A3" in group):
        if ("_31" in group):
            return "31"
        if ("_32" in group):
            return "32"
    return "00"
def get_device_from_groupname(group:str) -> str:
    if ("EnergyMeter" in group):
        return "EnergyMeter"
    if ("Camera" in group):
        return "Camera"
    if ("Spectro" in group):
        return "Spectro"
    if ("Iseo" in group):
        return "Iseo"
    if ("NiDaq" in group):
        return "NiDaq"
    if ("Oscilloscope" in group):
        return "Oscilloscope"
    if ("Saga" in group):
        return "Saga"
    if ("Opcpa" in group):
        return "Opcpa"
    if ("BeamAlignment-Near" in group):
        return "BeamAlignment-Near"
    if ("Jade" in group):
        return "Jade"
    return "Unknown"
def process_one_file(afile:str):
```

```
    logging.debug("Processing file %s" % afile)
    tt = afile.split("/")
    ttt = tt[3].split("_")
    dtstr = ttt[4].split("-")
    # year = int(dtstr[0])
    # month = int(dtstr[1])
    # day = int(dtstr[2])
    # dtstp = datetime.datetime(year,month,day).date()
    try:
        filedoc = {
            "file": afile,
            "interval": tt[1],
            "date": ttt[4],
            "processed": datetime.datetime.now()
        }
        # fileid = filecol.insertOne(filedoc).inserted_id
        rdict = []
        try:
            with h5py.File(afile) as f:
                for key in f.keys():
                    for grp in f.get(key).keys():
                        dev = get_device_from_groupname(grp)
                        tmpdict = {
                                "front": grp[:2],
                                "arm": get_arm_from_groupname(grp),
                                "head": grp[3:5],
                                "device": get_device_from_groupname(grp),
                                "timestamp": key,
                                "timestamps": key[:len(key)-4],
                                "data_group": grp,
                                "state": is_group_state_valid(f,key,grp),
                                }
                        if dev == "Camera":
                            tmpdict["valid"] = is_image_valid(f,key,grp)
                        if dev == "Spectro":
                            tmpdict["valid"] = is_spectrum_valid(f, key, grp)
                        if dev == "EnergyMeter":
                            tmpdict["valid"] = is_energy_metter_valid(f, key, grp)
```

```
                    rdict.append(tmpdict)
        except Exception as e:
            logging.error("Cannot process record: %s" % e)
        filedoc["records"] = rdict
        filecol.insert_one(filedoc)
    except Exception as e:
        logger.debug("Error processing file: %s" % e)
    logger.debug("Done processing file.")

with Pool(cpu_count() - 3) as pool:
    logging.debug("Starting pool processor")
    pool.map(process_one_file, myfiles)
    logging.debug("Pool processor finished")
logging.debug("Done processing files")
```

ANEXA 4 – Codul de agregare a datelor

```
[
  { $unwind: {path: "$records",},},
  { $match: { $and: [
       { "records.amp": { $in: ["A1", "A2", "A3"], }, },
       {
         "records.device": { $in: [ "Camera", "EnergyMeter", "Spectro", ], },
       },
       { "records.valid": true, },
     ], },},
  { $merge: { into: "amp-h5data", }, },
]
```

ANEXA 5 – Codul de validare

```
import h5py
from multiprocessing import Pool, cpu_count
import logging
import pymongo
import dill as pickle
import traceback

logging.basicConfig(filename="process_amp_triads.log",
                    format="%(asctime)s: %(message)s",
                    filemode="a",
                    level=logging.DEBUG)
logger = logging.getLogger("Threaded_processor")
logger.setLevel(logging.DEBUG)
logger.root.setLevel(logging.DEBUG)



uri = "mongodb://mongo:eli2023np@127.0.0.1:27017/?authMechanism=DEFAULT"
client = pymongo.MongoClient(uri)
db = client["eli-np"]
collection = db["amp_triplets"]
def is_group_state_valid(file:str, timestamp:str, group:str) -> bool:
    try:
        with h5py.File(file, 'r+') as f:
            dta = f[timestamp][group]["STATE"][()].decode()
            return "ON" in dta.upper()
    except Exception as e:
        logging.debug("Data processing exception: %s" % e)
        return False

def is_spectrum_valid(file:str, timestamp: str, group:str) -> bool:
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False
    with h5py.File(file,'r+') as f:
        spec = f[timestamp][group]["intensityFitted"][:]
        f = 0.0
```

```
        for i in spec:
            f += i
        if f > 0.0:
            return True
        else:
            return False
def is_energy_metter_valid(file:str, timestamp:str, group:str) -> bool:
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False
    try:
        with h5py.File(file, 'r+') as h5e:
            e1 = h5e[timestamp][group]["energy1Corrected"][0]
            e2 = h5e[timestamp][group]["energy2Corrected"][0]
            if e1 == 0.0 and e2 == 0.0:
                return False
            return True
    except Exception as e:
        logging.debug("Group not found in file exception: %s" % e)
        return False
def is_image_valid(file:str, timestamp:str, group:str, refshape=(102,128)) -> bool:
    if not is_group_state_valid(file,timestamp, group):
        logging.info("Image state is not ON")
        return False
    try:
        with h5py.File(file, "r+") as f:
            myshape = f[timestamp][group]["ROIImageLow"].shape
            # myimg = f[timestamp][group]
            if not myshape == refshape:
                logging.info("Invalid shape recorded %s" % myshape)
                return False
            return True
    except Exception as e:
        logging.debug("Exception while processing group: %s" % e)
        return False

def group_to_device_name(group:str) -> str:
    refdta = ["EnergyMeter", "Spectro", "Camera"]
```

```
    for i in refdta:
        if i in group:
            return i
    return "Unknown"
class TripletRecordProcessor:
    def __init__(
            self,
            dbname:str,
            destination_collection_name:str,
            filesprefix:str):
        #                                          self.client                                          =
pymongo.MongoClient("mongodb://mongo:eli2023np@127.0.0.1:27017/?authMechanism=DE
FAULT")
        # self.client = client
        self.dbname = dbname
        self.destination_collection_name = destination_collection_name
        self.filesprefix = filesprefix


    def mock(self, record:dict):
        self.client                                                                                  =
pymongo.MongoClient("mongodb://mongo:eli2023np@127.0.0.1:27017/?authMechanism=DE
FAULT", connect=False)
        logging.debug(record)
        def process_record(self, record:dict):
        try:
            with
pymongo.MongoClient("mongodb://mongo:eli2023np@127.0.0.1:27017/?authMechanism=DE
FAULT", connect=False) as local_client:
                fmaps = {"rec": record["_id"] }
                db = local_client[self.dbname]
                destination_collection = db[self.destination_collection_name]
                dgs = record["data_group"]
                fls = record["file"]
                tss = record["timestamp"]
                ldict = {}
                for i in range(len(dgs)):
                    lst = []
                    lf = fls[i]
                    if self.filesprefix and len(self.filesprefix) > 1:
                        lf = "%s/%s" % (self.filesprefix, fls[i])
```

```
                dname = group_to_device_name(dgs[i])
                state = is_group_state_valid(lf, tss[i], dgs[i])
                agrp = {"group": dgs[i], "file": lf, "timestamp": tss[i], "state": state}
                if state:
                    if dname == "Camera":
                        agrp["valid"] = is_image_valid(lf, tss[i], dgs[i])
                    if dname == "EnergyMeter":
                        agrp["valid"] = is_energy_metter_valid(lf, tss[i], dgs[i])
                    if dname == "Spectro":
                        agrp["valid"] = is_spectrum_valid(lf, tss[i],dgs[i])
                lst.append(agrp)
                if dname in ldict.keys():
                    ldict[dname] += lst
                else:
                    ldict[dname] = lst
            # copy keys to fmaps
            for key in ldict.keys():
                fmaps[key] = ldict[key]
            # push record to database
            # logger.info("Inserting: %s" % fmaps)
            _ = destination_collection.insert_one(fmaps)
            return True
        except Exception as e:
            tb = traceback.format_exc()
            logging.error("Cannot process record %s (%s) -- %s" % (record, e, tb))
            return False

def process_triplets(
        client:pymongo.MongoClient,
        dbname:str,
        source_collection_name:str,
        destination_collection_name:str,
        filesprefix:str= ""):
    # Connect to database
    db = client[dbname]
    source_collection = db[source_collection_name]
    # # Parallel processing logic
    # # Mongo does not allow multiprocessing
```

```
    with Pool(cpu_count() -3) as pool:
        processor = TripletRecordProcessor(dbname, destination_collection_name, filesprefix)
        with source_collection.aggregate(
            [
                {'$match': {'_id.amp': {'$in': ['A1', 'A2', 'A3']}, 'device': {'$size': 3}}},
                {'$lookup': {'from': 'amp_triplets_processed', 'localField': '_id', 'foreignField': 'rec',
'as': 'references'}},
                {'$match': {'references': []}}
            ]
        ) as cursor:
        # with source_collection.find({"device": {"$size": 3}}) as cursor:
            logging.info("Starting the pool of processors")
            pool.map(processor.process_record, cursor)
            logging.info("Pool processor ended successfuly")
        # pool.join()


process_triplets(client, "eli-np", "amp_triplets","amp_triplets_processed")
```

ANEXA 6 – DAG-ul de ingestie

```
# using dynamic dags

from airflow import DAG, XComArg
from airflow.decorators import task

from datetime import datetime

from airflow.providers.amazon.aws.operators.s3 import S3CopyObjectOperator,
S3ListOperator, S3DeleteObjectsOperator
from airflow.providers.mongo.hooks.mongo import MongoHook
from airflow.providers.amazon.aws.hooks.s3 import S3Hook
from airflow.operators.python import PythonOperator
import traceback
from airflow.hooks.base import BaseHook
from s3fs import S3FileSystem
import os, h5py, json

import logging, re

task_logger = logging.getLogger('airflow.task')

S3_INGEST_BUCKET = "eli-np-ingestion"
S3_H5_BUCKET = "eli-np-data"
S3_ERROR_BUCKET = "eli-np-error"
H5_FILE_REGEX = re.compile(".\.h5$", flags=re.I)
S3_CONN_NAME = "aws_bigdata_conn"
MONGO_CONN_NAME = "bigdata-mongo"
MONGO_COLLECTION = "eli-np-h5data"
MONGO_DATABASE = "bigdata"

with DAG(
    dag_id="eli_np_ingestion_v2",
    start_date=datetime(2022,9,1),
    schedule_interval='*/5 * * * *',
    catchup=False,
    max_active_runs=1,
    concurrency=96,
    max_active_tasks=100,
    doc_md=__doc__
) as dag:

    def list_files_with_limit():
        hook = S3Hook(aws_conn_id=S3_CONN_NAME, verify=False,)
        return hook.list_keys(
            bucket_name=S3_INGEST_BUCKET,
            max_items=500
```

```
        )

    list_files_to_ingest = PythonOperator(
        task_id="list_files_within_the_ingestion_bucket",
        python_callable=list_files_with_limit

    )

    @task
    def test_if_is_my_type_of_file(source_key_list: list) -> list:
        destination_list = []
        for key in source_key_list:
            # if  H5_FILE_REGEX.search(key):
            if ".h5" in key:
                destination_list.append(S3_H5_BUCKET)
            else:
                destination_list.append(S3_ERROR_BUCKET)
        return destination_list

    copy_dst_list = test_if_is_my_type_of_file(XComArg(list_files_to_ingest))

    @task
    def pair_up_src_dst(src:list, dst:list) -> list:
        src_dst_list = []
        for s, d in zip(src, dst):
            logging.warn(s,d)
            src_dst_list.append(
                {
                    "source_bucket_key": f"s3://{S3_INGEST_BUCKET}/{s}",
                    "dest_bucket_key": f"s3://{d}/{s}"
                }
            )
        logging.warn(src_dst_list)
        return src_dst_list

    copy_list = pair_up_src_dst(
        XComArg(list_files_to_ingest),
        copy_dst_list
    )

    @task(retries=3)
    def process_a_file(
            meta:dict,
            dest_filter:str="data",
            err_bucket:str="",
            aws_conn_id:str="",
            mongo_conn_id:str="",
            mongo_database:str="",
```

```
        mongo_collection:str="") -> list:

    def get_arm_from_groupname(group:str) -> str:
        if ("A1" in group):
            if ("_11" in group):
                return "11"
            if ("_12" in group):
                return "12"
            return "00"
        if ("A2" in group):
            return "00"
        if ("A3" in group):
            if ("_31" in group):
                return "31"
            if ("_32" in group):
                return

    def get_device_from_groupname(group:str) -> str:
        if ("EnergyMeter" in group):
            return "EnergyMeter"
        if ("Camera" in group):
            return "Camera"
        if ("Spectro" in group):
            return "Spectro"
        if ("Iseo" in group):
            return "Iseo"
        if ("NiDaq" in group):
            return "NiDaq"
        if ("Osciloscope" in group):
            return "Osciloscope"
        if ("Saga" in group):
            return "Saga"
        if ("Opcpa" in group):
            return "Opcpa"
        if ("BeamAlignment-Near" in group):
            return "BeamAlignment-Near"
        if ("Jade" in group):
            return "Jade"
        return "Unknown"


    if S3_H5_BUCKET in meta["dest_bucket_key"]:
        to_insert = []
        source_path = meta["source_bucket_key"]
        tt = source_path.split("/")
        source_file = tt[-1]
        tmp_dst = meta["dest_bucket_key"]
        ttt = tt[-1].split("_")
```

```
            idxs = [pos for pos, char in enumerate(tmp_dst) if char == os.sep]
            meta["dest_bucket_key"] = "".join([tmp_dst[:idxs[-1]], os.sep, ttt[4],
os.sep, tmp_dst[idxs[-1]+1:]])
            s3_uri = BaseHook.get_connection(S3_CONN_NAME).get_uri()
            s3_extra = json.loads(BaseHook.get_connection(aws_conn_id).get_extra())
            print("AWS extra config: %s" % str(s3_extra))
            s3 = S3FileSystem(
                key=s3_extra["aws_access_key_id"],
                secret=s3_extra["aws_secret_access_key"],
                client_kwargs=s3_extra)
            try:
                print("Opening file: %s" % source_path)
                with s3.open(source_path) as s3f:
                    print("S3 got file handle")
                    with h5py.File(s3f, mode='r') as f:
                        for h5key in f.keys():
                            for grp in f.get(h5key).keys():
                                tmpdict = {
                                    "front": grp[:2],
                                    "arm": get_arm_from_groupname(grp),
                                    "amp": grp[3:5],
                                    "device": get_device_from_groupname(grp),
                                    "file": meta["dest_bucket_key"],
                                    "interval": tt[1],
                                    "date": ttt[4],
                                    "timestamp": h5key,
                                    "timestamps": h5key[:len(h5key)-4],
                                    "data_group": grp}
                                if "STATE" in f.get(h5key).get(grp).keys():
                                    tmpdict["state"]:
f.get(h5key).get(grp).get("STATE")[()]
                                to_insert.append(tmpdict)
                print("Done with the file, connecting to mongo")
                mongo = MongoHook(conn_id=MONGO_CONN_NAME)
                col = mongo.get_collection(
                        mongo_collection=MONGO_COLLECTION,
                        mongo_db=MONGO_DATABASE)
                print("Got collection", col)
                # deduplicate data
                try:
                    print(col, type(col))
                    fstr = "%s" % meta["dest_bucket_key"]
                    col.delete_many({"file": fstr})
                    print("records deleted")
                except Exception as e:
                    traceback.print_exc()
                    print("New file, no records found")
                print("Should process data", to_insert)
```

```
                mongo.insert_many(
                    mongo_collection=MONGO_COLLECTION,
                    docs=to_insert,
                    mongo_db=MONGO_DATABASE)

            except Exception as e:
                print("Got an exception")
                traceback.print_exc()
                tmp_dst = meta["dest_bucket_key"]
                meta["dest_bucket_key"] = tmp_dst.replace(dest_filter, err_bucket)
        else:
            print("Not my type of file, ignoring")
        return meta

    file_processed = process_a_file.partial(
        dest_filter=S3_H5_BUCKET,
        err_bucket=S3_ERROR_BUCKET,
        aws_conn_id=S3_CONN_NAME,
        mongo_collection=MONGO_COLLECTION,
        mongo_database=MONGO_DATABASE,
        mongo_conn_id=MONGO_CONN_NAME).expand(meta=copy_list)

    copy_s3_file = S3CopyObjectOperator.partial(
        task_id="copy_files_to_destination_bucket",
        aws_conn_id=S3_CONN_NAME,
        verify=False,
        retries=3
    ).expand_kwargs(file_processed)

    cleanup_ingestion_bucket = S3DeleteObjectsOperator.partial(
        task_id="cleanup_ingestion_bucket",
        aws_conn_id=S3_CONN_NAME,
        bucket=S3_INGEST_BUCKET,
        verify=False,
        retries=3
    ).expand(keys=XComArg(list_files_to_ingest))

    copy_s3_file >> cleanup_ingestion_bucket
```

ANEXA 7 – DAG-ul de agregare

```
# using dynamic dags

from airflow import DAG
from airflow.decorators import task

from datetime import datetime

from airflow.providers.mongo.hooks.mongo import MongoHook
import h5py

import logging

task_logger = logging.getLogger('airflow.task')

MONGO_CONN_NAME = "bigdata-mongo"
MONGO_COLLECTION = "eli-np-h5data"
MONGO_DATABASE = "bigdata"

def is_group_state_valid(file:str, timestamp:str, group:str) -> bool:
    try:
        with h5py.File(file, 'r+') as f:
            dta = f[timestamp][group]["STATE"][()].decode()
            return "ON" in dta.upper()
    except Exception as e:
        logging.debug("Data processing exception: %s" % e)
        return False

def is_spectrum_valid(file:str, timestamp: str, group:str) -> bool:
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False
    with h5py.File(file,'r+') as f:
        spec = f[timestamp][group]["intensityFitted"][:]
        f = 0.0
        for i in spec:
            f += i
        if f > 0.0:
            return True
        else:
            return False

def is_energy_metter_valid(file:str, timestamp:str, group:str) -> bool:
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False
    try:
```

```
        with h5py.File(file, 'r+') as h5e:
            e1 = h5e[timestamp][group]["energy1Corrected"][0]
            e2 = h5e[timestamp][group]["energy2Corrected"][0]
            if e1 == 0.0 and e2 == 0.0:
                return False
            return True
    except Exception as e:
        logging.debug("Group not found in file exception: %s" % e)
        return False

def is_image_valid(file:str, timestamp:str, group:str, refshape=(102,128)) -> bool:
    if not is_group_state_valid(file,timestamp, group):
        logging.info("Image state is not ON")
        return False
    try:
        with h5py.File(file, "r+") as f:
            myshape = f[timestamp][group]["ROIImageLow"].shape
            # myimg = f[timestamp][group]
            if not myshape == refshape:
                logging.info("Invalid shape recorded %s" % myshape)
                return False
            return True
    except Exception as e:
        logging.debug("Exception while processing group: %s" % e)
        return False

def group_to_device_name(group:str) -> str:
    refdta = ["EnergyMeter", "Spectro", "Camera"]
    for i in refdta:
        if i in group:
            return i
    return "Unknown"

with DAG(
    dag_id="eli_np_aggregation",
    start_date=datetime(2022,9,1),
    schedule_interval='@daily',
    catchup=False,
    max_active_runs=1,
    concurrency=96,
    max_active_tasks=100,
    doc_md=__doc__
) as dag:
    @task
    def create_amp_triplets(is_success=True):
        mongo = MongoHook(conn_id=MONGO_CONN_NAME)
        col = mongo.get_collection(
                        mongo_collection=MONGO_COLLECTION,
```

```
                        mongo_db=MONGO_DATABASE)
        print("Got collection", col)
        agg_query = [
            { '$match': { 'amp': {'$in': ['A1', 'A2', 'A3']}, 'device': {'$ne':
'Unknown'}} },
            { '$group': {
                    '_id': {
                        'date': '$date',
                        'timestamps': '$timestamps',
                        'interval': '$interval',
                        'frontend': '$front',
                        'amp': '$amp',
                        'arm': '$arm'
                    },
                    'device': { '$addToSet': '$device' },
                    'file': { '$push': '$file' },
                    'data_group': { '$push': '$data_group'},
                    'timestamp': {'$push': '$timestamp'}
                }
            }, {'$merge': 'amp_triplets'}
        ]
        mongo.aggregate(
            mongo_collection=MONGO_COLLECTION,
            mongo_db=MONGO_DATABASE,
            allowDiskUse=True,
            maxTimeMS=680000,
            aggregate_query=agg_query)
        return True


    @task
    def data_cleansing(is_success=True):
        mongo = MongoHook(conn_id=MONGO_CONN_NAME)
        col = mongo.get_collection(
                        mongo_collection=MONGO_COLLECTION,
                        mongo_db=MONGO_DATABASE)
        print("Got collection", col)
        aggregation_query = [
            { '$unwind': {'path': '$records'}},
            { '$match': {
                 '$and': [
                    { 'records.amp': {'$in' : ['A1','A2','A3']}},
                    { 'records.device': {'$in':
['Camera','EnergyMeter','Spectro']}},
                    { 'records.valid': 'true' }
                ]
            }},
            { '$out': 'amp-h5data'}
```

```
        ]
            # { '$merge': {'into': 'amp-h5data'}}

        mongo.aggregate(
            mongo_collection=MONGO_COLLECTION,
            mongo_db=MONGO_DATABASE,
            allowDiskUse=True,
            maxTimeMS=680000,
            aggregate_query=aggregation_query)
        return True

    @task
    def create_prcessed_collection(is_success=True):
        try:
            # should throw exception when collection exists
            mongo = MongoHook(conn_id=MONGO_CONN_NAME)
            mongo.get_conn().get_database().create_collection("amp_triplets_process
ed")
        except:
            pass # ignore it
        return True

    clean = data_cleansing()
    ampd = create_amp_triplets(clean)
    create_prcessed_collection(ampd)
```

ANEXA 8 – DAG-ul de procesare a datelor

```
# using dynamic dags

from airflow import DAG, XComArg
from airflow.decorators import task

from datetime import datetime

from airflow.providers.amazon.aws.operators.s3 import S3CopyObjectOperator,
S3ListOperator, S3DeleteObjectsOperator
from airflow.providers.mongo.hooks.mongo import MongoHook
from airflow.providers.amazon.aws.hooks.s3 import S3Hook
from airflow.operators.python import PythonOperator
from airflow.hooks.base import BaseHook
from s3fs import S3FileSystem
import h5py, json

import logging

task_logger = logging.getLogger('airflow.task')

S3_CONN_NAME = "aws_bigdata_conn"
MONGO_CONN_NAME = "bigdata-mongo"
MONGO_DATABASE = "bigdata"

MONGO_SOURCE_COL = "amp_triplets"
MONGO_DEST_COL = "amp_triplets_processed"

__dic__ = "Simple documentation for this dag"

# '*/20 * * * *'


# Data processing functions

def group_to_device_name(group:str) -> str:
    refdta = ["EnergyMeter", "Spectro", "Camera"]
    for i in refdta:
        if i in group:
            return i
    return "Unknown"

def is_group_state_valid(file:h5py.File, timestamp:str, group:str) -> bool:
    try:
        dta = file[timestamp][group]["STATE"][()].decode()
        return "ON" in dta.upper()
    except Exception as e:
```

```
        logging.debug("Data processing exception: %s" % e)
        return False

def is_spectrum_valid(file:h5py.File, timestamp: str, group:str) -> bool:
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False

    spec = file[timestamp][group]["intensityFitted"][:]
    f = 0.0
    for i in spec:
        f += i
    # cannot use shorthand return f > 0.0 due to numpy.Bool cast
    if f > 0.0:
        return True
    else:
        return False

def is_energy_metter_valid(file:h5py.File, timestamp:str, group:str) -> bool:
    if not is_group_state_valid(file, timestamp, group):
        logging.info("Invalid state in h5 file group")
        return False
    try:
        e1 = file[timestamp][group]["energy1Corrected"][0]
        e2 = file[timestamp][group]["energy2Corrected"][0]
        if e1 == 0.0 and e2 == 0.0:
            return False
        return True
    except Exception as e:
        logging.debug("Group not found in file exception: %s" % e)
        return False

def is_image_valid(file:h5py.File, timestamp:str, group:str, refshape=(102,128)) ->
bool:
    if not is_group_state_valid(file,timestamp, group):
        logging.info("Image state is not ON")
        return False
    try:
        myshape = file[timestamp][group]["ROIImageLow"].shape
        if not myshape == refshape:
            logging.info("Invalid shape recorded %s" % myshape)
            return False
        return True
    except Exception as e:
        logging.debug("Exception while processing group: %s" % e)
        return False
```

```
with DAG(
    dag_id="eli_np_amp_process_v1",
    start_date=datetime(2022,9,1),
    schedule_interval='*/25 * * * *',
    catchup=False,
    max_active_runs=1,
    concurrency=96,
    max_active_tasks=100,
    doc_md=__doc__
) as dag:

    @task
    def get_unprocessed_records():
        mongo = MongoHook(conn_id=MONGO_CONN_NAME)
        agg_query = [
            {'$match': {'_id.amp': { '$in': [ 'A1', 'A2', 'A3' ] },'device.0':
{'$exists': 'true'}}},
            {'$lookup': { 'from': 'amp_triplets_processed', 'localField': '_id',
'foreignField': 'rec', 'as': 'references'} },
            {'$match': { 'references': { '$size': 0 }}},
            {'$limit': 500}
        ]
        return [ doc for doc in mongo.aggregate(
            mongo_collection=MONGO_SOURCE_COL,
            mongo_db=MONGO_DATABASE,
            allowDiskUse=True,
            maxTimeMS=68000000,
            aggregate_query=agg_query)]

    @task(retries=3)
    def process_a_record(
            meta:dict,
            aws_conn_id:str="",
        ) -> any:

        fmaps = {"rec": meta['_id']}
        dgs = meta['data_group']
        fls = meta['file']
        tss = meta['timestamp']
        ldict = {}

        for i in range(len(dgs)):
            lst = []
            lf = fls[i]
```

```
            dname = group_to_device_name(dgs[i])
            s3_extra = json.loads(BaseHook.get_connection(aws_conn_id).get_extra())
            s3 = S3FileSystem(
                key=s3_extra["aws_access_key_id"],
                secret=s3_extra["aws_secret_access_key"],
                client_kwargs=s3_extra)
            try:
                print("Opening file: %s" % lf)
                with s3.open(lf) as s3f:
                    with h5py.File(s3f, mode='r') as f:
                        state = is_group_state_valid(f, tss[i], dgs[i])
                        agrp = {"group": dgs[i], "file": lf, "timestamp": tss[i],
"state": state}
                        if state:
                            if dname == "Camera":
                                agrp["valid"] = is_image_valid(f, tss[i], dgs[i])
                            if dname == "EnergyMeter":
                                agrp["valid"] =  is_energy_metter_valid(f, tss[i],
dgs[i])
                            if dname == "Spectro":
                                agrp["valid"] = is_spectrum_valid(f, tss[i],dgs[i])
                        lst.append(agrp)
            except Exception as e:
                pass
            if dname in ldict.keys():
                ldict[dname] += lst
            else:
                ldict[dname] = lst
        for key in ldict.keys():
            fmaps[key] = ldict[key]
        # one shot insert - can be implemented with insert many in another DAG step
        # this approach allows for error to occure as they are handled later on the
next run
        try:
            mongo = MongoHook(conn_id=MONGO_CONN_NAME)
            mongo.insert_one(
                mongo_db=MONGO_DATABASE,
                mongo_collection=MONGO_DEST_COL,
                doc=fmaps
            )
            return True
        except:
            return False

    # instantiate the DAG

    list_records_to_process = get_unprocessed_records()
    records_processed = process_a_record.partial(
```

```
        aws_conn_id=S3_CONN_NAME).expand(meta=list_records_to_process)
```


ANEXA 9 – Generarea fișierului tfrecords folosit pentru antrenare

```
import pymongo, logging, h5py, os, tempfile
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt

logging.basicConfig(
   filename="00-create-tfrecords.log",
   format="%(asctime)s: %(message)s",
   filemode='a',
   level=logging.DEBUG)
logger = logging.getLogger("00-create-tfrecords")
logger.setLevel(logging.DEBUG)
logger.root.setLevel(logging.DEBUG)

def _bytes_feature(value):
  """Returns a bytes_list from a string / byte."""
  if isinstance(value, type(tf.constant(0))):
    value = value.numpy() # BytesList won't unpack a string from an EagerTensor.
  return tf.train.Feature(bytes_list=tf.train.BytesList(value=[value]))

def _float_feature(value):
  """Returns a float_list from a float / double."""
  if isinstance(value, type(tf.constant(0))):
    value = value.numpy()
  return tf.train.Feature(float_list=tf.train.FloatList(value=[value]))

def _int64_feature(value):
  """Returns an int64_list from a bool / enum / int / uint."""
  if isinstance(value, type(tf.constant(0))):
    value = value.numpy()
  return tf.train.Feature(int64_list=tf.train.Int64List(value=[value]))

def record_example(
      image, energymeter, spectra,
      amp, arm, front, timestamp, interval, date):
```

```
    feature = {
        "image": _bytes_feature(image),
        "energymeter": _float_feature(energymeter),
        "spectra": _bytes_feature(spectra),
        "amp": _int64_feature(amp),
        "arm": _int64_feature(arm),
        "front": _int64_feature(front),
        "timestamp": _bytes_feature(timestamp),
        "interval": _bytes_feature(interval),
        "date": _bytes_feature(date)
    }
    return tf.train.Example(features=tf.train.Features(feature=feature))

def write_features(filename, records):
  with tf.io.TFRecordWriter(filename) as writer:
    for record in records:
      writer.write(record.SerializeToString())

uri = "mongodb://<hidden for security purposes>"

client = pymongo.MongoClient(uri)
db = client["eli-np"]
collection = db["amp-to-ml"]

def camera_get_image(
        file:str, timestamp:str, group:str,
        record:str="ROIImageLow", plot:bool=False) -> np.ndarray:
    with h5py.File(file, "r+") as h5i:
        result = h5i[timestamp][group][record][:]
        if plot:
            plt.imshow(result)
        return result

def meter_get_energy(
        file:bool, timestamp:str, group:str,
        logEnergy:bool=False) -> float:
    with h5py.File(file, "r+") as h5e:
        e1 = h5e[timestamp][group]["energy1Corrected"][0]
```

```
        e2 = h5e[timestamp][group]["energy2Corrected"][0]
        if e1 > 0:
            if e2 > 0:
                result = (e1+e2)/2
            else:
                result = e1
        else:
            result = e2
        if logEnergy:
            logging.debug("Energy Meter %.4f" % result)
        return result

def image_to_string(src:np.ndarray) -> bytes:
    plt.imsave("tmpfile.jpg", src)
    image_string = open("tmpfile.jpg", 'rb').read()
    os.remove("tmpfile.jpg")
    return image_string

def get_encoded_amp(amp:str) -> int:
    if amp == "A1":
        return 1
    if amp == "A2":
        return 2
    if amp == "A3":
        return 3
    return 0

def get_encoded_arm(arm:str) -> int:
    if arm == "11" or arm == "31":
        return 1
    if arm == "12" or arm == "32":
        return 2
    return 0

def get_encoded_front(front:str) -> int:
    if (front == "LA"):
        return 1
    if (front == "LB"):
```

```
        return 2
    return 0


def process_record(record):
    nullspectra = np.array([0.0 for x in range(2047)]).tobytes()
    result = {}
    if "Camera" in record["device"]:
        idx = record["device"].index("Camera")
        result["image"] = camera_get_image("../%s" % record["file"][idx],
record["timestamp"][idx], record["group"][idx])
    if "EnergyMeter" in record["device"]:
        idx = record["device"].index("EnergyMeter")
        result["energy"] = meter_get_energy("../%s" % record["file"][idx],
record["timestamp"][idx], record["group"][idx])
    # No viable (correlated) spectra yet
    result["spectro"] = nullspectra
    result["timestamp"] = record["_id"]["timestamp"].encode()
    result["interval"] = record["_id"]["interval"].encode()
    result["date"] = record["_id"]["date"].encode()
    result["img"] = image_to_string(result["image"])
    record = record_example(
        image=result["img"],
        energymeter=result["energy"],
        spectra=nullspectra,
        amp=get_encoded_amp(record["_id"]["amp"]),
        arm=get_encoded_arm(record["_id"]["arm"]),
        front=get_encoded_front(record["_id"]["front"]),
        timestamp=result["timestamp"],
        interval=result["interval"],
        date=result["date"]
        )
    return record
records = []
for rec in collection.find({'$and': [
  {"device.1": {'$exists': 'true'}},
  {"device": "Camera"}
]}
):
```

```
    records.append(process_record(rec))

write_features("eli-np.tfrecords", records)
```


ANEXA 10 – Generarea și antrenarea modelului ML

```
import os
import warnings
import sys
import tensorflow as tf
tf.config.run_functions_eagerly(True)
tf.data.experimental.enable_debug_mode()
import tensorflow_io as tfio
import os
import numpy as np
from functools import partial
import mlflow

import mlflow.sklearn

from mlflow.models.signature import infer_signature
filenames = ["../eli-np.tfrecords"]

pattern = "s3://***.tfrecords"
# Read the tfrecords
tfrecord_format = {
        "image": tf.io.FixedLenFeature([], tf.string),
        "energymeter": tf.io.FixedLenFeature([], tf.float32),
        "spectra": tf.io.FixedLenFeature([], tf.string),
        "amp": tf.io.FixedLenFeature([], tf.int64),
        "arm": tf.io.FixedLenFeature([], tf.int64),
        "front": tf.io.FixedLenFeature([], tf.int64),
        "timestamp": tf.io.FixedLenFeature([], tf.string),
        "interval": tf.io.FixedLenFeature([], tf.string),
        "date": tf.io.FixedLenFeature([], tf.string)
}

def _parse_and_filter(example_proto):
    data = tf.io.parse_single_example(example_proto,tfrecord_format)
    return tf.io.decode_jpeg(data["image"], channels=3)

AUTOTUNE = tf.data.AUTOTUNE

def load_dataset(filenames):
    ignore_order = tf.data.Options()
    ignore_order.experimental_deterministic = False
    dataset = tf.data.TFRecordDataset(filenames)
    dataset.with_options(ignore_order)
    parsed_ds = dataset.map(_parse_and_filter)
    return parsed_ds

dataset = load_dataset(filenames)
```

```
# Split 13941 records into 12547 training and 1394 validation
dataset_train = dataset.take(12500) # 12500 records
dataset_val = dataset.skip(12541) # 1400 records


# Fix autoencoder datafile
dataset = dataset.map(lambda x : (x, x))

initial_learning_rate = 0.01
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=20, decay_rate=0.96, staircase=True
)

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(
    "my_model.h5", save_best_only=True
)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(
    patience=10, restore_best_weights=True
)

class ConvAutoEncoder:
    @staticmethod
    def build(width, height, depth, filters=(32, 16), latentDim=16):
        inputShape = ( width, height, depth)
        chanDim = -1
        inputs = tf.keras.layers.Input(shape=inputShape)
        x = inputs
        for f in filters:
            x = tf.keras.layers.Conv2D(f, (3, 3), strides=1, padding="same")(x)
            x = tf.keras.layers.LeakyReLU(alpha=0.2)(x)
            x = tf.keras.layers.BatchNormalization()(x)
            volumeSize = tf.keras.backend.int_shape(x)
            x = tf.keras.layers.Flatten()(x)
            latent = tf.keras.layers.Dense(latentDim)(x)
            encoder = tf.keras.Model(inputs, latent, name="encoder")
        latentInputs = tf.keras.layers.Input(shape=(latentDim,))
        x = tf.keras.layers.Dense(np.prod(volumeSize[1:]))(latentInputs)
        x = tf.keras.layers.Reshape((volumeSize[1], volumeSize[2],
volumeSize[3]))(x)
        for f in filters[::-1]:
            x = tf.keras.layers.Conv2DTranspose(f, (3, 3), strides=1,
                padding="same")(x)
            x = tf.keras.layers.LeakyReLU(alpha=0.2)(x)
            x = tf.keras.layers.BatchNormalization()(x)
        x = tf.keras.layers.Conv2DTranspose(depth, (3, 3), padding="same")(x)
        outputs = tf.keras.layers.Activation("sigmoid")(x)
        decoder = tf.keras.Model(latentInputs, outputs, name="decoder")
```

```
        autoencoder = tf.keras.Model(inputs, decoder(encoder(inputs)),
                                     name="autoencoder")
        return (encoder, decoder, autoencoder)

def make_model():
    (_encoder, _decoder, autoencoder) = ConvAutoEncoder.build(102, 128, 3,
latentDim=4, filters=(16,))
    opt = tf.keras.optimizers.Adam(learning_rate=lr_schedule)
    autoencoder.compile(optimizer=opt, loss=tf.keras.losses.MeanAbsoluteError())
    return autoencoder


if __name__ == "__main__":
    warnings.filterwarnings("ignore")
    np.random.seed(40)

    mlflow.set_tracking_uri("https://mlf.bigdata.nipne.ro")
    mlflow.set_experiment("eli-np-basic")
    mlflow.tensorflow.autolog(every_n_iter=1)

    batch_size = float(sys.argv[1]) if len(sys.argv) > 1 else 1
    l1_ratio = float(sys.argv[2]) if len(sys.argv) > 2 else 0.5
    epochs = int(sys.argv[3]) if len(sys.argv) > 3 else 5

    model = make_model()
    print("fitting model")
    hist = model.fit(
        dataset.batch(int(batch_size)),
        epochs=epochs,
        batch_size=1,
        validation_data=dataset.batch(int(batch_size)),
        callbacks=[checkpoint_cb, early_stopping_cb],
        verbose=True
        )
    mlflow.tensorflow.log_model(model, "eli-np-basic")
    test = dataset.take(int(batch_size))
    predictions = model.predict(test, batch_size=int(batch_size))
    signature = infer_signature(test.numpy(), predictions.numpy())
    mlflow.tensorflow.log_model(model, "eli-np-basic", signature=signature)
```