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 Tehnic 2.1

Servicii și aplicații informatice pentru administrarea și monitorizarea centrului CLOUDIFIN, precum și pentru asigurarea accesului utilizatorilor la resurse

Autori: Mihnea Dulea
Dragoș Ciobanu-Zabet
Bianca Neagu
Robert Poenaru
Ionuț Vasile

Versiunea: Finală (2.2)
Data: 19.05.2021
Distributie: Internă
Cod document: CBD.RT-2.1

Rezumat: Proiectul CeCBiD-EOSC a fost implementat de către echipa Departamentului Fizică Computațională și Tehnologia Informației din IFIN-HH între anii 2020-2023. Acest document raportează rezultatele obținute în cadrul Subactivității 2.1 – *"Realizarea de servicii și aplicații informatice noi pentru administrarea și monitorizarea centrului CLOUDIFIN"*, desfășurată în perioada 19.05.2020 – 18.05.2021. Sunt prezentate: serviciul centralizat de monitorizare a resurselor; serviciul de alertare în caz de funcționare anormală a infrastructurii; aplicațiile dezvoltate pentru automatizarea unor proceduri de management al Centrului Cloud și Big Data; programul software de autodescoperire a topologiei aplicațiilor, serviciilor și a proceselor; sistemul de înregistrare a utilizatorilor și a proiectelor de calcul; interfața de acces a utilizatorilor la resursele centrului CLOUDIFIN.

Lista reviziilor

Data	Versiunea	Editor/Autor/Coautori	Sumarul modificărilor/completărilor principale
01.04.2021	0.1	M. Dulea	Structura raport, versiune preliminară Cuprins
09.04.2021	0.2	M. Dulea	Prefață, cap. 1, versiune finală Cuprins
23.04.2021	0.3	D. Ciobanu, B. Neagu, R. Poenaru, I. Vasile	Versiuni preliminare contribuții Cap. 2-6
30.04.2021	1.0	M. Dulea	Editare versiune preliminară raport
10.05.2021	1.1	D. Ciobanu, B. Neagu, R. Poenaru, I. Vasile	Versiuni finale contribuții Cap. 2-6
14.05.2021	1.2	M. Dulea	Adăugat capitole Rezumat și Concluzii
18.05.2021	2.0	M. Dulea	Editare finală și corecții
19.05.2021	2.1	M. Dulea	Versiune finală raport

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 sa 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 sa 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 si multiple infrastructuri de date federate angajate în furnizarea de servicii catre EOSC.

Începand din anul 2018, IFIN-HH a contribuit la implementarea infrastructurii EOSC prin intermediul Departamentului Fizică Computațională și Tehnologia Informației (DFCTI, [5]), care a participat, în calitate de asociat al coordonatorului, Fundația EGI [6], la proiectul H2020 EOSC-Hub (2018-2020, [7]), destinat dezvoltării resurselor și serviciilor Cloud inițiale pentru EOSC. Sarcina DFCTI a fost de a furniza, prin intermediul centrului Cloud CLOUDIFIN [8], 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 cercetatorii din toate disciplinele științifice care necesită calcule intensive și Big Data. Astfel, proiectele EGI-ACE si CeCBiD-EOSC actioneaza 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 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 Cloud-ul European pentru Stiinta Deschisa. [9]

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 către 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, capabilă 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 următoarele:

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 început la data semnării contractului de finațare de catre Ministerul Educației și Cercetării (19.05.2020), iar finalizarea lui este planificată pentru 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

Bibliografie

[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] Departamentul Fizică Computațională și tehnologii Informaționale (DFCTI), https://dfcti.ifin.ro

[6] Fundatia EGI, https://www.egi.eu/about/egi-foundation/

[7] *Integrating and managing services for the EOSC*, https://www.eosc-hub.eu/

[8] Centrul de resurse Cloud al DFCTI, CLOUDIFIN, http://cloudifin.ifin.ro/

[9] Cerere de Finantare, proiect CeCBiD-EOSC.

[10] The Elastic Stack - https://www.elastic.co/elastic-stack [Mai 2021]

[11] JVM (Java Virtual Machine) Architecture - https://www.javatpoint.com/jvm-java-virtual-machine

[12] Getting Started with Logstash - https://www.elastic.co/webinars/getting-started-logstash

[13] Key-Value Pairs Explained https://experienceleague.adobe.com/docs/audience-manager/user-guide/reference/key-value-pairs-explained.html?lang=en#reference

[14] Droplets - https://www.digitalocean.com/products/droplets/

[15] OpenStack Compute (nova) https://docs.openstack.org/nova/latest/

[16] NGINX Open Source - https://www.nginx.com

[17] What is a Reverse Proxy vs. Load Balancer? https://www.nginx.com/resources/glossary

[18] About certbot - https://certbot.eff.org

[19] R. Poenaru and D. Ciobanu-Zabet, "*ELK Stack – Improving the Computing Clusters at DFCTI Through Log Analysis*" 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet), Bucharest, Romania, 2020, pp. 1-8, doi: 10.1109/RoEduNet51892.2020.9324856.

[20] Watchdog, python package: https://pypi.org/project/watchdog/ [Mai 2021]

[21] Alert System, https://github.com/basavyr/kibana-log-alerts [Mai 2021]

[22] ZELK vs ELK: Zebrium vs Elastic Machine Learning - https://www.zebrium.com/blog/zelk-vs-elk-zebrium-ml-vs-elastic-machine-learning-zebrium [Mai 2021]

Lista figurilor

Rezumat

Scopul Raportului Tehnic

Scopul acestui raport tehnic este de a prezenta rezultatele obținute în cadrul Subactivității 2.1 a proiectului CeCBiD-EOSC, privind serviciile și aplicațiile informatice dezvoltate și implementate pentru administrarea și monitorizarea centrului CLOUDIFIN, precum și pentru asigurarea accesului utilizatorilor la resursele de calcul.

Impact

RT-2.1 este un livrabil cheie al proiectului, care descrie implementarea infrastructurii software necesară pentru realizarea obiectivelor de cercetare-dezvoltare prevazute în subactivitățile:

2.3 - *Realizarea de servicii si aplicatii informatice pentru satisfacerea cerintelor de calcul si de stocare de date din faza operationala initiala a proiectului ELI-NP*;

2.4 – *Realizarea de servicii si aplicatii informatice pentru suportul activitatii de modelare si simulare a nanostructurilor complexe*;

2.5 – *Realizarea de servicii si aplicatii informatice pentru suportul activitatii de analiza a datelor de secventiere de noua generatie*.

De asemenea, rezultatele cuprinse în prezentul raport stabilesc cadrul tehnic pentru inițierea subactivității 2.2 - *Federalizarea resurselor oferite de catre CLOUDIFIN si alte centre Cloud si dezvoltarea unei interfete web pentru accesul utilizatorilor la aceste resurse*.

Descrierea completă a activităților și a dependențelor dintre ele poate fi găsită in Cererea de Finanțare a proiectului CeCBiD-EOSC [9].

Conținutul Raportului Tehnic

După o scurtă introducere privind contextul general, necesitatea, obiectivele și rezultatele preconizate ale subactivității 2.1, în capitolul 2 se prezintă infrastructura hardware și software implementată pentru dezvoltarea și testarea aplicațiilor propuse. Capitolul 3 este consacrat descrierii rezultatelor privind programarea serviciului centralizat de monitorizare a resurselor și a serviciului de alertare în caz de funcționare anormală a infrastructurii. Aplicațiile informatice dezvoltate pentru automatizarea unor proceduri de administrare din cadrul centrului de date sunt prezentate in cap. 4. În capitolul următor este descris programul dezvoltat pentru autodescoperirea topologiei aplicațiilor, serviciilor și a proceselor. Capitolul 6 este dedicat sistemului de înregistrare și de acces a utilizatorilor la resursele CCBD.

Concluziile Raportului Tehnic

Subactivitatea 2.1 se încheie cu realizarea rezultatului planificat nr. 18 al proiectului - "*Servicii și aplicații informatice pentru administrarea și monitorizarea centrului CLOUDIFIN, precum și pentru asigurarea accesului utilizatorilor*".

Infrastructura software dezvoltată este esențială atât pentru înregistrarea și asigurarea accesului virtual al utilizatorilor la centrul CLOUDIFIN, cat și pentru managementul resurselor și al serviciilor de calcul și de stocare de date oferite acestora.

O atenție specială a fost acordată realizării condițiilor de disponibilitate și eficiență a serviciilor furnizate, prin programarea unor instrumente software avansate de monitorizare și de alarmare a personalului de administrare în cazul funcționării anormale.

Rezultatele obținute în cadrul subactivității 2.1 permit continuarea cu succes a subactivităților de cercetare-dezvoltare și abordarea realizarii unei soluții de interconectare a centrelor Cloud.

1. Introducere

1.1 Context general și necesitate

Obiectivul principal al proiectului CeCBiD-EOSC este modernizarea infrastructurii Cloud și extinderea infrastructurii Big Data în vederea creșterii contribuției și vizibilității IFIN-HH în cadrul Cloud-ului European pentru Știinta Deschisă (EOSC).

IFIN-HH este integrat în infrastructura EOSC ca furnizor de resurse și servicii pentru comunitatea științifică națională și internațională prin intermediul centrului CLOUDIFIN, care a fost certificat în anul 2017 în Cloud-ul Federativ al Infrastructurii Europene de Calcul Avansat EGI (https://www.egi.eu/federation/egi-federated-cloud).

CLOUDIFIN este găzduit și administrat în cadrul Centrului de Operațiuni al Infrastructurii Naționale de Calcul Științific Avansat (Ngi-RO, http://ngi-ro.ifin.ro/). Deși beneficiază, împreuna cu celelalte componente ale infrastructurii NGI-RO, de serviciile generale de monitorizare și acces asigurate la nivel european de catre EGI (Fig. 1.1), la nivel local CLOUDIFIN a necesitat realizarea de servicii și aplicații informatice noi pentru îmbunătățirea administrării și monitorizarii resurselor, precum și pentru automatizarea accesului utilizatorilor, în perspectiva creșterii capacității de suport a comunității științifice.

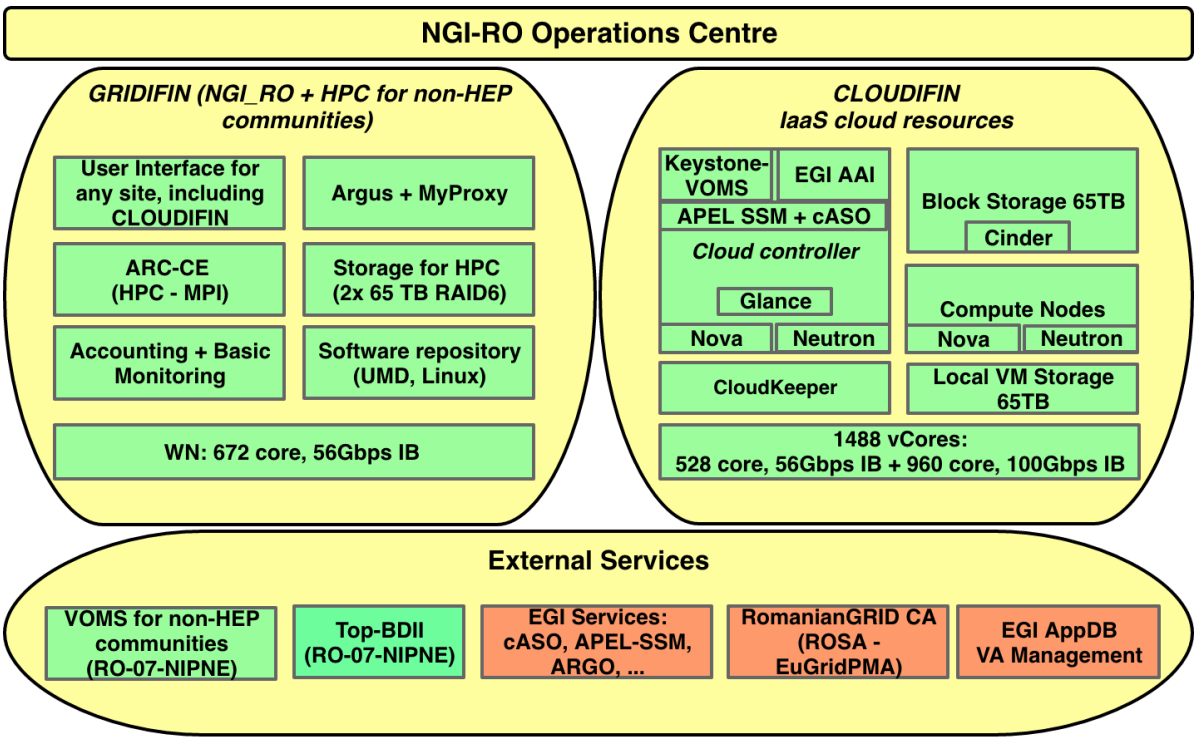


FIG. 1.1: Configurația Centrului de Operațiuni NGI-RO

Serviciile și aplicațiile dezvoltate in cadrul Subactivității 2.1 vor sta la baza dezvoltării soluției de interconectare a unor centre Cloud naționale într-un sistem federalizat, care va fi realizată în Subactivitatea 2.2. Prin aceasta se urmareste implementarea unui sistem de partajare coordonata a resurselor Cloud din țară dedicate cercetarii stiintifice, în special al centrelor Cloud care sunt finanțate prin proiectele POC propuse in cadrul apelului 398/2018, rezolvându-se astfel problema repartiției dezechilibrate a resurselor de calcul științific cu care se confruntă în prezent sistemul CDI național.

1.2 Obiectivele Subactivității 2.1

Obiectivele specifice ale Subactivității 2.1 sunt următoarele [8]:

1. Realizarea unui serviciu centralizat pentru preluarea, procesarea și reprezentarea grafica a datelor primite de la resursele Cloud si Big Data

Se colectează informații transmise de catre aplicatii si servicii, gazduite atat pe masini virtuale cat si pe masini fizice (din clusterul HPC existent). In urma filtrarii si analizei datelor primite de la clienti (aplicatii, servicii, etc), se definesc fluxuri de lucru atat pentru vizualizarea datelor cat si pentru actiuni necesare in caz de eroare/atentionare. In anumite cazuri cunoscute, fluxurile de lucru pot contine actiuni automate pentru corectarea / remedierea cauzei ce a provocat eroarea/atenționarea.

2. Dezvoltarea de aplicații informatice pentru administrarea centrului de resurse Cloud si Big Data

Se dezvoltă aplicații pentru: verificarea funcționării serviciilor; inventarierea resurselor hardware si software disponibile; instalarea la distanta pe nodurile de calcul a pachetelor software.

3. Dezvoltarea unei aplicatii software capabila de autodescoperirea topologiei aplicatiilor, serviciilor si a proceselor (fire de executie, conexiuni, dependinte software, etc.)

Aplicația dezvoltată trebuie să fie disponibilă atat pentru masini virtuale cat si fizice si sa ofere o imagine de ansamblu privind aplicatiile, serviciile si procesele in executie, unde si de catre ce utilizator sunt rulate, precum si catre cine s-au deschis conexiuni client-server.

4. Programarea interfetei web de acces a utilizatorilor la resursele CLOUDIFIN

Obiectiivul cuprinde dezvoltarea sistemului de acces la resurse, de inregistrare a utilizatorilor si a proiectelor de calcul, inclusiv aplicațiile back-end și baza de date aferentă.

1.3 Rezultate preconizate

- Implementarea unui serviciu de monitorizare centralizată a datelor de jurnalizare ale aplicațiilor și serviciilor de pe mașinile virtuale și fizice gazduite în DFCTI, inclusiv din clusterul CLOUDIFIN, precum și realizarea unei interfețe web pentru vizualizarea și analiza acestora.
- Dezvoltarea și implementarea unei aplicații capabilă să analizeze in timp real log-urile de monitorizare și să alerteze personalul administrativ cu privire la problemele/cauzele care au produs alertele respective.
- Aplicații informatice de automatizare a unor proceduri de administrare a centrului Cloud, privind funcționarea serviciilor pe nodurile de calcul, inventarierea resurselor hardware disponibile pentru utilizatori, inventarierea resurselor software disponibile, instalarea la distanță pe nodurile de calcul a pachetelor software.
- Program software capabil de autodescoperirea topologiei aplicatiilor, serviciilor și a proceselor.
- Infrastructura de înregistrare și autorizare a utilizatorilor și a proiectelor de calcul pentru care se solicită accesul la resursele CLOUDIFIN, cuprinzând interfața web de acces, aplicațiile back-end și baza de date aferentă acestora.

2. Mediul de dezvoltare și testare a aplicațiilor software

Pentru a face posibila realizarea si testarea serviciilor software de monitorizare a centrului CLOUDIFIN a fost dezvoltata o solutie pilot de tip Cloud bazata pe platforma de virtualizare OpenStack.

Platforma pilot care este monitorizata consta dintr-un nod central si 11 noduri de calcul interconectate prin retea lnfiniband, gazduite in cadrul clusterului Cloud existent.

Sistemul de operare instalat pe controller-ul Cloud și pe noduri a fost la data desfasurarii acestei activitati Linux CentOS 7.x.

Au fost instalate si configurate serviciile OpenStack care realizează controlul accesului la resurse, managementul mașinilor virtuale (VM-urilor) și al stocării de date, managementul imaginilor pentru VM-uri, precum și managementul rețelelor de comunicare de date între noduri si VM-uri.

Limbajul de programare principal utilizat a fost Python.

Pentru programarea aplicației web de acces a utilizatorilor la resursele Cloud a fost necesară instalarea de module și librării software specifice, inclusiv Flask, Jinja2, Flask-SocketIO și Python3.4+.

S-a instalat modulul Python Paho-MQTT, responsabil cu difuzarea, respectiv colectarea mesajelor primite de la clienți folosind protocolul de conectivitate mașina-la-mașina (M2M). Protocolul MQTT este conceput ca fiind unul de transport de mesaje de tip *publish/subscribe*.

Pentru a putea rula modulele Python dezvoltate in cadrul proiectului a fost necesara instalarea bibliotecii Flask de coduri si extensii Python.

Pentru afisarea informatiilor sortate si filtrate din mesajele de la clienti/aplicatii/senzori fara a fi necesara actiunea de actualizare a paginilor web, a fost instalat modulul Flask-SocketIO. Acesta ofera aplicatiilor Flask acces la comunicatii bidirectionale cu latenta redusa intre client si server.

De asemenea, pe clienți a fost necesară instalarea librariilor oficiale SocketIO dezvoltate în JavaScript, C++, Java și Swift.

A fost integrat în interfața web motorului de template-uri Jinja2, care este responsabil cu transmiterea dinamică a datelor din script-urile primite prin intermediul Python in fisierele HTML.

3. Serviciu centralizat de monitorizare a resurselor Cloud și Big Data

3.1 Obiectiv

Colectarea informațiilor transmise de catre aplicatii si servicii, gazduite atat pe masini virtuale cat si pe masini fizice (din clusterul HPC existent). In urma filtrarii si analizei datelor primite de la clienti (aplicatii, servicii, etc), se definesc fluxuri de lucru atat pentru vizualizarea lor cat si pentru actiuni necesare in caz de eroare/atentionare. In anumite cazuri cunoscute, fluxurile de lucru pot contine actiuni automate ce pot duce la corectarea/remedierea cauzei ce a provocat eroarea/atentionarea.

3.2 Serviciu centralizat de monitorizare a resurselor Cloud

Pentru preluarea, procesarea si reprezentarea grafica a datelor de monitorizare provenind de la diferite resurse computationale de tip Cloud, au fost implementate intr-o prima etapa serviciile oferite de stack-ul ELK (Elasticsearch + Logstash + Kibana, [9]).

Componenta Elasticsearch permite stocarea tuturor fisierelor de tip jurnal (*log*) intr-o forma ce asigura cautarea/analiza ulterioara facila a acestora.

Prin intermediul serviciului de tip JVM [10] (Java Virtual Machine), o instanta de Logstash [11] este capabila sa faca ingest de date din diferite surse (server local, server extern, resurse ce ruleaza in Cloud), pentru ca apoi acestea sa fie prelucrate si transmise catre stiva de stocare din Elasticsearch. Prelucrarea implica manipularea unor perechi de date de tipul "key-value" [12] astfel incat citirea acestora de catre Kibana (care ofera interfata grafica pentru analiza de log-uri) sa fie cat mai usor de realizat.

Aceste log-uri contin informatii despre anumite procese ce ruleaza sau vor fi initiate, procese care au fost intrerupte accidental din diferite cauze, starea resurselor, a conexiunii, etc. Analiza acestor date este cruciala atat pentru imbunatatirea functionarii resurselor de calcul, dar si pentru prevenirea unor potentiale probleme ce pot aparea dupa o utilizare indelungata a nodurilor de retea.

Acest pipeline a fost implementat mai intai pe o masina virtuala (VM) ce ruleaza un sistem de operare de tip RHEL: CentOS 7. Pentru testare s-au folosit mai multe surse de ingest de date: masini virtuale create on-demand prin servicii de cloud dedicate (i.e., DigitalOcean [13]), un cluster de tip Nova [14] din cadrul departamentului si diferite computere personale (PC) ce rulau pe reteaua locala. In toate dintre aceste cazuri, preluarea de log-uri a fost efectuata fara inrerupere, putand fi analizate in timp cvasi-*real* detalii legate de starea resurselor respective (CPU, RAM, etc.). Fluxul de lucru din cadrul testului este reprezentat schematic in Fig. 3.1.

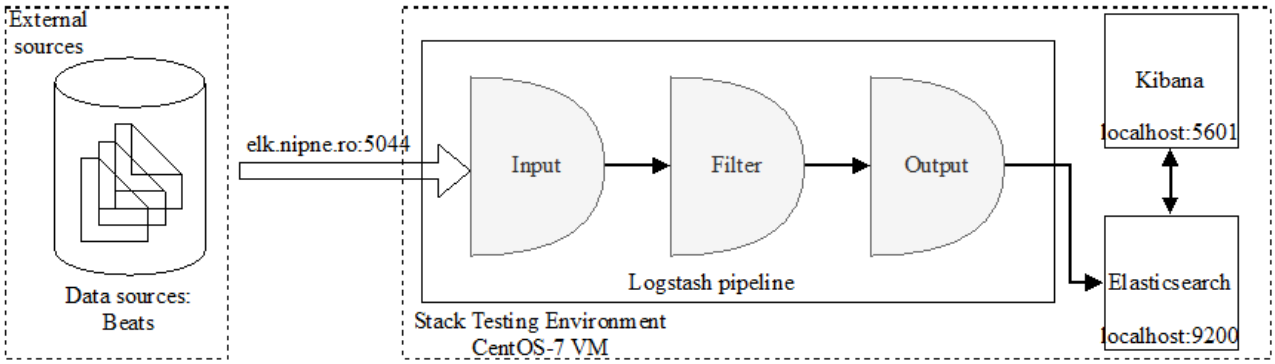


FIG. 3.1: Schema modului de lucru al stack-ului ELK pe masina virtuala de test

Prin intermediul unui serviciu de tip web-server, NGINX [15], s-a dezvoltat un website prin care se poate accesa Kibana. Astfel, cu ajutorul unui browser, s-a putut face o analiza / vizualizare a tuturor resurselor ce sunt preluate de catre stack-ul ELK.

In mod normal, o instanta Kibana poate fi accesata pe sistemul pe care a fost instalat (prin *localhost*), insa printr-un procedeu de *reverse-proxy* [16] implementat pe VM-ul de test, Kibana poate fi accesat de pe orice dispozitiv ce suporta un browser si o conexiune stabila la internet. Astfel, un utilizator poate analiza log-urile parsate de oriunde, fara a fi necesara conectarea acestuia la reteaua interna.

Tot prin intermediul serviciului NGINX a fost dezvoltat un website de tip static (accesibil la adresa https://elk.nipne.ro), in care au fost enumerate toate procedurile de set-up al stack-ului ELK. De asemenea, toate fisierele de configuratie pentru Elasticsearch, Logstash si Kibana sunt disponibile pe GitHub (link-uri acesibile pe pagina mentionata anterior). Website-ul suporta HTTPS si prezinta certificat de securitate SSL oferit prin Certbot [17]. Interfata grafica oferita de Kibana este reprezentata in Fig. 3.2.

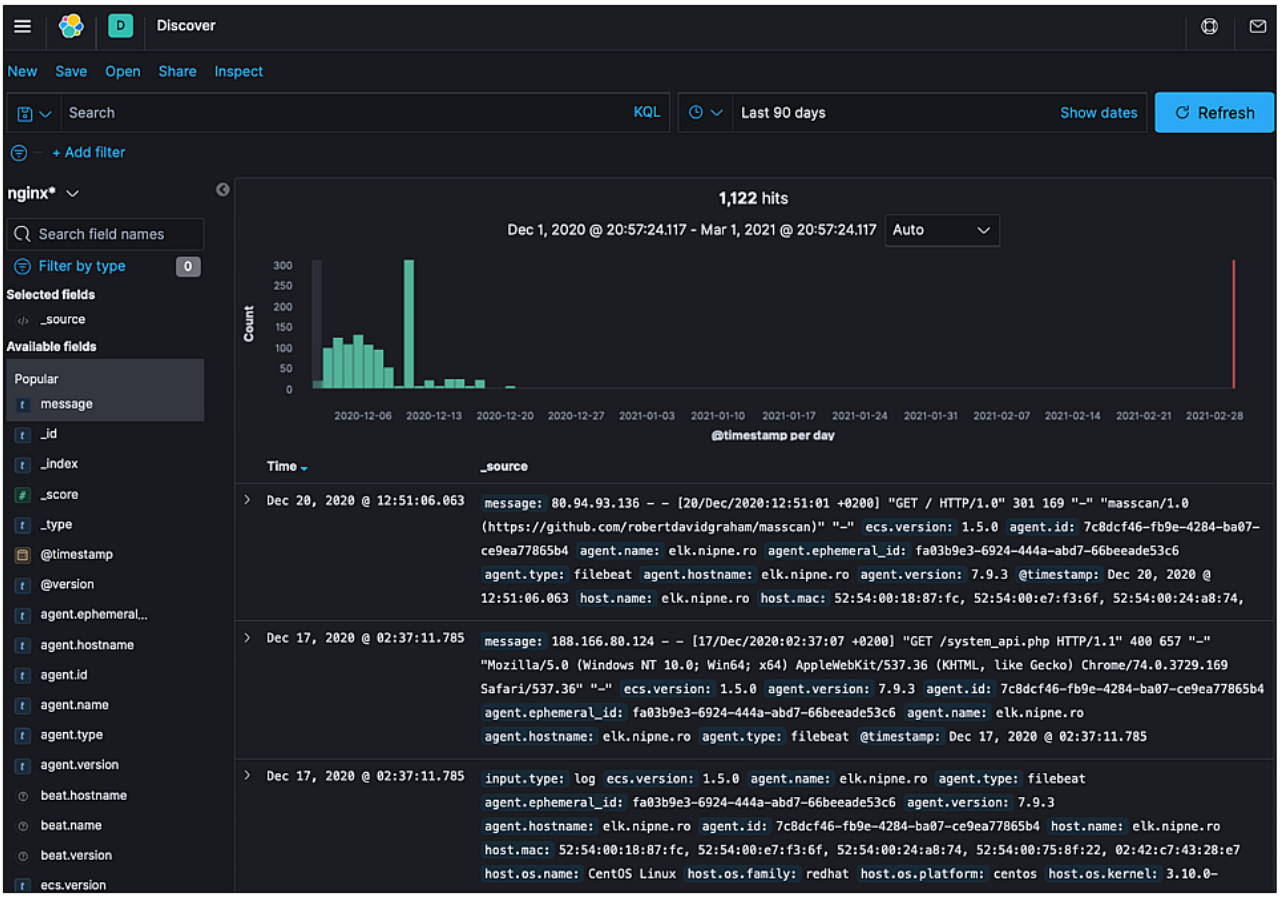


FIG. 3.2: Interfața grafică de monitorizare si analiza a log-urilor [18]

O descriere detaliata a intregului procedeu de set-up (ca si etapa de test) pentru stack-ul ELK a fost expusa in articolul [18] si prezentata in cadrul conferintei IEEE *19th RoEduNet Conference: Networking in Education and Research (RoEduNet).*

Pentru dezvoltarea ulterioară, se intenționează testarea soluției ELK prin creșterea numărului de surse de ingest (putând astfel analiza scalabilitatea serviciului) și, de asemenea, implementarea unui serviciu ZELK [21], ce ofera analiză de date bazată pe Machine Learning. Acest upgrade va mări rata de detectie a anomaliilor in functionarea resurselor monitorizate.

3.3 Serviciu de monitorizare si alertare pentru clustere de resurse de calcul

Fiecare resursa de calcul avansat din DFCTI poate efectua operatii complexe specifice sau poate rula diferite (micro)servicii (de exemplu, aplicatii de tip container cloud, librarii pentru dezvoltarea de interfete web, framework-uri de mentenata etc). Aceste procese necesita un timp de executie arbitrar de lung, de la cateva ore pana la o utlilizare continua pe perioada nedefinita (spre exemplu, un serviciu web-server precum NGINX este necesar sa ruleze permanent, fara intreruperi, astfel incat site-ul web pe care il gazduieste sa nu aiba timpi morti).

In timpul executiei de procese/calcule pe diferitele masini, se genereaza o cantitate mare de date de jurnalizare in fisiere de tip log (log-ul este un jurnal creat in timp real de catre fiecare proces in parte, in care diferitele fire de executie salveaza mesaje de avertizare sau mesaje de eroare cu privire la problemele ce au aparut). Analiza acestor log-uri este cruciala pentru a intelege cauzele care au dus la aparitia de probleme in rularea proceselor respective. Astfel, este necesara dezvoltarea unei proceduri de colectare, de stocare si, nu in ultimul rand, de analizare a log-urilor generate de catre intregul cluster de masini de calcul.

Intrucat fluxul de date log generate de la un cluster poate fi extrem de mare, un proces de automatizare al workflow-ului *ingest->store->analyse* este absolut necesar. Suita de aplicatii ELK [9] (Elasticsearch + Logstash + Kibana) ofera aceasta functionalitate. Descrierea implementarii ELK pe o masina virtuala din reteaua DFCTIi a fost facuta in capitolul precedent si publicata in lucrarea [18].

Interpretarea fisierelor de tip log de catre un utilizator poate fi o operatiune destul de complexa, mai ales daca structurarea informatiei este facuta intr-un format special (e.g., HTCondor, Apache, NGINX). Parsarea datelor dintr-un format arbitrar in unul usor interpretabil de tip *cheie-valoare (JSON)* este realizata prin Logstash.

3.3.1 Necesitate

Pe langa stocarea si analiza log-urilor primite de la resursele de calcul, o procedura automatizata de avertizare in timp real in cazul in care valorile observabilelor ies din intervalul de "functionare normală" poate asigura o mentenanta eficienta a intregului cluster pe care se ruleaza diferite calcule sau servicii.

Crearea de alerte in situatii critice nu poate fi asigurata non-stop de catre personalul administrativ (sysadmins); astfel, un serviciu automat de alertare va facilita acest lucru pentru buna functionare a serverelor/clusterelor, inclusiv a clusterului CLOUDIFIN.

Acest deziderat sta la baza implementarii unei aplicatii capabila sa analizeze in timp real log-uri si sa alerteze prin e-mail personalul administrativ cu privire la problemele/cauzele care au produs alertele respective.

3.3.2 Dezvoltarea unei aplicatii Python de alertare

Deoarece este extrem de flexibil, robust si scalabil, limbajul de programare Python a fost ales pentru dezvoltarea aplicatiei. La aceste caracteristici se adauga faptul ca Python este open-source si compatibil cu orice arhitectura si sistem de operare cunoscut.

In prima faza, pentru testarea serviciului de alertare, sunt utilizate doar log-urile privind resursele de sistem al fiecarei masini de calcul:

- procentaj utilizare CPU (CPU_USAGE)
- procentaj utilizare memorie RAM (MEM_USAGE)

Intregul procedeu de dezvoltare al serviciului de alertare poate fi descompus in urmatorii pasi:

- Crearea unui modul ce ofera posibilitatea de a trimite e-mail-uri catre o lista de clienti predefinita.
- Crearea unui modul pentru a realiza fisiere de output cu diferite date, care vor fi transmise ca atasamente la mail-urile de alertare.
- Implementarea unui modul care citeste fisierele log stocate pe masina virtuala in care este configurat si stack-ul ELK, si care mai apoi parseaza log-urile intr-un format similar cu cel configurat de Logstash. Din aceasta etapa se extrag valorile variabilelor de interes CPU_USAGE si MEM_USAGE.
- Crearea unui modul ce realizeaza o reprezentare grafica de tipul *line-plot,* in care valorile analizate (CPU/MEM_USAGE) sunt afisate in raport cu o perioada de timp setata in prealabil (de exemplu, ultimele 5 minute).Aa

3.3.3 Modulul Watch_Process()

Integrat in clasa *Reader* din codul sursa, modulul are ca rol monitorizarea continua a fisierelor log de interes, provenite de la orice resursa de calcul. Modulul este dezvoltat cu ajutorului pachetului **watchdog** [19]. Acest pachet ofera posibilitatea de a monitoriza orice fisier in timp real, iar in cazul oricarei modificare suferite de fisierul respectiv, watchdog va trimite un mesaj de avertizare, afisand ultima modificare. Cu fiecare eveniment nou ce soseste intr-unul din fisierele log monitorizate (watchdog suporta monitorizarea mai multor fisiere simultan), acesta va fi parsat, si va fi introdus in stive (Python *arrays*) pentru fiecare observabila analizata.

Pentru a adopta pachetul watchdog in codul sursa, a fost necesara utilizarea urmatoarelor module:

```
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import watchdog.events as eventer
```

In figura de mai jos, se poate vedea fluxul de lucru realizat de catre modulul Watch_Process.

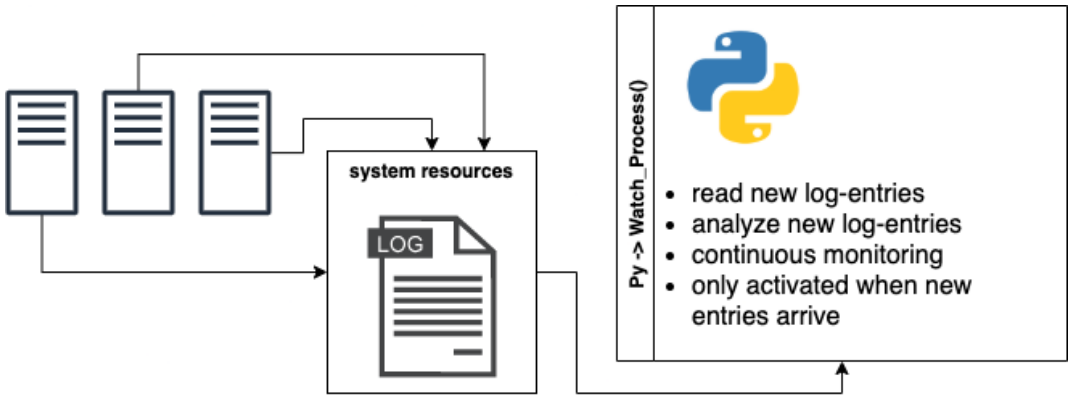


FIG. 3.3: Monitorizarea in timp real de catre watcher a datelor colectate in log-ul de pe nodul principal.

Procedeul de start al modulului se realizeaza prin codul urmator:

```
try:
      file_event_handler = Modified_State_Handler(log_file_path)
      except Exception as error:
            return -1
      else:
```

```
            pass
      try:
            observer = Observer()
      except Exception as error:
            return -1
      else:
            pass
      try:
            observer.schedule(file_event_handler,
                              path=log_file_path, recursive=True)
      except Exception as error:
            return -1
      else:
            pass
      # start the observer
      time.sleep(1)
      if(DEBUG_MODE):
            print(f'Starting the log file observer...')
      if(DEBUG_MODE):
            print(f'The reader process will refresh its cycle each {cycle_time}
      seconds...')
      observer.start()  # <---------------- command that starts the actual
watcher for the log-file
      #  keep track of the total execution time ⬇
      total_execution_time = time.time()
      # keep track of the total execution time ⬇
cycler = time.time()
```

Seturile de date ce sosesc intr-un fisier log sunt adaugate in stive prin modificarea unui EventHandler. Acesta actioneaza ca o functie de raspuns la modificarea starii unui fisier. Codul necesar pentru crearea de stive in care variabilele CPU_USAGE si MEM_USAGE sunt introduse este desfasurat mai jos:

```
class Modified_State_Handler(FileSystemEventHandler):
    def __init__(self, log_file_path):
        self.log_file_path = log_file_path


    def on_modified(self, event):
        event_path = event.src_path
        # check if the event was triggered by a file
```

```
        # print(f'OS: {Get_OS()}\nLog-File-Path: {event_path}')
        # print(f'New log-event in -> {event_path}')
        if(os.path.isfile(event_path) and event_path == self.log_file_path):
            # print(f'OS: {Get_OS()}\nLog-File-Path: {event_path}')
            # print(f'New log-event in -> {event_path}')
            # easy two-liner for getting the last line of the log-file
            with open(self.log_file_path, 'r') as reader:
                last_line = list(reader)[-1]
            try:
                # must append only the value of the cpu or memory
                cpu_stack.append(Reader.get_cpu_usage(last_line))
            except Exception as error:
                print(f'could not add CPU stats into the cpu stack')
                print(f'Reason -> {error}')
            else:
                pass
            try:
                # must append only the value of the cpu or memory
                mem_stack.append(Reader.get_mem_usage(last_line))
            except Exception as error:
                print(f'could not add MEM stats into the cpu stack')
                print(f'Reason -> {error}')
            else:
                pass
            try:
                if(len(machine_id) == 0):
                    machine_id.append(Reader.get_machine_id(last_line))
            except Exception as error:
                print(f'could not get machine ID')
                print(f'Reason -> {error}')
            else:
                pass
```

Codul contine segmente de tipul "`try/except`" ce au rolul de a preveni intreruperea aplicatiei pentru cazul in care apar probleme cu citirea valorilor din fisierul log. De asemenea, variabila "`machine_id`" este utilizata pentru a putea face diferenta intre resursele de sistem analizate de pe toate masinile de calcul aflate pe cluster.

3.3.4 Clasa Stats_Analyzer()

Aceasta clasa contine metodele necesare pentru analiza observabilelor de interes, si compararea acestora cu "valorile normale".

Domeniile valorilor normale ale parametrilor vor fi definite in prealabil de catre utilizatorul aplicatiei prin intermediul unui set de valori ca mai jos:

```
# thresholds are implemented as a dictionary, for easier manipulation
    thresholds = {"cpu": 80, "mem": 80}
```

In exemplul de mai sus, un procentaj de utilizare mai mare de 80% pentru oricare din observabile va fi considerat comportament neobisnuit, fiind necesara ridicarea unei alerte. Urmatoarele metode au fost implementate in clasa *Stats_Analyzer*:

```
@classmethod
    def Valid_Stacks(cls, system_stacks, valid_size):
```

Are rolul de a preveni analiza unor stive in care nu au mai sosit valori noi timp de 60 de secunde. Acest lucru indica faptul ca masina de calcul respectiva s-a oprit, sau conexiunea prin care are loc transferul de log-uri a incetat

```
@classmethod
      Def Analyze_CPU_Usage_Stack(cls, cpu_usage_stack, cpu_threshold):
```

Analizeaza stiva asociata observabilei CPU_USAGE. Calculeaza media valorilor din stiva, si se compara cu limita setata in prealabil.

```
@classmethod
      def Analyze_MEM_Usage_Stack(cls, mem_usage_stack, mem_threshold):
```

Analizeaza stiva asociata observabilei MEM_USAGE. Calculeaza media valorilor din stiva, si se compara cu limita setata in prealabil.

```
@classmethod
    def Stack_Report(cls, stack, stats_details, file_stack):
```

Salveaza toate valorile din stive intr-un fisier de output text, pentru a fi atasat intr-un mesaj de alerta.

```
@classmethod
    def Plot_Stack(cls, time_stamp, machine_id, failed_stack, time, threshold,
plot_stack_file, labels):
```

Realizeaza reprezentari grafice cu utilizarea CPU-ului sau memoriei RAM pentru un interval de timp dat. Graficele vor fi atasate in mesajul de alerta. Reprezentarea grafica s-a facut utilizand pachetul **matplotlib.**

Fluxul de lucru al clasei Stats_Analyzer() este urmatorul:

1. Fiecare observabila nou aparuta intr-un fisier log va fi adaugata in stiva corespunzatoare (CPU_USAGE si MEM_USAGE)
2. Dupa un anumit timp (fixat in prealabil tot de catre utilizator, prin variabila "`cycle_time`"), se va analiza fiecare stiva.
    a. Un eveniment nou in fisierul log este generat, in medie odata la 1-2 secunde.

   b. In functie de marimea variabilei "`cycle_time`", stivele vor avea o dimensiune direct proportionala cu aceasta, si de asemenea proportionala cu rata de aparitie a noilor evenimente in fisierul log.

3. Analiza unei stive (spre exemplu CPU_USAGE) va consta in comparatia valorii medii obtinute prin aplicarea functiei Analyze_CPU_Usage_Stack cu parametrul setat in viariabila thresholds. Similar si pentru MEM_USAGE, se va face o comparatie a mediei valorilor cu limita data de thresholds.
4. In cazul in care mediile acestor observabile vor fi mai mari decat limitele admise, **va fi necesara ridicarea unei alerte.**

3.3.5 Clasa Alerter()

Implementarea alertei consta in trimiterea unei atentionari prin e-mail catre o lista de clienti, in cazul in care analiza in timp real a stivelor ce corespund observabilelor de interes a produs un rezultat ce depaseste limitelor setate de utilizator (prin variabila "`thresholds`").

Pentru dezvoltarea functiilor ce trimit date prin serviciul e-mail este necesara utilizarea modulelor python descrise mai jos:

```
import email
import smtplib

import ssl

from email import encoders

from email.mime.base import MIMEBase

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText
```

Urmatoarele caracteristici are clasei *Alerter()* trebuie mentionate:

- Este necesara o adresa de e-mail care sa suporte receptionarea de mesaje prin protocolul **smtp** (de exemplu Gmail).
- Transmiterea textului va fi securizata (textul si atasamentele sunt encodate in base64) inainte de a fi trimise catre client.
- Atasamentele sunt create on-the-fly, in momentul in care mesajul va fi trimis.
- Fiecare alerta trimisa are un cod unic, ce poate fi identificat in text.

In figura de mai jos este redat un exemplu de alerta emisa dupa ce un comportament neobisnuit al resursei RAM a fost identificat pe una din masinile de test.

In exemplu, timpul de urmarire al fisierelor log (variabila "`cycle_time`") a fost setata la 60 de secunde. Aceasta inseamna ca watcher-ul implementat va analiza stivele odata la un minut.

Limita de utilizare pentru memoria RAM a fost setata la 50%.

Valorile stivelor sunt salvate in format text in primul fisier PDF atasat la mesaj (*mem_failed_stack.pdf*), iar reprezentarea utilizarii memoriei RAM de catre masina in cel de-al doilea attachment (*mem_usage.pdf*).

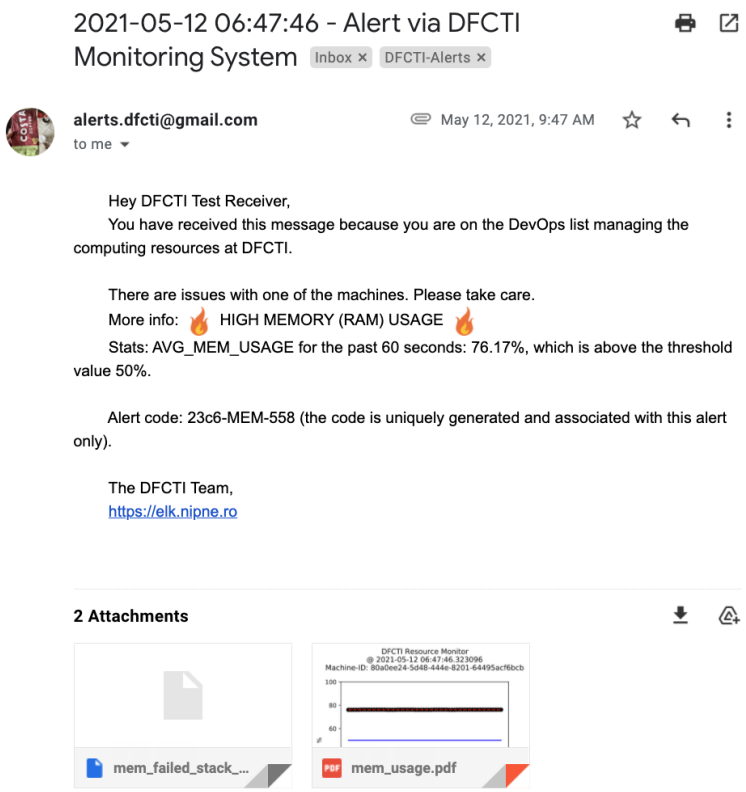


FIG. 3.4: Exemplu de mesaj de alertare

Un exemplu de reprezentare grafica pentru gradul de utilizare al CPU-ului este afisat in figura urmatoare:

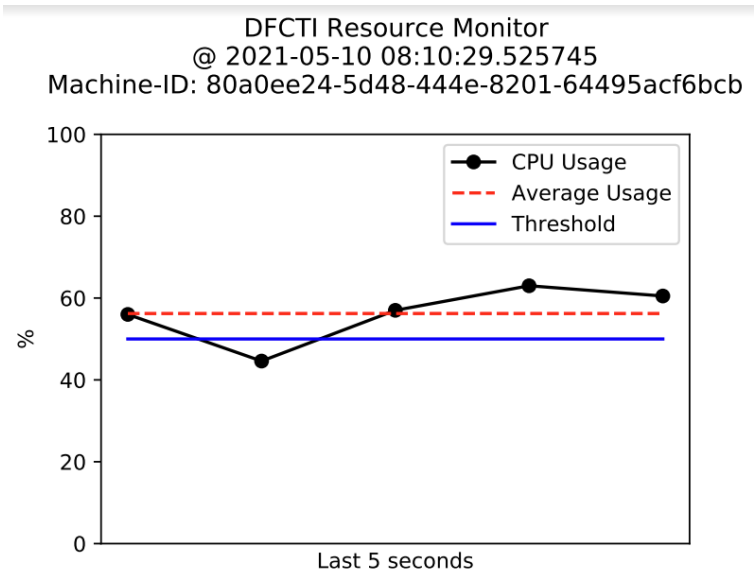


FIG. 3.5: Reprezentarea grafica a gradului de utilizare al CPU

În figura de mai sus este afisată utilizarea procesorului (*CPU_USAGE*) pentru una dintre mașini in ultimele 5 secunde. În acest exemplu, timpul de urmarire al watcher-ului a fost setat la cinci secunde, iar limita pentru CPU a fost setata la 50%. Alerta, evident a fost creata deoarece in ultimele cinci secunde, masina a avut un CPU_USAGE peste valoarea permisa.

Pentru crearea atașamentelor sunt utilizate următoarele metode implementate în clasa *Attachement* și, respectiv, *Alerter*.

```
Attachment.Create_Attachment(f'{datetime.utcnow()}---->
MEM_FAIL_STACK{adjusted_mem_stack}\n{mem_fail_value}',attach_filenames)

Alerter.SendAlert(alert, attach_filenames, email[1])
```

Intregul workflow pentru serviciul de alertare dezvoltat poate fi vizualizat in schema de mai jos:

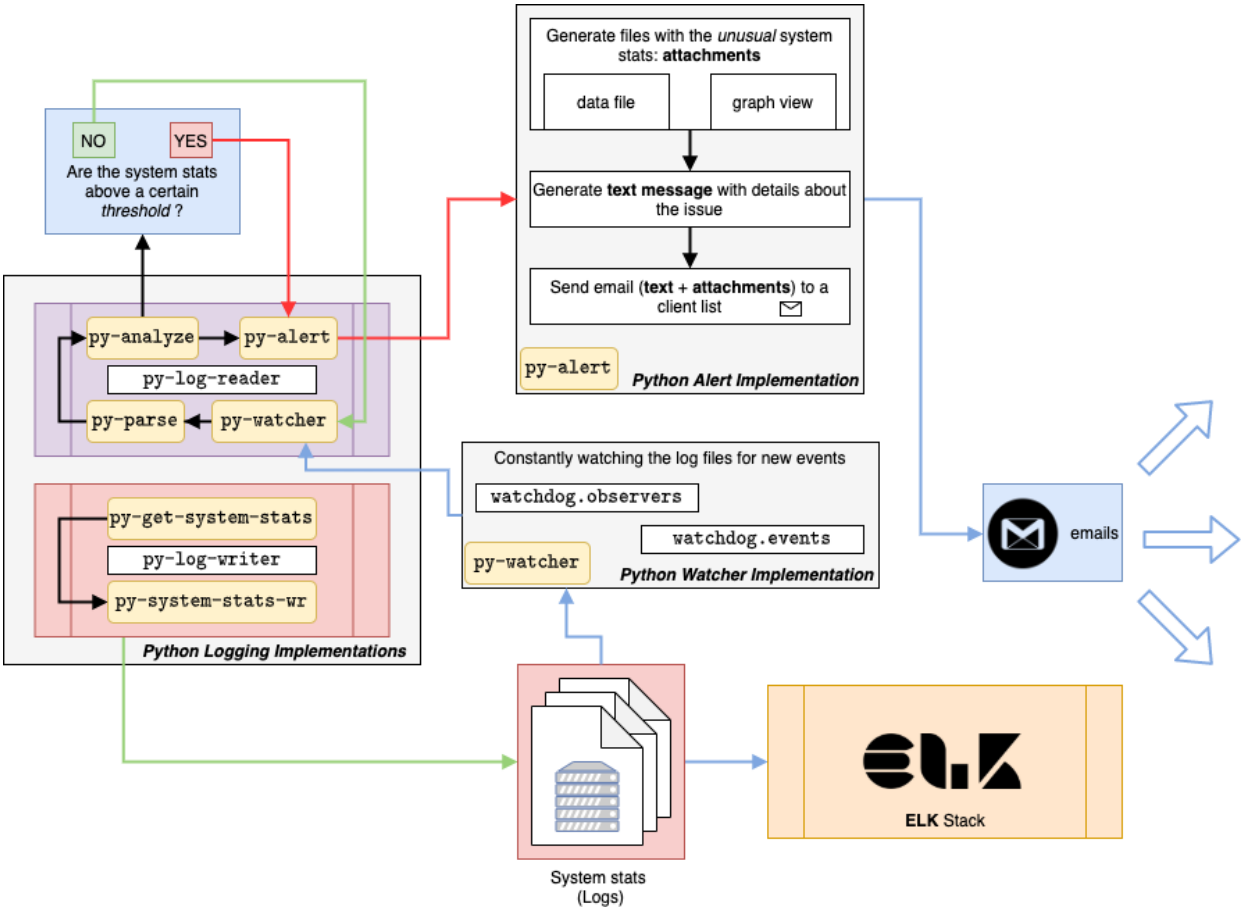


FIG. 3.6: Schema fluxului de lucru pentru serviciul de alertare

Codul sursa al aplicatiei (toate clasele, functiile si variabilele) este disponibil pe GitHub [20] https://github.com/basavyr/kibana-log-alerts/blob/main/log-reader/dfcti_log_reader.py .

3.4 Concluzii

- Pentru preluarea, procesarea si reprezentarea grafica a datelor de monitorizare provenind de la diferite resurse computationale de tip Cloud au fost implementate serviciile oferite de stack-ul ELK (Elasticsearch + Logstash + Kibana).
- Pentru analiza datelor de monitorizare, a fost dezvoltata in limbaj Python o aplicatie ce ruleaza sub forma unui script, compatibil cu sistemele de operare Linux si macOS, capabila sa ruleze si in mod "container" prin Docker.
- Aplicatia analizeaza fisierele log generate de anumite masini de calcul de pe un cluster (fiecare masina cu un ID unic), si le parseaza intr-un mod similar serviciului Logstash (configurat in stack-ul ELK).

- Analiza de log-uri identifica eventuale situatii in care valorile parametrilor de utilizare a resurselor de sistem precum CPU sau RAM depasesc limitele admise.
- In cazul identificarii situatiilor critice, aplicatia Python trimite alerte prin e-mail catre o lista de clienti, in care sunt prezentate detalii despre problemele aparute pe masinile monitorizate.
- Prin caracteristicile sale aplicatia contribuie la asigurarea functionarii optime a resurselor de calcul in regim de productie.

4. Aplicații informatice pentru administrarea CCBD

4.1 Obiectiv

Activitatea a avut drept scop dezvoltarea de aplicatii software pentru automatizarea unor proceduri utilizate frecvent în administrarea centrului CLOUDIFIN.

Acestea vizează:

- verificarea funcționării serviciilor pe nodurile de calcul;
- inventarierea resurselor hardware disponibile pentru utilizatori;
- inventarierea resurselor software disponibile;
- instalarea la distanta pe nodurile de calcul a pachetelor software.

4.2 Implementare

Comenzile de management al aplicațiilor sunt transmise de pe controller. Pentru exemplificare, se consideră mai jos instalarea de pachete software, în care caz acestea sunt de forma urmatoare:

```
mosquitto_pub -h ctrl-os -m '{"Host": "dual-c", "Category": "Manage-Apps",
"Commands": "yum install wget", "State": "init"}' -t "apps/messages/dual-c"
```

Dupa difuzarea comenzii, se face inserarea in baza de date CecBid – tabela CecBid de pe client (dual-c).

Conținutul înregistrării în baza de date dupa lansarea comenzii de mai sus este urmatorul:

id autoincrement	Host	Category	Commands	State	Date_n_Time
7	dual-c	Manage-Apps	yum install wget	init	17-May-2021 08:48:05:356117

Daemon-ul (procesul care ruleaza in background) verifica periodic daca sunt înregistrări noi in continutul bazai de date.

```
#----------------Manage_Apps-----------
conn = sqlite3.connect('/root/Proiect/CecBid.db')
conn.text_factory = str
c = conn.cursor()
c.execute("SELECT * FROM CecBid WHERE State == 'init'")
rows = c.fetchall()
for row in rows:
        os.system(str(row[3]))
        c.execute('UPDATE CecBid SET State = ?  WHERE Commands = ? AND State =
?', ('done', row[3], 'init'))
        value = json.dumps({"Host": "dual-c", "Category": row[2], "Commands":
row[3], "State": "done"});
        client.publish("apps/messages", value);
        time.sleep(1)
```

Daemon-ul verfica valoarea din campul "State". Daca aceasta este "init" (initializare), este/sunt extrasa/e comanda/comenzile de la adresa row[3].

Comanda este executata în Linux shell prin apelarea metodei python os.system(). Dupa executarea comenzii, se face update-ul in baza de date de pe client prin modificarea parametrului "State" din "init" in "done".

Inregistrarea din baza de date de pe client este in acest caz urmatoarea:

id autoincrement	Host	Category	Commands	State	Date_n_Time
7	dual-c	Manage-Apps	yum install wget	done	17-May-2021 08:49:15:987905

In final, starea comezii este transmisa si catre controller unde este salvata in baza de date de pe acesta:

id autoincrement	Host	Category	Commands	State	Date_n_Time
8	dual-c	Manage-Apps	yum install wget	done	17-May-2021 08:49:17:002202

Pentru administrarea site-ului CLOUDIFIN au fost dezvoltate aplicatiile software descrise mai jos.

4.2.1 Aplicația CheckServices

Aplicația *CheckServices* a fost dezvoltată pentru a verifica dacă un serviciu de pe nodurile de calcul este activ. Dacă în urma verificării se constată ca serviciul nu funcționează, sunt initializate comenzi de startare a acestuia iar output-ul comenzii de pornire este înregistrat în baza de date care este gazduita pe *controller*-ul cloud.

Aplicația poate rula și pe mașinile virtuale care sunt gazduite pe nodurile de calcul. Pentru a elimina o încărcare suplimentara a procesorului și a memoriei RAM de pe nodurile de calcul, parsarea și interpretarea output-ului sunt executate pe serverul principal.

Fluxul de lucru al aplicației de verificare și pornire automata (fara interventia administratorului site-ului cloud) a unui serviciu este urmatorul:

1. de pe serverul pricipal sunt lansate periodic probe de verificare a starii serviciului;
2. probele sunt difuzate prin intermediul protocolului de comunicatie *mqtt* (protocol care se ocupa de transportul mesajelor între nodurile de calcul) catre noduri;
3. proba de verificare a starii serviciului este executata pe nodul de calcul iar output-ul probei este trimis catre serverul principal, unde este înregistrat în baza de date;
4. output-ul primit este parsat si se verifica daca serviciul de pe nodul de calcul este pornit;
5. dacă serviciul de pe nodul de calcul este oprit, este inițializată comanda de pornire a serviciului;
6. după ce este pornit serviciul, în baza de date se înregistreaza informația cu privire la pornirea cu succes a serviciului respectiv.

4.2.2 Aplicația HardwareInventory

Pentru evidența resurselor ce pot fi alocate utilizatorilor a fost dezvoltata aplicatia *HardwareInventory*, care este executată pe fiecare nod de calcul, de unde colectează informațiile

privind resursele hardware disponibile pe nod, după care le transmite pentru a fi înregistrate în baza de date de pe serverul principal (*controller*-ul Cloud).

În baza de date sunt stocate informații privind:

- procentul de utilizare al procesorului (CPU);
- memoria RAM totală și disponibilă;
- capacitatea de stocare pe disc (HDD) totală și disponibilă.

Cu ajutorul acestor informatii stocate se poate cunoaste la orice moment de timp starea resurselor hardware pentru fiecare nod in parte.

4.2.3 Aplicația SoftwareInventory

Operațiunile de verificare a versiunii programelor ce ruleaza pe nodurile site-urilor de calcul se repetă destul de des în cazul software-ului *open source*, care necesită actualizări relativ frecvente din motive de îmbunătățire, de adăugare de noi funcționalități, sau de securitate. Aplicația *SoftwareInventory* automatizează operațiunile de verificare a versiunilor pachetelor software instalate pe noduri de calcul multiple.

Pentru inspectarea versiunii, de pe serverul principal (*controller*) sunt lansate periodic probe de verificare pe nodurile de calcul. Aceste probe sunt au forma unor comenzi care sunt stocate în baza de date de pe nodurile de calcul.

Programul python *run_apps.py* extrage din baza de date a fiecarui nod toate comenzile care au valoarea atributului de stare "`init`" și le executa. Dupa executarea comenzilor valoarea atributului de stare din baza de date este modificată in "done".

Output-urile comenzilor sunt apoi transmise catre modulul de parsare si interpretare care face apoi înregistrarea în baza de date de pe serverul principal cu output-ul filtrat.

Inreaga procedură descrisă mai sus este reprezentată in Fig. 4.1.

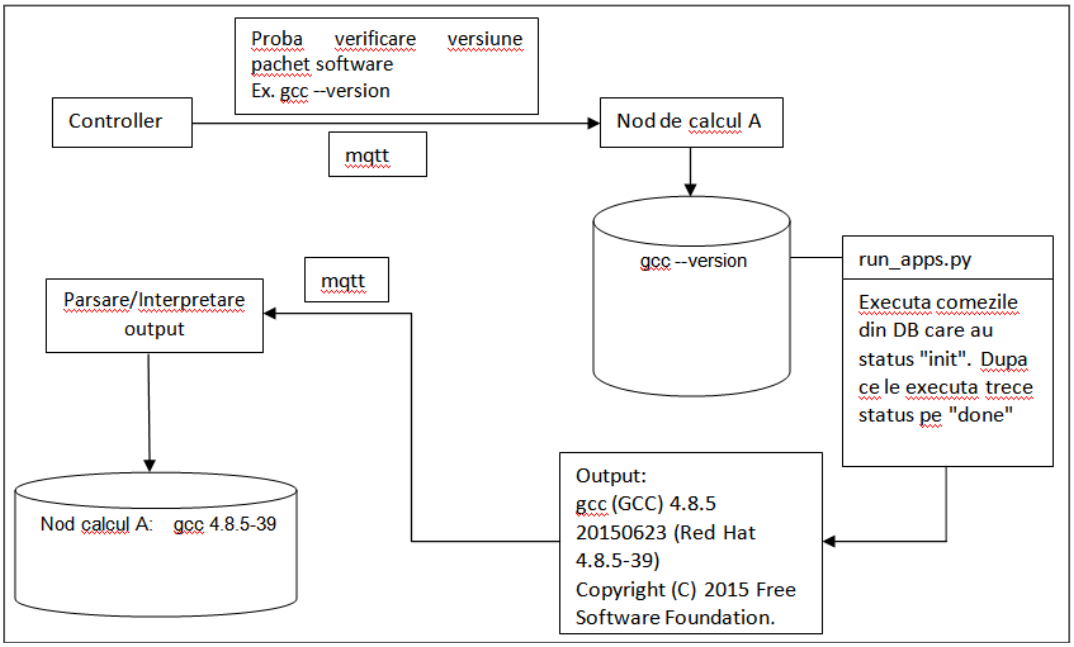


FIG. 4.1: Reprezentarea schematică a modului de acțiune al aplicației *SoftwareInventory*

Ca urmare a utilizării aplicației, în baza de date de pe *controller* exista un raport cu starea curentă a pachetelor software instalate pe fiecare nod de calcul. Acest raport poate fi examinat de către administratorul site-ului cloud în vederea actualizării pachetelor software respective.

În exemplul din Fig. 4.1, pentru inspectarea versiunii compilatorului gcc, se executa mai întâi de pe controller comanda

```
mosquitto_pub -h ctrl-os -m '{"Host": "dual-c", "Category": "Software_Inv", "Commands": "gcc --version", "State": "init"}' -t "apps/messages/dual-c"
```

Comanda difuzata de controller este stocata in baza de date CecBid - tabela `CecBid_Software_Inv` de pe client, avand atributul de stare "`init`".

Înregistrarea output-ului în baza de date a clientului se face prin apelarea functiei "`home_messages()`" listata mai jos:

```
def home_messages(jsonData):
        #Parse Data
        json_Dict = json.loads(jsonData)
        print json_Dict
        Category = json_Dict['Category]

        if Category == "Software_Inv":
                Host = json_Dict['Host']
                print "Host " + Host
                Category = json_Dict['Category']
                Commands = json_Dict['Commands']
                Arguments = json_Dict['Arguments']
                State = json_Dict['State']
                Data_and_Time = (datetime.today()).strftime("%d-%b-%Y %H:%M:%S:%f")

                #Push into DB Table
                dbObj = DatabaseManager()
                dbObj.add_del_update_db_record("insert into CecBid_Software_Inv (Host, Category, Commands, Arguments, State, Date_n_Time) values (?,?,?,?,?,?)",[Host, Category, Commands, Arguments, State, Data_and_Time])
                del dbObj
                print "Inserted Object Values into Database."
                print ""
```

În cadrul acestei funcții se verifică dacă variabila "`Category`" are valoarea "`Software_Inv`".

Dacă această condiție este îndeplinită, se face un insert nou al valorilor stringului JSON in baza de date de pe client.

Continutul inregistrarii in baza de date de pe client este urmatorul:

id autoincrement	Host	Category	Commands	Arguments	State	Date_n_Time
6	dual-c	Software_Inv	gcc --version		init	17-May-2021 08:48:08:656817

După ce sunt extrase inregistrarile cu atributul de stare "init", daemon-ul de pe nodul de calcul executa comenzile din cadrul atributului “Commands” din baza de date.

```
          #------------Software_Inv-------------
      c.execute("SELECT * FROM CecBid_Software_Inv WHERE State == 'init'")
      rows = c.fetchall()
      for row in rows:
        p = subprocess.Popen((row[3], row[4]), stdout=subprocess.PIPE,
shell=True)
         (output, err) = p.communicate()
         p_status = p.wait()
         output_end = output.split("\n")
         c.execute('UPDATE CecBid_Software_Inv SET State = ?  WHERE Commands = ?
AND State = ?', ('done', row[3], 'init'))
         value = json.dumps({"Host": row[1], "Category": row[2], "Commands":
row[3], "Arguments": row[4], "Output": output, "State": "done"});
         client.publish("apps/messages", value);
         time.sleep(1)
```

Dupa executia comenzilor linux shell, este facuta modificarea atributul de stare din “init” in “done”.

Continutul inregistrarii in baza de date de pe client se modifica dupa cum urmeaza:

id autoincrement	Host	Category	Commands	Arguments	State	Date_n_Time
6	dual-c	Software_Inv	gcc --version		done	17-May-2021 08:48:10:542817

Totodata, dupa modificarea facuta in baza de date de pe client este transmis un semnal catre controller ce contine si output-ul comenzii shell executata pe client. Acest string de date este stocat in baza de date de pe controller.

Continutul inregistrarii in baza de date de pe controller este urmatorul:

Id autoincrement	Host	Category	Commands	Arguments	Output	State	Date_n_Time
8	dual-c	Software_Inv	gcc --version		gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)\nCopyright (C) 2015 Free Software Foundation, Inc.\nThis is free software; see the source for copying conditions. There is NO\nwarranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.\n\n	done	17-May-2021 08:48:10:564378

4.2.4 Aplicația ManageApps

Actualizarea manuală a pachetelor software identice de pe noduri de calcul multiple de către administratorul resurselor devine nepractică în cazul site-urilor mari deoarece necesita accesul prin login pe fiecare nod de calcul și lansarea manuala a comenzii de upgrade.

Cu ajutorul aplicației *ManageApps* pachetele software se pot instala automat pe nodurile de calcul fără a mai fi nevoie ca administratorul resurselor să acceseze fiecare nod pentru a instala pachetele respective.

Fluxul de lucru al aplicației este următorul:

1. administatorul site-ului cloud specifică într-un raport ce pachete software trebuie instalate și pe ce noduri;
2. aplicația transmite acest raport către nodurile de calcul respective;
3. dupa transmiterea raportului, acesta este stocat in baza de date de pe serverul client cu valoarea atributului de stare "`init`", specificând astfel ca software-ul urmează să fie instalat;
4. fiecare nod de calcul vizat execută instalarea software-ului necesar;
5. la finalizarea operațiunii de instalare atributul de stare din baza de date de pe client este modificat din "`init`" in "`done`".)
6. fiecare nod trimite către controller un semnal de succes al operațiunii prin adaugarea in baza de date a valorii atributului de stare "`done`". Prin aceasta modificare, raportul nu mai este retransmis pentru executie pe nodurile de calcul.

4.3 Concluzii

În perspectiva creșterii numărului utilizatorilor site-ului CLOUDIFIN ca urmare a diversificării ofertei de resurse pentru deservirea unor noi comunitați științifice din EOSC, personalul de administrare a centrului necesită, pentru evitarea suprasolicitării, automatizarea procedurilor care au rată mare de repetiție, cum sunt instalările de pachete software pe noduri de calcul multiple, verificarea funcționării serviciilor, evidența resurselor disponibile.

În capitolul 4 se descriu soluțiile software elaborate în DFCTI pentru rezolvarea acestor probleme.

Acestea cuprind proceduri automate de verificare și (posibilă) intervenție asupra nodurilor de calcul multiple privind: managementul aplicațiilor; verificarea stării de funcționare a serviciilor; inventarierea resurselor hardware și software existente; actualizarea la distanță a pachetelor software.

Consecința imediata a utilizării aplicațiilor dezvoltate este scăderea simțitoare a timpului de lucru alocat procedurilor respective, micșorarea posibilității de producere a unor eroari umane, ceea ce contribuie la creșterea de ansamblu a eficienței activitatii de management.

5. Autodescoperirea topologiei aplicațiilor, serviciilor și a proceselor

5.1 Obiectiv

Dezvoltarea unei aplicatii software capabila de autodescoperirea topologiei aplicatiilor, serviciilor si a proceselor (fire de executie, conexiuni, dependinte software, etc.)

Aplicația trebuie să fie disponibilă atat pentru masini virtuale cat si fizice si va oferi o imagine de ansamblu privind aplicatiile, serviciile si procesele in executie, unde si de catre ce utilizator sunt rulate, precum si catre cine s-au deschis conexiuni client-server.

5.2 Implementare

A fost dezvoltată o aplicație software în limbajul de programare Python pentru descoperirea topologiei aplicațiilor, serviciilor și a proceselor din infrastructura de calcul avansat..

Aplicatia software are o structură modulară, fiecare modul fiind programat atât pentru a extrage informatii de la nodurile de calcul cât și pentru a lansa comenzi specifice administrării de la distanță, fără să necesite intervenția locală a administratorului pe serverul client.

În continuare sunt descrise modulele aplicației împreuna cu fragmente relevantei din codul sursă al acestora.

5.2.1 Disponibilitatea bibliotecilor și dependențelor software

Pentru ca aplicatiile software de simulare sa poata rula pe masinile din cloud/cluster sunt necesare o serie de librarii si dependinte software. S-a identificat utilitatea creerii un modul capabil sa afiseze lista librariilor software necesare rularii in conditii optime a software-ului de simulare. Modulul dezvoltat este capabil sa afiseze informatii primite de la mai multe masini fizice și/sau virtuale.

Pentru a putea rula si colecta informatii de la masinile client, sunt necesare modulului python *paho-mqtt* si aplicația "*mqtt_Listen_Data.py*", care a fost programată în cadrul proiectului în limbaj python și al cărei cod este reprodus mai jos într-un caz particular ales pentru exemplificare:

```
import paho.mqtt.client as mqtt
import subprocess
import json


x = subprocess.Popen(['ldd', 'namd2'], stdout=subprocess.PIPE)
output, err = x.communicate()


client = mqtt.Client()
client.connect("server_name",1883,60)
client.publish("apps/messages/client/ldd", output);
client.disconnect();
```

In acest exemplu de cod este executata comada "`ldd namd2`" care verifica dependintele librariilor software necesare pentru ca binarele namd2, epoch, siesta sa poata rula.

Rezultatul este transmis catre serverul *mqtt* (de mesagerie) care filtreaza si parseaza datele primite. A fost evitată metoda de parsare a rezultatelor la sursa, adica pe serverul client (nodul monitorizat), pentru a se evita astfel incarcarea serverului.

Mesajele clientilor sunt transmise pe canale de difuzare "`apps/messages/client/ldd`" catre server.

În exemplul prezentat mesajul trimis de client este de forma urmatoare:

```
\tlinux-vdso.so.1 =>  (0x00007ffd317f1000)\n
\tlibdl.so.2 => /lib64/libdl.so.2 (0x00007fd0517d3000)\n
\tlibtcl8.5.so => /lib64/libtcl8.5.so (0x00007fd0514ab000)\n
\tlibpthread.so.0 => /lib64/libpthread.so.0 (0x00007fd05128f000)\n
\tlibm.so.6 => /lib64/libm.so.6 (0x00007fd050f8d000)\n
\tlibmpi_cxx.so.1 => not found\n
\tlibmpi.so.1 => not found\n
\tlibstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007fd050c86000)\n
\tlibgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007fd050a70000)\n
\tlibc.so.6 => /lib64/libc.so.6 (0x00007fd0506a2000)\n
\t/lib64/ld-linux-x86-64.so.2 (0x00007fd0519d7000)\n
```

Pe server, fluxul de date primit este filtrat si parsat. Dupa eliminarea '\t' (tab) si '\n' (enter) dar si a spatiilor, datele ajung de forma urmatoare – un array cu multe elemente. Este cea mai convenabila metoda de a stoca intr-o variabila mai multe valori care, ulterior sunt apelate.

```
['linux-vdso.so.1', '=>', '', '(0x00007ffd317f1000)', 'libdl.so.2', '=>',
'/lib64/libdl.so.2', '(0x00007fd0517d3000)', 'libtcl8.5.so', '=>',
'/lib64/libtcl8.5.so', '(0x00007fd0514ab000)', 'libpthread.so.0', '=>',
'/lib64/libpthread.so.0', '(0x00007fd05128f000)', 'libm.so.6', '=>',
'/lib64/libm.so.6', '(0x00007fd050f8d000)', 'libmpi_cxx.so.1', '=>',
'not_found', 'libmpi.so.1', '=>', 'not_found', 'libstdc++.so.6', '=>',
'/lib64/libstdc++.so.6', '(0x00007fd050c86000)', 'libgcc_s.so.1', '=>',
'/lib64/libgcc_s.so.1', '(0x00007fd050a70000)', 'libc.so.6', '=>',
'/lib64/libc.so.6', '(0x00007fd0506a2000)', '/lib64/ld-linux-x86-64.so.2',
'(0x00007fd0519d7000)', '']
```

Fluxul de date transmis de catre client poate avea un interval de repetitie de ordinul secundelor / orelor. Acest interval poate fi setat de catre administrator. Posibilitatea de setare a intervalului de repetitie este util pentru a nu se incarca reteau de date inutil. Cu toate acesteal, pentru unele servicii importante este necesar ca acest intervalul sa fie mai scurt.

Pe server, elementele array-ului sunt introduse intr-un dictionar (colectie de date de tipul – key:value)

```
    soname1=li[0],
        path1=li[2],
        mem1=li[3]
```

Aceasta colectie de date este trimisa de server catre browserul client prin intermediul modulului python *Flask-SocketIO*. *Socket.IO* este o bibliotecă care permite comunicarea în timp real, bidirecțională și bazată pe evenimente între browser și server.

În interfata web sunt afisate informații legate de numele librariilor software, locația unde sunt stocate, locația din memoria RAM.

Library dependency (NAMD) - dual-c

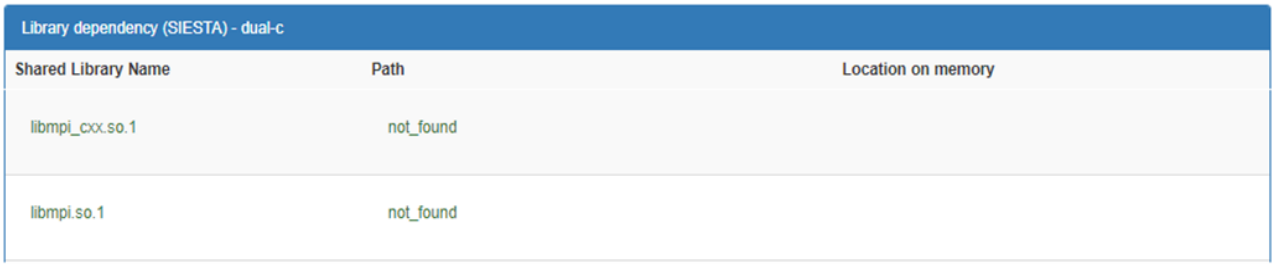
Shared Library Name	Path	Location on memory
linux-vdso.so.1		(0x00007ffcc1eda000)
libdl.so.2	/lib64/libdl.so.2	(0x00007f7f02793000)
libtcl8.5.so	/lib64/libtcl8.5.so	(0x00007f7f0246b000)
libpthread.so.0	/lib64/libpthread.so.0	(0x00007f7f0224f000)
libm.so.6	/lib64/libm.so.6	(0x00007f7f01f4d000)
libstdc++.so.6	/lib64/libstdc++.so.6	(0x00007f7f01c46000)
libgcc_s.so.1	/lib64/libgcc_s.so.1	(0x00007f7f01a30000)
libc.so.6	/lib64/libc.so.6	(0x00007f7f01662000)

FIG. 5.1: Pagina web privind bibliotecile software

Au fost create module de verificare a dependentelor software pentru aplicațiile/pachetele software științifice de modelare/simulare NAMD, EPOCH si SIESTA, care sunt utilizate in prezent în activitatea de cercetare din IFIN-HH.

FIG. 5.2: Paginile web de verificare a dependențelor software pentru NAMD, EPOCH si SIESTA

Acest modul de descoperire a topologiilor aplicatiilor software de simulare (NAMD, EPOCH, SIESTA) este deosebit de util utilizatorului, deoarece ii pune la dispozitie lista cu librariile software necesare rularii software-ului de simulare. Daca o dependinta nu este rezolvata, utilizatorul va lua legatura cu administratorul pentru a corecta aceasta problema. Mai jos avem situatia in care doua librarii software lipsesc, iar aplicatia nu va putea rula.

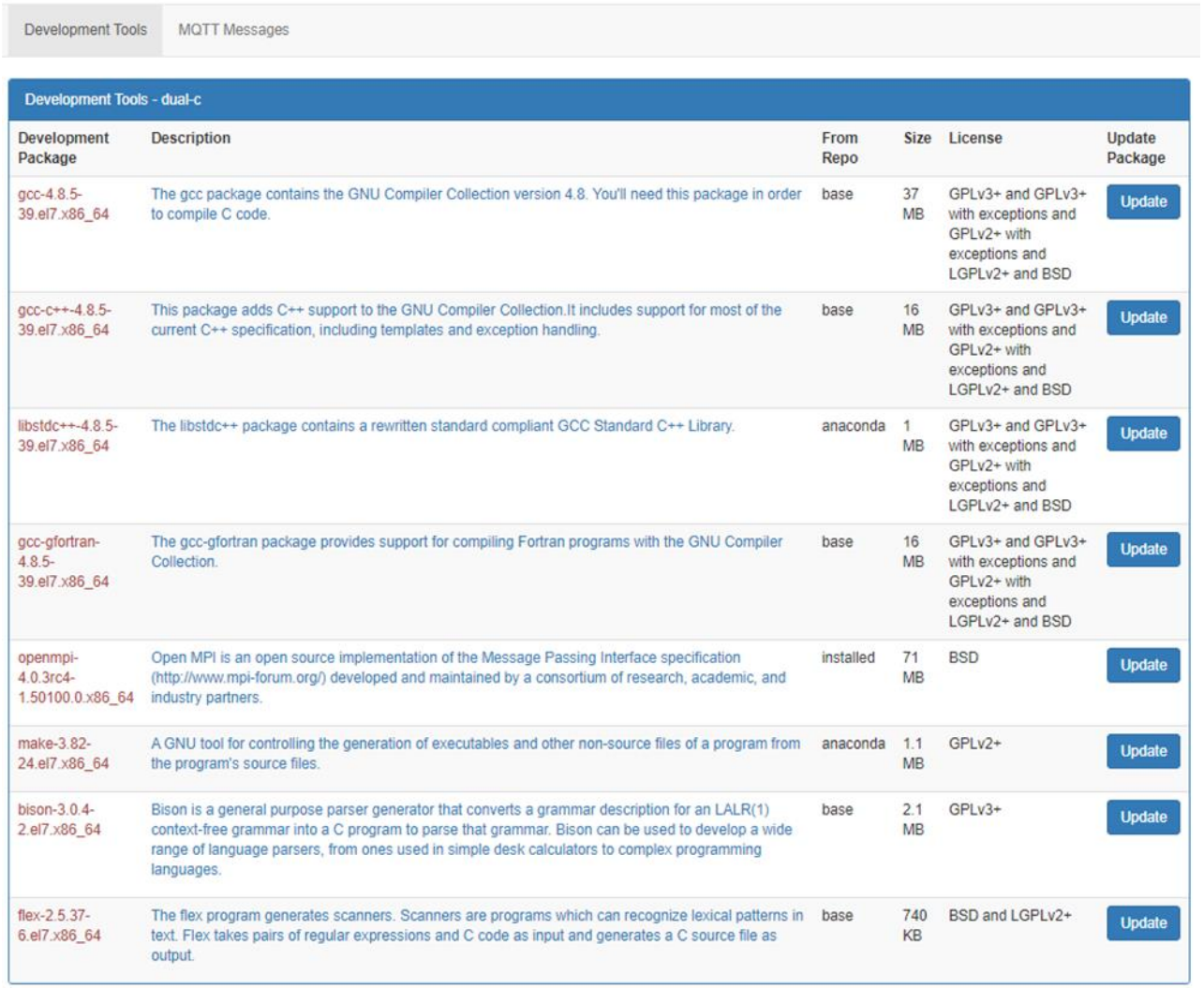


Library dependency (SIESTA) - dual-c

Shared Library Name	Path	Location on memory
libmpi_cxx.so.1	not_found	
libmpi.so.1	not_found	

FIG. 5.3: Exemplu de semnalare a dependențelor nerezolvate in cazul programului SIESTA

Pentru administrare s-a programat un modul care faciliteaza procesele de actualizare a pachetelor software de dezvoltare. Mecanismul de colectare/transmitere/afisare date este similar ca cel prezentat mai sus pentru descoperire topologie aplicatie software. In interfata web sunt afisate urmatoarele: denumirea pachetului software imrepuna cu versiunea acestuia, descrierea pachetului, din ce repository a fost instalat pachetul software, spatiul ocupat de pachetul software pe disc, date legate de licenta si posibilitatea de a se face update prin intermediul butonului de update.



Development Tools | MQTT Messages

Development Tools - dual-c

Development Package	Description	From Repo	Size	License	Update Package
gcc-4.8.5-39.el7.x86_64	The gcc package contains the GNU Compiler Collection version 4.8. You'll need this package in order to compile C code.	base	37 MB	GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and LGPLv2+ and BSD	Update
gcc-c++-4.8.5-39.el7.x86_64	This package adds C++ support to the GNU Compiler Collection.It includes support for most of the current C++ specification, including templates and exception handling.	base	16 MB	GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and LGPLv2+ and BSD	Update
libstdc++-4.8.5-39.el7.x86_64	The libstdc++ package contains a rewritten standard compliant GCC Standard C++ Library.	anaconda	1 MB	GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and LGPLv2+ and BSD	Update
gcc-gfortran-4.8.5-39.el7.x86_64	The gcc-gfortran package provides support for compiling Fortran programs with the GNU Compiler Collection.	base	16 MB	GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and LGPLv2+ and BSD	Update
openmpi-4.0.3rc4-1.50100.0.x86_64	Open MPI is an open source implementation of the Message Passing Interface specification (http://www.mpi-forum.org/) developed and maintained by a consortium of research, academic, and industry partners.	installed	71 MB	BSD	Update
make-3.82-24.el7.x86_64	A GNU tool for controlling the generation of executables and other non-source files of a program from the program's source files.	anaconda	1.1 MB	GPLv2+	Update
bison-3.0.4-2.el7.x86_64	Bison is a general purpose parser generator that converts a grammar description for an LALR(1) context-free grammar into a C program to parse that grammar. Bison can be used to develop a wide range of language parsers, from ones used in simple desk calculators to complex programming languages.	base	2.1 MB	GPLv3+	Update
flex-2.5.37-6.el7.x86_64	The flex program generates scanners. Scanners are programs which can recognize lexical patterns in text. Flex takes pairs of regular expressions and C code as input and generates a C source file as output.	base	740 KB	BSD and LGPLv2+	Update

FIG. 5.4: Iterfața web de actualizare a pachetelor software de dezvoltare

Informatiile afisate in interfata web la tab-ul "Development Tools", pot fi colectate atat de la masini fizice cat si de la masini virtuale. Avantajul modulului este ca informatiile sunt afisate intr-o interfata grafica iar administratorul nu mai este nevoit sa se logheze pe fiecare server in parte pentru a executa operatii de verificare/update pachete.

5.2.2 Utilizarea resurselor hardware

A fost dezvoltat un modul care afișează informații privind utilizarea resurselor hardware.

Pe serverul client in cadrul programului python "*mqtt_Listen_Data.py*" au fost apelate functii din cadrul modulului *psutil – python*. Acest modul este un utilitar de sistem programat în python, responsabil de preluarea informatiilor privind procesele care ruleaza, precum si despre utilizarea resurselor sistemului (CPU, memorie RAM, stocare pe discuri, retea). Modulul, care este util atât pentru monitorizarea sistemului, cât și pentru profilarea si gestionarea proceselor aflate in derulare, împrumută multe funcționalități oferite in linie de comanda de utilitarele Linux, cum sunt *ps*, *top*, *iotop*, *lsof*, *netstat*, *free*, *psutil*.

Pentru **nodul client** a fost dezvoltat programul python "*mqtt_Listen_Data.py*", al cărui cod este reprodus mai jos

```
import psutil
import paho.mqtt.client as mqtt
MQTT_Broker = "ctrl-os"
MQTT_Port = 1883
Keep_Alive_Interval = 45
MQTT_Topic = "apps/messages/dual-c/#"
def on_message(mosq, obj, msg):
        # This is the Master Call for saving MQTT Data into DB
        # For details of "sensor_Data_Handler" function please refer "sensor_data_to_db.py"
        print "MQTT Topic: " + msg.topic
        print "Data: " + msg.payload
        if msg.topic == "apps/messages/dual-c/list_cpu_times":
           if msg.payload == "list_cpu_times-c":
             out = ("{0}".format(psutil.cpu_times()))
             mqttc.publish("apps/messages/dual-c/list_cpu_times", out)

        if msg.topic == "apps/messages/dual-c/list_cpu_count":
           if msg.payload == "list_cpu_count-c":
             out = ("psutil.cpu_count() = {0}".format(psutil.cpu_count()))
             mqttc.publish("apps/messages/dual-c/list_cpu_count", out)

        if msg.topic == "apps/messages/dual-c/list_disk_partitions":
           if msg.payload == "list_disk_partitions-c":
             out = ("{0}".format(psutil.disk_partitions()))
             mqttc.publish("apps/messages/dual-c/list_disk_partitions", out)
```

```
        if msg.topic == "apps/messages/dual-c/disk_usage":
           if msg.payload == "disk_usage-c":
             out = ("{0}".format(psutil.disk_usage('/')))
             mqttc.publish("apps/messages/dual-c/disk_usage", out)
        if msg.topic == "apps/messages/dual-c/users_connected":
           if msg.payload == "users_connected-c":
             out = ("{0}".format(psutil.users()))
             mqttc.publish("apps/messages/dual-c/users_connected", out)

        if msg.topic == "apps/messages/dual-c/memory_usage":
           if msg.payload == "memory_usage-c":
             out = ("{0}".format(psutil.virtual_memory()))
             mqttc.publish("apps/messages/dual-c/memory_usage", out)
mqttc.loop_forever()
```

In cadrul programului sunt puse conditii care verifica daca mesajele primite sunt pe subiectul "`apps/messages/dual-c/list_cpu_times`" iar continutul mesajului este "`list_cpu_times-c`", atunci este executata functia "*psutil.cpu_times()*". Rezultatul executiei functiei este incapsulat intr-o variabila "`out`" si este exportata prin intermediul protocolului *mqtt* pe subiectul "*apps/messages/dual-c/list_cpu_times*" la serverul de mesagerie.

Pentru **serverul de mesagerie** a fost dezvoltat programul python "*app.py*", al cărui cod este reprodus in continuare

```
@mqtt.on_message()
def handle_mqtt_message(client, userdata, message):
 if message.topic == "apps/messages/dual-c/list_cpu_times":
    data = dict(
        topic=message.topic,
        payload=message.payload.decode(),
        qos=message.qos,
    )
    socketio.emit('list_cpu_times-c', data=data)
 if message.topic == "apps/messages/dual-c/list_cpu_count":
    data = dict(
        topic=message.topic,
        payload=message.payload.decode(),
        qos=message.qos,
    )
    socketio.emit('list_cpu_count-c', data=data)
 if message.topic == "apps/messages/dual-c/list_disk_partitions":
    data = dict(
        topic=message.topic,
        payload=message.payload.decode(),
```

```
        qos=message.qos,
    )
    socketio.emit('list_disk_partitions-c', data=data)
if message.topic == "apps/messages/dual-c/disk_usage":
    data = dict(
        topic=message.topic,
        payload=message.payload.decode(),
        qos=message.qos,
    )
    socketio.emit('disk_usage-c', data=data)
if message.topic == "apps/messages/dual-c/users_connected":
    data = dict(
        topic=message.topic,
        payload=message.payload.decode(),
        qos=message.qos,
    )
    socketio.emit('users_connected-c', data=data)
if message.topic == "apps/messages/dual-c/memory_usage":
    data = dict(
        topic=message.topic,
        payload=message.payload.decode(),
        qos=message.qos,
    )
    socketio.emit('memory_usage-c', data=data)
```

Datele sosite la serverul mqtt sunt asamblate si transmise catre motorul de reconstrucție (*rendering*) a șabloanelor – Jinja.

Fisierele șablon (template) sunt stocate in directorul "*template*" din cadrul serverului web. Fisierele template sunt fisiere ce contin date statice precum si parti pentru datele dinamice. Datele dinamice sunt cele primite de la aplicatia python *app.py*. Un template este reconstruit cu date specifice pentru a produce un document final. Pentru *rendering*-ul template-urilor Flask utilizează biblioteca de template Jinja.

In continuare este prezentat fisierul template ***messages.html*** care este reconstruit cu Jinja.

Pentru ca datele colectate de aplicatia app.py sa poata ajunge la client, aplicația client *messages.html* așteaptă evenimente de la aplicatia "*app.py*" cu `eventName` specificat, în cazul de față `list_cpu_times-c`, `list_cpu_count-c`, `list_disk_partitions-c`, `disk_usage-c`, `users_connected-c`, `memory_usage-c`.

Atunci cand un eveniment primit se potriveste cu eventName-ul specificat, este apelata functia *function(data)*, unde "*data*" contine datele dinamice care vor fi afisate in interfata web. `EventName` este de tipul string.

```
  socket.on('list_cpu_times-c', function(data) {
      console.log(data);
      var payload = data['payload'];
```

```
      var text = '(' + data['topic'] + ' qos: ' + data['qos'] + ') ' +
data['payload'];
      $('#list_cpu_times-c').text(payload).html();
    })

    socket.on('list_cpu_count-c', function(data) {
      console.log(data);
      var payload = data['payload'];
      var text = '(' + data['topic'] + ' qos: ' + data['qos'] + ') ' +
data['payload'];
      $('#list_cpu_count-c').text(payload).html();
    })

    socket.on('list_disk_partitions-c', function(data) {
      console.log(data);
      var payload = data['payload'];
      var text = '(' + data['topic'] + ' qos: ' + data['qos'] + ') ' +
data['payload'];
      $('#list_disk_partitions-c').text(payload).html();
    })

    socket.on('disk_usage-c', function(data) {
      console.log(data);
      var payload = data['payload'];
      var text = '(' + data['topic'] + ' qos: ' + data['qos'] + ') ' +
data['payload'];
      $('#disk_usage-c').text(payload).html();
    })

    socket.on('users_connected-c', function(data) {
      console.log(data);
      var payload = data['payload'];
      var text = '(' + data['topic'] + ' qos: ' + data['qos'] + ') ' +
data['payload'];
      $('#users_connected-c').text(payload).html();
    })

    socket.on('memory_usage-c', function(data) {
      console.log(data);
      var payload = data['payload'];
      var text = '(' + data['topic'] + ' qos: ' + data['qos'] + ') ' +
data['payload'];
```

```
        $('#memory_usage-c').text(payload).html();
    })
```

Pe partea de tip server a aplicației, in cadrul programului python *app.py* a fost definita functia "`handle_mqtt_message(client, userdata, message)`", care asteapta mesajele pe categorii de subiecte primite de la clienti. Fiecare client are propriul sau subiect (topic), astfel incat acesta sa poata fi mapat de server iar mesajele trimise de catre server sa fie interpretate doar de el.

Daca se doreste configurarea mai multor servere client in acelasi timp, acest lucru este posibil prin adaugarea caracterului de *wildcard* multi-level "`#`" la sfarsitul subiectului.

Daca se foloseste topicul sub forma: `apps/messages/#`, inseamna ca este inclus intregul grup de servere client. De exemplu:

`apps/messages/client_1`

`apps/messages/client_2`

`...`

Astfel, daca este primit un mesaj pe subiectul "`apps/messages/dual-c/list_cpu_times`", mesajul este asamblat sub un dictionar (colectie de date de tipul – key:value) ce contine subiectul, payload (mesajul), qos (quality of service).

Toate datele de mai sus sunt transmise catre browserul client si sunt afisate in interfata din Fig. 5.5.

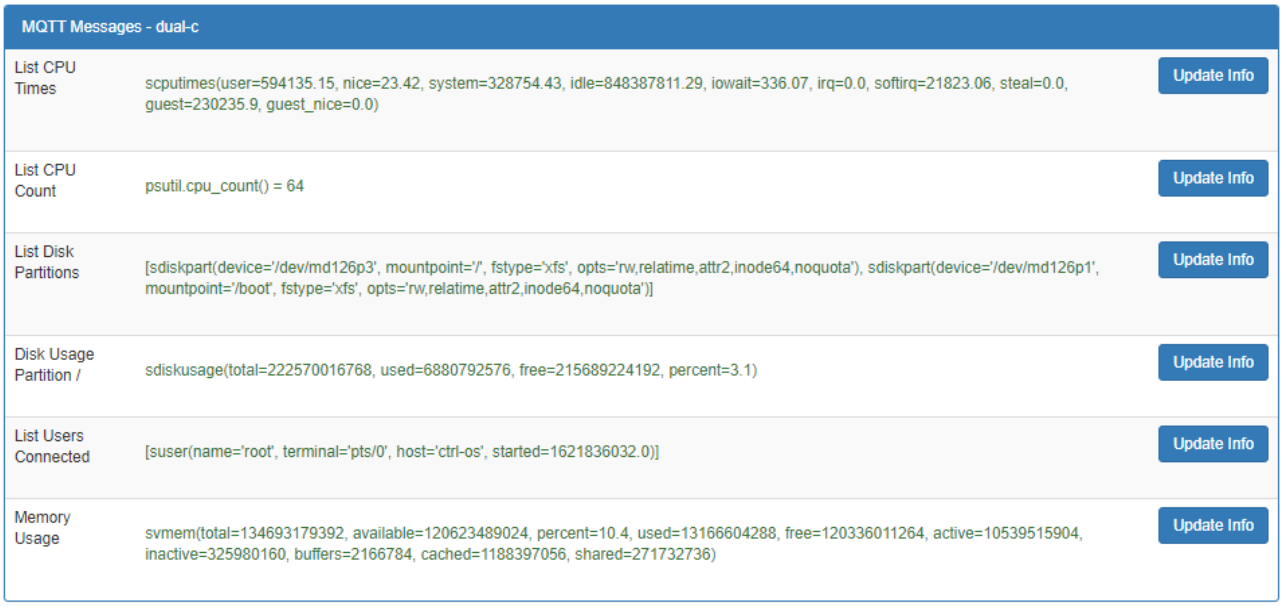


FIG. 5.5: Interfața web a mesajelor primite de la clienți

Informatiile din interfata web sunt afisate pentru toți utilizatorii.

Proba "*List CPU Times*" – returneaza o lista cu incarcare pe CPU "*scputimes(user=595861.14, nice=23.42, system=329549.01, idle=849447899.58, iowait=336.52, irq=0.0, softirq=21853.55, steal=0.0, guest=231538.37, guest_nice=0.0) "*;

Proba "List CPU Count" – afiseaza numarul de nuclee logice din sistem daca argumentul logical din cadrul functiei este False (psutil.cpu_count(logical=False)), intoarce doar numarul de nuclee fizice (sunt excluse procesoarele cu hyperthreading) " 64 ";

Proba "List Disk Partitions" – intoarce o lista cu toate partitiile de disc montate, inclusiv numele dispozitivului, calea unde este mapat, tipul sistemului de fisiere, aceasta functie este similara cu comanda "df" "[sdiskpart(device='/dev/md126p3', mountpoint='/', fstype='xfs',

opts='rw,relatime,attr2,inode64,noquota'), sdiskpart(device='/dev/md126p1', mountpoint='/boot', fstype='xfs', opts='rw,relatime,attr2,inode64,noquota')] ";

Proba "Disk Usage Partition /" - intoarce o lista cu statisticile de utilizare a discului despre calea data ca nume definit, in cazul nostru "/", inclusiv spatiul total, utilizat si liber, exprimat in octeti, plus procentul de utilizare "sdiskusage(total=222570016768, used=6884974592, free=215685042176, percent=3.1) ";

Proba "List Users Connected" – furnizează o lista cu utilizatorii conectati in prezent la sistemul client, cuprinzand urmatoarele campuri:

user: numele utilizatorului,

terminal: tty sau pseudo-tty asociat cu utilizatorul,

host: numele gazdei,

started: timpul de creare ca numar in virgula mobila exprimat in epoch time "suser(name='root', terminal='pts/0', host='ctrl-os', started=1621836032.0)]"

Aceasta pagina nu necesita actiuni de actualizare din partea utilizatorului, actualizarea informatiei afisate se face automat pentru ca este folosita tehnologia *SocketIO*.

5.2.3 Administrarea serviciilor cloud

Pentru administrarea și vizualizarea stării serviciilor cloud a fost programat un modul care colectează informații de la clienții cloud (noduri de calcul din clusterul cloud). Informatia primita de la clienti este actualizata automat, fara interventie din partea administratorului.

Informațiile respective sunt afișate într-un mod ce permite pornirea/oprirea și afisarea stării serviciilor cloud (Fig. 5.6).

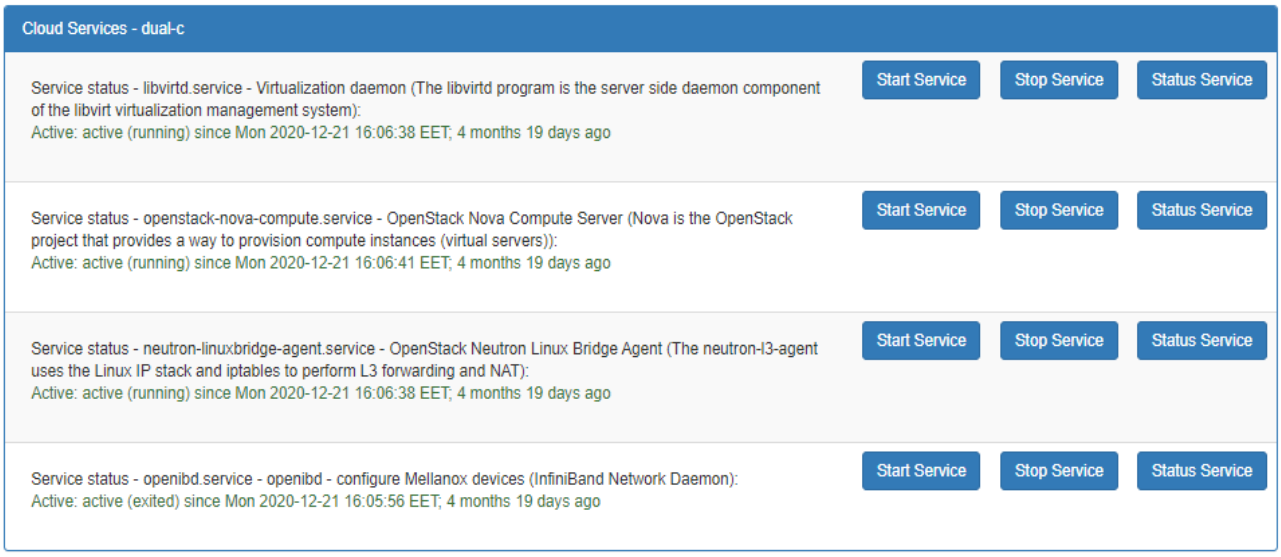


FIG. 5.6: Interfața de administrare a serviciilor Cloud

In interfata de administrare a serviciilor cloud, au fost incluse urmatoare servicii:

- ❑ libvirtd.service - Service status - libvirtd.service - Virtualization daemon (The libvirtd program is the server side daemon component of the libvirt virtualization management system):
- ❑ openstack-nova-compute.service - OpenStack Nova Compute Server (Nova is the OpenStack project that provides a way to provision compute instances (virtual servers)):
- ❑ neutron-linuxbridge-agent.service - OpenStack Neutron Linux Bridge Agent (The neutron-l3-agent uses the Linux IP stack and iptables to perform L3 forwarding and NAT):

- ❑ openibd.service - openibd - configure Mellanox devices (InfiniBand Network Daemon):

Operatiile de pornire/oprire ale serviciului sunt valabile doar pentru administrator. Pentru utilizator sunt afisate doar informatiile legate de starea serviciului dar fara posibilitatea controlarii starii serviciului de catre acesta.

Daca serviciul monitorizat este pornit, in interfata web va fi afisat mesajul "active (running)", data de cand serviciul a fost pornit, dar si un rezultat pe numar de luni/zile de serviciu activ.

5.2.4 Administrarea serviciilor HPC

Pentru administrarea și vizualizarea stării serviciilor HPC a fost dezvoltat un modul care colectează informații de la nodurile de calcul din clusterul HPC. Aceste informații sunt afișate într-o interfață grafică ce permite pornirea, oprirea, precum și afișarea stării serviciilor HPC (Fig. 5.7).

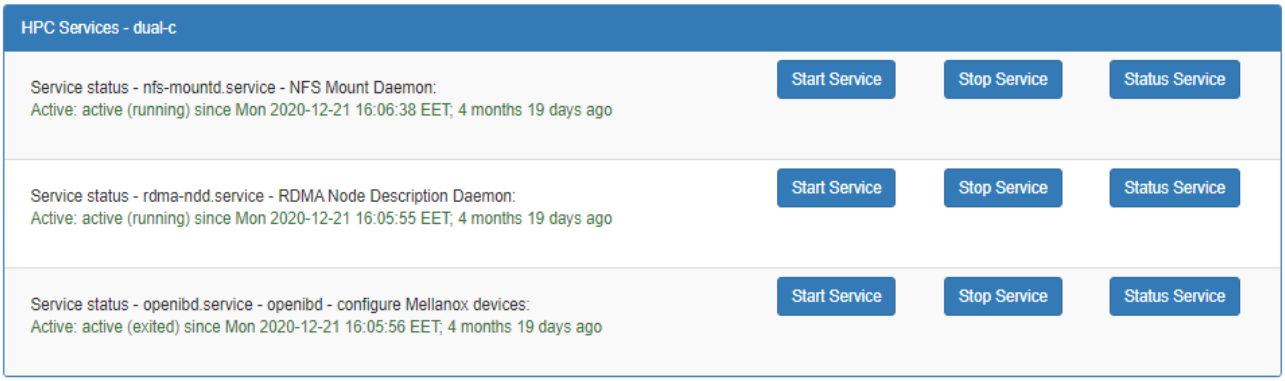


FIG. 5.7: Interfața de administrare a serviciilor HPC

In interfata de administrare a serviciilor HPC, au fost incluse urmatoare servicii:

- ❑ nfs-mountd.service - NFS Mount Daemon:

In arhitectura de calcul HPC, procesul care se ocupa de partajarea datelor pe nodurile de calcul este foarte important. Aceste date partajate contin: fisierele de intrare necesare programelor software de simulare, programele software de simulare, librarii necesare programelor software.

- ❑ rdma-ndd.service - RDMA Node Description Daemon:

RDMA este tehnologia care se permite nodurilor de calcul din clusterul HPC sa schimbe date intre ele direct prin memoria RAM fara implicarea procesorului, cache sau sistemul de operare.

- ❑ openibd.service - openibd - configure Mellanox devices:

Ca si in cazul serviciilor Cloud, modificarea starii serviciilor este permisa doar administratorului. Operatiile permise adminstratorului sunt de pornire/oprire/vizualizare serviciu. Pentru utilizator sunt afisate informatii legate de starea serviciului dar fara posibilitatea controlarii starii serviciului de catre acesta.

5.3 Concluzii

În continuarea activității de automatizare a procedurilor de informare și management privind resursele și serviciile de calcul începută în capitolul precedent, s-a dezvoltat un program software modular pentru autodescoperirea topologiei aplicatiilor, serviciilor și a proceselor, atât de pe mașini virtuale cât și fizice. Astfel, programul are avantajul că modulele sale pot fi utilizate nu numai pe nodurile cloud ci și pe nodurile unui cluster HPC – care poate fi fizic sau virtual (construit în cadrul infrastructurii CLOUDIFIN).

Modulele programate și prezentate în detaliu în raport realizează următoarele operațiuni:

a) colectarea de pe noduri de calcul multiple a informațiilor privind existența și disponibilitatea bibliotecilor software, precum și a dependențelor necesare pentru rularea diferitelor aplicații științifice;

b) informarea, prin intermediul unor mesaje cu format prestabilit, privind procesele care ruleaza si despre gradul de utilizare a resurselor nodurilor de calcul;

c) informarea (cu actualizare automată) privind serviciile cloud de pe nodurile de calcul și administrarea acestor servicii;

d) informarea și administrarea serviciilor HPC.

Interfața web a modulelor permite vizualizarea comodă a starii serviciilor si profilarea acestora.

Prin aceste funcționalități, prezentate explicit și exemplificate cu cazuri comune, aplicația programată este deosebit de utilă pentru managementul resurselor și serviciilor de calcul furnizate de clustere de mari dimensiuni.

6. Interfața web de acces a utilizatorilor la resursele CLOUDIFIN

6.1 Obiectiv

Obiectivul acestei subactivități este realizarea infrastructurii de înregistrare și autorizare a utilizatorilor și a proiectelor de calcul pentru care se solicită accesul la resursele CLOUDIFIN. În acest scop au fost dezvoltate interfața web de acces, aplicațiile back-end și baza de date aferentă.

6.2 Procedura de înregistrare și de solicitare a resurselor

Conform regulamentului de utilizare a infrastructurii de calcul avansat din IFIN-HH (*CC/IFIN-HH Resource Usage Policy*), în general solicitarea de acces la resursele și serviciile oferite de către DFCTI se face completând un formular electronic (AAF - *Access Application Form*) cu informații privind cerințele tehnice și aspectele științifice relevante ale proiectului de calcul propus. Daca solicitantul nu este deja înregistrat ca utilizator al sistemelor de calcul avansat din DFCTI, acesta completează, de asemenea, în AAF, datele de identificare personală necesare pentru crearea unui cont de autorizare a accesului.

Informațiile introduse în AAF sunt analizate din punct de vedere stiintific și tehnic de către un Comitet de evaluare care, în cazul acceptarii proiectului de calcul propus, stabilește condițiile și limitele de utilizare a resurselor, pe care le transmite solicitantului.

Dacă solicitantul accepta aceste condiții și nu este deja înregistrat ca utilizator, administratorul resurselor îl inregistrează ca utilizator și îi comunică acestuia parola inițială de acces la cont, pe care solicitantul o va folosi pentru accesul la resurse printr-o interfață web.

Dupa acces, utilizatorul are posibilitatea să specifice, în limitele impuse de către Comitetul de evaluare, resursele necesare pentru diferite etape ale proiectului de calcul propriu.

6.3 Software de programare utilizat

Pentru dezvoltarea infrastructurii de înregistrare și autorizare a utilizatorilor și a proiectelor de calcul s-au folosit exclusiv resurse *Open Source*.

Programarea web s-a realizat utilizând HTML, CSS si Bootstrap. Funcționalitatea paginilor este dată de codul dezvoltat în limbajul de programare Python.

Conform design-ului, informația cuprinsă în formularele completate de utilizatori în interfața web sunt stocate într-o bază de date gestionata de RDBMS sqlite3.

Conexiunea dintre interfata grafica si partea de back-end a aplicatiei s-a realizat utilizând Jinja2. Prin intermediul acestuia sistemul preia solicitările utilizatorilor și alocă automat resursele necesare.

6.4 Implementare

În cazul particular al centrului CLOUDIFIN, s-a proiectat o interfață web de acces multifuncțională, astfel încât să fie utilă în oricare dintre următoarele cazuri posibile:

1) accesarea resurselor prin nume de user/parolă de catre un utilizator deja înregistrat;
2) solicitarea de înregistrare ca utilizator împreună cu o propunere de proiect de calcul;

3) accesarea cu user/parolă de către un utilizator înregistrat pentru a propune un nou proiect de calcul;
4) solicitarea de înregistrare a unui nou utilizator pe un proiect de calcul existent;
5) solicitarea de acces în cazul pierderii parolei (*password recovery*).

A fost exclusă posibilitatea înregistrării unui utilizator fără să existe cel puțin un proiect de calcul asociat acestuia.

6.4.1 Pagina de acces

Pentru a beneficia de acces la resursele Cloud, solicitantul / utilizatorul va folosi link-ul paginii web de acces, http://cloudifin.ifin.ro:8080/ sau https://ccbd.ifin.ro/.

Pagina web de acces, reprezentată în Fig. 6.1, permite autentificarea utilizatorilor inregistrati pe baza de nume de cont si parola.

FIG. 6.1: Pagina de acces a utilizatorilor

De asemenea, pagina oferă vizitatorilor posibilitatea de a solicita:

1) *accesarea resurselor unui proiect de calcul in curs*, prin completarea campurilor "*Utilizator*" (de regula adresa e-mail de contact a acestuia) si "*Parola*", **dacă** este utilizator inregistrat;
2) *înregistrarea ca utilizator si propunerea unui nou proiect de calcul*, prin selectarea butonului "*Proiect și utilizator nou*", **dacă** nu a fost inregistrat;
3) *Înregistrarea unui nou proiect de calcul*, prin completarea campurilor "*Utilizator*", "*Parola*" si selectarea casetei "*Proiect nou*", **dacă** solicitantul este user inregistrat;
4) *înregistrarea ca utilizator pe un proiect de calcul existent* (prin selectarea link-ului "*Utilizator nou în proiect"*);
5) *atribuirea unei noi parole* in cazul pierderii parolei, **dacă** solicitantul este utilizator inregistrat, prin selectarea link-ului "*Recupereaza parola".*

6.4.2 Înregistrarea noilor proiecte de calcul și a utilizatorilor

Un utilizator nou poate fi inregistrat doar daca este asociat unui proiect de calcul. Acesta din urma poate fi:

a) proiect existent (care a fost aprobat anterior si este in curs de derulare);
b) proiect propus cu ocazia solicitarii de inregistrare.

In cazul b) solicitantul foloseste link-ul "*Proiect si utilizator nou*" din Fig. 6.1, care il directioneaza catre pagina reprezentata in Fig. 6.2.

Campurile formularului "*Inregistreaza utilizator si proiect nou*" sunt grupate dupa patru categorii de informatii:

1. Informatii Personale

   Completand aceste campuri solicitantul se identifica si furnizeaza datele de contact.

2. Informatii Supervizor

   Solicitantul introduce informatii despre coordonatorul proiectului de calcul asociat (care poate fi el insusi sau alta persoana).

3. Detalii proiect

   Se specifica titlul proiectului de calcul propus, aria de cercetare si perioada propusa pentru derularea acestuia. Documentul continand descrierea stiintifica si detaliile tehnice ale proiectului, in format PDF, se incarca in baza de date folosind butonul "*Browse*".

4. Resursele solicitate

   In aceasta sectiune a formularului solicitantul specifica natura si capacitatea resurselor de calcul necesare pentru derularea proiectului propus: numarul total de core-uri CPU, timpul estimat de utilizare al acestora si timpul total de utilizare a infrastructurii; memoria RAM necesara; capacitatea totala de stocare pe termen lung; biblioteci si aplicatii software necesare; limbajele de programare care vor fi utilizate.

   Solicitarile suplimentare, de exemplu privind accesul la resurse GPGPU, por fi specificate intr-un camp text la rubrica "Alte cerinte specifice".

La apasarea butonului "*Trimite*" programul apeleaza o functie python prin intermediul careia informatiile inserate de utilizator in formular sunt salvate in baza de date sqlite.

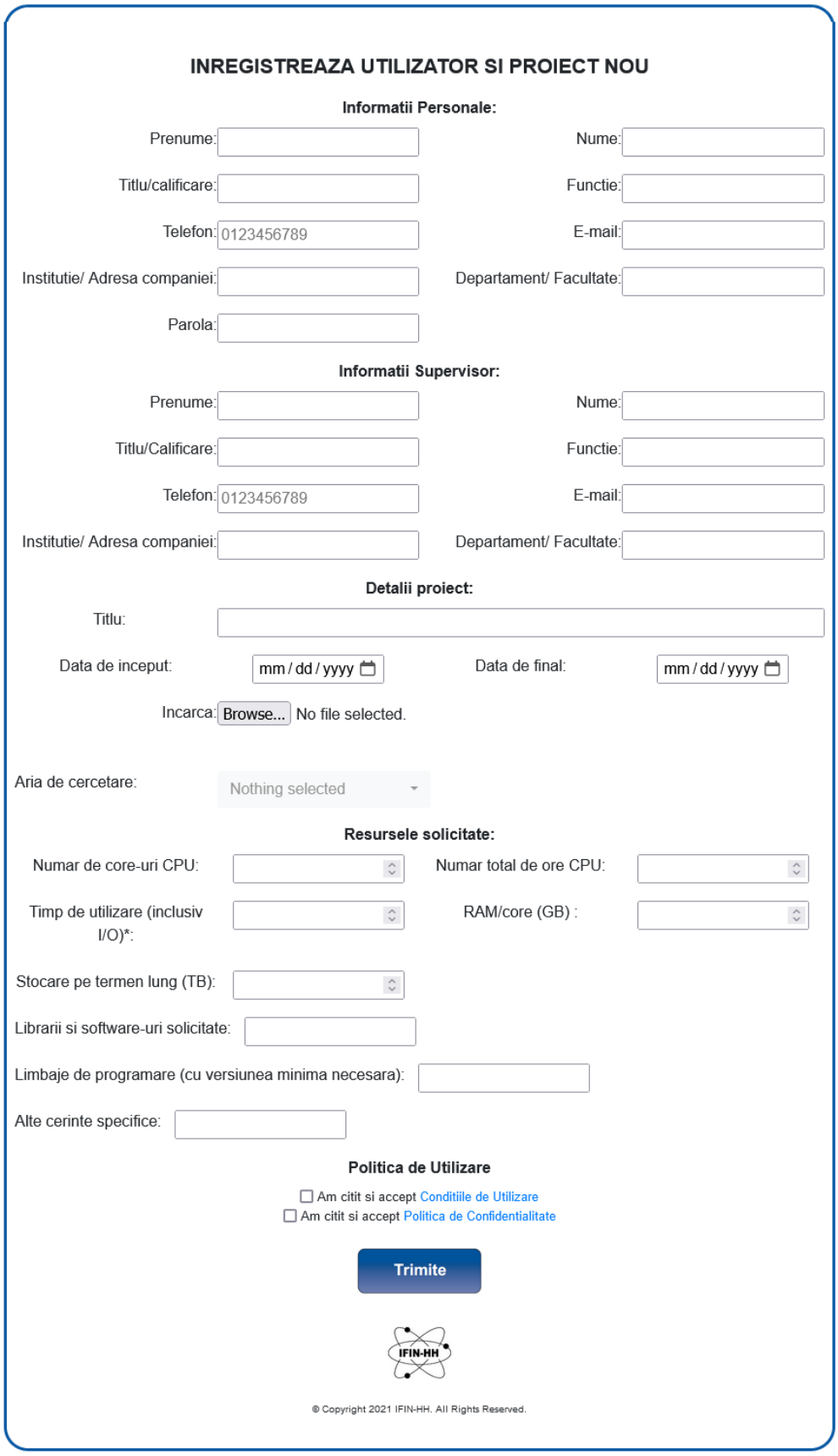


FIG. 6.2: Pagina/formularul de înregistrare a unui nou utilizator și a proiectului propus

6.4.3 Înregistrarea unui nou proiect de către un utilizator existent

In cazul in care utilizatorul are deja un cont creat anterior in cadrul unui proiect de calcul si doreste sa mai inregistreze un proiect, acesta completeaza datele de acces ("*Utilizator*" si "*Parola*"), si bifeaza caseta "*Proiect nou*" de pe pagina de acces (Fig. 6.1).

Dupa logarea avand caseta bifata, interfata afiseaza pagina web "*Inregistreaza proiect nou*", reprodusa in Fig. 6.3, in care campurile din rubrica "*Informatii personale*" ale formularului sunt precompletate cu datele personale ale utilizatorului autentificat, preluate din baza de date (prenume, nume, titlu/calificare, functie, telefon de contact, email de contact, adresa institutiei, departament/facultate):

```
@app.route('/formConfirmLogin', methods=["POST"])
def form():
      if ((user==1) and (request.form.get('bifa') == 'on')):
            return render_template("inregistrare_proiect_nou.html", utilizator=
utilizator)
```

La fel ca in cazul descris in sectiunea 6.4.2, dupa completarea formularului, la apasarea butonului "*Trimite*" apeleaza o functie python prin care informatiile inserate de user in formular sunt salvate in baza de date sqlite:

```
@app.route('/afisare_proiect_inregistrat', methods=["POST"])
def proiect_nou():
      sqlite_insert_query = """INSERT INTO proiect_nou (
            prenume, nume, institutie, functie, telefon, email, computing_tech,
            lider_grup, titlu, startPreferredDate, endPreferredDate,
            startUndesiredDate, endUndesiredDate, contactLocal, ariiCercetare,
            metode, nr_cpu, nr_cores, nr_ram, nr_tb, librarie, limbaje,
            cerinte,experienta, plan_proiect, nume_fisier_proiect)
      VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"""
return render_template("afisare_proiect.html"
```

Pentru toate formularele, datele completate sunt preluate de sistem doar daca utilizatorul bifeaza casetele prin care confirma ca a citit si accepta conditiile de utilizare a sistemului si, respectiv, politica de confidentialitate, care sunt publicate pe site-ul web.

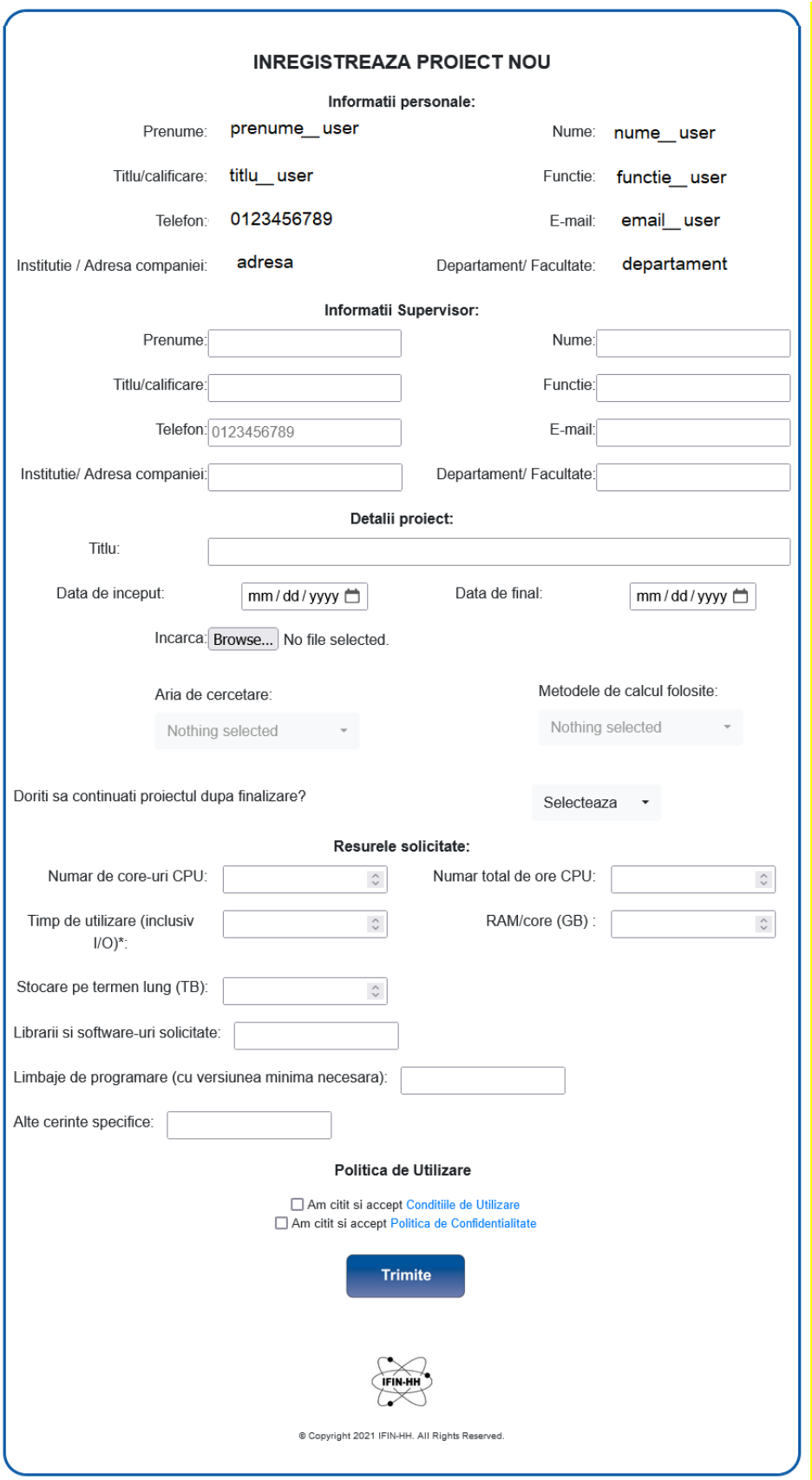


FIG. 6.3: Pagina de înregistrare a unui nou proiect de către un utilizator existent

6.4.4 Înregistrarea unui nou utilizator pe un proiect existent

Butonul "*Utilizator nou in proiect*" apeleaza o functie Python prin care este afișată pagina html cu formularul care trebuie completat pentru a crea un cont de utilizator și în care sunt listate pentru a putea fi selectate numele proiectelor deja înregistrate (acest lucru e necesar deoarece utilizatorul trebuie să selecteze proiectul pentru care isi creaza contul):

```
@app.route('/creare_cont_utilizator.html')
def inregistrare_utilizatori():
      c.execute('''SELECT DISTINCT proiect FROM table_proiecte;''')
      proiecte = c.fetchall()
      total_proiecte = [list(i) for i in rows]
      for proiect in proiecte:
                  print(proiect)
      print (total_proiecte)
      return render_template("creare_cont_utilizator.html")
```

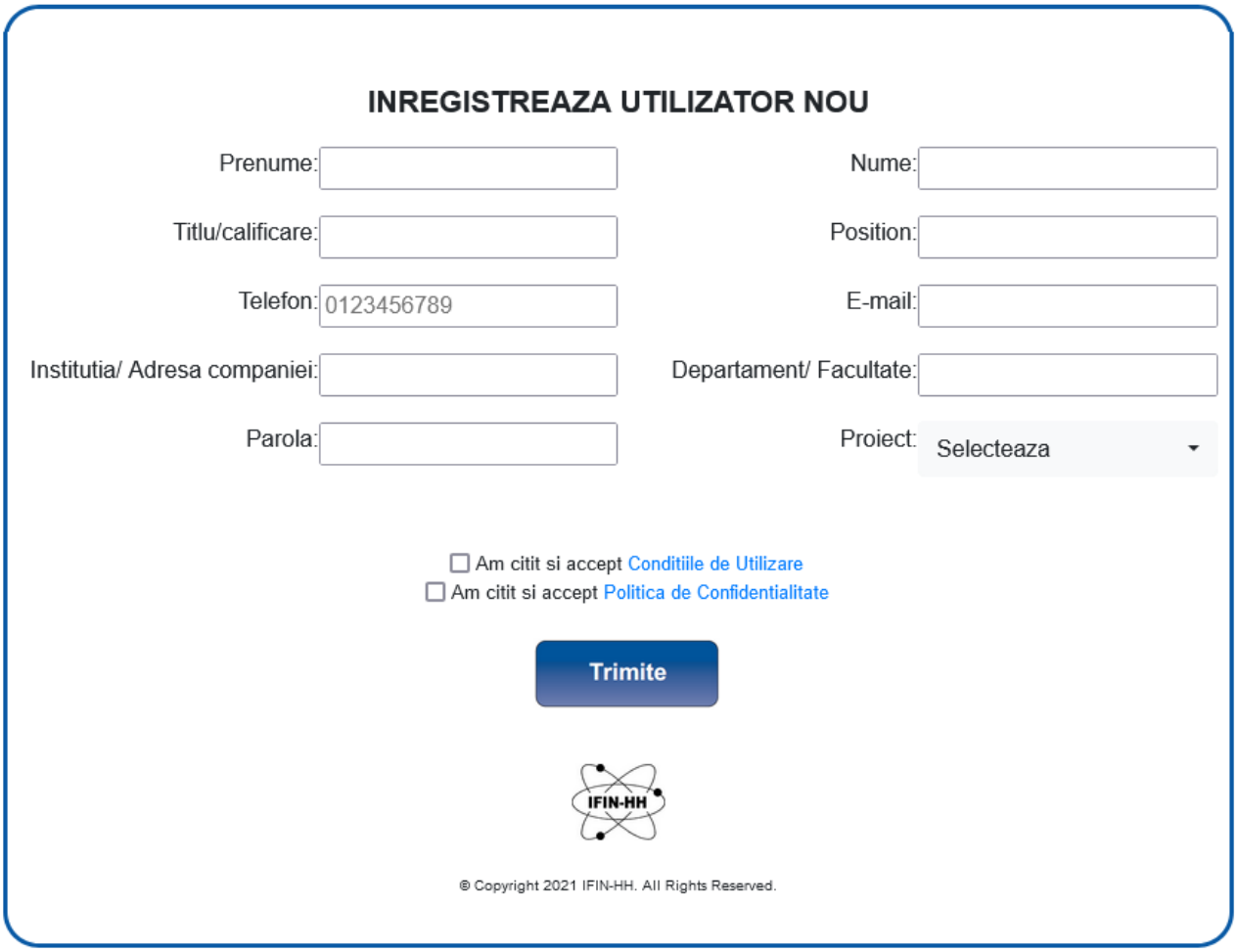


FIG. 6.4: Pagina de înregistrare a unui nou utilizator pe un proiect existent

La apăsarea butonului "*Trimite*" se apeleaza o functie prin care informatiile utilizatorului sunt inserate in baza de date:

```
@app.route('/inserare_utilizator', methods=["POST"])
def inserare _utilizator():
```

```
                sqlite_insert_query = """INSERT INTO tabel_utilizatori
(prenume, nume, institutie, functie, telelefon, email, pass, titlu_proiect)
                                    VALUES(?,?,?,?,?,?,?,?)"""

      cursor.execute(sqlite_insert_query,(prenume,nume,institutie,functie,
                                          telefon,email, pass, proiect))

      return render_template("inserare_utilizatori.html)
```

Solicitarea utilizatorului va fi transmisa automat prin e-mail coordonatorului proiectului de calcul respectiv, care poate să accepte sau nu cooptarea în proiect a utilizatorului respectiv.

6.4.5 Schimbarea parolei de acces

Dacă utilizatorul a uitat/pierdut parola de acces, acesta va acționa butonul *"Recupereaza parola"* din formularul din Fig. 6.1 pentru eliberarea unei noi parole.

În acest caz este apelată o funcție Python prin intermediul căreia se afișează pagina web cu formularul care trebuie completat pentru a obține noua parolă:

```
@app.route('/parola_recuperare.html')
def parola_ recuperare ():
      return render_template("parola_ recuperare.html")
```

FIG. 6.5: Pagina de solicitare a comunicării unei noi parole de acces

Utilizatorul va completa adresa proprie de e-mail și va acționa butonul *"Submit"* din formularul *"Recupereaza parola"*, demarând astfel procedura standard de eliberare a unei noi parole provizorii.

6.5 Concluziile capitolului

Ansamblul aplicațiilor prezentate în acest capitol, care realizează interacțiunea cu infrastructura CLOUDIFIN, sunt esențiale atât pentru înregistrarea și accesul utilizatorilor cât și pentru înregistrarea proiectelor de calcul propuse.

Design-ul paginii web de acces principale permite alegerea rapidă dintr-un meniu exhaustiv a acțiunilor pe care solicitanții dorec să le execute.

Pentru fiecare opțiune posibilă sunt prezentate explicit codurile back-end atașate butoanelor, formularele web și interacțiunile cu baza de date.

Criteriile principale urmărite în elaborarea design-ului web au fost simplitatea prezentarii grafice și satisfacerea promptă a cerințelor utilizatorilor.

7. Concluzii

Livrabilul RT-2.1 trece în revistă principalele rezultate obținute în primul an al proiectului în cadrul Subactivității 2.1 – "*Realizarea de servicii și aplicații informatice noi pentru administrarea și monitorizarea centrului CLOUDIFIN*".

În perioada de raportare s-au proiectat și dezvoltat, în conformitate cu prevederile planului de realizare din propunerea de proiect, instrumentele software și serviciile necesare atât pentru managementul eficient al infrastructurii CLOUDIFIN, cât și pentru satisfacerea cerințelor utilizatorilor privind accesul la resurse și servicii.

Dezvoltarea de noi instrumente de monitorizare și management a avut drept scop principal automatizarea unor proceduri repetitive utilizate de către administratorii centrului Cloud, pentru evitarea supraaglomerării acestora în perspectiva creșterii semnificative a resurselor și a numărului de utilizatori odată cu conectarea de noi comunități științifice interesate de beneficiile EOSC.

În această categorie se înscriu serviciul centralizat de monitorizare a resurselor Cloud din secțiunea 3.2 și aplicațiile de administrare prezentate in Cap. 4. De asemenea, serviciul de monitorizare și alertare pentru clustere de resurse de calcul, descris în detaliu în secțiunea 3.3, are o importanță deosebită pentru menținerea calității serviciilor oferite utilizatorilor.

Alte instrumente software, cum este programul dezvoltat pentru autodescoperirea topologiei aplicațiilor, serviciilor și a proceselor din Cap. 5, se adresează atât managerilor centrelor de resurse (prin modulele care permit administrarea acestora) cât și utilizatorilor (doar pentru informare).

Soluția dezvoltată pentru acces și înregistrarea utilizatorilor și a proiectelor de calcul, ale cărei pagini web și aplicații back-end sunt prezentate în Cap. 6, implementează și integrează, plecând de la o interfață unică, multiplele proceduri existente privind interacția cu (potențialii) beneficiari ai resurselor.

Rezultatele prezentate în raport sunt esențiale pentru demararea Subactivității 2.2 – "*Federalizarea resurselor oferite de catre CLOUDIFIN sialte centre Cloud si dezvoltarea unei interfete web pentru accesul utilizatorilor la aceste resurse*", ce se va derula în cel de-al doilea an al proiectului. Astfel, interfața web de acces la resursele federalizate va fi dezvoltată plecând de la soluția realizată pentru CLOUDIFIN, fiind o extensie a acesteia. De asemenea, aplicațiile software realizate în această etapă pentru monitorizarea și autodescoperirea resurselor vor constitui modelele unor programe cu funcționalități similare pentru federația centrelor Cloud.

În concluzie, sfârșitul primului an de derulare a proiectului CeCBiD-EOSC marchează realizarea în totalitate a obiectivului principal al acestei etape, rezultatul propus nr. 18 - *Servicii si aplicatii informatice pentru administrarea si monitorizarea centrului CLOUDIFIN, precum si pentru asigurarea accesului utilizatorilor*.

