Universitatea Națională de Știință și Tehnologie POLITEHNICA București

Facultatea de Automatică și Calculatoare,
Departamentul de Calculatoare

TEZĂ DE DOCTORAT

Rezumat

Middleware în Sisteme Cluster și Grid

Conducător Științific:
Prof. Dr. Ing. Nicolae Țăpuș

Autor:
drd. Sergiu Weisz

București, 2024

National University of Science and Technology POLITEHNICA Bucharest

Faculty of Automatic Control and Computers,
Computer Science and Engineering Department

PHD THESIS

Summary

Middleware in Cluster and Grid Computing

Scientific Adviser:
Prof. Dr. Ing. Nicolae Țăpuș

Author:
drd. Sergiu Weisz

Bucharest, 2024

Abstract

Lumea actuală a fost schimbată pentru totdeauna prin introducerea calculului distribuit ca paradigmă de dezvoltare. În loc să facă aplicațiile cu un singur nucleu mai rapide, un proces cu randamente în scădere, cercetarea s-a mutat la calcul cu mai multe procese, mai multe procesoare și mai multe noduri. Prin scalarea orizontală a procesării, putem crește capacitățile de procesare pentru sarcinile de lucru la un cost mai mic decât creșterea puterii de procesare per nucleu.

Cluster computing vine să ajute în acest sens, prin introducerea unui model care organizează aplicațiile pe baza cazului de utilizare, oferind instrumente de management pentru a ajuta utilizatorii să implementeze joburi, să personalizeze mediile de lucru și să adauge simplu monitorizare, reducând cheltuielile generale pentru fiecare utilizator pentru a-și gestiona procesele. Cluster computing oferă o soluție rentabilă pentru grupurile de utilizatori care lucrează în interiorul aceleiași instituții pentru a partaja resurse, permițând colaborări mai bune și o cantitate mai mare de resurse disponibile de care să profite.

Grid computing vine ca o evoluție a modelului de cluster computing. Un grid este alcătuită din mai multe clustere eterogene, care rulează același software de management și coordonare pe propriile noduri. Mediile grid reduc bariera de intrare pentru calculul de înaltă performanță, deoarece o instituție poate participa la un grid dedicându-și resursele gridului și obținând acces la mai multe resurse decât sunt disponibile altfel, în special în contextul Big Science care necesită zeci sau sute de mii de nuclee pentru a participa în domeniul cercetării. Mediile grid aduc un model colaborativ pentru cercetare, permițând mai multor instituții să stabilească un cadru comun de procesare pe care îl pot implementa și utiliza indiferent de configurația cluster-ului local.

Având în vedere creșterea cererii de putere de calcul atât în cluster, cât și în grid computing următoarea teză implementează optimizări pentru utilizarea resurselor în medii grid prin îmbunătățirea transferurilor de date prin reducerea overheadului de transfer. Managerii de operațiuni ale gridului trebuie să țină pasul cu cererea de resurse din partea utilizatorilor gridului și cu fluxurile lor de lucru. În această teză, explorăm posibilitatea creșterii de resurse disponibile în mediile grid prin integrarea joburilor multi-core în medii grid, scăzând utilizarea memoriei și permițând mai multă procesare în paralel.

Paradigma de calcul distribuit poate fi adaptată în multe domenii, nu doar pur cercetare sau comercial. Educația a primit un impuls prin trecerea la digitalizarea resurselor și prin adoptarea e-learning-ului. Aplicații precum Moodle, Draw.IO sau Microsoft Teams

au permis profesorilor și colaboratorilor să elimine o parte din povara organizării cursurilor de pe umerii educatorilor. O cale care nu a fost explorată la fel de mult este aceea de a automatiza dezvoltarea materialelor de clasă, implementarea și verificarea temelor, activități care sunt dificil de realizat și necesită multă muncă manuală din partea profesorilor. Această teză prezintă un framework construit folosind concepte de calcul distribuit, cum ar fi containerizarea, integrarea continuă și distribuția continuă, oferind educatorilor posibilitatea de a crea și implementa automat materiale de clasă. Pentru a reduce și mai mult volumul de muncă al profesorilor, am dezvoltat un middleware de verificare a temelor care se integrează cu platforma de e-learning Moodle, mutând verificarea temelor de la profesori la scripturi automatizate implementate în infrastructurile publice sau private. Noul verificator de teme a fost construit pe baza tehnicilor utilizate de software-ul de gestionare a gridului, putându-se integra cu mai multe infrastructuri instituționale bazate pe GitLab și permițând studenților și profesorilor să programeze procesul de verificare în cozile partajate între clase.

Odată cu evoluția cluster computing, a fost dezvoltată o gamă largă de instrumente de gestiune pentru implementarea, întreținerea și operarea clusterelor pentru diferite cazuri de utilizare bazate pe aplicații, containere sau mașini virtuale. Această teză dezvoltă un model pentru managementul clusterelor folosind mașini virtuale, implementând OpenStack împreună cu serviciile sale pentru a oferi utilizatorilor libertatea de a-și configura și gestiona aplicațiile într-un mediu securizat. Pentru a asigura timpi de nefuncționare reduși, au fost luate măsuri pentru a implementa mașini virtuale pentru utilizatori cu o disponibilitate ridicată, profitând de funcționalitățile actuale de cloud computing, cum ar fi verificări continue de sănătate și dispozitive de stocare montate în rețea, pentru a realiza o migrare rapidă a mașinilor virtuale. Monitorizarea mediilor complexe este o sarcină care necesită instrumente avansate care se integrează flexibil cu aplicațiile pentru a le verifica starea și a alerta administratorii înainte de apariția problemelor, intenționând să folosească cât mai mult posibil din resursele disponibile. Am dezvoltat un model de monitorizare a infrastructurilor orientate spre virtualizare pe baza instrumentelor open-source disponibile în prezent, care ar permite administratorilor să fie alertați în cazul unor evenimente, oferindu-le informații adecvate pentru a preveni problemele sau a le remedia.

Contents

Chapter 1

Introducere

Pe măsură ce digitalizarea a fost integrată în mai multe fațete ale vieții noastre, cerințele de resurse au crescut. Într-un mediu digitalizat resursele de calcul trebuie utilizate eficient și trebuie găsite noi tipuri de resurse de calcul pentru a ține pasul. Este sarcina noastră să creștem utilizarea resurselor prin găsirea de noi mijloace de integrare și utilizare a resurselor în moduri inovatoare. Odată cu creșterea utilizării resurselor și cerințele de calcul la scară largă, noi metode trebuie investigate pentru gestionarea resurselor.

Calculul s-a mutat din ce în ce mai mult de la procesarea pe dispozitiv, necesitând procesoare puternice care pot fi utilizate numai împreună cu hardware greoi, generând cantități mari de căldură și trebuind să fie conectate la circuite puternice la un model de calcul descentralizat, cu cererile utilizatorilor fiind trimise către mașini la distanță care pot fi gestionate mai eficient. Mutarea sarcinilor de lucru în medii la distanță permite rularea acestora la o scară mai mare, combinând mai multe dispozitive pentru a ajuta la îndeplinirea sarcinii. Mediul de calcul distribuit a permis utilizarea pe scară largă a computerelor în scopuri științifice, educaționale și de afaceri, facilitând utilizarea unui mediu digital de către instituții și scăzând bariera de intrare în știință, educație și probleme grele de calcul.

Un cluster este un mediu distribuit bazat pe o combinație de hardware și software care rulează servicii specializate care răspund la solicitările utilizatorilor. Clusterele folosesc hardware special conceput, care garantează longevitatea și funcționalitatea, construite pentru a permite o răcire mai bună și un consum de energie mai mare decât un computer obișnuit de acasă. Software-ul de gestionare a clusterelor poate fi implementat în multe moduri diferite, în funcție de cazurile de utilizare a clusterului, unele punând accent pe securitate și disponibilitate, în timp ce altele încearcă să obțină o suprasolicitare cât mai mică posibil, astfel încât serviciile cluster să poată profita de resursele disponibile.

Infrastructurile de rețea au apărut ca o consecință a nevoilor mai mari de putere de procesare. Un grid de calcul este o asociere de clustere care comunică între ele prin rețea și au medii software similare. Acest lucru permite ca o aplicație să fie distribuită nu numai în interiorul aceluiași cluster, ci în mai multe clustere, distribuite în locații, chiar și în țări diferite, profitând de resursele disponibile pentru a calcula un singur rezultat în paralel pe mai multe mașini. Infrastructurile grid rulează software specializat pentru a

raporta disponibilitatea resurselor, pentru a programa transferuri de fișiere distribuite și pentru a începe sarcinile de lucru pe grid. Prin intrarea în medii grid, instituțiile pot avea mai multă putere de procesare disponibilă la un moment dat, garantând că o anumită cantitate din propriile resurse va fi utilizată de alte instituții.

Un middleware este un software specializat care gestionează software-ul în medii cluster și grid. Un middleware poate fi o aplicație care pornește un script pe o anumită mașină sau se asigură că anumite fișiere sunt prezente pe o soluție de stocare în interiorul clusterului. Acestea sunt o componentă critică a procesării distribuite, deoarece sunt folosite ca coloană vertebrală pentru software-ul distribuit pe care îl rulează.

1.1 Obiectivele Tezei

Obiectivul principal al tezei este de a crește eficiența utilizării resurselor în cloud, cluster și grid computing, îmbunătățirea proceselor, scăderea costului de gestiune și adăugarea de noi tipuri de resurse pentru a fi utilizate. Având nevoie de mai multă putere de procesare pentru sarcinile complexe de lucru din rețea, trebuie să ne asigurăm că resursele disponibile astăzi sunt utilizate la cea mai mare capacitate. În acest scop, teza urmărește să implementeze îmbunătățiri ale managementului datelor, transferurilor de date și locației datelor pentru a reduce timpul petrecut în așteptarea transferurilor și ștergerii datelor.

Mediile grid moderne au introdus nevoia de joburi multi-core care să poată partaja memoria între procese. S-a dovedit că joburile cu mai multe nuclee reduc utilizarea memoriei în mediile grid [22]. Teza intenționează să implementeze un algoritm de programare a joburilor multi-core într-un mediu grid, specializat în mediul grid ALICE. Folosind suportul pentru joburi multi-core implementat, cozile de scavenging pot fi explorate și exploatate ca o nouă cale pentru rularea sarcinilor de lucru din rețea fără costuri suplimentare. Creșterea resurselor disponibile necesită, de asemenea, integrarea de noi tipuri de facilități de calcul care pot fi utilizate pentru a adăuga o nouă putere de procesare.

Această teză explorează configurațiile disponibile pentru gestionarea clusterelor în mediile de cercetare și educaționale, care necesită timp de funcționare ridicat și reziliență pentru a servi utilizatorii. Obiectivul este de a implementa o infrastructură de cloud privat cu înaltă disponibilitate pentru a găzdui resurse educaționale și instituționale, cum ar fi servicii orientate spre utilizator, sarcini de lucru de cercetare și mașini virtuale sandbox pentru studenți.

Durata de funcționare a serviciilor nu este doar o măsură a implementării serviciilor care sunt auto-suficiente, deoarece chiar și acestea pot eșua. O infrastructură robustă trebuie să implementeze servicii de monitorizare și alertă care pot fi utilizate pentru a evalua starea clusterelor și a serviciilor și pot alerta părțile responsabile cu privire la problemele care ar putea apărea. Gestionând eficient monitorizarea, programarea sarcinilor de lucru și gestionarea stocării, un cluster poate oferi utilizatorilor un mediu de lucru care permite proiectelor lor să se dezvolte, să înflorească și să reușească. Pentru a asigura disponibilitatea serviciului, un sistem de monitorizare trebuie implementat și în interiorul unui cluster.

Dezvoltarea middleware poate fi valorificată pentru a permite nu numai calcularea eficientă pentru cercetare și afaceri, ci și pentru educație. Teza abordează dezvoltarea middleware din perspectiva automatizării educaționale, folosind software middleware pentru a îmbunătăți rezultatele învățării pentru studenți prin automatizarea sarcinilor efectuate în mod regulat manual de profesori, permițând educatorilor să petreacă mai mult timp predând și mai puțin timp efectuând acțiuni repetitive. Teza își propune să dezvolte servicii educaționale care ajută la automatizarea proceselor educaționale, cum ar fi construirea documentației și verificarea temelor, utilizând paradigmele de calcul distribuite prin acțiuni de pipelining și paralelizarea procesării pe mai multe mașini.

1.2 Contribuțiile Tezei

O problemă de impact în grid computing este asigurarea că datele sunt distribuite corect în mediul grid, pentru a crește paralelizarea sarcinilor prin eliminarea blocajului componentei de stocare. Datele trebuie transferate dintr-un depozit central sau dintr-o sursă de origine către diferite soluții de stocare, cum ar fi arhive și zone de depozitare. **Am optimizat transferurile de date pentru grila ALICE**, în timpul unei campanii critice de transfer de date, prin optimizarea performanței transferului de date în legăturile de rețea cu lățime de bandă foarte variabilă și soluții hardware și software eterogene de stocare a datelor. Am reușit să petrecem mai puțin timp așteptând ca fișierele să fie transferate la punctul de procesare prin reducerea timpului irosit în gestiunea fișierelor, timp care a fost petrecut procesând datele pentru experimentul ALICE.

Noul framework de procesare a datelor a fost rescris de echipa ALICE cu scopul de a rula procesarea datelor pe mai multe nuclee. **În această teză arătăm îmbunătățirile pe care le-am făcut în frameworkul JAliEn pentru a lua în considerare joburile multi-core, permițând rularea mai multor joburi multi-core în paralel**. Creșterea numărului de nuclee utilizate de un workload a fost făcută pentru a **scăderea utilizării memoriei per nucleu**, ceea ce duce la posibilitatea de a programa mai multe joburi în paralel, deoarece centrele de calcul sunt mai limitate de cantitatea de RAM pe care o pot furniza pe procesor [39], decât numărul de nuclee CPU pe care le pot achiziționa. Noua arhitectură a necesitat reproiectarea middlewareului, monitorizarea mai multor joburi și adăugarea de cod pentru a limita resursele alocate pentru joburi.

În timp ce integram suportul pentru joburi cu mai multe nuclee, **am implementat un mecanism pentru alocarea dinamică a joburilor pe baza resurselor disponibile** ca atare, suntem capabili să preluăm sarcina de programare de la Sistemul Local de Management al Resurselor (LRMS) și să o luăm asupra middleware pentru a programa procesarea. Integrarea suportului pentru cozile în care programarea se face numai pe nod, nu pe proces, a permis experimentului ALICE să profite de clustere care altfel nu ar fi fost disponibile. Întrucât middleware-ul grid este mai în ton cu parametrii de programare a lucrărilor decât LRMS, putem face alegeri de programare mai bune prin gestionarea programării întregului nod.

În procesul de creștere a capacității de resurse a unui mediu de rețea, trebuie să luăm în considerare resursele neexploatate sau tipurile de resurse pe care le putem integra fără costuri suplimentare. În procesarea jobului pe întregul nod, există un mod de

planificare a lucrărilor în care procesele pot fi programate, iar dacă este programat un job cu prioritate mai mare, jobul curent care rulează este oprit, numit scavenging. Cozile scavenging au fost identificate în această teză ca o posibilă cale de creștere a resurselor disponibile, prin programarea joburilor pe acestea cu riscul ca procesarea să fie întreruptă de alte tipuri de joburi. **Teza demonstrează viabilitatea procesului de alocare a timpului în clustere care permit o programare flexibilă, în ciuda riscului ca timpul de procesare să fie irosit atunci când procesarea este oprită anticipat.**

În contextul digitalizării educaționale, software-ul middleware joacă un rol în gestionarea serviciilor și funcționalităților oferite studenților și profesorilor. Profesorii și studenții trec prin sarcini repetitive, rezolvate manual, care pot fi automatizate folosind software middleware construit pe baza lecțiilor învățate în dezvoltarea de software pentru medii cluster și grid. **Teza adaptează software-ul middleware și soluțiile de dezvoltare software, cum ar fi Git, software de gestionare a locurilor de muncă, integrare continuă și implementare continuă (CI/CD) la cazurile de utilizare educațională** cu intenția de a ajuta profesorii și studenții să petreacă mai mult timp învățând.

Ca parte a tezei, abordăm sarcina de a gestiona clustere instituționale și de a le integra atât în mediile de cercetare, cât și de producție pentru servicii instituționale. **Teza implementează o strategie de gestionare a clusterelor bazată pe mașini virtuale, integrând servicii de mare disponibilitate și plan de control care sunt instalate automat**. Utilizarea mașinilor virtuale permite administratorilor de sisteme să îmbunătățească securitatea clusterului și izolarea serviciilor, oferind în același timp utilizatorilor flexibilitate în implementarea propriilor sarcini de lucru și controlul mediului de lucru, necesitând implicarea minimă din partea administratorilor. Furnizarea de resurse disponibile este esențială pentru a încuraja utilizatorii să adopte o platformă de cluster, deoarece aceștia pot avea încredere în ea pentru a-și găzdui sarcinile de lucru și serviciile fără întreruperi.

Monitorizarea infrastructurilor cluster este o sarcină folosită de majoritatea implementărilor profesionale, deoarece permite vizualizarea tiparelor de utilizare a resurselor, care pot fi utilizate pentru prevenirea sau detectarea timpurie a problemelor și notificarea pentru răspunsul la incident. Fiind o sarcină atât de critică, există o gamă largă de soluții pentru a acoperi acest caz de utilizare, cum ar fi Grafana [24], CheckMk [3] și multe altele, fiecare acoperind cazuri de utilizare a calculatoarelor distribuite. **Această teză propune o arhitectură de referință pentru monitorizarea clusterelor** folosind Prometheus pentru colectarea punctelor de date și Grafana pentru vizualizarea datelor folosind tablouri de bord personalizate.

1.3 Structura Tezei

Teza urmează structura de mai jos:

1. Capitolul 1 definește domeniul de aplicare, obiectivele și contribuțiile majore ale tezei. Capitolul evidențiază nevoia de scalare a sistemelor distribuite nu numai prin achiziționarea mai multor resurse hardware, ci și prin reducerea overheadului software, integrarea de noi modele de resurse și adăugarea de suport pentru noi

tipuri de resurse.

2. Capitolul 2 aprofundează subiectul eficienței utilizării resurselor în mediile grid, concentrându-se pe căi de îmbunătățire a middleware-ului pentru a reduce utilizarea resurselor per nucleu, a crește eficiența transferului de fișiere și a adăuga noi tipuri de resurse pentru a obține mai multe resurse disponibile pe grid. Secțiunea 2.1 arată modul în care experimentul ALICE de la CERN gestionează transferurile de date, studiind transferurile efectuate în perioada 2022-2023, când datele experimentale au fost mutate în medii de stocare pentru arhivare sau prelucrare, discutând despre optimizări făcut pentru creșterea vitezei de transfer, ceea ce se poate aplica transferurilor de fișiere grid în general. Secțiunea 2.2 vorbește despre munca pe care am făcut-o pentru a adapta middleware-ul experiment ALICE pentru a gestiona joburi cu mai multe nuclee pe rețea, cu avantajul de a reduce utilizarea memoriei per nucleu, având mai puține zone de memorie duplicate

3. Capitolul 3 introduce necesitatea unei adoptări mai mari a digitalizării în educație, concentrându-se pe educația IT, care conține multe elemente care pot fi automatizate pentru a scădea numărul de sarcini repetitive efectuate de profesori și elevi. Secțiunea 3.1 explică munca depusă pentru construirea platformei openedubuilder, un middleware pentru automatizarea managementului materialelor de clasă; automatizarea creării și implementării materialelor folosind tehnici CI/CD și Git [35]. Secțiunea 3.2 propune utilitarul vmchecker, un middleware de verificare automată a temelor, care primește cererile elevilor și profesorilor de a rula teste pe codul scris de elev. vmchecker a fost construit cu un manager de verificare centralizat și integrând plugin pentru a funcționa cu platforma de e-learning Moodle [25].

4. Capitolul 4 oferă un șablon pentru organizarea și implementarea unui mediu cluster folosind virtualizarea pentru a fi utilizat în rularea unui site grid, oferind servicii specializate pentru o instituție de învățământ, folosind clusterul UNSTPB ca studiu de caz. Secțiunea 4.1 definește pașii care trebuie făcuți pentru a realiza o infrastructură cu disponibilitate ridicată pentru servicii de găzduire de resurse și pentru joburi de cercetare într-un mediu de cluster bazat pe mașini virtuale OpenStack, care oferă utilizatorilor posibilitatea de a găzdui servicii și sarcini sensibile care ar trebui să ruleze fără întrerupere. Secțiunea 4.2 descrie soluțiile curente de monitorizare utilizate pentru menținerea gradului de conștientizare a clusterului și alertarea administratorilor în cazul unor evenimente. Datorită numărului mare de soluțiilor de monitorizare disponibile, oferim descrieri pentru soluțiile disponibile în prezent și propunem un șablon pentru monitorizarea clusterelor și stocărilor utilizate pentru încărcăturile de lucru în rețea bazate pe Grafana, Prometheus și AlertManager.

Chapter 2

Îmbunătățirea Utilizării Resurselor Grid

Accentul în grid computing este pe a rula sarcinile de lucru cât mai reped posibil, răspândind procesarea pe mai multe clustere pentru a scala procesarea pe verticală. Adăugarea de noi resurse în mediul grid se face prin adăugarea de noi site-uri sau prin angajamente anuale [33], unde site-urile plănuiesc achiziționarea și integrarea hardware nou. Deoarece aceste procese sunt lente și iterative, sistemele actuale trebuie îmbunătățite pentru a permite mai puține cheltuieli generale și o eficiență crescută.

Aplicațiile middleware sunt sistemul nervos al unui mediu grid. Acestea coordonează mai multe site-uri, joburi, verifică diverse disponibilități ale sistemului, programează joburi, gestionează datele, aplică restricții de acces utilizatorilor și multe altele. Ca o resursă vitală, acestea sunt primul loc în care pot fi aplicate îmbunătățiri pentru a adăuga funcții. Ca parte a acestei teze, au fost identificate două căi posibile pentru creșterea eficienței utilizării resurselor în mediile grid prin modificarea middleware-ului gridului.

Locurile din gridul de calcul ALICE necesită fișiere de intrare care trebuie să fie localizate pe un sistem de stocare din apropiere. Fișierele trebuie să fie transferate după procesarea directă pe diferite site-uri pentru procesare ulterioară sau în stocarea arhivelor pentru a fi stocate pentru reprocesare. O metodă de creștere a vitezei de transfer pentru transferurile de fișiere în gridul ALICE a fost implementată și detaliată în Secțiunea 2.1.

Utilizarea memoriei este o problemă în procesarea paralelă, deoarece este o resursă care reprezintă o limită dură în ceea ce privește extinderea procesării. CPU-urile pot fi supra-abonate, mai ales în mediile de utilizare mixtă în care nu toate procesele sunt dependente de CPU, iar IO-ul poate fi saturat și comunicarea poate fi încă stabilită, dar lățimea de bandă actuală nu va scala. Pe de altă parte, memoria poate fi extinsă doar prin swapping, aducând o penalizare mare de performanță. În acest context, experimentul ALICE a căutat să reducă utilizarea memoriei per job prin implementarea joburilor multi-core care utilizează mecanismul de partajare a memoriei în Linux. Secțiunea 2.2 detaliază modul în care suportul de planificare a fost integrat pentru joburile care folosesc nuclee multiple în middleware-ul JAliEn utilizat de experimentul ALICE, capabil

să gestioneze joburi cu mai multe nuclee în mediul grid.

Următorul capitol prezintă modalitățile de a crește eficiența procesului în mediile grid și măsurile implementate în mediul grid ALICE pentru a crește utilizarea resurselor.

2.1 Îmbunătățirea Infrastructurii de Transferuri

Detectorul ALICE [17] a fost actualizat pentru LHC Run 3 (2022–2025) cu un sistem de achiziție de date fără declanșare. În acest mod, rata datelor este cu câteva ordine de mărime mai mare decât în datele anterioare luarea perioadelor. Printr-o compresie în timp real (O2) efectuată aproape de configurație experimentală, ALICE este capabilă să reducă dimensiunea fluxului de date de 10 ori, la aproximativ 100 GB/s [18]. Gestionarea eficientă a acestor date necesită o schimbare profundă a diferitelor niveluri de persistență a datelor și circulație.

Bufferul de disc ALICE O2 a fost proiectat și implementat ca element de stocare pentru fișierele comprimate de O2 care sunt stocate în fișiere de tip cadrul de timp comprimat (CTF) [43]. CTF-urile pot fi reduse în continuare la dimensiuni mai mici ale fișierelor cu cel puțin 95% prin procesul de skimming. CTF-urile reduse sunt arhivate după ce au fost convertite în fișiere tabulare de tip AO2D, care este utilizat pentru analiza datelor de fizică. Fișierul AO2D este un fișier de date bazat pe tabel care reprezintă experimentul reconstruit datele într-un mod ușor de analizat. Acestea sunt distribuite pe elementele de stocare grid și sunt folosite de fluxul de lucru de analiză.

De la începutul punerii în funcțiune a detectorului pentru Run 3 la sfârșitul anului 2021, gestionarea datelor de pe bufferul de disc O2 a prezentat o problemă, deoarece metode de a controla fluxul de date și de a-l adnota în funcție de tip nu a fost totuși complet proiectat și implementat. Acesta a fost mai ales cazul pentru date de calibrare și testare specifice detectorului, care nu sunt convertite în AO2D. Una dintre primele sarcini ale noului sistem de management al datelor a fost separarea diferitelor tipuri de date și pentru a le redirecționa către setul specific de elemente de stocare, de exemplu tipul de arhivă, și pentru a elimina datele procesate din discul buffer O2, păstrându-l astfel liber pentru CTF-urile primite. Proiectul constă și în îndepărtarea datelor cu calitate proastă și date nepotrivite pentru analiza - această acțiunea este realizată prin opțiuni stabilite în jurnalul de rulare al experimentului ALICE și grupul de coordonare al fizicienilor.

2.1.1 Configurația Sistemelor de Stocare în Experimentul ALICE

Gridul de calcul ALICE este alcătuită dintr-o varietate de soluții software și hardware pentru configurarea și gestiunea sistemelor de stocare și fiecare sistem de stocare oferind un nivel de performanță diferit în funcție de funcționalitate.

Figura 2.1 arată fluxul de date de la detectorul ALICE și de la producția Monte Carlo la diferitele sisteme de stocare și lățimea de bandă a acestora și capabilități. Diagrama nu prezintă datele citite de sarcinile de analiză.

Printre aceste sisteme de stocare pot fi utilizate trei sisteme principale pentru mutarea datelor:

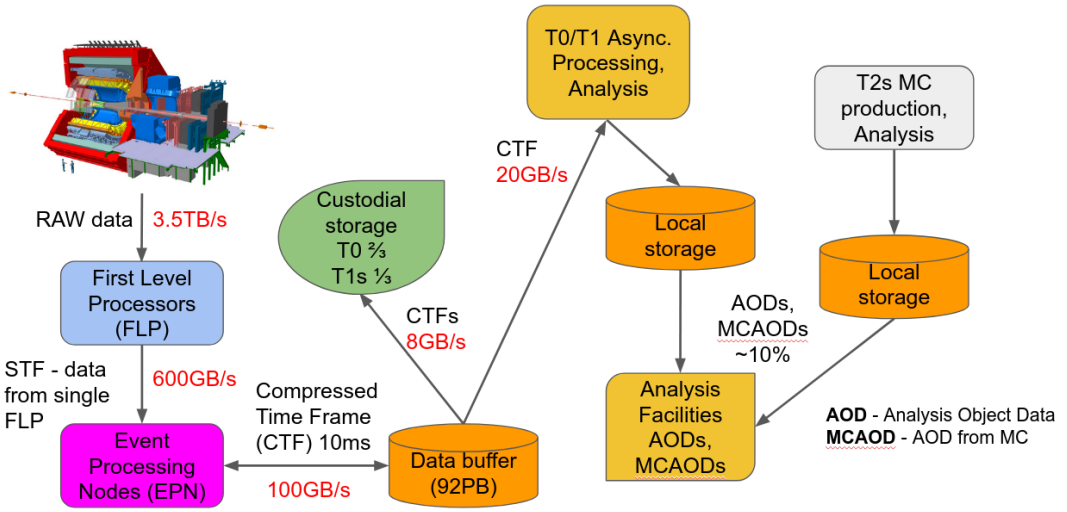


Figure 2.1: Fluxul de date al experimentului ALICE.

- EOSALICEO2 buffer, folosit pentru stocarea temporară a datelor brute și a fișierelor CTF înainte de procesare, skimmare și arhivare;
- EOSALICE, care este principalul sistem de stocare dedicat ALICE în centrul de date CERN. Acesta stochează fișierele AO2D produse de procesul reconstrucție și simulare MC, precum și rezultatul analizei fizice sarcini.
- CERN CTA este un sistem de stocare pe bandă magnetică (stocare de custodială) folosită pentru stocarea pe termen lung a fișierelor CTF și AO2D.

În gridul ALICE există aproximativ 60 de sisteme de stocare, notate în figură ca „Local storage" a centrelor corespunzătoare T1 și T2 și a site-urilor de analiză. Acestea joacă, de asemenea, un rol cheie în imaginea de gestionare a datelor.

Protocolul utilizat pentru transferul și gestionarea fișierelor pe sistemele de stocare și sistemele de stocare respective în uz sunt descrise în cele ce urmează.

XRootD

Experimentul ALICE folosește protocolul XRootD [16] pentru a transfera fișiere. Acest protocol este un standard pentru experimentele de fizică, așa cum este interfață cu biblioteca ROOT [23] și sistemul său I/O. ROOT este un framework utilizat pe scară largă în comunitatea HEP pentru a scrie aplicații reconstrucție și analiză software. Framework-ul folosește drept container fișierele `.root` cu structură internă asemănător cu un director de fișiere UNIX. Un fișier `.root` poate conține directoare și obiecte organizate în număr nelimitat de niveluri și stocate pe mașină în format independent. Protocolul XrootD este conștient de structura internă a acestor fișiere și folosește aceste cunoștințe pentru a efectua un transfer optim de date pe o rețea locală sau în internet (LAN și WAN) prin ajustarea dinamică a dimensiunilor buffer-ului de transfer și prin

citirea înainte a obiectelor întregi din fișierele `.root`.

EOS

EOS [30] este o soluție de gestionare a stocării construită la CERN folosind aplicația și protocolul XRootD, cu scopul de a oferi suport pentru protocolul XRootD pentru sisteme de stocare multi-PB într-un mod eficient. O instanță EOS este implementată pentru fiecare dintre cele patru experimente LHC la CERN și la multe site-uri grid din întreaga lume.

EOSALICE

Principalul sistem stocare de producție pentru ALICE este un element de stocare bazat pe EOS găzduit în centrul de date CERN IT. Este construit din JBOD-uri ieftine de înaltă densitate conectate la nodurile de server front-end prin carduri SAS HBA de mare viteză. Datele stocate pe această instanță sunt folosite pentru a rula joburi situate la CERN, ceea ce înseamnă că sarcinile citesc și scriu cantități mari de date în acest spațiu de stocare [19]. Acest lucru face ca spațiul de stocare EOS ALICE să fie cel mai important element de stocare din gridul ALICE.

EOSALICEO2 Buffer

Bufferul EOSALICEO2 este zona de stocare în care se află rezultatele în timp real din compresia datelor rezultate prin procesarea evenimentelor de către nodurile de tip EPN. Bufferul trebuie să ofere spațiu suficient pentru a stoca CTF-urile pentru o perioadă de preluare a datelor Pb–Pb, de ordinul a 80PB cu o rată medie de intrare de aproximativ 100 GB/s. După sfârșitul perioadei Pb–Pb, datele sunt citite și procesate în fișiere AO2D, care sunt stocate pe grid în elemente de stocare pentru analize ulterioare. Totalitatea CTF-urilor Pb–Pb sunt copiate de pe EOSALICEO2 în sistemul de stocare custodial, la CERN (2/3 din volum) și la 6 centre T1 (1/3 din volum). Reconstrucția se desfășoară pe site-ul CERN și T1 sisteme și pe nodurile EPN în afara perioadelor de preluare a datelor.

Arhivarea Folosind Banda Magnetică

Arhivele pe bandă magnetică sunt o formă de stocare pe termen lung potrivită pentru păstrarea datelor utilizate rar. Costul, exprimat în preț pe terabyte, este încă considerat [26] mult mai avantajos decât pentru stocarea pe hard disk iar sistemele moderne de bandă oferă un mare nivel de redundanță a datelor, făcându-l astfel un format foarte sigur. Dezavantajele utilizarea suportului pe bandă este viteza de ieșire mai mică și accesul liniar, în plus față este și complexitatea inerentă și suportul special necesar. Din moment ce bariera de intrare pentru înfiinţare şi operarea arhivelor de bandă magnetică este destul de mare, acestea sunt desfășurate numai în centrele de calcul mari.

2.1.2 Middleware-ul de Transferuri al experimetnului ALICE

Experimentul ALICE produce, stochează și citește o cantitate mare de date. Prin urmare necesită un framework robust care poate gestiona numărul considerabil de operații cu fișiere (copiați, mutați, citiți, eliminați) care se fac zilnic.

Această subsecțiune evidențiază modul în care fișierele și transferurile de fișiere sunt gestionate de serviciile centrale ALICE.

Înregistrarea Fișierelor CTF

Fișierele CTF care sunt generate de EPN-uri sunt stocate inițial pe SSD-urile interne ale nodurilor. Acestea sunt mutate în memoria buffer EOSALICEO2 utilizând un sistem special, EPN2EOS, care copiază datele, înregistrează fișierele în catalogul ALICE și, după înregistrarea cu succes, le șterge din EPN SSD.

În catalogul ALICE, fiecărui fișier îi sunt asociate următoarele logice și fizice atribute la înregistrare:

- Un LFN, care este folosit pentru a face referire la un fișier într-un format compatibil POSIX, folosind o cale într-un sistem de fișiere, de exemplu */alice/data/2023/LHC23a/539457/ctf_run00539457.root*;
- Un GUID, folosit pentru a identifica fișierul în mod unic, care este echivalentul unui numărul de inod într-un sistem de fișiere UNIX. Exemplul de fișier de mai sus are următorul GUID:
  cfa6adab-1ac0-11ee-a3ab-0242f0fa34f9;
- Un nume de fișier fizic (PFN), care conține locația fizică a unui fișier și protocolul pentru a-l accesa. Un fișier poate avea mai multe PFN, dacă are mai multe replici activate diferite noduri de stocare. Exemplu pentru un fișier cu 2 replici:
  pfn = root://eosalice.cern.ch:1094/13/19787/cfa6adab
  pfn = root://mgm.spacescience.ro:1094/13/19787/cfa6adab;
- ACL, care arată dreptul de proprietate asupra fișierului și permisiunile de acces.

Gestionarea Transferurilor

Întregul sistem ALICE Grid Central Services este construit în Java și aceasta include software-ul de gestionare a transferurilor. Un utilizator poate solicita transferuri de fișiere fie printr-un client Java independent sau folosind `alimonitor` [1].

Când este solicitat un transfer, un apel este trimis către o instanță de servicii centrale cu o listă de fișiere de transferat, un mesaj de transfer pentru a-l identifica, a SE sursă și țintă și natura operațiunii - mutare sau replicare. Instanța Serviciilor Centrale primește cererea și o adaugă la `transfer_requests` tabelul unei baze de date PostgreSQL[13]. Fiecare cerere de transfer are un ID, o destinație, un nume și în în cazul operațiunii de mutare, la succesul transferului de fișiere, replica va fi ștearsă pe elementul de stocare marcat ca sursă. De asemenea, intrarea în baza de date va conține intrări pentru fiecare transfer de fișiere, astfel încât contoarele de stare să fie stocate într-un mod persistent. Baza de date va determina starea unui transfer, care poate fi RUNNING, SUCCESFUL sau ERROR, dacă unele dintre transferuri nu au putut fi efectuate. Un transfer a avut succes dacă nu au existat erori pentru întregul set de dosarele solicitate.

Optimizări de Transfer

Pentru unele sisteme de stocare, de exemplu bufferul EOSALICEO2, SE-urile țintă sunt limitate la sisteme de stocare custodial datorită tipului de date. În astfel de cazuri există o optimizare statică care limitează lățimea de bandă de transfer la fiecare SE individual și stabilește o porțiune specifică a datelor care ar trebui să fie primite de SE țintă. Pentru celelalte transferuri, optimizarea metodele sunt descrise mai jos.

Transferurile Clienților Primul tip de optimizare este în ceea ce privește numărul de transferuri concurente care sunt permise pe stocare. Dacă numărul de fișiere care sunt în curs transfer la un element de stocare este mare, dar dimensiunile fișierelor este mică, lăţimea de bandă maximă poate fi menţinută numai dacă un număr mare de transferuri este început în paralel. Acest lucru se întâmplă deoarece timpul de transfer este dominat de supra-sarcina de configurare mai degrabă decât operația de copiere în sine. Trimiterea de fișiere mari cu multe fire de execuție este, de asemenea, sub-optimă, deoarece viteza va fi deja limitată de debitul disponibil al rețelei sau de lățimea de bandă limitări ale stocării receptoare.

Astfel, limita de transfer a clientului este un echilibru al celor două cazuri și este stabilită per element de stocare țintă printr-o configurație stocată într-o bază de date bazată pe LDAP. În prezent, această valoare trebuie schimbat manual, pentru a reflecta evoluția elementelor de stocare țintă sau lățimea de bandă a rețelei, acesta este un compromis operațional rezonabil elementele de stocare și modificările WAN sunt rare.

Thirt Party Copy O modalitate simplă de a copia folosind `xrdcp` este să fie transferate fișierele dintr-unul punct final la altul prin gazda care inițiază transferul. Acesta este modul în care comanda `xrdcp` operează implicit, deoarece este modul predominant de client interacțiunea cu stocarea pe un nod de procesare. Această metodă pune o limitare severă a transferurilor de date între nodurile de stocare existente Structura grid eterogenă, unde cele două elemente de la/la care ar trebui date poate fi copiat la o distanță foarte mare, crescând astfel considerabil latența transferului. Aceasta prezintă, de asemenea, o problemă de scalare, deoarece gazda va direcționa tot traficul și mai multe sisteme trebuie să fie setate pentru a asigura o lățime de bandă suficientă pentru toate transferurile solicitate. Third Party Copy (TPC) este un mecanism care permite inițierea procesului de copiere de o mașină centrală cu transferul efectiv de date realizat direct între sursă şi noduri de stocare destinație. Acest lucru permite în mod natural un flux de date mai eficient și mai direct, și permite utilizarea uneia sau mai multor mașini centrale pentru a gestiona toate transferurile de date. TPC este o opțiune complet implementată a `xrdcp` și a tuturor elementelor de stocare ALICE.

2.1.3 Rezultatele Campaniei de Transfer din 2022

Această subsecțiune prezintă ciclul de viață al datelor pe bufferul EOSALICEO2 și reglarea diferitelor operațiuni de transfer la elementele de stocare secundare.

Ciclul de Viață al Datelor pe Bufferul EOSALICEO2

Figura 2.2 arată ciclul de acumulare a datelor pe bufferul EOSALICDEO2 în perioadele de preluare a datelor din anul 2022 (iulie până în octombrie), acceleratorul LHC de

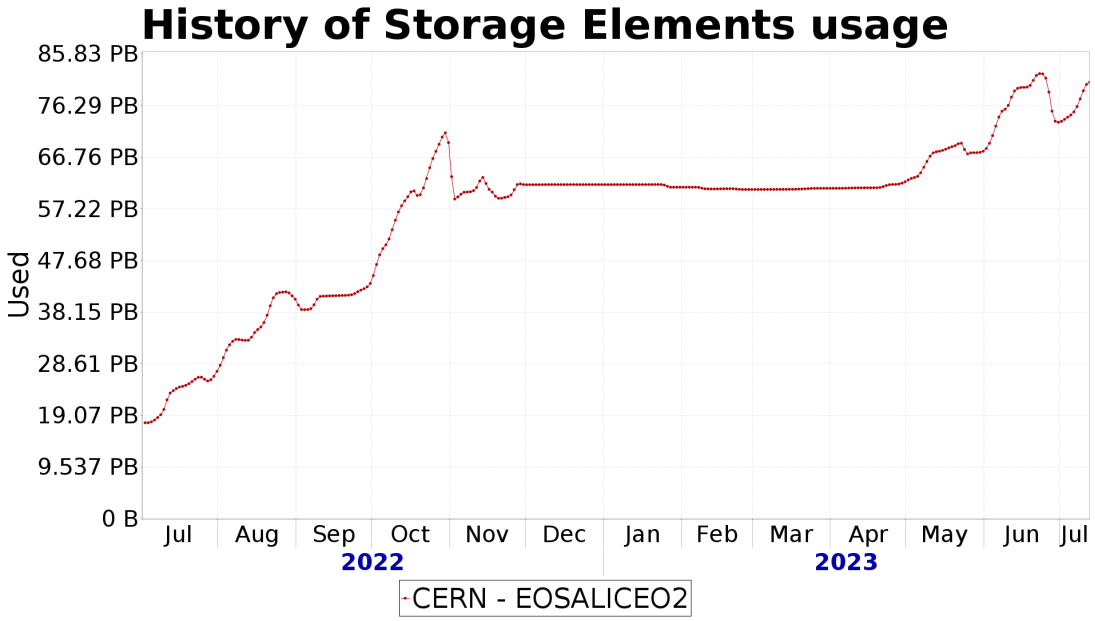


Figure 2.2: Acumularea datelor în bufferul EOSALICEO2.

la sfârșitul anului și perioade de preluare a datelor din anul 2023 (mai până în iulie). Volumul tipic de 10 PB/lună de date este colectat în timpul perioadelor de preluare a datelor și pe tot parcursul anului, o parte din el este îndepărtată din buffer, indicată de panta descendentă a curbei. În perioada de timp din figura 2.2, bufferul a primit 146 PB de date de la EPN-uri, dintre care 17,7 PB au fost transferate și 54 PB au fost șterse. Trebuie remarcat faptul că disponibilitatea bufferului EOSALICEO2 a fost peste 99,8% pe parcursul întregii perioade și 100% în timpul preluării datelor. Restul de 0,2% sunt în perioada de nefuncționare programată pentru upgrade-urile planificate ale software-ului EOS. Cea mai mare parte a datelor din buffer va fi eliminată înainte de perioada Pb–Pb din noiembrie 2023 pentru a face loc pentru volumul mare așteptat în timpul preluării datelor Pb–Pb.

Transferul de Date Către EOSALICE

Transferurile de date de la EOSALICEO2 la EOSALICE sunt una dintre căile standard de date în ciclul de gestionare a datelor ALICE. Figura 2.3 arată o rată tipică și frecvența transferurilor către EOSALICE. Include doar datele transferate de la EOSALICEO2 și nu arată datele scrise prin procesul de reconstrucție sau analiza datelor, care este substanțial mai mari.
Volumul total de date transferate în perioada din Fig. 2.3 este de 221 TB, iar viteza este limitat la aproximativ 20 GBps în configurația instrumentelor de transfer. Deși mai sus viteza este ușor de realizat, această valoare a fost aleasă pentru a minimiza efectul asupra citirii și scrierii de date cu volum mare pe instanța EOS.

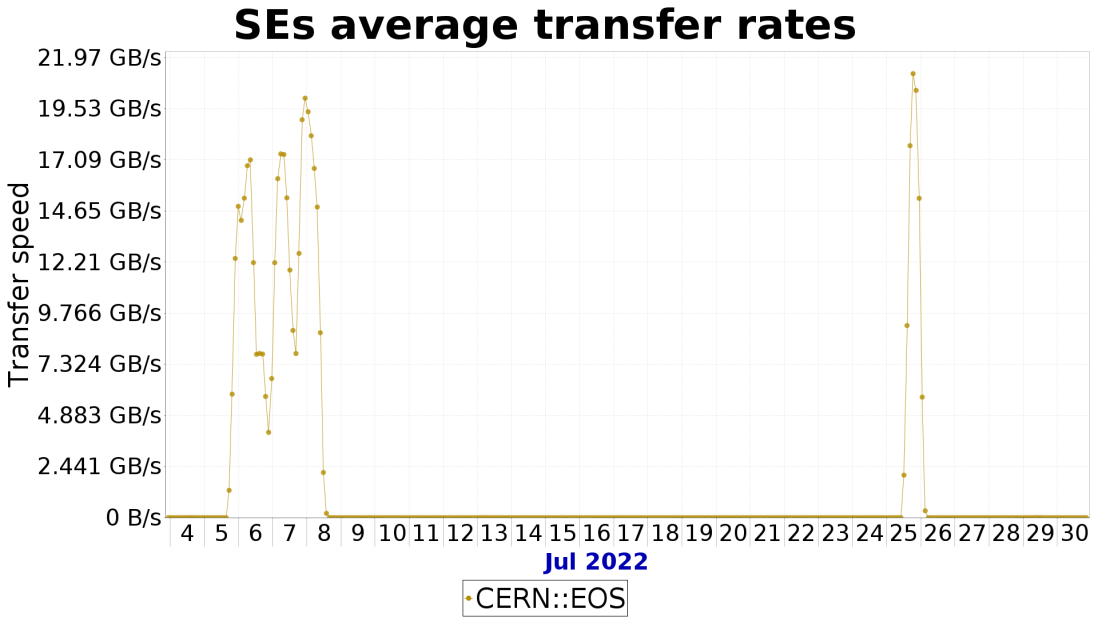


Figure 2.3: Viteza de transfer EOSALICEO2 la EOSALICE.

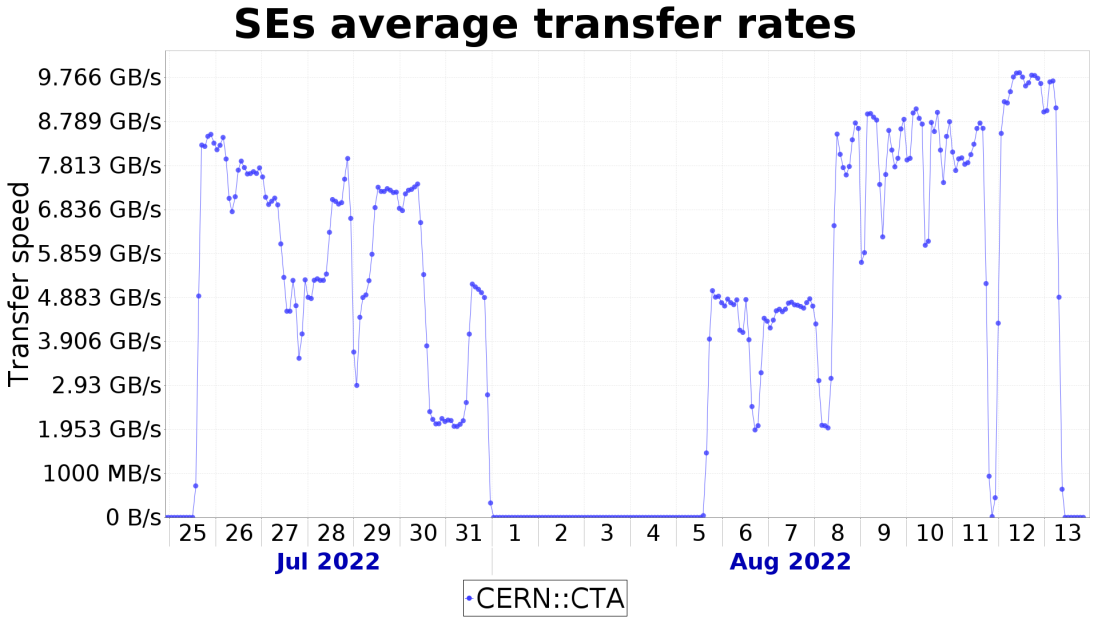


Figure 2.4: Transfer către arhiva CERN în iulie-august 2022.

Transferuri Către Depozitul de Custodie CERN

Două perioade de transfer de date sunt prezentate în Fig. 2.4 și 2.5. În perioada iulie – august 2022, datele acumulate între noiembrie 2021 și iunie 2022 au fost mutate la arhiva pe bandă magnetică. Fișierele au dimensiunea de 1 GB cu mici variații și 1250 fire simultane au fost suficiente pentru a satura lățimea de bandă setată de 10 GBps. În perioada inițială din iulie, stocarea custodială a fost reglată pentru performanță, indicată de variațiile frecvente ale viteza de transfer. Pe 5 august am început transferul

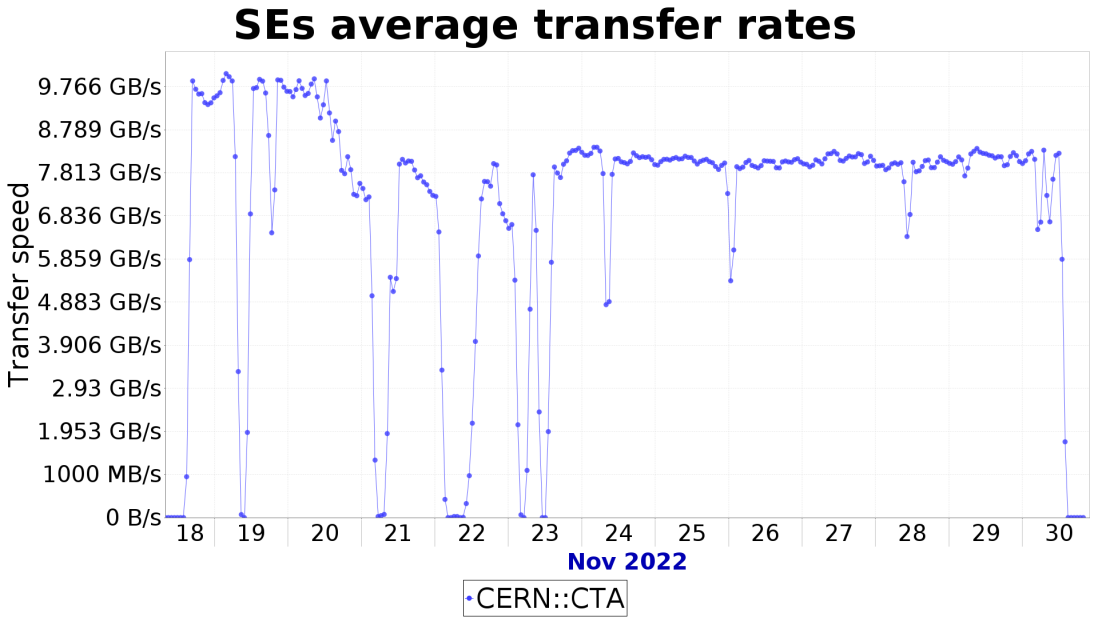


Figure 2.5: Transfer către arhiva CERN în noiembrie 2022.

rulajelor din octombrie 2021, cu fișiere de dimensiuni mai mici. Acest lucru a necesitat o creștere a numărului de transferuri paralele la 3000, realizat pe 8 august. Cu acest parametru final în vigoare, viteza nominală de 10 GBps a fost atinsă.
În noiembrie 2022 a existat un alt set de transferuri de fișiere mari, așa cum se arată în Fig. 2.5 cu viteză stabilă pentru stocarea în custodie. Pe 21 noiembrie, această viteză a scăzut la aproximativ 7,5 GBps datorită concurenței mari a operațiunilor cu bandă din celelalte experimente și o ușoară scădere a cotei de benzi magnetice pentru ALICE.

În total, au fost transferați 17,7 PB de date.

Controlul Lățimii de Bandă de Transfer

Una dintre metodele standard de a controla lățimea de bandă de transfer către orice sistem de stocare este numărul de fire paralele. Având în vedere o dimensiune medie cunoscută a fișierului și dacă este disponibilă lățimea de bandă nu este un factor limitator, crescând sau micșorând numărul de fire are ca rezultat creșterea/scăderea aproape liniară a vitezei de transfer. Acesta este o banală, dar foarte eficientă metodă de control și este cea primară folosită în transferul ALICE sistem. Viteza de transfer este afectată și de costul general de setare a fiecărui individ totuși, are o greutate mai mică pentru dimensiuni rezonabile ale fișierelor (de ordinul lui 100 MB+). Devine dominant doar pentru fișierele foarte mici (de ordinul a zeci de KB).
Efectul numărului de fire asupra vitezei de transfer este ilustrat în Fig. 2.6. În perioada 5–13 august se datorează saltul de la 8 august schimbarea de la 1250 la 3500 fire paralele (factor 2,8) pentru a se adapta pentru dimensiunea mică a fișierelor transferate. Viteza de transfer a crescut de la 4,3 GBps la 9,8 GBps, un factor de 2,3. Diferența de aproximativ 20% dintre cei doi factori se datorează supraîncărcării de transfer și a unor probleme în elementul de stocare receptor.

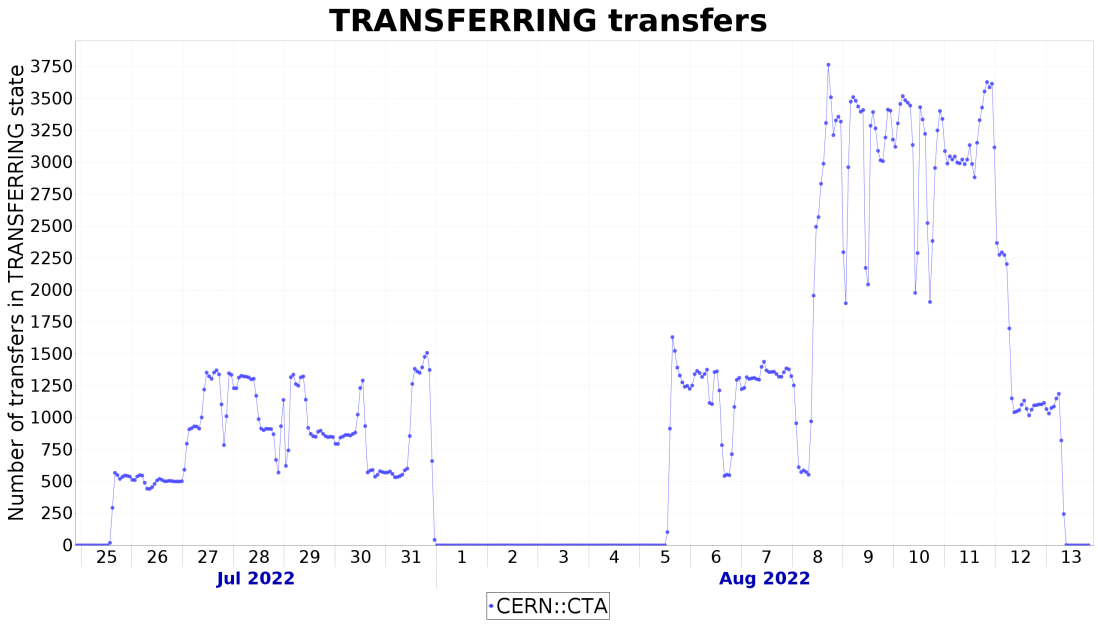


Figure 2.6: Numărul de clienți de transfer către bandă magnetică.

2.1.4 Concluzii și Direcții Ulterioare

Sistemul de management al datelor experimentale ALICE a fost actualizat pentru LHC Run 3 pentru a face față volumelor considerabil mai mari de date. Un nou mediu de stocare, EOSALICEO2, a fost introdus în sistem, capabil să să găzduiască mai mult de 100 PB de date și să suporte rate de date susținute de ordinul a 100 GBps. În același timp, gestionarea datelor celor peste 60 de sisteme de stocare din întreaga lume au fost actualizate cu noi metode și instrumente de transfer capabile de controlarea fluxurilor de date automat și adaptarea la variațiile condițiilor de un sistem de calcul distribuit. Noul management al datelor este pe deplin în producție și și-a demonstrat abilitățile de a stoca și distribui date de peste 150PB pe an și în același timp să mențină starea de funcționare a elementelor de stocare, care deservesc un număr mare de noduri de calcul care fac reconstrucția și analiza datelor.

În viitor vom continua să actualizăm sistemul de management al datelor cu mai multe funcții automate, una dintre acestea este declanșatoarele de transfer dinamic. Acest nou mod de funcționare va fi activat la finalizarea unei acțiuni anterioare, de exemplu, sfârșitul procesării unui set de fișiere și va permite mutarea, copierea sau ștergerea setului de fișiere. Acesta poate deasemenea să declanșează o operațiune, de exemplu începerea analizei asupra unui set de date la replicarea acestuia la un anumit nod de stocare.

2.2 Scăderea Utilizării Memoriei prin Integrarea Joburilor cu Nuclee Multiple

Gridul este în continuă evoluție, iar resursele disponibile cresc. Pentru a profita la maximum de ele, serviciile și clienții frameworklui au evoluat și ele și au fost reimplementate

în noul framework. Această reimplementare a permis implementările să fie realizate mai ușor și cu o scalabilitate mai mare. Au fost îmbunătățite și mecanismele de autentificare și autorizare, utilizând certificatele furnizate de Autoritatea de Certificare JAliEn X.509.

Gridul ALICE este un efort de a pune în comun resursele de la diferite instituții care au scopul comun de a analiza evenimentele capturate de detectorul ALICE. Prin combinarea resurselor, fizicienii pot folosi mai multe resurse la un moment dat decât le are propria instituție la dispoziție, ceea ce duce la avansarea și iterația mai rapidă a obiectivelor cercetătorilor. Gridul ALICE reunește elementele de calcul, de stocare și resursele de rețea ale multitudinii de centre de calcul distribuite din jurul lumii. Acest lucru permite cantitățile masive de date colectate să fie procesate și analizate de către fizicieni într-un timp relativ scurt și asigură creșterea necesară a resurselor combinate pe măsură ce sunt mai multe date colectate de experiment în frameworkul operațiunii LHC în curs.

Software-ul care permite cercetătorilor să ruleze joburi pe ALICE Grid este middleware-ul JAliEn. Este alcătuit din serviciile JAliEn Computing Elements care rulează pe noduri dedicate pe fiecare site (numite VOBox-uri), care se conectează la infrastructura lor și trimit joburi generice din Grid către cluster. O altă componentă a frameworklui sunt Serviciile Centrale, situate la CERN, care funcționează ca un punct central de management care trimite joburi pentru a rula pe site-uri, gestionează fișierele de intrare și de ieșire cerute de joburi, le catalogează și orientează clienții către Noduri de stocare în grid în care să-și scrie rezultatul.

Pentru LHC Run 3, experimentul ALICE și-a modernizat detectorul și a schimbat modelul de achiziție de date de la un mod declanșat la unul de streaming, cu creșteri bruște ale lățimii de bandă și ale cerințelor de stocare. Schimbarea paradigmei de la procesarea evenimentelor la vizualizarea cadrelor de date continue de 10 ms a necesitat o rescrie completă a software-ului experimental, de la simulare și reconstrucție până la frameworkul de analiză. Pentru o procesare eficientă a noului tip de date, frameworkul necesită alocații mai mari de memorie fizică per job, de ordinul 10 până la 20 GB. Schimbarea lor nu este o opțiune, deoarece întregul conținut al fișierului de date este accesat și astfel eficiența procesorului ar fi afectată dramatic.

Noul software de experiment este multi-proces și permite utilizarea eficientă a mai multor sloturi de bază pe site-uri, menținând în același timp cererea existentă de 2 GB per core ratio de la furnizorii de resurse. Această teză abordează modificările care au fost aduse experimentului Grid middleware JAliEn pentru intermedierea și rularea joburilor multi-core.

2.2.1 Arhitectura Middleware-ului JAliEn

Mediul Java ALICE (JAliEn) este middleware-ul folosit de experimentul ALICE pentru a stoca, procesa și analiza datele obținute din fenomenele fizice [29]. Este o evoluție a vechiului middleware AliEn [20] și a fost dezvoltat ținând cont de nevoile în evoluție ale viitoarei LHC Run 3. Infrastructura Worldwide LHC Computing Grid (WLCG) [33], care este compusă din mai mult de 200 de centre de calcul din 39 de țări din întreaga lume, este utilizat pentru funcționarea sa.

Frameworkul JAliEn are două tipuri de servicii, dintre care unele rulează pe fiecare dintre site-urile Grid și altele care sunt rulate central pe serverele găzduite la CERN. Figura 2.7 arată cum sunt organizate componentele principale ale frameworkului și cum se leagă între ele.

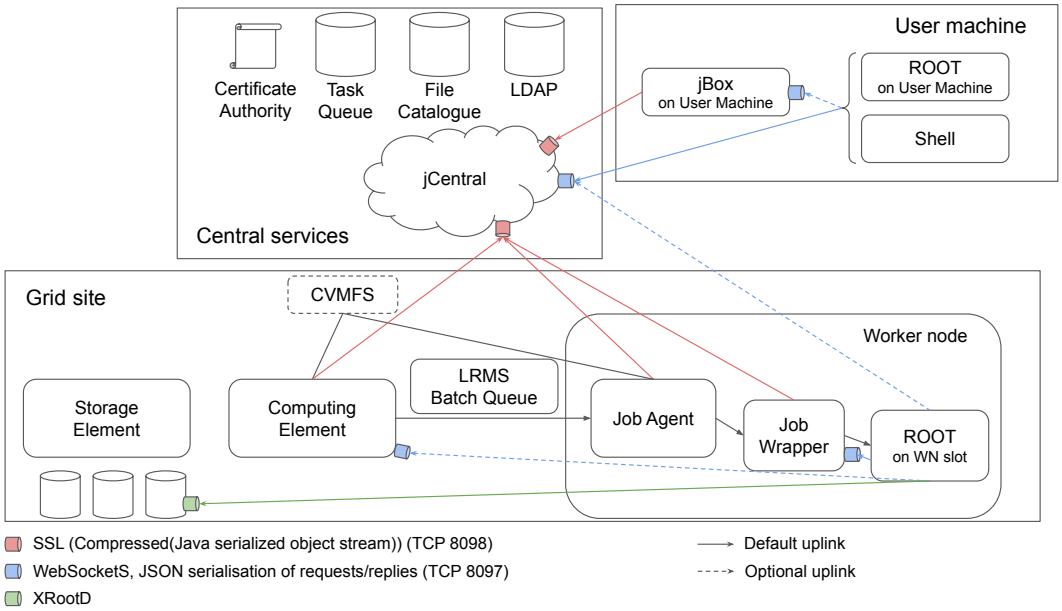


Figure 2.7: Diagrama principalelor componente care integrează frameworkul

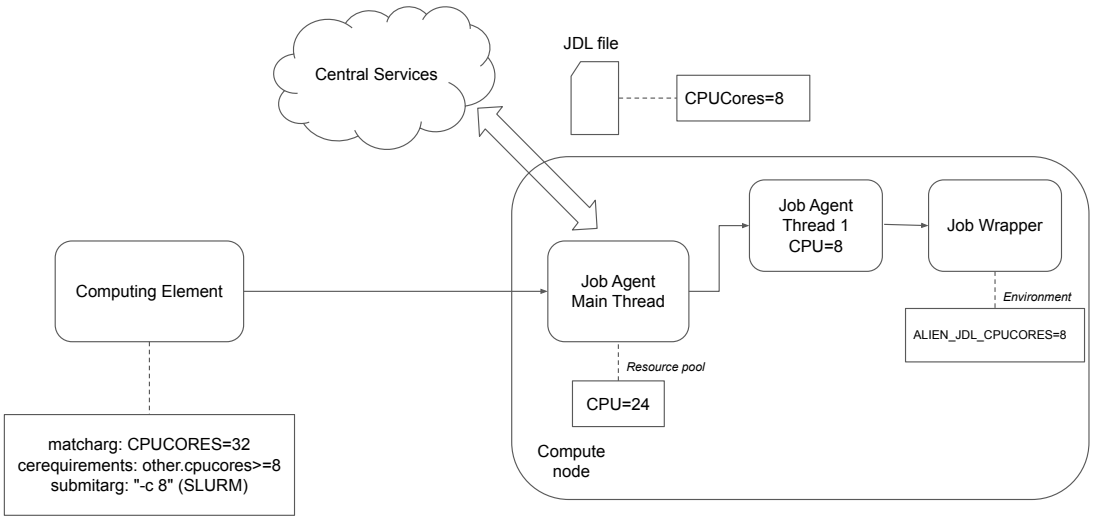


Figure 2.8: Diagrama definiției parametrilor CPU Cores și mecanismul utilizat pentru executarea joburilor cu mai multe nuclee.

Servicii Centrale

Frameworkul are o arhitectură centralizată, ceea ce înseamnă că jobul executat în diferite locații ale WLCG se conectează la Serviciile Centrale (CS) de la CERN. Punctul de gestionare centralizat trimite joburi pentru a rula pe site-uri, gestionează fișierele de intrare și de ieșire cerute de joburi, le cataloghează și direcționează clienții către nodurile de stocare Grid adecvate pentru a-și scrie rezultatul. Având entități centrale de gestionare a datelor și a sarcinilor, sistemul oferă capabilități de echilibrare a joburilor între multiplele sale site-uri.

Site-ul Grid

Gridul ALICE este compus în prezent din 53 de site-uri unde se execută joburi care studiază fenomenele fizice. Programarea joburilor ține cont de localitatea datelor (conform catalogului central de fișiere) pentru a minimiza latența IO și astfel resursele consumate de job.

Computing Element Computing Element (CE) este poarta de acces către fiecare dintre site-urile din Grid. Acesta folosește un punct de prezență persistent pe site numit VOBox (Virtual Organization Box) și de acolo trimite joburi generice la coada de așteptare a site-ului prin intermediul unui Sistem Local de Management al Resurselor (LRMS), la instrucțiunile de la Serviciile Centrale. CE îndeplinește și funcții de monitorizare și contabilitate a resurselor.

Noduri de Lucru Nodurile de lucru sunt mașinile unde se execută joburile, fiind configurate cu un mediu care conține toate instrumentele necesare. Acest mediu poate fi configurat fie direct pe mașină, fie într-un container.

Job Agent și Job Wrapper Joburile generice începute de LRMS lansează instanțe Job Agent (JA) pe nodurile de lucru. Ele se conectează la Serviciile Centrale și își publică resurselor disponibile, primind un job care se potrivește cu constrângerile. Apoi JA furnizează un Job Wrapper (JW) care execută sarcina utilă reală într-un director cu nisip sau într-un container, acolo unde este disponibil.

2.2.2 Integrarea Suportului pentru Joburi cu Multi-core în JAliEn

Natura ratei ridicate de transfer a noului experiment necesită o schimbare în frameworkul de analiză, din cauza abundenței mari de date, acest framework va funcționa diferit și va trebui să funcționeze într-un mod multi-core. Trebuie să fim capabili să oferim software-ului suport middleware. Scopul nostru este să găsim o modalitate prin care să gestionăm acest nou tip de locuri de joburi prin refactorizarea modulului în care facem planificarea, gestionarea resurselor și lansarea joburilor. O consecință a acestei refactorizări este că acum putem rula software-ul nostru într-un mod eficient pe o coadă de programare a întregului nod, datorită componentelor de gestionare a resurselor pe care le-am adăugat.

În consecință, prelucrarea și analiza datelor va necesita o schimbare a frameworkului de analiză, precum și o creștere a resurselor utilizate în acest scop. În ceea ce privește resursele CPU, se estimează că nevoile aproximative vor crește cu 35%. frameworkul

a trebuit să fie adaptat pentru a-și crește lățimea de bandă, utilizând în același timp resursele disponibile mai eficient. Această teză prezintă implementarea logicii care permite executarea joburilor multi-core, pe lângă faptul că continuă să ofere suport pentru joburi single-core. Modificările au implicat o refactorizare în mai multe domenii, cum ar fi programarea joburilor, managementul resurselor și lansarea joburilor.

În sistem înainte de implementarea noastră - Job Agent-ul nu controla nimic despre resurse. Ceea ce făcea a fost să ceară un job. Acest lucru nu este suficient atunci când vrem să gestionăm slotul cu mai multe nuclee - vrem mai multă flexibilitate (să fie permis să rulăm mai multe tipuri de joburi). Prima dintre aceste modificări structurale este în domeniul programării sloturilor, implicând modificări ale arhitecturii Sistemului Local de Management al Resurselor (LRMS) utilizate.

Pentru a rezolva această problemă, implementarea bazată pe LRMS care rulează a fost modificată pentru a permite multi-threading, permițând middleware-ului să declanșeze jobul care rulează fire de execuție. Această abordare are beneficii nu numai în ceea ce privește programarea, ci și în ceea ce privește consumul de resurse RAM, întrucât noul model nu face uz de o instanță JVM pentru fiecare dintre Job Agent-urile lansate. Procesul de solicitare a noilor locuri de muncă, adică comunicarea cu Serviciile Centrale, a variat și din cauza naturii neomogene a Rețelei. Implicarea thread-urilor Job Agent în acel proces este necesară pentru a putea configura granularitatea joburilor care urmează să fie executate în funcție de resursele disponibile la un moment dat.

Un nou parametru, *CPUCores*, a fost adăugat la sintaxa JDL pentru a permite definirea joburilor multi-core. Cu acest nou parametru, numărul de nuclee care urmează să fie utilizate este definit și apoi luat în considerare de către Serviciile Centrale la programarea noilor joburi.

În plus, definiția site-urilor a fost extinsă pentru a include constrângerile acestora în ceea ce privește nucleele CPU disponibile și distribuția lor între sloturile de joburi anunțate. În definirea Computing Element-elor în sistemul LDAP au fost adăugați trei parametri care se referă la cerințele și posibilitățile fiecăruia dintre nodurile de lucru. Figura 2.8 ilustrează definirea parametrilor implicați și pașii urmați pentru executarea sarcinilor multi-core.

Înainte de a adăuga suport pentru execuția joburilor cu mai multe nuclee, frameworkul a fost capabil să ofere tuturor ciclurilor CPU doar un nucleu CPU pe slot.

Computing Element-ul a fost configurat pentru a prelua parametrul *matcharg:CPUCORES* din baza de date LDAP atunci când este inițializat. Acest parametru poate avea două comportamente: fie să fie setat să folosească o cantitate fixă de nuclee, care vor fi utilizate ca alocare implicită pentru slot, fie să fie declarat să folosească întregul nod, ceea ce înseamnă că va fi folosit pentru a detecta automat numărul de nuclee care vor fi utilizate și anunțate către Serviciile Centrale. În acest ultim caz, parametrului *matcharg:CPUCORES* i se atribuie o valoare de 0. În exemplul prezentat în figura 2.8, Job Agent poate folosi un total de 32 de nuclee CPU pentru executarea jobului. De asemenea, interogează baza de date LDAP pentru a vedea dacă există un număr predefinit de nuclee care trebuie alocate atunci când se trimit joburi în coada batch LRMS.

Apoi, comunică cu Serviciile Centrale pentru a primi joburi potrivite, iar Serviciile

Centrale răspunde cu fișierele JDL corespunzătoare. Odată ce Job Agent a generat threaduri pentru a rula aceste joburi, numărul de nuclee utilizate este scăzut din pool-ul de resurse și sunt alocate threadului în cauză. Întrucât resursele Grid nu sunt omogene, JobAgent-ultrebuie să publice resurselor disponibile către Serviciile Centrale pentru a primi locuri de muncă adaptate resurselor sale disponibile. Prin urmare, atunci când Job Agent comunică cu Serviciile Centrale pentru a solicita un nou job, acesta trebuie să raporteze numărul de nuclee CPU libere, precum și cantitatea de memorie RAM și spațiu pe disc disponibil. Pe baza acestor informații, planificatorul Serviciilor Centrale efectuează procesul de potrivire pentru a găsi un loc de muncă care corespunde cerințelor și îl adaugă la informațiile contabile. Lista 2.1 arată pașii de programare luați de JobAgent pentru saturarea unui slot, așa cum este descris mai sus.

```
1  retries = 0
2  TTL = getRunnerTimeToLive()
3
4  if cpuCores == 0 then
5      availableCores = min(getCpuCores(), getRamCapacity()/4)
6  else
7      availableCores = cpuCores;
8  end if
9
10 while availableCores > 0 and TTL > 0 and retries < 5 do
11     jdl = getNextJob(availableCores, TTL)
12
13     if jdl == null then
14         retries = retries + 1
15         sleep(300s)
16         continue
17     end if
18
19     retries = 0
20     availableCores = availableCores - jdl.JobCores
21     runningJobs = runningJobs + 1
22
23     startJob(jdl)
24 done
```

Listing 2.1: Algoritm de planificare JobAgent

Izolarea Sesurselor Într-un Context de Planificare a Întregului Nod

Deoarece noul frameworku lansează sub-procesele și le coordonează, în loc să facem toate calculele într-un singur proces, nu putem fi siguri că un singur job nu ar lansa atât de multe procese pe cât are sistemul de resurse libere. Am implementat un mecanism care menține consumul de resurse în limitele alocării inițiale.

Izolarea Procesoarelor În scenariul nostru de cercetare, dorim să ne asigurăm că nucleele CPU sunt utilizate numai de procesul căruia i-au fost alocate. Atunci când

un singur job este trimis către un site Grid, mecanismele interne ale LRMS ale acestuia care se ocupă de programarea sarcinilor sunt, de asemenea, responsabile pentru garantarea izolării resurselor. Deși planificarea întregului nod oferă frameworkului flexibilitate și are potențialul de executare eficientă a joburilor, ea în sine nu oferă izolarea resurselor pentru fiecare dintre procese. Din acest motiv, este necesară furnizarea sistemului cu instrumente suplimentare pentru a îndeplini această funcție. Sarcina de a găsi instrumente în acest scop nu este atât de ușoară pe cât ar părea, deoarece există multe constrângeri limitative, în mare parte cauzate de puținele privilegii acordate pe site-urile Grid.

Există o varietate de moduri în care utilizarea procesorului a unui job poate fi izolată, cum ar fi *cgroups* versiunile 1 și 2 [31], *isolcpus* [21] și *taskset* [15]. Fiecare dintre aceste opțiuni gestionează izolarea CPU în moduri diferite, iar unele site-uri au implementat deja unele dintre ele independent de frameworkul JAliEn.

Soluția Propusă Propunem să folosim comanda *taskset* pentru a fixa fiecare job la un set de nuclee de aceeași dimensiune cu numărul de nuclee CPU solicitate, pentru a ne asigura că joburile nu le pot depăși alocarea. Am implementat un mecanism care verifică nucleele care au fost deja fixate și, evitându-le, selectează un set de nuclee pe care să ruleze un nou job.

Deși fiecare situație are propriile sale particularități, putem distinge trei scenarii principale:

- Site-uri în care nu se aplică nicio constrângere asupra nucleelor CPU care urmează să fie utilizate. În acest caz, suntem liberi să folosim *taskset* și să fixăm procesele de încărcare utilă la nuclee explicite.
- Site-uri în care procesele noastre sunt deja constrânse să ruleze pe anumite nuclee. Afinitatea sarcinii care le va fi atribuită poate fi setată fie prin *cgroup* (*cpuset*), fie prin *taskset*. Deși configurarea se poate face folosind diferite instrumente, măștile de afinitate CPU ale proceselor pot fi văzute folosind comanda *taskset*.
- Site-uri în care sunt folosite containere. În aceste site-uri, *cpuset* sau *taskset* ar trebui să fie configurate de către administratorii de sistem deoarece nu este posibilă inspectarea proceselor externe din interiorul containerului, deoarece acestea sunt izolate.

2.2.3 Reducerea Observată a Utilizării Memoriei ș Utilizarea Cozilor Scavenging

Mecanismul JobRunner a permis executarea de noi fluxuri de lucru pe noi tipuri de sisteme. Pentru a evalua arhitectura actualizată de execuție a joburilor, am studiat numărul de joburi care au rulat până acum folosind noul mecanism, câte dintre aceste joburi sunt joburi multi-core și dacă obiectivul de a reduce cantitatea de memorie utilizată per miezul a fost realizat.

Rularea Joburilor Multi-Nucleu

Am ales să analizăm două tipuri diferite de joburi, deoarece acestea ne oferă o imagine de ansamblu asupra modului în care joburile multi-core vor rula pe Grid atunci când utilizarea lor crește popularitatea și pentru că au rulat într-un număr suficient de mare de noduri pentru a obține rezultate semnificative .

Aceste două tipuri de locuri de muncă sunt:

- Joburi de reconstrucție care convertesc datele brute ale experimentului în format AOD. Aceste joburi încarcă în memorie fișierele comprimate pentru a le analiza, astfel încât rulează pe site-uri cu o cantitate mare de memorie care poate găzdui fișierele brute analizate;
- Joburi de analiză, care utilizează pipe-uri și pornesc mai multe procese care citesc evenimente rezultate din conversia menționată mai sus sau din alte producții. Aceste joburi accesează un număr mare de fișiere în paralel și efectuează o cantitate mare de operațiuni I/O, așa că necesită o conexiune rapidă între nodurile de calcul și stocarea conectată la site.

Aceste două tipuri de joburi au cerințe diferite, așa că rulează pe site-uri Grid diferite. Din cauza situației fluide a software-ului de încărcare utilă și a faptului că acesta nu a fost rulat la scară largă, vom limita observațiile la două cozi: CERN-CORONA, care rulează joburi de reconstrucție, și Wigner-KFKI-8core, care execută joburi de analiză.

Am observat că, în timp ce atât CORONA, cât și Wigner rulează joburi multi-core, aceste joburi nu folosesc toate resursele pe care site-urile le oferă. Site-urile trebuie, de asemenea, să ruleze joburi cu un singur nucleu pentru a nu risipi resursele CPU. Acest lucru dezvăluie că Grid nu s-a mutat pe deplin la infrastructura de locuri de muncă multi-core.

Utilizarea Memoriei Per-Nucleu După cum era de așteptat, cantitatea de memorie RAM utilizată per job a crescut de la o medie de 3,17 GB, la o utilizare medie a memoriei de 9,04 GB. Acestea fiind spuse, dacă împărțim la numărul de nuclee, observăm că amprenta de memorie pe nucleu a scăzut la 1,13 GB. Aceste rezultate sunt în concordanță cu ceea ce ne așteptam, ceea ce înseamnă că vom putea programa mai multă putere de calcul per slot, deoarece resursa care îi lipsește la nivelul grid este în principal memoria. Prin reducerea utilizării memoriei, vom putea rula mai puține joburi care fac mai multă muncă de calcul.

Izolarea Utilizării Proceselor

Pentru a testa izolarea proceselor, am ales să rulăm un prototip de joburi de simulare care nu au fost încă implementate complet pe Grid. Folosim aceste joburi pentru testare, deoarece sunt cel mai probabil să ruleze pe mai multe procesoare decât este necesar, deoarece folosesc un mecanism de fork care este predispus să lanseze mai mult de opt procese care ar putea fi eventual programate în același timp.

Observăm din figura 2.9 că, deși joburile cu mai multe nuclee solicită opt nuclee, de fapt folosesc mai mult decât această cantitate de procesoare în medie. Există joburi a căror utilizare a procesorului crește până la 1600%, ceea ce înseamnă că folosesc complet 16

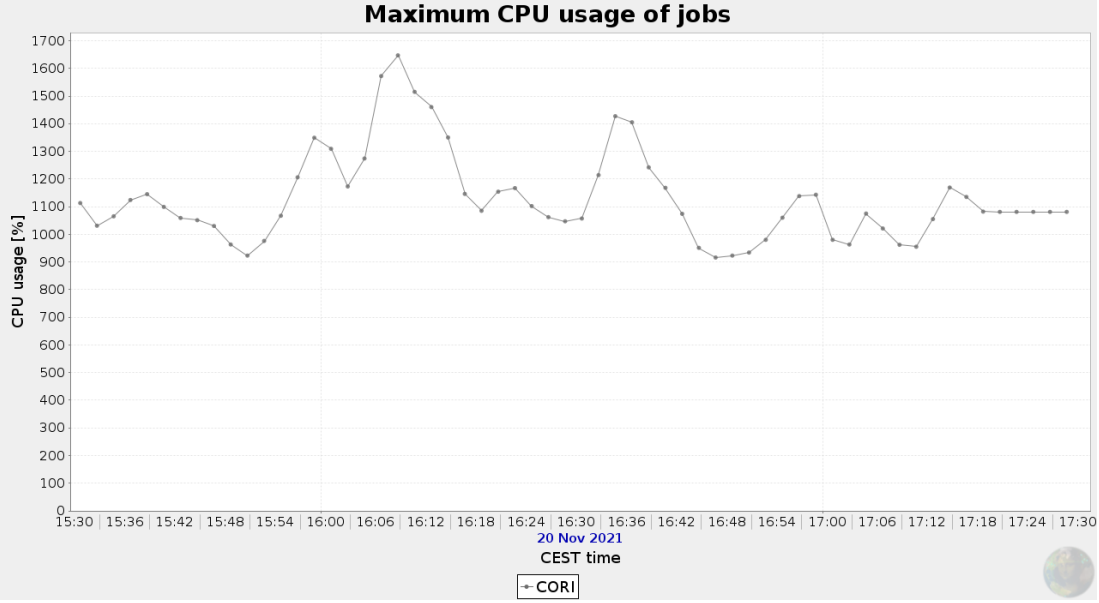


Figure 2.9: Utilizarea procesorului pe job înainte de izolarea sarcinilor

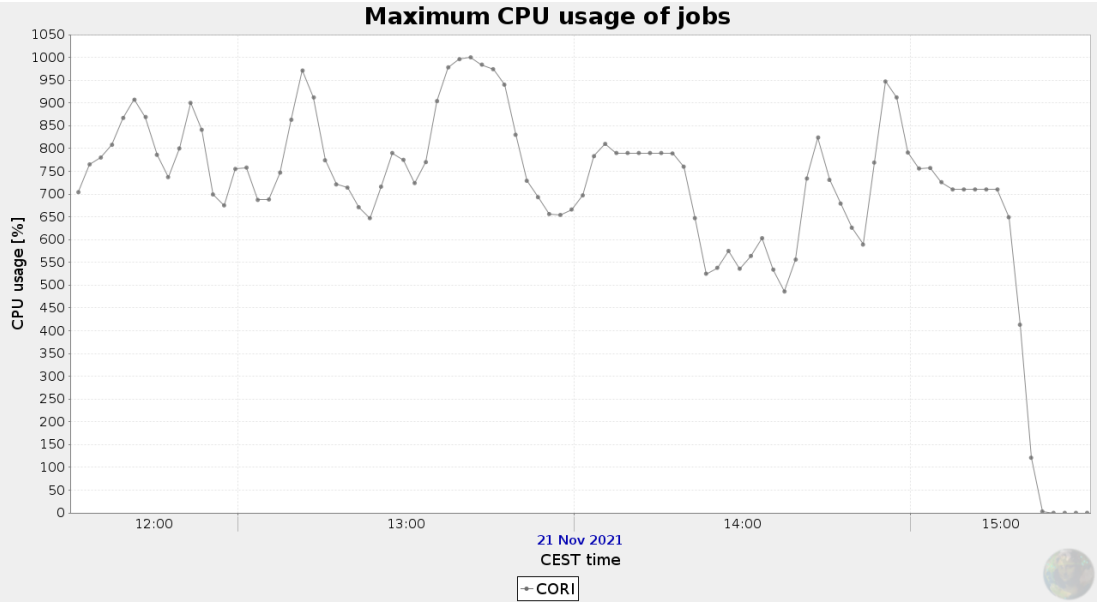


Figure 2.10: Utilizarea procesorului pe job după izolarea sarcinilor

nuclee. Ne așteptăm ca aceasta să fie o problemă care va escalada în viitor, pe măsură ce mai mulți fizicieni încep să folosească noul framework.

Figura 2.10 reprezintă utilizarea procesorului a joburilor după implementarea sistemului de izolare a sarcinilor. După cum era de așteptat, utilizarea procesorului a scăzut până la aproape 800%, echivalent cu 8 procesoare care lucrează la 100%. În grafic putem observa un vârf de 1000%, dar acest lucru este cauzat de metoda de contabilitate folosită de software-ul de monitorizare.

Desfășurarea Joburilor Oportuniste

Pentru a testa integrarea cu noi tipuri de coadă, am instalat JAliEn Computing Element pe două noi tipuri de resurse:

- Supercomputerul Cori [28], condus de Centrul Național de Calcul Științific de Cercetare Energetică și găzduit la Laboratorul Național Lawrance Berkeley, care a fost implementat ca loc de testare;
- Clusterul HPCS (High Performance Computing Service) găzduit la LBNL, care folosește cozi de eliminare.

Cele două sisteme selectate reprezintă resurse noi pe care Gridul ALICE le poate folosi deoarece site-urile au deja acces la ele, dar nu erau încă folosite pentru executarea joburilor.

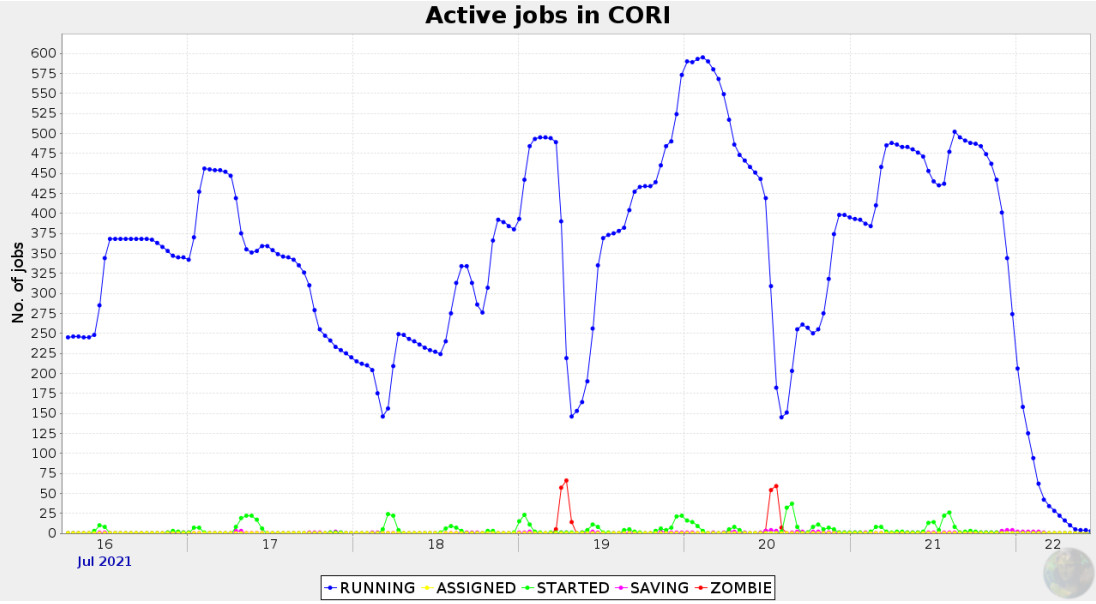


Figure 2.11: Distribuția joburilor pe sistemul CORI

Site-urile HPC După implementarea noului framework de joburi pe supercomputerul Cori, am reușit să un TTL (Time To Live) și rata de submisie a joburilor corecte, astfel încât să existe o cantitate constantă de joburi care rulează pe supercomputer. Acest flux constant de locuri de muncă poate fi văzut în figura 2.11.

Cozi Scavenging În implementarea pe clusterul HPCS, vedem că în timp ce joburile rulează, obținem alocări care sunt echivalente cu un site Grid mic. Acest lucru se întâmplă deoarece am ales să limităm numărul maxim de joburi care pot rula simultan pe site, deoarece joburile din cozile de eliminare vor fi preemptate și oprite. Dacă alegem să rulăm un număr mai mare de joburi pe acest cluster, mai multe joburi vor fi eliminate deoarece vor fi un număr mai mare de joburi care vor fi preemptate. Figura 2.12 ilustrează modul în care joburile sunt oprite de joburi cu prioritate mai mare.

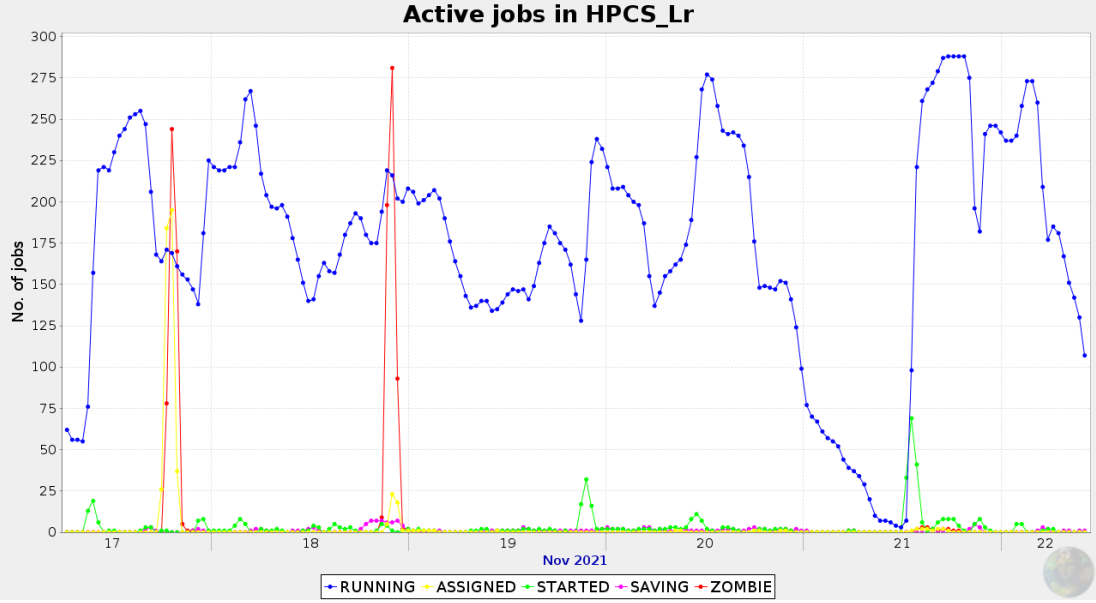


Figure 2.12: Distribuția joburilor pe sistemul Lawrencium

2.2.4 Concluzie

Am implementat un mecanism capabil să gestioneze sloturi multi-core și chiar noduri întregi alocate de site-urile gridului ALICE și să utilizeze resursele date pentru a lansa joburi single și multi-core din Grid cu o granularitate definită de utilizator. Această nouă caracteristică permite experimentului să ruleze noua generație de joburi, dar permite, de asemenea, implementarea software-ului pe site-uri noi, cum ar fi site-uri cu programare a nodurilor întregi și supercomputere. În plus, această teză propune o soluție pentru fragmentarea sloturilor pe noduri și pentru izolarea procesoarelor.

Cozile cu programarea noduri întregi au fost integrate în gridul ALICE, văzută folosind exemplul site-ului Lawrencium, găzduit la Laboratorul Național Lawrance Berkeley, care rulează ca o coadă de scavenging, folosind resurse în mod oportunist, fără ca experimentul să fie taxat pentru lor.

Alte întrebări de cercetare care decurg din această teză sunt următoarele:

- Care este impactul containerizării joburilor în ceea ce privește eficiența joburilor odată ce noul framework a fost implementat pe scară largă în Grid;
- Cum ar putea fi realizată automatizarea implementării Computing Element pe supercalculatoare;

Chapter 3

Adaptarea mediilor grid la mediile educaționale

Proiectul Open Education Hub este o inițiativă care propune un nou mod în care resursele clasei pot fi organizate în era digitală. Resurse educaționale deschise contribuie la îmbunătățirea calității materialelor didactice care sunt utilizate în acest nou context digital.

Educația a întârziat în implementarea progreselor pe care dezvoltarea de software le-a permis în alte domenii, cum ar fi automatizarea, găzduirea resurselor în cloud și aprovizionarea externă pentru contribuții.

Urmând principiile 5R pentru resursele educaționale deschise (open education) (**r**etain, **r**euse, **r**evise, **r**emix, **r**edistribute), am identificat un set de obiective pentru organizarea clasei într-o era digitală:

- Resursele trebuie să fie ușor accesibile online sau offline, astfel încât mai mulți utilizatori să poată avea acces la ele și să contribuie ulterior;
- Clasele trebuie să folosească cât mai multă automatizare; cadrele didactice sunt însărcinate cu îndrumarea elevilor, punându-i pe aceștia să facă sarcini administrative înfrânge acest scop;
- Resursele trebuie să fie ușor de contribuit, deoarece acest lucru permite părților interesate să contribuie fără bătăi de cap.

Accentul muncii noastre este pe crearea unor mecanisme de automatizare pentru a **reduce timpul petrecut în construirea și întreținerea materialelor**, permițând profesorilor să se concentreze asupra actului de predare. Intenționăm să **creștem disponibilitatea resurselor** prin crearea de resurse de găzduire online pentru profesori, folosind frameworkul nostru de generare Open Education. Folosind un flux de lucru centrat pe Git, facem **ușoară contribuția părților** prin crearea de solicitări de modificare prin Pull Request-uri sau ridicând probleme prin funcționalitățile de gestiunea proiectelor disponibile pe platformele bazate pe Git. Frameworkul a fost implementat pentru trei clase și alte clase și-au exprimat interesul de a migra la acesta.

Materialele de clasă de construcție sunt doar un aspect al pregătirii clasei. Evaluarea

elevilor este o altă fațetă a predării care necesită atenția profesorilor. În contextul educației în informatică, evaluarea elevilor se poate face prin chestionare, examene sau teme practice. Open Education Hub prezintă o modalitate prin care evaluarea testelor și examenelor poate fi, de asemenea, automatizată.

Deși există platforme pentru gestionarea și automatizarea chestionarelor și examenelor la clasă, problema notării temelor la scară nu este încă rezolvată. Instrumentele actuale permit o flexibilitate limitată în evaluare și nu oferă integrare cu software-ul existent de management al învățării care a fost adoptat ca parte a efortului de digitalizare a învățării.

Propunem verificatorul de teme VMchecker pentru a rezolva problema pierderii de timp în timpul evaluării sarcinilor. VMchecker a fost integrat cu succes în cadrul a 11 clase de la Universitatea Națională de Științe și Tehnologie POLITEHNICA BUCUREȘTI, acoperind cazuri de utilizare care nu ar putea folosi alte soluții software existente, cum ar fi virtualizarea și joburi multi-core. A crescut disponibilitatea verificării temelor pentru acasă, eliminând overheadul de configurare a verificării temelor prin utilizarea șabloanelor și a tehnicilor CI/CD pentru a implementa verificarea temelor.

Această teză aduce următoarele contribuții:

- un framework pentru automatizarea construcției și întreținerii materialelor de clasă;
- procese, instrumente și materiale care pot fi folosite de educatori pentru a-și crea materialele de clasă;
- o platformă de management al resurselor educaționale și un framework care construiește și implementează clase, permițând altora să contribuie la materiale;
- un instrument pentru automatizarea sarcinii de verificare și notare a temelor.

Secțiunea 3.1 prezintă obiectivele, arhitectura și implementarea platformei de construire a conținutului, numită oer-builder. Secțiunea 3.2 discută provocările automatizării verificării temelor și abordarea noastră de a rezolva această problemă prin implementarea unui nou utilitar numit vmchecker. Secțiunea 3.3 dezbate rezultatele obținute după implementare, clasele pe care le-am migrat și realizările utilitarului vmchecker în timpul anului școlar. Secțiunea 3.4 încheie teza evidențiind munca depusă și oportunitățile de creștere pentru proiectul de educație deschisă.

3.1 Open Education Resources Builder

Conținutul educațional a fost găzduit online încă de la începuturile internetului [34] sub formă de wiki-uri, tutoriale, postări pe blog și altele. Internetul a crescut, prin natura sa, accesul la resursele educaționale și a crescut cantitatea de informații care este proliferată. Ca parte a proiectului Open Education Hub, am început să creăm instrumente care facilitează construirea, distribuirea și utilizarea materialelor educaționale pe internet.

Problema cu resursele educaționale actuale este că, în multe cazuri, acestea nu sunt construite într-o manieră OER. Resursele nu pot fi remixate, sau reutilizate fără bătăi de cap, sau au licențe restrictive, mai ales dacă provin de la instituții de învățământ

superior, care le folosesc pentru a genera mai multe venituri. Ca parte a resurselor OER, ar trebui să fie ușor de contribuit și de remixat. Pentru a realiza acest lucru, resursele ar trebui să fie organizate într-un format ușor de schimbat în maximum.

3.1.1 Obiectivele Generatorului de Conținut Educațional

Open Education Hub dorește să folosească un framework prin care studenții și profesorii să poată construi și să contribuie la resursele educaționale. Utilizarea soluțiilor software existente pentru a permite contribuțiile OER este o prioritate, deoarece aceasta înseamnă mai puțin timp petrecut cu menținerea codului și adăugarea de noi funcții. O soluție existentă poate evolua fără a fi nevoie de schimbarea framework-ului.

Un generator de OER ar trebui să permită o configurare automată a resurselor necesare, pentru a petrece mai puțin timp gestionând framework-ul și mai mult timp creând conținut. Frameworkul ar trebui să aibă opțiuni pentru verificarea automată a diferitelor aspecte, cum ar fi greșelile de ortografie sau formatarea greșită.

Deși există generatori de documentație, aceștia se concentrează mai mult pe generarea de conținut, cum ar fi manuale și tutoriale. Acestea nu acoperă o gamă largă de materiale, cum ar fi chestionare, tutoriale, diapozitive și lucrări interactive. Acest lucru limitează cantitatea de resurse pe care o clasă poate include în ele dacă este folosită ca o experiență ieșită din cutie. Un generator de conținut trebuie să fie ușor de utilizat și configurat. Ar trebui luați un număr minim de pași pentru a începe crearea și implementarea conținutului. Conținutul care este implementat trebuie să fie ușor de copiat și reconstruit de către alte părți interesate. Textul trebuie să fie metoda principală de stocare a conținutului, deoarece poate fi editat cu ușurință fără a necesita instrumente speciale. Ar trebui folosit ori de câte ori este posibil, chiar și pentru diagrame, deoarece modificările pot fi urmărite cu ușurință folosind un instrument de versiune.

3.1.2 Arhitectura oer-builder

Proiectul oer-builder a fost construit ca un proiect modular, care poate fi flexibil în integrarea diferitelor tipuri de materiale într-o clasă. De asemenea, constructorul trebuia să permită integrarea diferitelor tipuri de cazuri de utilizare.

Git a fost ales ca soluție de gestionare a textului pentru materialele de clasă. Deoarece avem nevoie ca toate materialele de clasă să fie în format text, astfel încât să le putem edita cu aplicații de bază, utilizarea Git a fost o soluție viabilă.

Clasele trebuie să fie construite într-un format de text îmbogățit care poate fi convertit în conținut implementabil. Textul îmbogățit trebuie să accepte linkuri, formule, liste detaliate, enumerare, imagini, GIF-uri și multe altele. Am ales să creăm conținut în format Markdown (MD), deoarece este folosit de mulți producători de documentație existenți. Markdown este extensibil și textul poate fi convertit în el din alte formate, cum ar fi RST sau LaTeX, folosind instrumente deja existente, permițând o migrare mai ușoară de la alte forme de conținut.

Conținut Tutorial

Am ajuns la concluzia că produsul minim viabil pentru o clasă ar fi o clasă interactivă în care studenții ar urma tutoriale disponibile pe o pagină online.

Această cerință s-ar mapa direct cu utilizarea unui generator de documentație. Docusaurus a fost ales ca generator de documentație deoarece este complet și oferă pluginuri pentru funcționalitate suplimentară. Docusaurus acceptă formatul text Markdown, care acceptă utilizarea formatării bogate și a avertismentelor și poate include chiar și cadre iframe în pagină, permițându-ne să adăugăm funcții mai avansate paginilor.

Un utilizator trebuie să creeze un fișier de configurare în care trebuie să specifice căile către fișierele markdown pe care dorește să le includă și numele capitolului. Pot fi incluse și opțiuni avansate ale modulului, pentru a personaliza cursul.

Slide-uri

În mediul academic, slide-urile sunt stocate și afișate în mod regulat în format PDF sau PPT. Am ales să nu folosim acest format deoarece, în timp ce slide-urile conțin text, ele sunt stocate în format binar Slide-urile PDF și PPT fac din integrarea feedbackului un proces manual care nu poate fi urmărit și automatizat cu ușurință.

revela-md [14] este un instrument care încorporează fișiere markdown în conținut JavaScript pentru a fi vizualizat în interiorul unei ferestre de browser. Conținutul JavaScript poate fi integrat în alte forme de conținut sau poate fi vizualizat singur.

Quiz Fără Notare

oer-builder integrează suport pentru chestionare în materialul tutorial. Testele sunt o modalitate de a verifica studenții cu privire la cunoștințele acumulate în timpul tutorialelor. Un test este, de asemenea, un instrument pentru a-i menține implicați cu conținutul atunci când acesta se înclină spre a fi mai teoretic. Ele pot fi notate sau nu. Deoarece nu legăm oer-buildar de nicio infrastructură instituțională, nu există nicio modalitate de a autentifica studenții pentru a-și stoca notele.

A fost creat un șablon comun de chestionar pentru a reprezenta o întrebare și un set de răspunsuri, cu răspunsul corect marcat.

3.1.3 Implementarea Conținutului

Implementarea conținutului este acțiunea de a face conținutul creat local de creatori și colaboratori accesibil publicului. Acest lucru necesită o platformă unde conținutul poate fi găzduit De asemenea, conținutul trebuie să fie într-un format care să poată fi accesat folosind software-ul de bază.

Acțiunile de implementare permit implementarea automată a codului la modificări, nefiind nevoie de nicio acțiune din partea utilizatorului după ce codul este încărcat în Git. Atât GitHub, cât și GitLab oferă, de asemenea, intrări DNS, astfel încât paginile pot fi ușor accesibile dacă utilizatorul cunoaște numele proiectului și al repository-ului. oer-builder produce cod HTML, care poate fi vizualizat într-un browser de internet.

Deoarece codul este HTML, acesta poate fi găzduit ca conținut static pe multe tipuri de platforme.

Utilizarea GitHub ne permite să atingem obiectivul de automatizare a acțiunii de integrare a modificărilor. Acest lucru ne permite să executăm sarcini automate atunci când textul este modificat, cum ar fi verificarea ortografiei și formatării acestuia.

3.2 Verificator de Teme

Una dintre cele mai mari provocări ale predării este evaluarea muncii studenților. Pentru un profesor, acțiunea de notare a muncii este repetitivă și necesită atenție.

În informatică există o gamă largă de instrumente care fac evaluarea codului. Acesta este un proces care se întâmplă atât în context comercial, dar și în cercetare și educație. Evaluarea codului este utilizată pentru a verifica funcționalitatea și comportamentul așteptat al unei aplicații sau al unui proiect.

Temele în educația IT iau forma unor atribuiri de cod sau proiecte, în care studenții trebuie să implementeze un set de teme în interiorul unui șablon sau cadru. Sistemul de notare pentru aceste teme se bazează pe un scor pentru funcționalitatea codului și pe bune practici de scris cod, cum ar fi utilizarea anumitor modele de proiectare sau utilizarea eficientă a alocării memoriei.

Profesorii folosesc instrumente de evaluare a codurilor pentru a testa temele, proiectele sau lucrările de laborator ale studenților pe baza unui set de teste de referință scrise de profesori. Rezultatul instrumentului de evaluare ar trebui să fie un scor pentru funcționalitatea temei și feedback-ul de cod.

Peisajul actual al instrumentelor de evaluare a temelor este alcătuit din software care trebuie să fie licențiat de la companii sau care nu automatizează suficient munca. Instrumentele existente nu sunt integrate cu platformele populare de e-learning, cum ar fi Moodle. Acest lucru crește overheadul de integrare a unui instrument de evaluare în infrastructura instituțională.

VMchecker propune o soluție software gratuită și open source pentru evaluarea temelor pentru acasă, care utilizează integrări cu alte instrumente, cum ar fi Moodle pentru utilizatorul front-end și gestionarea duratei de viață a temelor și GitLab pentru rularea codului într-un mediu sigur.

3.2.1 Obiective

VMchecker este construit pentru a satisface obiectivele proiectului Open Education Hub. Acestea sunt aceleași obiective ca și cele stabilite pentru oer-builder, dar au fost particularizate pentru verificarea temelor după cum urmează.

VMchecker trebuie să fie ușor de utilizat și configurat de către managerii de infrastructură, profesori și studenți. Acest lucru se poate face prin automatizare, oferind scripturi de configurare pentru a permite configurarea automată. Containerele sunt folosite în dezvoltarea software-ului modern pentru a utiliza o soluție simplă pentru implementarea software-ului. Ne propunem să folosim containere pentru a permite administratorilor de infrastructură să pornească infrastructura fără a avea nevoie de un mediu special.

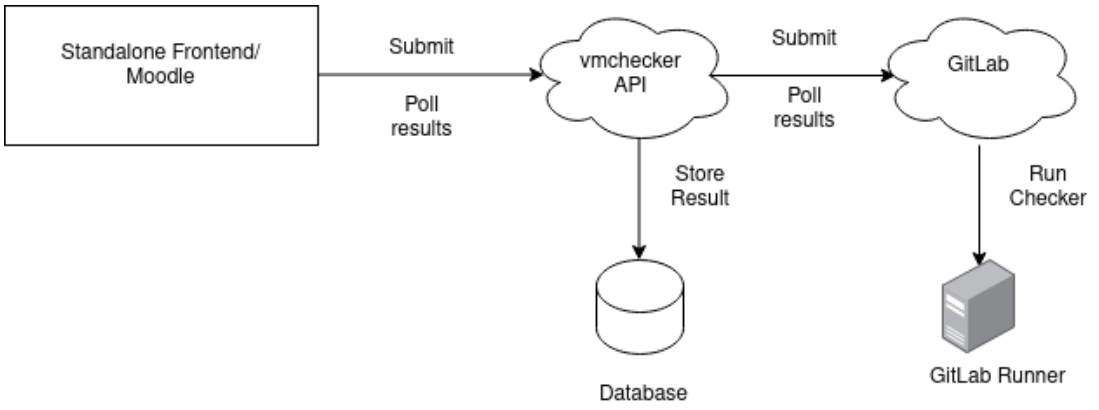


Figure 3.1: Arhitectura VMchecker

Scopul VMchecker este să fie o soluție software care necesită cât mai puțin cod scris. O soluție low-code este o soluție cu un cost întreținere redus, care necesită mai puține intervenții din partea programatorilor. O bază de cod mai mică lasă mai puțin spațiu pentru erori de adăugat de către dezvoltatorii de software. Dorim să folosim o soluție care se bazează pe alte stive de software, astfel încât să putem profita de suportul comunității oferit de aceste soluții.

VMchecker își propune să învețe studenții cum să folosească instrumente de versiune a codului, cum ar fi Git, astfel încât aceștia să poată gestiona un mediu de cod de producție în timp ce lucrează la proiecte.

Pe măsură ce utilizarea resurselor crește și și apar mai mulți doritori pentru integrarea cu VMchecker, acesta ar trebui să poată mări cantitatea de teme pe care le verifică în paralel. O abordare paralelă înseamnă că studenții nu trebuie să aștepte în cozi lungi pentru verificarea temelor.

Având în vedere că VMchecker ar trebui să fie o soluție gratuită și open source, ar trebui să fie construit pe soluții gratuite și open source. VMchecker trebuie să fie compatibil cu cerința OER 5R. Utilizarea soluțiilor open source ne permite să corectăm soluția pe care o folosim dacă întâmpinăm probleme.

Multe instituții de învățământ folosesc platforme de e-learning pentru a gestiona cursurile. Aceste instrumente permit utilizatorilor să încarce materiale, să testeze studenții și să noteze teme folosind o singură aplicație. Integrarea VMchecker cu platformele de e-learning face ca integrarea să fie mai atractivă pentru instituțiile care se bazează deja pe aceste sisteme. Instituțiile de învățământ au nevoie de un proces fără întreruperi de la trimiterea temelor până la notarea într-un caiet de note partajat.

3.2.2 Arhitectura

VMchecker a fost construit cu o arhitectură modulară, astfel încât să se potrivească nevoilor diverse ale instituțiilor de învățământ.

Figura 3.1 afișează componentele minime necesare pentru a avea un serviciu VMchecker funcțional.

- front-end - primește teme de la studenți, le încarcă în middleware-ul VMchecker și primește informațiile despre rularea temei după ce s-a terminat;
- middleware - primește cereri de la front-end și încarcă temele în infrastructura de verificare a temelor, care rulează tema, așteaptă rezultatul și apoi îl încarcă pe front-end;
- assignment runner - primește cereri de verificare a temelor de la serviciul middleware și rulează temele folosind o rețetă specifică definită de atribuire.

Front-end

Prima componentă necesară este un front-end care prezintă studenților o zonă de încărcare a temelor și oferă profesorilor un loc pentru a gestiona temele și pentru a descărca și încărca teme. Pentru primul exemplu implementat la UNSTPB, am folosit un plugin integrat în Moodle care se conectează la modulul de atribuire deja existent încorporat în Moodle. Folosind Moodle, am reușit să reducem cantitatea de cod necesară pentru gestionarea misiunilor, deoarece acest lucru este deja gestionat de Moodle.

Middleware

Middleware-ul este aplicația care gestionează cererile de la front-end. Oferă un API REST, astfel încât să poată fi utilizate mai multe front-end-uri, chiar și în paralel, deoarece acestea trebuie să creeze cereri HTTP pentru operațiunile de gestionare a atribuirii în VMchecker. Middleware-ul VMchecker gestionează durata de viață a verificării temelor.

Integrarea GitLab

Se poate rula cod în infrastructura GitLab CI/CD prin crearea unui commit care declanșează o conductă care rulează un script configurat. VMchecker trebuie configurat prin front-end cu un proiect și acces la instanța GitLab a proiectului. Acest lucru permite middleware-ului să creeze un commit prin intermediul middleware-ului care declanșează verificarea atribuirii. Middleware-ul verifică cea mai recentă conductă care rulează pentru acel comitere specific. Când conducta s-a terminat, preia rezultatul. Scriptul de verificare este configurat de managerul de joburi și va rula operațiunile necesare pentru verificarea temelor. Operațiunea de verificare poate include compilarea codului și linting și va afișa un rezultat pentru utilizatori. Middleware-ul stochează o listă de sarcini care sunt în proces de verificare.

GitLab Runner

Un pipeline GitLab CI/CD are nevoie de resurse fizice pentru a porni scriptul de verificare. Resursele CI/CD sunt gestionate de runneri GitLab, care pot fi disponibile gratuit pe platformă sau pot fi configurate pe depozit dacă sunt necesare condiții specifice.

3.2.3 Cazuri de utilizare VMchecker

VMchecker a fost creat având în vedere scenarii pentru trei utilizatori diferiți:

- administratorul de sistem instituțional care administrează VMchecker și trebuie să îl mențină și să asigure un acord privind nivelul de servicii, mai ales în situații de încărcare mare, cum ar fi termenele limită pentru teme;
- personalul didactic al cărui rol este acela de a crea o temă și de a nota temele studenților odată cu trecerea termenului limită;
- elevul care încarcă tema, verifică rezultatul și primește nota odată ce este notată de profesor.

Fiecare dintre acești utilizatori trebuie să interacționeze cu VMchecker printr-o interfață ușor de utilizat. Sarcina de a construi o interfață front-end pentru utilizatori necesită o cantitate mare de muncă de programare pentru a gestiona, a menține la zi și a implementa noi funcții. Acesta este motivul pentru care am ales să folosim un front-end care a fost deja construit, întreținut și utilizat la marile instituții de învățământ.

3.2.4 Infrastructura Containerelor

Problema cu utilizarea aceluiași script de verificare a temelor pentru mulți studenți este că aceștia pot avea diferite biblioteci și aplicații instalate pe sistemele lor. Medii diferite pot duce la anumite inconsecvențe de optimizare. Pentru a atenua acest lucru, recomandăm profesorilor să configureze infrastructura de verificare în VMchecker, astfel încât scripturile să ruleze în interiorul unui container. Echipa VMchecker oferă un șablon minim care poate fi extins în funcție de nevoile individuale ale clasei.

Infrastructura containerelor prezintă o abordare mai ușoară decât cea întâlnită în soluțiile actuale, care fie necesită o conexiune la internet pentru verificarea de la distanță, fie o mașină virtuală, care utilizează mai multe resurse. Utilizarea containerelor permite patch-uri mai ușoare pentru mediu, de exemplu instalarea unei noi biblioteci, deoarece modificările vor fi trimise progresiv, studentul nefiind nevoit să descarce o imagine complet nouă.

3.3 Rezultate

Organizația Open Education Hub pe GitHub a fost creată pentru a automatiza funcționarea diferitelor componente care iau parte la managementul clasei și la crearea clasei. A fost creată o metodologie pentru a ghida profesorii în construirea și întreținerea orelor de OER.

3.3.1 oer-builder

Organizația GitHub a fost creată pentru a menține clasele care au fost construite folosind metodologia și a căror întreținere este trecută la Open Education Project. Acest lucru eliberează creatorii de conținut de sarcina gestionării Pull Request-urilor și a problemelor cu materialele.

Clasele pot fi forkuite și remixate liber de către alte proiecte sau instituții; automatizarea a fost pusă în aplicare pentru a permite claselor să aibă implementare automată și linting folosind API-ul GitHub.

Trei clase au fost integrate în organizația GitHub Open Education Hub. Aceste cursuri au fost rescrise pentru a profita de oer-builder. Acestea includ chestionare, slide-uri, figuri, tutoriale și teme.

Clasa Sisteme de Operare

Clasa de Sisteme de Operare a fost prima integrată. Stide-urile se bazează pe clasa care se predă la Universitatea Națională de Știință și Tehnologie POLITEHINCA București (UNSTPB). Aceasta a fost migrată dintr-un format bazat pe wiki unde conținutul a fost găzduit pe servere instituționale. În timp ce wiki-ul a fost ușor de accesat pentru studenți, includearea feedback-ul studenților a fost dificil fără a oferi studenților acces la întregul spațiu de lucru al clasei, care includea documentație internă pentru echipa de sisteme de operare.

Clasa este axată pe activitățile didactice efectuate de studenți ca parte a laboratoarelor. În consecință, cea mai mare parte a depozitului este ocupată de conținut scris și de teme.

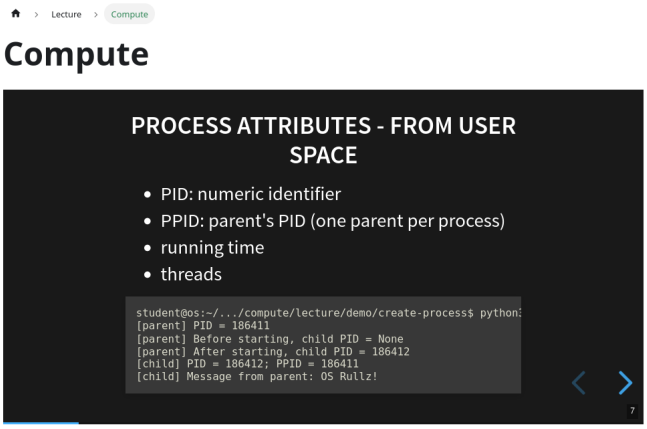


Figure 3.2: Slide-urile materiei Sisteme de Operare

Figura 3.2 afișează diapozitive create folosind generatorul integrat în pagina web a clasei de sisteme de operare.

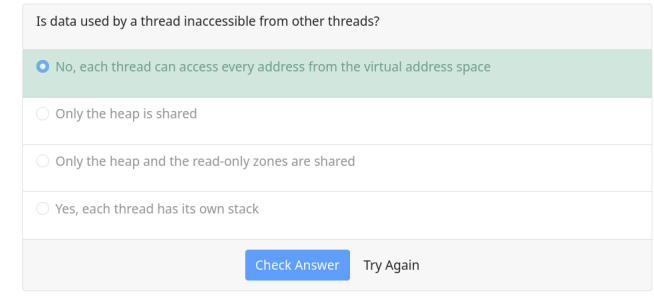


Figure 3.3: oer-builder quiz răspuns corect

Formatul de chestionar pe care l-am implementat este afișat în figurile 3.3 și 3.4. Putem vedea că afișează răspunsul corect și o explicație pentru ce este corect.

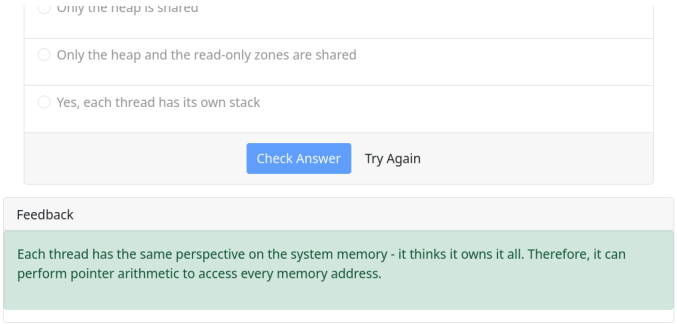


Figure 3.4: Feedback pentru quiz în oer-builder

Clasa a trecut prin două semestre de predare bazate pe noile materiale. Ideea de a oferi feedback și de a ridica probleme în GitHub a fost promovată în rândul studenților, aceștia fiind recompensați cu un pin fizic de Sisteme de Operare pentru serviciul comunității.

Studenții au deschis 32 de cereri de tragere care au fost integrate în materialele de clasă.

Noul proces de construire a conținutului a permis echipei să crească numărul de oameni care pot lucra în paralel la conținut. Folosind GitHub, fiecare capitol nou de conținut a avut un Pull Request separat. Acest lucru a ajutat la organizarea procesului de feedback, deoarece modificările puteau fi discutate pe pagină, iar sugestiile au fost făcute in-line. Conținutul a fost verificat automat pentru erori de format și ortografie, permițând utilizatorilor să se concentreze asupra calității conținutului. La integrarea noului conținut, acesta este implementat automat, eliminând nevoia ca un administrator de sistem să facă acest lucru și să coordoneze diferitele integrări.

CCAS

Clasa CCAS este o clasă de statistică predată la Universitatea din Islanda. Acesta este un curs axat pe lucrări de laborator bazate pe o clasă scrisă pentru limbajul de programare R.

Clasa a fost migrată dintr-o carte de laborator care a fost construită în LaTeX și a fost distribuită ca document PDF. Această abordare nu este ușor de implementat, deoarece este nevoie de un spațiu comun cu studenții sau de un site web public. Nu este ușor de căutat și nu este ușor să contribui, deoarece codul sursă nu este partajat. O altă problemă este că LaTeX necesită mai multă expertiză decât un format de text Markdown pentru a scrie, iar pachetele necesare pentru a-l utiliza sunt mari și nu sunt omniprezente pe toate platformele.

Un script bazat pe utilitarul `pandoc` a fost folosit pentru a migra conținutul LaTeX la Markdown. Deoarece clasele erau deja într-un format text, stocate central, procesul de conversie a acestora într-un alt format a fost mai ușor decât a avea resursele dispersate în mai multe clase sau mai multe PDF-uri.

Faptul că am putut folosi scripturi pentru a converti automat conținutul demonstrează că alte clase și-ar putea converti conținutul pentru a lucra cu oer-builder și să profite de caracteristicile acestuia.

Security Summer School

Școala de vară de securitate, Security Summer School (SSS), este un atelier axat pe predarea principiilor de securitate. Este predat online, cu resurse găzduite online. Resursele au fost deja construite folosind Docsy [4], care este un generator de documentație care convertește fișierele MD în HTML. SSS utiliza deja GitHub Actions pentru a automatiza implementarea resurselor și verificarea Pull Request-urilor.

Deși configurarea a fost făcută urmând principiile OER, a fost dificil de replicat de către alte instituții și de a face modificări la nivel local, deoarece folosea fișiere de configurare specifice Docsy. Un utilizator ar trebui să învețe parametrii de configurare Docsy pentru a afla cum să contribuie la proiect. Acest lucru a reprezentat un obstacol în încurajarea altor utilizatori și instituții să contribuie cu conținut.

SSS a fost migrat la Open Education Hub. A fost configurat să ruleze oer-builder și să se integreze cu Docusaurus. Migrarea a fost ușor de făcut, deoarece a implicat doar mutarea fișierelor Markdown într-un nou depozit care a fost configurat folosind generatorul.

3.3.2 VMchecker

VMchecker a fost introdus în mediul de producție la UNSTPB din septembrie 2022. Middleware-ul este conectat la instanța UNSTPB Moodle prin pluginul Moodle VMchecker. Middleware-ul a fost configurat să se conecteze la instanța GitLab implementată la UPB pentru că am dorit să profităm de spațiul mare de stocare oferit de instituție și de integrarea cu ID-uri instituționale. Integrarea permite studenților să-și dea permisiuni asistenților pentru a primi recenzii și ajutor. VMchecker a fost folosit de unsprezece clase diferite, fiecare cu structuri și cerințe diferite de atribuire. Următoarele sunt cerințe demne de remarcat și observații făcute din anumite clase:

- Clasa Internele Sistemelor de Operare a necesitat pornirea mașinilor virtuale pentru verificarea codului, sarcină care a fost acceptată.
- Clasa Algoritmi Paraleli și Distribuiți a necesitat suport multi-core și izolarea resurselor, astfel încât timpul de rulare pentru diferiți utilizatori va fi calculat în același mod pentru toți utilizatorii, deoarece temele au fost evaluate pe baza timpului de rezolvare.
- Temele materiei Arhitectura Sistemelor de Calcul sunt dependente de GPU, acest lucru necesită un GitLab Runner specific care are acces la clusterul UNSTPB, unde temele pot fi trimise la gridul instituțional; soluția la această cerință a crescut dimensiunea potențială a infrastructurii de verificare VMchecker de la un runner care rulează pe o mașină virtuală din interiorul clusterului până la întreaga infrastructură grid instituțională, măsurând la 600 de nuclee și mai mult de 2,5 TB de memorie disponibilă.
- Clasa Protocoale de comunicare a necesitat utilizarea capabilităților de rețea, care nu au fost activate implicit în interiorul containerelor.

Peste 40.000 de teme au fost verificate pe parcursul anului școlar. Atribuțiile au folosit infrastructura containerului pentru a rula același mediu de atribuire

3.4 Concluzii și Direcții Viitoare

Proiectul Open Education Hub a dezvoltat și implementat instrumente care îi ajută pe profesori să reducă sarcina administrativă impusă profesorilor. Am făcut acest lucru creând proceduri de lucru, șabloane și framework-uri care automatizează sarcinile de integrare și implementare de noi materiale și verificarea sarcinilor. Aceste acțiuni au fost realizate în același timp cu creșterea disponibilității resurselor educaționale prin postarea tuturor materialelor online într-o manieră open source, permițând terților să consume conținutul și să-l refolosească.

Am construit un instrument Open Source care îmbunătățește procesul de creare a materialelor și de publicare a acestora. Creând instrumentul, am migrat trei clase la standardul OER, îmbunătățind procesul de menținere a claselor.

Sarcina de verificare automată a temelor a fost îmbunătățită la UNSTPB prin integrarea VMchecker. Un număr mare de clase au preluat deja noul instrument de verificare a temelor, deoarece se extinde mai bine decât alte instrumente și necesită mai puțini pași de configurare. Folosind infrastructura instituțională existentă, am reușit să creștem performanța verificării temelor, facilitând în același timp procesul de notare, creare și gestionare a temelor.

Procesul de conversie a început deja pentru alte clase de la UNSTPB. Vizăm câștiguri ușoare, clase care sunt deja bazate pe text, care pot fi convertite mai ușor decât să le scriem de la zero.

Ne propunem să creștem implicarea studenților și a terților cu resursele publice disponibile în Open Education Hub. Pentru a realiza acest lucru, dorim să implementăm recompense digitale folosind Smiley Coins, o criptomonedă educațională. Integrarea acestuia într-un mediu de producție va fi o prioritate, deoarece am implementat deja o Proof of Concept pentru clasa Sisteme de operare.

Chapter 4

Clustere cu Disponibilitare Înaltă pentru Calculul Grid

Arhitecturile de cluster au devenit o cârje pe care instituțiile și companiile se sprijină pentru a oferi utilizatorilor și clienților servicii care altfel nu ar fi fost disponibile. Servicii precum Single Sign On, web hosting, spații de stocare partajate, platforme de învățare, toate au nevoie de o infrastructură de suport care să le gestioneze durata de viață, să aloce resurse hardware și să ofere garanții de securitate.

Mașinile virtuale pot fi folosite ca o astfel de infrastructură de suport, deoarece oferă un mediu izolat, care rulează propriul sistem de operare independent. Utilizatorii pot profita de permisiunile din mediile cluster care altfel nu ar fi acordate, cum ar fi accesul ca utilizatori privilegiați sau privilegiile de configurare a rețelei, fără a pune în pericol infrastructura cluster-ului de bază.

Deoarece mediile virtualizate sunt acum folosite pentru a găzdui servicii critice, acestea trebuie să fie întotdeauna disponibile pentru a-și servi utilizatorii. Clusterele trebuie proiectate luând în considerare toleranța la erori, pentru a maximiza timpul de funcționare al serverului și al mașinii virtuale. Trebuie implementate măsuri astfel încât starea cluster-ului să fie monitorizată în mod continuu, astfel încât defecțiunile să poată fi prevenite înainte de apariția lor, pentru a nu duce la oprirea serviciului. În cazul în care apare o astfel de defecțiune, aceasta trebuie detectată și remediată cât mai curând posibil. Sistemele de monitorizare îndeplinesc această nevoie, permițând ca punctele de date să fie colectate din sistemele gazdă și să fie utilizate pentru a declanșa alerte cu privire la evenimente precum defecțiunile nodurilor, degradarea discului sau blocarea rețelei.

Următorul capitol investighează metodele de implementare pentru clustere de înaltă disponibilitate, concentrându-se pe utilizarea sistemului OpenStack, testându-le în clusterul UNSTPB. A fost implementat un mecanism de implementare cu disponibilitate ridicată, permițând timpi de răspuns rapid în cazul defecțiunilor serverului gazdă. Sistemele de monitorizare au fost proiectate și implementate în clusterul UNSTPB, adunând date de la servicii și hardware pentru a gestiona starea aplicațiilor care rulează.

4.1 Crearea Medii cu Disponibilitate Înaltă

Pe măsură ce nevoia de resurse de calcul crește, cresc și cheltuielile generale pentru gestionarea acestor resurse. În vremurile noastre, resursele de calcul pe care le folosesc majoritatea oamenilor nu sunt găzduite pe propriile mașini, laptopuri, telefoane sau stații de lucru; acestea sunt găzduite în infrastructuri cloud, unde pot fi gestionate de administratorii de sistem folosind software specializat pentru a gestiona resursele. Avantajul de a face acest lucru este că vor fi mutate toate generarea de căldură, consumul de energie și costurile de întreținere departe de utilizatori și către managerii centrelor de date, beneficiind în același timp de economii de scară atunci când vine vorba de achiziționarea de resurse, consumul de energie și eficiența răcirii.

Majoritatea oamenilor folosesc cloudul - de la oameni obișnuiți care urmăresc filme până la programatori, cercetători și personal non-tehnic care își descarcă sarcinile de lucru de pe computerele locale la sistemele accesate de la distanță.

Instituțiile pot alege fie să cumpere resurse de la furnizori mari de infrastructură sau servicii, cum ar fi Amazon Web Service, Google Cloud Platform, Oracle Cloud, Digital Ocean și alții, fie pot cumpăra resurse și le pot implementa într-un cluster local. Avantajul rulării serviciilor într-o platformă cloud deja implementată este faptul că, deși pe termen lung ar putea costa mai mult instituția, costul inițial al implementării unui serviciu va fi mai mic, deoarece nu va trebui să cumpărați hardware-ul pentru-l rula, iar prețul va fi scalat dinamic, pe baza utilizării resurselor [27]. Principalul dezavantaj este că pentru instituțiile la scară mai mare, cloud-urile private pot deveni mai economice, deoarece diferența dintre cloud-urile private și cloud-urile publice poate duce la echivalentul cu mai multe salarii de angajare cu normă întreagă.

Cazul nostru de utilizare ar fi o instituție de cercetare mare, cu peste 100 de noduri fizice, care se așteaptă să ruleze mai mult de 1000 de mașini virtuale la scară medie sau mare. Pentru acest tip de caz de utilizare, este mai rentabil să implementați o soluție de cloud privat.

Deoarece OpenStack [32] este cel mai mare manager de cloud privat și unul dintre cele mai mari proiecte open-source, am ales să-l implementăm pe nodurile găzduite la Universitatea Politehnica din București. Deoarece OpenStack este o aplicație atât de mare, a fost dezvoltată pentru a avea multe componente care gestionează diferite subsisteme, cum ar fi serviciul de gestionare a imaginilor, serviciul de autentificare, serviciul de gestionare a rețelei și multe altele.

Intenția noastră pentru cloudul OpenStack este să găzduim atât mașini virtuale pentru studenți, care sunt utilizate pentru rularea diferitelor sarcini de lucru de testare și laboratoare, dar și pentru găzduirea de servicii esențiale, cum ar fi verificatoare de teme, baze de date, platforme de autentificare și altele. Din acest motiv, avem nevoie de un mecanism care să garanteze o disponibilitate ridicată fără a crește utilizarea resurselor nodurilor.

Această teză își propune să exploreze modalitățile prin care un operator poate implementa un cluster OpenStack la scară mare și care sunt opțiunile disponibile în care se poate crește disponibilitatea resurselor în cazul defecțiunii nodului sau a procesului fără a crește costurile hardware ale clusterului.

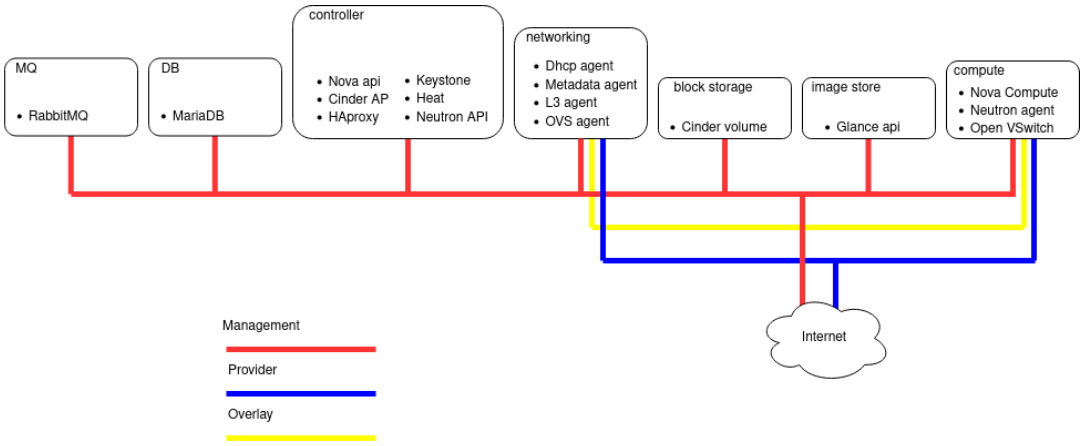


Figure 4.1: Arhitectura nodurilor OpenStack

4.1.1 Infrastructura Clusterului OpenStack

Componentele OpenStack Necesare Pentru Rularea Unui Cluster

Deoarece OpenStack este un sistem mare și modular, un administrator de site poate alege ce servicii sunt necesare pentru a rula un cluster care este personalizat pentru particularitățile sarcinilor de lucru.

Infrastructura clusterului de la UPB are un volum de lucru mixt: echipele de cercetare rulează diferite servicii pentru scopurile lor orientate spre exterior, cum ar fi diseminarea proiectelor, accesul utilizatorilor externi și altele, și sarcini de lucru de calcul, cum ar fi învățarea automată, procesarea imaginilor și fizica. Al doilea tip de utilizare este pentru studenții care își desfășoară o parte din temele și lucrările de laborator pe infrastructura cloud, ceea ce pune o sarcină pe planul de control deoarece construiesc și șterg multe mașini.

Serviciile pe care am ales să le implementăm la UPB sunt reprezentate în Figura 4.1.

În această subsecțiune vom arunca o privire asupra tuturor serviciilor implementate și vom explica utilizarea lor în interiorul clusterului, precum și modificările pe care le-am făcut pentru a rula.

Keystone Keystone [10] este serviciul de gestionare a identității, care se ocupă de autentificarea și autorizarea utilizatorilor și serviciilor. În mod implicit, stochează toți utilizatorii într-o bază de date. Deoarece facem parte dintr-o instituție mare, clusterul trebuie să accepte autentificarea folosind ID-uri instituționale. Pentru a realiza acest lucru, am configurat Keystone să se conecteze la instanța LDAP în care sunt stocate datele utilizatorului. Keystone este configurat pe nodul controlerului.

Baza de Date și Cozi de Mesaje Baza de date MariaDB[5] și instanțele RabbitMQ[36] au fost configurate de Kolla-ansible pe noduri separate, deoarece sunt servicii critice care trebuie să poată comunica, chiar dacă planul de control este offline, deoarece calculul și serviciile de rețea comunică, de asemenea, prin aceste noduri.

Serviciul de Imagini Serviciul de imagini Glance [8] stochează imaginile de bază pentru mașini, de pe care acestea vor fi copiate pe hipervizoare. Acest serviciu este configurat cu setările implicite pe un nod separat de celelalte servicii din planul de control, deoarece trebuie conectat la un dispozitiv de stocare mare, deoarece va stoca multe fișiere mari și are nevoie de o conexiune de stocare bună.

Nova Serviciul OpenStack Nova [12] este componenta de bază care gestionează programarea resurselor mașinii virtuale, managementul de viață.

Acest serviciu are multe componente care sunt gestionate de Kolla-ansible, cum ar fi:

- Nova API, care primește solicitări de la utilizator despre operațiunile VM, cum ar fi pornirea, oprirea, crearea, ștergerea și altele;
- Nova Conductor, implementat pe nodul controller, care gestionează comunicarea cu sistemele care găzduiesc VM-uri;
- Nova Scheduler, care alege sistemele gazdă în care vor rula VM-urile;
- Placement, care este un serviciu separat, dar care este strâns integrat cu Nova, deoarece gestionează inventarul de resurse, știind ce sisteme au ce resurse disponibile;
- Nova compute, care este un demon instalat pe mașinile de calcul care pornește efectiv mașinile virtuale, pregătește mediul, descarcă fișierele imagine și se asigură că mașinile virtuale rulează corect.

Neutron Neutron [11] este serviciul OpenStack care gestionează rețeaua. OpenStack oferă suport pentru rularea mașinilor virtuale pe rețele deja existente sau poate oferi rețele definite prin software (SDN) care pot fi create după bunul plac de către utilizatori. Acest avantaj ne permite să alocăm în mod dinamic IP-uri publice și, de asemenea, să izolăm comunicarea dintre mașinile virtuale critice și cele obișnuite prin crearea de rețele virtuale care rulează prin rețeaua suprapusă, așa cum se vede în figura 4.1.

Cinder Cinder [7] este soluția de stocare bloc folosită de OpenStack. Acest serviciu permite unei mașini virtuale să utilizeze un sistem de stocare la distanță, conectat prin protocoale de stocare la distanță. Avantajul eliminării stocării VM din nod este faptul că dacă există o defecțiune a nodului datele nu se vor pierde, deoarece stocarea este pe alt mediu. Volumele de stocare sunt utilizate în principal pentru servicii critice, pe care nu și le permit timpul de nefuncţionale. Nu putem folosi Cinder pentru toate mașinile virtuale, pentru că va fi să fie o scurgere prea mare a resurselor.

Heat Serviciul Heat [9] este serviciul de orchestrare OpenStack. Acesta în sine nu este un serviciu de bază, dar este utilizat pe scară largă deoarece poate ușura munca utilizatorului atunci când creează în mod repetat mașini virtuale similare. Este nevoie de o specificație de resursă ca intrare și generează mașinile virtuale solicitate, API-ul de solicitare fiind același cu popularul API AWS CloudFormation [2].

4.1.2 Automatizarea Instalării OpenStack

Întrucât întreaga infrastructură implică un număr mare de noduri care trebuie configurate, dorim să avem o soluție în care administratorii de sistem depun cât mai puțin efort posibil în integrarea mașinilor în sistem. Pentru stiva de software necesară pentru a rula OpenStack, trebuie să ne asigurăm că aceleași versiuni de aplicații sunt instalate pe toate sistemele și dorim să avem fișierele de configurare potrivite pentru rolul tuturor stațiilor din sistem. Pentru a ne asigura că îndeplinim toate aceste condiții, ne dorim să automatizăm cât mai mult sistemul de instalare și configurare a aplicației, asigurând astfel omogenitatea soluțiilor software și configurarea sistemelor cu cât mai puțină intervenție din partea administratorilor.

Automatizarea Implementării Sistemului de Bază

Primul pas în pregătirea unui sistem pentru a rula pe infrastructura OpenStack este instalarea sistemului de operare și a aplicațiilor de bază de care are nevoie OpenStack. Pentru a ne asigura că toate sistemele folosesc aceleași aplicații, folosim soluția de automatizare a instalării furnizată de sistemul de operare CentOS, deoarece oferă o interfață de scripting ușoară prin care putem specifica ce aplicații să instalați. Un alt avantaj al soluției oferite de CentOS, numită Kickstart, este că, pe lângă instalarea aplicațiilor, putem configura partiționarea discului și setăm opțiuni pentru aplicațiile instalate.

Deoarece dorim să automatizăm procesul de instalare a sistemelor de operare cât mai mult posibil, am ales să folosim sistemul de boot PXE pentru instalarea de la distanță a sistemului de operare. PXE boot este un mecanism care instruiește un sistem să pornească un sistem de operare de bază prin rețea. Odată ce sistemul de operare este încărcat în rețea, acesta va efectua un set de operațiuni. În cazul CentOS, serverul va descărca scriptul Kickstart descris anterior și va interpreta fiecare linie din acesta. Odată ce rulează scriptul, sistemul va avea instalat un nou sistem de operare cu toate aplicațiile necesare pentru instalarea și rularea ulterioară a sistemului OpenStack.

Automatizarea Instalării Serviciilor OpenStack Folosind Kolla Ansible

În funcție de serviciul OpenStack în cauză, există o listă de aplicații de care depinde, de la aplicația OpenVSwitch pentru serviciul de gestionare a rețelei, până la biblioteca libvirt pentru serviciul de gestionare a mașinilor virtuale de pe sistemul gazdă. Pe lângă dependențe, fiecare serviciu are nevoie de un fișier de configurare specific serviciului. Astfel, devine o povară pentru administrator să configureze manual fiecare serviciu OpenStack pe fiecare sistem gazdă și să se asigure că toate aplicațiile rămân în aceeași stare operațională.

Din acest motiv, am ales să folosim soluția de automatizare a instalațiilor de service bazată pe Kolla Ansible. Kolla Ansible este un set de rețete scrise în formatul specific suitei software Ansible, care ne permite să instalăm și să configurăm întreaga instalare OpenStack folosind doar două fișiere de configurare majore: un fișier de configurare în care specificăm opțiunile de sistem, `/etc /kolla/globals.yaml` și un fișier de configurare în care configuram asocierea dintre numele serviciilor și nodurile pe care trebuie instalate. `/etc/kolla/multinode`.

Avantajul utilizării Kolla Ansible este că folosește containere Docker pentru a se asigura continuu că aplicațiile necesare pentru a rula serviciul rulează aceeași versiune de software cu aceleași fișiere de configurare încărcate pe toate nodurile infrastructurii. Fiecare serviciu rulează într-un container Docker, astfel fiecare serviciu este izolat de celelalte, asigurând securitatea tuturor serviciilor, în cazul în care unul dintre ele este atacat.

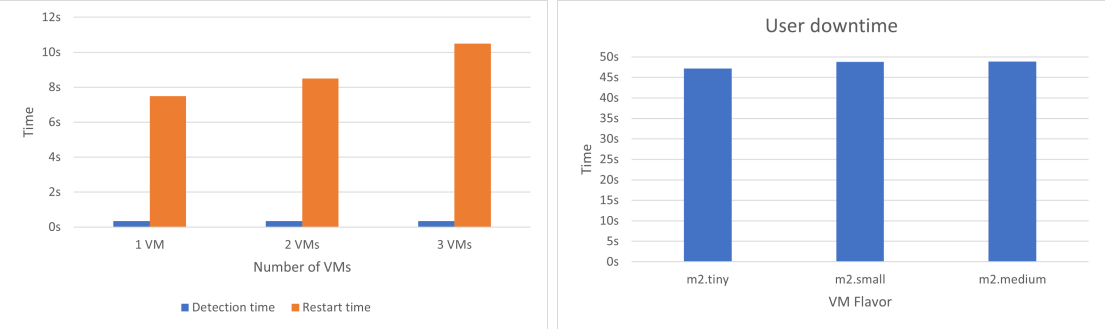
Automatizarea instalării sistemului de autentificare Pentru a instala sistemul de autentificare a fost necesară modificarea fișierelor sursă ale serviciului Kolla Ansible, deoarece nu permitea configurarea numelor de utilizator predefinite, în special pentru utilizatorii admin, care gestionează fiecare serviciu individual. Trebuie să facem acest lucru deoarece utilizatorii admin sunt predefiniti în infrastructura LDAP deja existentă a universității.

Automatizarea Instalării Serviciului Nova Deoarece fiecare server fizic pe care rulăm mașinile virtuale are o configurație diferită, în funcție de modelul de server folosit, placa de rețea folosită și alte aspecte legate de hardware pe care nu le putem controla, a fost necesară modificarea fișierului de configurare în care am definit asocieri între roluri și noduri. Pentru a ușura configurarea nodurilor și a reduce complexitatea fișierului de configurare, am atribuit fiecărei familii de noduri un nume, apoi am mapat familia de noduri la tipul de serviciu care va rula pe nod.

4.1.3 Rularea Mașinilor Virtuale de Înaltă Disponibilitate

Disponibilitatea ridicată nu numai că ajută la satisfacția utilizatorilor prin asigurarea unor timpi de funcționare mai mari fără a fi nevoie de intervenția umană, dar oferă o plasă de siguranță pentru serviciile importante sau critice implementate în interiorul OpenStack. Disponibilitatea ridicată a resurselor în OpenStack poate fi obținută folosind serviciul Masakari [6]. Masakari oferă un mecanism de recuperare pentru instanțele VM în cazul defecțiunilor la mai multe niveluri în cadrul implementării. Arhitectura sa folosește monitoare independente pentru procese, instanțe și gazde, generând notificări pentru ca motorul Masakari să proceseze și să declanșeze o secvență de recuperare. Deoarece implementarea noastră se bazează pe instrumentul Kolla Ansible, care implementează automat serviciile OpenStack în containere, funcția de monitorizare a procesului a Masakari nu va fi inclusă din cauza limitărilor impuse de separarea spațiului de nume al procesului în mediile containerizate. Monitorul de proces poate fi implementat manual pe gazdă și configurat pentru a monitoriza procesele care lansează containerele în execuție, repornind containerele menționate dacă este necesar. Deoarece acest lucru complică procesul de implementare, alegem să ne concentrăm doar pe monitorizarea instanțelor și a gazdei în timpul testării noastre.

Pentru a minimiza nevoia de intervenție rapidă asupra infrastructurii în cazul unei probleme cu funcționarea mașinilor virtuale sau a nodurilor de calcul pe care rulează, putem folosi o soluție de înaltă disponibilitate. OpenStack oferă serviciul Masakari pentru monitorizare și recuperare în cazul defecțiunilor suferite de mașinile virtuale, nodurile de calcul sau serviciile (procesele) necesare pentru o funcționare corectă. Acest serviciu oferă o disponibilitate ridicată prin repornirea resurselor OpenStack (mașini virtuale, procese) atunci când este detectată o eroare.



(a) Timp de detectare și repornire pentru un număr variabil de VM

(b) Timp de nefuncţionare a utilizatorului pentru diferite resurse VM

Figure 4.2: Performanța monitorizării instantelor

Monitorizarea mașinii virtuale urmărește procesele hipervizorului începute pe nodurile de calcul (în cazul nostru `qemu`); în cazul încetării bruște a unui astfel de proces, mașina virtuală este reluată în execuție folosind comanda inițială. Monitorizarea nodurilor de calcul utilizează software deja existent (Corosync și Pacemaker [41]); atunci când este detectată o oprire a unui nod de calcul, se declanșează procesul de evacuare: mașinile virtuale de pe acel nod vor fi lansate din nou în execuție pe nodurile disponibile. În ambele cazuri este posibil să se specifice, prin utilizarea metadatelor, repornirea sau evacuarea unui subset de mașini virtuale.

4.1.4 Performanța Recuperării Mașinii Virtuale

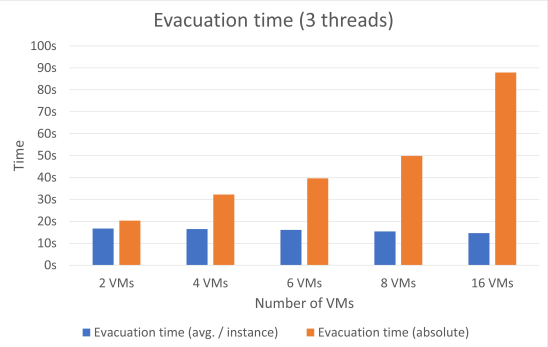
Testele au fost împărțite în diferite scenarii, parametri variați, cum ar fi flavorul mașinii virtuale, dimensiunea discului, sistemul de operare și numărul de VM. Flavorurile folosite în testele noastre pot fi văzute în Tabelul 4.1. Rezultatele și un rezumat al fiecărui scenariu vor fi prezentate în paragrafele următoare.

Flavor și vCPU și RAM	
m1.small	1 1 GB
m2.tiny	1 512 MB
m2.small	1 2 GB
m2.mediu și 2 și 4 GB	

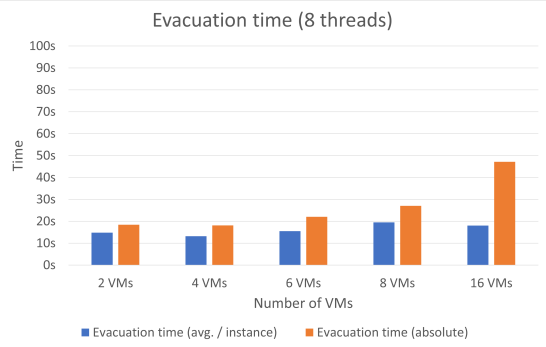
Table 4.1: Flavoruri virtuale

Monitorizarea Instanțelor Pentru monitorizarea instanțelor am măsurat mai întâi timpul de detectare și timpul de repornire, variind numărul de mașini virtuale din proces. Am implementat până la trei mașini virtuale Ubuntu Server 22.04, m2.tiny cu un disc de 10 GB.

Rezultatele sunt prezentate în Figura 4.2a. Timpul de detectare rămâne constant la 350 ms, timpul total pentru repornirea tuturor VM-urilor crescând liniar odată cu numărul de VM.



(a) 3 fire de evacuare



(b) 8 fire de evacuare

Figure 4.3: Timp de evacuare pentru un număr variabil de mașini virtuale și fire de evacuare

Al doilea test a fost făcut cu intenția de a măsura timpul de nefuncționare al utilizatorului. Pentru aceasta, am implementat un singur Ubuntu Server 22.04 VM cu un disc de 10 GB, variind flavor mașinii virtuale de la m2.tiny la m2.medium. Din figura 4.2b putem observa că resursele unei mașini virtuale nu influențează timpul de nefuncționare din perspectiva utilizatorului. Acest lucru se datorează parțial serverului nostru web care deservește conținut static; o aplicație mai complexă ar putea beneficia de mai multe resurse.

Monitorizarea Gazdei Pentru monitorizarea gazdei, am măsurat timpul de evacuare, atât absolut, cât și mediu pe caz, și timpul de nefuncționare al utilizatorului. În cazul monitorizării gazdei, Masakari permite utilizatorului să configureze numărul de fire utilizate pentru procesul de evacuare. Acest prim scenariu variază numărul de VM-uri de la 2 la 16, în trepte, cu 3 (implicit) și 8 fire de evacuare. Controlerul care procesa aceste notificări avea 16 nuclee CPU. Mașinile virtuale erau Ubuntu Server 22.04 cu un disc de 10 GB și flavor m1.small.

Din Figura 4.3a putem observa că timpul de evacuare absolut este similar cu timpul mediu de evacuare atunci când numărul de VM-uri este mai mic sau egal cu numărul de fire, crescând brusc la trecerea acestui prag. Figura 4.3b arată rezultatele pentru 8 fire. Deoarece numărul de fire provoacă o sarcină mai mare pe nodul controlerului, acest lucru crește timpul mediu de evacuare, dar timpul de evacuare absolut este în general mai bun decât cu mai puține fire, urmând tendința văzută anterior.

Timpul de nefuncţionare al utilizatorului nu este influenţat în mod absolut (redus sau mărit). În schimb, cu mai multe fire de evacuare decât nuclee CPU, timpul de nefuncționare al utilizatorului începe să varieze mai mult decât înainte, așa cum se vede în Figura 4.4.

Variațiile precum alegerea sistemului de operare și flavor de VM nu au avut un impact perceptibil asupra timpului de evacuare sau a timpului de nefuncționare general al utilizatorului.

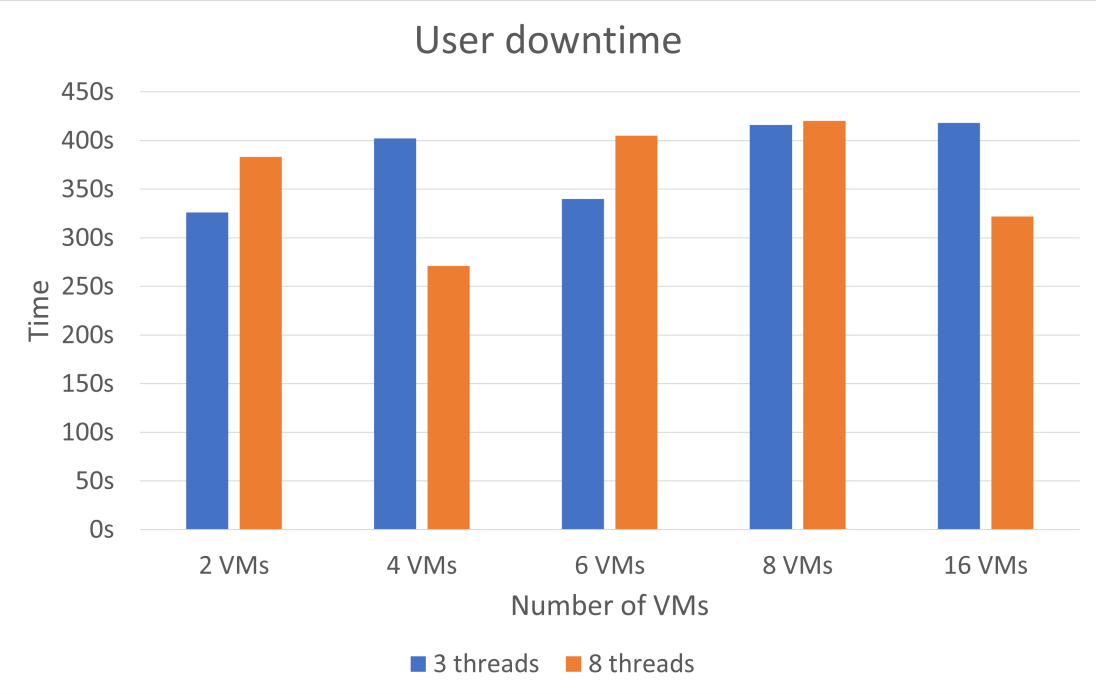


Figure 4.4: Timpul de nefuncționare al utilizatorului pentru un număr variabil de mașini virtuale și fire de evacuare

4.1.5 Concluzii și Direcții Biitoare

În subsecțiunile anterioare am analizat modalitățile prin care o instituție poate implementa o infrastructură cloud privată pentru a economisi costuri prin a nu rula mașinilor virtuale într-o infrastructură cloud publică. Am explorat diferitele strategii de implementare și ce adaugă acestea la instalarea OpenStack și am argumentat de ce UPB a ales Kolla-ansible ca sistem de implementare pentru OpenStack.

Kolla-ansible ne oferă un echilibru bun între configurabilitatea rolurilor Ansible și ușurința de utilizare a implementării containerelor Docker pre-construite pe propriul nostru cluster și am arătat cum acest lucru a îmbunătățit strategia de implementare pentru cloud-ul privat UPB, evidențiind totodată diferitele componente OpenStack implementate folosind OpenStack.

De asemenea, am explorat performanța disponibilității ridicate a instanțelor și gazdei în OpenStack, folosind Masakari. Deși disponibilitatea ridicată a instanței funcționează conform așteptărilor, am întâmpinat câteva probleme cu monitorizarea gazdei. O astfel de problemă a fost stabilitatea, marcată de apariții aleatorii de dezactivare a serviciului pe noduri sănătoase. Intenționăm să investigăm cauza principală a acestui lucru.

O problemă evidențiată de cercetarea noastră a fost performanța, cu până la 7 minute de întrerupere a utilizatorului în cazul unei defecțiuni a gazdei. Analiza ne-a arătat că jumătate din acest timp este petrecut în bug-uri referitoare la procesarea notificărilor sau în valorile implicite pentru timeout-uri configurabile. Planul nostru este să investigăm necesitatea timeout-urilor, deoarece necesitatea acestora nu este documentată sau declarată clar.

De asemenea, vom continua să îmbunătățim securitatea ofertei de cloud privat, luând în considerare liniile directoare de securitate OpenStack și bazându-ne pe acestea.

4.2 Monitorizarea Infrastructurilor Cluster

Pe măsură ce dimensiunea clusterelor IT instituționale a crescut, probabilitatea defecțiunii sistemelor s-a transformat dintr-un eveniment rar într-un cost așteptat al afacerilor. Pentru ca serviciile esențiale să nu fie afectate, problemele de sistem trebuie detectate sau împiedicate să apară.

Administratorii de sistem folosesc instrumente de monitorizare pentru a extrage informații despre starea nodurilor și a aplicațiilor. Starea poate fi reprezentată de valori, cum ar fi rata de solicitare, utilizarea memoriei, timpul de răspuns pentru aplicații sau utilizarea resurselor pentru gazde. Jurnalele pot fi, de asemenea, o sursă de informații care poate indica probleme cu nodurile sau serviciile.

Există o gamă largă de instrumente de monitorizare și observabilitate care pot fi implementate și configurate. Fiecare soluție are avantaje, cum ar fi căutarea optimizată a informațiilor, suport pentru vizualizare, scalabilitate și configurare ușoară.

Vom prezenta un caz de utilizare pentru implementarea soluțiilor de monitorizare într-un mediu cluster HPC situat la Universitatea Națională de Științe și Tehnologie Politehnica București (UNSTPB). Descrierea infrastructurii va include serviciile și tipurile de noduri pe care le integrăm și problemele pe care le-am avut în implementarea instrumentelor de monitorizare.

4.2.1 Monitorizarea Infrastructurii

Pe baza state of the artului pentru soluțiile de monitorizare, a trebuit să alegem ce sistem să folosim în implementarea instituțională. Soluția de monitorizare selectată ar trebui să fie ușor de instalat și configurat. Ar trebui să ofere un sistem flexibil de configurare a nodurilor care să permită împărțirea gazdelor în grupuri. Ar trebui să putem vizualiza starea diferitelor sisteme implementate, cu panouri detaliate suplimentare disponibile pentru scufundări adânci. Ar trebui configurat un sistem de alertă care verifică stările nodurilor și notifică administratorii cu privire la întreruperi și defecțiuni ale sistemului.

Următorul capitol prezintă infrastructura implementată la UNSTPB, care îndeplinește cerințele menționate mai sus, utilizând software gratuit și open source. Vom detalia configurațiile realizate, soluțiile software alese și vom afișa vizualizările rezultate.

Prezentare Generală a Infrastructurii Clusterului

Clusterul UNSTPB este alcătuit dintr-o infrastructură cloud privată OpenStack [32] care servește nevoile de calcul ale personalului didactic și de cercetare. Cloudul privat rulează pe mașini fizice distribuite în trei centre de date care servesc atât ca noduri de calcul, cât și ca noduri de servicii, instalate prin instrumentul de implementare `kolla-ansible` [40]. Clusterul găzduiește, de asemenea, un grid SLURM [42] care rulează sarcini de lucru intensive în calcul pentru utilizatorii care doresc să ruleze joburi OpenMPI [37] pe mai multe mașini sau aplicații accelerate GPU. Atât software-ul grid,

cât și cloud folosesc un cluster de stocare CEPH [38] care este implementat într-o singură locație de centru de date.

Gestionarea Jurnalelor OpenStack

Pentru a realiza acest lucru, trebuie să luăm măsuri preventive pentru a ne asigura că sistemul are cât mai mult timp de funcționare posibil și că problemele pot fi rezolvate înainte ca acestea să împiedice funcționarea sistemului. Mesajele de jurnal pot fi ingerate într-un sistem centralizat de gestionare a jurnalelor, astfel încât să putem inspecta cu ușurință starea de sănătate a serviciilor și să investigăm cu ușurință jurnalele de servicii.

Am ales să folosim soluții standard din industrie, cum ar fi ELK Stack, constând din Logstash, Elasticsearch și Kibana, pentru a reduce efortul de a menține ingerarea, analizarea și afișarea informațiilor de jurnal. Folosim syslog pentru a agrega jurnalele de servicii și le trimitem la un sistem central de agregare a jurnalelor.

Am instalat Kibana pentru a vizualiza jurnalele odată ce sunt încărcate. Figura 4.5 afișează un filtru de jurnal utilizat în Kibana pentru a afișa jurnalele OpenStack. Filtrul detectează mesajele de tip `WARNING` lansate de serviciul `nova-compute` din cadrul serviciului Nova.

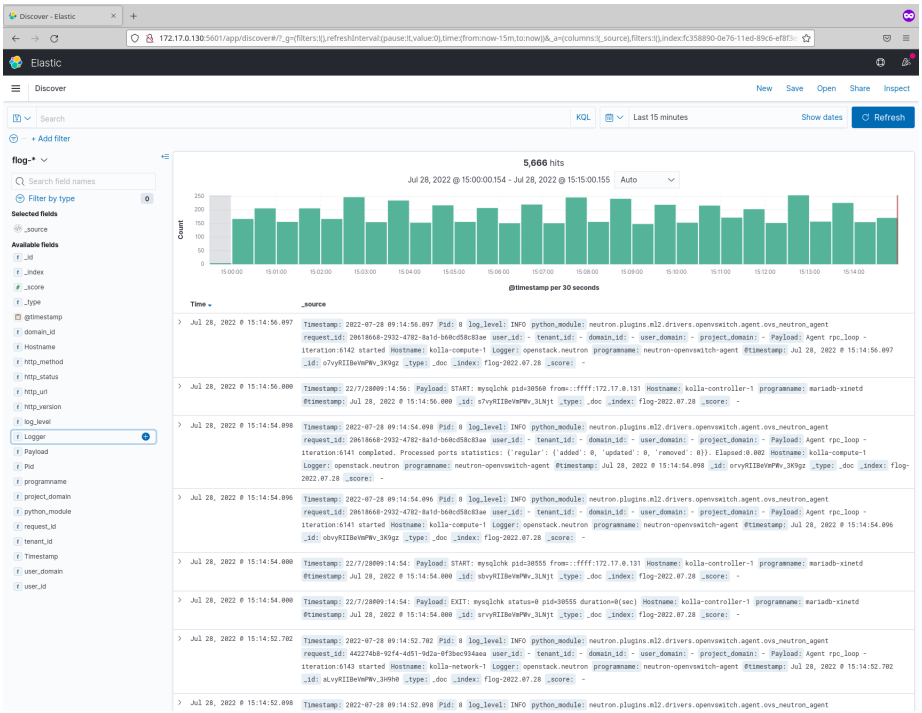


Figure 4.5: Căutarea jurnalelor în Kibana

Implementarea Serviciului de Colectare a Datelor de Rulare

Infrastructura OpenStack găzduiește un număr mare de servicii și noduri, ceea ce necesită o metodă scalabilă pentru a monitoriza în mod regulat starea de sănătate a gazdei și a serviciilor. Jurnalele nu sunt suficiente pentru a inspecta starea de sănătate a unui serviciu, deoarece sunt o modalitate de a afișa anumite erori, dar nu afișează un mesaj

simplu de analizat. Metricile pot fi folosite pentru a analiza dintr-o privire starea generală a serviciilor și sistemelor. Acestea pot fi utilizate pentru a determina atât parametrii operaționali în care se așteaptă să ruleze serviciile și sistemele, cât și abaterile de la comportamentul așteptat. Jurnalele ne-ar permite doar să vedem când un sistem eșuează efectiv, nu procesul iterativ al eșecului său.

Pentru colectarea datelor statistice am ales să folosim serviciul Prometheus. Prometheus funcționează prin conectarea la o listă de puncte finale configurate pentru a exporta parametrii de rulare.

Am ales să folosim Prometheus deoarece rulează folosind modelul pull, unde se conectează la servicii, în loc de modelul push, unde serviciile se conectează direct la un nod central. Dezavantajul utilizării Prometheus este că trebuie să asigurăm conectivitatea între sistemul pe care rulează Prometheus și toate stațiile de la care dorim să colectăm informații.

Prometheus oferă o interfață grafică sub forma unei pagini web unde putem rula interogări și vizualiza rezultatele.

Deoarece a fost necesară configurarea serviciului Prometheus pentru mai mult de 100 de noduri fizice, fiecare găzduind mai multe servicii, am dezvoltat un set de scripturi care generează fișiere de configurare Prometheus pe baza unui set de servicii și grupuri de noduri care sunt asociate cu.

Deoarece serviciul Prometheus ingerează o cantitate mare de date, am ales să implementăm o politică de păstrare a datelor. Această politică are scopul de a automatiza eliminarea datelor vechi de monitorizare.

Am integrat următoarele informații folosind diferiți exportatori:

- Starea și statisticile serviciului OpenStack;
- Utilizarea resurselor nodului de calcul;
- Starea serviciului web, cum ar fi GitLab, servicii de conectare, platforme de e-learning și sisteme de stocare;
- Durata de viață a certificatului SSL pentru serviciile web gestionate;
- Puterea generatorului centrului de date și starea combustibilului;
- Scanner ICMP pentru verificarea latenței între sisteme;
- Metrici de control ale serviciului grid;
- Statistici de stocare, cum ar fi utilizarea lățimii de bandă și timpul IO;
- Valori de utilizare a GPU.

Investigarea stării de sănătate a serviciilor folosind tabloul de bord

Pentru investigarea avansată a sănătății sistemelor, am ales să folosim serviciul Grafana pentru a genera grafice și tablouri de bord din care putem vizualiza imagini complexe bazate pe serii de timp. Am ales să folosim Grafana datorită suportului extins de vizualizare, deoarece diferite servicii au nevoi diferite de vizualizare. Grafana ne oferă

funcționalitatea de interogare a serviciului Prometheus, astfel încât să putem folosi interogările Prometheus Query Language pe care le-am folosit până acum direct din tabloul de bord Prometheus pentru a vizualiza cu ușurință informațiile.

Grafana este configurat să acceseze informații în funcție de intervalul de timp la care solicităm accesul. Întrucât au existat cazuri în care sunt prea multe sisteme pentru care colectăm informațiile, am ales să adăugăm fiecărei metrici din Prometheus o etichetă care definește tipul de serviciu oferit, astfel încât să putem filtra valorile în interogări în funcție de tipul de serviciu pe care dorim să-l investigăm.

Avantajul utilizării Grafana pentru grafice este că are o suită foarte flexibilă de moduri de vizualizare, care ne permite să prezentăm informații în multe moduri diferite, de la grafice obișnuite și rezumate numerice până la hărți termice și grafice histograme. Putem integra alte tablouri de bord în Grafana creându-ne propriile tablouri de bord sau utilizând tablourile de bord existente publicate de comunitate.

Am integrat în tablourile de bord Grafana pentru vizualizarea următoarelor elemente:

- Starea infrastructurii OpenStack;
- Latența conexiunii între noduri;
- Stare serviciilor web;
- Stare generatorului de energie;
- Consumul de resurse al nodurilor gazdă;
- Starea sitemelor de stocare CEPH;
- Starea și utilizarea serviciilor Grid;
- Nivelul de utilizare al GPU-urilor.

Monitorizarea Stării Discurilor

UNSTPB rulează o cantitate mare de hard disk-uri pentru diferite cazuri de utilizare. Hard disk-urile se uzează în timp, ceea ce necesită monitorizarea erorilor care pot apărea din cauza defecțiunii hard diskului.

Trebuia să avem o metodă pentru a verifica starea de sănătate a discului și pentru a preveni o situație în care ar putea eșua în timpul încărcărilor la cerere mare. Pentru a inspecta starea discului la un moment dat, putem folosi S.M.A.R.T. valorile expuse de disc, care ne oferă o perspectivă asupra uzurii discului. S.M.A.R.T. valorile care ne interesează sunt rata de eroare de citire, temperatura de rulare a discului și câte erori apar la transmiterea datelor de pe disc la sistem. Acestea pot indica primele semne că un disc s-ar putea defecta în viitorul apropiat.

Pentru a interpreta S.M.A.R.T. caracteristici folosim valorile recomandate de producatori pentru starea discurilor.

Coloanele care ne interesează sunt:

Pentru a expune valorile într-un mediu ușor de citit, am folosit un script care colectează S.M.A.R.T. și le expune folosind serviciul `node_exporter` deja instalat. Serviciul

Prometheus, instalat automat pe instanța Grafana, extrage valorile. Acestea sunt folosite pentru a genera ieșire tabloul de bord Grafana prin care putem vizualiza starea discurilor, unde discurile suspecte de defecțiune pot fi evidențiate prin intermediul metricilor S.M.A.R.T.

4.2.2 Infrastructura de Alertă

În această subsecțiune vom discuta despre alegerile făcute pentru a implementa un sistem automat de alertă care poate fi configurat folosind metricile existente pentru a trimite mesaje către administratorii de sistem dacă sunt detectate probleme cu sistemele.

Deoarece avem acces la o cantitate mare de metrici prin accesarea serviciului Prometheus, putem determina cu ușurință starea sistemelor la un moment dat.

Problemele de infrastructură trebuie detectate cât mai devreme posibil, iar administratorii trebuie anunțați în prealabil atunci când un serviciu nu funcționează în parametrii corespunzători. Din acest motiv dorim să avem cea mai rapidă metodă posibilă de a fi alertați atunci când apar probleme.

Am ales să folosim AlertManager ca sistem de alertă. Avantajul utilizării AlertManager este că se poate conecta direct la Prometheus și poate interoga baza de date de valori. Astfel, putem folosi AlertManager pentru a interoga constant starea sistemului și a trimite mesaje către administratorii responsabili în cazul în care un serviciu nu mai funcționează sau au apărut defecțiuni tehnice la un nod.

Am activat serviciul AlertManager să trimită e-mail direct administratorilor de sistem pentru orice eveniment critic care are loc, pentru a permite notificarea timpurie a administratorilor. Dacă apar evenimente succesive, administratorii vor fi anunțați pentru fiecare. Dacă alertele sunt alarme false sau sunt o problemă recunoscută, administratorii pot dezactiva regulile folosind expresii pentru a dezactiva în mod granular regulile de alertă.

Am creat notificări AlertManager pentru următoarele situații:

- Servicii de rețea care nu pot fi contactate;
- Prea mult spațiu pe disc folosit pe un nod;
- Servicii de calcul care nu pot fi contactate;
- Servicii web care nu pot fi contactate;
- Prea multă memorie folosită pe noduri;
- Certificatul unui serviciu web va expira în următoarele două săptămâni;
- Un sistem nu poate fi contactat.

Folosim valorile expuse prin S.M.A.R.T. pentru a detecta o eroare de disc înainte ca aceasta să apară. Folosim `WORST` și `THRESH` pentru a identifica dacă o unitate a dat deja semne de defecțiune. Dacă valoarea `WORST` se apropie de valoarea `THRESH`, înseamnă că unitatea va eșua în curând.

4.2.3 Concluzii și Direcții Ulterioare

Am cercetat sistemul optim de monitorizare pentru gestionarea valorilor pentru o infrastructură cloud privată care găzduiește și sisteme instituționale și de e-learning.

Fiecare soluție de monitorizare viabilă a fost analizată și am implementat Prometheus pentru a descărca valori de la noduri, sisteme de stocare și servicii. Grafana a fost folosit pentru a afișa valorile într-un nou tablou de bord care le permite administratorilor să vadă dintr-o privire starea clusterului.

Am implementat un sistem de alertă pentru a anunța administratorul atunci când serviciile eșuează, certificatele SSL expiră sau când discurile prezintă semne de degradare. Noul sistem ne-a permis să prevenim incidentele de defecțiune a discurilor prin înlocuirea discurilor și reconstruirea matricelor de stocare la timp și să observăm evenimente anormale în comportamentele de serviciu care duc la remedieri de erori.

În viitor, ne propunem să mutăm configurația de monitorizare de la mașinile virtuale într-un mediu bazat pe containere folosind configurare automată, astfel încât să poată fi mai ușor de instalat și replicat pentru utilizare ulterioară.

Chapter 5

Concluzie

Calculul distribuit a permis să se realizeze mai multe procesări în paralel, accelerând atât cercetarea, cât și educația, facilitând utilizatorilor să ruleze sarcini complexe de lucru și automatizarea într-un mediu de la distanță. Cererea pentru mai multe procesări paralele a condus la adoptarea cluster computing și grid computing. În acest mediu nou și dinamic, trebuie să descoperim noi modalități de a include resurse în cluster și grid computing și de a le adapta la sarcinile noastre de lucru, astfel încât să putem profita de puterea lor. Pentru a profita de aceste resurse nou introduse, trebuie să fim, de asemenea, capabili să le folosim eficient, petrecând mai puțin timp în sarcini non-informatice, cum ar fi pregătirea mediului, cum ar fi transferul de fișiere.

5.1 Rezumat

Această teză a propus noi căi de utilizare a middleware-ului pentru a îmbunătăți mediile cluster și grid prin creșterea eficienței utilizării resurselor prin îmbunătățirea paralelismului în timpul transferurilor de fișiere și accesarea noilor tipuri posibile de resurse, cum ar fi cozile scavenging. Prin implementarea suportului pentru joburi cu mai multe nuclee, am activat mediul ALICE grid la sarcini de lucru cu o amprentă mai mică de memorie pe nucleu, permițând mai multe joburi pe care le rulați într-un mediu cu memorie limitată.

Am implementat aceleași tehnici middleware folosite pentru a gestiona infrastructurile grid pentru a automatiza activitățile educaționale, cum ar fi verificarea sarcinilor și construirea materialelor folosite în scop educațional Am folosit soluții existente centrate pe dezvoltatori, cum ar fi GitLab și constructori de documentație CI/CD pentru a ne atinge obiectivele de a automatiza verificarea temelor de clasă.

O infrastructură de cloud computing de înaltă disponibilitate a fost implementată în clusterul UNSTPB, folosind tehnici moderne de instalare, oferind medii virtuale rezistente la defect pentru utilizatorii orientați spre cercetare și educație. Teza a definit prin studiul de caz al modelelor de implementare de referință a clusterului UNSTPB pentru monitorizarea infrastructurilor și implementări de cluster de înaltă disponibilitate utilizate pentru virtualizarea serviciilor educaționale și a încărcăturilor de lucru în rețea.

5.2 Contribuții

Teza analizează posibilele optimizări care pot fi făcute transferurilor în grid pentru a crește lățimea de bandă de transfer și a reduce timpii de transfer. În contextul campaniei de transfer al experimentului ALICE, am reușit să **scăderea timpilor de transfer** pentru datele generate de experiment prin creșterea ratei de transferuri gestionate de serviciile grid în funcție de numărul de fișiere transferate și de dimensiunea acestora . În timpul testelor de transfer, am reușit să atingem rata anunțată de 100 GBps, ceea ce ar permite experimentului să transfere în mod continuu date către site-uri de stocare fără ca bufferul principal să fie umplut. Prin creșterea ratelor de transfer am accelerat procesarea datelor, deoarece acestea pot fi accesate mai repede de către părțile interesate.

Am scăzut utilizarea memoriei pentru grila ALICE prin implementarea suportului pentru gestionarea joburilor cu mai multe nuclee, permițând executarea joburilor cu o utilizare mai mică de memorie pe grid, oferind astfel site-urilor cu deficit de memorie capacitatea de a rula mai multe joburi în paralel fără creșterea memoriei RAM disponibilă. În implementarea suportului pentru joburi multi-core am luat în considerare problema încărcărilor de lucru care rulează pe mai multe nuclee decât alocate, **demonstrând capacitatea de a limita nucleele CPU utilizate prin utilizarea mecanismului `taskset`**.

Am identificat în această teză o clasă de clustere care ar putea fi integrate în mediul grid ALICE dacă s-ar adăuga suport pentru gestionarea mai multor joburi în paralel printr-un singur job pilot. În experimentul ALICE, am implementat suport pentru rularea mai multor încărcături utile din aceleași joburi pilot și gestionarea unui slot cu mai multe nuclee sau a unui nod întreg, luând în considerare utilizarea resurselor jobului și resursele disponibile în prezent. **Prin implementarea suportului pentru sloturi cu mai multe joburi, am reușit să integrăm mai multe resurse de calcul în rețea.**

Ca parte a acestei teze, software-ul a fost **proiectat și dezvoltat pentru a ajuta educatorii în automatizarea generării documentației**. Aplicația a fost dezvoltată pe baza tehnicilor DevOps pentru generarea de materiale educaționale în mod modular, care în primă fază implementează Docusaurus pentru a genera conținut HTML pe baza fișierelor Markdown și publica fișierele pe Internet folosind GitLab CI/CD. Întrucât clasele sunt construite și publicate automat online, alți profesori pot refolosi materialul, îl pot edita folosind Git, îl pot îmbunătăți și îl pot publica automat, fără a fi nevoie să-și creeze propriul material.

O soluție de verificare a temelor, numită vmchecker, a fost dezvoltată, bazată pe principiile de gestionare a locurilor de muncă din spatele software-ului de gestionare a rețelei, folosind sarcini de lucru pentru a corecta temele studenților prin pipeline-uri CI/CD automatizate. vmchecker permite atât utilizatorilor, cât și profesorilor să verifice temele folosind același mediu și scripturi și încarcă automat rezultatele verificării într-un front-end, cum ar fi platforma de e-learning Moodle. Platforma fiind implementată în clusterul UNSTPB și utilizată pentru mai mult de 10 materii, a verificat peste 40.000 de teme pentru studenți și a redus timpul petrecut de profesori pentru notarea și verificarea manuală a temelor elevilor.

Teza propune o arhitectură pentru implementarea mediilor cluster instituționale care pun accent pe fiabilitate, flexibilitate și ușurință în utilizare pentru utilizatorul final. **Un exemplu de arhitectură a fost implementat în clusterul UNSTPB**, rulând OpenStack și găzduind proiecte de cercetare și servicii instituționale într-un mod de disponibilitate înaltă, pe sisteme bazate pe mașini virtuale.

Domeniul monitorizării clusterelor a fost explorat în această teză, concentrându-ne pe instrumente care pot ajuta administratorii să mențină o vedere de ansamblu asupra stării clusterului la un moment dat. Au fost investigate instrumente de monitorizare care pot permite utilizatorilor să depaneze sistemele complexe și să permită utilizarea măsurilor preventive, cum ar fi alertele pentru a preveni problemele de service. **Teza propune o arhitectură de referință implementată în clusterul UNSTPB bazată pe Grafana, Prometheus și AlertManager**, configurată pentru a extrage datele nodurilor, datele de serviciu și starea sistemelor de stocare și să alerteze părțile implicate în cazul unei defecțiuni, cum ar fi o eroare pe discuri sau un serviciu care nu răspunde la cereri.

Bibliography

[1] alimonitor transfers web page. https://alimonitor.cern.ch/transfers. Accessed: 2023-02-12.

[2] Amazon Web Services Cloudformation. https://docs.aws.amazon.com/AWSCloudFormation/latest/A Accessed: 2023-02-12.

[3] Checkmk website. https://www.checkmk.com. Accessed: 2023-10-04.

[4] Docsy documentation builder. https://www.docsy.dev/. Accessed: 2023-09-30.

[5] MariaDB. mariadb.org.

[6] Masakari. https://docs.openstack.org/masakari/latest/. Accessed: 2023-02-12.

[7] OpenStack Cinder. https://docs.openstack.org/cinder/latest/. Accessed: 2023-02-12.

[8] OpenStack Glance. https://docs.openstack.org/glance/latest/. Accessed: 2023-02-12.

[9] OpenStack Heat. https://docs.openstack.org/heat/latest/. Accessed: 2023-02-12.

[10] OpenStack Keystone. https://docs.openstack.org/keystone/latest/. Accessed: 2023-02-12.

[11] OpenStack Neutron. https://docs.openstack.org/neutron/latest/. Accessed: 2023-02-12.

[12] OpenStack Nova. https://docs.openstack.org/nova/latest/. Accessed: 2023-02-12.

[13] PostgreSQL. https://www.postgresql.org/. Accessed: 2023-02-12.

[14] reveal-md git repository. https://github.com/webpro/reveal-md. Accessed: 2023-09-30.

[15] taskset(1) — linux manual page. https://man7.org/linux/man-pages/man1/taskset.1.html. Accessed: 2023-10-04.

[16] XRootD documentation. https://www.xrootd.org. Accessed: 2023-02-12.

[17] K. Aamodt et al. The ALICE experiment at the CERN LHC. *JINST*, 3:S08002, 2008.

[18] B Abelev et al. Upgrade of the ALICE Experiment: Letter Of Intent. *J. Phys. G*, 41:087001, 2014.

[19] Maria Arsuaga-Rios, Vladimír Bahýl, Manuel Batalha, Cédric Caffy, Eric Cano, Niccolo Capitoni, Cristian Contescu, Michael Davis, David Alvarez, Jaroslav Guenther, Edouardos Karavakis, Oliver Keeble, Julien Leduc, Fabio Luchetti, Luca Mascetti, Steven Murray, Mihai Patrascoiu, Andreas Peters, Michal Simon, and Rainer Toebbicke. Lhc data storage: Preparing for the challenges of run-3. *EPJ Web of Conferences*, 251:02023, 01 2021.

[20] Stefano Bagnasco, L Betev, P Buncic, F Carminati, C Cirstoiu, C Grigoras, A Hayrapetyan, A Harutyunyan, AJ Peters, and P Saiz. Alien: Alice environment on the grid. In *Journal of Physics: Conference Series*, volume 119, page 062012. IOP Publishing, 2008.

[21] Emiliano Betti, Daniel Pierre Bovet, Marco Cesati, and Roberto Gioiosa. Hard real-time performances in multiprocessor-embedded systems using asmp-linux. *EURASIP Journal on Embedded Systems*, 2008:1–16, 2007.

[22] Sebastien Binet, P Calafiura, MK Jha, W Lavrijsen, C Leggett, D Lesny, H Severini, D Smith, S Snyder, M Tatarkhanov, et al. Multicore in production: Advantages and limits of the multiprocess approach in the atlas experiment. In *Journal of Physics: Conference Series*, volume 368, page 012018. IOP Publishing, 2012.

[23] R. Brun and F. Rademakers. ROOT: An object oriented data analysis framework. *Nucl. Instrum. Meth. A*, 389:81–86, 1997.

[24] Mainak Chakraborty and Ajit Pratap Kundan. Grafana. In *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*, pages 187–240. Springer, 2021.

[25] Jason Cole and Helen Foster. *Using Moodle: Teaching with the popular open source course management system.* " O'Reilly Media, Inc.", 2007.

[26] Michael C. Davis, Vladímir Bahyl, Germán Cancio, Eric Cano, Julien Leduc, and Steven Murray. CERN Tape Archive - from development to production deployment. *EPJ Web Conf.*, 214:04015, 2019.

[27] Kris Jamsa. *Cloud computing.* Jones & Bartlett Learning, 2022.

[28] Kathy Kincade. Cori supercomputer now fully installed at berkeley lab. https://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2016/cori-supercomputer-now-fully-installed-at-berkeley-lab/. Accessed: 2023-10-04.

[29] M Martinez Pedreira, C Grigoras, and V Yurchenko. Jalien: the new alice high-performance and high-scalability grid framework. In *EPJ Web of Conferences*, volume 214, page 03037. EDP Sciences, 2019.

[30] Andreas Peters and Lukasz Janyst. Exabyte scale storage at cern. *Journal of Physics: Conference Series*, 331, 12 2011.

[31] Rami Rosen. Resource management: Linux kernel namespaces and cgroups. *Haifux, May*, 186:70, 2013.

[32] Omar Sefraoui, Mohammed Aissaoui, Mohsine Eleuldj, et al. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3):38–42, 2012.

[33] Jamie Shiers. The worldwide lhc computing grid (worldwide lcg). *Computer physics communications*, 177(1-2):219–223, 2007.

[34] Vandana Singh and Alexander Thurman. How many ways can we define online learning? a systematic literature review of definitions of online learning (1988-2018). *American Journal of Distance Education*, 33(4):289–306, 2019.

[35] Diomidis Spinellis. Git. *IEEE software*, 29(3):100–101, 2012.

[36] Alvaro Videla and Jason JW Williams. *RabbitMQ in action: distributed messaging for everyone*. Manning, 2012.

[37] David W Walker and Jack J Dongarra. Mpi: a standard message passing interface. *Supercomputer*, 12:56–68, 1996.

[38] Sage Weil, Scott A Brandt, Ethan L Miller, Darrell DE Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06)*, pages 307–320, 2006.

[39] Sergiu Weisz and Marta Bertran Ferrer. Adding multi-core support to the alice grid middleware. In *Journal of Physics: Conference Series*, volume 2438, page 012009. IOP Publishing, 2023.

[40] Sergiu Weisz, Vlad-Iulius Năstase, Maria-Elena Mihăilescu, Darius Mihai, and Nicolae Ţăpuş. Deploying large private clouds for high availatibily services. In *2023 24th International Conference on Control Systems and Computer Science (CSCS)*, pages 67–74. IEEE, 2023.

[41] Yoji Yamato, Yukihisa Nishizawa, Shinji Nagao, and Kenichi Sato. Fast and reliable restoration method of virtual resources on openstack. *IEEE Transactions on Cloud Computing*, 6(2):572–583, 2015.

[42] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[43] Chiara Zampolli. Alice data processing for run 3 and run 4 at the lhc, 2020.