Gestionarea Colaborativă a Distribuției și Execuției Proceselor într-un Cluster OpenMPI

Ioan CÎRJĂ

Facultatea de Automatică și Calculatoare,
Universitatea Tehnică "Gheorghe Asachi" din Iași,
Iași, România
ioan.cirja@student.tuiasi.ro

Ș.l.dr.ing. Silviu Dumitru PAVĂL

Departamentul de Calculatoare,
Facultatea de Automatică și Calculatoare,
Universitatea Tehnică "Gheorghe Asachi" din Iași,
Iași, România
silviu-dumitru.paval@academic.tuiasi.ro

***Abstract*—Proiectul propune dezvoltarea unei aplicații destinate gestionării colaborative a proceselor executate într-un cluster OpenMPI, cu scopul de a facilita accesul la resurse de calcul paralelizat și distribuit. Soluția oferă o interfață web pentru trimiterea, și configurarea programelor distribuite, precum și monitorizarea actualizărilor și rezultatelor în timp real asupra stării execuției, astfel, eliminând complexitatea utilizării directe a liniei de comandă. Proiectul poate oferi sprijin în domeniul educațional și în cercetare, unde accesul la infrastructuri distribuite trebuie să fie cât mai accesibil și ușor de utilizat.**

***Index Terms*—OpenMPI, cluster, aplicație web**

I. Introducere

În contextul actual al dezvoltării tehnologice, clusterele utilizate în procese ce implică calcul distribuit și care sunt bazate pe standardul MPI (Message Passing Interface) reprezintă o modalitate des abordată în rezolvarea problemelor complexe care necesită resurse computaționale mari sau diferite tipuri de optimizări. **MPI (Message Passing Interface)** [5] este un standard deschis, conceput pentru programarea calculului paralel și care definește un set de funcționalități și API-uri care permit comunicarea între mai multe procese ce rulează pe nuclee diferite, procesoare distincte sau chiar pe sisteme distribuite conectate într-o rețea. Printre capabilitățile importante ale standardului MPI putem enumera:

- **Comunicarea punct-la-punct** - utilizată pentru a trimite și primi mesaje direct între procese;
- **Comunicarea colectivă** - permite coordonarea și schimbul de date între grupuri de procese;
- **Topologiile de procese [4]** - permit manipularea etichetelor proceselor în diverse structuri, cum ar fi topologia cartesiană, hipercub, etc.

Aria de aplicabilitate a standardului MPI este foarte largă, fiind utilizat în simulări științifice, procesarea datelor, aplicații de inteligență artificială, analize de date etc. Cu toate acestea, interacțiunea cu un cluster MPI poate fi considerată un proces laborios, care crește rapid în complexitate. Utilizarea unor instrumente software care să simplifice această sarcină ar putea facilita un nivel crescut de eficiență pentru utilizatori.

Motivarea alegerii acestei teme provine din dorința de a aprofunda și extinde noi metodologii de utilizare a standardelor MPI și a bibliotecilor care le implementează [3] în cadrul unor unități de procesare individuale sau a clusterelor. Acest proiect urmărește să creeze o aplicație web care să faciliteze accesul la resursele de calcul distribuit într-un mod rapid, fără a fi necesară o configurare inițială.

II. Stadiul Actual

Dezvoltarea soluțiilor de interacțiune, vizualizare și monitorizare a calculului distribuit, precum joburile MPI, se bazează în mare parte pe linia de comandă și pe comenzi specifice complexe. O serie de soluții care pot fi utilizate în acest scop sunt *TotalView Debugger*, *DDT Debugger* sau *Paraver*; cu toate acestea, se observă limitări în ceea ce privește interacțiunea cu utilizatorul și prezentarea interfeței, aplicațiile fiind concentrate pe aspectul de depanare a codului si nu al monitorizării per ansamblu.

Abordarea gestionării joburilor destinate procesărilor computaționale paralelizate folosind o interfață grafică nu este frecvent întâlnită. Aceasta implică dezvoltarea și mentenanța unor resurse software și infrastructuri capabile să gestioneze calculul distribuit. Cu toate acestea, o astfel de aplicație oferă o perspectivă nouă asupra utilizării soluțiilor MPI și avantaje precum utilizarea în scop academic [1] [2], învățare și accesul facil la procese de calcul paralel, fără necesitatea pregătirii prealabile a unei infrastructuri complexe și a întreținerii acesteia.

Golul în cercetare derivă din faptul că, deși astfel de soluții software pot sprijini eficient utilizarea și studierea calculului distribuit, ele nu sunt adoptate pe scară largă.

III. Obiective

Tema proiectului constă în dezvoltarea unei aplicații care expune o interfață web pentru gestionarea taskurilor trimise spre execuție într-un cluster OpenMPI.

Aplicația va permite utilizatorilor să trimită taskuri spre execuție, care au diferite configurații de parametri, flaguri și executabile, să selecteze topologia și modul de distribuție al taskurilor în cluster, să monitorizeze periodic starea acestora și să vizualizeze rezultatele finale, acestea fiind centralizate în secțiuni dedicate ale aplicației. Accesul la funcționalitate se va face în baza unui cont de utilizator. Aplicația web va

afișa date despre noduri, va permite vizualizarea conținutului fișierelor și setarea preferințelor. Printre obiectivele principale putem enumera:

- **Dezvoltarea unei interfețe web** - presupune alcătuirea unui design și implementarea acestuia folosind un framework de frontend.
- **Implementarea logicii de bussiness** - va gestiona comunicarea cu clusterul OpenMPI, inclusiv trimiterea și monitorizarea taskurilor. De asemenea, va implementa logica aplicației, serviciile web și legăturile dintre ele.
- **Automatizarea monitorizării taskurilor** - reprezintă unul dintre obiectivele principale al aplicației și presupune crearea unui sistem care poate monitoriza execuția proceselor trimise către cluster în timp real, oferind notificări, actualizări și elemente de output.
- **Testarea și securizarea aplicației** - aplicația va fi supusă unor teste unitare, de integrare și de performanță. Se vor avea în vedere și vor fi tratate diferite cazuri de utilizare și comportamentul aplicației în astfel de situații.
- **Redactarea documentației proiectului** - documentația va include descrierea detaliată a arhitecturii aplicației, a proceselor de implementare a backendului și frontendului, precum și un ghid de utilizare.

IV. Metodologie

A. Resurse utilizate

Elaborarea proiectului necesită o serie de resurse software, printre care enumerăm:

- FastAPI – framework destinat implementării logicii de backend, alcătuită din microservicii Python.
- OpenMPI – utilizat de către cluster pentru a permite paralelizarea și calculul distribuit pe unitățile de procesare.
- Docker– utilizat pentru containerizarea aplicației și extindere.
- Angular– framework utilizat pentru dezvoltarea interfeței utilizator.
- Python, JavaScript/TypeScript – limbaje utilizate pentru logica backend, algoritmi și interogarea clusterului, respectiv pentru dezvoltarea interfeței utilizator.
- Firebase Real Time Database – utilizat în stocarea persistentă a datelor aferente microserviciilor; prezintă suport pentru sincronizare în timp real, într-un mod nonrelațional
- Mediile de dezvoltare PyCharm Professional și Visual Studio Code – utilizate pentru scrierea și depanarea codului.
- Alte instrumente – comenzi bash, SSH și diferite instrumente de monitorizare, precum cron jobs sau Celery.

Resursele hardware sunt reprezentate de unitățile de procesare din cadrul clusterului MPI, împreună cu configurația lor. Clusterul include noduri multiple optimizate pentru calcule paralele, fiecare echipat cu procesoare multi-core (CPU) de înaltă performanță și memorie RAM suficientă pentru a gestiona volume mari de date și operații intensive.

Metodologia de lucru se bazează pe un proces iterativ, cu revizuiri periodice pentru a îmbunătăți și a soluționa problemele care pot apărea pe parcurs.

B. Metode și algoritmi

În cadrul dezvoltării la nivel frontend, arhitectura este bazată pe componente care gestionează logica aferentă diferitelor părți din interfața utilizator, ex: componenta de autentificare, componenta de încărcare a unui job. Intercomunicarea dintre acestea se realizează prin module specifice Angular.

În cadrul dezvoltării backend, activitatea principală presupune implementarea microserviciilor aferente aplicației, integrarea funcționalităților acestora printr-o metodă de centralizare, e.g. gateway, și comunicarea cererilor și răspunsurilor HTTP cu interfața utilizator. Clientul, prin intermediul frontendului, va trimite o solicitare către API-ul centralizat, iar acesta o va redirecționa către microserviciul potrivit, în funcție de cerere. Odată ce microserviciul va efectua procesările necesare cererii, precum comunicarea cu baza de date, trimiterea joburilor către cluster, alcătuirea comenzilor specifice OpenMPI și conexiunea SSH, acesta va crea un răspuns care va fi returnat și vizualizat de utilizator prin intermediul interfeței.

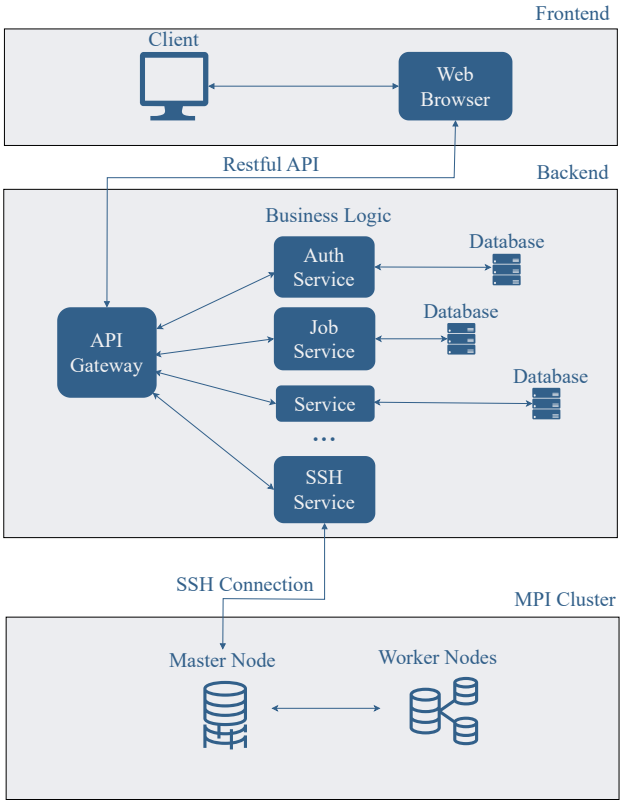

Fig. 1. Diagrama generală a aplicației

Alegerea arhitecturii de microservicii [6] derivă din faptul că aplicația deține o serie de funcționalități care pot fi ușor

separate și scalate, iar resursele aferente unui microserviciu pot fi gestionate mai eficient (exemplu: serviciul SSH nu necesită o conexiune la baza de date).

Implementarea la nivel backend urmărește o reprezentare RESTful, urmărind un model client-server cu comunicări uniforme de tip HTTP (GET, POST, PUT, DELETE) și reprezentări standard ale resurselor (de exemplu, JSON, XML). Solicitările clientului trebuie să conțină toate informațiile necesare pentru a putea fi înțelese și procesate de server, iar răspunsurile oferite de server vor conține coduri de stare HTTP pentru detalierea rezultatului solicitării. De asemenea, se va implementa principiul HATEOAS (Hypermedia as the Engine of Application State) [7], astfel încât răspunsurile trebuie să conțină linkuri către alte resurse, permițând clientului să navigheze prin aplicație fără a avea cunoștințe prealabile despre structura acesteia.

Backendul va trebui să gestioneze logica de interacțiune cu clusterul OpenMPI, monitorizarea recurentă a joburilor și comunicarea în timp real. Pentru acest lucru se vor folosi o combinație de strategii de implementare, precum alocarea de resurse dedicate pentru procesarea și așteptarea rezultatelor unui job, dar și implementarea unor servicii recurente care interoghează starea și resursele clusterului pe parcursul rulării aplicației.

Pe lângă acestea, se urmărește transmiterea configurației, specificarea parametrilor de execuție (nr. de noduri, selecția de host-uri, resursele alocate unui proces, transmiterea de fișiere între nodurile de cluster) ca fiind opțiuni ce vor fi transmise prin cadrul interfeței.

De asemenea, se vor utiliza o serie de API-uri care facilitează conexiunea cu resurse externe, precum scanarea fișierelor executabile pentru un potențial malware înainte de a fi prelucrate de cluster și conexiunea în timp real cu baza de date.

C. Diagrame de secvență

Diagramele de secvență descriu o serie de pași esențiali din interacțiunile componentelor aplicației și oferă o viziune de ansamblu asupra fluxului de execuție. Mai jos sunt prezentate trei porțiuni semnificative:

- **Conectarea la cluster prin SSH** - diagrama ilustrează procesul de conectare prin SSH atunci când se dorește interacțiunea cu clusterul. Procesul se realizează automat odată ce frontendul înregistrează trimiterea unui job sau interogarea statusului.
- **Pașii pentru încărcarea unui job** - se prezintă procesul de încărcare a unui job, incluzând validarea fișierelor și transmiterea acestora către cluster, precum și configurarea parametrilor jobului.
- **Verificarea statusului joburilor** - diagrama descrie procesul de verificare a statusului joburilor executate pe cluster și obținerea rezultatelor finale ale execuției.

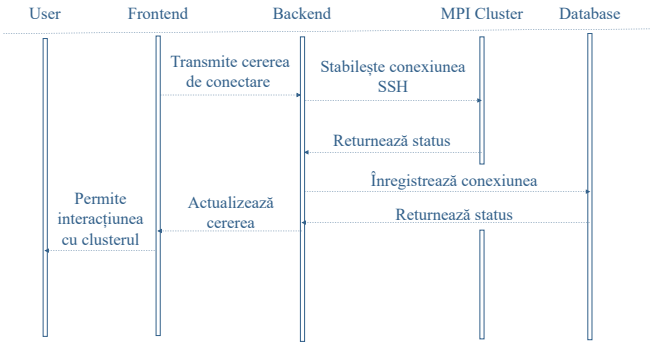

Fig. 2. Conectarea la cluster prin SSH.

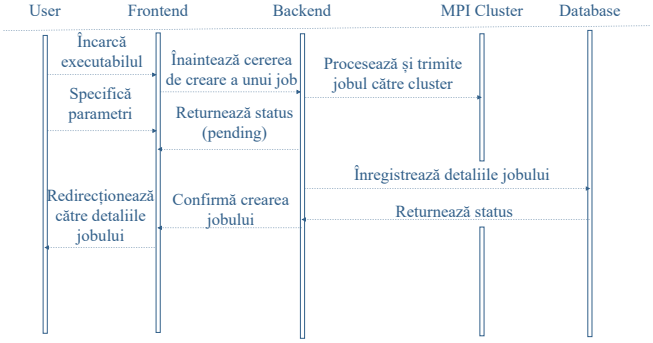

Fig. 3. Pașii pentru încărcarea unui job.

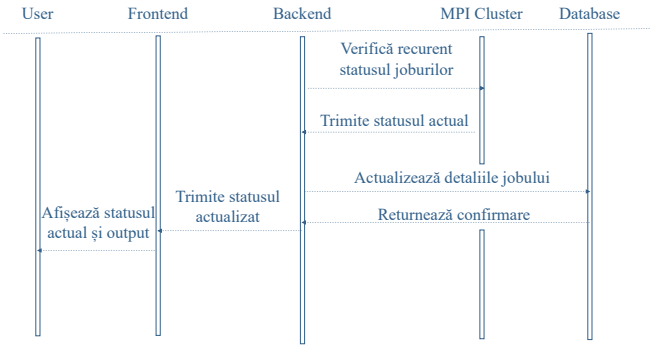

Fig. 4. Verificarea statusului joburilor.

D. Wireframes

Această secțiune prezintă wireframe-uri relevante pentru structura și funcționalitățile interfeței aplicației:

- **Formularul de încărcare a unui job** - se prezintă modul în care utilizatorii pot încărca fișiere și specifica setările necesare pentru execuția acestora.
- **Pagina de status a unui job** - pagina de status arată cum utilizatorii pot vizualiza statusul curent al unui job

și rezultatele obținute, într-o formă ușor de urmărit. Configurarea și rezultatele unui job constituie două secțiuni separate ce pot fi accesate prin butoane expand/collapse.

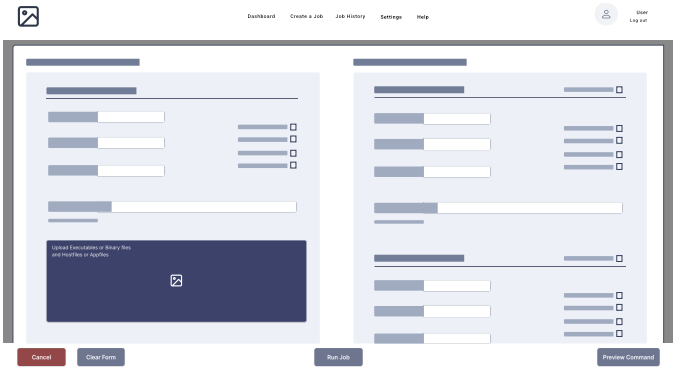

Fig. 5. Wireframe - încărcarea unui job.

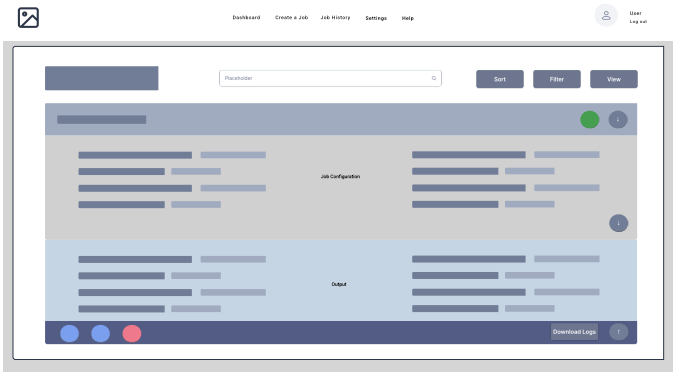

Fig. 6. Wireframe - pagina de status a unui job.

V. Prezentarea prototipului aplicației

La nivelul interfeței, odată ce a fost creat un cont destinat unui utilizator obișnuit, care nu are drepturi de administrator asupra clusterului, acesta se poate autentifica și își poate vizualiza contul, setările și joburile aferente.

Centrul aplicației este reprezentat de paginile pentru crearea unui job, respectiv vizualizarea statusului în timp real. Pentru a crea un job, utilizatorul trebuie să navigheze pe pagina aferentă și să introducă o configurație minimă pentru un job. Astfel, enumerăm posibilele date:

- nume și descriere, folosite pentru a explica și identifica scopul jobului și ce se dorește a fi vizualizat în cadrul rezultatelor;
- executabilul, aplicația MPI ce implementează o logică paralelizată;
- fișierele adiționale ce pot fi utilizate de executabil, precum fișiere input sau volume de date;
- configurarea nodurilor pe cluster, prin care utilizatorul trebuie să creeze o distribuție a proceselor pe nodurile puse la dispoziție. Această configurare determină unde vor fi create procesele aferente jobului MPI în cadrul clusterului. Numărul maxim de procese alocat unui job trebuie să coincidă cu suma distribuției proceselor în cadrul clusterului, pentru a evita utilizarea excesivă a resurselor;
- variabile de mediu destinate executabilului;
- parametri MPI utilizați în crearea jobului, precum maparea pe socket/node, suprasubscrierea, afișarea de date adiționale despre configurarea jobului, redistribuirea proceselor pe noduri mai puțin ocupate etc.

În urma creării configurației jobului, acesta, împreună cu datele sale aferente, va fi preluat și procesat de către backend. Acesta analizează configurarea, scanează executabilul pentru potențial malware, iar dacă stabilește că datele configurației sunt valide, jobul va fi marcat ca fiind pregătit pentru execuție, starea sa la nivel backend devenind *pending*.

Backendul urmărește în timp real valabilitatea nodurilor și ce joburi, procese și configurații se execută într-un moment dat în interiorul clusterului, iar de îndată ce stabilește disponibilitatea de a executa un job, alocă resurse separate dedicate execuției respective și așteaptă rezultatul de la cluster.

Supraincărcarea clusterului cu un număr prea mare de joburi la un moment dat, provenite de la unul sau mai mulți utilizatori este oprită prin gestionarea unor permisiuni asociate fiecărui cont din cadrul aplicației. Acest lucru este realizat de către un cont administrator, care stabilește pentru fiecare cont următoarele:

- numărul de procese pentru un job;
- numărul de joburi asociat unui utilizator;
- numărul de joburi pe un nod pentru un utilizator;
- timpul maxim de așteptare pentru execuția unui job;
- numărul maxim de joburi ce rulează pentru un utilizator;
- numărul maxim de joburi ce sunt în starea *pending* pentru un utilizator;
- etc.

De asemenea, administratorul dispunde de metrici ce pot fi modificate pentru a limita accesul unui utilizator in raport cu alți utilizatori.

Odată ce rezultatele produse de execuția unui job sunt agregate, acestea sunt puse la dispoziția utilizatorului. Acesta poate vizualiza toate datele aferente unui job într-o pagină dedicată și dispune de un set de acțiuni ce pot fi efectuate, precum: terminarea unui job, eliminarea din istoric, salvarea configurației, prelucrarea fișierelor asociate jobului, repornirea jobului etc.

De asemenea, utilizatorul poate vizualiza un istoric care conține distribuția proceselor pe noduri, utilizarea clusterului, documentația, sau poate solicita o creștere a resurselor alocate contului său. Administratorul poate modifica resursele fiecărui utilizator, precum și să le restricționeze permanent sau temporar accesul către cluster.

Principala complexitate a sistemului provine din caracterul asincron al gestionării joburilor, unde procesarea se face în fundal, iar statusul lor trebuie actualizat pe măsură ce

evoluează. Aplicația alocă un fir de execuție separat dedicat fiecărui job trimis spre execuție, pentru a aștepta rezultatele. Însă această abordare se poate dovedi deficitară odată cu scalarea aplicației și înregistrarea unui număr mare de joburi.

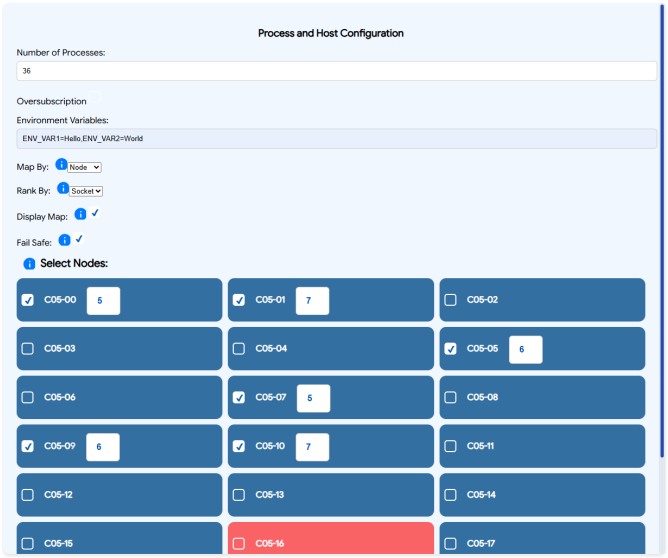

Fig. 7. Configurarea unui job în cadrul clusterului

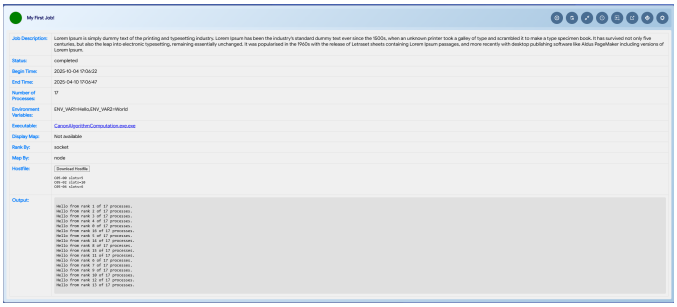

Fig. 8. Afișarea statusului jobului și a rezultatelor în timp real

Fig. 9. Istoric al joburilor

VI. Concluzii Preliminare

Conform celor prezentate anterior, aplicația propusă are ca obiectiv principal facilitarea gestionării taskurilor trimise spre execuție într-un cluster OpenMPI, prin intermediul unei interfețe grafice web. Astfel, aceasta oferă un cadru software eficient pentru interacțiunea cu infrastructura de calcul paralel, contribuind la optimizarea procesului de lucru și eliminând necesitatea configurării manuale inițiale a clusterului.

Avantajul central al aplicației constă în oferirea unui acces facil și direct la un cluster de calcul distribuit. În mod obișnuit, utilizarea OpenMPI într-un mediu de tip cluster presupune un proces complex și consumator de timp, care implică numeroase configurări tehnice prealabile, înainte ca utilizatorul să poată trimite joburi spre execuție. Aplicația propusă simplifică semnificativ această etapă, punând la dispoziție un cluster deja configurat, ceea ce accelerează vizualizarea rezultatelor generate de implementările ce utilizează cod paralelizat pe baza bibliotecii MPI.

Limitarea principală a aplicației este reprezentată de numărul restrâns de utilizatori care pot accesa simultan clusterul. Deși configurația hardware a nodurilor este una performantă, resursele disponibile nu permit procesarea unui volum de ordinul sutelor sau miilor de joburi în paralel. Astfel, scalarea aplicației presupune implicit și extinderea resurselor de calcul ale clusterului pentru a menține performanța în condiții de utilizare intensivă.

Din perspectiva aplicabilității, sistemul poate fi utilizat în contexte educaționale , în centre de cercetare sau în cadrul echipelor de dezvoltare care implementează algoritmi paraleli. Interfața, împreună cu mecanismele de abstractizare a interacțiunii cu infrastructura tehnică recomandă aplicația și utilizatorilor care nu dețin experiență avansată în lucrul cu medii de calcul distribuit.

Ca direcție de dezvoltare ulterioară, se propune extinderea procesului de configurare a unui job, astfel încât să fie acoperită o gamă mai largă de parametri și funcționalități specifice OpenMPI. În plus, se are în vedere creșterea complexității interacțiunii cu clusterul prin introducerea unui mecanism de comunicare mai granular, integrarea unor unelte specializate pentru analiza performanței execuției și diversificarea protocoalelor de comunicare suportate de aplicație.

Bibliografie

[1] Building Web-based Services for Practical Exercises in Parallel and Distributed Computing, J. C. Carretero, F. Garcia-Carballeira, and J. M. Perez, (2018),

[2] Teaching Parallel Programming in Containers: Virtualization of a Heterogeneous Local Infrastructure, M. Weigold and T. Arndt, (2023),

[3] A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard, Gropp, W., Lusk, E., Doss, N., Skjellum, A. (1996), *Parallel Computing*, 22(6), 789-828,

[4] The Scalable Process Topology Interface of MPI 2.2, Torsten Hoefler, Rolf Rabenseifner, Hubert Ritzdorf, Bronis R. de Supinski, Rajeev Thakur, Jesper Larsson Träff,

[5] MPI 4.1 Standard,

[6] Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation, G. Blinowski, A. Ojdowska, and A. Przybyłek (2022),

[7] Application of HATEOAS Principle in RESTful API Design, F. Aydemir and F. Basciftci, (2022),