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

Cîrjă Ioan

## 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 constă în 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 de diplomă constă în dezvoltarea unei aplicații care expune o interfață web pentru gestionarea taskurilor trimise spre execuție într-un cluster OpenMPI [9].

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 [8] – framework destinat implementării logicii de backend, alcătuită din microservicii Python.
- OpenMPI [9] – utilizat de către cluster pentru a permite paralelizarea și calculul distribuit pe unitățile de procesare.
- Docker [10] – utilizat pentru containerizarea aplicației și extindere.
- Angular [11] – framework utilizat pentru dezvoltarea interfeței utilizator.
- Python, JavaScript/TypeScript [12] – limbaje utilizate pentru logica backend, algoritmi și interogarea clusterului, respectiv pentru dezvoltarea interfeței utilizator.
- Firebase Real Time Database [13] – 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 [14] – 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.

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 ușor (exemplu: serviciul SSH nu necesită o conexiune la baza de date).

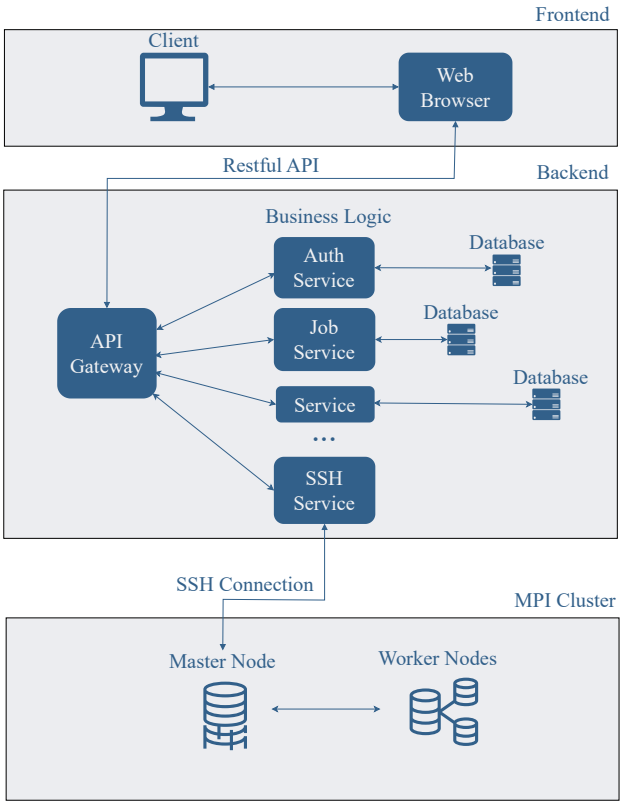


Fig. 1. Diagrama generală a aplicației

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.

## V. Activități Desfășurate până în prezent

În cadrul proiectului, au fost documentate și realizate următoarele activități:

### A. Realizarea diagramelor 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.

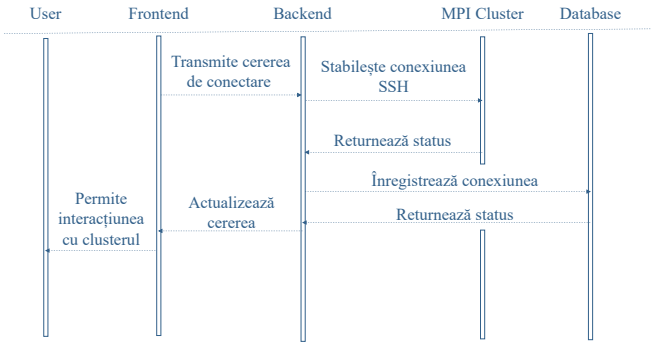


Fig. 2. Conectarea la cluster prin SSH.

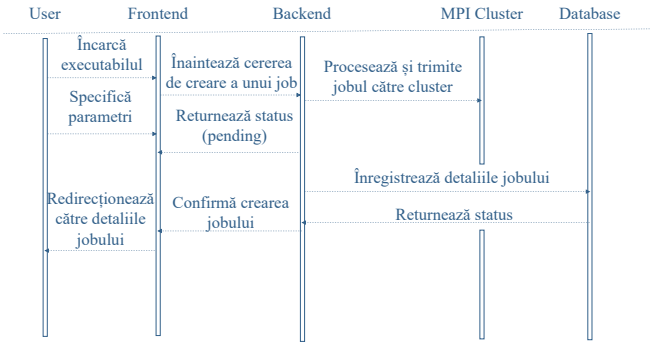


Fig. 3. Pașii pentru încărcarea unui job.

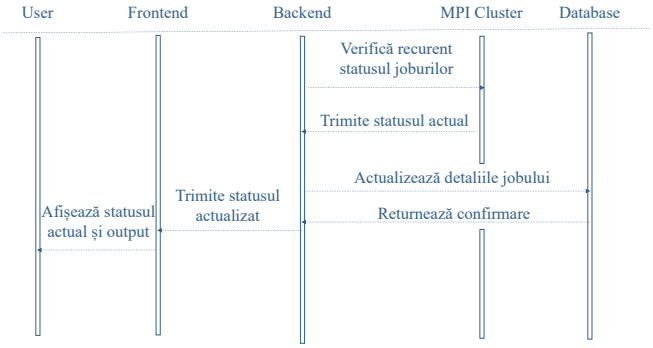


Fig. 4. Verificarea statusului joburilor.

### B. Dezvoltarea unui prototip wireframe

Au fost create wireframe-uri care ilustrează structura principalelor componente ale interfeței utilizatorului:

- **Formularul de încărcare a unui job** - se prezintă modul în care utilizatorii pot încărca fișiere și specifica setările necesare pentru execuția acestora.

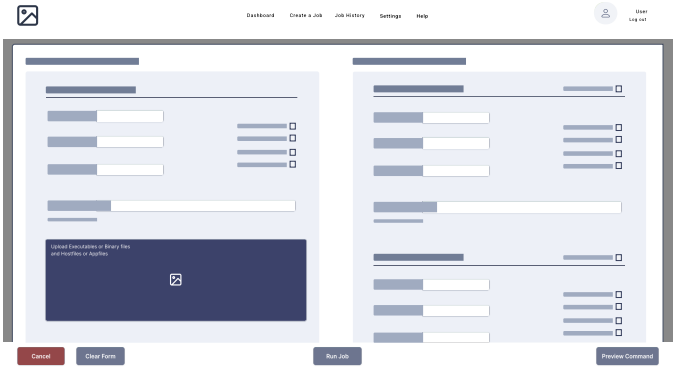


Fig. 5. Wireframe - încărcarea unui job.

- **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.

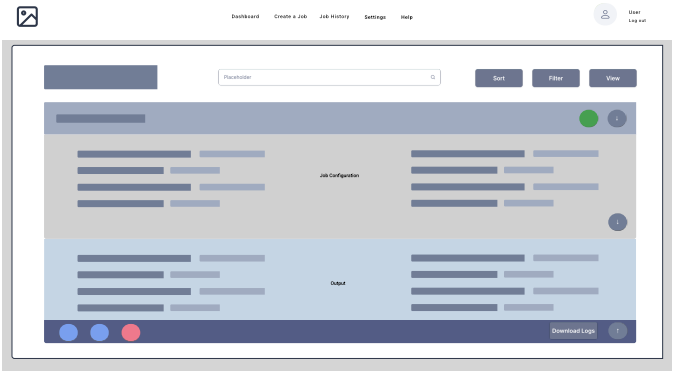


Fig. 6. Wireframe - pagina de status a unui job.

### C. Dezvoltarea unui prototip funcțional al aplicației

În cadrul dezvoltării prototipului funcțional al aplicației, s-au implementat următoarele componente și funcționalități principale:

- **Setarea structurii aplicației** - a fost definită structura fișierelor aplicației, împreună cu fișierele de configurare și conexiunile cu API-uri externe (VirusTotal pentru scanare de fișiere, configurarea conexiunii Firebase pentru persistența datelor).

- **Inițializarea microserviciilor esențiale** - au fost dezvoltate și configurate două microservicii principale, sub formă de prototip:
  - **Microserviciul de autentificare** - gestionează accesul la aplicație și identitatea.
  - **Microserviciul de încărcare a unui job** - permite utilizatorilor să încarce sarcini care vor fi transmise și executate de către cluster. Conține o componentă de procesare a cererilor de la utilizatori și o componentă care gestionează conexiunea SSH (crearea fișierelor, trimiterea comenzilor OpenMPI, interogarea nodurilor de calcul etc.).

- **Realizarea conexiunii cu clusterul MPI** - utilizând protocolul SSH pentru interacțiune bidirecțională, utilizatorii pot încărca un job, specificând o configurație minimală, cum ar fi:
  - Fișierul **hostfile** pentru definirea nodurilor utilizate.
  - Numărul de procese MPI ce vor fi lansate.
  - Activarea opțiunii de **oversubscription**, dacă este necesar.

- **Persistența datelor** - aplicația salvează statusul fiecărui job încărcat. Inițial, statusul este setat ca **pending**, iar ulterior este actualizat cu rezultatul comenzii de execuție.

- **Autentificare** - a fost realizat un modul de autentificare atât la nivel frontend, cât și backend. Utilizatorii își pot crea un cont, se pot loga și, în baza tokenului de securitate JWS, pot trimite un job către cluster.

- **Sistem de monitorizare a progresului joburilor** - în backend, actualizarea statusului joburilor se face prin intermediul unui **FastAPI background task** care rulează în mod asincron și așteaptă finalizarea execuției fiecărui job. Această abordare poate fi îmbunătățită în viitor prin utilizarea unor servicii care verifică periodic statusul joburilor și actualizează datele acestora.

- **Compatibilitate cu diverse tipuri de fișiere** - aplicația permite încărcarea fișierelor de tip **.cpp**, **.exe** sau **.bin**, care pot fi transmise către cluster. Fișierele **.exe** și **.bin** se vor executa automat, iar cele **.cpp** necesită o comandă adițională pentru compilare executată manual.

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. În stadiul actual, aplicația alocă un thread 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. Abordarea poate fi extinsă prin crearea unui job recurent care interoghează statusul tuturor joburilor care nu sunt finalizate.

## VI. Rezultate corelate cu obiectivele semestrului I

În cadrul proiectului, obiectivele descrise și realizate aferente semestrului I presupun stabilirea direcției de dezvoltare a aplicației, prin tehnologiile ce vor fi utilizate, resursele hardware, software, arhitectura aplicației și conectarea componentelor principale printr-o implementare de nivel prototip.

Prototipul implementat este capabil să îndeplinească una din funcționalitățile de bază ale aplicației, și anume trimiterea joburilor și monitorizarea statusului. Utilizatorul se poate înregistra, autentifica, poate trimite un job, specifica parametrii și verifica statusul acestuia odată ce va fi actualizat, dar are acces și la un istoric al joburilor. Abordarea microserviciilor și a conexiunii SSH produce rezultate favorabile, astfel încât timpul de execuție al unui job este în parametri normali, iar comunicarea HTTP este realizată.

Proiectul poate fi valorificat, în contextul tehnologiilor actuale, prin accesibilitatea pe care o oferă în comunicarea cu un sistem de calcul distribuit. În etapele viitoare, se dorește extinderea parametrilor asociați unui job, în funcție de comenzile și flagurile puse la dispoziție de OpenMPI, utilizat în cadrul proiectului, îmbunătățirea sistemului de monitorizare a clusterului, implementarea unui file browser pentru a vizualiza conținutul nodurilor de calcul, o gestiune mai detaliată a

joburilor, monitorizarea resurselor, retrimiterea joburilor și definitivarea interfeței utilizator.

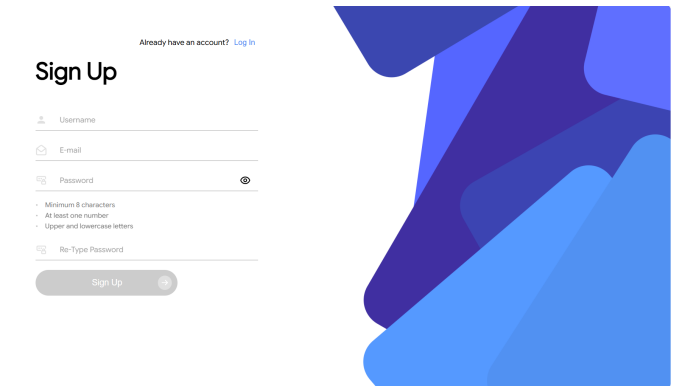


Fig. 7. Pagina de SignUp

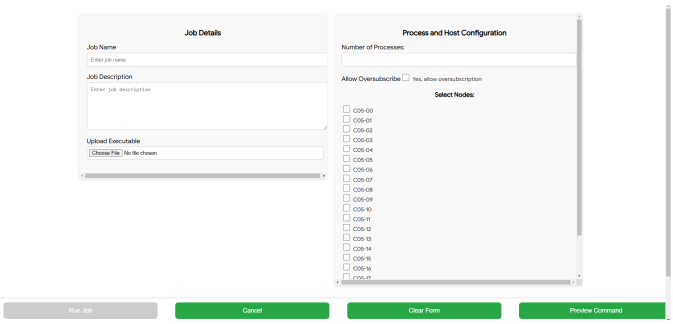


Fig. 8. Înregistrarea unui job pentru execuție

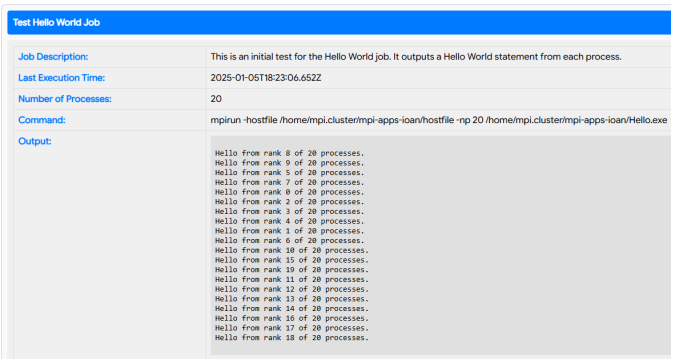


Fig. 9. Afișarea statusului jobului și a rezultatelor în timp real

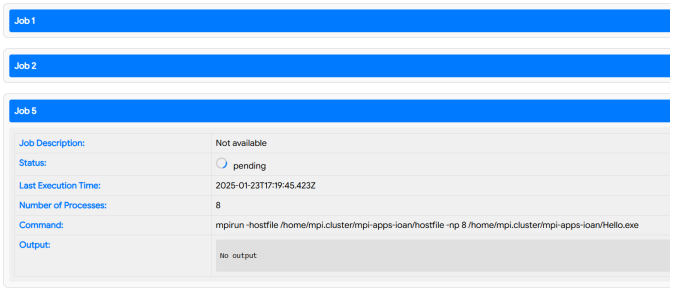


Fig. 10. Istoric al joburilor

## VII. Concluzii Preliminare

Conform celor menționate anterior, putem reitera faptul că aplicația are scopul de a gestiona taskurile trimise spre execuție într-un cluster OpenMPI și adresează unele dintre golurile identificate în soluțiile existente, precum lipsa interfețelor grafice accesibile, având potențialul de a reprezenta un produs software care oferă accesibilitate în interacțiunea cu un cluster OpenMPI. Arhitectura este bazată pe microservicii, iar comunicarea cu clusterul se va realiza prin SSH.

Prototipul prezintă una dintre funcționalitățile principale, și anume, trimiterea joburilor către cluster, în baza unui cont de utilizator, precum și vizualizarea rezultatelor în timp real.

Ca direcții viitoare, se va considera extinderea procesului de încărcare a unui job, acoperind o gamă mai largă de funcționalități OpenMPI și facilitând o comunicare mult mai complexă și granulară cu clusterul.

## 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),
[8] FastAPI,
[9] OpenMPI,
[10] Docker,
[11] Angular,
[12] Documentație limbaje Python și TypeScript,
[13] Firebase Real Time Database,
[14] PyCharm Professional și Visual Studio Code.