Virtualni klasteri računala

Cener, Luka

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:604398

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni prijediplomski studij Računarstvo

VIRTUALNI KLASTERI RAČUNALA

Završni rad

Luka Cener

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Luka Cener
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4628, 27.07.2021.
JMBAG:	0165090556
Mentor:	izv. prof. dr. sc. Zdravko Krpić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Virtualni klasteri računala
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Cilj rada je istražiti mogućnosti i prednosti izrade klastera virtualnih strojeva. U praktičnom dijelu rada upotrebom nekog od alata za virtualizaciju (VirtualBox, VMWare Workstation ili sl.) stvoriti klaster od 3 ili više virtualnih strojeva s pripadajućom mrežnom konfiguracijom i odgovarajućim alatima. Na navedenom klasteru implementirati osnovne barem dvije komunikacijske paradigme raspodiljenih računalnih sustava poput klijent-poslužitelj, peer-to-peer, paradigme dijeljene memorije, razmjene poruka i sl. Tema rezervirana za: Luka Cener

Datum prijedloga ocjene završnog rada od strane mentora:	19.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	26.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O IZVORNOSTI RADA

Osijek, 26.09.2024.

Ime i prezime Pristupnika:	Luka Cener
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4628, 27.07.2021.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Virtualni klasteri računala**

izrađen pod vodstvom mentora izv. prof. dr. sc. Zdravko Krpić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD **3**

1.1. Zadatak završnog rada **3**

2. PREGLED PODRUČJA **4**

3. RAČUNALNI KLASTERI **8**

3.1. Izrada okruženja virtualnog klastera računala **11**

3.1.1. Virtualizacija resursa 11

3.1.2. Mrežne postavke 13

4. PROGRAMSKI ALATI POTREBNI ZA RAD **17**

4.1. Secure Shell (SSH) **17**

4.2. Network File System (NFS) **19**

4.3. Message Passing Interface (MPI) **20**

4.4. ClusterSSH **22**

4.5. Ganglia **23**

5. VREDNOVANJE PROGRAMSKOG RJEŠENJA **29**

6. ZAKLJUČAK **37**

LITERATURA **39**

SAŽETAK **41**

ABSTRACT **42**

1. UVOD

U današnjem svijetu računarstva, sve veće količine podataka i složeniji projekti zahtijevaju sve veće računalne resurse. Ovisno o potrebama projekta, te zahtjeve mogu zadovoljiti tradicionalni ili virtualni klasteri računala. Tradicionalni klasteri računala, koji se sastoje od međusobno povezanih fizičkih računala, pružaju visoku razinu kontrole i optimizacije za specifična radna opterećenja. Međutim, u okruženjima gdje korisnik mijenja zahtjeve kao što su povećanje računalne snage ili skaliranje resursa, prilagodba tradicionalnih klastera računala može biti skupa i dugotrajna. U takvim se situacijama koriste virtualni klasteri računala, čije prilagođavanje specifičnim potrebama je olakšano.

Virtualni klasteri računala sastoje se od međusobno povezanih virtualnih računala koji se nalaze na jednom ili više fizičkih računala. Virtualizacija omogućava agregaciju resursa fizičkih računala i njihovo korištenje u obliku jednog virtualnog klastera računala s velikom snagom obrade podataka. Također se mogu dinamički skalirati dodavanjem i uklanjanjem virtualnih računala, što ih čini idealnim za promjenjiva radna opterećenja. Osim toga, virtualni se klasteri mogu lako premjestiti između različitih fizičkih lokacija (računala), što ih čini fleksibilnim rješenjem za poduzeća s više lokacija. Navedene kvalitete su razlog zašto su virtualni klasteri računala postali popularni u raznim područjima, uključujući znanstvena istraživanja i visoko obrazovanje.

Ovaj se završni rad sastoji od 6 poglavlja. U drugom su poglavlju predstavljeni primjeri računalnih klastera te njihovo postavljanje. U trećem su poglavlju objašnjene temeljne postavke računalnog klastera u sklopu ovoga rada. U četvrtom su poglavlju prikazani alati potrebni za učinkovit rad računalnog klastera. Poglavlje pet prikazuje testiranje računalnog klastera te su vrednovani rezultati dobiveni testiranjem. Šesto je poglavlje zaključak.

1.1. Zadatak završnog rada

Motivirani rastućim potrebama za računalnim resursima i ograničenjima tradicionalnih računalnih klastera, ovaj završni rad ima za cilj implementaciju i testiranje virtualnog klastera računala. Zadatak je ovog rada izgraditi funkcionalan virtualni klaster računala inspiriran poznatim primjerima, fokusirajući se na praktične aspekte implementacije i upravljanja. To uključuje korištenje alata za virtualizaciju i upravljanje resursima te testiranje performansi i učinkovitosti.

2. PREGLED PODRUČJA

U postojećim radovima koji se bave izradom virtualnih klastera računala može se uočiti da se ovi sustavi koriste u različite svrhe, uključujući znanstvena istraživanja, obrazovne svrhe te industrijske aplikacije. Autori koriste različite hipervizore zbog potrebe za fleksibilnošću i optimizacijom resursa, omogućujući bolje upravljanje skaliranjem i dinamičkom alokacijom resursa. Različiti programski alati, poput MPI (*Message Passing Interface*) knjižnica i mrežnih protokola, često se koriste radi poboljšanja paralelne obrade i učinkovite komunikacije između čvorova unutar klastera računala. U ovom poglavlju bit će predstavljeni radovi koji su slično koncipirani kao i ovaj rad, s posebnim naglaskom na korištenje različitih tehnologija, alata i metoda za postizanje funkcionalnih virtualnih klastera računala

U [1], za izradu virtualnog klastera računala korišteno je računalo s 4-jezgrenim Intel i7 procesorom. Virtualizacija resursa izvedena je korištenjem VirtualBoxa. Virtualni klaster računala sastoji se od prednjeg čvora (engl. *front-end node*) i nekoliko računalnih čvorova (engl. *compute nodes*), svi implementirani kao virtualna računala unutar VirtualBox alata. Mrežno povezivanje i raspodjela zadataka na računalne čvorove upravljani su prednjim čvorom, a komunikacija između čvorova ostvarena je putem privatne mreže. Svi čvorovi koriste PelicanHPC operacijski sustav, baziran na Debian GNU/Linux distribuciji. PelicanHPC omogućuje paralelno računanje koristeći MPI knjižnice. Koriste se dvije MPI implementacije: LAM/MPI i OpenMPI, omogućujući izvođenje paralelnih programa pisanih u jezicima poput C, C++ i Fortran. U procesu su korišteni i alati poput iperf za testiranje mrežne propusnosti i htop za nadzor resursa. Testiranje virtualnog klastera računala obavljeno je pomoću LINPACK mjerila (engl. *benchmark*), koji mjeri performanse u rješavanju linearnih jednadžbi. Postignuta je najveća brzina od 5.383 milijardi operacija s pomičnim zarezom po sekundi (engl. *Floating-point Operations Per Second, FLOPS*).

U navedenom radu korišten je VirtualBox za virtualizaciju resursa, pri čemu je klaster sastavljen od prednjeg čvora i nekoliko računskih čvorova. Ovaj pristup je sličan rješenju u ovom radu jer je također korišten VirtualBox za kreiranje virtualnih čvorova. Razlika se očituje u tome što su u ovom radu primijenjene novije verzije operacijskih sustava i drugačije mrežne postavke radi poboljšanja komunikacije među čvorovima. Također, rad je poslužio kao motivacija za odabir MPI-ja kao alata za paralelno računarstvo, zbog njegove široke primjene i fleksibilnosti.

Izgradnja virtualnog klastera računala opisana u [2] temelji se na korištenju Amazon Elastic Compute Cloud (EC2) infrastrukture za izradu besplatnog računalnog klastera visokih performansi (engl. *High Performance Computing, HPC*). Za automatizaciju procesa postavljanja, konfiguracije

i upravljanja klasterima računala na Amazon EC2 korišten je StarCluster alat, razvijen na MIT-u. Virtualni klaster računala sastavljen je od 8 t2.micro čvorova, koje Amazon besplatno nudi u trajanju od 12 mjeseci, s ograničenjem od 750 sati mjesečno. U svrhu testiranja performansi u većoj konfiguraciji, virtualni računalni klaster proširen je na 16 t2.micro čvorova koristeći plaćene resurse. Svaki t2.micro čvor opremljen je jednim virtualnim procesorom (vCPU) i 1 GB RAM-a, a koristi Linux operacijski sustav.

Ovaj virtualni klaster računala testiran je pomoću LINPACK mjerila, pri čemu je postignuta maksimalna brzina obrade od 94,37 GFLOPS-a. Ostali testovi, poput HPCC (*HPC Challenge*) mjerila, pokazali su da je virtualni računalni klaster sposoban za obradu velikih količina podataka, iako su uočena određena ograničenja performansi mreže zbog niže brzine prijenosa podataka između čvorova. Korišteni su i alati kao što su iperf za testiranje mrežnih performansi te MPI knjižnice za paralelnu obradu zadataka. Ovaj virtualni klaster računala može se primijeniti u znanstvenim istraživanjima, posebno za razvoj i testiranje prototipova, a kasnije se može nadograditi na snažniji sustav za složenije simulacije i zahtjevnije računalne zadatke.

U opisanom radu je korišten Amazon EC2 kao platforma za izradu besplatnog HPC klastera, što se razlikuje od rješenja u ovom radu, jer klaster računala funkcionira lokalno, bez korištenja infrastrukture u oblaku. Međutim, ideja o automatizaciji postavljanja i upravljanja klasterima računala motivirala je integraciju automatiziranih procesa unutar lokalnog okruženja ovog rada, čime se olakšalo upravljanje i testiranje virtualnog klastera računala.

U [3] izgrađen je virtualni klaster računala pomoću tehnologija OpenNebula i KVM. Osnovna fizička struktura sastavljena je od recikliranih stolnih računala Dell OptiPlex 745 s procesorima Intel Core 2 Duo i 2 GB RAM-a. Na svakom fizičkom računalu instaliran je KVM (*Kernel-based Virtual Machine*) alat za virtualizaciju resursa, koji omogućuje pokretanje i upravljanje virtualnim računalima. Fizička računala koriste se samo kao domaćini za virtualna računala. Za upravljanje svim aspektima virtualnog klastera računala koristi se OpenNebula. Također se koriste alati MPI i TORQUE, koji omogućuju paralelnu obradu i zakazivanje zadataka unutar virtualnog klastera računala. Virtualni klaster računala namijenjen je edukacijskim i istraživačkim svrhama, pružajući studentima priliku da postavljaju, konfiguriraju i testiraju virtualni HPC klaster. Testiranje klastera računala provedeno je pomoću OSU Micro-Benchmarks, gdje su vrednovane performanse u radu s paralelnim aplikacijama, čineći ovaj klaster računala prikladnim za simulacije, obuku i razvoj u području računarstva visokih performansi.

U navedenom radu se koristi OpenNebula i KVM tehnologija za izgradnju virtualnog klastera računala. Sličnost s ovim rješenjem očituje se u korištenju besplatnih i otvorenih alata za virtualizaciju. Ovaj rad također je utjecao na implementaciju alata za upravljanje zadacima, kao što je ClusterSSH, kako bi se olakšalo upravljanje klasterom računala.

U [4], za izgradnju virtualnog klastera računala korišten je VMware ESXi alat za virtualizaciju resursa, uz korištenje PCI prolazne (engl. *passthrough*) tehnologije, koja omogućava virtualnim računalima direktan pristup fizičkim grafičkim procesorima. Virtualni klaster računala sastavljen je od dva glavna poslužitelja, svaki opremljen Intel Xeon E5506 procesorom, 32 GB RAM-a, Tesla K20 grafičkom karticom te podrškom za InfiniBand i 10 Gb Ethernet kao prijenosne medije za komunikaciju među čvorovima.

Operacijski sustav na svim poslužiteljima je CentOS, dok su virtualna računala također bazirana na CentOS operacijskom sustavu. Ovaj virtualni klaster računala koristi se CUDA tehnologijom za paralelno računanje na grafičkim procesorima te MPI-jem za raspodjelu zadataka između čvorova. Uz te alate, korišteni su i iperf za testiranje mrežnih performansi te različiti CUDA SDK alati, kao što su alignedTypes, asyncAPI, BlackScholes i matrixMul, za mjerenje performansi grafičkog procesora. Virtualni klaster računala testiran je pomoću HPL (*High Performance Linpack*) mjerila kako bi se procijenile performanse u različitim konfiguracijama prijenosnih medija (1 Gb Ethernet, 10 Gb Ethernet, InfiniBand).

Namjena ovog virtualnog klastera računala je optimizacija računalno zahtjevnih simulacija i složenih znanstvenih izračuna, osobito onih koje se oslanjaju na masovno paralelno procesiranje. Testovima je pokazano kako 10 Gb Ethernet nudi bolju učinkovitost od InfiniBanda u virtualiziranom okruženju, dok PCI prolazna tehnologija omogućuje performanse bliske fizičkom grafičkom procesoru.

U radu u kojem je primijenjen VMware ESXi s PCI prolaznom tehnologijom, naglasak je stavljen na obradu pomoću grafičkog procesora, što se razlikuje od rješenja u ovom radu, koji se fokusira isključivo na obradu putem procesora. Sličnost se ogleda u korištenju MPI-ja za raspodjelu zadataka među čvorovima.

Izgradnja virtualnog HPC klastera u [5] temelji se na korištenju Docker tehnologije za rješavanje problema softverske ovisnosti u HPC okruženju, omogućujući izolaciju softverskog okruženja i fleksibilnu izgradnju računalnog klastera neovisno o fizičkoj infrastrukturi. Računalni se klaster sastoji od glavnog čvora i 2 čvora za obradu. Svaki je čvor smješten na fizičkom poslužitelju s Intel Xeon E5-2630 procesorom, 64 GB RAM-a i SAS diskom od 146 GB, na kojima su

implementirani Docker kontejneri. Operacijski sustavi na fizičkim poslužiteljima su CentOS 7.1, dok se kontejneri temelje na CentOS 6.7 operacijskom sustavu s instaliranim OpenMPI knjižnicama za paralelnu obradu. Za upravljanje komunikacijom i automatsko otkrivanje resursa korišten je Consul poslužitelj, koji omogućuje dinamičko skaliranje i automatsko otkrivanje usluga. Dodatni korišteni alati su Docker Engine za upravljanje kontejnerima, te Consul-template za dinamičko ažuriranje konfiguracijskih datoteka temeljeno na uslugama registriranim u alatu Consul. Za testiranje mrežne propusnosti među čvorovima je korišten iperf alat. Komunikacija između kontejnera ostvarena je putem prilagođenih mrežnih mostova, omogućujući direktnu povezanost bez potrebe za NAT prijevodom. Ovaj je virtualni HPC klaster namijenjen optimizaciji znanstvenih simulacija i složenih izračuna, kao što su simulacije dinamike fluida i molekularne dinamike. Izvedene simulacije, poput MPI zadataka, demonstrirale su njegovu sposobnost obrade velikih količina podataka, čime utvrđuje pogodnost za primjenu u znanosti.

Docker tehnologija, koja je korištena za izgradnju HPC klastera u opisanom radu, usporediva je s rješenjem u ovom radu zbog fokusiranja na izolaciju softverskih okruženja. Iako je u ovom radu primijenjena tradicionalnija virtualizacija resursa putem VirtualBoxa, Dockerov pristup, koji omogućuje lako skaliranje i izolaciju aplikacija, potaknuo je razmatranje uvođenja kontejnera u budućnosti, posebno u svrhu lakšeg upravljanja ovisnostima između aplikacija u virtualnom klasteru računala.

Svi navedeni radovi pokazuju kako su virtualni klasteri računala korišteni u različitim kontekstima, od znanstvenih istraživanja do industrijskih simulacija, te koriste širok raspon tehnologija za virtualizaciju resursa, poput VirtualBoxa, OpenNebule, VMware ESXi-a i Dockera. Korištenje MPI-ja za paralelno računanje ističe se kao standard u raspodjelu zadataka među čvorovima. Glavne prednosti virtualnih klastera računala su njihova mogućnost dinamičkog prilagođavanja resursa opterećenju i jednostavnije skaliranje, dok su ključni izazovi vezani uz složenost postavljanja virtualnih klastera računala. Iako se u ovom radu obrađuje manji i jednostavniji virtualni klaster računala, spominjanje masivnijih virtualnih klastera računala važno je kako bi se ukazalo na potencijal skalabilnosti i šire primjene ovakvih sustava. Na ovaj način, prikazuje se kako se uz ograničene resurse može implementirati funkcionalan virtualni klaster računala, koji može poslužiti kao temelj za daljnje proširenje i optimizaciju.

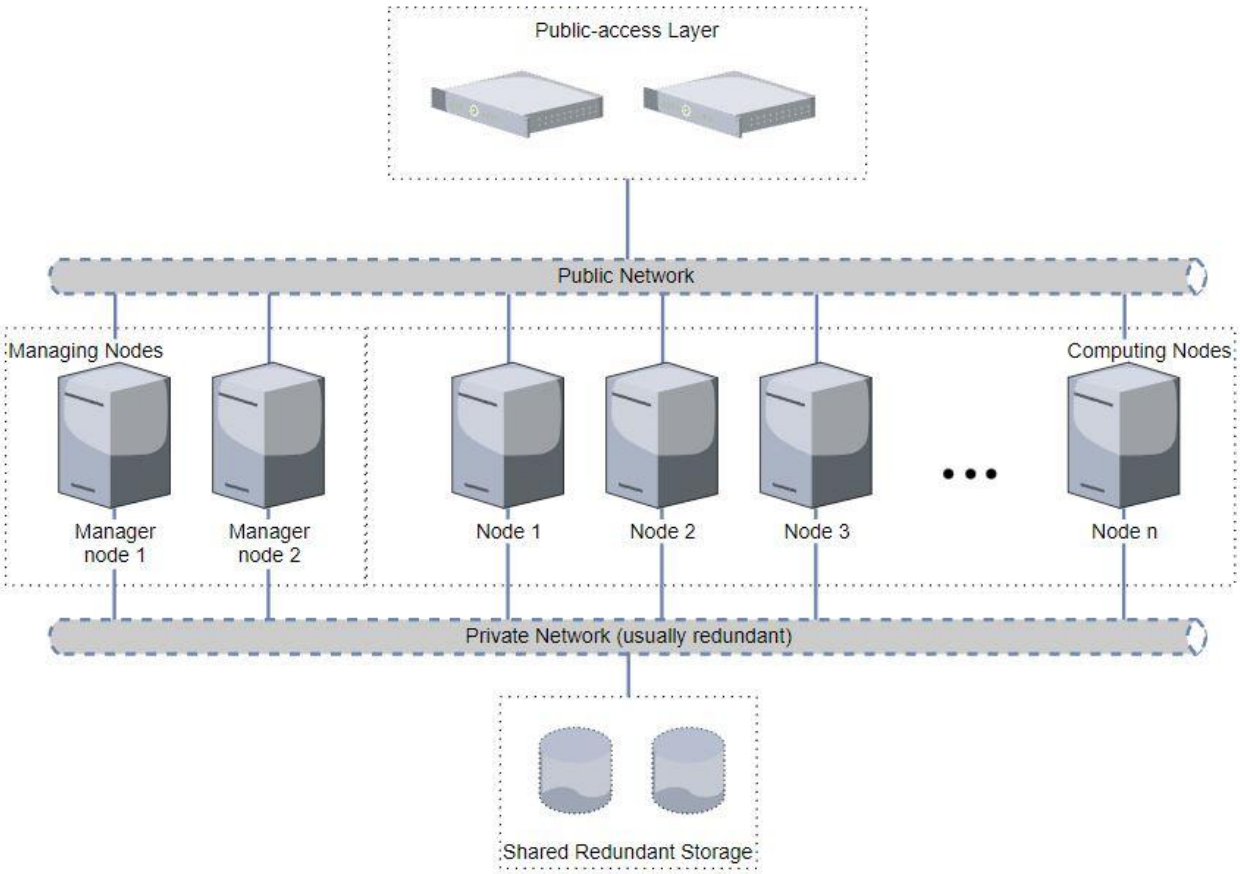
3. RAČUNALNI KLASTERI

Prema [6], računalni je klaster skup povezanih računala koji se promatra kao jedinstveni sustav. Svako računalo u klasteru računala se promatra kao čvor. Čvorovi su međusobno povezani putem brzih lokalnih mreža te svaki čvor pokreće svoju instancu operacijskog sustava. Uglavnom svaki čvor u računalnom klasteru koristi isti hardver i operacijski sustav, ali to nije nužno uvijek tako. Glavna svrha klastera računala je poboljšanje performansi i osiguravanje neprekidne dostupnosti usluga, u usporedbi s pojedinačnim računalima, uz značajno niže troškove. Imaju širok raspon primjenjivosti i implementacije, u rasponu od malih poslovnih klastera računala s nekoliko čvorova do najbržih superračunala na svijetu.

Računalni klasteri mogu se kategorizirati na sljedeći način:

- **Računalni klasteri visokih performansi** (engl. *High Performance Computing, HPC*): Ovi klasteri računala dizajnirani su za rješavanje složenih računskih zadataka koji zahtijevaju veliku računalnu snagu. Najčešće se koriste u znanstvenim istraživanjima, simulacijama, modeliranju i drugim aplikacijama koje zahtijevaju brzu obradu velike količine podataka.
- **Računalni klasteri za ravnomjerno opterećenje** (engl. *Load Balancing, LB*): Ovaj tip klastera računala ravnomjerno raspodjeljuje zadatke između čvorova, čime se izbjegava preopterećenje pojedinih čvorova. To omogućuje bolje iskorištavanje resursa i poboljšava performanse sustava.
- **Računalni klasteri s visokom dostupnošću** (engl. *High Availability, HA*): Ovi klasteri računala osiguravaju neprekidan rad sustava i maksimalnu dostupnost usluga, čak i u slučaju kvarova. Ako jedan čvor zakaže, ostali preuzimaju njegovu funkciju kako bi se izbjegli prekidi u radu.
- **Računalni klasteri s visokom propusnošću** (engl. *High Throughput, HT*): Ova vrsta klastera računala fokusira se na izvršavanje velikog broja manjih zadataka tijekom vremena. Optimizirani su za paralelno izvođenje velikog broja nezavisnih zadataka.
- **Hibridni računalni klasteri**: Ovi klasteri računala kombiniraju različite specijalizirane uređaje, poput matematičkih koprocesora, grafičkih obradbenih jedinica (GPU) ili drugih prilagođenih hardverskih komponenti. Koriste se za specifične zadatke koji zahtijevaju posebne resurse.

Osnovna je arhitektura računalnog klastera prikazana na slici 3.1.



Slika 3.1. Shematski prikaz osnovne arhitekture računalnog klastera[1]

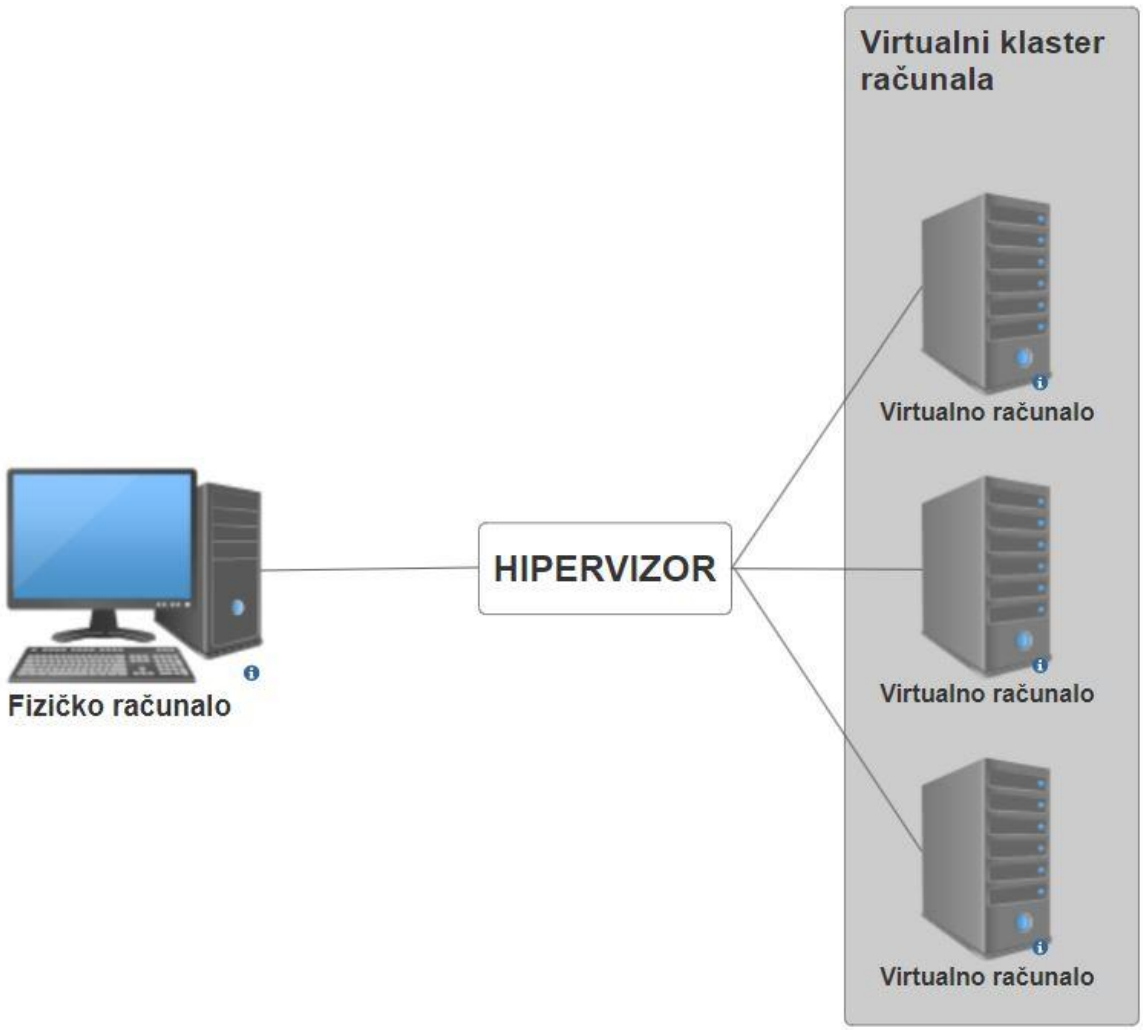
Iz slike se mogu vidjeti glavne komponente osnovne arhitekture računalnog klastera, a to su:

- **Radnički čvorovi** (engl. *Computing Nodes*): Obrađuju korisničke zadatke i mogu varirati od običnih stolnih do visokokvalitetnih poslužitelja.
- **Upravljački čvorovi** (engl. *Managing Nodes*): Nadziru rad klastera računala, te se mogu konfigurirati da rade na radničkim čvorovima radi optimizacije resursa.
- **Privatna mreža** (engl. *Private Network*): Omogućuje komunikaciju između čvorova.
- **Zajedničko spremište** (engl. *Shared Redundant Storage*): Omogućuje pristup podacima svim čvorovima.

[1]Computer Clusters, Types, Uses and Applications, dostupno na: https://www.baeldung.com/cs/computer-clusters-types [14.9.2024.]

- **Javni sloj pristupa** (engl. *Public-access layer*): Virtualizira pristup klasteru računala tako da izgleda kao jedinstveni sustav, raspodjeljujući zahtjeve prema aktivnim čvorovima.

Osim na tradicionalne računalne klastere, važno je osvrnuti se na razvoj modernih tehnologija koje unaprjeđuju ove sustave. Pojam virtualizacije donijela je značajne promjene u načinu na koji se klasteri računala dizajniraju i koriste. Arhitektura virtualnog klastera računala sastoji se od fizičkog računala na kojem je instaliran hipervizor, softver koji omogućuje stvaranje i upravljanje virtualnim računalima. Hipervizor omogućuje da jedno fizičko računalo pokreće više virtualnih strojeva, od kojih svaki može imati vlastiti operacijski sustav i aplikacije. Čvorovi u virtualnom klasteru računala međusobno komuniciraju putem mreže, omogućujući razmjenu podataka i koordinaciju resursa. Slika 3.2. prikazuje arhitekturu virtualnog klastera računala.



Slika 3.2. Arhitektura virtualnog klastera računala

Prednosti virtualnih klastera računala uključuju bolju fleksibilnost, učinkovitije korištenje resursa i jednostavniju skalabilnost, dok omogućuju izolaciju i lakšu replikaciju okruženja. Specifičnost

je virtualnih klastera računala njihova sposobnost brze prilagodbe, što olakšava migraciju resursa ili oporavak od grešaka bez potrebe za fizičkim promjenama.

3.1. Izrada okruženja virtualnog klastera računala

Za izradu virtualnog klastera računala u okviru ovog rada, koristilo se Lenovo IdeaPad 3 fizičko računalo čije su specifikacije prikazane u tablici 3.3.

Tablica 3.3. Specifikacije fizičkog računala

Fizičko računalo	*Lenovo IdeaPad 3*
Procesor	4-jezgreni AMD Ryzen 7 3700U s radnim taktom od 2.3 Ghz
Memorija (RAM)	8 GB
Pohrana	PCIe NVMe SSD kapaciteta 512 GB
Grafička kartica	AMD Radeon RX Vega 10 Graphics
Operacijski sustav	Windows 10 Pro

U nastavku su opisane i podešene dvije glavne konfiguracije, a to su virtualizacija resursa i mrežne postavke. Nakon što su podešene te konfiguracije, virtualnost virtualnog klastera računala prestaje i nadalje se s njim može ophoditi kao s tradicionalnim klasterom računala.

3.1.1. Virtualizacija resursa

S obzirom da se virtualni klasteri računala sastoje od virtualnih računala, potreban je alat za virtualizaciju resursa. Ovaj alat omogućuje simulaciju kompletnih računala unutar hardvera jednog fizičkog računala. Postoje brojni programi za virtualizaciju resursa, a najpopularniji su:

- **VMware Workstation**: Pruža napredne funkcije i stabilnost za kreiranje virtualnih okruženja, često se koristi u velikim IT korporacijama i laboratorijima za razvoj softvera.
- **Oracle VM VirtualBox**: Besplatan i jednostavan za korištenje, koristan za studente i programere za testiranje aplikacija u različitim operacijskim sustavima
- **Microsoft Hyper-V**: Integriran u Windows-u, omogućuje stvaranje virtualnih računala na različitim platformama, a često se koristi u poslovnim mrežama za testiranje

- **KVM** (*Kernel-based Virtual Machine*): Otvoreni softver integriran u Linux-u, vrlo je učinkovit i koristi se u računalnom oblaku (engl. *Computing cloud*) za održavanje više poslužitelja.

Za izradu virtualnih računala u sklopu ovog rada korišten je program VirtualBox. VirtualBox daje najbolje rješenje za izradu virtualnih klastera računala što se može vidjeti u [7]. Objašnjeno u [8], VirtualBox je popularan alat za virtualizaciju jer omogućuje korisnicima pokretanje više operacijskih sustava istovremeno na jednom računalu. Ovaj alat nudi korisnički prijateljsko sučelje, visoku razinu fleksibilnosti i podršku za širok raspon operacijskih sustava, što ga čini idealnim za potrebe virtualizacije resursa.

U ovom radu izradit će se tri virtualna računala od kojih će jedan biti glavni čvor (engl. *master node*), a preostala dva će biti radnički čvorovi (engl. *worker nodes*). Ova tri virtualna računala čine minimalan skup računala potreban za smislen i funkcionalan klaster računala. U tablici 3.4. prikazane su specifikacije za sva tri računala.

Tablica 3.4. Specifikacije virtualnih računala

	Glavni čvor	*Radnički čvor 1*	*Radnički čvor 2*
Procesori	3	2	2
Memorija (RAM)	3-GB	2-GB	2-GB
Operacijski sustav	Ubuntu (64-bit)	Ubuntu (64-bit)	Ubuntu (64-bit)
Ime hosta	master	worker1	worker2

Sva tri računala služit će se Linux operacijskim sustavom, Ubuntu distribucijom temeljenom na Debian-u. Za master čvor koristit će se Ubuntu 24.04 LTS jer je to najnovija Ubuntu desktop verzija. Omogućuje lakši rad s grafičkim alatima i drugim aplikacijama za nadzor i administraciju cijele mreže čvorova. Za oba radna čvora biti će korištena minimalna verzija Ubuntu 18.04. Radnim čvorovima nije potrebno grafičko ili korisničko sučelje, već su fokusirani isključivo na izvršavanje zadatka koje im dodjeljuje master čvor. Prema [9], Ubuntu je najčešće korištena platforma za radne stanice na svijetu. Razlozi njezine popularnosti su sljedeći:

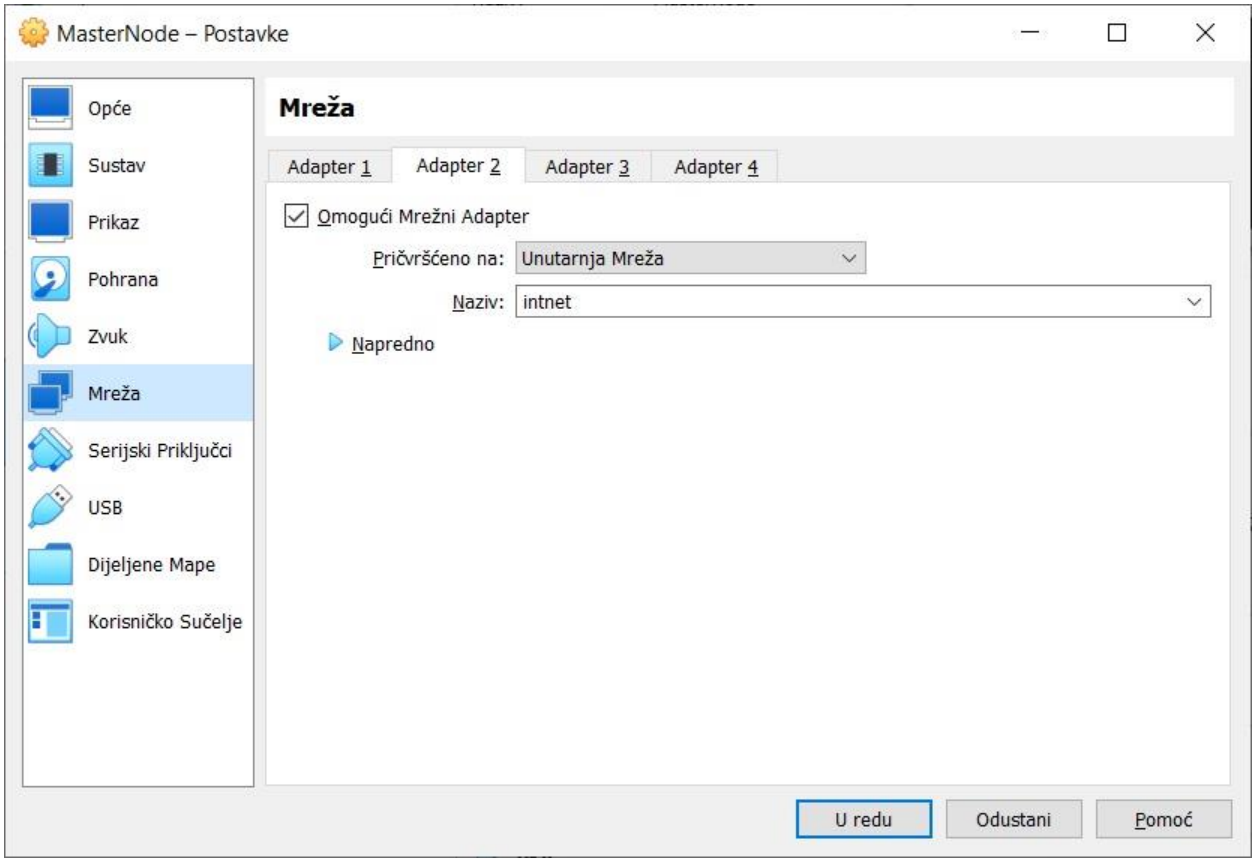
- Sigurnost: Ubuntu se redovito ažurira sigurnosnim ispravkama i nudi niz alata za zaštitu podataka i sustava.

- Lagana izvedba: Besplatan je, te se može preuzeti s njihove službene stranice. Ne zahtijeva puno resursa te radi glatko na uređajima niske razine.
- Prilagođenost prema korisniku: Ubuntu nudi jednostavno sučelje, te velik broj aplikacija od kojih su neke dostupne isključivo za ovaj operacijski sustav.

3.1.2. Mrežne postavke

Nakon virtualizacije resursa potrebna je konfiguracija mrežnih postavki kako bi virtualna računala mogla komunicirati međusobno. Umrežavanje virtualnih računala izvesti će se pomoću virtualne mrežne kartice. Virtualna se mrežna kartica koristi u virtualiziranim okruženjima i klasterima računala kako bi omogućila virtualnim računalima povezivanje na mrežu i međusobnu komunikaciju. Funkcioniraju kao mrežna kartica unutar fizičkog računala, ali su u potpunosti softverski definirani.

Za početak potrebna je definicija mrežne kartice. Mrežna se kartica brine za priključivanje računala na lokalnu mrežu preko koje će komunicirati. Kroz opciju „Mreža“ u postavkama virtualnog računala potrebno je omogućiti novi mrežni adapter. Slika 3.5. prikazuje potreban izgled postavki, te će se iste podesiti na sva tri računala.



Slika 3.5. Definiranje mrežne kartice

Nakon omogućavanje mrežne kartice, slijedi konfiguracija mreže putem naredbenog retka. Na master čvoru potrebno je pokrenuti terminal i izvršiti naredbe prikazane na slici 3.6.

```
$ sudo apt install net-tools
$ ifconfig
```

Slika 3.6. Naredbe za instalaciju alata „net-tools“ i prikaz mrežnih konfiguracija

Naredbom „sudo apt install net-tools“ instaliran skup alata za mrežnu naredbu i dijagnostiku. Ovi alati korisni su za pregled i upravljanje mrežnim postavkama. Jedan je od tih alata iskorišten u sljedećoj naredbi. Naredba „ifconfig“ prikazuje informacije o mrežnim karticama u sustavu.

Izvršenjem naredbe, prikazane su informacije o dvjema mrežnim karticama:

- **enp0s3**: Ova je mrežna kartica konfigurirana za pristup internetu
- **enp0s8**: Ova mrežna kartica još nema definirane postavke

Postavke mrežnih kartica konfiguriraju se kroz datoteku „/etc/network/interfaces“. Ova datoteka omogućuje definiranje IP adresa, usmjeravanje, IP maskiranja, zadanih ruta i drugih mrežnih parametara za svaku mrežnu karticu na virtualnom računalu. Unutar datoteke će se definirati postavke mrežne kartice prikazane na slici 3.7.

```
auto enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
address 192.168.100.100
netmask 255.255.255.0
network 192.168.100.0
broadcast 192.168.100.255
```

Slika 3.7. Datoteka „/etc/network/interfaces“

Naredbom „auto enp0s3“ mrežni adapter „enp0s3“ je konfiguriran da se automatski aktivira pri pokretanju sustava. IP adresa se automatski dodjeljuje putem DHCP protokola, prema naredbi „iface enp0s3 inet dhcp“. Mrežni adapter „enp0s8“ je također postavljen da se automatski aktivira pri pokretanju, ali mu je dodijeljena statička IP adresa. U statičkoj konfiguraciji su specificirane sljedeće postavke:

- „address 192.168.100.100“: Dodjeljuje IP adresu čvora
- „netmask 255.255.255.0“: Definira mrežnu masku

- „network 192.168.100.0“: Određuje mrežu kojoj mrežni adapter pripada
- „broadcast 192.168.100.255“: Postavlja „broadcast“ adresu za komunikaciju unutar mreže

Na taj način, mrežni adapter „enp0s3“ je konfiguriran za dobivanje dinamičke IP adrese, dok je mrežni adapter „enp0s8“ ručno konfiguriran sa statičkom IP adresom unutar lokalne mreže.

Na sličan će se način definirati postavke mrežnih kartica kod radnih čvorova gdje će jedina razlika biti drugačija IP adresa za sučelje za komunikaciju. Na master čvoru, IP adresa je „192.168.100.100“. U nastavku su prikazane IP adrese za radne čvorove.

- IP adresa worker1 čvora:
  `address 192.168.100.101`
- IP adresa worker2 čvora:
  `address 192.168.100.102`

Nakon što su unesene promjene u datoteku te promjene neće automatski stupiti na snagu. Kako bi osigurali pravilnu primjenu mrežnih pravila i konfiguracije, potrebno je ponovno pokrenuti „network daemon“. Naredba za ponovno pokretanje „network daemon“ prikazana je na slici 3.8.

```
$ sudo systemctl restart systemd-networkd
```

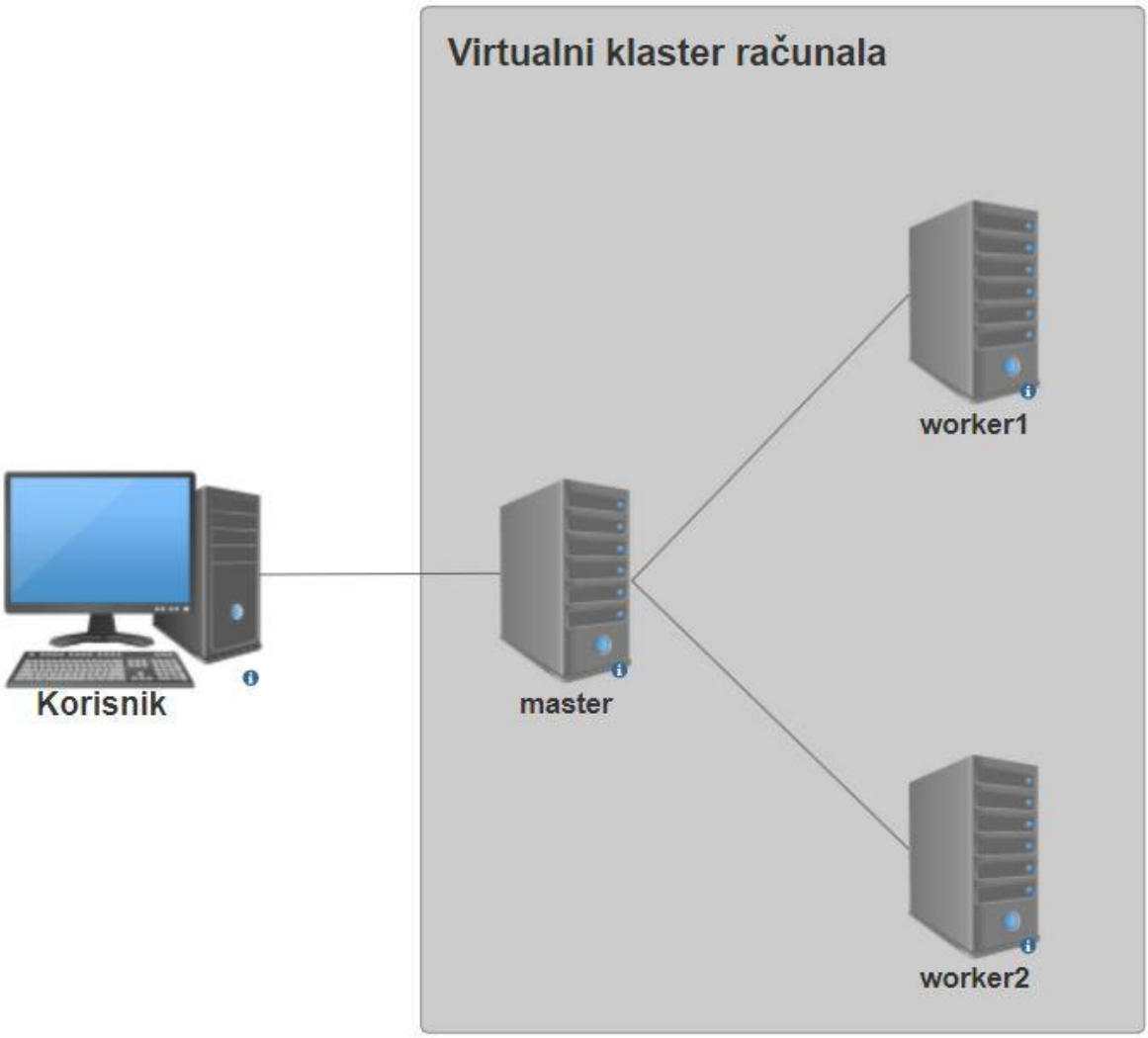
Slika 3.8. Naredbe za ponovno pokretanje „network daemon“

Kako bi bilo osigurano da su nove postavke primijenjene, može se izvršiti naredba prikazana na slici 3.9. Omogućuje provjeru trenutnog stanja mrežnih adaptera i njihovih postavki.

```
$ ip a
```

Slika 3.9. Naredbe za provjeru stanja mrežnih adaptera

Temeljne postavke su podešene, te je klaster računala spreman za daljnji razvoj i implementaciju klasterskih aplikacija. Virtualni se klaster računala u sklopu ovog rada nadalje može smatrati kao tradicionalni klaster računala.



Slika 3.10. Arhitektura virtualnog klastera računala u okviru ovog rada

Slika 3.10. prikazuje arhitekturu virtualnog klastera računala u okviru ovog rada. Na korisnikovom računalu se nalazi hipervizor, točnije VirtualBox, pomoću kojeg su stvoreni svi virtualni čvorovi. Linija koja povezuje korisnika s master čvorom označuje vezu putem koje korisnik šalje zahtjeve glavnom čvoru. Zatim glavni čvor raspodjeljuje zadatke na radničke čvorove, što je prikazano linijama koje povezuju master s worker1 i worker2. Ove linije također predstavljaju mrežnu komunikaciju koja omogućuje prijenos podataka između glavnog čvora i radničkih čvorova unutar virtualnog klastera.

4. PROGRAMSKI ALATI POTREBNI ZA RAD

Za pravilan rad računalnog klastera i njegovu veću primjenjivost potrebno je na njega instalirati odgovarajuće programske alate. Alati za raspoređivanje poslova, kao što su Slurm i Torque, osiguravaju učinkovitu raspodjelu radnih zadataka među čvorovima. Alati za nadgledanje resursa, kao što su Ganglia i Zabbix, omogućuju nadzor performansi sustava, kao što su korištenje procesora, memorije i mrežnih resursa. Za komunikaciju između čvorova koriste se komunikacijske biblioteke, kao što je MPI. Alati za paralelno programiranje, kao što su OpenMP, omogućuju rastavljanje složenih zadataka na manje podzadatke koji se izvršavaju istovremena na različitim čvorovima računalnog klastera. Datotečni sustavi, kao što je NFS (*Network File System*), omogućuju zajedničko pohranjivanje i pristup podacima između čvorova. Za izbjegavanje podopterećenja koriste se alati za ravnotežu opterećenja, kao što su HAProxy i NGINX, koji osiguravaju stabilnost sustava. Ovi su alati ključni za održavanje i upravljanje računalnim klasterom te omogućuju učinkovitu izvedbu raspodijeljenih zadataka.

U virtualnom klasteru računala sklopu ovog rada koristit će se 5 programskih alata, a to su:

- **Secure Shell** (SSH): Mrežni protokol koji omogućuje uspostavu komunikacijskog kanala između dva čvora putem mreže.
- **Network File System** (NFS): Omogućuje dijeljenje datoteka između čvorova preko mreže kao da su lokalno pohranjene na diskovima.
- **Message Passing Interface** (MPI): Standardizirana biblioteka za komunikaciju između čvorova, namijenjena paralelnom računarstvu.
- **ClusterSSH**: Alat koji omogućuje istovremeno upravljanje s više računala preko jednog terminala.
- **Ganglia**: Nadzire performanse sustava, dizajniran za jednostavno praćenje resursa poput procesora i memorije na čvorovima.

Ovi alati čine minimalan skup programskih alata potreban za funkcionalan rad računalnog klastera. Zbog njihove funkcionalnosti i lake integracije, ovi alati postali su standard u industriji za upravljanje i nadzor računalnih klastera te se mogu pronaći u mnogim računalnim sustavima.

4.1. Secure Shell (SSH)

Sigurna ljuska (engl. *Secure Shell, SSH*), opisan u [10], jest kriptografski mrežni protokol koji je ključan za siguran daljinski pristup preko mreže. Osobito je popularan među sistemskim

administratorima jer omogućuje šifriranu komunikaciju i strogu autentifikaciju između dva uređaja, osiguravajući zaštitu od neovlaštenih pristupa. Neke od prednosti korištenja SSH-a u klasteru računala su centralizirano upravljanje i fleksibilnost. Omogućuje upravljanje svim računalima u klasteru računala s jednog klijentskog računala, poboljšavajući učinkovitost i produktivnost te podržava različite operacijske sustave i platforme što olakšava integraciju u heterogena okruženja.

Naredba „sudo apt install openssh-client“ izvršena je na master čvoru i služi za instalaciju SSH paketa. Naredba „sudo apt-get install openssh-server“ služi kako bi se radni čvorovi mogli povezati na master čvor, ali je potrebna lozinka mastera. Naredbe su prikazane na slici 4.1.

```
$ sudo apt install openssh-client
$ sudo apt-get install openssh-server
```

Slika 4.1. Naredbe za instalaciju SSH paketa i uspostavu veze za radne čvorove

Povezivanje se izvršava naredbom „ssh“ uz pripadajuću IP adresu čvora, koja je prikazana na slici 4.2.

```
$ ssh {IP adresa čvora na kojeg se želimo povezati}
```

Slika 4.2. Naredba za povezivanje

MPI, jedan od alata koji će se kasnije detaljnije objasniti u ovom radu, zahtjeva poslužitelja za komunikaciju sa radnim čvorovima bez upotrebe lozinke. To će se postići izvršavanjem naredbi prikazanih na slici 4.3.

```
$ ssh-keygen –t rsa
$ scp id_rsa.pub 192.168.100.101:~/.ssh/id_rsa.pub
$ scp id_rsa.pub 192.168.100.102:~/.ssh/id_rsa.pub
$ cat id_rsa.pub >> authorized_keys
```

Slika 4.3. Naredbe za generiranje i kopiranje ključeva

Naredbom „ssh-keygen –t rsa“ generiran je par privatnog i javnog ključa u „/.ssh“ direktoriju master čvora. Sigurnosne su kopije javnog ključa poslane prema radnim čvorovima. Nakon pokretanja naredbe „cat id_rsa.pub >> authorized_keys“ u oba radna čvora u „/.ssh“ direktoriju, master može pristupiti radnim čvorovima bez upotrebe lozinke.

Posljednje je preostalo omogućiti povezivanje između radnih čvorova bez upotrebe lozinki. Postiže se naredbama „ssh-keygen -t rsa 4096“ i „ssh-copy-id 192.168.100.102“, prikazanim na slici 4.4.

```
$ ssh-keygen -t rsa 4096
$ ssh-copy-id 192.168.100.102
```

Slika 4.4. Naredbe za generiranje i kopiranje ključeva na worker1 čvoru

Naredbe su izvršene na worker1 čvoru. Generira se par privatnog i javnog ključa te se autorizacijski podaci šalju na worker2 čvor. Isti postupak treba napraviti na worker2 čvoru. Jedina je izmjena *host* kojem šaljemo autorizacijske podatke. Naredba je prikazana na slici 4.5.

```
$ ssh-copy-id 192.168.100.101
```

Slika 4.5. Naredba za kopiranje ključeva na worker2 čvoru

4.2. Network File System (NFS)

Mrežni datotečni sustav (engl. *Network File System, NFS*) jest mrežni protokol za pohranu podataka koji omogućuje korisnicima na računalu pristupanje datotekama na udaljenom računalu preko mreže, slično kao što bi pristupili lokalnom datotečnom sustavu. Ova se potreba pojavila u vremenu kada su tvrdi diskovi imali mali kapacitet, a računala minimalan prostor za pohranu. Većina kapaciteta i prostora nalazila se na poslužiteljima i sličnim sustavima. Korisno bi bilo kad bi korisnici mogli pristupiti i manipulirati tim podacima na transparentan način, kao da rade s lokalnim podacima i direktorijima. To je upravo ono što NFS omogućuje kao što je pojašnjeno u [11]. Čini da udaljenje datoteke na drugom računalu izgledaju i ponašaju se kao lokalne datoteke na korisnikovom računalu.

Na master čvoru se izvršavaju naredbe prikazane na slici 4.6. Instaliraju se svi paketi potrebni za NFS poslužitelj, te se kreira zajednički direktorij. U ovom je radu direktorij nazvan „mirror“.

```
$ sudo apt-get install nfs-kernel-server portmap
$ sudo mkdir /mirror
```

Slika 4.6. Naredbe za instalaciju NFS paketa i kreiranje zajedničkog direktorija

Potrebno je promijeniti dozvolu pristupa zajedničkom direktoriju naredbom „sudo chmod 777 /mirror“ tako da svi čvorovi imaju potpuni pristup zajedničkom direktoriju.

Zatim je potrebno urediti „/etc/exports“ datoteku za NFS poslužitelj koja određuje koji direktorij će NFS poslužitelj dijeliti i s kojim čvorovima će dijeliti taj direktorij. U toj je datoteci potrebno upisati liniju prikazana na slici 4.7.

```
/mirror 192.168.100.0/24(rw,sync,no_root_squash,no_subtree_check)
```

Slika 4.7. Datoteka „/etc/exports“

Ovom linijom utvrđuje se da je „/mirror“ direktorij koji će biti dijeljen putem NFS-a, mrežu klijenata kojima je dozvoljen pristup direktoriju te sve dozvole i mogućnosti za čvorove. Kako bi se učitale i primijenile promjene u „/etc/exports“, izvršiti će se naredba prikazana na slici 4.8.

```
$ sudo exportfs -a
```

Slika 4.8. Naredba za primjenu promjena u „/etc/exports“

U svakom radnom čvoru će se izvršiti naredbe, prikazane na slici 4.9., za instalaciju potrebnih NFS paketa te kreiranje direktorija.

```
$ sudo apt-get install nfs-common portmap
$ sudo mkdir -p /mirror
```

Slika 4.9. Naredbe za instalaciju NFS paketa i kreiranje zajedničkog direktorija

Nakon što se pripremio direktorij potrebno je pokrenuti NFS poslužitelj na master čvoru. Naredba za pokretanje NFS poslužitelja prikazana je na slici 4.10.

```
$ sudo /etc/init.d/nfs-kernel-server start
```

Slika 4.10. Naredba za pokretanje NFS poslužitelja

Prilikom svakog pokretanja radnog računala potrebno je izvršiti naredbu za montiranje dijeljenog direktorija s udaljenog poslužitelja na radni čvor. Naredba je prikazana na slici 4.11. te se izvršava na svakom radnom čvoru.

```
$ sudo mount 192.168.100.100:/mirror /mirror
```

Slika 4.11. Naredba za montiranje zajedničkog direktorija na radnom čvoru

4.3. Message Passing Interface (MPI)

Sučelje za prosljeđivanje poruka (engl. *Message Passing Interface, MPI*) je prenosivi standard za razmjenu poruka namijenjen za rad u paralelnoj računalnoj arhitekturi. Omogućava komunikaciju između različitih procesa koji se mogu izvoditi na više računala ili više procesora unutar istog računala, kao što je spomenuto u [12]. Postoji nekoliko MPI implementacija koje su unaprijedile industriju paralelnog softvera te potaknule razvoj prijenosnih i skalabilnih paralelnih aplikacija. Neke od najpoznatijih i najčešće korištenih MPI implementacija su MPICH, OpenMPI, Intel MPI i MVAPICH. U sklopu ovog rada odabrana je MPICH implementacija zbog široke dostupnosti i kompatibilnosti s različitim računalnim sustavima. MPICH je dobro dokumentirana MPI implementacija i prilagodljiv je na male i velike računalne klastere i ovdje će se koristiti za

paralelno raspodjelu zadataka među virtualnim čvorovima unutar virtualnog računala klastera, omogućujući učinkovitiju obradu podataka.

U „mirror“ direktoriju master čvora preuzme se MPICH paket verzije 4.2.2., trenutno najnovija MPICH verzija, te se zatim raspakira. Naredbe su prikazane na slici 4.12. Objašnjeno u [13], MPICH je široko prenosiva implementacija MPI standarda. Razvijen je na Argonne National Laboratory i koristi se kao osnovna komponenta u mnogim HPC sustavima.

```
$ sudo wget http://www.mpich.org/static/downloads/4.2.2/mpich-4.2.2.tar.gz
$ sudo tar xvf mpich-4.2.2.tar.gz
```

Slika 4.12. Naredbe za preuzimanje MPICH paketa i raspakiranje istih

U direktoriju „mpich.4.2.2/“ u kojem su raspakirani paketi potrebno je instalirati alate potrebne za kompilaciju softvera. Nakon toga potrebno je konfigurirati MPICH. Naredbe za potrebnu instalaciju i konfiguraciju su prikazane na slici 4.13.

```
$ cd mpich.4.2.2/
$ sudo apt-get install build-essential
$ sudo ./configure --prefix=/mirror/mpich2 CPPFLAGS=-fcommon --disable-f77 -
disable-fc
```

Slika 4.13. Naredbe za instalaciju potrebnih alata i konfiguraciju MPICH

Naredba konfigurira MPICH prije instalacije s postavkama koje uključuju instalaciju u „/mirror/mpich2“ direktorij, onemogućava podršku za Fortran iz razloga što se ne koristi u virtualnom klasteru računala u sklopu ovog rada, i dodaje „-fcommon“ zastavicu za kompilaciju kako bi se riješili potencijalni problemi s više deklaracija varijabli.

Nakon preuzimanja i raspakiravanja MPICH paketa, koriste se naredbe za kompajliranje MPICH prikazane na slici 4.14.

```
$ sudo make
$ sudo make install
$ export PATH=/mirror/mpich2/bin:$PATH
$ export LD_LIBRARY_PATH=”/mirror/mpich2/lib:$ LD_LIBRARY_PATH”
```

Slika 4.14. Naredbe za kompajliranje MPICH

Naredba „make“ kompajlira MPICH iz izvornog koda, a naredba „make install“ instalira kompajlirani softver u direktorij specificiran tijekom konfiguracije (u ovom slučaju „/mirror/mpich2“). Nadalje, potrebno je postaviti varijable okruženja kako bi sustav prepoznao gdje se nalaze izvršne datoteke i biblioteke MPICH-a. Ažuriraju se PATH i

LD_LIBRARY_PATH varijable u koje se uključuju MPICH datoteke i biblioteke. Kako bi ove promjene bile trajne i dostupne svim čvorovima, potrebno je urediti datoteku „/etc/environment“. U ovoj se datoteci uređuje PATH varijabla tako da uvijek uključuje direktorij s MPICH izvršnim datotekama, čak i nakon ponovnog pokretanja sustava. Dio linije prikazan je na slici 4.15.

```
PATH=”/mirror/mpich2/bin:...”
```

Slika 4.15. Datoteka „/etc/environment“

Dalje, potrebno je urediti datoteku „hosts“ u kojoj se upisuju IP adrese čvorova koji će se koristiti za MPI komunikaciju. Adrese master, worker1 i worker2 čvorova koje upisujemo u „hosts“ datoteku prikazane su na slici 4.16.

```
192.168.100.100
192.168.100.101
192.168.100.102
```

Slika 4.16. Datoteka „hosts“

4.4. ClusterSSH

ClusterSSH je alat za upravljanje više Linux poslužitelja s jednog računala. Kao što je objašnjeno u [14], omogućuje istovremeno izvršavanje naredbi na svim čvorovima u računalnom klasteru koji su povezani SHH protokolom. Ovi čvorovi obično imaju istu konfiguraciju, pa je potrebno izvršiti istu naredbu na svakom od njih. Ovaj alat je vrlo koristan administratorima koji često moraju obavljati iste zadatke na velikom broju računala.

Sve se naredbe za podešavanje ClusterSSH alata izvršavaju na master čvoru. Potrebno je pokrenuti naredbu za instalaciju ClusterSSH, koja je prikazana na slici 4.17.

```
$ sudo apt-get install clusterssh
```

Slika 4.17. Naredba za instalaciju ClusterSSH

Dalje je potrebno urediti „/etc/clusters“ datoteku. Ona sadrži informacije o tome koja računala i klasteri računala su povezani s ClusterSSH. Unutar datoteke definiramo radne čvorove gdje je worker1 čvor označen kao „pc1“, worker2 čvor označen kao „pc2“ i oba čvora zajedno kao „all“. Pri definiranju pojedinačnog radnog čvora, uz oznaku upisujemo pripadajuću adresu za povezivanje. Uređena datoteka „/etc/clusters“ prikazana je na slici 4.18.

```
clusters pc1 pc2 all

pc1 luka@192.168.100.101

pc2 luka@192.168.100.102

all pc1 pc2
```

Slika 4.18. Datoteka „/etc/clusters“

Naredba prikazana na slici 4.19. omogućava pristup grafičkom sučelju sa svih računala. Prilikom svakog korištenja ClusterSSH alata, potrebno je uvijek izvršiti ovu naredbu. Razlog je tome što je ova naredba privremena i vrijedi samo za trenutnu sesiju grafičkog sučelja.

```
$ xhost +
```

Slika 4.19. Naredba za pristup grafičkom sučelju sa svih računala

4.5. Ganglia

Ganglia je sustav za praćenje performansi i stanja distribuiranih sustava, računalnih klastera i mreža. Koristi se za prikazivanje statistike u realnom vremenu ili snimljenih podataka o podacima poput opterećenja CPU-a, iskorištenja memorije i propusnosti mreže na mnogim čvorovima simultano. Prema [15], trenutno se koristi na tisućama distribuiranih sustava diljem svijeta te se može povećati za upravljanje računalnim klasterima s nekoliko tisuća računala.

Sljedeći je niz naredbi izvršen na master čvoru. Prvo se instaliraju potrebni paketi izvršavanjem naredbe prikazane na slici 4.20.

```
$ sudo apt-get install ganglia-monitor rrdtool gmetad ganglia-webfrontend
```

Slika 4.20. Naredba za instalaciju potrebnih Ganglia paketa

Potrebno je kopirati zadanu Apache konfiguracijsku datoteku za Ganglijin webfrontend u direktorij gdje Apache učitava konfiguraciju za web stranice. Ovaj je korak potreban kako bi Apache web poslužitelj mogao pravilno poslužiti Ganglijino web sučelje. Naredba je prikazana na slici 4.21.

```
$ sudo cp /etc/ganglia-webfrontend/apache.conf /etc/apache2/sites-
enabled/ganglia.conf
```

Slika 4.21. Naredba za kopiranje Apache konfiguracijske datoteke

Potrebno je urediti konfiguracijsku datoteku „/etc/ganglia/gmetad.conf“. Na slici 4.22. prikazana je linija u datoteci koju je potrebno izmijeniti.

```
data_source "my cluster" localhost
```

Slika 4.22. Datoteka „/etc/ganglia/gmetad.conf“ prije izmjene

Slika 4.23. prikazuje promijenjenu liniju u datoteci „/etc/ganglia/gmetad.conf“.

```
data_source "feritcluster" 50 192.168.100.100:8649
```

Slika 4.23. Datoteka „/etc/ganglia/gmetad.conf“ poslije izmjene

Naziv virtualnog klastera računala je označen s „feritcluster“, 50 je vrijeme za osvježavanje podataka (u sekundama), 192.168.100.100 je IP adresa master čvora na portu 8649.

Potrebno je urediti datoteku „/etc/ganglia/gmond.conf“. Unutar datoteke izmijenit će se nekoliko atributa. Potrebno je promijeniti ime klastera računala iz „unspecified“ u „feritcluster“. U udp_send_channel potrebno je promijenit atribut „host“ u IP adresu master čvora. Komentirati sve ostale linije osim porta tako što stavimo # na početku linije. „udp_send_channel“ govori gdje će svaki čvor slati podatke putem UDP protokola. U „udp_recv_channel“ će se iskomentirati sve linije osim porta koji mora biti 8649. Ovo se postavlja samo na master čvoru iz razloga što on jedini prikuplja informacije ostalih čvorova. Unutar „tcp_accept_channel“ potrebno je ostaviti samo port koji mora bit 8649. Promijenjeni atributi bi trebali izgledati kao na slici 4.24.

```
…
cluster {
        name = „feritcluster“
        …
}
udp_send_channel {
        host = 192.168.100.100
        # mcast_join = …
        port = 8649
        # ttl = …
}
udp_recv_channel {
        # mcast_join = …
        port = 8649
        # bind = …
}
tcp_accept_channel {
        port = 8649
}
…
```

Slika 4.24. Datoteka „/etc/ganglia/gmond.conf“

Zadnja instalacija koju je potrebno napraviti na master čvoru je za stariju verziju PHP-a. Trenutna i najnovija verzija PHP-a 8.3 nije kompatibilna s Ganglia alatom. Zadnja verzija PHP-a koja je kompatibilna s Ganglia alatom je PHP 7.2 verzija, a ista će se instalirati na master čvoru izvršavanjem naredbe prikazane na slici 4.25.

```
$ sudo apt-get install apache2
```

Slika 4.25. Naredba za instalaciju Apache

Nakon instalacije Apache, poslužitelj se pokreće izvršenjem naredbi „sudo systemctl start apache2.service“ i „sudo systemctl enable apache2.service“, prikazanih na slici 4.26.

```
$ sudo systemctl start apache2.service
$ sudo systemctl enable apache2.service
```

Slika 4.26. Naredbe za pokretanje Apache poslužitelja

Zatim se instaliraju potrebni moduli i svojstva te prikladna PHP 7.2 verzija izvršavanjem naredbi prikazanih na slici 4.27.

```
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:ondrej/php
$ sudo apt-get install php7.2 libapache2-mod-php7.2 php7.2-common php7.2-gmp
php7.2-curl
$ sudo apt-get install php7.2-intl php7.2-mbstring php7.2-xmlrpc php7.2-mysql
php7.2-gd
$ sudo apt-get install php7.2-xml php7.2-cli php7.2-zip
```

Slika 4.27. Naredbe za instalaciju modula i svojstva te prikladne PHP 7.2 verzije

Potrebno je urediti „etc/php/7.2/apache2/php.ini“ datoteku. Unutar datoteke potrebno je promijeniti postavke za omogućavanje učitavanja datoteka, otvaranje URL-ova te dopuštanja kratkih PHP oznaka. Promijenjene postavke prikazane su na slici 4.28.

```
file_uploads = On
allow_url_fopen = On
short_open_tag = On
```

Slika 4.28. Promijenjene linije u datoteci „etc/php/7.2/apache2/php.ini“

Spreme se promjene te se ponovno pokrene Apache poslužitelj izvršavanjem naredbe prikazane na slici 4.29.

```
$ sudo systemctl restart apache2.service
```

Slika 4.29. Naredba za ponovno pokretanje Apache poslužitelja

Potrebno je onesposobiti trenutnu PHP 8.3 verziju izvršavanjem naredbe prikazane na slici 4.30.,

```
$ sudo a2dismod php8.3
$ sudo systemctl restart apache2
```

Slika 4.30. Naredbe za onesposobljavanje PHP 8.3 verzije

te osposobiti PHP 7.2 verziju izvršavanjem naredbe prikazane na slici 4.31.

```
$ sudo a2enmod php7.2
$ sudo systemctl restart apache2
```

Slika 4.31. Naredbe za osposobljavanje PHP 7.2 verzije

Nakon instalacije prikladne PHP verzije pokreće se Ganglia alat na master čvoru. Pokretanje Ganglia alata izvršava se naredbama prikazanim na slici 4.32.

```
$ sudo systemctl start ganglia-monitor
$ sudo systemctl start gmetad
```

Slika 4.32. Naredbe za pokretanje Ganglia alata na master čvoru

Sljedeći niz naredbi izvršen je u oba radna čvora. Na svim čvorovima će se instalirati Ganglia monitor izvršavanjem naredbe prikazane na slici 4.33.

```
$ sudo apt-get install ganglia-monitor
```

Slika 4.33. Naredba za instalaciju Ganglia alata na radnim čvorovima

Potrebno je urediti datoteku „/etc/ganglia/gmond.conf“ kao i na master čvoru. Izmijenit će se ime klastera računala iz „unspecified“ u „feritcluster“. Unutar „udp_send_channel“ će se izmijeniti parametar „host“ u IP adresu master čvora. Također je potrebno iskomentirati linije „mcast_join“ i „ttl“ te će se ostaviti port 8649. Iste izmijene kao na master čvoru.

Potrebno je iskomentirati cijeli dio za „udp_recv_channel“ i „tcp_accept_channel“. To se radi iz razloga što radni čvorovi ne trebaju primati podatke o opterećenju od drugih čvorova. Radni čvorovi samo šalju podatke prema master čvoru.

Unutar „globals“ potrebno je promijeniti parametar „deaf“ da ima vrijednost „yes“. Taj parametar postavlja radne čvorove u način rada gdje oni ne pokušavaju slušati dolazne podatke od drugih čvorova, već samo šalju vlastite podatke prema master čvoru. Time se sprječava da radni čvorovi nepotrebno troše resurse na obradu podataka koje ne moraju primati. Datoteka „/etc/ganglia/gmond.conf“ treba izgledati kao na slici 4.34.

```
globals {
        …
        deaf = yes
        …
}
cluster {
        name = „feritcluster“
        …
}
udp_send_channel {
        host = 192.168.100.100
        # mcast_join = …
        port = 8649
        # ttl = …
}
udp_recv_channel {
        # mcast_join = …
        # port = 8649
        # bind = …
}
tcp_accept_channel {
        # port = 8649
}
…
```

Slika 4.34. Datoteka „/etc/ganglia/gmond.conf“ na radnom čvoru

Zadnja korak je pokretanje Ganglia monitor na radnim čvorovima izvršavanjem naredbe prikazane na slici 4.35.

```
$ sudo systemctl start ganglia-monitor
```

Slika 4.35. Naredba za pokretanje Ganglia alata na radnim čvorovima

5. VREDNOVANJE PROGRAMSKOG RJEŠENJA

U ovom poglavlju prikazat će se testiranja provedena na virtualnom klasteru računala u okviru ovog rada. Testiranje je nužan korak u vrednovanju performansi i funkcionalnosti klastera računala. S obzirom na to da je klaster računala dizajniran za obradu velikih količina podataka i izvršavanje paralelnih zadataka, važno je razumjeti kako se on ponaša pod različitim opterećenjima i u stvarnim scenarijima korištenja. Klasteri računala mogu se testirati na razne načine. Propusnost i latencija mreže testira pomoću alata poput „iperf“ i „ping“ kako bi se procijenila brzina prijenosa podataka. Performanse procesora testiraju se alatima kao što su Linpack i HPCG kako bi se procijenila sposobnost izvođenja paralelnih zadataka. Testiranje memorijskih performansi postiže se alatima poput STREAM, kojim se mjeri brzina pristupa memoriji.

Testiranja provedena na virtualnom klasteru računala u okviru ovog rada bazirat će se na MPI programu za množenje matrica te simultano čitanje i pisanje datoteka na radnim čvorovima. Iskorištenost različitih resursa tijekom testiranja prikazana je Ganglia alatom.

MPI program za množenje matrica napisan je u C jeziku i naziva se „matrix_multiplication.c“. Program je strukturiran prema master-worker modelu, gdje se jedan proces ponaša kao master proces, zadužen za upravljanje cjelokupnim zadatkom, dok su svi ostali procesi definirani kao radni procesi. Uloga master procesa je inicijalno generirati matrice, rasporediti zadatak među radnim procesima te na kraju prikupiti rezultate i sastaviti konačnu matricu. Radni procesi su odgovorni za obavljanje stvarnog izračuna, čime se rasterećuje master proces i omogućuje paralelno izvođenje zadatka na više procesora.

Master proces najprije generira matrice A i B koristeći slučajne jednoznamenkaste brojeve. Nakon toga, redci matrice A šalju se radnim procesima, dok se matrica B šalje u cijelosti svakom radnom procesu jer je potrebna za izračun svakog retka rezultirajuće matrice. Svaki radni proces prima dodijeljene retke matrice A, provodi množenje s odgovarajućim stupcima matrice B te izračunava dio rezultirajuće matrice C. Nakon dovršetka izračuna, radni procesi šalju rezultate natrag master procesu, koji prikuplja sve rezultate i formira konačnu matricu C. Na slici 5.1. prikazan je pseudokod cjelokupne radnje.

```
Inicijaliziraj MPI okruženje
Dohvati ID trenutnog procesa i ukupan broj procesa

Ako je master proces:
    - Generiraj matricu A s nasumičnim jednoznamenkastim brojevima
    - Generiraj matricu B s nasumičnim jednoznamenkastim brojevima
    - Počni mjerenje vremena
    - Podijeli redke matrice A među radne procese
    - Pošalji redke matrice A i cijelu matricu B svakom radnom procesu
    - Čekaj da radni procesi završe izračun
    - Primi rezultat izračuna od svakog radnog procesa
    - Zaustavi mjerenje vremena
    - Prikaz rezultirajuće matrice C
    - Prikaz vremena potrebnog za izračun

Ako je radni (slave) proces:
    - Primi dodijeljene redke matrice A od master procesa
    - Primi cijelu matricu B od master procesa
    - Izračunaj svoj dio matrice C množenjem primljenih redaka matrice A
      s odgovarajućim stupcima matrice B
    - Pošalji rezultate natrag master procesu

Završi MPI okruženje
```

Slika 5.1. Pseudokod MPI programa za množenje matrica

Algoritam za množenje matrica implementiran je ručno, korištenjem trostruke petlje koja prolazi kroz elemente matrica A i B, bez korištenja gotovih knjižnica ili kernel funkcija optimiziranih za ovu vrstu operacija. Množenje se provodi prema standardnoj formuli za množenje matrica, gdje se svaki element matrice C računa kao zbroj umnožaka elemenata iz odgovarajućih redaka matrice A i stupaca matrice B.

Program koristi MPI funkcije „MPI_Send“ i „MPI_Recv“ za slanje i primanje podataka između master i radnog procesa. Radni procesi primaju potrebne podatke od mastera, izvršavaju izračune te vraćaju rezultate koristeći iste komunikacijske funkcije. Ova struktura omogućava raspodjelu posla među procesima, čime se skraćuje ukupno vrijeme izvođenja programa, osobito kod velikih matrica.

Program nakon izvršenja prikazuje matricu A, matricu B, matricu C te potrebno vrijeme za izračunavanje matrice C.

Uz MPI program koji će se izvršiti na virtualnom klasteru računala, na fizičkom računalu će se izvršiti preoblikovan program za množenje matrica. Program za fizičko računalo prilagođen je tako da obavlja izračune uz pomoć OpenMP (engl. *Open Multiprocessing*) alata, zadržavajući istu funkcionalnost algoritma za množenje matrica. OpenMP alat omogućuje fizičkom računalu da

iskoristi više obradbenih jedinica pri izvršavanju programa čime se značajno smanjuje vrijeme izvršavanja u odnosu na serijsko izvršavanje. Programi su testirani za 5 različitih dimenzija matrica te su rezultati prikazani u tablici 5.2.

Tablica 5.2. Rezultati testiranja programa za množenje matrica

Dimenzija matrica	Vrijeme virtualnog klastera računala	Vrijeme fizičkog računala
10 x 10	0.057 ms	~0 ms
50 x 50	4.359 ms	0.0002 ms
100 x 100	15.287 ms	0.34 ms
500 x 500	0.43 s	0.120 s
1000 x 1000	3.026 s	1.189 s

Iz dobivenih rezultata može se uočiti da fizičko računalo optimizirano pomoću OpenMP alata daje bolje rezultate u usporedbi s virtualnim klasterom računala, pogotovo kod manjih i srednjih matrica. Ovi rezultati su očekivani s obzirom na to da se komunikacijski *overhead* unutar virtualnog klastera računala, koji uključuje prijenos podataka između radnih čvorova, pojavljuje čak i kod optimizirane izvedbe klastera računala. S druge strane, fizičko računalo ima mogućnost direktnog pristupa svim svojim resursima bez potrebe za komunikacijom između zasebnih instanci, što omogućuje brže izvođenje zadataka.

Kod većih dimenzija matrica (500x500 i 1000x1000), virtualni klaster računala i dalje pokazuje zadovoljavajuće rezultate, ali fizičko računalo uz OpenMP ostvaruje bolju izvedbu zbog eliminacije komunikacijskih uskih grla. Međutim, virtualni klaster računala je posebno koristan u scenarijima gdje je potrebno kreirati više različitih virtualnih instanci koje istovremeno obrađuju različite zadatke, čime se osigurava učinkovito korištenje resursa u heterogenim i dinamičkim okruženjima.

Važno je razumjeti svrhu virtualnog klastera računala. Virtualni klaster računala, bez obzira na to koliko je dobro optimiziran, neće nadmašiti performanse fizičkog računala na kojem je implementiran, kada to računalo direktno pokreće paralelni program pomoću OpenMP alata. Ključna prednost virtualnih klastera računala nije u nadmašivanju fizičkog računala u performansama, već u njihovoj fleksibilnosti i sposobnosti da podrže višestruke paralelne zadatke u izoliranim okruženjima. Virtualni klaster računala omogućuje skalabilnost i dinamičku raspodjelu resursa, što ga čini pogodnim za okruženja u kojima postoji potreba za istovremenim izvršavanjem više različitih zadataka na različitim virtualnim čvorovima.

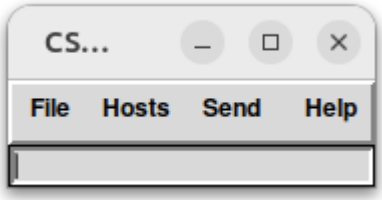
Nakon pokretanja MPI programa, koristit će se ClusterSSH alata za paralelno izvođenje naredbi na radnim čvorovima. Koristan je za testiranje iz razloga što omogućuje istovremeno nadgledanje resursa više radnih čvorova.

Naredba za pokretanje ClusterSSH alata za oba radna čvora prikazana je na slici 5.3.

```
$ cssh all
```

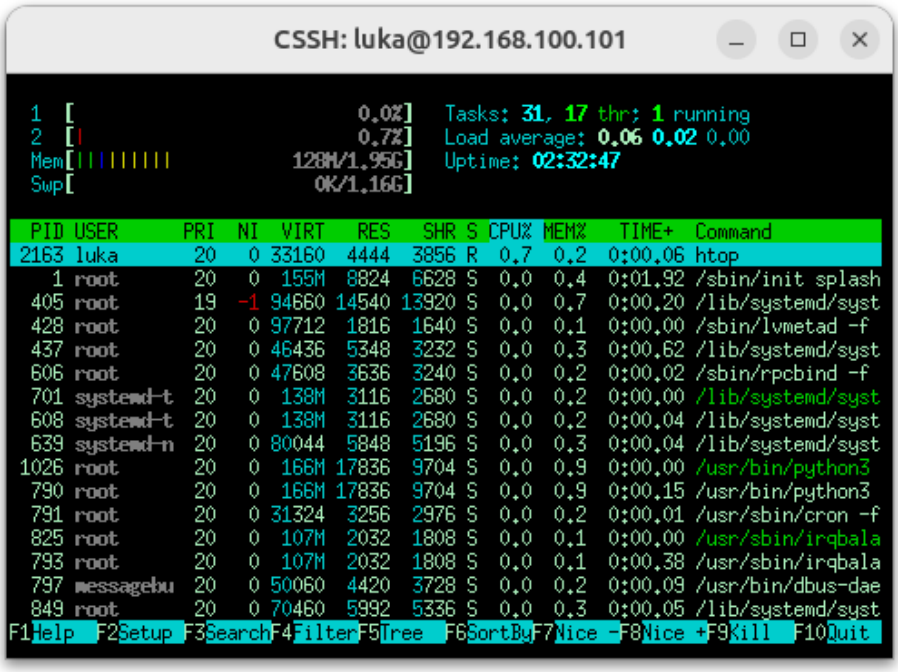
Slika 5.3. Naredba za pokretanje ClusterSSH alata

Ova naredba omogućuje simultano upravljanje svim čvorovima koji su ranije bili definirani u konfiguraciji „all“. Uspješno izvedena naredba trebala bi otvoriti prozor s naredbenim retkom prikazano na slici 5.4.

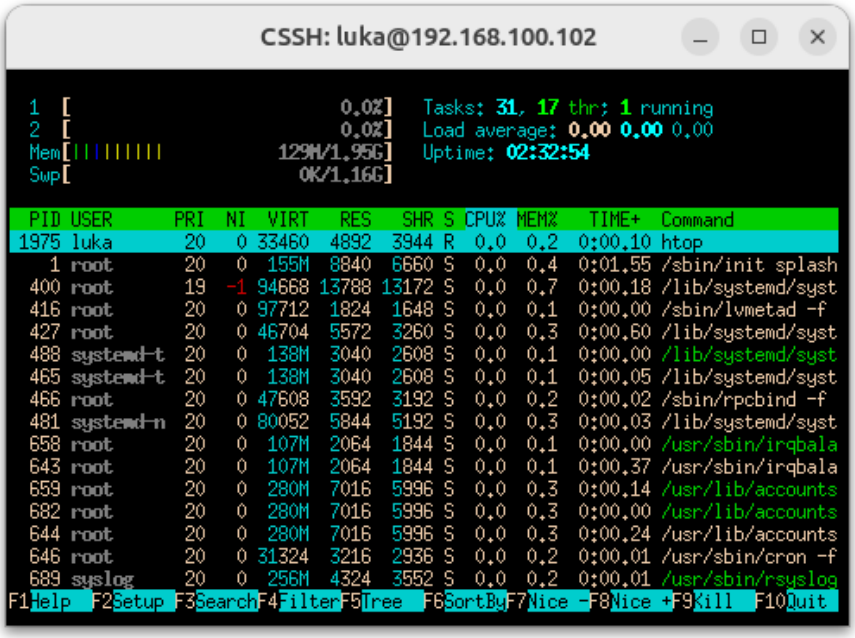


Slika 5.4. Naredbeni prozor ClusterSSH alata

Naredbe koje se unose u naredbenom retku se istovremeno unose na oba radna čvora. Unijet će se naredba „htop“ koja omogućuje nadzor korištenja procesora i memorije na oba radna čvora u realnom vremenu. Na slikama 5.5. i 5.6. prikazani su radni čvorovi nakon izvedene naredbe.



Slika 5.5. Worker1 čvor nakon izvedene naredbe „htop“



Slika 5.6. Worker2 čvor nakon izvedene naredbe „htop“

Alat ClusterSSH poslužio je za izvođenje simultanog čitanja i pisanja datoteka na oba radna čvora čime je testirana učinkovitost NFS-a. Za ovo testiranje koristimo „dd“ naredbu koja simulira pisanje i čitanje datoteke. Naredba za pisanje datoteke prikazana je na slici 5.7.

```
$ dd if=/dev/zero of=/mirror/test_file bs=1M count=100 oflag=direct
```

Slika 5.7. Naredba za pisanje datoteke

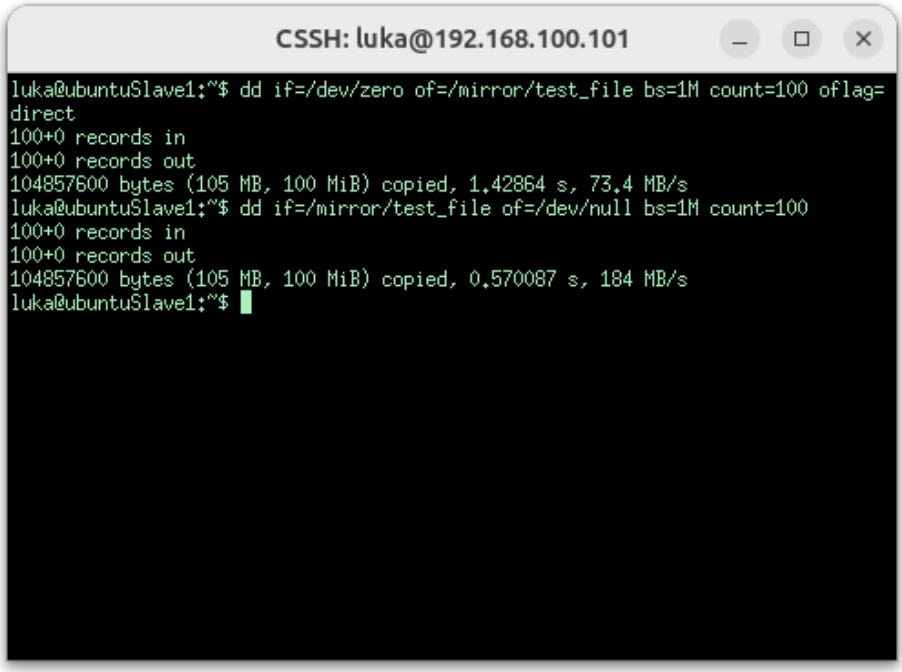
Ova naredba stvara datoteku od 100 MB koja je zapisana s nulama. Datoteka je zapisana u „mirror“ direktoriju.

Naredba za čitanje datoteke s „mirror“ direktorija prikazana je na slici 5.8.

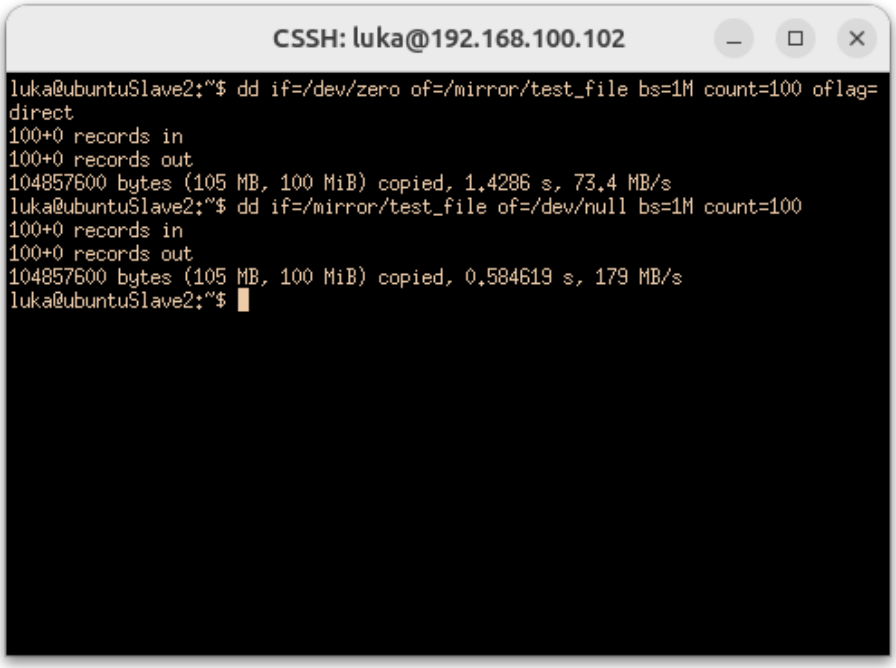
```
$ dd if=/mirror/test_file of=/dev/null bs=1M count=100
```

Slika 5.8. Naredba za čitanje datoteke

Rezultati dobiveni na dvama radnim čvorovima mogu se vidjeti na slikama 5.9. i 5.10.

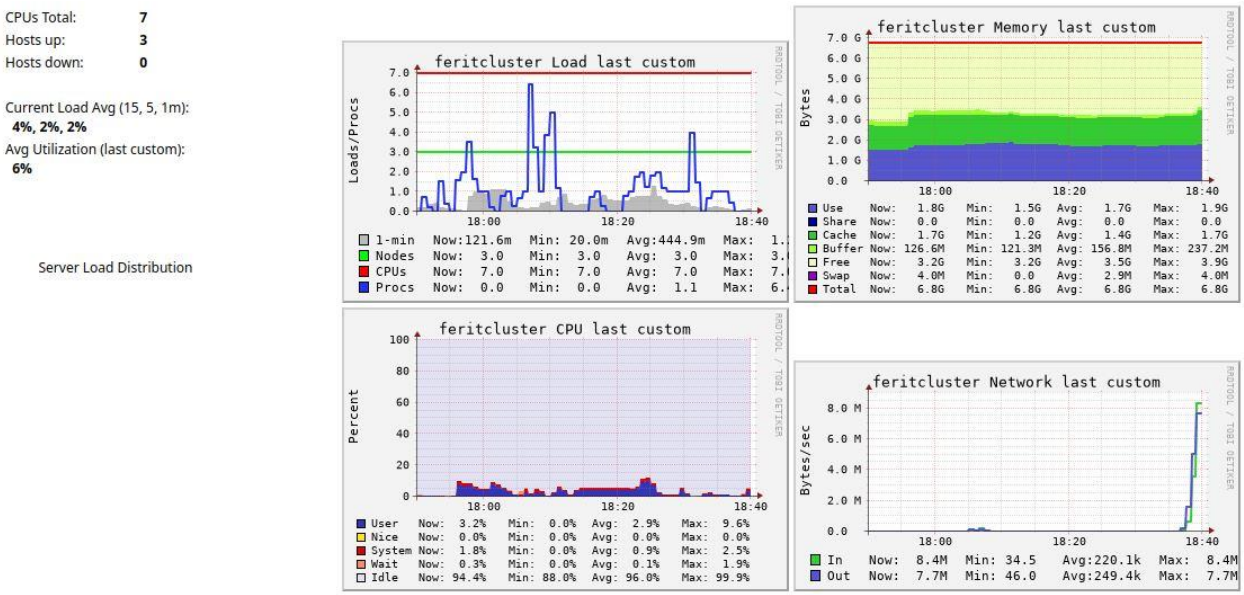


Slika 5.9. Worker1 čvor nakon izvedenih „dd“ naredbi



Slika 5.10. Worker2 čvor nakon izvedenih „dd“ naredbi

Tijekom testova praćenje je resursa ključni dio evaluacije performansi računalnog klastera. Grafove performansi računalnog klastera mogu se vidjeti na slici 5.11. Dostupni su na web sučelju Ganglia alata: http://192.168.100.100/ganglia/



Slika 5.11. Vizualizirana trenutna iskorištenost različitih resursa virtualnog klastera računala pomoću alata Ganglia

Opterećenje sustava (Current Load Avg) pokazuje da je klaster računala većinu vremena pod niskim opterećenjem, s trenutnim opterećenjem od 4%, 2%, i 2% za posljednjih 15, 5 i 1 minutu. U određenim trenucima zabilježeni su skokovi u opterećenju, pri čemu je opterećenje u jednom

trenutku doseglo maksimalnu vrijednost od 7 zadataka, što znači da su sve procesorske jezgre bile zauzete.

Korištenje memorije pokazuje stabilno stanje s malim promjenama u upotrebi, pri čemu je trenutno zauzeto 1.5 GB RAM-a, dok je 1.76 GB memorije priručna (engl. *cache*). Iskorištenost procesora također pokazuje da je većina procesorskih resursa u mirovanju. Povremeni skokovi u procesorskim aktivnostima poklapaju se s periodima izvođenja intenzivnih zadataka.

Što se tiče mrežnog prometa, najveći dio vremena promet je bio nizak, no zabilježen je nagli skok na kraju grafikona, pri čemu je ulazni promet dosegao 8.4 MB/s, a izlazni promet 7.7 MB/s. Ovi skokovi sugeriraju intenzivnu razmjenu podataka među čvorovima, što je povezano s NFS operacijama čitanja i pisanja datoteka. Općenito, alat Ganglia je omogućio detaljno praćenje resursa klastera računala, a rezultati pokazuju da sustav radi stabilno s povremenim skokovima u opterećenju, mrežnom prometu i korištenju procesora.

6. ZAKLJUČAK

U ovom radu istražena je i implementirana tehnologija virtualnih klastera računala, koja predstavlja važan alat u suvremenom računarstvu zbog svoje sposobnosti dinamičkog skaliranja i fleksibilnog korištenja resursa. Pregledom postojećih tehnologija i napora za izradu klastera računala, uključujući primjere energetski učinkovitih i visokoperformansnih klastera računala, te načine njihove primjene u znanstvenim i industrijskim okruženjima. Ovaj pregled omogućio je jasnije razumijevanje potreba i prednosti računalnih klastera, što je poslužilo kao temelj za daljnju izradu virtualnog klastera računala unutar ovog rada.

Na temelju tih informacija, rad se fokusirao na praktičnu implementaciju virtualnog klastera računala koristeći VirtualBox, uz konfiguraciju triju virtualnih računala - jednog glavnog (master) i dva radnička (worker) čvora. Ova struktura omogućila je osnovnu, ali funkcionalnu simulaciju klastera računala, koji se zatim koristio za testiranje uz pomoć programskih alata poput MPI za paralelnu obradu zadataka, NFS za dijeljenje datoteka između čvorova te SSH za sigurnu komunikaciju. Uz pomoć alata ClusterSSH omogućeno je upravljanje svim čvorovima s jednog sučelja, što je dodatno poboljšalo učinkovitost rada klastera računala. Ganglia je korištena za detaljan nadzor performansi klastera računala, pružajući uvid u korištenje resursa poput procesora, memorije i mreže tijekom izvršavanja zadataka.

Testiranje je provedeno pomoću MPI programa za paralelno množenje matrica, no rezultati su pokazali da fizičko računalo, optimizirano korištenjem OpenMP-a, nadmašuje virtualni klaster računala u svim dimenzijama matrica. Unatoč tome, virtualni klaster računala i dalje nudi fleksibilnost i mogućnost skaliranja koja omogućuje pokretanje višestrukih paralelnih zadataka u izoliranim okruženjima, što ga čini pogodnim za specifične scenarije gdje je potrebno simultano izvršavanje različitih zadataka. Dok fizičko računalo može ostvariti bolje rezultate zahvaljujući direktnoj uporabi resursa, virtualni klaster računala omogućuje pristup resursima kroz dinamičku raspodjelu i skaliranje. Osim toga, testiranje pisanja i čitanja datoteka putem NFS-a ukazalo je na stabilnost mrežnih postavki i mogućnost zajedničkog korištenja resursa između čvorova.

Alat Ganglia, alat za nadzor rada klastera računala, omogućio je grafički prikaz iskorištenosti različitih resursa virtualnog klastera računala u okviru ovog rada. Povremeni skokovi u opterećenju bili su povezani s intenzivnim operacijama čitanja i pisanja. Općenito, rezultati testiranja pokazali su da je virtualni klaster računala stabilan i prikladan za izvođenje složenih zadataka u okruženju s minimalnim brojem čvorova.

Ovaj rad ukazuje na to da virtualni klasteri računala, uz pravilnu konfiguraciju i korištenje alata za upravljanje i nadzor, mogu biti učinkovito rješenje za aplikacije koje zahtijevaju paralelnu obradu podataka. Daljnje istraživanje i razvoj mogu se usmjeriti na proširenje broja čvorova, optimizaciju mrežnih postavki i testiranje u složenijim scenarijima, čime bi se dodatno unaprijedila skalabilnost i performanse klastera računala za različite vrste aplikacija.

LITERATURA

[1] R. Jedynak, „PRACTICAL IMPLEMENTATION OF A VIRTUAL COMPUTER CLUSTER“, Internacional Scientific Conference, Radom, 2011.

[2] J., Xiong, S.-H., Shi, S., Zhang, „Build and Evaluate a Free Virtual Cluster on Amazon Elastic Compute Cloud for Scientific Computing“, *Int. J. Online Biomed. Eng. IJOE*, izd. 08, sv. 13, str. 121–132, kolovoz 2017.

[3] J., St. John, T., Hacker, „Developing Virtual Clusters for High Performance Computing Using OpenNebula“, u *2012 ASEE Annual Conference & Exposition Proceedings*, str. 25.439.1-25.439.9, San Antonio, Texas, 2012.

[4] C.-T., Yang, S.-T., Chen, Y.-S., Lo, E., Kristiani, Y.-W., Chan, „On construction of a virtual GPU cluster with InfiniBand and 10 Gb Ethernet virtualization“, *J. Supercomput.*, izd. 12, sv. 74, str. 6876–6897, prosinac 2018.

[5] H.-E., Yu, W., Huang, „Building a Virtual HPC Cluster with Auto Scaling by the Docker“, rujan 2015.

[6] Wikipedia, Computer cluster, [online], dostupno na: https://en.wikipedia.org/wiki/Computer_cluster, [4.9.2024].

[7] Tim Anderson’s IT Writing, VirtualBox is amazing, 50% faster than Virtual PC on my PC, [online], dostupno na: https://www.itwriting.com/blog/660-virtualbox-is-amazing-50-faster-than-virtual-pc-on-my-pc.html [5.6.2008.].

[8] Oracle, Oracle VM VirtualBox, [online]. dostupno na: https://www.oracle.com/virtualization/virtualbox/ [28.6.2024.].

[9] Noviantika G., What Is Ubuntu? A Quick Beginner’s Guide, [online], dostupno na: https://www.hostinger.com/tutorials/what-is-ubuntu [1.4.2022.] .

[10] Venafi, What Is SSH (Secure Shell)? All about the SSH Protocol, [online], dostupno na: https://venafi.com/machine-identity-basics/what-is-ssh-secure-shell/ [29.6.2024.].

[11] HrOpenWiki, Dokumentacija NFS, [online], dostupno na: http://wiki.open.hr/wiki/Dokumentacija_NFS [30.6.2024.].

[12] TechTarget, What is Message Passing Interface (MPI)?, [online], dostupno na: https://www.techtarget.com/searchenterprisedesktop/definition/message-passing-interface-MPI [29.6.2024.].

[13] Ubuntu, mpich binary package in Ubuntu Trusty arm64, [online], dostupno na: https://answers.launchpad.net/ubuntu/trusty/arm64/mpich [1.7.2024.].

[14] U.S. Department of Veterans Affairs, Cluster Secure Shell (ClusterSSH), [online], dostupno na: https://www.oit.va.gov/Services/TRM/ToolPage.aspx?tid=16304 [30.6.2024.].

[15] NVidia Developer, Ganglia Monitoring System, [online], dostupno na: https://developer.nvidia.com/ganglia-monitoring-system [30.6.2024.].

SAŽETAK

S obzirom na rastuće potrebe za velikim računalnim resursima i složenim radnim opterećenjima, virtualni klasteri računala postaju dobro rješenje za optimizaciju računalne snage i upravljanje resursima. U ovom radu prikazana je izrada i implementacija virtualnog klastera računala koristeći alate kao što su VirtualBox, SSH, NFS, MPI i Ganglia, s ciljem osiguravanja učinkovitog dijeljenja zadataka i nadziranje performansi. Kroz paralelno množenje matrica testirani su kapaciteti klastera računala, a rezultati pokazuju značajnu uštedu vremena obrade uz povećanje broja radnih čvorova. Glavne prednosti rješenja uključuju fleksibilnost, skalabilnost i stabilnost sustava. Zaključno, implementirano rješenje potvrđuje korisnost virtualnih klastera računala u modernom računarstvu, osobito u okruženjima koja zahtijevaju dinamičku raspodjelu resursa.

Ključne riječi: programski alati, računalni čvor, računalni klasteri, virtualizacija

ABSTRACT

VIRTUAL COMPUTING CLUSTERS

Considering the growing needs for large computing resources and complex workloads, virtual computing clusters are becoming a good solution for optimizing computing power and resource management. This paper presents the creation and implementation of a virtual computing cluster using tools such as VirtualBox, SSH, NFS, MPI and Ganglia, with the aim of ensuring efficient task sharing and performance monitoring. Computer cluster capacities were tested through parallel matrix multiplication, and the results show a significant saving in processing time with an increase in the number of worker nodes. The main advantages of the solution include flexibility, scalability and system stability. In conclusion, the implemented solution confirms the usefulness of virtual computer clusters in modern computing, especially in environments that require dynamic resource allocation.

Keywords: computer clusters, computing node, software tools, virtualization