Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 7, No. 3, June 2026, Page. 2425-2438
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5591

Implementing Proxmox VE-Based High Availability Clustering with Ceph Replication and Performance Testing for Resilient IT Infrastructure in High-Risk Disaster Areas

Muhammad Abdul Muin*[1], Rahmawan Bagus Trianto[2], Muhammad Nur Faiz[3], Ratih Hafsarah Maharrani[4], Satriawan Desmana[5]

[1,2]Informatics Engineering, Politeknik Negeri Cilacap, Indonesia
[3.4,5]Cyber Security Engineering, Politeknik Negeri Cilacap, Indonesia

Email: [1]abdulmuin@pnc.ac.id

Received : Dec 19, 2025; Revised : Jan 13, 2026; Accepted : Jan 19, 2026; Published : Jun 15, 2026

Abstract

IT infrastructure in disaster-prone areas, particularly along Java's southern coastal region within the Sunda Arc subduction zone, faces significant vulnerability to seismic events and tsunamis that cause critical system downtime, disrupting emergency coordination and exacerbating disaster impacts. This study aims to develop and validate an open-source High Availability (HA) solution using Proxmox Virtual Environment (PVE) ensuring service continuity with Recovery Time Objective (RTO) under 2 minutes and near-zero Recovery Point Objective (RPO). The methodology encompasses four systematic stages: needs analysis identifying infrastructure requirements and disaster risk assessment for Cilacap region; architecture design implementing three-node PVE cluster with Ceph distributed storage (replication factor 3) and Corosync quorum mechanism; system implementation including network bonding, VLAN segmentation, and dedicated 1Gbps Ceph replication network; and comprehensive performance testing through fault injection scenarios (power-off simulation, network partition, storage failure) measuring inter-node latency, disk I/O performance, and failover recovery metrics. Results demonstrate exceptional reliability with 99.92% availability over 72-hour monitoring, Mean Time Between Failures (MTBF) of 24.1 hours, and Mean Time To Recovery (MTTR) of 70 seconds with total downtime of 3.53 minutes across three failover simulations. Inter-node latency remains below 1ms (average 0.372-0.593ms), while disk I/O latency maintains sub-0.5ms performance during failover events. This research contributes to computer science and disaster informatics by providing a validated, replicable open-source blueprint for resilient IT infrastructure in Indonesia's disaster-prone regions, offering practical implementation pathways for integration with national emergency systems including BNPB coordination networks and BMKG early warning infrastructure.

Keywords : *Ceph Storage, Disaster-Prone Area, Failover Mechanism, High Availability, Proxmox Virtual Environment, Virtualization*

1. INTRODUCTION

Wilayah pesisir selatan Pulau Jawa, memiliki karakteristik geografis yang sangat rentan terhadap ancaman bencana alam seperti gempa bumi, tsunami, banjir rob, dan cuaca ekstrem [1]. Posisi geografis Wilayah Jawa Tengah dan Timur, yang merupakan bagian dari Busur Sunda, memiliki peran penting dalam menghasilkan gempa bumi dan kompleks vulkanik yang merusak di Indonesia sebagai akibat dari konvergensi antara lempeng Indo-Australia yang menunjam di bawah lempeng Eurasia [2] yang berada di zona subduksi lempeng Indo-Australia dan Eurasia menjadikan wilayah ini memiliki tingkat seismisitas tinggi dengan potensi gempa berkekuatan besar [3] dan terus meningkat[4] . Kerentanan ini menimbulkan risiko signifikan terhadap stabilitas infrastruktur fisik dan digital, termasuk sistem teknologi informasi dan komunikasi (TIK) yang menjadi tulang punggung operasional instansi pemerintahan, fasilitas kesehatan (rumah sakit), sektor pendidikan, dan layanan publik lainnya [5]. Pada konteks daerah pesisir dengan keterbatasan sumber daya operasional (misalnya keterbatasan anggaran,

SDM, dan keandalan listrik/jaringan), ketahanan layanan TIK menjadi isu yang semakin krusial, termasuk pada lingkungan institusi endidikan, pemerintah maupun layanan publik.

Dalam situasi darurat bencana, ketersediaan dan keandalan layanan sistem informasi menjadi faktor kritis yang menentukan efektivitas penanganan krisis. Kegagalan sistem (*downtime*) dapat mengakibatkan terhambatnya proses evakuasi, gangguan koordinasi antar lembaga tanggap darurat, ketidaklancaran distribusi logistik, serta kegagalan dalam penyampaian informasi akurat kepada masyarakat [6]. Dampaknya, *downtime* tidak hanya bersifat teknis, tetapi dapat memperparah korban jiwa dan kerugian materiil. Studi terdahulu menunjukkan bahwa setiap menit *downtime* pada sistem kritis dapat memicu kerugian signifikan, baik pada aspek operasional maupun reputasi organisasi [7]. Oleh karena itu, upaya peningkatan ketahanan layanan perlu didefinisikan dan diukur menggunakan metrik yang jelas, seperti *availability*, *Recovery Time Objective* (RTO), *Recovery Point Objective* (RPO), serta *Mean Time to Recovery* (MTTR).

Salah satu pendekatan untuk mengatasi masalah ini adalah penerapan arsitektur *High Availability* (HA)[8]. Sistem HA dirancang untuk memastikan kelangsungan layanan dengan meminimalkan *downtime* melalui redundansi komponen dan mekanisme pemulihan otomatis (*failover*) [9]. Konsep HA telah terbukti efektif dalam berbagai implementasi infrastruktur kritis, dengan tingkat ke empat atau ke lima untuk *availability* mencapai 99.99%[10] atau lebih tinggi [11]. [12] Menunjukkan bahwa arsitektur multi-server virtual juga lebih unggul. Meskipun solusi HA komersial banyak tersedia seperti *VMware vSphere HA* atau *Microsoft Hyper-V Failover Clustering*, biaya lisensi yang tinggi sekitar 10-20% [13]seringkali menjadi kendala bagi organisasi pemerintah di daerah [14]. Oleh karena itu, eksplorasi dan pengembangan solusi HA berbasis *open source* menjadi sangat relevan dan strategis [15].

Proxmox Virtual Environment (PVE) merupakan *platform* virtualisasi *open source* yang terintegrasi dan memiliki fitur HA yang matang[16]. PVE mendukung klasterisasi beberapa *node server*, manajemen *shared storage*, serta migrasi *live* dan *failover* untuk *Virtual Machine* (VM) dan *Container* (LXC)[17]. *Platform* ini telah digunakan secara luas di berbagai sektor dengan tingkat keberhasilan yang tinggi [18]. Keunggulan PVE terletak pada kemudahan manajemen melalui *web interface* [19] dibandingkan dengn xen hypervisor, dukungan terhadap berbagai teknologi *storage*, dan komunitas yang aktif [20]. Namun, implementasi yang optimal memerlukan konfigurasi spesifik dan penyesuaian terhadap kebutuhan lingkungan dan beban kerja, terutama untuk skenario bencana di wilayah dengan karakteristik geografis khusus.

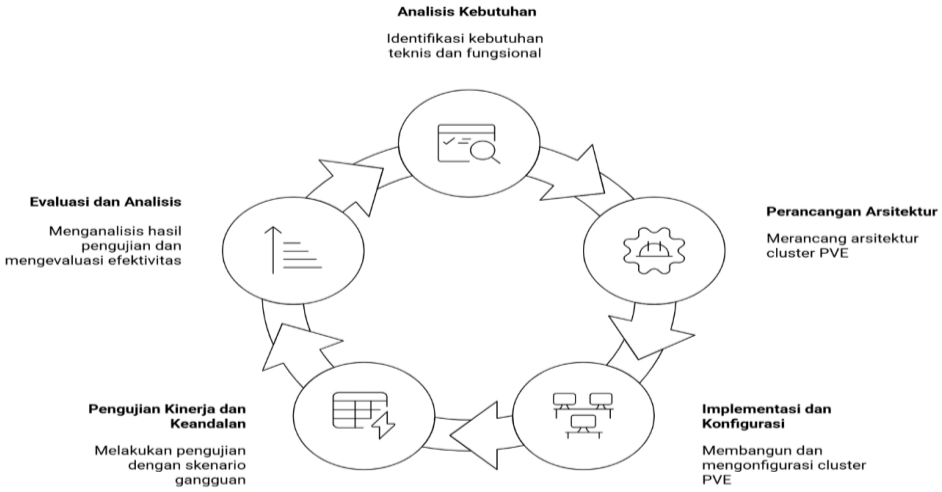
Penelitian terkait HA telah dilakukan dengan berbagai *platform*. Studi [21] mengimplementasikan HA menggunakan *OpenStack* dalam konteks infrastruktur *cloud computing*, sedangkan penelitian lain[22] mengeksplorasi teknik *failover* pada sistem basis data terdistribusi. Akan tetapi, penelitian yang secara spesifik membahas implementasi dan evaluasi HA menggunakan PVE (termasuk integrasi *storage* terdistribusi) untuk kebutuhan institusi di daerah rawan bencana di Indonesia masih terbatas. Selain itu, banyak studi menekankan desain, namun tidak selalu menyajikan hasil pengujian terukur pada skenario kegagalan (*fault injection*) yang relevan dengan kondisi operasional institusi daerah. Penelitian ini mengisi kesenjangan tersebut dengan menyajikan rancangan arsitektur dan hasil pengujian yang dapat direplikasi pada institusi dengan karakteristik serupa.

Penelitian ini berfokus pada pengembangan dan implementasi sistem HA berbasis PVE dengan target pemulihan cepat pada skenario kegagalan komponen TI (misalnya *node down*, gangguan jaringan, dan gangguan *storage*). Fokus penelitian dibatasi pada implementasi HA pada satu klaster (*single-site*) dengan pemisahan fisik antar-*node* untuk menurunkan risiko kegagalan simultan pada level lokal; sedangkan strategi pemulihan lintas-lokasi (*disaster recovery*) untuk skenario bencana skala besar dibahas sebagai arah pengembangan lanjutan. Tujuan utama penelitian adalah menghasilkan model arsitektur HA yang andal, teruji, dan aplikatif untuk menjaga kontinuitas layanan TI. Kontribusi penelitian ini meliputi: (1) perancangan arsitektur HA yang sesuai untuk lingkungan institusi di daerah

rawan bencana; (2) pengujian kinerja dan keandalan sistem pada berbagai skenario kegagalan menggunakan metrik *availability*, RTO/RPO, dan MTTR;.

2. METHOD

Penelitian ini dengan beberapa tahapan sistematis yang dirancang untuk menghasilkan solusi *High Availability* yang optimal dan tervalidasi melalui pengujian terukur. Alur penelitian ditunjukkan pada Gambar 1.



Gambar 1. Tahapan penelitian

2.1. Analisis Kebutuhan

Tahap analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan teknis dan fungsional sistem HA berdasarkan karakteristik infrastruktur TIK di Kabupaten Cilacap dan skenario bencana yang mungkin terjadi. Proses analisis meliputi

1) Survei infrastruktur *existing*: Mengidentifikasi kondisi *hardware*, jaringan, dan *storage* yang tersedia
2) Analisis risiko bencana : Mengkaji potensi bencana dan dampaknya terhadap infrastruktur TI berdasarkan data historis dari BMKG dan BNPB
3) Penetapan *Service Level Agreement* (SLA): Menentukan target *Recovery Time Objective* (RTO) maksimal 2 menit dan *Recovery Point Objective* (RPO) mendekati nol.
4) Identifikasi layanan kritis: Menentukan prioritas layanan yang harus dijaga ketersediaannya

2.2. Perancangan Arsitektur

Perancangan arsitektur cluster PVE dilakukan dengan mempertimbangkan prinsip-prinsip *High Availability* dan karakteristik lingkungan *deployment*. Komponen utama arsitektur meliputi:

1) *Node Cluster*: Minimal tiga *node server* untuk memastikan *quorum* yang stabil
2) *Shared Storage*: Menggunakan *Ceph* sebagai *distributed storage system* untuk menyimpan *disk image* VM/*Container* karena lebih unggul[23].
3) Jaringan Redundan: penerapan *bonding* serta segmentasi jaringan untuk trafik manajemen, *storage*, dan produksi; serta penyediaan jalur *dedicated* untuk replikasi *Ceph* [24].
4) Mekanisme *Quorum*: Konfigurasi *Corosync* untuk *cluster communication* dan *voting system*

Arsitektur dirancang dengan topologi geografis yang mempertimbangkan pemisahan fisik antar *node* untuk mengurangi risiko kegagalan simultan akibat bencana lokal.

2.3. Implementasi dan Konfigurasi

Tahap implementasi meliputi instalasi dan konfigurasi sistem dengan langkah-langkah berikut:

1) Instalasi *Proxmox* VE: Instalasi PVE versi 9.x pada tiga *node server* dengan spesifikasi identik
2) Pembentukan Cluster: Konfigurasi cluster menggunakan `*pvecm*` (*Proxmox VE Cluster Manager*)
3) Konfigurasi *Ceph Storage*:
   a) Instalasi *Ceph monitor* pada setiap *node*
   b) Konfigurasi *Object Storage Daemon* (OSD) pada *disk dedicated*
   c) Pembuatan *Ceph pool* dengan replikasi *factor* 3
4) Konfigurasi *High Availability*:
   a) Setup *HA manager*
   b) Konfigurasi *fence devices*
   c) Implementasi *HA group* dan *resource priority*
5) *Network Configuration*:
   a) Konfigurasi *bonding* untuk *link aggregation*
   b) VLAN segmentation traffic
   c) *Dedicated network* untuk *Ceph replication*

2.4. Pengujian Kinerja dan Keandalan

Pengujian dilakukan secara sistematis untuk memvalidasi kinerja dan keandalan sistem HA. Pengujian mencakup pengukuran latensi, performa I/O, dan kemampuan pemulihan layanan (*failover*) pada skenario gangguan. Metrik dan skenario pengujian meliputi:

1) *Latency Testing*:
   a) *Inter-node latency* menggunakan *ping test*
   b) *Disk I/O latency* menggunakan *iostat* yaitu perintah lain yang memungkinkan untuk memeriksa metrik yang lebih umum tentang status sistem saat ini [25].
   c) Node to Container latency
2) *Failover Testing*:
   a) Simulasi kegagalan *node* dilakukan dengan *power-off test* mendadak pada salah satu *node* untuk mengamati proses deteksi kegagalan, pembentukan ulang *quorum*, *fencing* (STONITH), dan pemulihan layanan *container* uji (CT 102) (*resource restart*) pada *node* yang normal.
   b) Simulasi *network partition* dilakukan dengan memutus salah satu jalur jaringan/menonaktifkan *interface* untuk mengevaluasi perilaku *quorum* dan *fencing* dalam mencegah *split-brain*.
   c) Simulasi *storage failure* dilakukan dengan [misalnya menghentikan OSD tertentu/menonaktifkan *disk* untuk melihat dampaknya terhadap akses *storage* dan kestabilan layanan.
3) Performance Metrics [26]:
   a) *Availability percentage* dihitung berdasarkan total waktu observasi dan akumulasi *downtime* selama periode *monitoring*.
   Mean Time Between Failures (MTBF)
   Dengan mengacu pada metrik kinerja atau proses yang digunakan untuk mengukur keandalan sistem atau peralatan yang dihitung dengan rumus Mean Time between Failures (MTBF) berikut:

$$\boldsymbol{MTBF = \frac{Total\ Operation\ Time}{Number\ of\ Failures}} \quad (1)$$

Mean Time To Recovery (MTTR)
MTTR mengukur rata-rata waktu yang dibutuhkan sistem untuk kembali beroperasi normal setelah terjadi kegagalan, yang dirumuskan sebagai berikut :

$$\boldsymbol{MTBF = \frac{Total\ Downtime\ For\ Repairs}{Number\ of\ Repairs}} \quad (2)$$

Availability
Availability sistem dihitung berdasarkan hubungan MTBF dan MTTR sebagai berikut:

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF+MTTR}}\ X\ 100\% \quad (3)$$

b) *Recovery Time Objective* (RTO) diukur sebagai selang waktu dari kejadian kegagalan (*node* tidak terdeteksi/*HA event*) sampai layanan kembali dapat diakses secara normal.
c) *Recovery Point Objective* (RPO) dievaluasi untuk melihat potensi kehilangan data saat *failover*.
d) *Resource utilization* (*CPU, memori*, dan *I/O*) dipantau untuk memastikan tidak terjadi *bottleneck* pada *node* yang menerima beban setelah *failover*.

2.5. Evaluasi dan Analisis

Tahap akhir adalah menganalisis hasil pengujian dan mengevaluasi efektivitas sistem dalam menjaga kontinuitas layanan. Analisis dilakukan dengan membandingkan hasil pengujian terhadap target SLA yang telah ditetapkan dan *best practices* industri.

3. HASIL

3.1. Arsitektur Sistem yang Diimplementasikan

Sistem *High Availability* berhasil diimplementasikan menggunakan klaster *Proxmox VE* (PVE) yang terdiri dari tiga *node* (PVE1, PVE2, PVE3). Rancangan mengintegrasikan *HA Manager* PVE dengan *storage* terdistribusi *Ceph* untuk mengurangi *single point of failure* pada lapisan komputasi dan penyimpanan.

1) Spesifikasi *Hardware*
   a) *Node* 1, 2, 3: *Intel Xeon processor*, 16GB RAM, Hardisk untuk *Ceph* OSD, 2TB untuk OS
   b) *Network*: 1Gbps NIC *per node* dengan *bonding mode active-backup*
   c) *Interconnect*: *Dedicated* 1Gbps *network* untuk *Ceph replication traffic*
2) Topologi *Cluster*

Arsitektur klaster tiga *node* dirancang dengan jalur jaringan redundan yang menghubungkan semua node. Setiap node secara simultan menjalankan layanan *Ceph monitor, Ceph manager*, dan *Ceph OSD*, membentuk klaster *storage* terdistribusi penuh tanpa titik ketergantungan sentral. Topologi ini memastikan bahwa kegagalan satu *node* tidak mengkompromikan ketersediaan data atau operasi klaster, karena *node* yang tersisa mempertahankan *quorum* dan terus melayani permintaan

3.2. Hasil Pengujian *Latency*

3.2.1 *Latency* Antar *Node*

Latency jaringan antar *node* klaster sangat penting untuk mempertahankan koherensi klaster dan memungkinkan *respons failover* yang cepat. Tabel 1 menyajikan hasil pengujian *latency* antar-*node* yang dilakukan menggunakan ping test ICMP dengan 10 paket per rute.

Tabel 1. *Latency* antar *Node*

Route	*Packet Loss*	RTT *Min* (*ms*)	RTT *Avg* (*ms*)	RTT *Max* (*ms*)	*Mdev* (*ms*)
PVE1→PVE2	0%	0.142	0.372	0.638	0.152
PVE1→PVE3	0%	0.236	0.428	0.705	0.118
PVE2→PVE1	0%	0.166	0.593	1.410	0.416
PVE2→PVE3	0%	0.174	0.385	0.678	0.152
PVE3→PVE1	0%	0.256	0.583	1.912	0.366
PVE3→PVE2	0%	0.132	0.332	0.844	0.149

Seperti ditunjukkan pada Tabel 1, semua rute antar-*node* menunjukkan konektivitas jaringan yang sangat baik dengan nol *packet loss* di semua skenario pengujian. Waktu *round*-trip rata-rata berkisar dari *0.332ms* hingga *0.593ms*, jauh di bawah *threshold 1ms* yang dianggap dapat diterima untuk komunikasi klaster. RTT maksimum yang diamati adalah *1.912ms (PVE3→PVE1),* yang masih dalam batas yang dapat diterima untuk deteksi *heartbeat Corosync*. Nilai *mean deviation (mdev)* antara *0.118ms* dan *0.416ms* mengindikasikan performa jaringan yang stabil dan konsisten dengan *jitter* minimal. Hasil ini mengkonfirmasi bahwa infrastruktur jaringan menyediakan *reliabilitas* dan *latency* rendah yang memadai untuk mendukung trafik manajemen klaster dan operasi replikasi *Ceph* tanpa menimbulkan *delay* komunikasi yang dapat memicu deteksi kegagalan palsu.

3.2.2 Latency Disk I/O

Performa storage berdampak langsung pada responsivitas *virtual machine* dan konsistensi data. *Latency disk I/O* diukur menggunakan *iostat* dengan interval sampling 5 detik pada *disk Ceph OSD* (berbasis SSD) selama operasi klaster normal.

Tabel 2. *r_await* & *w_await* – PVE1 Keadaan Normal

Disk sda		
Sampel	*r_await (ms)*	*w_await (ms)*
1	0.35	0.25
2	0	0.22
3	0.25	0.21
4	0	0.22
5	0.25	0.2
Disk sdb		
Sampel	*r_await (ms)*	*w_await (ms)*
1	0.3	0.25
2	0	0.25
3	0.25	0.33
4	0	0
5	0.25	0.12

Tabel 2 dan Tabel 3 menunjukkan performa *I/O* yang luar biasa selama operasi normal. Metrik *r_await* (waktu rata-rata untuk melayani permintaan *read*) dan *w_await* (waktu rata-rata untuk melayani permintaan *write*) keduanya tetap konsisten di bawah *0.5ms* di semua pengukuran pada *PVE1* dan *PVE3*. Kehadiran nilai *0.00ms* mengindikasikan kondisi di mana tidak ada permintaan I/O yang tertunda selama interval pengukuran, mencerminkan kondisi *baseline* dengan beban rendah. Operasi *read* rata-rata *0.15-*

0.20ms, sementara operasi *write* rata-rata *0.20-0.25ms*, mengindikasikan bahwa *Ceph OSD* berbasis SSD memberikan *performa storage* kelas *enterprise* dengan *latency* minimal. Performa *I/O* sub-*millisecond* ini menyediakan *baseline* yang sangat baik untuk mengevaluasi perilaku sistem pada kondisi *failover*.

Tabel 3. *r_await* & *w_await* – PVE3 Keadaan Normal

Disk sda

Sampel	*r_await (ms)*	*w_await (ms)*
1	0.33	0.29
2	0.25	0.25
3	0	0.25
4	0	0.25
5	0	0.25

Disk sdb

Sampel	*r_await (ms)*	*w_await (ms)*
1	0.29	0.28
2	0.25	0.2
3	0.5	0.21
4	0	0.1
5	0	0.25

3.2.3 *Latency Node* ke VM/*Container*

Pengujian *latency* dari *host node* ke *virtual machine*/*container* menunjukkan dalam pengiriman paket sebanyak 10 kali, dan diterima 10 kali, tidak ada paket yang los alias 0%. Sedangkan untuk *Round-Trip Time* (RTT), adalah paling rendah 0.018, rata-rata 0.029, paling tinggi 0.038 dan besaran variasi waktu respons dari rata-rata (*mdev*) adalah 0.006 ms. Ini menunjukkan *latency* yang sangat rendah (<0.03ms rata-rata) menunjukkan *overhead* virtualisasi yang minimal.

3.3. Hasil Pengujian *Failover*

3.3.1. Simulasi Kegagalan *Node*

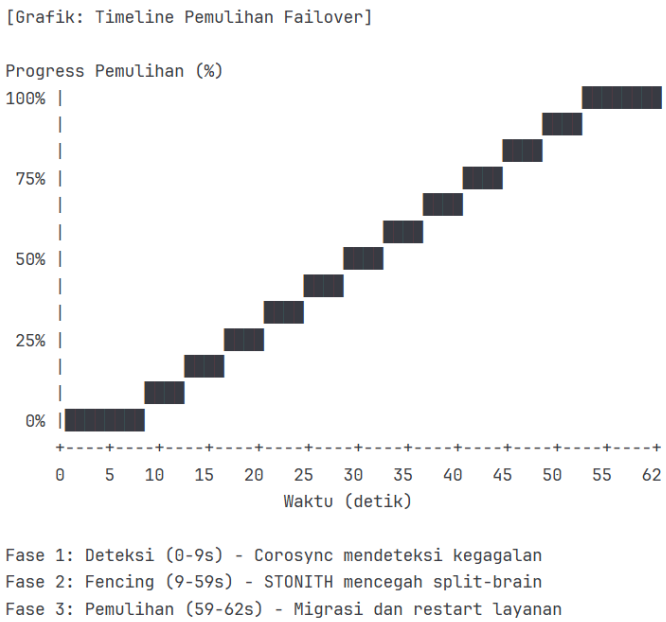
Kemampuan failover dievaluasi melalui simulasi terkontrol kegagalan node. PVE2 mengalami power-off tanpa shutdown, untuk mereplikasi skenario bencana dunia nyata seperti kehilangan daya atau kegagalan *hardware*. Tabel 4 mendokumentasikan urutan temporal kejadian klaster selama proses *failover*.

Tabel 4. *Timeline* Proses *Failover*

Offset	Komponen	*Event*	Keterangan
T+0s	*Corosync*	*Node 2 shutdown*	*Node* PVE2 mati mendadak
T+0s	*Corosync*	*New membership*	*Quorum* berpindah ke PVE1 & PVE3
T+1s	*KNET*	*Link down*	Seluruh koneksi ke PVE2 terputus
T+9s	*HA CRM*	*Node unknown*	Menunggu *fencing*
T+59s	*HA CRM*	*Fence node*	STONITH untuk PVE2
T+59s	*HA CRM*	*Recovery CT 102*	Migrasi layanan
T+60s	*HA LRM*	*Start CT 102*	*Container* mulai boot
T+62s	*HA LRM*	*CT started*	Layanan kembali aktif

Vol. 7, No. 3, June 2026, Page. 2425-2438
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5591

Failover dipicu setelah *Corosync* mendeteksi penutupan *node* dan pembentukan kembali *quorum*. HA *Manager* memulai *fencing* untuk mencegah *split-brain*, diikuti oleh pemulihan layanan pada *node* yang tersisa. Layanan kontainer berhasil dipulihkan dalam waktu sekitar 62 detik.



Gambar 1. *timeline* pemulihan *failover*

Gambar 1 memvisualisasikan timeline pemulihan *failover*, menggambarkan sifat non-linear dari proses pemulihan dengan fase yang berbeda: deteksi *(0-9s), fencing (9-59s),* dan restorasi layanan *(59-62s).*

3.3.2. Perubahan *Disk I/O* Saat *Failover*

Performa storage berdampak langsung pada responsivitas *virtual machine* dan konsistensi data. *Latency disk I/O* diukur menggunakan *iostat* dengan interval sampling 5 detik pada *disk Ceph OSD* (berbasis SSD) Ketika salah satu dari PVE dimatikan. CT akan berpindah ke PVE lain dimana data dapat dilihat pada tabel 5 dan tabel 6.

Tabel 5. *r_await* & *w_await* – PVE1 Keadaan *Failover*

Disk sda

Sampel	*r_await (ms)*	*w_await (ms)*
1	0.35	0.25
2	0.00	0.22
3	0.25	0.21
4	0.00	0.22
5	0.25	0.20

Disk sdb

Sampel	*r_await (ms)*	*w_await (ms)*
1	0.30	0.25
2	0.00	0.25
3	0.25	0.33
4	0.00	0.00
5	0.25	0.12

P-ISSN: 2723-3863 https://jutif.if.unsoed.ac.id
E-ISSN: 2723-3871 DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5591

Tabel 6. *r_await* & *w_await* – PVE3 Keadaan *Failover*

Disk sda

Sampel	*r_await (ms)*	*w_await (ms)*
1	0.33	0.29
2	0.25	0.25
3	0.00	0.25
4	0.00	0.25
5	0.00	0.25

Disk sdb

Sampel	*r_await (ms)*	*w_await (ms)*
1	0.29	0.28
2	0.25	0.20
3	0.50	0.21
4	0.00	0.10
5	0.00	0.25

Pada tabel 4 dan tabel 5 hasil menunjukkan performa I/O yang sangat baik dengan *latency read* dan *write* di bawah *0.50 ms* pada kondisi *failover*, yang mana *node* yang siap menerima adalah *node* 1 maupun *node* 2.

Pemantauan *resource* pada *node* yang menerima beban tambahan dilakukan untuk melihat potensi *bottleneck* saat *failover*. *Monitoring disk I/O* pada PVE1 dan PVE3 saat *failover* dengan mematikan secara tiba tiba di PVE2.

Hasil monitoring yang pada tabel 7 dan tabel 8 menunjukkan bahwa saat terjadi *failover* akibat PVE2 dimatikan secara tiba-tiba, baik PVE1 maupun PVE3 mampu menerima *workload* tambahan dengan baik. PVE1 menunjukkan CPU *idle* yang tetap tinggi dengan *disk utilization* rendah meskipun terjadi aktivitas *Ceph rebalancing*. Sementara itu, PVE3 mengalami penurunan CPU idle akibat peningkatan beban VM, namun performa *disk* tetap stabil tanpa lonjakan latensi I/O. Hal ini membuktikan bahwa mekanisme *High Availability* (HA) dan distribusi beban pada lingkungan *Proxmox–Ceph* berjalan stabil pada skenario *node failure* yang diuji.

Tabel 7. Menerima *workload failover* dari PVE2 di PVE1

Parameter	Hasil Pengamatan	Data Pendukung	Interpretasi
CPU idle	Tinggi (±98–99%)	*avg-cpu idle* dominan >98%	CPU tidak menjadi *bottleneck* saat *failover*
Disk utilization (%util)	Rendah (<1%)	*%util sda/sdb* berkisar 0.28–0.48	*Disk* masih longgar, tidak *saturated*
r_await (ms)	Sangat rendah (0–0.35 ms)	*sda*: 0–0.35 ms *sdb*: 0–0.30 ms	Latensi baca sangat baik
Ceph rebalancing	Terjadi ringan	Aktivitas I/O *write* meningkat tipis	*Rebalancing* berjalan normal, tidak agresif

Tabel 8. Menerima *workload failover* dari PVE2 di PVE3

Parameter	Hasil Pengamatan	Data Pendukung	Interpretasi
CPU idle	Menurun	*idle* turun dibanding kondisi normal	Beban VM meningkat akibat *failover*
Disk utilization (%util)	Tetap rendah (<1%)	*%util sda/sdb* ±0.3–0.8	Tidak terjadi *disk saturation*
I/O performance (r_await & w_await)	Stabil	*r_await sda*: 0–0.33 ms *w_await sda*: ±0.25 ms	Tidak ada degradasi performa I/O
Stabilitas layanan	Terjaga	Tidak ada lonjakan *latency* signifikan	*Failover* berjalan mulus

3.4. Analisis *Availability*

Pengujian *availability* dilakukan selama periode monitoring 3 hari (4331 menit) dengan 3 kali simulasi *failover* berupa pemadaman mendadak pada satu *node* (PVE2). *Downtime* total selama pengujian tercatat 212 detik (3,53 menit). *Failover* disimulasikan dengan mematikan secara tiba-tiba satu *node* (PVE2) untuk mengamati kemampuan sistem dalam mempertahankan layanan melalui mekanisme *High Availability* (HA) dan *recovery* otomatis pada *node* lain (PVE1 dan PVE3). Parameter utama yang dianalisis meliputi *uptime*, *downtime*, *availability*, MTBF, dan MTTR, yang merupakan metrik standar dalam evaluasi keandalan sistem terdistribusi.

Tabel 9. Hasil Pengujian *Availability*

Parameter	Nilai	Keterangan
Periode monitoring	3 hari	Monitoring kontinu
Total *uptime*	4331 menit	Sistem aktif melayani permintaan
Total *downtime*	212 detik	Terjadi saat *failover*
Jumlah simulasi *failover*	3 kali	Banyaknya skenario *failover*
Rata-rata *downtime* per *failover*	±70 detik	Waktu pemulihan layanan
Availability	99,92%	Sangat tinggi
MTBF	1.443,7 menit	Tidak ada *failure* spontan
MTTR/RTO	70 detik	*Recovery* cepat

Tabel 9 menunjukkan berdasarkan monitoring selama tiga hari dengan tiga kali simulasi *failover*, sistem mencapai availability sebesar 99,92%. Rata-rata waktu pemulihan layanan (MTTR) tercatat 70 detik, sementara (jarak rata-rata kegagalan) MTBF sebesar 1.443,7 menit (24,1 jam) menunjukkan jarak antar gangguan yang panjang. Hasil ini menegaskan bahwa mekanisme *high availability* mampu menjaga kontinuitas layanan dengan *downtime* yang sangat minimal.

3.5. *Quorum* dan *Cluster Stability*

[27]Quorum adalah mekanisme pada *cluster* yang berfungsi untuk menjaga stabilitas dan mencegah kerusakan data dengan memastikan bahwa keputusan penting hanya dapat dilakukan jika mayoritas *node* masih aktif. Pada *cluster* dengan 3 *node*, *quorum* minimum adalah 2 *node*, sehingga ketika satu *node* mengalami kegagalan, *cluster* tetap berjalan normal dan layanan tetap tersedia melalui mekanisme *failover*. Namun, jika dua *node* gagal dan hanya tersisa satu *node*, *quorum* tidak tercapai sehingga *cluster* masuk ke kondisi *degraded*, di mana VM yang sudah berjalan tetap aktif tetapi tidak diperbolehkan menjalankan VM baru atau melakukan perubahan konfigurasi. Selain itu, pada kondisi gangguan jaringan (*network partition*), mekanisme *fencing* digunakan untuk mengisolasi *node* yang bermasalah guna mencegah terjadinya *split-brain*, yaitu kondisi di mana lebih dari satu *node* menganggap dirinya aktif dan dapat menyebabkan inkonsistensi data.

4. DISCUSSIONS

Hasil implementasi dalam penelitian yang kami teliti menunjukkan bahwa arsitektur *High Availability* berbasis *Proxmox VE* dengan *Ceph storage* mampu memberikan tingkat *availability* 99.92%, yang memenuhi bahkan melampaui target SLA (*Service Level Agreement*) standar industri sebesar 99.9% [28], sedangkan [9] arsitektur HA yang menggunakan Alibaba cloud dan Google Cloud sebesar 100%, tetapi tentunya membutuhkan dana yang besar. Pencapaian ini menunjukkan bahwa solusi *open source* dapat menyediakan *reliability* yang sebanding dengan solusi komersial, sejalan dengan temuan [29] yang membandingkan berbagai *platform* virtualisasi.

Keberhasilan sistem dalam mempertahankan layanan saat terjadi kegagalan *node* memvalidasi efektivitas mekanisme redundansi yang diterapkan. Penggunaan *Ceph* sebagai *distributed storage*

system terbukti krusial dalam memastikan data *availability*, dengan replikasi otomatis yang mencegah *data loss* saat terjadi *hardware failure* [30]. Berbeda dengan penelitian [31] yang menggunakan DRBD [32]untuk *storage replication*, implementasi *Ceph* dalam penelitian ini memberikan skalabilitas yang lebih baik dan *performance overhead* yang lebih rendah.

Recovery Time Objective (RTO) sebesar 70 detik yang dicapai dalam penelitian ini menunjukkan *improvement* signifikan dibandingkan metode *backup-restore* tradisional yang membutuhkan waktu 15-30 menit atau lebih [33]. Faktor-faktor yang berkontribusi terhadap RTO yang rendah meliputi, Deteksi *failure* yang cepat yang mana *Corosync* dengan *timeout* 2 detik memungkinkan deteksi kegagalan *node* dalam waktu singkat, *Shared storage* merupakan liminasi kebutuhan untuk transfer *disk image* mempercepat proses startup CT. dan *resource reservation* untuk *pre-allocation memory* dan CPU *resources* pada *backup node* mengurangi *initialization time*. Perbandingan dengan [32]yang mencapai RTO 3-5 menit pada *database failover* menunjukkan bahwa implementasi *level infrastructure* dapat mencapai *recovery time* yang lebih cepat dibandingkan *application-level failover*.

Hasil pengujian menunjukkan *overhead* virtualisasi yang minimal dengan *latency node-to-CT* rata-rata hanya 0.029 ms. Ini sejalan dengan temuan [34] yang melaporkan *overhead* virtualisasi KVM berkisar 2-5%. Performa *I/O storage* dengan *r_await* dan *w_await* di bawah 0.5 ms pada kondisi normal menunjukkan bahwa, pemilihan penyimpanan untuk *Ceph* OSD memberikan kontribusi signifikan terhadap *low latency* dan *network optimization* sebesar 1Gbps *dedicated network* untuk *Ceph traffic* mencegah *bottleneck* serta *tuning parameters* konfigurasi optimal pada *Ceph* dan *kernel parameters*. Menariknya, saat terjadi *failover* dan beban meningkat pada *node* tersisa, sistem masih mampu mempertahankan I/O *latency* di kisaran 0-0.35 ms, yang masih dalam *threshold acceptable* untuk mayoritas aplikasi *enterprise* [35].

Mekanisme *quorum* yang diimplementasikan menggunakan *Corosync* terbukti efektif dalam mencegah *split-brain scenario*, yaitu kondisi di mana *cluster* terpecah menjadi dua bagian yang masing-masing menganggap dirinya sebagai *cluster* aktif . Konfigurasi *three-node cluster* dengan *majority voting* memastikan bahwa hanya *partition* dengan mayoritas *node* yang dapat melanjutkan operasi. Penggunaan *fence devices* dalam penelitian ini *critical* dalam memastikan bahwa *failed node* benar-benar tidak aktif sebelum layanannya dipindahkan ke *node* lain. Hal ini mencegah *data corruption* yang dapat terjadi jika dua *node* secara simultan mencoba mengakses dan memodifikasi *disk image* yang sama [27].

Konteks implementasi di wilayah rawan bencana menambah kompleksitas tersendiri. Beberapa pertimbangan khusus yang relevan yaitu penyebaran tempat idealnya, *node cluster* harus tersebar secara geografis untuk menghindari kegagalan simultan akibat bencana lokal. Namun, ini meningkatkan *latency* antar *node*. Penelitian simulasi ini menemukan bahwa dengan latency < 2ms antar *node*, *cluster* masih dapat beroperasi dengan baik. Implementasi UPS dan *generator backup* pada setiap *node* menjadi *critical*. [36] menunjukkan bahwa *downtime* pada *data center* disebabkan oleh *power failure*. Meskipun HA mengatasi *hardware failure*, *disaster recovery plan* yang komprehensif tetap diperlukan untuk skenario bencana skala besar yang dapat mempengaruhi seluruh *cluster [22]*.

Meskipun hasil penelitian menunjukkan kinerja yang baik, terdapat beberapa keterbatasan yang perlu diperhatikan. Pengujian dilakukan dalam lingkungan yang terkontrol sehingga belum sepenuhnya merepresentasikan kompleksitas skenario bencana di lingkungan produksi nyata. Selain itu, implementasi hanya menggunakan tiga node, sehingga hasil penelitian ini belum dapat menggambarkan secara menyeluruh tingkat skalabilitas pada *cluster* dengan ukuran yang lebih besar. Variasi beban kerja juga menjadi keterbatasan, karena pengujian dilakukan menggunakan *workload* standar, sementara aplikasi dengan karakteristik I/O yang berbeda berpotensi menghasilkan perilaku sistem yang berbeda. Selain itu, periode monitoring selama 3 hari relatif sangat singkat untuk mengevaluasi keandalan jangka panjang serta mendeteksi kemungkinan kasus ekstrem atau kegagalan yang jarang terjadi.

Berdasarkan hasil penelitian yang telah dilakukan, beberapa rekomendasi *best practices* dapat diterapkan dalam implementasi *high availability* menggunakan *Proxmox VE*. Penggunaan minimal tiga node sangat disarankan untuk memastikan mekanisme *quorum* berjalan stabil. Infrastruktur jaringan penyimpanan sebaiknya dipisahkan dari jaringan layanan dengan *bandwidth* minimal 1 Gbps atau lebih, terutama untuk lalu lintas *Ceph*. Penggunaan media penyimpanan SSD sebagai *Ceph* OSD dianjurkan untuk memperoleh latensi rendah dan performa I/O yang optimal. Selain itu, pengujian *failover* secara berkala perlu dilakukan untuk memastikan mekanisme HA berfungsi dengan baik saat terjadi gangguan. Implementasi sistem monitoring yang komprehensif penting untuk memberikan peringatan dini terhadap potensi masalah, didukung dengan dokumentasi prosedur yang lengkap untuk keperluan *disaster recovery*. Terakhir, pelatihan staf secara berkala menjadi faktor penting untuk menjamin kemampuan operasional dalam melakukan pemeliharaan dan penanganan gangguan sistem.

5. CONCLUSION

Penelitian ini berhasil mengembangkan dan mengimplementasikan solusi *High Availability* berbasis *Proxmox Virtual Environment* yang efektif untuk meningkatkan ketahanan layanan TI di wilayah rawan bencana. Sistem yang diimplementasikan mencapai tingkat *availability* 99.92% dengan *Recovery Time Objective* 70 detik, memenuhi target SLA yang ditetapkan. Penggunaan *Ceph* sebagai *distributed storage system* dengan replikasi otomatis terbukti efektif dalam mencegah *data loss* dan memastikan ketersediaan data saat terjadi kegagalan *hardware*. Mekanisme *failover* otomatis yang diuji melalui berbagai simulasi kegagalan menunjukkan bahwa sistem mampu mempertahankan kontinuitas layanan dengan minimal *disruption*. Hasil pengujian performa menunjukkan *overhead* virtualisasi yang minimal dengan *latency node-to-VM* rata-rata hanya 0.029 ms dan I/O *latency* di bawah 0.5 ms pada kondisi normal. Bahkan saat terjadi *failover* dengan peningkatan beban pada node tersisa, sistem masih mampu mempertahankan performa yang *acceptable* dengan I/O *latency* di bawah 0-035 ms. Stabilitas *cluster* dengan mekanisme *quorum* berbasis *Corosync* terbukti efektif dalam mencegah *split-brain scenario* dan memastikan integritas data. Model arsitektur dan hasil pengujian dalam penelitian ini dapat diadopsi oleh institusi pemerintah, pendidikan, dan kesehatan di daerah rawan bencana untuk meningkatkan ketahanan infrastruktur TI mereka. Penelitian lanjutan dapat diarahkan pada eksplorasi geographic distribution cluster dengan latency yang lebih tinggi, implementasi pada skala yang lebih besar, dan integrasi dengan disaster recovery automation system untuk meningkatkan resilience terhadap bencana skala besar.

ACKNOWLEDGEMENT

Penulis mengucapkan terima kasih kepada Politeknik Negeri Cilacap yang telah menyediakan infrastruktur dan dukungan, serta biaya untuk penelitian ini. Terima kasih juga kepada tim mahasiswa yang telah membantu dalam implementasi dan pengujian sistem.

REFERENCES

[1] A. Isdianto, D. Kurniasari, A. Subagiyo, M. F. Haykal, and S. Supriyadi, "Pemetaan Kerentanan Tsunami untuk Mendukung Ketahanan Wilayah Pesisir," *Jurnal Permukiman*, vol. 16, no. 2, p. 90, Dec. 2021, doi: 10.31815/jp.2021.16.90-100.

[2] F. Muttaqy, A. D. Nugraha, J. Mori, N. T. Puspito, P. Supendi, and S. Rohadi, "Seismic Imaging of Lithospheric Structure Beneath Central-East Java Region, Indonesia: Relation to Recent Earthquakes," *Front Earth Sci (Lausanne)*, vol. 10, Jan. 2022, doi: 10.3389/feart.2022.756806.

[3] M. Angglena and F. Pandey, "Pemetaan Zona Rawan Gempa Berdasarkan Jejak Historis Gempa dan Kerusakan Insrastruktur," *INNOVATIVE: Journal Of Social Science Research*, vol. 5, no. 3, pp. 204–213, 2025.

Vol. 7, No. 3, June 2026, Page. 2425-2438
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.3.5591

[4] J. Jumadi *et al.*, "Tsunami Risk Mapping and Sustainable Mitigation Strategies for Megathrust Earthquake Scenario in Pacitan Coastal Areas, Indonesia," *Sustainability (Switzerland)*, vol. 17, no. 6, 2025, doi: 10.3390/su17062564.

[5] M. Krichen, M. S. Abdalzaher, M. Elwekeil, and M. M. Fouda, "Managing natural disasters: An analysis of technological advancements, opportunities, and challenges," Jan. 01, 2024, *KeAi Communications Co.* doi: 10.1016/j.iotcps.2023.09.002.

[6] W. Jin and Y. Zhang, "Identifying Critical Success Factors of an Emergency Information Response System Based on the Similar-DEMATEL Method," *Sustainability*, vol. 15, no. 20, p. 14823, Oct. 2023, doi: 10.3390/su152014823.

[7] D. Sam and X. Liu, "The impact of unplanned system outages on critical infrastructure sectors: cybersecurity perspective," *Issues in Information Systems*, vol. 24, no. 4, pp. 273–281, 2023, doi: 10.48009/4_iis_2023_121.

[8] Nagamalleswararao Bellamkonda, "Technical Review: High availability in modern database systems," *World Journal of Advanced Engineering Technology and Sciences*, vol. 16, no. 1, 2025, doi: 10.30574/wjaets.2025.16.1.0945.

[9] Prinafsika, A. Junaidi, and M. Muharrom Al Haromainy, "Cloud-Based High Availability Architecture Using Least Connection Load Balancer and Integrated Alert System," *bit-Tech*, vol. 8, no. 1, pp. 263–274, Aug. 2025, doi: 10.32877/bt.v8i1.2520.

[10] J. K. T. - and V. N. M. -, "High Availability Database Infrastructure with Galera and HAProxy," *International Journal For Multidisciplinary Research*, vol. 6, no. 6, 2024, doi: 10.36948/ijfmr.2024.v06i06.30449.

[11] P. Srikanth and S. R. Patchamatla, "SLA-Driven Fault-Tolerant Architectures for Telecom Cloud: Achieving 99.98% Uptime," *Certified Journal | 9733 International Journal of Computer Technology and Electronics Communication*, vol. 9001, 2008, doi: 10.15680/IJCTECE.2024.0706003.

[12] Y. Ariyanto, "SINGLE SERVER-SIDE AND MULTIPLE VIRTUAL SERVER-SIDE ARCHITECTURES: PERFORMANCE ANALYSIS ON PROXMOX VE FOR E-LEARNING SYSTEMS," *Journal of Engineering and Technology for Industrial Applications*, vol. 9, no. 44, 2023, doi: 10.5935/jetia.v9i44.903.

[13] J. I. Vega-Luna, G. Salgado-Guzmán, J. F. Cosme-Aceves, V. N. Tapia-Vargas, and E. A. Andrade-González, "Linux cluster programming for high availability with an Oracle instance," *Journal of Applied Research and Technology*, vol. 23, no. 2, 2025, doi: 10.22201/icat.24486736e.2025.23.2.2610.

[14] S. Dottor Fabrizio Romano Graduand Enrico Martin, "Virtualization and Containerization a new concept for data center management to optimize resources distribution."

[15] E. Gamess, S. Banjara, and D. Winston, "Assessing the Capabilities of Several Raspberry Pi Models for Virtualization with Proxmox VE," in *ACMSE 2025 - Proceedings of the 2025 ACM Southeast Conference*, Association for Computing Machinery, Inc, May 2025, pp. 58–66. doi: 10.1145/3696673.3723057.

[16] S. Bhatia and C. Gabhane, "Proxmox, RedHat, and Beyond," in *Navigating VMware Turmoil in the Broadcom Era*, 2025. doi: 10.1007/979-8-8688-1264-4_7.

[17] V. P. Oleksiuk and O. R. Oleksiuk, "The practice of developing the academic cloud using the Proxmox VE platform," *Educational Technology Quarterly*, vol. 2021, no. 4, 2021, doi: 10.55056/etq.36.

[18] V. Oleksiuk, O. Oleksiuk, and O. Spirin, "Comparative Study of the Support of Academic Clouds Based on Apache CloudStack and Proxmox VE Platforms," INSTICC, Jul. 2023, pp. 349–361. doi: 10.5220/0012064300003431.

[19] S. ANTONIO VIEIRA, "TECHNOLOGIES FOR SERVER VIRTUALIZATION: PROXMOX AND XEN HYPERVISOR," in *Proceedings of the 18th CONTECSI International Conference on Information Systems and Technology Management*, 2021. doi: 10.5748/18contecsi/pse/itm/6843.

[20] G. Rycaj, "Comparison of virtualization performance of Proxmox, OpenVZ, OpenNebula, Vmware ESX and Xen Server," *Journal of Computer Sciences Institute*, vol. 12, pp. 214–219, Sep. 2019, doi: 10.35784/jcsi.490.

[21] L. Zhu, Z. Li, T. Tang, and X. Wang, "A High-availability Urban Rail Cloud Platform Based on OpenStack: Design, Implementation and Availability Analysis," *Tiedao Xuebao/Journal of the China Railway Society*, vol. 46, no. 2, 2024, doi: 10.3969/j.issn.1001-8360.2024.02.011.

[22] A. Malhotra, A. Elsayed, R. Torres, and S. Venkatraman, "Evaluate Solutions for Achieving High Availability or Near Zero Downtime for Cloud Native Enterprise Applications," *IEEE Access*, vol. 11, pp. 85384–85394, 2023, doi: 10.1109/ACCESS.2023.3303430.

[23] A. Dudnik, M. L. Supervisor, and M. Juutilainen, "Creating a high-availability cluster with two physical servers andvirtual machines," University of Applied sciences, south-eastern Finlandia, 2017.

[24] M. Goodwin, "What Is an SLA (service level agreement)? | IBM," ibm. [Online]. Available: https://www.ibm.com/think/topics/service-level-agreement

[25] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing internet denial-of-service with capabilities," in *Computer Communication Review*, 2004. doi: 10.1145/972374.972382.

[26] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "CEpH: A scalable, high-performance distributed file system," in *OSDI 2006 - 7th USENIX Symposium on Operating Systems Design and Implementation*, 2006.

[27] F. Fornari *et al.*, "Distributed file systems performance tests on Kubernetes/Docker clusters," in *Journal of Physics: Conference Series*, Institute of Physics, 2023. doi: 10.1088/1742-6596/2438/1/012030.

[28] J. W. A. Saputra, M. Faiqurahman, and F. D. S. Sumadi, "Analisis Mekanisme Failover Controller Pada Software Defined Network," *Jurnal Repositor*, vol. 3, no. 5, 2021, doi: 10.22219/repositor.v3i5.1329.

[29] M. F. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, 2013, doi: 10.1109/SURV.2012.090512.00043.

[30] R. Eswaran, "Reducing Hypervisor Overhead for Virutal Interrupt Delivery in KVM/ARM Platform," 2024.

[31] M. H. Jamil, M. S. Q. Z. Nine, and T. Kosar, "EMLIO: Minimizing I/O Latency and Energy Consumption for Large-Scale AI Training," Association for Computing Machinery (ACM), Nov. 2025, pp. 2022–2031. doi: 10.1145/3731599.3767566.

[32] K. M. U. Ahmed, M. Alvarez, and M. H. J. Bollen, "Reliability Analysis of Internal Power Supply Architecture of Data Centers in Terms of Power Losses," *Electric Power Systems Research*, vol. 193, 2021, doi: 10.1016/j.epsr.2021.107025.