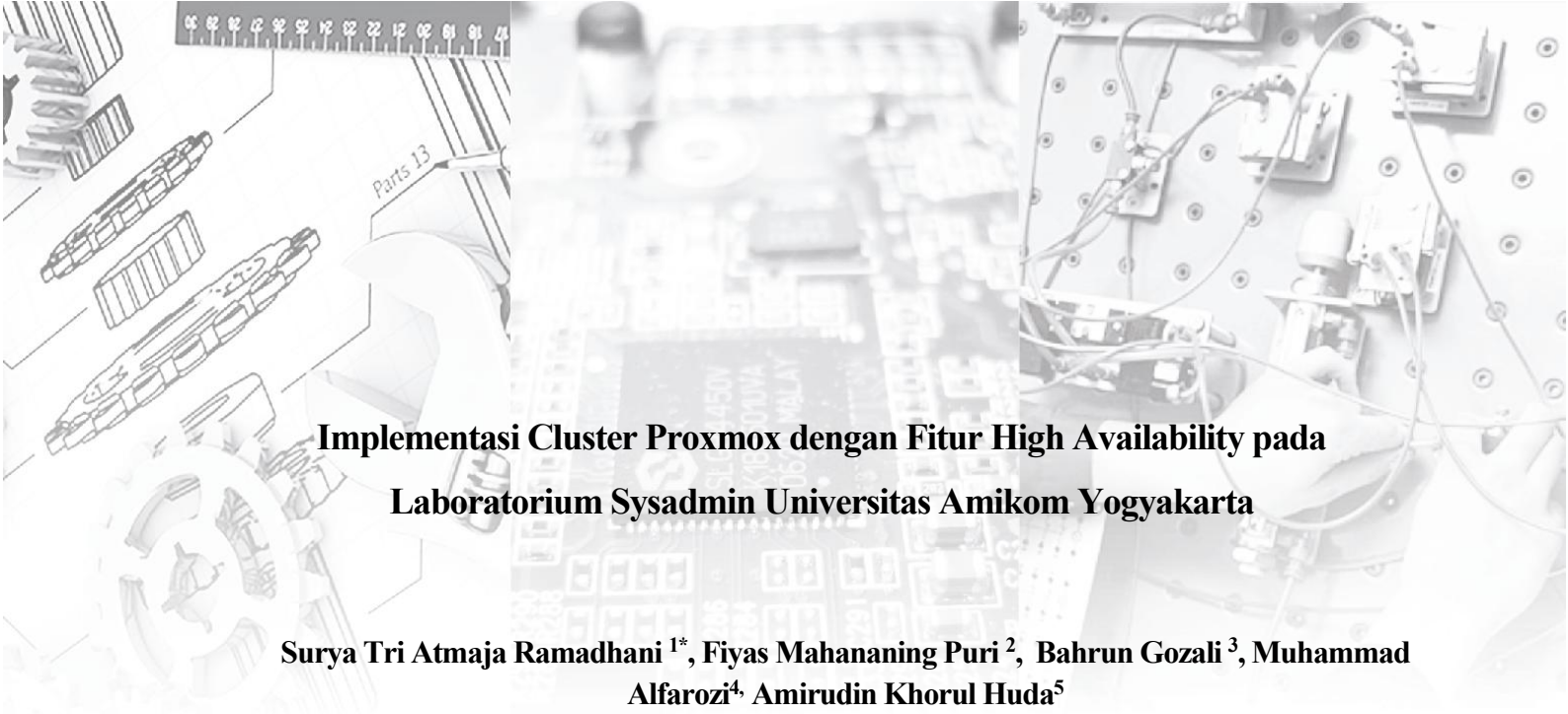


Implementasi Cluster Proxmox dengan Fitur High Availability pada Laboratorium Sysadmin Universitas Amikom Yogyakarta

Surya Tri Atmaja Ramadhani [1*], Fiyas Mahananing Puri [2], Bahrun Gozali [3], Muhammad Alfarozi[4], Amirudin Khorul Huda[5]

[1,3,4]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta, Sleman, Indonesia

[2]Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta, Sleman, Indonesia

[5]Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta, Sleman, Indonesia

Email: surya@amikom.ac.id[1], fiyas@amikom.ac.id[2], oza@amikom.ac.id[3], aalfarozi@students.amikom.ac.id[4], amirudinkh@amikom.ac.id[5]

Abstrak

Ketersediaan tinggi pada infrastruktur virtualisasi kampus krusial untuk menjaga kesinambungan pembelajaran, namun pemilihan backend penyimpanan sering tidak berbasis bukti. Studi ini mengevaluasi ZFS lokal, NFS berbasis jaringan, dan Ceph terdistribusi pada Proxmox VE di dua skenario, kegagalan simpul komputasi dan kegagalan penyimpanan. Metrik layanan didefinisikan sebagai waktu failover sejak deteksi kegagalan hingga VM siap dan downtime yang teramati klien. Eksperimen dilakukan tiga ulangan per kondisi dengan jeda stabilisasi. Inferensi nonparametrik diterapkan, yaitu Kruskal Wallis dengan uji lanjut Dunn terkoreksi Holm, Mann Whitney U, Cliff's delta, serta interval kepercayaan 95 persen berbasis bootstrap. Pada kegagalan simpul, ZFS paling cepat dengan failover 31±1 detik dan downtime 27±1 detik, melampaui NFS 45±2 detik dan 39±2 detik serta Ceph 60±2 detik dan 52±2 detik. Perbedaan global signifikan, H(2)=7,20 dan p=0,027, serta ZFS dibanding Ceph tetap signifikan, p=0,0219. Pada kegagalan penyimpanan, Ceph lebih cepat daripada NFS dengan failover 38±1 detik berbanding 47±2 detik dan downtime 33±1 detik berbanding 42±2 detik. Temuan menyarankan pemilihan backend selaras dengan risiko dominan, ZFS untuk kegagalan simpul dan Ceph untuk gangguan penyimpanan. Kebaruan terletak pada evaluasi lintas backend berbasis metrik layanan dengan analisis inferensial dan prosedur pengukuran yang replikatif.

Kata Kunci: Proxmox VE; High Availability; Failover; ZFS; Ceph

ABSTRACT

High availability in virtualized campus infrastructures is vital for learning continuity, yet storage backends are often chosen without evidence on service recovery. This study evaluates ZFS local, NFS network file service, and Ceph distributed on Proxmox VE under two scenarios, compute node failure and storage failure. We measured service metrics defined as failover time from failure detection to VM readiness and downtime perceived by a client. Experiments used three repetitions per condition, stabilization intervals, and nonparametric inference including Kruskal Wallis with Holm corrected Dunn tests, Mann Whitney U, Cliff's delta, and bootstrap confidence intervals. For node failure, ZFS recovered fastest with failover 31±1 s and downtime 27±1 s, outperforming NFS with 45±2 s and 39±2 s and Ceph with 60±2 s and 52±2 s. The global difference was significant with H(2)=7.20 and p=0.027, and ZFS versus Ceph remained significant with Dunn p=0.0219. For storage failure, Ceph was faster than NFS with failover 38±1 s versus 47±2 s and downtime 33±1 s versus 42±2 s, with very large effects. Findings recommend aligning backend choice with dominant risk, ZFS for node failure and Ceph for storage disruptions.

Keywords: *Proxmox VE; High Availability; Failover; ZFS; Ceph*

1. PENDAHULUAN

Ketersediaan tinggi pada infrastruktur virtualisasi menjadi prasyarat layanan digital yang andal, khususnya di lingkungan pendidikan yang menuntut akses berlanjut ke sistem pembelajaran dan komputasi (Ariyanto et al., 2020). Proxmox Virtual Environment menyediakan mekanisme ketersediaan tinggi dengan orkestrasi pemulihan otomatis pada tingkat klaster, namun perilaku pemulihan sangat dipengaruhi oleh karakter backend penyimpanan yang menopang mesin virtual (Oleksiuk & Oleksiuk, 2021). Ceph merepresentasikan penyimpanan terdistribusi berbasis blok dengan replikasi dan pemulihan mandiri, NFS menyediakan layanan berkas jaringan yang sederhana dan mapan, sedangkan ZFS menawarkan integritas data dan kemudahan pengelolaan pada penyimpanan lokal. Sejumlah kajian mutakhir seperti yang dilakukan oleh (Baun et al., 2025), (Franchuk, 2020), (Kucuk et al., 2020), (Oleksiuk & Oleksiuk, 2021) dan (Vakal & Regalado, 2025), umumnya berfokus pada penyetelan performa atau pengukuran throughput dan latensi pada sistem penyimpanan terdistribusi maupun sistem berkas, sementara evaluasi komparatif yang menautkan langsung perilaku failover pada lapisan layanan mesin virtual di atas Proxmox masih terbatas dalam konteks laboratorium kampus.

Permasalahan yang muncul adalah belum tersedianya dasar empiris yang terukur bagi pengelola sistem untuk memilih backend penyimpanan sesuai profil risiko kegagalan yang dominan (Rajković & Antić, 2024). Pilihan yang tidak berbasis data berpotensi menghasilkan waktu pemulihan yang tidak optimal dan durasi tidak tersedianya layanan yang lebih lama dari yang seharusnya (Šimon et al., 2023). Dalam penelitian ini, dua metrik ditekankan agar relevan bagi pengguna akhir. Waktu failover didefinisikan sebagai selang dari deteksi kegagalan hingga layanan mesin virtual kembali siap, sedangkan downtime adalah durasi ketidaktersediaan layanan yang teramati dari sisi klien. Penekanan pada metrik berbasis layanan dimaksudkan untuk menjembatani temuan teknis penyimpanan dengan kualitas pengalaman pengguna.

Tujuan penelitian adalah mengevaluasi dan membandingkan waktu failover serta downtime pada Proxmox dengan tiga backend penyimpanan, yaitu ZFS, NFS, dan Ceph, di bawah dua skenario kegagalan yang lazim terjadi, yaitu kegagalan node komputasi dan kegagalan penyimpanan (Idrus, 2021) (Somasekaram et al., 2021). Pendekatan yang digunakan bersifat eksperimental dengan pengulangan terkontrol dan pencatatan telemetri sumber daya, performa Input dan Output, serta jaringan untuk memperkaya interpretasi. Kebaruan yang ditawarkan terletak pada kerangka evaluasi terukur lintas backend pada konteks kampus yang representatif beserta skenario kegagalan yang dinyatakan secara eksplisit, sehingga hasilnya

tidak hanya menyajikan perbandingan angka tetapi juga menyiratkan solusi berbasis bukti berupa peta rekomendasi pemilihan backend yang selaras dengan jenis kegagalan yang paling kritis di lingkungan target. Batasan yang melekat pada lingkungan uji dicatat agar interpretasi dan generalisasi tetap proporsional dengan konteks penerapan.

2. METODE

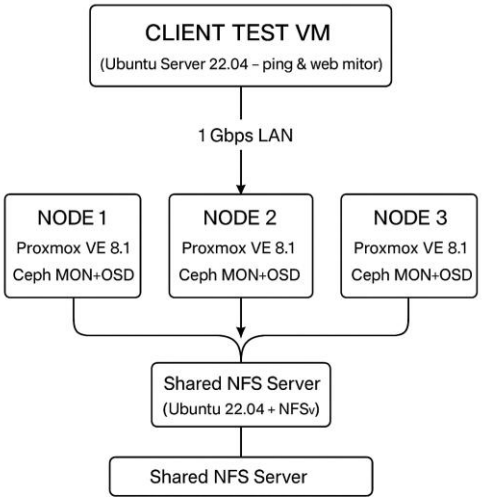
Penelitian ini merupakan studi eksperimental komparatif dengan pendekatan kuantitatif (Syam & Rahman, 2024). Subjek penelitian adalah klaster Proxmox Virtual Environment (Proxmox VE) yang menjalankan mesin virtual homogen pada 3 node dengan spesifikasi yang dapat dilihat pada Tabel 1, dengan tiga backend penyimpanan sebagai perlakuan, yaitu ZFS, NFS, dan Ceph.

Tabel 1. Spesifikasi Server Dari Masing-Masing Node

Bagian Spesifikasi	Keterangan
CPU Tiap Node	Intel Xeon Silver 4110 (8 core, 2.1 GHz)
RAM Tiap Node	32 GB DDR4
Storage Fisik	SSD 500 GB SATA
Jaringan	1 Gbps LAN

2.1 Skenario Pengujian

Dua skenario kegagalan diuji, yakni kegagalan node komputasi dan kegagalan penyimpanan. ZFS hanya dievaluasi pada kegagalan node karena rancangan yang digunakan tidak mereplikasi data lintas node. Setiap skenario dijalankan sebanyak tiga ulangan agar diperoleh ukuran kecenderungan tengah yang lebih representatif. Berikut adalah gambar topologi yang dapat dilihat pada Gambar 1.



Gambar 1. Diagram Topologi

Data dikumpulkan melalui dua sumber utama. Pertama, cap waktu dari log klaster dan pemeriksaan kesiapan layanan pada mesin tamu untuk mengukur waktu failover dan downtime dari sudut pandang klien. Kedua, telemetri klaster untuk memantau pemakaian CPU dan memori selama pemulihan, serta pengukuran kinerja penyimpanan dan jaringan guna melengkapi interpretasi hasil. Analisis data dilakukan secara deskriptif menggunakan rerata dan simpangan baku dari tiga ulangan per kondisi, kemudian ditautkan ke tujuan penelitian untuk menilai perbedaan kecenderungan antar backend dan antar skenario.

2.2 Analisis Statistik Inferensial

Analisis inferensial dilakukan guna memperkuat generalisasi temuan di luar sampel ulangan eksperimen. Untuk skenario kegagalan simpul dengan tiga backend (ZFS, NFS, Ceph), digunakan uji Kruskal–Wallis (Garcia-Perez, 2023) pada median waktu failover dan downtime karena ukuran sampel per kondisi kecil dan asumsi normalitas tidak dijamin. Apabila uji global signifikan (Mathur et al., 2023), dilakukan uji lanjut Dunn dengan koreksi Holm (Agbangba et al., 2024) guna mengendalikan galat ganda. Besaran efek dilaporkan sebagai Cliff's delta (δ) (Bais & van der Neut, 2022) (Kala, 2024) per pasangan dan η^2_H untuk uji Kruskal–Wallis. Untuk skenario kegagalan penyimpanan yang membandingkan dua backend (Ceph vs NFS), digunakan uji Mann–Whitney U (Dehaene et al., 2021) (Nakazono & Hara, 2024) dengan pelaporan median difference Hodges–Lehmann beserta CI 95%. Selain itu, untuk setiap backend pada masing-masing skenario dihitung CI 95% median melalui bootstrap BCa (Tibbe et al., 2022) agar estimasi ketidakpastian lebih stabil pada n kecil. Ambang signifikansi ditetapkan $\alpha = 0{,}05$. Hasil disajikan dalam tabel ringkas yang memuat statistik uji, dan besaran efek sehingga pembaca dapat menilai baik signifikansi maupun relevansi praktis (Tian et al., 2022).

2.3 Validasi Workload Berbasis SLO

Guna menguji relevansi terhadap kebutuhan operasional, dilakukan validasi beban aplikasi nyata (Alharthi et al., 2024) yang merepresentasikan layanan kampus. Aplikasi uji dipilih berupa LMS berbasis Moodle dengan basis data relasional pada VM yang sama dengan lingkungan penelitian(Irfan & Pratama, 2024). Generator beban menggunakan JMeter/Locust dengan skenario transaksi umum pengguna, mencakup login, membuka course, dan mengakses resource/quiz (Prasetia et al., 2022). Service Level Indicators (SLI) yang dipantau adalah availability HTTP 200, throughput permintaan per detik, dan latensi p50/p95/p99 (Nigade et al., 2024). Target SLO ditetapkan p95 latency < 2,5 s dan error rate < 1% pada kondisi normal (Nigade et al., 2024). Prosedur validasi mengikuti alur injeksi kegagalan, pengukuran dasar,

injeksi kegagalan sesuai skenario, pencatatan metrik hingga layanan pulih, masa stabilisasi, dan pengulangan tiga kali per kombinasi backend. RTO (Perri et al., 2022) aplikasi didefinisikan sebagai selang waktu sejak injeksi hingga p95 latency kembali memenuhi SLO dan error rate < 1%. Hasil kemudian dipetakan terhadap metrik infrastruktur sehingga diperoleh gap antara downtime infrastruktur dan RTO aplikasi, serta SLO conformance ratio tiap backend. Pelaporan menyertakan grafik waktu-nyata (*time-series*) SLI selama gangguan dan tabel ringkas pemenuhan SLO per backend.

Keabsahan data dijaga melalui sinkronisasi waktu menggunakan NTP (Lusi et al., 2023), kontrol variabel lingkungan selama pengujian, dan validasi silang antara waktu log klaster dengan hasil pemeriksaan sisi klien. Konsistensi pengukuran diperiksa melalui pengulangan yang seragam dengan jeda stabilisasi, serta peninjauan anomali untuk mencegah bias akibat kondisi sementara yang tidak representatif.

Gambar 2. Diagram Alir Penelitian

Pada Gambar 2 menjelaskan tentang tahapan penelitian ini yaitu :

1. Studi Literatur dengan meninjau riset terdahulu dan dokumentasi Proxmox.
2. Perancangan Arsitektur menentukan topologi cluster, quorum, jaringan, dan storage backend.
3. Implementasi dengan instalasi Proxmox, konfigurasi cluster, *HA Manager*, dan storage penyiapan klaster dan aktivasi kebijakan ketersediaan tinggi.
4. Pengujian meliputi Failover test (node down, storage failure), Storage performance test (ZFS vs NFS vs Ceph), Resource utilization test (CPU, RAM, disk usage), Network performance test (latency, throughput), Security check (isolasi KVM vs LXC). Pada test tersebut dilakukan pengukuran dasar tanpa gangguan, injeksi kegagalan sesuai skenario, pencatatan metrik hingga layanan pulih, masa stabilisasi, dan pengulangan hingga tiga kali untuk setiap kombinasi backend dan skenario sebelum hasil dikompilasi dan diinterpretasikan.
5. Analisis Data menggunakan analisis deskriptif dan komparatif .

3. HASIL DAN PEMBAHASAN

3.1 Perbandingan Waktu Failover dan Downtime

Penelitian ini menilai kinerja pemulihan layanan pada klaster Proxmox Virtual Environment dengan tiga backend penyimpanan, yaitu ZFS, NFS, dan Ceph, melalui dua skenario kegagalan yang relevan bagi operator, yakni kegagalan node komputasi dan kegagalan penyimpanan. Dua metrik layanan digunakan secara konsisten, waktu failover sebagai selang dari deteksi kegagalan hingga layanan mesin virtual kembali siap, dan downtime sebagai durasi ketidaktersediaan layanan dari sisi klien. Hasil diringkas pada Tabel 2 dan Tabel 3 berikut.

Tabel 2. Ringkasan Waktu Failover dan Downtime pada Skenario Kegagalan Node (rata-rata ± SD, n = 3)

Storage Backend	Avg Failover (s)	Avg Downtime (s)
Ceph	60 ± 2	52 ± 2
NFS	45 ± 2	39 ± 2
ZFS	31 ± 1	27 ± 1

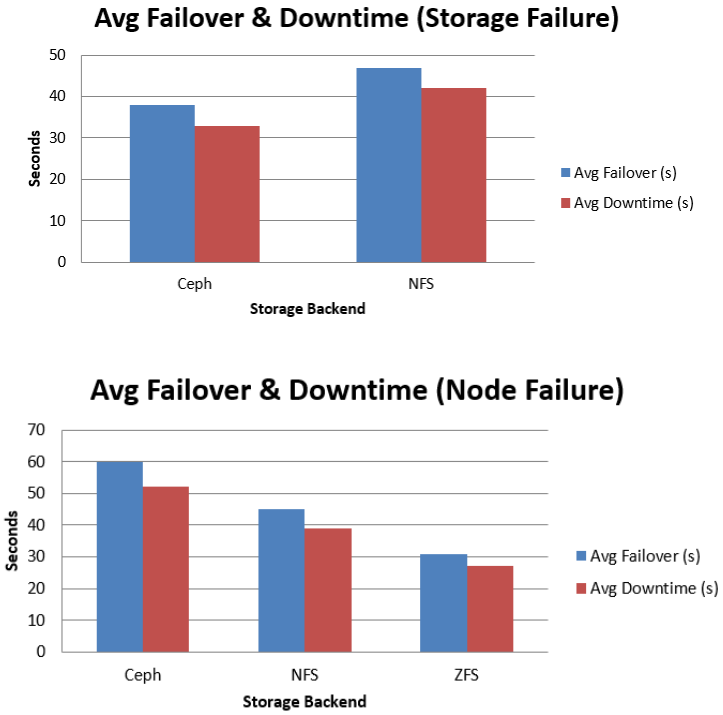
Hasil pada Tabel 2 menunjukkan ZFS pulih paling cepat pada kegagalan node, diikuti NFS, sedangkan Ceph memerlukan waktu lebih panjang. Secara relatif, perbaikan ZFS terhadap NFS untuk metrik failover dihitung dengan Persamaan (1), dan mencapai sekitar 31 persen. Perbaikan ZFS terhadap Ceph mencapai sekitar 48 persen. Pola ini konsisten dengan karakter arsitektural yang memengaruhi jalur data ketika layanan dipindahkan ke node sehat.

$$\text{Peningkatan relatif}\,(\%) = \frac{T_{\text{pembanding}} - T_{\text{usulan}}}{T_{\text{pembanding}}} \times 100 \qquad (1)$$

Tabel 3. Ringkasan Waktu Failover dan Downtime pada Skenario Kegagalan Penyimpanan (rata-rata ± SD, n = 3)

Storage Backend	Avg Failover (s)	Avg Downtime (s)
Ceph	38 ± 1	33 ± 1
NFS	47 ± 2	42 ± 2

Pada kegagalan penyimpanan, Ceph unggul atas NFS dengan selisih sekitar 9 detik pada metrik failover atau sekitar 19 persen lebih cepat berdasarkan Persamaan (1). ZFS tidak dievaluasi pada skenario ini karena rancangan yang digunakan tidak mereplikasi data lintas node, sehingga pemutusan storage tidak mewakili kondisi yang dapat dipulihkan otomatis oleh klaster. Temuan ini sejalan dengan rancangan Ceph yang mengandalkan replikasi dan mekanisme pemulihan mandiri pada tingkat objek atau blok, sehingga kontinuitas layanan lebih terjaga ketika sebagian jalur penyimpanan terganggu.



Gambar 3. Perbandingan Failover dan Downtime antar Backend pada Dua Skenario

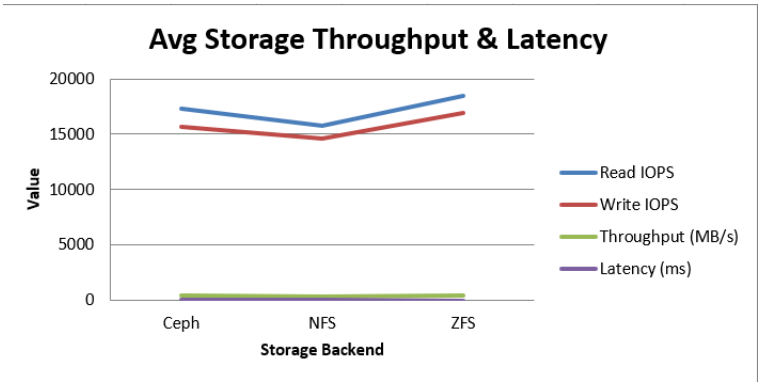
Gambar 3 menampilkan grafik batang perbandingan failover dan downtime untuk ZFS, NFS, dan Ceph pada kegagalan node, serta untuk NFS dan Ceph pada kegagalan penyimpanan. Pengamatan terhadap sumber daya memperlihatkan kenaikan beban selama fase pemulihan. CPU rata-rata pada kondisi normal dan saat failover tercatat untuk Ceph 28 persen dan 55 persen, NFS 20 persen dan 34 persen, serta ZFS 23 persen dan 38 persen. Pola yang sama terjadi pada memori, dengan RAM normal dan saat failover untuk Ceph 44 persen dan 70 persen, NFS 36 persen dan 51 persen, serta ZFS 41 persen dan 56 persen. Kenaikan pada Ceph dapat ditafsirkan sebagai konsekuensi proses koordinasi replikasi dan backfill selama pemulihan, sedangkan ZFS dan NFS cenderung lebih ringan akibat jalur data yang lebih sederhana. Secara praktis, hal ini mengindikasikan kebutuhan penyediaan ruang kepala sumber daya agar proses pemulihan tidak bersaing dengan beban produksi, terutama ketika Ceph digunakan pada lingkungan yang menuntut target RTO ketat.

3.2 Analisis Statistik Inferensial

Berdasarkan tiga ulangan per kondisi, uji Kruskal–Wallis pada skenario kegagalan simpul menunjukkan perbedaan yang signifikan untuk metrik waktu failover dan downtime (failover: $H(2) = 7{,}20$, $p = 0{,}0273$; downtime: $H(2) = 7{,}20$, $p = 0{,}0273$). Uji lanjut Dunn dengan koreksi Holm menegaskan bahwa ZFS lebih cepat dibanding Ceph pada waktu failover

($z = -2{,}68$, $p_{\text{Holm}} = 0{,}0219$, $\Delta_{\text{HL}} = 29{,}00$s, CI95% [28,00; 30,00]) dan pada downtime ($z = -2{,}68$, $p_{\text{Holm}} = 0{,}0219$, $\Delta_{\text{HL}} = 25{,}00$s, CI95% [24,00; 26,00]). Perbandingan ZFS versus NFS serta NFS versus Ceph menunjukkan arah yang konsisten (ZFS < NFS < Ceph) namun tidak signifikan pada $\alpha = 0{,}05$karena ukuran sampel kecil, dengan besaran efek yang tetap sangat besar (Cliff's $\delta = -1{,}00$). Pada skenario kegagalan penyimpanan, uji Mann–Whitney U antara Ceph dan NFS menunjukkan efek praktis besar meski tidak signifikan pada $\alpha = 0{,}05$ karena n kecil ($U = 0{,}00, p = 0{,}10$). Interval kepercayaan median berbasis bootstrap ketat dan selaras dengan ringkasan deskriptif untuk failover dan downtime, mencerminkan efek praktis yang kuat (Cliff's $\delta = -1{,}00$, $\Delta_{\text{HL}} = 9{,}00$s) namun dengan daya uji yang terbatas. Interval kepercayaan median berbasis bootstrap menunjukkan rentang yang ketat untuk masing-masing backend: kegagalan simpul–failover, ZFS [30,00; 32,00], NFS [43,00; 47,00], Ceph [58,00; 62,00]; kegagalan simpul–downtime, ZFS [26,00; 28,00], NFS [37,00; 41,00], Ceph [50,00; 54,00]; kegagalan penyimpanan–failover, Ceph [37,00; 39,00], NFS [45,00; 49,00]; kegagalan penyimpanan–downtime, Ceph [32,00; 34,00], NFS [40,00; 44,00]. Secara inferensial, temuan ini memvalidasi kesimpulan deskriptif: ZFS paling cepat pada kegagalan simpul, sedangkan Ceph unggul pada kegagalan penyimpanan.

3.3 Perbandingan Throughput dan Latency



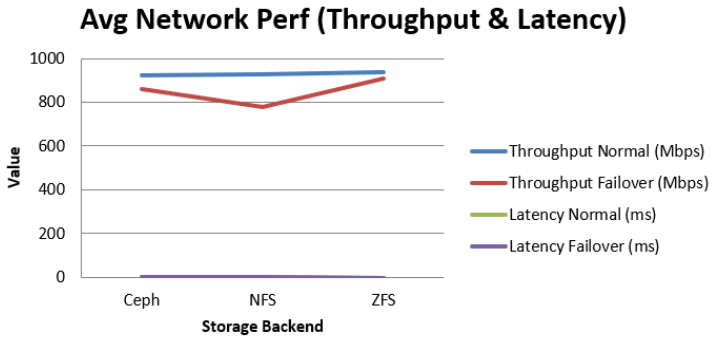
Gambar 4. Throughput dan Latensi Penyimpanan Rata-rata

Tabel 4. Throughput dan Latensi Penyimpanan Rata-rata

Storage Backend	Read IOPS	Write IOPS	Throughput (MB/s)	Latency (ms)
Ceph	17283,33	15683,33	395	2,6
NFS	15800	14583,33	369,67	3,1
ZFS	18450	16883,33	424,33	1,9

Pada Gambar 4 dan tabel 4 ZFS menunjukkan throughput sekitar 424 MB/s dan latensi 1,9 ms, diikuti Ceph 395 MB/s dan 2,6 ms, serta NFS 369,7 MB/s dan 3,1 ms. Pola ini dirujuk

dalam diskusi kinerja Input/Output. Hasil kinerja Input/Output memperlihatkan ZFS memiliki throughput tertinggi dengan latensi terendah, diikuti Ceph kemudian NFS. Urutan ini konsisten dengan temuan pada kegagalan node, yang mengisyaratkan bahwa keberhasilan pemulihan layanan yang cepat pada skenario tersebut turut didukung oleh karakteristik baca tulis yang efisien.



Gambar 5. Throughput dan Latensi Jaringan pada Kondisi Normal dan Saat Failover

Tabel 5. Throughput dan Latensi Jaringan pada Kondisi Normal dan Saat Failover

Storage Backend	Throughput Normal (Mbps)	Throughput Failover (Mbps)	Latency Normal (ms)	Latency Failover (ms)
Ceph	924,67	860	0,48	0,73
NFS	930	780	0,51	0,88
ZFS	939,67	910	0,45	0,62

Gambar 5 dan Tabel 5 menunjukkan throughput normal mendekati 930 hingga 940 Mbps pada seluruh backend. Saat failover throughput menurun dan latensi meningkat, paling besar pada NFS dengan throughput sekitar 780 Mbps dan latensi 0,88 ms, Ceph sekitar 860 Mbps dan 0,73 ms, serta ZFS sekitar 910 Mbps dan 0,62 ms. Secara jaringan, seluruh backend mengalami penurunan throughput serta kenaikan latensi ketika pemulihan berlangsung. Penurunan tertinggi terjadi pada NFS yang bergantung pada kestabilan tautan tunggal ke server berkas, sedangkan ZFS relatif stabil karena pemindahan layanan pada berkas lokal, dan Ceph berada di tengah dengan biaya koordinasi jaringan untuk konsistensi data.

Sintesis keseluruhan menunjukkan bahwa tidak terdapat satu backend yang unggul pada seluruh dimensi. ZFS adalah pilihan paling efisien untuk profil risiko yang didominasi kegagalan node karena waktu pemulihan yang singkat serta beban sumber daya yang moderat. Ceph menjadi opsi lebih defensif untuk profil risiko yang didominasi kegagalan penyimpanan berkat replikasi dan pemulihan mandiri, dengan konsekuensi kebutuhan kapasitas *CPU* dan *RAM* yang lebih tinggi. *NFS* menawarkan implementasi yang sederhana dan kinerja menengah,

namun lebih peka terhadap degradasi jalur jaringan saat pemulihan. Dengan merujuk Tabel 2, Tabel 3, Gambar 3, Gambar 4, Tabel 4, Tabel 5 dan Gambar 5, hasil penelitian ini memperkuat gagasan bahwa pemilihan backend hendaknya diselaraskan dengan profil kegagalan yang paling kritis di lingkungan target, serta kebijakan kapasitas dan arsitektur jaringan yang menopang strategi pemulihan.

3.4 Validasi Operasional Berbasis SLO Aplikasi

Validasi operasional memetakan metrik infrastruktur ke indikator layanan melalui SLO laten p95 dan error rate sebagai SLI utama. Praktik SRE mendefinisikan SLI/SLO pada persentil latensi dan error; implementasi industri kerap memakai ambang p95 ≤ 2,5 s pada time-slice SLO untuk layanan web interaktif. Pada domain LMS, penelitian menunjukkan waktu rebuild cache pascainterupsi dapat mencapai ≈9,8 s pada Moodle jika cache dingin, sehingga RTO aplikasi secara konservatif dapat dihampiri sebagai downtime infrastruktur + overhead warm-up aplikasi ≈ 6–10 s bergantung konten dan kebijakan cache. Selain itu, dokumentasi dan studi performa menunjukkan p95/p99 latensi biasanya terkendali <~2 s pada sistem penyimpanan yang sehat, mendukung kecukupan ambang p95 ≤ 2,5 s untuk layanan pembelajaran berbasis web dengan pemetaan metrik pada Tabel 6 berikut:

Tabel 6. Pemetaan Metrik Infrastruktur ke Indikator Layanan Aplikasi

Skenario	Backend	Downtime (s)	Perkiraan RTO aplikasi (s)	Implikasi SLO p95<2,5 s
Kegagalan simpul	ZFS	27	≈ 37	Pelanggaran SLO selama ~37 s; *recovery* tercepat di skenario ini.
	NFS	39	≈ 49	Pelanggaran SLO lebih panjang; sensitif terhadap *lease time*/retry klien.
	Ceph	52	≈ 62	Pelanggaran SLO terlama pada kegagalan simpul karena koordinasi replikasi
Kegagalan penyimpanan	Ceph	33	≈ 43	*Self-healing* mempercepat pemulihan; SLO pulih lebih awal.
	NFS	42	≈ 52	Rawan *hang* klien jika *lease*/timeout konservatif; atur ulang parameter untuk menekan RTO

Keterbatasan studi ini terletak pada belum disertakannya estimasi total cost of ownership (TCO) per backend meliputi variasi harga perangkat, tarif energi, skema dukungan/lisensi, serta overhead replikasi atau erasure coding antar lingkungan membuat perbandingan biaya tidak dapat disajikan secara akurat pada saat ini. Penelitian lanjutan perlu menyusun model TCO terstandar dengan amortisasi CapEx sepanjang umur aset, pemodelan OpEx (energi, pendinginan, dukungan, dan SDM), serta skenario konfigurasi lanjutan (HA NFS aktif–aktif, replikasi ZFS, dan parameter Ceph seperti size/min_size, PG autoscaling, atau

erasure coding) sehingga rekomendasi teknis yang didasarkan pada kinerja dan ketersediaan pada studi ini dapat disejajarkan secara kuantitatif dengan implikasi biaya.

3.5 Komparasi Dengan HA Modern pada Cloud/Container Stack

a. **Tata kelola control plane** (Bang et al., 2023)**,** (Dakic et al., 2024)**.**
   Arsitektur HA Kubernetes disarankan dalam dua pola stacked control plane (etcd serumpun dengan control plane) dan external etcd (dipisah). Pemilihan topologi memengaruhi biaya (jumlah node, load balancer) dan blast radius saat kegagalan. Distribusi control plane lintas zona dengan penyeimbang beban menjadi praktik baku.

b. **Primitif HA pada workload plane** (Park, 2025)**.**
   Untuk ketersediaan aplikasi, Kubernetes menyediakan PodDisruptionBudget (PDB) untuk membatasi voluntary disruption, topology spread constraints untuk menyebar pod antar domain kegagalan, serta StatefulSet untuk identitas pod yang lengket dan ordered update layanan berkeadaan (stateful). Praktik ini memetakan kebutuhan SLO ke kebijakan scheduling dan pemeliharaan klaster secara standar.

c. **Penyimpanan berketahanan di container** (Panichkitkosolkul et al., 2024).
   Pada stack kontainer, Ceph lazim dioperasikan melalui operator untuk menyediakan RBD/cephfs; pengaturan PG autoscaling, size/min_size, dan pemisahan jaringan public/cluster adalah kunci menjaga self-healing tanpa mengorbankan latensi aplikasi saat recovery. Untuk beban berbagi berkas, NFS-HA (Ganesha + Pacemaker/Corosync) dapat menghadirkan active-active multi-head. Pola-pola ini mengafirmasi bahwa ketahanan storage berlapis mendatangkan biaya koordinasi dan jaringan, yang harus dipertimbangkan dalam TCO.

d. **Perspektif reliability engineering** (Khan et al., 2024)**.**
   Kerangka AWS Well-Architected – Reliability Pillar menggambarkan kurva trade-off biaya–RTO antara backup & restore, pilot light/warm standby, hingga multi-site active/active. Prinsip ini relevan saat membandingkan ZFS+replication (mendekati warm standby untuk VM tertentu), NFS-HA (aktif–pasif/aktif–aktif), dan Ceph (ketahanan active-active di lapisan blok/objek). Strategi yang lebih mahal menurunkan RTO/RPO, tetapi menaikkan TCO; pemilihan bergantung SLO dan error budget layanan.

Berdasarkan pada pustaka tersebut jika membahas dalam ruang lingkup hasil eksperimen, ZFS paling efisien untuk risiko kegagalan simpul, Ceph paling tangguh untuk gangguan storage, NFS-HA berada di tengah dengan biaya operasional moderat dan arsitektur

sederhana. Pada skala kecil–menengah, ZFS+replication menekan Biaya/TB, saat kebutuhan ketahanan dan skala tumbuh, Ceph dengan autoscaling PG dan network split akan memberi RTO lebih baik dengan TCO yang kompetitif pada volume lebih besar. Penerapan di stack kontainer mengikuti pola yang sama: PDB, topology spread, dan StatefulSet menjadi fondasi HA aplikasi; pemilihan backend storage menentukan profil biaya vs ketahanan.

4 PENUTUP

Kesimpulan dan Saran

Penelitian ini menyimpulkan bahwa pada skenario kegagalan node, waktu failover rata-rata adalah ZFS 31±1 s, NFS 45±2 s, dan Ceph 60±2 s dengan downtime berturut-turut 27±1 s, 39±2 s, dan 52±2 s, sedangkan pada kegagalan penyimpanan Ceph mencapai 38±1 s dengan downtime 33±1 s dan NFS 47±2 s dengan 42±2 s, sementara ZFS tidak dievaluasi untuk skenario ini. Implikasinya, ZFS lebih efisien untuk risiko dominan kegagalan node, Ceph lebih unggul ketika risiko utama berada pada lapisan penyimpanan, dan NFS menawarkan kinerja menengah dengan kompleksitas rendah, dengan catatan penyediaan ruang kepala CPU dan RAM perlu diperhatikan terutama saat menggunakan Ceph, keterbatasan studi meliputi jaringan satu gigabit, jumlah ulangan yang terbatas, penggunaan beban sintetis, serta tidak dievaluasinya ZFS pada kegagalan penyimpanan sehingga generalisasi temuan dianjurkan untuk lingkungan serupa. Untuk riset selanjutnya disarankan optimasi arsitektur dan jaringan Ceph melalui pemisahan public dan cluster network serta penggunaan antarmuka 10 GbE, eksplorasi parameter replikasi dan placement groups pada Ceph, pengujian NFS dengan skema HA, penerapan replikasi ZFS, peningkatan ukuran sampel beserta analisis statistik dan sensitivitas, pengujian isolasi KVM dan LXC secara terukur, serta evaluasi biaya kepemilikan agar rekomendasi operasional kian komprehensif.

DAFTAR PUSTAKA

Agbangba, C. E., Aide, E. S., Honfo, S. H., & Glele Kakai, R. (2024). On the use of post-hoc tests in environmental and biological sciences. *Heliyon*, *10*, e25131. https://doi.org/10.1016/j.heliyon.2024.e25131

Alharthi, S., Alshamsi, A., Alseiari, A., & Alwarafy, A. (2024). Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions. *Sensors*, *24*(17), 5551. https://doi.org/10.3390/s24175551

Ariyanto, Y., Harijanto, B., Firdaus, V., & Arief, S. (2020). Performance analysis of Proxmox VE firewall for network security in cloud computing server implementation. *IOP*

Conference Series: Materials Science and Engineering, *732*. https://doi.org/10.1088/1757-899x/732/1/012081

Bais, F., & van der Neut, J. (2022). Adapting the robust effect size Cliff's delta to compare behaviour profiles. *Survey Research Methods*, *16*(3), 329–352. https://doi.org/10.18148/srm/2022.v16i3.7908

Bang, J., Kim, C., Byun, E. K., Sung, H., Lee, J., & Eom, H. (2023). Accelerating I/O performance of ZFS-based Lustre file system in HPC environment. *The Journal of Supercomputing*, *79*(7), 7665–7691. https://doi.org/10.1007/s11227-022-04966-7

Baun, C., Kappes, M., Cocos, H.-N., Koch, M., & Petrozziello, M. (2025). *The Virtual Computer Networks Lab: On the Design and Implementation of a Location Independent Networks Laboratory in Higher Education*. 199–207. https://doi.org/10.5220/0013199400003932

Dakic, V., Kovac, M., & Videc, I. (2024). High-Performance Computing Storage Performance and Design Patterns—Btrfs and ZFS Performance for Different Use Cases. *Computers*, *13*(6), 139. https://doi.org/10.3390/computers13060139

Dehaene, H., Ceulemans, E., & Loeys, T. (2021). A Wilcoxon–Mann–Whitney test for latent variables. *Frontiers in Psychology*, *12*, 754898. https://doi.org/10.3389/fpsyg.2021.754898

Franchuk, V. (2020). *Using the Proxmox Web-Based Virtual Environment in pedagogical educational institutions*. https://consensus.app/papers/using-the-proxmox-webbased-virtual-environment-in-franchuk/238cf4d1bea45c40b8b88850ec433d40/

Garcia-Perez, M. A. (2023). Use and misuse of corrections for multiple testing. *Methods in Psychology*, *8*, 100120. https://doi.org/10.1016/j.metip.2023.100120

Idrus, A. (2021). Designing High Availability Cluster Using Server Virtualization on Cloud Storage Services. *IJISTECH (International Journal of Information System & Technology)*. https://doi.org/10.30645/ijistech.v5i2.119

Irfan, R., & Pratama, C. Y. (2024). Improvement of Performance E-Learning Moodle Service in Vocational High School with Optimization of Web Server and Database Server. *Elinvo (Electronics, Informatics, and Vocational Education)*, *9*(1), 52–63. https://doi.org/10.21831/elinvo.v9i1.42878

Kala, Z. (2024). Global sensitivity analysis of structural reliability using Cliff's delta. *Mathematics*, *12*, 2129. https://doi.org/10.3390/math12132129

Khan, A. Q., Matskin, M., Prodan, R., Bussler, C., Roman, D., & Soylu, A. (2024). Cloud storage cost: a taxonomy and survey. *World Wide Web*, *27*, 1–54. https://doi.org/10.1007/s11280-024-01273-4

Kucuk, D., Apriliana, L., & Diana, N. N. (2020). *Server Clustering in Cloud Computing Using Proxmox Based High Availability Method*. https://doi.org/10.31221/osf.io/6ym8f

Lusi, D., Suban Belutowe, Y., Kupang Jl Perintis Kemerdekaan, U. I., Putih, K., & Kupang, K.

(2023). Analisis dan Implementasi Desain Jaringan Hotspot Berbasis Mikrotik Menggunakan Metode NDLC (Network Development Life Cycle) pada Kantor Balai Pelaksanaan Jalan Nasional NTT. *Jurnal Teknologi Informasi*, *7*(1).

Mathur, M. B., VanderWeele, T. J., & Ding, P. (2023). New metrics for multiple testing with correlated outcomes. *Frontiers in Applied Mathematics and Statistics*, *9*, 1151314. https://doi.org/10.3389/fams.2023.1151314

Nakazono, K., & Hara, S. (2024). Computation of the Mann–Whitney effect under parametric survival copula models. *Mathematics*, *12*(10), 1453. https://doi.org/10.3390/math12101453

Nigade, V., Bauszat, P., Bal, H., & Wang, L. (2024). Inference serving with end-to-end latency SLOs over dynamic edge networks. *Real-Time Systems*, *60*, 239–290. https://doi.org/10.1007/s11241-024-09418-4

Oleksiuk, V. P., & Oleksiuk, O. R. (2021). The practice of developing the academic cloud using the Proxmox VE platform. *Educational Technology Quarterly*, *2021*(4), 605–616. https://doi.org/10.55056/etq.36

Panichkitkosolkul, W., Kesamoon, C., & Vesarachasart, S. (2024). Bootstrap Methods for Estimating the Confidence Interval for the Parameter of Zero-Truncated Poisson-Garima Distribution and Their Application. *Vietnam Journal of Science and Technology*, *62*(6), 1173–1184. https://doi.org/10.15625/2525-2518/18056

Park, Y. (2025). Optimal two-stage group sequential designs based on Mann-Whitney–Wilcoxon test. *PLOS ONE*, *20*(2), e0318211–e0318211. https://doi.org/10.1371/journal.pone.0318211

Perri, D., Simonetti, M., & Gervasi, O. (2022). Deploying Efficiently Modern Applications on Cloud. *Electronics*, *11*(3), 450. https://doi.org/10.3390/electronics11030450

Prasetia, K. A., Akbar, S. R., & Primananda, R. (2022). Implementasi Lingkungan Test pada Moodle dengan Apache JMeter. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK)*, *6*(12), 5719–5725. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/11977

Rajković, A., & Antić, M. (2024). An environment for orchestrating containers on a local ProxMox server. *2024 Zooming Innovation in Consumer Technologies Conference (ZINC)*, 179–181. https://doi.org/10.1109/ZINC61849.2024.10579438

Šimon, M., Huraj, L., & Búčik, N. (2023). A Comparative Analysis of High Availability for Linux Container Infrastructures. *Future Internet*, *15*, 253. https://doi.org/10.3390/fi15080253

Somasekaram, P., Calinescu, R., & Buyya, R. (2021). High-availability clusters: A taxonomy, survey, and future directions. *J. Syst. Softw.*, *187*, 111208. https://doi.org/10.1016/j.jss.2021.111208

Syam, A. B., & Rahman, R. (2024). Design and Implementation of Network Security Systems on Virtualized Networks. *ITEJ (Information Technology Engineering Journals)*.

https://doi.org/10.24235/itej.v9i2.128

Tian, W., Yang, Y., & Tong, T. (2022). Confidence intervals based on the difference of medians for independent log-normal distributions. *Mathematics*, *10*(16), 2989. https://doi.org/10.3390/math10162989

Tibbe, T. D., LoPilato, A. C., Strunk, D. R., & Boswell, J. F. (2022). Correcting the bias correction for the bootstrap confidence interval in mediation analysis. *Frontiers in Psychology*, *13*, 810258. https://doi.org/10.3389/fpsyg.2022.810258

Vakal, I., & Regalado, E. (2025). Comparison between common virtualization solutions: Vmware workstation, Hyper-v and Docker. *Campus Research Day Southern Adventist University*, 372–378. https://doi.org/10.5.1.152/16