Perbandingan Performa Cloud Native Storage pada Kubernetes

Muhammad Istiqlal[1], I Made Suartana[2]

[1,2] Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya
[1]muhammadistiqlal.22064@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

Abstrak **-** Kubernetes membutuhkan solusi penyimpanan data persisten yang terintegrasi secara *native*. Rook-Ceph dan Longhorn adalah dua solusi *cloud native storage* terpopuler dengan arsitektur terdistribusi yang berbeda. Penelitian ini membandingkan performa kedua teknologi tersebut di dalam klaster Kubernetes mandiri berbasis RKE2. Pengujian menggunakan *Flexible I/O Tester* (FIO) dan Kubestr melalui lima skenario: *Database* OLTP, *Pure Latency*, *Web Server*, *Data Analytics*, dan *Logging/Backup*. Metrik utama mencakup IOPS, *throughput*, latensi, dan penggunaan sumber daya. Hasil penelitian menunjukkan performa sangat dipengaruhi karakteristik beban kerja. Longhorn mendominasi skenario *mixed workload* (OLTP) dengan lonjakan performa melampaui 500% (549,57 IOPS *read*; 239,05 IOPS *write* vs Rook-Ceph 87,18 IOPS *read*; 38,85 IOPS *write*). Longhorn juga unggul pada latensi penulisan *Pure Latency* (2,62 ms vs 13,04 ms) dan *throughput Logging/Backup* (52,64 MB/s vs 44,94 MB/s). Sebaliknya, Rook-Ceph sangat superior pada skenario *read-heavy* seperti *Web Server* dengan 27.254,95 IOPS dan 106,48 MB/s (vs Longhorn 11.959,81 IOPS; 46,73 MB/s) serta *Data Analytics* dengan *throughput* 582,64 MB/s (vs 258,88 MB/s). Kesimpulannya, keunggulan Rook-Ceph pada operasi pembacaan didukung arsitektur klaster RADOS terdistribusi dan *caching* efisien, meski *overhead* tinggi pada penulisan acak. Longhorn, dengan arsitektur *microservices* pengontrol tunggal, sangat lincah untuk beban kerja transaksional dan penulisan, namun rentan *bottleneck* saat pembacaan paralel berkonkurensi tinggi.

Kata Kunci **- Rook-Ceph, Longhorn, *Cloud Native Storage*, Kubernetes, *Performance*.**

I. PENDAHULUAN

Kubernetes telah menjadi platform orkestrasi kontainer yang dominan dalam pengembangan arsitektur *cloud-native* modern [1]. Namun, sifat kontainer yang sementara (*ephemeral*) mengharuskan adanya solusi penyimpanan data persisten yang terintegrasi secara *native* agar aplikasi *stateful* dapat berjalan secara andal. Dalam ekosistem ini, sistem penyimpanan terdistribusi menjadi solusi utama karena kemampuannya menjamin skalabilitas, ketersediaan tinggi, dan ketahanan terhadap kegagalan.

Dua solusi *cloud native storage* yang mendapat adopsi luas di lingkungan Kubernetes adalah Rook-Ceph dan Longhorn. Kedua teknologi ini menawarkan pendekatan arsitektur yang sangat berbeda. Rook-Ceph, sebagai proyek *graduate* dari *Cloud Native Computing Foundation* (CNCF), dikenal atas fleksibilitas dan skalabilitasnya yang masif di lingkungan yang kompleks. Di sisi lain, Longhorn yang berstatus sebagai proyek inkubasi CNCF hadir dengan arsitektur *lightweight* berbasis *microservices* yang mudah diimplementasikan, sehingga sangat sesuai untuk klaster dengan sumber daya komputasi yang lebih terbatas.

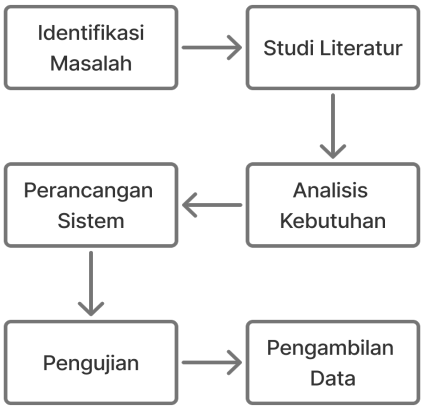
Berbagai penelitian terdahulu telah berupaya mengevaluasi performa media penyimpanan pada lingkungan Kubernetes. Andersson [2] membandingkan performa OpenEBS, Portworx, dan Rook-Ceph pada platform *public cloud* secara spesifik, namun belum menyertakan Longhorn sebagai variabel uji. Studi terpisah oleh Kampadais dkk. [3] difokuskan murni pada optimalisasi mesin *software-defined storage* Longhorn untuk memaksimalkan kapabilitas *hardware* modern, tanpa melakukan komparasi langsung secara ekstensif. Mengingat pesatnya perkembangan ekosistem *cloud-native*, evaluasi komparatif antara teknologi penyimpanan yang mapan dan yang sedang berkembang pesat pada lingkungan *on-premise* yang identik sangat dibutuhkan.

Dalam operasional infrastruktur aplikasi nyata, efisiensi penggunaan sumber daya dan kecepatan menangani beban kerja *Input/Output* (I/O) menjadi tolok ukur utama kualitas layanan. Oleh karena itu, penelitian ini bertujuan untuk membandingkan secara komprehensif performa *block storage* antara Rook-Ceph dan Longhorn. Komparasi kinerja dievaluasi di dalam lingkungan klaster Kubernetes *self-managed* menggunakan distribusi RKE2. Pengujian dilakukan secara sistematis melalui lima skenario simulasi beban kerja menggunakan utilitas *Flexible I/O Tester* (FIO) dan Kubestr guna mengukur metrik krusial seperti IOPS, *throughput*, dan latensi secara objektif.

II. METODOLOGI PENELITIAN

A. Metode Penelitian

Penelitian ini menggunakan metode eksperimental kuantitatif yang dilakukan dalam lingkungan laboratorium infrastruktur terkontrol. Evaluasi difokuskan pada pengukuran metrik *Input/Output* dari dua teknologi *cloud native storage* di dalam klaster Kubernetes [4]. Sebagaimana diilustrasikan pada Gbr 1, alur pelaksanaan penelitian ini disusun secara sistematis yang diawali dari fase identifikasi masalah, perancangan sistem, hingga bermuara pada proses pengambilan data.

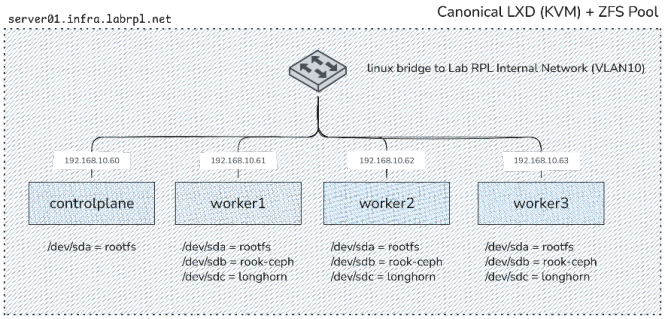


Gbr. 1 *Flow* Penelitian

B. Lingkungan Pengujian dan Topologi

Pengujian dilakukan secara terisolasi pada sebuah klaster Kubernetes *self-managed* menggunakan distribusi RKE2. Infrastruktur fisik disimulasikan menggunakan empat buah *Virtual Machine* (VM), yang dialokasikan menjadi satu node *control plane* dan tiga *worker node*. Guna memastikan konsistensi performa komputasi, setiap VM dibekali spesifikasi yang identik, yakni 8 vCPU dan 8 GB RAM.

Dari sisi topologi penyimpanan, node *control plane* menggunakan satu disk berkapasitas 50 GB khusus untuk sistem operasi (OS). Sementara itu, setiap *worker node* dilengkapi dengan tiga disk terpisah: satu disk 50 GB untuk OS, serta dua *raw disk* berkapasitas masing-masing 30 GB. Kedua *raw disk* ini didedikasikan secara eksklusif untuk *backend* penyimpanan Rook-Ceph dan Longhorn secara mandiri. Rincian mengenai arsitektur jaringan dan pembagian alokasi media penyimpanan pada masing-masing instans tersebut dapat dilihat secara lebih jelas pada Gbr 2 dan Gbr 3.



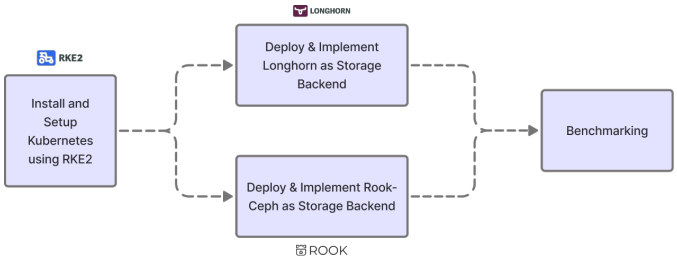
Gbr. 2 Topologi Instance dan Alokasi Storage Klaster Pengujian

```
iqlal@server01:~$ lxc list --project iqlal-research-skripsi -c ns4t | grep -v "flannel"
+--------------+---------+---------------------+-----------------+
|     NAME     |  STATE  |          IPV4       |       TYPE      |
+--------------+---------+---------------------+-----------------+
| controlplane | RUNNING | 192.168.10.60 (enp5s0) | VIRTUAL-MACHINE |
+--------------+---------+---------------------+-----------------+
| worker1      | RUNNING | 192.168.10.61 (enp5s0) | VIRTUAL-MACHINE |
+--------------+---------+---------------------+-----------------+
| worker2      | RUNNING | 192.168.10.62 (enp5s0) | VIRTUAL-MACHINE |
+--------------+---------+---------------------+-----------------+
| worker3      | RUNNING | 192.168.10.63 (enp5s0) | VIRTUAL-MACHINE |
+--------------+---------+---------------------+-----------------+
```

Gbr. 3 List Instance dan IP Address-nya

C. Perancangan Sistem

Alur perancangan sistem dieksekusi secara bertahap, diawali dengan instalasi klaster Kubernetes RKE2 sebagai basis fondasi orkestrasi kontainer. Merujuk pada diagram alur di Gbr 4, setelah klaster RKE2 dipastikan stabil, langkah selanjutnya adalah melakukan proses *deployment* dan implementasi terhadap kedua *backend* penyimpanan secara bergantian sebelum akhirnya memasuki tahap *benchmarking*.



Gbr. 4 Alur Implementasi Cloud Native Storage dan Benchmarking

Pada fase implementasi, Rook-Ceph dikonfigurasi melalui mekanisme Kubernetes Operator yang secara otomatis mengorkestrasi serangkaian *daemon* penyusun *cluster* terdistribusi, seperti OSD, MON, dan MGR [5]. Sebagai komparasi, Longhorn diimplementasikan menggunakan *package manager* Helm. Longhorn secara otomatis men-*deploy Engine controller* dan *Replica* sebagai *microservices* terisolasi yang langsung menuliskan data ke *filesystem* lokal di setiap *worker node* pembentuk klaster [6].

D. Skenario Pengujian

Proses pengambilan data performa mengandalkan utilitas *Flexible I/O Tester* atau FIO [7] yang dijalankan melalui modul pemindai Kubestr. Pengujian dirancang untuk berjalan selama 60 detik menggunakan ukuran berkas data sebesar 1 GB di setiap sesi. Pengujian performa melibatkan lima profil simulasi beban kerja yang mewakili kebutuhan aplikasi di dunia nyata:

1) Database OLTP: mensimulasikan aktivitas basis data transaksional yang padat menggunakan kombinasi operasi acak baca dan tulis dengan rasio 70 banding 30. Parameter yang digunakan meliputi ukuran blok data sebesar 8k serta kedalaman antrean atau *iodepth* sebanyak 64.

2) Pure Latency: berfokus untuk mengukur tingkat latensi murni dari perangkat keras dengan menjalankan operasi baca tulis tunggal secara bergantian tanpa adanya paralelisasi, menggunakan kedalaman antrean *iodepth* bernilai 1.

3) Web Server: menyajikan beban kerja yang berfokus pada aktivitas pembacaan acak penuh atau *random read* 100 persen. Simulasi ini menggunakan ukuran blok kecil sebesar 4k dengan tingkat kepadatan antrean iodepth sebanyak 64.

4) Data Analytics: menguji kecepatan transfer data berskala besar yang berurutan menggunakan skema sequential read 100 persen. Operasi ini menembakkan ukuran blok data masif sebesar 1M dengan tingkat kedalaman antrean *iodepth* 16.

5) Logging Backup: menyerupai karakteristik skenario analitik data namun difokuskan penuh untuk aktivitas penulisan berurutan atau *sequential write* 100 persen. Pengujian ini memakai ukuran blok data sebesar 1M dengan tingkat kedalaman antrean *iodepth* bernilai 16.

E. Pengambilan Data

Fase pengambilan data merupakan tahapan krusial untuk menjamin validitas hasil komparasi antara kedua *cloud native storage*. Ekstraksi data performa media penyimpanan dilakukan secara otomatis dari berkas keluaran berformat JSON yang dihasilkan oleh Kubestr setelah setiap skenario pengujian FIO selesai dieksekusi [7]. Metrik utama yang dicatat meliputi *Input/Output Operations Per Second* (IOPS), *throughput* (kecepatan transfer data aktual), latensi (waktu respons rata-rata), serta *error rate* untuk mengevaluasi tingkat kegagalan operasi di bawah beban ekstrem [7].

Selain performa *Input/Output*, penelitian ini juga memantau dampak komputasi terhadap perangkat keras pada masing-masing *node* secara *real-time*. Pemantauan ini menggunakan pendekatan *observability* yang ringan melalui Prometheus Node Exporter yang di-*deploy* sebagai *DaemonSet* guna mengekspos metrik penggunaan CPU dan memori. Sebagai pengganti Prometheus server dan Grafana, pengumpulan data infrastruktur ditarik secara rutin menggunakan aplikasi *scraper* mandiri berbasis Golang, lalu nilai rata-rata penggunaannya dirangkum untuk kemudian dianalisis.

III. Hasil dan Pembahasan

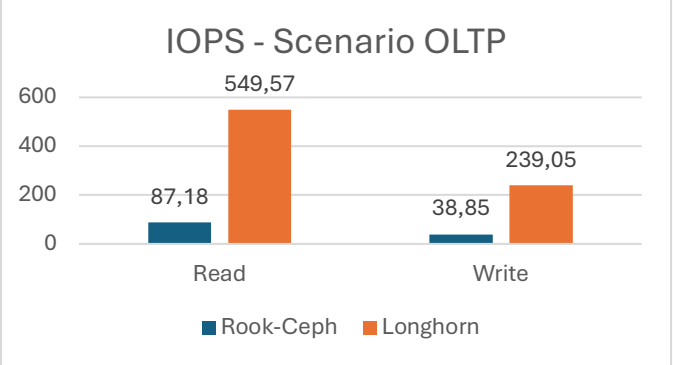
Evaluasi kinerja antara Rook-Ceph dan Longhorn diukur secara komprehensif untuk melihat tingkat IOPS, *throughput*, latensi, dan dampaknya terhadap konsumsi sumber daya klaster. Hasil pengujian menunjukkan bahwa performa kedua *cloud native storage* ini memiliki keunggulan yang sangat spesifik tergantung pada karakteristik beban kerja (*workload*) yang diberikan.

A. Pengujian Performa pada Skenario Database OLTP

Skenario Database OLTP digunakan untuk mensimulasikan karakteristik beban kerja basis data transaksional yang padat, dengan karakteristik kombinasi operasi acak baca dan tulis menggunakan rasio 70:30, ukuran blok 8k, serta *queue depth* (iodepth) sebesar 64.

Tabel I
Tabel Hasil Benchmark IOPS OLTP

Metrics	Parameter	Rook-Ceph	Longhorn	Selisih
IOPS	Read	87.18	549.57	+530.5%
	Write	38.85	239.05	+515.5%
Throughput	Read	0.70 MB/s	4.31 MB/s	+515.7%
	Write	0.31 MB/s	1.87 MB/s	+503.2%



Gbr. 5 Hasil Benchmark OLTP

Berdasarkan Tabel I dan Gbr 5, Longhorn menunjukkan dominasi performa yang sangat signifikan dibandingkan dengan Rook-Ceph. Longhorn berhasil mencatatkan lonjakan performa hingga melebihi 500% pada metrik IOPS dan *throughput* pembacaan maupun penulisan acak.

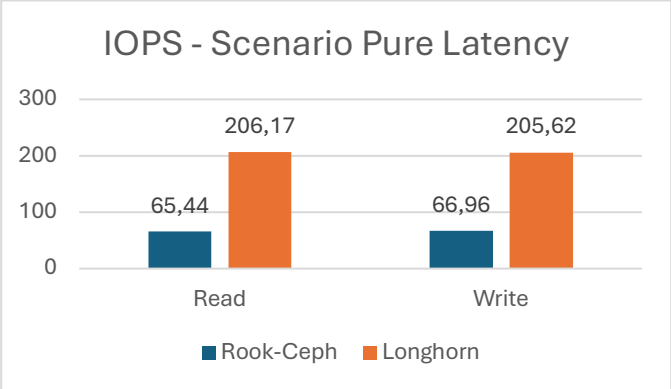
Tingginya performa Longhorn pada skenario ini disebabkan oleh arsitekturnya yang berbasis *microservices* terisolasi (*lightweight*). *Engine controller* Longhorn mampu melakukan bypass banyak lapisan abstraksi dan langsung menuliskan blok data ke dalam *sparse files* pada *filesystem* lokal di node replika. Sebaliknya, Rook-Ceph mengalami fenomena *write amplification* yang parah akibat arsitektur BlueStore yang kompleks. Setiap instruksi penulisan acak pada Ceph mewajibkan adanya pembaruan metadata pada RocksDB, pencatatan transaksi ke dalam Write-Ahead Log (WAL) [8], serta kalkulasi matematis algoritma CRUSH untuk mendistribusikan data antar-OSD, sehingga menciptakan *overhead* komputasi tinggi yang memangkas performa transaksional secara drastis.

B. Pengujian Performa pada Skenario Pure Latency

Skenario *Pure Latency* dirancang untuk mengisolasi dan mengukur waktu respons murni dari mesin penyimpanan tanpa adanya pengaruh dari paralelisasi beban kerja, yang menjadi salah satu parameter penting dalam evaluasi *benchmark cloud native storage* [9]. Pengujian dilakukan menggunakan operasi baca-tulis tunggal dengan nilai *iodepth* sama dengan 1.

Tabel III
Tabel Hasil Benchmark Pure Latency

Metrics	Parameter	Rook-Ceph	Longhorn	Selisih
IOPS	Read	65.44	206.17	+215.1%
	Write	66.96	205.62	+207.1%
Latency	Read	1.91 ms	2.22 ms	+16.2%
	Write	13.04 ms	2.62 ms	−79.9%
Throughput	Read	0.25 MB/s	0.80 MB/s	+220.0%
	Write	0.26 MB/s	0.80 MB/s	+207.7%



Gbr. 6 Hasil Benchmark Pure Latency

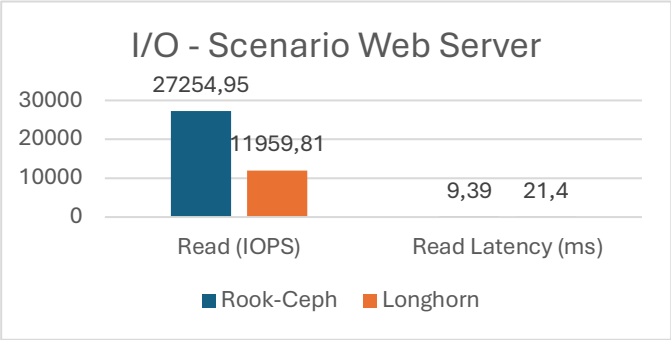
Data pada Tabel II dan Gbr 6 mempertegas karakteristik dasar dari jalur data (*data path*) masing-masing teknologi. Longhorn mencatatkan *write latency* yang sangat rendah (2,62 ms), hampir 5 kali lipat lebih cepat daripada Rook-Ceph yang menyentuh angka 13,04 ms. Hal ini membuktikan minimnya hambatan pemrosesan internal pada Longhorn saat menulis data langsung ke disk lokal. Namun, untuk operasi pembacaan tunggal, Rook-Ceph sedikit lebih unggul (1,91 ms) dibandingkan Longhorn (2,22 ms) berkat optimalisasi *caching* internal yang bekerja di level sub-sistem BlueStore.

C. *Pengujian Performa pada Skenario Web Server*

Skenario *Web Server* merepresentasikan beban kerja yang padat aktivitas pembacaan acak penuh (*100% random read*) menggunakan ukuran blok data kecil (4k) dengan intensitas antrean konkurensi tinggi (*iodepth* 64).

TABEL IIIII
TABEL HASIL BENCHMARK WEB SERVER

Metrics	Parameter	Rook-Ceph	Longhorn
IOPS	Read	27254.95	11959.81
Latency	Read	9.39 ms	21.40 ms
Throughput	Read	106.48 MB/s	46.73 MB/s



Gbr. 7 Hasil Benchmark Web Server

Pada skenario ini, peta kekuatan berbalik secara absolut. Merujuk pada Tabel III dan Gbr 7, Rook-Ceph mendominasi penuh dengan memproses 27.254,95 IOPS dan *throughput* mencapai 106,48 MB/s, berbanding terbalik dengan Longhorn yang tertahan pada 11.959,81 IOPS. Latensi pembacaan Rook-Ceph juga jauh lebih rendah dan stabil di angka 9,39 ms.

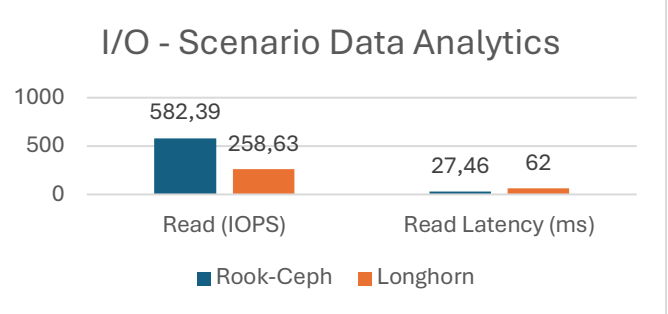
Keunggulan mutlak Rook-Ceph ini didorong oleh implementasi algoritma CRUSH yang memungkinkan klien melakukan query posisi data secara matematis dan langsung membacanya dari banyak Object Storage Daemon (OSD) secara paralel tanpa intervensi centralized metadata server. Sementara itu, Longhorn mengalami bottleneck karena arsitekturnya mewajibkan seluruh arus I/O traffic mengantre melalui satu antarmuka iSCSI (tgtd) pada Engine single-controller, membuat waktu respons membengkak hingga 21,40 ms saat menangani puluhan ribu permintaan baca simultan.

D. *Pengujian Performa pada Skenario Data Analytics*

Skenario *Data Analytics* menguji kemampuan penyimpanan dalam menangani transfer data berskala besar secara berurutan (100% *sequential read*) dengan ukuran blok data masif sebesar 1M, yang karakteristiknya sering ditemui pada infrastruktur data berskala masif [10].

TABEL IVV
TABEL HASIL BENCHMARK IOPS OLTP

Metrics	Parameter	Rook-Ceph	Longhorn	Selisih
IOPS	Read	582.39	258.63	-55.6%
	Read Lat	27.46 ms	62.00 ms	+125.7%
Throughput	Read	582.64 MB/s	258.88 MB/s	-55.6%



Gbr. 8 Hasil Benchmark Web Server

Hasil pengujian pada Tabel IV dan Gbr 8 memperlihatkan *throughput* Rook-Ceph menembus angka 582,64 MB/s, sedangkan Longhorn hanya mampu menghasilkan kecepatan transfer sebesar 258,88 MB/s. Fenomena lonjakan performa sekuensial pada Rook-Ceph ini disokong secara agresif oleh fitur *Adaptive Replacement Cache* (ARC) dan mekanisme *read-ahead*. Mekanisme ini menebak runtutan blok data berikutnya lalu menempatkannya langsung ke dalam memori utama (RAM) sebelum diminta oleh aplikasi, sehingga proses *streaming data* analitik berjalan tanpa hambatan fisik disk.

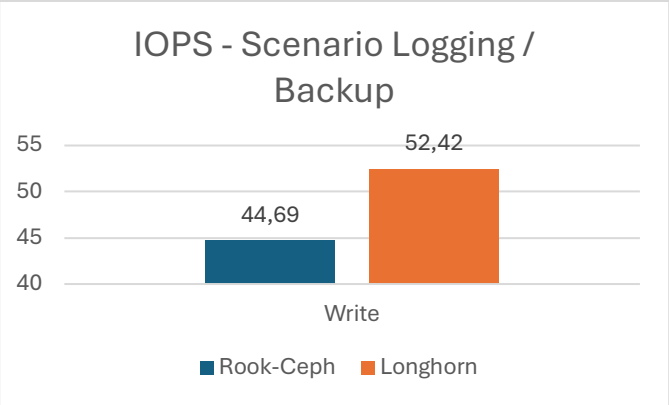
E. *Pengujian Performa pada Skenario Logging dan Backup*

Berkebalikan dengan analisis data, skenario *Logging/Backup* memfokuskan pengujian pada aktivitas

penulisan berurutan penuh (*100% sequential write*) dengan ukuran blok besar (1M) dan tingkat kedalaman antrean *iodepth* 16.

TABEL V
TABEL HASIL BENCHMARK IOPS OLTP

Metrics	Parameter	Rook-Ceph	Longhorn	Selisih (%)
IOPS	Write	44.69	52.42	+17.3 %
Throughput	Write	44.94 MB/s	52.64 MB/s	+17.1 %



Gbr. 9 Hasil Benchmark Web Server

Sesuai dengan hasil *benchmark* pada Gbr 9, pola pada beban kerja penulisan Longhorn kembali mengungguli Rook-Ceph dengan raihan *throughput* sebesar 52,64 MB/s berbanding 44,94 MB/s (Tabel V). Meskipun selisihnya tidak se-ekstrem pada skenario OLTP, hasil ini menegaskan bahwa untuk aktivitas dumping file log aplikasi atau proses backup *image* kontainer secara sekuensial yang sangat menuntut tingkat resiliensi tinggi [11], arsitektur *lightweight engine* milik Longhorn tetap memberikan efisiensi *data path* yang lebih baik daripada mekanisme klaster terdistribusi milik Ceph.

F. Utilisasi Sumber Daya Komputasi

TABEL VV
TABEL HASIL RESOURCES USAGE

Storage Engine	Rata-rata Konsumsi CPU (%)	Rata-rata Konsumsi RAM (%)
Rook-Ceph	5% – 8%	31,94%
Longhorn	5% – 8%	31,57%

Berdasarkan data Tabel VI, tingkat penggunaan CPU dari kedua sistem terpantau sangat stabil dan berada pada rentang yang aman (5% hingga 8%). Hal ini menunjukkan bahwa keterbatasan performa (*bottleneck*) yang terjadi pada skenario-skenario sebelumnya murni disebabkan oleh efisiensi arsitektur perangkat lunak beserta beban *runtime* kontainer [12] dalam memproses operasi I/O (*I/O bound*) dan bukan karena kelangkaan daya komputasi prosesor (*CPU bound*).

Pada metrik memori, worker node yang menjalankan daemon Rook-Ceph (OSD) mengonsumsi RAM yang sedikit lebih tinggi (31,94%) dibandingkan Longhorn (31,57%). Selisih alokasi RAM ini merupakan kompensasi logis dari agresivitas mekanisme *caching* internal Ceph (BlueStore/ARC) yang sengaja memanfaatkan memori utama sebagai penampung sementara demi mendongkrak performa IOPS pembacaan acak.

IV. KESIMPULAN

Analisis performa pada klaster Kubernetes RKE2 menunjukkan bahwa Longhorn memiliki keunggulan dominan pada skenario operasi penulisan (*write-intensive*) dan beban kerja campuran (*mixed*), seperti basis data transaksional (OLTP), *pure latency*, dan *logging/backup*. Pada skenario Database OLTP, Longhorn memicu lonjakan performa melampaui 500% dengan mencatatkan 549,57 IOPS *read* dan 239,05 IOPS *write*, jauh mengungguli Rook-Ceph yang tertahan di angka 87,18 IOPS *read* dan 38,85 IOPS *write*. Jalur pemrosesan data (*data path*) Longhorn yang lebih ringkas tanpa lapisan metadata kompleks juga berhasil memangkas hambatan latensi penulisan pada skenario *Pure Latency* hingga hanya 2,62 ms berbanding 13,04 ms pada Rook-Ceph , serta mencetak *throughput* penulisan sekuensial yang lebih tinggi pada skenario *Logging/Backup* sebesar 52,64 MB/s berbanding 44,94 MB/s. Dengan karakteristik tersebut, Longhorn menjadi solusi yang sangat ideal untuk diterapkan pada klaster berskala kecil hingga menengah atau lingkungan *edge computing* yang menuntut kemudahan operasional dan kecepatan penulisan tinggi.

Sebaliknya, Rook-Ceph terbukti mendominasi secara mutlak pada skenario pengujian yang didominasi oleh operasi pembacaan data intensif (*read-heavy*) dengan tingkat konkurensi tinggi berkat efisiensi distribusi data melalui algoritma CRUSH dan fitur *caching* RAM BlueStore. Pada pengujian skenario *Web Server*, Rook-Ceph mampu melayani beban kerja hingga 27.254,95 IOPS dengan *throughput* 106,48 MB/s dan latensi rendah sebesar 9,39 ms, sementara Longhorn mengalami *bottleneck* antrean yang membuatnya tertahan di angka 11.959,81 IOPS dengan *throughput* 46,73 MB/s dan latensi membengkak ke 21,40 ms. Pada skenario *Data Analytics*, Rook-Ceph juga unggul telak dengan capaian *throughput* pembacaan sekuensial sebesar 582,64 MB/s berbanding 258,88 MB/s milik Longhorn. Perbedaan kinerja ini sejalan dengan konsumsi sumber daya perangkat keras, di mana agresivitas mekanisme *caching* internal Rook-Ceph mengonsumsi RAM rata-rata sedikit lebih besar (31,94%) dibandingkan Longhorn (31,57%) , sedangkan tingkat penggunaan CPU kedua sistem tetap terpantau stabil dan identik di kisaran 5% hingga 8% karena batasan performa murni bersifat *I/O bound*, bukan akibat limitasi kapasitas komputasi prosesor.

REFERENSI

[1] N. Poulton, *The Kubernetes Book 2025*. Nigel Poulton Ltd, 2023.

[2] F. Andersson, "Cloud-Native Storage Solutions For Kubernetes," 2023. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-22919
[3] K. Kampadais, A. Chazapis, and A. Bilas, "Optimizing the Longhorn Cloud-native Software Defined Storage Engine for High Performance," arXiv preprint arXiv:2502.14419, 2025.
[4] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 970-973.
[5] "Rook - Rook Ceph Documentation," Rook.io. [Online]. Available: https://rook.io/docs/rook/latest-release/Getting_Started/intro/.
[6] "Longhorn | The Longhorn Documentation," Longhorn.io. [Online]. Available: https://longhorn.io/docs/1.9.0/.
[7] "Fio - Flexible I/O tester rev. 3.38," ReadTheDocs. [Online]. Available: https://fio.readthedocs.io/en/latest/fio_doc.html.
[8] N. Fisk, *Mastering Ceph - Second Edition*. O'Reilly Media.
[9] A. Merenstein, et al., "CNSBench: A Cloud Native Storage Benchmark," *19th USENIX Conference*, 2021, pp. 263-276.
[10] M. Imran, V. Kuznetsov, P. Paparrigopoulos, S. Trigazis, and A. Pfeiffer, "Evaluation and Implementation of Various Persistent Storage Options for CMSWEB Services in Kubernetes Infrastructure at CERN," Journal of Physics: Conference Series, vol. 2438, no. 1, p. 012035, 2023.
[11] L. Baumann, et al., "Monitoring Resilience in a Rook-managed Containerized Cloud Storage System," *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 89-94.
[12] M. H. Bhattacharya and H. K. Mittal, "Exploring the Performance of Container Runtimes within Kubernetes Clusters," International Journal of Computing, pp. 509-514, 2023.