Analisis Pengaruh Strong Consistency dan Eventual Consistency terhadap Write Latency dan Replication Delay pada MongoDB Replica Set

Wildan Ariel Heradi[1*], Rizqi Nur Hidayatullah[2], Muthia Lutfi Qurata'ain[3], Ivan Noviandra[4], Agustina Srirahayu[5]

[1]Teknik Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa
[1*]240103150@mhs.udb.ac.id

[2] Teknik Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa
[2]240103147@mhs.udb.ac.id

[3] Teknik Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa
[3]240103142@mhs.udb.ac.id

[4]Teknik Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa
[4]240103134@mhs.udb.ac.id

[5]Teknik Informatika/Fakultas Ilmu Komputer
Universitas Duta Bangsa
[5]Agustina@udb.ac.id

Abstrak*— Sistem basis data terdistribusi menjadi solusi untuk meningkatkan skalabilitas, ketersediaan layanan, dan toleransi kegagalan pada pengelolaan data berskala besar. Salah satu implementasinya adalah MongoDB Replica Set yang memanfaatkan replikasi antara node *Primary* dan *Secondary*. Namun, mekanisme replikasi menimbulkan tantangan dalam menjaga konsistensi data tanpa mengurangi performa sistem. Penelitian ini bertujuan menganalisis pengaruh *Strong Consistency* dan *Eventual Consistency* terhadap *write latency* dan *replication delay* pada MongoDB Replica Set. Penelitian menggunakan metode kuantitatif eksperimental pada lingkungan Docker yang terdiri atas satu node *Primary* dan dua node *Secondary*. Data uji dibuat secara otomatis menggunakan Python dengan jumlah 10.000, 50.000, dan 100.000 dokumen. Pengujian dilakukan menggunakan konfigurasi *WriteConcern("majority")* untuk *Strong Consistency* dan *WriteConcern(w=1)* untuk *Eventual Consistency*. Hasil menunjukkan bahwa *Eventual Consistency* menghasilkan *write latency* yang sedikit lebih rendah pada volume 10.000 dokumen, sedangkan *Strong Consistency* memberikan nilai lebih rendah pada volume 50.000 dan 100.000 dokumen. Selain itu, rata-rata *replication delay* sebesar 0,023796 detik menunjukkan proses sinkronisasi data berlangsung stabil. Secara keseluruhan, kedua konfigurasi konsistensi menghasilkan performa yang relatif setara pada lingkungan pengujian. Oleh karena itu, pemilihan tingkat konsistensi lebih dipengaruhi oleh kebutuhan jaminan konsistensi data dibandingkan pertimbangan performa sistem.

Kata kunci— MongoDB Replica Set, Strong Consistency, Eventual Consistency, Write Latency, Replication Delay.

Abstract*— Distributed database systems are a solution to improve scalability, service availability, and fault tolerance in large-scale data management. One implementation is MongoDB Replica Set, which utilizes replication between *Primary* and *Secondary* nodes. However, the replication mechanism poses challenges in maintaining data consistency without reducing system performance. This study aims to analyze the effect of *Strong Consistency* and *Eventual Consistency* on *write latency* and *replication delay* in MongoDB Replica Set. The study uses an experimental quantitative method in a Docker environment consisting of one *Primary* node and two *Secondary* nodes. Test data is automatically generated using Python with a number of 10,000, 50,000, and 100,000 documents. Tests are conducted using the *WriteConcern("majority")* configuration for *Strong Consistency* and *WriteConcern(w=1)* for *Eventual Consistency*. The results show that *Eventual Consistency* produces slightly lower *write latency* at a volume of 10,000 documents, while *Strong Consistency* provides lower values at volumes of 50,000 and 100,000 documents. Furthermore, the average *replication delay* of 0.023796 seconds indicates a stable data synchronization process. Overall, both consistency configurations produced relatively equivalent performance in the test environment. Therefore, the choice of consistency level is more influenced by the need for data consistency assurance than by system performance considerations.

Keywords*— *MongoDB Replica Set, Strong Consistency, Eventual Consistency, Write Latency, Replication Delay.

I. PENDAHULUAN

Perkembangan teknologi informasi telah meningkatkan kebutuhan pengelolaan data dalam jumlah besar sehingga mendorong penggunaan sistem basis data terdistribusi. Sistem ini memungkinkan data disimpan pada beberapa node yang saling terhubung untuk meningkatkan skalabilitas, ketersediaan layanan, dan toleransi terhadap kegagalan dibandingkan sistem basis data terpusat [1]. Namun, pengelolaan data pada lingkungan terdistribusi juga menghadirkan tantangan dalam menjaga konsistensi data antar node.

Konsistensi merupakan salah satu aspek penting dalam sistem terdistribusi karena berkaitan dengan keseragaman data pada seluruh node. Berdasarkan Teorema CAP, sistem tidak dapat secara bersamaan menjamin Consistency, Availability, dan Partition Tolerance secara maksimal, sehingga diperlukan kompromi sesuai kebutuhan sistem yang dibangun [2]. Oleh karena itu, pemilihan model konsistensi menjadi faktor yang dapat memengaruhi performa dan keandalan sistem.

MongoDB merupakan salah satu basis data NoSQL yang mendukung implementasi sistem terdistribusi melalui fitur Replica Set. Arsitektur ini terdiri atas satu node Primary dan beberapa node Secondary yang berfungsi menjaga ketersediaan data melalui mekanisme replikasi. Penelitian sebelumnya menunjukkan bahwa Replica Set mampu meningkatkan ketersediaan layanan dan mendukung proses sinkronisasi data secara efektif [3].

Dua model konsistensi yang umum digunakan pada sistem terdistribusi adalah Strong Consistency dan Eventual Consistency. Perbedaan mekanisme kedua model tersebut dapat memengaruhi performa sistem, terutama pada parameter write latency dan replication delay. Penelitian sebelumnya menunjukkan bahwa tingkat konsistensi dapat memberikan pengaruh yang berbeda terhadap performa basis data NoSQL terdistribusi [4]. Selain itu, peningkatan konsistensi data juga diketahui berkontribusi terhadap kualitas dan keandalan akses data pada sistem NoSQL [5].

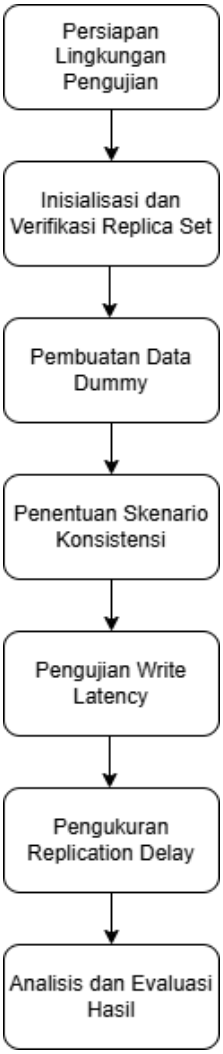
Meskipun berbagai penelitian telah membahas performa MongoDB Replica Set dan konsistensi pada basis data terdistribusi, kajian yang secara khusus membandingkan pengaruh Strong Consistency dan Eventual Consistency terhadap write latency dan replication delay pada MongoDB Replica Set berbasis Docker lokal masih relatif terbatas [6]. Keterbatasan tersebut menjadi dasar dilakukannya penelitian ini.

Berdasarkan kondisi tersebut, penelitian ini bertujuan menganalisis pengaruh Strong Consistency dan Eventual Consistency terhadap write latency dan replication delay pada MongoDB Replica Set. Pengujian dilakukan menggunakan satu node Primary dan dua node Secondary pada lingkungan Docker dengan volume data 10.000, 50.000, dan 100.000 dokumen. Hasil penelitian diharapkan dapat menjadi referensi dalam menentukan strategi konsistensi yang sesuai pada implementasi sistem basis data terdistribusi berbasis MongoDB.

II. METODOLOGI PENELITIAN

Penelitian ini bersifat kuantitatif dengan data yang dikumpulkan langsung melalui serangkaian eksperimen pada sistem MongoDB Replica Set yang dijalankan di dalam lingkungan Docker. Data yang digunakan merupakan data primer, yakni data yang dihasilkan sendiri oleh peneliti melalui proses pengujian langsung. Sebagai bahan uji, peneliti menghasilkan data secara otomatis menggunakan Python dan memasukkannya langsung ke dalam MongoDB Replica Set. Data uji dibuat dalam tiga variasi jumlah dokumen, yaitu 10.000, 50.000, dan 100.000 dokumen. Variasi jumlah data tersebut digunakan untuk mensimulasikan beban kerja yang berbeda sehingga pengaruh tingkat konsistensi terhadap performa sistem dapat diamati secara lebih jelas.

Eksperimen menjadi metode utama dalam pengumpulan data penelitian ini. Sistem yang diuji adalah MongoDB Replica Set dengan susunan satu node Primary dan dua node Secondary. Pengujian dijalankan dalam dua kondisi konsistensi yang berbeda: pertama, Strong Consistency yang memanfaatkan WriteConcern("majority"), dan kedua, Eventual Consistency yang menggunakan WriteConcern(w=1) [7]. Dari kedua kondisi tersebut, peneliti mengamati dua parameter utama, yaitu write latency dan replication delay.



Gambar 1. Tahapan Penelitian

1. *Persiapan Lingkungan Pengujian*

MongoDB Replica Set dikonfigurasi menggunakan Docker, terdiri atas satu node Primary bernama mongo1 serta dua node Secondary bernama mongo2 dan mongo3, ketiganya terhubung dalam satu jaringan virtual lokal. Setelah infrastruktur berhasil berdiri, peneliti melakukan inisialisasi dan pengecekan sistem.

2. *Inisialisasi Replica Set*

Tahapan ini dilakukan menggunakan perintah rs.initiate(), kemudian kondisi sistem diperiksa melalui rs.status() guna memastikan seluruh node telah terhubung dan proses replikasi berlangsung sebagaimana mestinya. Setelah sistem dipastikan berjalan normal, peneliti beralih ke tahap persiapan data uji.

3. *Pembuatan Data Dummy*

Data uji dibuat secara otomatis menggunakan Python dan dimasukkan langsung ke dalam MongoDB Replica Set. Setiap dokumen memiliki struktur data yang seragam sehingga proses pengujian dapat dilakukan secara konsisten pada seluruh skenario. Data yang dihasilkan kemudian digunakan sebagai beban kerja dalam pengujian write latency dan replication delay pada konfigurasi Strong Consistency maupun Eventual Consistency.

4. *Penentuan Skenario Konsistensi*

Skenario pengujian didasarkan pada dua konfigurasi konsistensi, yaitu Strong Consistency dengan WriteConcern ("majority") dan Eventual Consistency dengan WriteConcern(w=1) [8]. Setiap skenario kemudian dijalankan untuk mengukur seberapa cepat sistem menyelesaikan proses penulisan data.

5. *Pengujian Write Latency*

Write latency diukur guna mengetahui durasi yang diperlukan sistem untuk menuntaskan operasi tulis pada masing-masing volume data yang diuji [9]. Agar hasil yang diperoleh lebih dapat diandalkan, setiap skenario diulang sebanyak tiga kali. Di samping write latency, peneliti juga mengamati replication delay sebagai parameter tambahan.

6. *Pengukuran Replication Delay*

Replication delay dihitung berdasarkan jeda waktu antara saat penulisan data selesai di node Primary dengan saat data tersebut mulai tersedia di node Secondary [10]. Seluruh data hasil pengukuran kemudian dikumpulkan untuk dianalisis pada tahap berikutnya.

7. *Analisis dan Evaluasi Hasil*

Pada tahap ini, hasil pengukuran write latency dan replication delay dari konfigurasi Strong Consistency dan Eventual Consistency dibandingkan untuk mengetahui pengaruh masing-masing tingkat konsistensi terhadap performa MongoDB Replica Set. Hasil perbandingan tersebut kemudian dianalisis sebagai dasar dalam penyusunan kesimpulan penelitian.

III. HASIL DAN PEMBAHASAN

A. Persiapan Lingkungan Pengujian

Sebelum proses pengujian dilakukan, lingkungan eksperimen terlebih dahulu dipersiapkan menggunakan Docker Desktop dan MongoDB Community Server. Lingkungan pengujian terdiri atas tiga container MongoDB yang berperan sebagai satu node primary dan dua node secondary dalam konfigurasi Replica Set.

Setelah seluruh container berhasil dijalankan, dilakukan verifikasi untuk memastikan setiap node dapat saling terhubung dan berkomunikasi dengan baik. Tahap ini diperlukan agar pengujian Strong Consistency dan Eventual Consistency dapat dilakukan pada lingkungan yang sesuai dengan skenario sistem terdistribusi.

Keberhasilan persiapan lingkungan pengujian ditunjukkan melalui status container yang aktif serta keberhasilan pembentukan MongoDB Replica Set sebelum proses pengukuran write latency dan replication delay dilakukan.

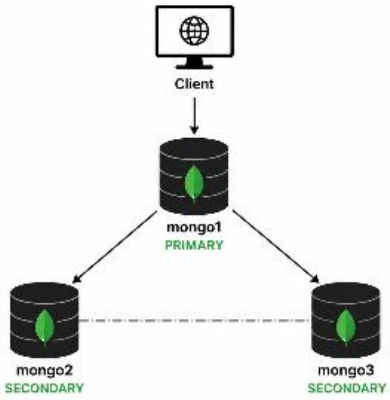
Detail konfigurasi lingkungan pengujian yang digunakan dapat dilihat pada Tabel 1.

Tabel I. Konfigurasi Lingkungan Pengujian MongoDB Replica Set

No	Nama Node	Peran Node	Hostname	Port	Status
1	Mongo1	Primary	Mongo1	27017	Aktif
2	Mongo2	Secondary	Mongo2	27018	Aktif
3	Mongo3	Secondary	Mongo3	27019	Aktif

Berdasarkan Tabel I, seluruh node MongoDB berhasil dijalankan dan berada dalam kondisi aktif. Node mongo1 berperan sebagai Primary, sedangkan mongo2 dan mongo3 berperan sebagai Secondary. Konfigurasi tersebut membentuk MongoDB Replica Set yang digunakan sebagai lingkungan pengujian untuk mengukur write latency dan replication delay pada skenario Strong Consistency dan Eventual Consistency.

Berdasarkan hasil implementasi tersebut, seluruh node terpantau dalam kondisi aktif dan dapat berkomunikasi satu sama lain tanpa kendala. Arsitektur MongoDB Replica Set yang digunakan pada penelitian ini ditunjukkan pada Gambar 2. Dengan konfigurasi tersebut, lingkungan pengujian dinyatakan siap digunakan untuk proses pengukuran write latency dan replication delay.



Gambar 2. Arsitektur MongoDB Replica Set pada Lingkungan Docker

B. Inisialisasi dan Verifikasi Replica Set

Setelah lingkungan pengujian selesai dipersiapkan, tahap berikutnya adalah melakukan inisialisasi MongoDB Replica Set menggunakan perintah rs.initiate(). Proses ini dilakukan untuk menghubungkan ketiga node MongoDB ke dalam satu Replica Set sehingga mekanisme replikasi data dapat berjalan sesuai dengan skenario pengujian yang telah dirancang.

Untuk memastikan proses inisialisasi berhasil, dilakukan verifikasi menggunakan perintah rs.status(). Hasil verifikasi menunjukkan bahwa node mongo1 berhasil terpilih sebagai Primary, sedangkan node mongo2 dan mongo3 berstatus Secondary. Kondisi tersebut menandakan bahwa seluruh node telah tergabung dalam Replica Set yang sama dan mekanisme replikasi telah aktif berjalan dengan baik.

Berdasarkan hasil verifikasi tersebut, tidak ditemukan kendala pada proses konfigurasi maupun komunikasi antar node. Dengan terbentuknya satu node Primary dan dua node Secondary, lingkungan pengujian dinyatakan siap digunakan untuk proses pengujian Strong Consistency dan Eventual Consistency melalui pengukuran write latency dan replication delay.

Tabel II. Output rs.status()

Nama Node	Status
Mongo1	Primary
Mongo2	Secondary
Mongo 3	Secondary

C. Pembuatan Data Dummy

Pada tahap ini, data uji dibuat secara otomatis menggunakan Python dan dimasukkan langsung ke dalam MongoDB Replica Set. Data tersebut digunakan sebagai beban kerja untuk mengukur performa sistem pada berbagai tingkat konsistensi yang diterapkan.

Pengujian dilakukan menggunakan tiga variasi jumlah data, yaitu 10.000, 50.000, dan 100.000 dokumen. Variasi jumlah data tersebut dipilih untuk mensimulasikan peningkatan beban penulisan data sehingga pengaruh Strong Consistency dan Eventual Consistency terhadap performa sistem dapat diamati secara lebih jelas.

Setiap dokumen yang dihasilkan memiliki struktur data yang seragam, sehingga seluruh skenario pengujian dilakukan pada kondisi yang sama. Dengan pendekatan tersebut, perbedaan hasil yang diperoleh lebih mencerminkan pengaruh konfigurasi konsistensi yang digunakan dibandingkan perbedaan karakteristik data.

Data uji yang telah dibuat kemudian digunakan pada proses pengukuran write latency dan replication delay untuk masing-masing skenario konsistensi yang diterapkan pada MongoDB Replica Set.

Tabel III. Variasi jumlah data uji

Skenario	Jumlah Dokumen
Skenario 1	10.000
Skenario 2	50.000
Skenario 3	100.000

Berdasarkan Tabel III, pengujian dilakukan pada tiga tingkat beban yang berbeda untuk melihat perubahan performa sistem ketika jumlah data yang diproses semakin meningkat. Variasi jumlah data tersebut digunakan pada seluruh skenario pengujian sehingga hasil yang diperoleh dapat dibandingkan secara konsisten.

D. Penentuan Skenario Konsistensi

Setelah data uji selesai dipersiapkan, peneliti menerapkan dua konfigurasi konsistensi yang masing-masing akan menjadi skenario dalam pengujian. Konfigurasi pertama menggunakan pendekatan Strong Consistency melalui WriteConcern("majority"), sementara konfigurasi kedua menerapkan Eventual Consistency melalui WriteConcern(w=1). Kedua konfigurasi tersebut diterapkan pada MongoDB Replica Set yang telah berhasil diinisialisasi pada tahap sebelumnya. Penerapan dua konfigurasi yang berbeda memungkinkan pengukuran performa sistem pada tingkat konsistensi yang berbeda sehingga pengaruhnya terhadap write latency dan replication delay dapat diamati secara langsung.

Melalui kedua skenario tersebut, peneliti dapat membandingkan performa MongoDB Replica Set pada kondisi Strong Consistency dan Eventual Consistency.

Tabel IV. Konfigurasi Skenario Konsistensi

Skenario	Konfigurasi
Strong Consistency	WriteConcern(”majority”)
Eventual Consistency	WriteConcern(w=1)

E. Pengujian Write Latency

Pengujian write latency dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan sistem dalam menuntaskan proses penulisan data pada masing-masing konfigurasi, baik Strong Consistency

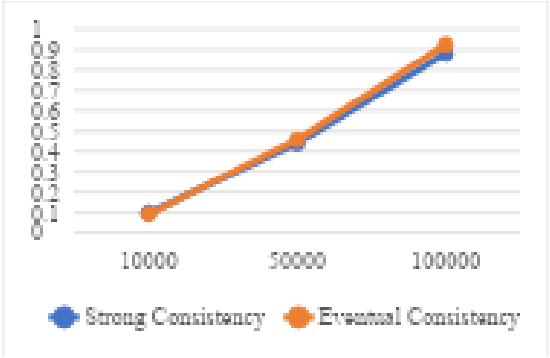
maupun Eventual Consistency. Pengukuran dilakukan pada tiga variasi jumlah data, yaitu 10.000, 50.000, dan 100.000 dokumen. Hasil pengujian write latency ditunjukkan pada Tabel 5.

Tabel V. Hasil Pengujian Write Latency (detik)

Jumlah Data	Strong Consitency	Eventual Consistency
10.000	0.0989	0.0915
50.000	0.4390	0.4575
100.000	0.8804	0.9247

Berdasarkan Tabel V, Eventual Consistency menunjukkan write latency yang sedikit lebih rendah pada pengujian dengan 10.000 dokumen, yaitu sebesar 0,0915 detik dibandingkan Strong Consistency sebesar 0,0989 detik. Namun, pada volume data 50.000 dan 100.000 dokumen, Strong Consistency justru menghasilkan write latency yang sedikit lebih rendah dibandingkan Eventual Consistency.

Meskipun demikian, selisih waktu yang diperoleh pada seluruh skenario pengujian relatif kecil. Hasil tersebut menunjukkan bahwa perbedaan tingkat konsistensi tidak memberikan pengaruh yang signifikan terhadap write latency pada lingkungan MongoDB Replica Set yang digunakan dalam penelitian ini.



Gambar 3. Grafik Perbandingan Write Latency antara Strong Consistency dan Eventual Consistency

Gambar 3 menunjukkan bahwa nilai write latency meningkat seiring bertambahnya jumlah dokumen yang diproses. Pola peningkatan yang ditunjukkan oleh Strong Consistency dan Eventual Consistency terlihat hampir serupa pada seluruh variasi data. Kondisi ini mengindikasikan bahwa peningkatan volume data memiliki pengaruh yang lebih besar terhadap write latency dibandingkan perbedaan konfigurasi konsistensi yang diterapkan pada lingkungan pengujian.

Selain itu, seluruh node MongoDB dijalankan pada satu lingkungan Docker dalam mesin fisik yang sama sehingga latensi komunikasi antar node relatif rendah. Kondisi tersebut menyebabkan perbedaan overhead antara WriteConcern("majority") dan WriteConcern(w=1) tidak terlihat secara signifikan selama proses pengujian berlangsung.

F. Pengukuran Replication Delay

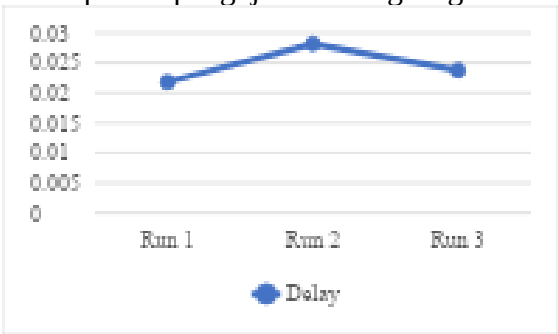
Pengukuran replication delay dilakukan untuk mengetahui seberapa cepat data yang telah berhasil ditulis pada node Primary dapat direplikasi dan tersedia pada node Secondary. Parameter ini digunakan untuk mengevaluasi efektivitas mekanisme replikasi pada MongoDB Replica Set dalam menjaga sinkronisasi data antar node. Hasil pengukuran replication delay ditunjukkan pada Tabel 6.

Tabel VI. Hasil Pengujian Replication Delay

Pengujian	Delay (Detik)
Run 1	0.021871
Run 2	0.028211
Run 3	0.021306
Rata-Rata	0.023796

Berdasarkan Tabel VI, nilai replication delay yang diperoleh berada pada kisaran 0,021 hingga 0,028 detik dengan rata-rata sebesar 0,023796 detik. Nilai tersebut menunjukkan bahwa proses replikasi data dari node Primary ke node Secondary berlangsung dalam waktu yang sangat singkat.

Perbedaan nilai yang muncul pada setiap pengujian juga relatif kecil, sehingga mengindikasikan bahwa mekanisme replikasi berjalan secara stabil. Hasil ini menunjukkan bahwa MongoDB Replica Set mampu menjaga sinkronisasi data antar node dengan baik selama proses pengujian berlangsung.



Gambar 4. Grafik Hasil Pengukuran Replication Delay pada MongoDB Replica Set

Gambar 4 menunjukkan bahwa nilai replication delay pada setiap sesi pengujian cenderung konsisten dan tidak mengalami fluktuasi yang signifikan. Stabilitas tersebut menunjukkan bahwa mekanisme replikasi MongoDB Replica Set mampu mendistribusikan data antar node secara efisien pada lingkungan pengujian yang digunakan.

Rendahnya nilai replication delay yang diperoleh juga menunjukkan bahwa proses sinkronisasi data dapat dilakukan dengan cepat setelah operasi penulisan pada node Primary selesai dilakukan. Dengan demikian, MongoDB Replica Set mampu mempertahankan ketersediaan data pada node Secondary tanpa menimbulkan penundaan

replikasi yang berarti selama pengujian berlangsung.

G. Analisis dan Evaluasi Hasil

Berdasarkan hasil pengujian yang telah dilakukan, kedua konfigurasi konsistensi menunjukkan performa yang relatif serupa pada lingkungan MongoDB Replica Set yang digunakan. Hasil pengukuran write latency memperlihatkan bahwa Eventual Consistency sedikit lebih cepat pada volume data 10.000 dokumen, sedangkan Strong Consistency menghasilkan waktu yang sedikit lebih rendah pada volume data 50.000 dan 100.000 dokumen. Namun, selisih yang diperoleh pada seluruh skenario pengujian relatif kecil sehingga tidak menunjukkan perbedaan performa yang signifikan.

Hasil tersebut mengindikasikan bahwa penerapan WriteConcern("majority") pada Strong Consistency tidak memberikan tambahan overhead yang besar dibandingkan WriteConcern(w=1) pada Eventual Consistency dalam lingkungan pengujian yang digunakan. Dengan demikian, kedua konfigurasi dapat menjalankan proses penulisan data dengan performa yang hampir setara meskipun memiliki tingkat jaminan konsistensi yang berbeda.

Dari sisi replikasi, nilai replication delay yang diperoleh berada pada kisaran 0,021 hingga 0,028 detik dengan rata-rata sebesar 0,023796 detik. Nilai tersebut menunjukkan bahwa proses sinkronisasi data antara node Primary dan Secondary berlangsung dengan cepat dan stabil. Hasil ini membuktikan bahwa mekanisme replikasi MongoDB Replica Set mampu menjaga ketersediaan data pada node Secondary tanpa menimbulkan keterlambatan yang berarti.

Temuan penelitian juga menunjukkan bahwa karakteristik lingkungan pengujian memiliki pengaruh terhadap hasil yang diperoleh. Seluruh node MongoDB dijalankan pada satu mesin fisik yang sama melalui Docker sehingga latensi komunikasi antar node relatif rendah. Kondisi tersebut menyebabkan perbedaan performa antara Strong Consistency dan Eventual Consistency menjadi kurang terlihat secara signifikan karena proses komunikasi dan replikasi berlangsung dalam waktu yang sangat singkat.

Secara keseluruhan, hasil penelitian menunjukkan bahwa Strong Consistency dan Eventual Consistency memberikan performa yang hampir setara dalam pengujian write latency, sementara MongoDB Replica Set mampu mempertahankan replication delay yang rendah dan stabil. Oleh karena itu, pemilihan tingkat konsistensi pada MongoDB Replica Set lebih dipengaruhi oleh kebutuhan jaminan konsistensi data yang diinginkan dibandingkan pertimbangan performa pada lingkungan pengujian yang digunakan.

IV. KESIMPULAN

Penelitian ini berhasil menganalisis pengaruh Strong Consistency dan Eventual Consistency terhadap write latency dan replication delay pada MongoDB Replica Set yang terdiri atas satu node Primary dan dua node Secondary. Pengujian dilakukan menggunakan volume data sebesar 10.000, 50.000, dan 100.000 dokumen dengan konfigurasi WriteConcern

("majority") untuk Strong Consistency dan WriteConcern(w=1) untuk Eventual Consistency.

Hasil pengujian write latency menunjukkan bahwa kedua konfigurasi menghasilkan performa yang relatif setara. Eventual Consistency memperoleh write latency yang sedikit lebih rendah pada volume 10.000 dokumen, sedangkan Strong Consistency menghasilkan nilai yang sedikit lebih rendah pada volume 50.000 dan 100.000 dokumen. Namun, selisih yang diperoleh pada seluruh skenario pengujian relatif kecil sehingga tidak menunjukkan perbedaan performa yang signifikan.

Hasil pengukuran replication delay menunjukkan rata-rata sebesar 0,023796 detik dengan nilai yang stabil pada setiap pengujian. Temuan ini menunjukkan bahwa mekanisme replikasi MongoDB Replica Set mampu menjaga sinkronisasi data antar node dengan

cepat dan efisien selama proses pengujian berlangsung.

Berdasarkan hasil penelitian, dapat disimpulkan bahwa Strong Consistency dan Eventual Consistency memberikan performa yang hampir setara pada lingkungan MongoDB Replica Set yang dijalankan dalam satu mesin fisik menggunakan Docker. Oleh karena itu, pemilihan tingkat konsistensi lebih dipengaruhi oleh kebutuhan jaminan konsistensi data yang diinginkan dibandingkan pertimbangan performa sistem pada lingkungan pengujian yang digunakan.

Untuk penelitian selanjutnya, pengujian dapat dilakukan menggunakan beberapa mesin fisik yang terpisah, jumlah node yang lebih banyak, serta volume data yang lebih besar sehingga pengaruh tingkat konsistensi terhadap performa sistem terdistribusi dapat diamati secara lebih komprehensif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh anggota tim yang telah bekerja sama dalam perancangan arsitektur MongoDB Replica Set, pelaksanaan pengujian, serta analisis data pada penelitian ini. Ucapan terima kasih juga disampaikan kepada Universitas Duta Bangsa Surakarta, khususnya Fakultas Ilmu Komputer, atas dukungan fasilitas dan kesempatan yang diberikan selama penelitian berlangsung.

REFERENSI

[1] W. Uriawan, R. A. Fauzan, R. Z. Faroj, P. Pitriani, and R. Firmansyah, "Implementing Replica Set: Strategy to Improve the Performance of NoSQL Database Cluster in MongoDB," 2024, doi: 10.20944/preprints202407.0449.v1.

[2] S. Ferreira, J. Mendonça, B. Nogueira, W. Tiengo, and E. Andrade, "Impacts of data consistency levels in cloud-based NoSQL for data-intensive applications," *J. Cloud Comput.*, vol. 13, no. 1, 2024, doi: 10.1186/s13677-024-00716-7.

[3] A. A. E. Alflahi, M. A. Y. Mohammed, and A. Alsammani, "Enhancement of Database Access Performance by Improving Data Consistency in a Non-relational Database System (NoSQL)," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 14816 LNCS, pp. 194–205, 2024, doi: 10.1007/978-3-031-65223-3_13.

[4] S. Ferreira, J. Mendonca, B. Nogueira, W. Tiengo, and E. Andrade, "Benchmarking Consistency Levels of Cloud-Distributed NoSQL Databases Using YCSB," *IEEE Access*, vol. 13, no. March, pp. 63428–63438, 2025, doi: 10.1109/ACCESS.2025.3558923.

[5] E. Dritsas and M. Trigka, "Database Systems in the Big Data Era: Architectures, Performance, and Open Challenges," *IEEE Access*, vol. 13, no. June, pp. 95068–95084, 2025, doi: 10.1109/ACCESS.2025.3572059.

[6] T. Acquah, R. Amankwah, and B. Appiah, "Empirical Insights into Replication Models for Distributed Database Environments," *Int. J. Comput. Appl.*, vol. 186, no. 53, pp. 27–34, 2024, doi: 10.5120/ijca2024924212.

[7] P. Bhosale, "Data Consistency Models in Distributed Systems: CAP Theorem Revisited," *Int. J. Sci. Technol. IJSAT23031408*, vol. 14, no. 3, pp. 1–11, 2023.

[8] H. A. Mahmoud and H. Maseeh Yasin, "Data Integrity and Consistency Challenges in Distributed Database Systems," *Eng. Technol. J.*, vol. 10, no. 05, pp. 5077–5086, 2025, doi: 10.47191/etj/v10i05.36.

[9] D. K. Siripurapu, "Real-Time Data Synchronization in Distributed Systems: a Comprehensive Analysis of Architectures, Strategies, and Implementation Technologies," *Int. J. Res. Comput. Appl. Inf. Technol.*, vol. 8, no. 1, pp. 3412–3424, 2025, [Online]. Available: https://iaeme.com/MasterAdmin/Journal_uploads/IJRCAIT/VOLUME_8_ISSUE_1/IJRCAIT_08_01_245.pdf

[10] A. Myrhorodskyi, "Comparison of data consistency models in distributed database management systems," vol. 22, pp. 101–112, 2025, doi: 10.31649/vitce/3.2025.101.