Jurnal Teknik Informatika (JUTIF)
Vol. 6, No. 1, February 2025, pp. 411-418

DOI: https://doi.org/10.52436/1.jutif.2025.6.1.4371
p-ISSN: 2723-3863
e-ISSN: 2723-3871

A COMPARATIVE STUDY OF MULTI-MASTER REPLICATION OF NOSQL DATABASE SERVER WITH VARYING DATA FORMATS

Dwi Kurnia Wibowo[*1], Agus Darmawan[2], Devi Astri Nawangnugraeni[3]

[1,2,3]Informatics, Engineering Faculty, Universitas Jenderal Soedirman, Indonesia
Email: [1]dwi.kurnia@unsoed.ac.id, [2]agus.darmawan@unsoed.ac.id, [3]devi.nawangnugraeni@unsoed.ac.id

(Article received: January 27, 2025; Revision: February 10, 2025; published: February 19, 2025)

Abstract

NoSQL Databases are currently an effective solution for managing large data sets distributed across many Servers. NoSQL Database design is usually based on its usability. Specifically related to the system or application to be built. This research aims to measure the Transfer Rate, CPU usage, Memory usage, query execution time for Create, Insert, Delete and remote replication query bandwidth in the Multi-Master Server replication process using two document stored NoSQL Database applications namely CouchBase and CouchDB by entering three different data models namely JSON, XML and CSV. The experimental results show that the Transfer Rate with CSV data format on CouchBase has the lowest value with an average of 111.41 kbps. CPU usage with XML data format on CouchBase has the lowest value with an average of 13.89%. Memory usage with JSON data format on CouchBase has the lowest value with an average of 1.68%. Query Execution Time Create with XML data format on CouchBase has the lowest value with an average of 1.16 seconds. Query Execution Time Insert on CouchBase with CSV data format has the lowest value with an average of 33.28 seconds. Bandwidth Query Execution Time Insert with CSV data format on CouchBase has the lowest value with an average of 24.78 mb. Query Execution Time Delete with JSON, XML and CSV data formats on CouchDB has the lowest value with an average of 1.5 seconds. Further research recommendations are to test Multi-Master Server Replication using other data formats and parameters or test the performance of data migration to other Databases with different data formats.

Keywords: *Database Server, Document Stored, Data Format, Multi-Master Server*, *NoSQL*

STUDI KOMPARASI REPLIKASI MULTI-MASTER PADA SERVER BASIS DATA NOSQL DENGAN BERBAGAI FORMAT DATA

Abstrak

NoSQL Database saat ini merupakan solusi yang efektif untuk mengelola set data besar yang didistribusikan di banyak *Server*. Desain basis data *NoSQL* biasanya didasarkan pada kegunaannya. Secara spesifik berhubungan dengan system atau aplikasi yang akan dibangun. Penelitian ini bertujuan untuk melakukan pengukuran ukuran *Transfer Rate*, penggunaan *CPU*, penggunaan *Memory, query execution time* untuk *Create, Insert, Delete* dan *bandwidth query remote* replikasi pada proses replikasi *Multi-Master Server* menggunakan dua aplikasi *NoSQL Database* berbasis *document stored* yaitu *CouchBase* dan *CouchDB* dengan memasukkan tiga model data yang berbeda yaitu *JSON, XML* dan *CSV*. Hasil percobaan menunjukkan bahwa *Transfer Rate* dengan format data *CSV* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 111.41 *kbps*. Penggunaan *CPU* dengan format data *XML* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 13.89%. Penggunaan *Memory* dengan format data *JSON* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 1.68%. *Query Exceution Time Create dengan format data XML* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 1.16 detik. *Query Execution Time Insert* pada *CouchBase* dengan format data *CSV* memiliki nilai yang paling rendah dengan rata-rata 33.28 detik. *Bandwidth Query Execution Time Insert* dengan format data *CSV* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 24.78 *mb*. *Query Execution Time Delete* dengan format data *JSON, XML* dan *CSV* pada *CouchDB* memiliki nilai yang paling rendah dengan rata-rata 1.5 detik. Rekomendasi penelitian selanjutnya melakukan Analisis Replikasi *Multi-Master Server* menggunakan format data dan parameter lain atau menguji performa migrasi data ke *Database* lain dengan format data yang berbeda.

Kata kunci: *Database Server, Document Stored, Data Format, Multi-Master Server*, *NoSQL*

1. PENDAHULUAN

NoSQL Database saat ini merupakan solusi yang efektif untuk mengelola set data besar yang didistribusikan di banyak *Server*. Desain basis data untuk sistem *NoSQL* biasanya didasarkan pada kegunaannya. Secara spesifik berhubungan dengan *system* atau aplikasi yang akan dibangun. Ada empat kategori *NoSQL Database*, berbasis dokumen, berbasis kolom, berbasis *key-value* dan berbasis grafik. *MongoDB*, *CouchBase*, *Cassandra, HBase* dan *Redis* adalah *NoSQL Database* yang paling unggul yang diuji menggunakan metode *YCSB* [1], [2], [3], [4].

Database berbasis *document stored* popular karena tiga alasan utama. Pertama, *NoSQL Database* berbasis *document stored* memudahkan penggunaan antara pemrograman berorientasi objek dan model *Database*. Kedua, karena format dokumen pada *NoSQL* berbasis *document stored* dapat berdiri sendiri dan dijalankan secara independen dari program pembuat *NoSQL Database*. Ketiga, *Database* berbasis *Document stored* selaras dengan paradigma pemrograman berbasis *web* yang saat ini dominan digunakan, khususnya model pemrograman *AJAX* [1], [5], [6].

Percobaan yang dilakukan sebelumnya pada pemrosesan data *NoSQL* di *web* sebelum diekspos sebagai *HTML* untuk konsumsi manusia, dan sebagai *XML* melalui antarmuka layanan *web*. Sebagai format perantara untuk memfasilitasi perhitungan statistik, *CSV* dihasilkan secara internal. Untuk menghubungkan data *OpenAIRE* dengan data terkait *web* dikeluarkan sebagai *Linked Open Data* (*LOD*). Pengukuran kinerja pembuatan *LOD* dengan pekerjaan *MapReduce* di atas *HBase*, dengan memetakan file *CSV* sebagai perantara dan dengan memetakan output *XML*[7], [8], [9], [10], [11].

Beberapa percobaan pengukuran kinerja replikasi *NoSQL Database* yang pernah dilakukan sebelumnya diantaranya membandingkan data dengan format *CSV* dengan *XML* untuk optimasi perhitungan statistik data di *web*. Penelitian yang dilakukan untuk mengukur waktu respon *query* dari aplikasi *NoSQL Database* berbasis *Document stored* dengan *MongoDB*. Penelitian yang dilakukan mengukur proses replikasi untuk *master-slave Server* dengan menggunakan format data yang berbeda yaitu *JSON* dan *XML*. Berdasarkan ketiga penelitian tersebut belum dilakukan dalam replikasi *Multi-Master Server* dan menggunakan format data yang sama [1], [12], [13], [14].

Replikasi adalah mekanisme membuat banyak salinan dari objek (*file*, sistem *file*, *Database*, dan sebagainya) dengan tujuan untuk menyediakan high-availability, integritas tinggi, kinerja tinggi, atau kombinasi dari itu. Salinan dapat disimpan di beberapa *Server*. Keberadaan salinan dapat meningkatkan ketersediaan data dan salinan data dapat membantu mengurangi komunikasi dan biaya permintaan. Terdapat dua mekanisme replikasi yaitu replikasi *multi-master server* dan *master-slave server*[8], [15], [16], [17], [18], [19].

Replikasi *multi-master* juga dikenal sebagai replikasi lanjutan atau replikasi simetris. Replikasi *multi-master* adalah metode replikasi basis data yang memungkinkan data disimpan oleh sekelompok komputer dan diperbarui oleh anggota grup mana pun. Semua anggota responsif terhadap permintaan data klien. Semua anggota grup ditetapkan sebagai *master* pada setiap *Server*[8], [20], [21], [22].

Replikasi *master-slave* juga dikenal sebagai replikasi asimetris. Merupakan jenis replikasi yang mereplikasi data pada *slave Server*. Satu node digunakan sebagai *master* atau primer. *Master Server* merupakan sumber otoritas untuk data dan biasanya bertanggung jawab untuk memproses setiap pembaruan data. *Server* lainnya adalah *slave*, atau sekunder. Proses replikasi dilakukan dengan menyinkronkan *slave server* pada *master server*[8], [17], [19].

Penggunaan format data *JSON, XML,* dan *CSV* untuk *NoSQL Database* perlu dilakukan spesifikasi performa sehingga dapat menjadikan rekomendasi bagi pengguna *NoSQL Database* dalam memilih data yang akan ditransaksikan. Pada penelitian ini dilakukan replikasi *Multi-Master Server* pada dua *NoSQL Database* berbasis *document stored* yaitu *CouchBase* dan *CouchDB*. Menetapkan parameter ukuran *Transfer Rate* dikarenakan pada penelitian ini dilakukan pada proses replikasi yang erat hubungannya dengan konektivitas antar *Server*[5], [23], [24], [25], [26], [27]. Ukuran *transfer* saat replikasi berpengaruh terhadap seberapa besar data yang dibawa yang mana menjadi alasan pengukuran penggunaan *CPU* dan *Memory* dilakukan. Pengukuran *query execution time* juga menjadi parameter kinerja replikasi aplikasi *NoSQL Database* karena *query execution time* merupakan waktu yang digunakan pada sebuah *remote query client* dalam melakukan proses replikasi pada masing-masing format data yang digunakan. Proses replikasi dilakukan pada satu jaringan yang bersifat lokal *(LAN)*.

2. METODOLOGI PENELITIAN

Metodologi penelitian pada makalah ini berisikan tentang tahapan-tahapan penelitian yang akan dilakukan untuk pengukuran *Transfer Rate*, penggunaan *CPU*, penggunaan *Memory*, penggunaan *bandwidth* pada *query remote* replikasi dan *query execution time* pada replikasi *Multi-Master Server NoSQL Database* berbasis *Document stored* dalam proses replikasi dengan format data *JSON*, *XML* dan *CSV*.

Gambar 1. Bagan Metodologi Penelitian

Gambar 1 menjelaskan tahapan-tahapan dalam melakukan penelitian pengukuran *Transfer Rate*, penggunaan *CPU*, penggunaan *Memory*, penggunaan *bandwidth query remote* replikasi dan *query execution time* pada replikasi *NoSQL Database* berbasis *Document stored* dalam proses replikasi dengan format data *JSON*, *XML* dan *CSV* yaitu :

1. Studi Pustaka
   Studi Pustaka dilakukan dengan memelajari hasil penelitian sebelumnya.
2. Menentukan *State of The Art*
   Menentukan gap penelitian berdasarkan hasil penelitian sebelumnya.
3. Simulasi & Pengujian Replikasi
   Simulasi dilakukan dengan identifikasi kebutuhan untuk persiapan sistem yang meliputi kebutuhan *hardware, software* dan koneksi jaringan. Pengujian replikasi dilakukan pada jaringan yang sama dan bersifat lokal. Pegujian yang dilakukan dengan membangun program untuk instruksi replikasi antar *server*.
4. Pengumpulan Data Hasil Pengujian
   Pengumpulan Data Hasil Pengujian dilakukan dengan kode sumber pencatatan yang ditanamkan pada program instruksi replikasi antar *server*. Data yang dicatat adalah performa *create, insert, delete, transfer rates, CPU usages,* dan *memory usages.*
5. Melakukan Analisis Data dan Menarik Kesimpulan
   Berdasarkan hasil pengumpulan data dilakukan analisis data dengan pendekatan uji statistik untuk mengetahui rata-rata performa setiap format data untuk proses *create, insert, delete, transfer rates, CPU usages,* dan *memory usages.*

3. HASIL DAN PEMBAHASAN

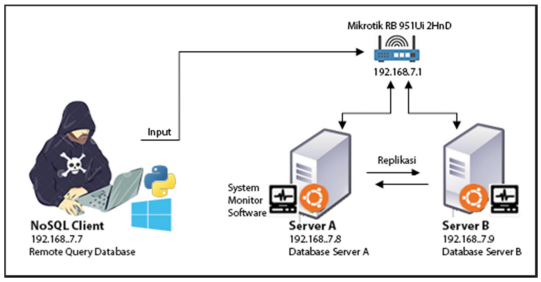
Studi Pustaka dilakukan dengan pencarian dan pembelajaran hasil penelitian sebelumnya dengan kata kunci pencarian di *google sholar*: *replication, NoSQL,* dan *document stored.* Hasil pencarian dengan berdasarkan kata kunci yang ditentukan sejumlah 143 jurnal internasional dari rentan tahun 2020-2024. Selanjutnya dilakukan proses *sorting* untuk kebutuhan referensi arah penelitian yang berkaitan dengan format data di *NoSQL Database* sehingga mendapatkan hasil sejumlah 77 jurnal internasional. Proses *sorting* terakhir mengerucut pada basis *file* yaitu *document stored* yang menghasilkan 27 jurnal internasional. 27 jurnal internasional berdasarkan 3 tahap *sorting* dijadikan referensi pada penelitian ini dan digunakan untuk menentukan *State of The Art.*

Berdasarkan 27 jurnal internasional hasil *sorting* dilakukan *mapping* dan menghasilkan temuan *gap* penelitian. Penelitian sebelumnya belum ada yang membahas performa replikasi antar *NoSQL Database Server* yang bersifat *Master to Master.* Penelitian sebelumnya belum ada yang membahas perbandingan format data *JSON, XML,* dan *CSV* padar proses replikasi data pada *NoSQL Database*.

Setelah ditentukan gap penelitian tahap selanjutnya adalah melakukan simulasi dan pengujian. Simulasi dilakukan dengan persiapan kebutuhan *hardware, software,* dan koneksi jaringan. Kebutuhan *hardware* meliputi dua unit *laptop* yang digunakan sebagai *Master Server* A, *Master Server B* dan satu *laptop* sebagai *client* untuk pengukuran replikasi *NoSQL Database* berbasis *Document stored* dalam penelitian ini. Kemudian satu buah *Router* sebagai *AP* untuk kebutuhan jaringan. Konfigurasi *hardware* terdiri dari dua *Server* sebagai *Master Server A* dan *Master Server B* dengan spesifikasi yang berbeda.

Kebutuhan *software* ini dilakukan meliputi persiapan sistem operasi, *NoSQL Database*, dan *tools* untuk mengukur *Transfer Rate*, penggunaan *CPU*, penggunaan *Memory*, dan *execution time* pada aplikasi *NoSQL Database* bebabasis *Document stored*. Kedua *server* dipasang sistem operasi *Linux Ubuntu* yang dipasang *System Monitoring Software* bernama *HTOP* dan aplikasi *System Monitoring Transfer Rate Data* bernama *NLOAD.*

Data yang dianalisis dalam penelitian ini merupakan data *query* untuk proses *Insert* secara berulang (*looping*). Data yang dimasukkan berupa data buku. Data dimuat dengan kombinasi antara huruf dan angka dan data disesuaikan dalam format penulisan *JSON*, *XML* dan *CSV*.



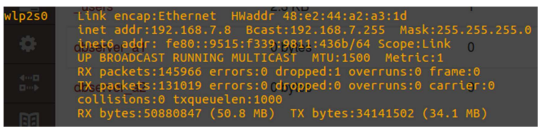
Gambar 2. Arsitektur Sistem Replikasi *NoSQL Database*.

Sistem operasi *Linux Ubuntu* dipasang pada kedua *Server,* kemudian dipasang aplikasi *NoSQL Database* berbasis *Document stored*. Aplikasi untuk *system monitoring* bernama *HTOP* yang digunakan untuk mengukur *Transfer Rate*, penggunaan *CPU*,

penggunaan *Memory*, *query execution time* replikasi dan *bandwidth query remote* untuk *Create, Insert, Delete*. Aplikasi *system monitoring Transfer Rate data* bernama *NLOAD* digunakan untuk mengukur *Transfer Rate* replikasi antar *Server*.

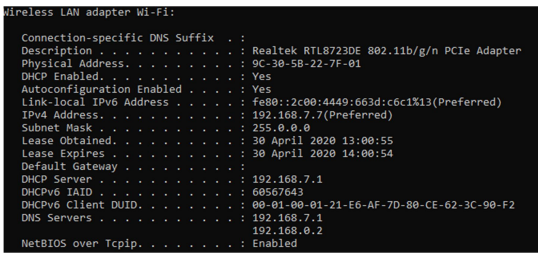
Pada tahapan konfigurasi replikasi setiap aplikasi *NoSQL Database* adalah kedua laptop diatur sebagai *master Server* dan satu laptop lainnya diatur sebagai *client*. Tahapan konfigurasi dilakukan pada setiap aplikasi *NoSQL Database* agar pada saat dilakukan proses *Insert* data, aplikasi *NoSQL Database* pada kedua *master Server* secara langsung akan melakukan proses replikasi.

Pengaturan *setting IP address* untuk infrastruktur jaringan dalam penelitian ini melibatkan keterhubungan antara satu komputer *client* dan dua komputer *Server* yang akan diuji dalam memproses aplikasi *NoSQL Database* berbasis *Document stored*. Pengukuran dilakukan dengan menggunakan pengaturan jaringan dalam satu network yang sama dengan menggunakan media nirkabel. Untuk media nirkabel menggunakan *AP point RouterBoard Mikrotik 951Ui 2HnD*. *Range IP address* yang digunakan dalam jaringan ditampilkan pada Gambar 2. *Setting IP address* pada kedua *Server* dan satu *client* ditampilkan pada Gambar 3, Gambar 4 dan Gambar 5.



Gambar 3. *IP address Server A.*

Gambar 4. *IP address Server B.*

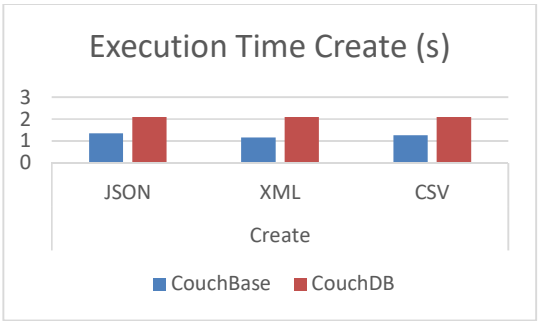


Gambar 5. *IP Address Client*.

Tahapan *Testing System* pada penelitian ini meliputi proses yang akan digunakan untuk pengukuran *Transfer* Rate, penggunaan *CPU*, penggunaan *Memory*, execution time untuk *Create*, *Insert*, *Delete* dan bandwidth query remote replikasi pada replikasi *NoSQL Database* berbasis Document stored yaitu memasukkan data dengan jumlah 1.000, 2.000, 3.000, 4.000, 5.000 secara berurutan dan masing-masing input data dapat dilakukan replikasi.

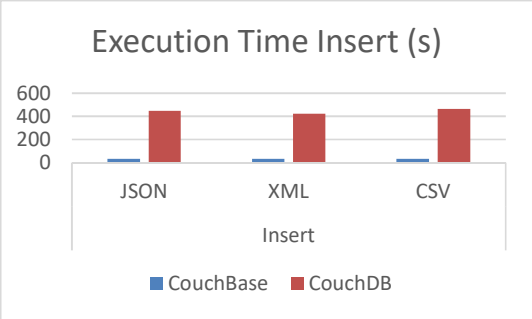
Python driver digunakan sebagai metode akses untuk memasukkan data dari *NoSQL Client* ke aplikasi *NoSQL Database* yang berada di Master *Server* dihubungkan melalui melalui jaringan dengan alamat jaringan yang sama. Kemudian *script Python driver* menjalankan program untuk melakukan import pada library sesuai aplikasi *NoSQL Database* yang akan dihubungkan, melakukan koneksi dengan *Database Server*, melakukan koneksi dengan *Database*, melakukan pengulangan dengan jumlah data yang telah ditentukan, dan memasukkan data yang dituliskan di dalam *scrip*t tersebut.

Hasil pengukuran diketahui bahwa data rata-rata yang didapatkan menunjukkan penggunaan format data pada *Database* dan penggunaan aplikasi *NoSQL Database* yang berbeda dapat mempengaruhi ukuran penggunaan sumber daya *client* pada *remote query execution time Create*. Hal tersebut dibuktikan dengan hasil *query execution time Create* pada *client* yang lebih cepat pada program *remote query Create* aplikasi *NoSQL Database CouchBase* saat melakukan *Create* dengan format data *XML*. Hasil pengukuran ditampilkan dalam grafik pada Gambar 6.



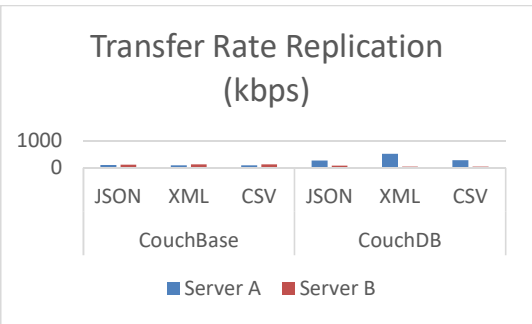
Gambar 6. *Execution Time Create*

Hasil pengukuran diketahui bahwa data rata-rata yang didapatkan menunjukkan penggunaan format data pada *Database* dan penggunaan aplikasi *NoSQL Database* yang berbeda dapat mempengaruhi ukuran penggunaan sumber daya *client* pada *remote query execution time Insert*. Hal tersebut dibuktikan dengan hasil *query execution time Insert* pada *client* yang lebih cepat pada program *remote query Insert* aplikasi *NoSQL Database CouchBase* saat melakukan *Insert* dengan format data *XML*. Hasil pengukuran ditampilkan dalam grafik pada Gambar 7.



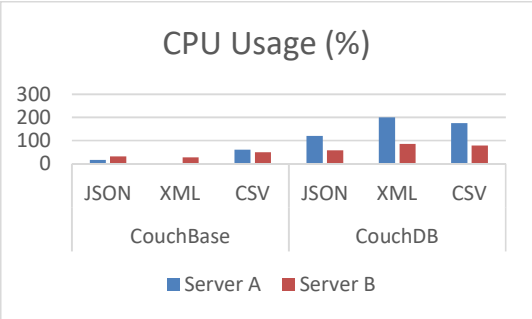
Gambar 7. *Execution Time Insert*

Hasil pengukuran diketahui bahwa data rata-rata yang didapatkan menunjukkan penggunaan format data pada *Database* dan penggunaan aplikasi *NoSQL Database* yang berbeda dapat mempengaruhi daya *Transfer Rate*. Hal tersebut dibuktikan dengan hasil *Bandwidth* pada kedua *master Server* yang lebih rendah pada format *JSON, XML* dan *CSV* saat melakukan *Insert* pada *NoSQL Database CouchBase* dengan format data *CSV*. Hasil pengukuran ditampilkan dalam grafik pada Gambar 8.



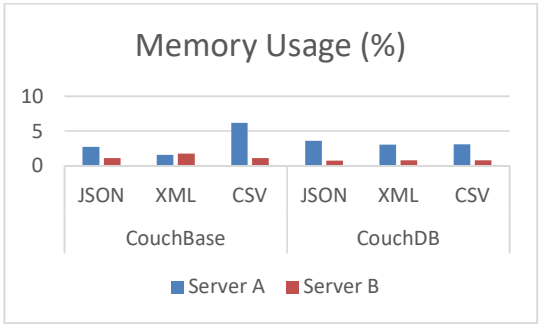
Gambar 8. *Transfer Rate* Replikasi

Hasil pengukuran diketahui bahwa data rata-rata yang didapatkan menunjukkan penggunaan format data pada *Database* dan penggunaan aplikasi *NoSQL Database* yang berbeda dapat mempengaruhi ukuran penggunaan sumber daya *Server* pada *CPU*. Hal tersebut dibuktikan dengan hasil penggunaan *CPU* pada kedua *master Server* yang lebih rendah pada aplikasi *NoSQL Database CouchBase Insert* dengan format data *XML*. Hasil pengukuran ditampilkan dalam grafik pada Gambar 9.



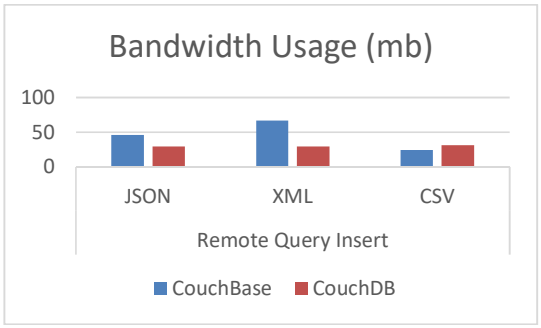
Gambar 9. Penggunaan *CPU*

Hasil pengukuran diketahui bahwa data rata-rata yang didapatkan menunjukkan penggunaan format data pada *Database* dan penggunaan aplikasi *NoSQL Database* yang berbeda berdampak pada penggunaan *Memory*. Hal tersebut dibuktikan dengan hasil penggunaan *Memory* pada kedua *master Server* yang lebih rendah pada aplikasi *NoSQL Database CouchBase* saat melakukan *Insert* dengan format data *JSON*. Hasil pengukuran ditampilkan dalam grafik pada Gambar 10.



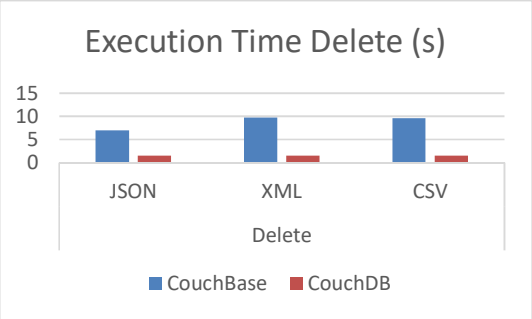
Gambar 10. Penggunaan *Memory*

Hasil pengukuran diketahui bahwa data rata-rata yang didapatkan menunjukkan penggunaan format data pada *Database* dan penggunaan aplikasi *NoSQL Database* di sisi *client* pada *remote query Insert Database*. Hal tersebut dibuktikan dengan hasil *bandwidth remote query Insert Database* pada *client* yang lebih cepat pada program aplikasi *NoSQL Database CouchBase* saat melakukan *Insert* dengan format data *CSV*. Hasil pengukuran ditampilkan dalam grafik pada Gambar 11.



Gambar 11. *Bandwidth Usage Query Insert*

Hasil pengukuran diketahui bahwa data rata-rata yang didapatkan menunjukkan penggunaan format data pada *Database* dan penggunaan aplikasi *NoSQL Database* yang berbeda dapat mempengaruhi ukuran penggunaan sumber daya *client* pada *remote query execution time Delete*. Hal tersebut dibuktikan dengan hasil *query execution time Delete* pada *client* yang lebih cepat pada program *remote query Delete* aplikasi *NoSQL Database CouchDB* saat melakukan *Delete* dengan format data *JSON, XML* dan *CSV*. Hasil pengukuran ditampilkan dalam grafik pada Gambar 12.



Gambar 12. *Execution Time Delete*

Rate yang paling baik waktu *Database CouchBase* dengan rata-rata 111.41 *kbps*, paling rendah dibandingkan dengan format data *XML* dan *JSON*. Memiliki rata-rata paling rendah juga dari aplikasi *CouchDB* dengan format data *JSON, XML* dan *CSV*. Format data *XML* memiliki penggunaan *CPU* yang paling baik saat isi *Database* CouhBase dengan rata-rata 13.89%, lebih rendah dibandingkan format data *JSON* dan *CSV*. Memiliki rata-rata paling rendah juga dari aplikasi *CouchDB* dengan format data *JSON, XML* dan *CSV*.

Format data *JSON* memiliki penggunaan *Memory* yang paling baik saat isi *Database CouchBase* dengan rata-rata 1.68%, lebih rendah dibandingkan dengan format data *XML* dan *CSV*. Memiliki rata-rata paling rendah juga dari aplikasi *CouchDB* dengan format data *JSON, XML* dan *CSV*. Format data *CSV* memiliki query *execution time Insert* yang paling baik saat isi *Database CouchBase* menggunakan dengan rata-rata 33.28 detik, lebih cepat dibandingkan dengan format data *JSON* dan *CSV*. Memiliki rata-rata paling rendah juga dari aplikasi *CouchDB* dengan format data *JSON, XML* dan *CSV*.

Format data *CSV* memiliki *bandwidth usage* query *execution time Insert* yang paling baik saat isi *Database CouchBase* dengan rata-rata 24.78 *mbps*, lebih rendah dibandingkan dengan format data *JSON* dan *CSV*. Memiliki rata-rata paling rendah juga dari aplikasi *CouchDB* dengan format data *JSON, XML* dan *CSV*. Format data *JSON, XML* dan *CSV CouchDB* memiliki query *execution time Delete* yang paling baik saat isi *Database CouchDB* dengan rata-rata 1.5 detik, lebih cepat dari aplikasi *CouchBase* dengan format data *JSON, XML* dan *CSV*.

4. KESIMPULAN

Hasil percobaan menunjukkan bahwa *Transfer Rate* dengan format data *CSV* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 111.41 *kbps*. Penggunaan *CPU* dengan format data *XML* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 13.89%. Penggunaan *Memory* dengan format data *JSON* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 1.68%. *Query Exceution Time Create dengan format data XML* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 1.16 detik. *Query Execution Time Insert* pada *CouchBase* dengan format data *CSV* memiliki nilai yang paling rendah dengan rata-rata 33.28 detik. *Bandwidth Query Execution Time Insert* dengan format data *CSV* pada *CouchBase* memiliki nilai yang paling rendah dengan rata-rata 24.78 *mb*. *Query Execution Time Delete* dengan format data *JSON, XML* dan *CSV* pada *CouchDB* memiliki nilai yang paling rendah dengan rata-rata 1.5 detik.

Rekomendasi penelitian selanjutnya bisa dilakukan Analisis Replikasi *Multi-Master Server* menggunakan format data lainnya dan *Database Server* lainnya. Penelitian selanjutnya bisa dilakukan Analisis Replikasi *Multi-Master Server* dengan parameter lainnya. Penelitian selanjutnya bisa dilakukan pengembangan yaitu menguji performa migrasi data antar *NoSQL Database* yang berbeda dengan format data yang berbeda.

DAFTAR PUSTAKA

[1] L. Bao, J. Yang, C. Q. Wu, H. Qi, X. Zhang, and S. Cai, "*XML*2HBase: Storing and querying large collections of *XML* documents using a *NoSQL Database* system," *J Parallel Distrib Comput*, vol. 161, pp. 83–99, Mar. 2022, doi: 10.1016/j.jpdc.2021.11.003.

[2] S. Kim, Y. Hoang, T. T. Yu, and Y. S. Kanwar, "GeoYCSB: A Benchmark Framework for the Performance and Scalability Evaluation of Geospatial *NoSQL Database*s," *Big Data Research*, vol. 31, p. 100368, Feb. 2023, doi: 10.1016/j.bdr.2023.100368.

[3] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Ş. Győrödi, G. A. Gabor, and G. D. Pecherle, "Performance Analysis of *NoSQL* and Relational *Database*s with *CouchDB* and MySQL for Application's Data Storage," *Applied Sciences*, vol. 10, no. 23, p. 8524, Nov. 2020, doi: 10.3390/app10238524.

[4] K. Mavrogiorgos, A. Kiourtis, A. Mavrogiorgou, and D. Kyriazis, "A Comparative Study of MongoDB, ArangoDB and *CouchDB* for Big Data Storage," in *2021 5th International Conference on Cloud and Big Data Computing (ICCBDC)*, New York, NY, USA: ACM, Aug. 2021, pp. 8–14. doi: 10.1145/3481646.3481648.

[5] I. Carvalho, F. Sá, and J. Bernardino, "Performance Evaluation of *NoSQL* Document *Database*s: *CouchBase*, *CouchDB*, and MongoDB," *Algorithms*, vol. 16, no. 2, p. 78, Feb. 2023, doi: 10.3390/a16020078.

[6] - Rianto, M. A. Rifansyah, R. Gunawan, I. Darmawan, and A. Rahmatulloh, "Comparison of *JSON* and *XML* Data Formats in Document Stored *NoSQL Database* Replication Processes," *Int J Adv Sci Eng Inf Technol*, vol. 11, no. 3, pp. 1150–1156, Jun. 2021, doi: 10.18517/ijaseit.11.3.11570.

[7] Z. Brahmia, H. Hamrouni, and R. Bouaziz, "*XML* data manipulation in conventional and temporal *XML Database*s: A survey," *Comput Sci Rev*, vol. 36, p. 100231, May 2020, doi: 10.1016/j.cosrev.2020.100231.

[8] M. Aggarwal, S. B. Bajaj, and V. Jaglan, "Performance Analysis of Degree of Redundancy for Replication in Distributed *Database* System," in *2022 1st International Conference on Informatics (ICI)*, IEEE, Apr. 2022, pp. 176–180. doi: 10.1109/ICI53355.2022.9786886.

[9] F. Castro-Medina, L. Rodriguez-Mazahua, A. López-Chau, M. A. Abud-Figueroa, and G. Alor-Hernández, "FRAGMENT: A Web Application for *Database* Fragmentation, Allocation and Replication over a Cloud Environment," *IEEE Latin America Transactions*, vol. 18, no. 06, pp. 1126–1134, Jun. 2020, doi: 10.1109/TLA.2020.9099751.

[10] S. Lee *et al.*, "X-SSD: A Storage System with Native Support for *Database* Logging and Replication," in *Proceedings of the 2022 International Conference on Management of Data*, New York, NY, USA: ACM, Jun. 2022, pp. 988–1002. doi: 10.1145/3514221.3526188.

[11] P. Nagaraj, V. Muneeswaran, A. V. S. R. Pavan Naidu, N. Shanmukh, P. V. Kumar, and G. S. Satyanarayana, "Automated E-Commerce Price Comparison Website using PHP, XAMPP, MongoDB, Django, and Web Scrapping," in *2023 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, Jan. 2023, pp. 1–6. doi: 10.1109/ICCCI56745.2023.10128573.

[12] D. Yedilkhan, A. Mukasheva, D. Bissengaliyeva, and Y. Suynullayev, "Performance Analysis of Scaling *NoSQL* vs SQL: A Comparative Study of MongoDB, Cassandra, and PostgreSQL," in *2023 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, IEEE, May 2023, pp. 479–483. doi: 10.1109/SIST58284.2023.10223568.

[13] L. Chen, A. Davoudian, and M. Liu, "A workload-driven method for designing aggregate-oriented *NoSQL Database*s," *Data Knowl Eng*, vol. 142, p. 102089, Nov. 2022, doi: 10.1016/j.datak.2022.102089.

[14] C. J. F. Candel, D. Sevilla Ruiz, and J. J. García-Molina, "A unified metamodel for *NoSQL* and relational *Database*s," *Inf Syst*, vol. 104, p. 101898, Feb. 2022, doi: 10.1016/j.is.2021.101898.

[15] A. Pellegrini, "Replication of Computational Results Report for ' *Green Simulation with Database Monte Carlo* ,'" *ACM Transactions on Modeling and Computer Simulation*, vol. 31, no. 1, pp. 1–4, Jan. 2021, doi: 10.1145/3426823.

[16] Y. Lu, X. Yu, L. Cao, and S. Madden, "Epoch-based commit and replication in distributed OLTP *Database*s," *Proceedings of the VLDB Endowment*, vol. 14, no. 5, pp. 743–756, Jan. 2021, doi: 10.14778/3446095.3446098.

[17] R. Mucha, B. Balis, C. Grigoras, and J. Kitowski, "*Database* Replication for Disconnected Operations with Quasi Real-Time Synchronization," *Computer Science*, vol. 24, no. 3, Oct. 2023, doi: 10.7494/csci.2023.24.3.4831.

[18] Y.-H. Zeng, Z.-N. Yin, H. Luo, and F. Gao, "DeOri 10.0: An Updated *Database* of Experimentally Identified Eukaryotic Replication Origins," *Genomics Proteomics Bioinformatics*, vol. 22, no. 5, Dec. 2024, doi: 10.1093/gpbjnl/qzae076.

[19] M. Aggarwal, S. B. Bajaj, and V. Jaglan, "Performance Analysis of Degree of Redundancy for Replication in Distributed *Database* System," in *2022 1st International Conference on Informatics (ICI)*, IEEE, Apr. 2022, pp. 176–180. doi: 10.1109/ICI53355.2022.9786886.

[20] M.-J. Dong, H. Luo, and F. Gao, "DoriC 12.0: an updated *Database* of replication origins in both complete and draft prokaryotic genomes," *Nucleic Acids Res*, vol. 51, no. D1, pp. D117–D120, Jan. 2023, doi: 10.1093/nar/gkac964.

[21] A. Menon and D. J. Mallinson, "Policy Diffusion Speed: A Replication Study Using the State Policy Innovation and Diffusion *Database*," *Political Studies Review*, vol. 20, no. 4, pp. 702–716, Nov. 2022, doi: 10.1177/14789299211052828.

[22] A. E. A. Raouf, A. Abo-Alian, and N. L. Badr, "A Predictive Multi-Tenant *Database* Migration and Replication in the Cloud Environment," *IEEE Access*, vol. 9, pp. 152015–152031, 2021, doi: 10.1109/ACCESS.2021.3126582.

[23] M. Nuriev, R. Zaripova, O. Yanova, I. Koshkina, and A. Chupaev, "Enhancing MongoDB query performance through index optimization," *E3S Web of Conferences*, vol. 531, p. 03022, Jun. 2024, doi: 10.1051/e3sconf/202453103022.

[24] A. Somasundar, M. Chilakarao, B. R. Krishnam Raju, S. Kumari Behera, C. V. Ramana, and P. K. Sethy, "MongoDB integration with Python and Node.js, Express.js," in *2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, IEEE, Jan. 2024, pp. 1–5. doi: 10.1109/ICAECT60202.2024.10469546.

[25] T. Capris, P. Melo, N. M. Garcia, I. M. Pires, and E. Zdravevski, "Comparison of SQL and *NoSQL Database*s with different workloads: MongoDB vs MySQL evaluation," in *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, IEEE, Oct. 2022, pp. 214–218. doi: 10.1109/ICDABI56818.2022.10041513.

[26] P. Tripathi, M. H. Miraz, and S. Joshi, "Comparative Analysis of MongoDB and InfluxDB for Time Series Data Management in IoT Environments: A Study on Performance, Scalability, and Concurrency," in *2023 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA)*, IEEE, Dec. 2023, pp. 39–42. doi: 10.1109/CoNTESA61248.2023.10384962.

[27] A. C. F. Spengler and P. S. L. de Souza, "The impact of using *CouchDB* on Hyperledger Fabric performance for heterogeneous medical data storage," in *2021 XLVII Latin American Computing Conference (CLEI)*, IEEE, Oct. 2021, pp. 1–10. doi: 10.1109/CLEI53233.2021.9640180.