Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 7, No. 7, Juli 2023, hlm. 3085-3094 http://j-ptiik.ub.ac.id

Analisis Kinerja Mekanisme *Caching* MongoDB Cluster pada Moodle

Ilham Fathur Ilmi

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: ilmi.if@protonmail.com

Abstrak

Moodle merupakan salah satu platform pendidikan digital digunakan secara meluas. Penggunaan platform digital seperti Moodle bergantung pada beberapa faktor yang meliputi kemudahan pengguna, kenyamanan, norma subjektif, kepuasan, dan interaktifitas. Namun, efesiensi kinerja Moodle sangat dipengaruhi oleh banyaknya fitur yang disediakan, tingginya frekuensi penggunaan dan banyaknya pengguna. Untuk mengatasi inefisiensi kinerja Moodle, mekanisme *caching* dengan MongoDB Cluster diajukan dalam menangani beban trafik yang tinggi dari pengguna. Perbandingan pengujian kinerja Moodle pada aktivitas *enrol*, *view course*, dan *quiz* dengan skenario jumlah pengguna 20, 60, dan 140 *user* dilakukan untuk memperoleh gambaran peningkatan kinerja Moodle dengan penerapan mekanisme *caching* MongoDB Cluster. Hasil pengujian menunjukkan bahwa penerapan mekanisme *caching* MongoDB Cluster yang diajukan belum berhasil meningkatkan kinerja Moodle. Penurunan *throughput* terendah didapati pada aktivitas view course dengan skenario 60 *user* (-8,26%), sedangkan penurunan *throughput* tertinggi berada pada aktivitas *quiz* dengan skenario 20 *user* (-16,96%). Peningkatan *latency* terendah didapati pada aktivitas *enrol* dengan skenario 20 *user* (-1,70%), sedangkan peningkatan *latency* tertinggi berada pada aktivitas *quiz* dengan skenario 20 *user* (-27,54%). Salah satu penyebabnya adalah penggunaan *storage engine* WiredTiger pada MongoDB Cluster tanpa konfigurasi lebih lanjut tidak cukup memadai untuk bisa memberikan peningkatan kinerja untuk kebutuhan mekanisme *caching*.

Kata kunci: *e-learning, moodle, caching, optimisasi, basis data terdistribusi, mongodb cluster*

Abstract

Moodle is one of the most widely used digital education platforms. The use of digital platforms such as Moodle depends on several factors including user-friendliness, convenience, subjective norms, satisfaction, and interactivity. However, the performance efficiency of Moodle is strongly influenced by the number of features provided, the high frequency of use and the number of users. To overcome the performance inefficiency of Moodle, a caching mechanism with MongoDB Cluster is proposed to handle high traffic load from users. Comparison of Moodle performance testing on enrol, view course, and quiz activities with scenarios of 20, 60, and 140 users was conducted to obtain an overview of Moodle performance improvement with the application of MongoDB Cluster caching mechanism. The test results show that the application of the proposed MongoDB Cluster caching mechanism has not succeeded in improving Moodle performance. The lowest throughput decrease was found in the view course activity with a 60 user scenario (-8.26%), while the highest throughput decrease was in the quiz activity with a 20 user scenario (-16.96%). The lowest latency increase was found in the enrol activity with a scenario of 20 users (-1.70%), while the highest latency increase was in the quiz activity with a scenario of 20 users (-27.54%). One of the reasons is that using the WiredTiger storage engine on the MongoDB Cluster without further configuration is not sufficient enough to provide performance improvements for the needs of the caching mechanism.

Keywords: *e-learning, moodle, caching, optimization, distributed database, mongodb cluster*

1. PENDAHULUAN

Pendidikan yang merupakan salah satu sendi kehidupan publik, berkembang seiring dengan perkembangan teknologi informasi. Perkembangan terkini telah memungkinkan integrasi pendidikan dengan teknologi informasi sehingga dapat membuat penyelenggaraan pendidikan berlangsung lebih efektif dan efisien

dari masa-masa yang lampau dengan mengayakan kemungkinan penyelenggaraan pendidikan pada batas ruang dan waktu yang ada, meminimalisir upaya yang diperlukan dalam interaksi antar pihak dan sumber pembelajaran, serta menyediakan fasilitas pembelajaran yang disesuaikan dengan personalisasi kecenderungan pengguna (Xin, et al., 2021). Sebagai platform pendidikan digital, *Learning Management System* telah digunakan dalam jumlah yang cukup signifikan. Salah satu LMS yang banyak digunakan adalah Moodle. Berdasarkan statistik yang dirilis oleh Moodle pada tahun 2023, terdapat 154 situs Moodle terdaftar dengan 375 juta pengguna dari 239 negara (Moodle, 2023b). Selain itu penelitian yang dilakukan Roy, et al. (2008) di Open University UK menemukan bahwa penggunaan platform pendidikan berbasis daring seperti LMS mengurangi 87% penggunaan listrik dan 85% emisi CO_2 pada penyelenggaraan pendidikan konvensional di tingkat pendidikan tinggi.

Jumlah penggunaan platform pendidikan digital seperti LMS bergantung pada beberapa faktor, seperti kemudahan penggunaan, kenyamanan, norma subjektif, kepuasan, dan interaktifitas (Fındık-Coşkunçay, et al., 2018). Akan tetapi, inefisiensi kinerja serta ketidaksesuaian antara aspek kegunaan fungsi dan kemudahan penggunaan menjadi tantangan tersendiri dalam perkembangan platform pendidikan digital. Inefisiensi kinerja LMS sangat dipengaruhi oleh banyaknya fitur yang disediakan, tingginya frekuensi penggunaan, dan banyaknya pengguna (Roy, et al., 2018). Inefisiensi kinerja Moodle yang tidak tertangani menjadi tantangan tersendiri yang terus semakin buruk seiring semakin banyak jumlah penggunanya. Inefisiensi kinerja Moodle menjadi tantangan yang berdampak pada kondusifitas pembelajaran yang diselenggarakan pada platform tersebut (Garone, et al., 2019). Inefisiensi kinerja Moodle berupa buruknya *throughput* dan *latency* sistem yang dirasakan oleh pengguna menjadi hambatan dalam aktivitas pembelajaran pada platform pendidikan digital (Roy, et al., 2018).

Menurut David, et al. (2022), permasalahan inefisiensi kinerja Moodle dapat ditangani dengan *vertical scaling* pada *server*. Akan tetapi metode *vertical scaling* bersifat tetap dalam arti tidak dapat disesuaikan dengan fluktuasi penggunaan *server* secara fleksibel, serta membutuhkan biaya perbaruan perangkat yang tinggi. Kekurangan pada metode *vertical scaling* dapat ditangani dengan *horizontal scaling* pada *server* dengan dukungan layanan komputasi awan. Namun keduanya tidak menyentuh persoalan inefisiensi kinerja dari sisi aplikasi Moodle. Jika inefisiensi kinerja dari sisi aplikasi Moodle tidak dilakukan diperbaiki, maka sumber daya *server* yang digunakan tidak dapat berjalan secara optimal sebagaimana seharusnya. Inefisiensi kinerja Moodle dapat diperbaiki dengan penggunaan mekanisme *caching* sebagai manajemen penggunaan data untuk efektivitas dan efisiensi sumber daya (Bernardo & Bontà, 2020; Sathiyamoorthi, et, al., 2020).

2. KAJIAN LITERATUR

Dalam 5 tahun terakhir, penelitian yang dilakukan pada topik penerapan mekanisme *caching* untuk mengatasi permasalahan inefisiensi kinerja Moodle masih cenderung sedikit. Beberapa penelitian seperti (Mihai, et al., 2023), (Hamamoto, et al., 2022), dan (Mgongo, et al., 2022) hanya melibatkan penggunaan mekanisme *caching* pada Moodle dalam penelitiannya tanpa analisis mendalam terkait kinerja mekanisme *caching* yang digunakan. Selain itu penelitian terkini yang disertai pembahasan kinerja mekanisme *caching* yang lebih elaboratif dilakukan (David, et al., 2022) dan (Ramadhan, I. & Wulandari, L., 2022). Di antara sedikit penelitian tersebut, hampir seluruhnya menggunakan Redis dalam mekanisme *caching* yang diajukan kecuali (Mgongo, et al., 2022) yang menggunakan react redux dengan penjabaran yang cenderung kurang mendalam mengenai mekanisme *caching* yang digunakan. (David, et al., 2022) mempertimbangkan penggunaan Memcached dalam mekanisme *caching* yang dilakukan dengan dukungan *throughput* yang lebih tinggi, akan tetapi kemudian memilih menggunakan Redis untuk dengan pertimbangan *CPU usage* dan *cache miss* yang lebih sedikit dalam mekanisme *caching* yang diajukan dalam penelitiannya.

Moodle mendukung mekanisme *caching* dengan berbagai basis data sebagai *cache store*-nya, seperti APC user cache (APCu), Memcached, MongoDB, dan Redis untuk mengoptimasi kinerjanya,. Mekanisme *caching* merujuk pada penyimpanan sementara dan pengaksesan kumpulan data yang sering diakses yang telah diproses sebelumnya di memori.

Mekanisme *caching* bertujuan untuk mengurangi pemrosesan ulang yang memakan waktu sehingga dapat meningkatkan kecepatan pengaksesan kumpulan data tersebut dengan mengurangi *disk storage I/O* (Sathiyamoorthi, et al., 2020). Penerapan mekanisme *caching* pada sisi client, proxy, maupun *server* bertujuan untuk mengoptimasikan penggunaan sumber daya perangkat. (Zulfa, et al., 2020) Penerapannya pada *server* menghasilkan *throughput* yang lebih tinggi serta *latency* yang lebih rendah di sisi pengguna (Paschos, et al., 2020).

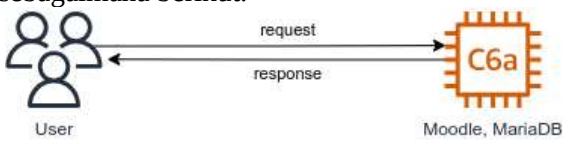
Mekanisme *caching* pada Moodle diatur pada daftar *cache definition* yang menentukan kategori data spesifik yang digunakan dalam mekanisme *caching* (Moodle, 2023a). Penelitian ini menggunakan MongoDB sebagai *cache store* dari mekanisme *caching* yang dilakukan pada Moodle. Mekanisme *caching* Moodle dengan MongoDB memiliki dukungan pada *cache definition* dengan mode *application cache*. Hal ini cukup krusial dikarenakan secara kuantitas *application cache* mencakup 57 dari keseluruhan 87 cache definition yang diatur secara *default* pada Moodle. Dalam hal ini, mekanisme *caching* yang digunakan pada *application cache* sangat berpengaruh pada peningkatan kinerja Moodle yang diharapkan. Selain itu Moodle memiliki dukungan bagi penggunaan MongoDB dengan *server* terkluster dalam mekanisme *caching* yang dilakukan. Penerapan *server* terkluster merupakan solusi dari persoalan penurunan kinerja Moodle yang diakibatkan oleh peningkatan beban trafik pengguna (David, et al., 2022; Ramadhan, I. & Wulandari, L., 2022).

MongoDB Cluster adalah terapan basis data terdistribusi pada MongoDB. Dukungan untuk melakukan *horizontal scaling* pada sisi basis data disediakan oleh MongoDB melalui fitur *sharded cluster* yang membagi *collection* data berdasarkan *chunk* ke beberapa *node server* berbeda, sedangkan dukungan untuk penyediaan *high availability* yang lebih tinggi disediakan melalui fitur *replica set* (MongoDB, 2023a). Sebuah *replica set* terdiri dari sebuah *node server* sebagai *primary* yang secara *default* melayani *request* dari *client* berupa operasi *read* dan *write*, serta dua atau lebih *server* lain yang dapat berperan sebagai *secondary* maupun *arbiter*. *Node server* yang berperan sebagai *secondary* mereplikasi data dari *node primary* secara rutin sebagai *failover*. Selain menerima replikasi data dari *node server primary*, dengan pengaturan *read preference*, *server secondary* juga dapat digunakan menjadi *server* yang berperan melayani *request* pembacaan data dari *client* melalui MongoDB PHP driver (MongoDB, 2023d).

Maka untuk memecahkan permasalahan inefisiensi kinerja Moodle yang telah disampaikan, penelitian ini mengajukan penerapan mekanisme *caching* pada Moodle menggunakan MongoDB Cluster. Penelitian ini dilakukan guna memberi gambaran mengenai dampak implementasi mekanisme *caching* MongoDB Cluster pada peningkatan kinerja Moodle. Adapun untuk lingkungan dan sarana implementasi, penelitian ini menggunakan penelitian (Mwakisole, et al., 2019), (Espinoza-Guerrero & Bayani-Abbasy, 2019) dan (Ibrahim, 2019) untuk rujukan dalam membangun arsitektur sistem yang diusulkan. Sistem diimplementasikan menggunakan layanan komputasi awan sebagai lingkungan implementasi dengan rujukan pada (Mwakisole, et al., 2019) dan (Espinoza-Guerrero & Bayani-Abbasy, 2019), serta diimplementasikan menggunakan docker compose untuk mengotomasi kontainerisasi serta untuk membangun lingkungan isolasi pada *server* dengan rujukan pada (Ibrahim, 2019).

3. PERANCANGAN

Arsitektur dasar Moodle tanpa penerapan mekanisme *caching* disimulasikan peneltian ini sebagaimana berikut.



Gambar 1. Arsitektur Sistem Tanpa Mekanisme *Caching* MongoDB Cluster

Service Moodle dan MariaDB berjalan di sebuah instans *server* virtual Amazon EC2 C6a.large. *User* mengakses Moodle pada instans *server* virtual tersebut dalam kegiatan pembelajaran digital yang dilangsungkan. Moodle menggunakan MariaDB dalam kebutuhan basis data yang menyimpan data yang bersifat informasi, seperti data identitas akun pengguna, kursus, aktivitas, tugas, forum, kuis, sumber daya belajar lainnya, nilai, dan pesan antar pengguna yang serluruh data tersebut disimpan pada tabel-tabel pada basis data MariaDB. Sedangkan data lainnya yang berformat file seperti foto pengguna, konten pembelajaran, dan file konfigurasi lainnya disimpan oleh Moodle pada *storage disk* Amazon EC2 C6a.large. Alamat IP yang diakses

oleh pengguna menggunakan *elastic IP* yang memungkinkan alokasi alamat IP konstan untuk memudahkan akses pengguna.

Adapun untuk memberikan gambaran bagi arsitektur yang diusulkan, analisis kebutuhan dan perancangan dilakukan. Analisis kebutuhan terdiri dari analisis kebutuhan aplikasi dan perangkat yang merujuk pada McCabe (2007). Perancangan menguraikan rancangan arsitektur yang diusulkan berdasarkan analisis kebutuhan yang dilakukan.

Pada analisis kebutuhan aplikasi, seluruh aplikasi yang digunakan pada sistem dihimpun pada Tabel 1.

Tabel 1. Spesifikasi Aplikasi Pada Sistem Dengan Mekanisme *Caching* MongoDB Cluster

No	Aplikasi	Keterangan
1	Docker	Aplikasi yang berperan dalam otomasi kontainerisasi dan konfigurasi dari seluruh aplikasi yang akan dijalankan di setiap *server*.
2	Moodle	Aplikasi utama pada penelitian ini yang menjadi platform pembelajaran daring yang melibatkan pengajar dan siswa.
3	MariaDB	Aplikasi yang digunakan oleh Moodle sebagai basis data yang menyimpan berbagai informasi yang digunakan pengguna seperti informasi pengguna, kelas, nilai, forum, komentar, maupun kuis.
4	MongoDB	Aplikasi basis data yang digunakan sebagai instans *cache store* dalam mekanisme *caching* pada Moodle.

Selanjutnya kebutuhan minimum perangkat keras dari setiap aplikasi yang digunakan pada sistem dihimpun pada Tabel 2.

Tabel 2. Kebutuhan Minimum/Rekomendasi Aplikasi Pada Sistem Dengan Mekanisme *Caching* MongoDB Cluster

Aplikasi	Kebutuhan Minimum/ Rekomendasi CPU	Kebutuhan Minimum/ Rekomendasi Memori	Perangkat
Docker	64-bit *kernel*	4 GB RAM	*Server* Utama
Moodle	2 GHz *dual core*	1 GB RAM	*Server* Utama
MariaDB	-	-	*Server* Utama
MongoDB	64-bit *kernel*	4 GB RAM	*Server* Utama (MongoDB Client) dan Seluruh *Server* MongoDB

Kebutuhan minimum/rekomendasi aplikasi tersebut selanjutnya akan menjadi *input* dalam melakukan analisis kebutuhan perangkat. *Input* tersebut digunakan untuk memetakan kebutuhan perangkat bagi arsitektur yang diusulkan. Pada analisis kebutuhan perangkat, seluruh perangkat yang digunakan pada sistem dihimpun pada Tabel 3.

Tabel 3. Spesifikasi Perangkat Pada Sistem Dengan Mekanisme *Caching* MongoDB Cluster

No	Perangkat	Keterangan
1	*Server* Utama	*Server* yang menjalankan aplikasi Moodle yang diakses oleh *user* beserta MariaDB sebagai basis data yang digunakan.
2	MongoDB *Primary* (Mongo1)	*Server* yang menjalankan MongoDB dengan *role primary* dari 3 *node server* pada MongoDB Cluster yang digunakan dalam mekanisme *caching*.
3	MongoDB *Secondary*-1 (Mongo2)	*Server* yang menjalankan MongoDB dengan *role secondary* pertama dari 3 *node server* pada MongoDB Cluster yang digunakan dalam mekanisme *caching*.
4	MongoDB *Secondary*-2 (Mongo3)	*Server* yang menjalankan MongoDB dengan *role secondary* kedua dari 3 *node server* pada MongoDB Cluster yang digunakan dalam mekanisme *caching*.

Rincian perangkat yang digunakan pada sistem dihimpun dalam tabel deskripsi perangkat pada Tabel 4.

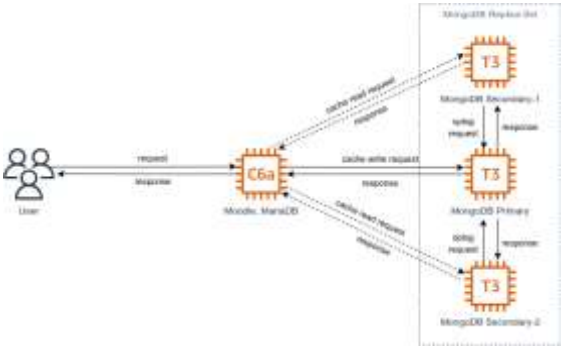
Tabel 4. Deskripsi Perangkat Pada Sistem Dengan Mekanisme *Caching* MongoDB Cluster

Perangkat	Instans Komputasi Awan	Prosesor	Memori	Aplikasi
Server Utama	Amazon EC2 C6a.large	64-bit (x86) 2 vCPU 3rd generation AMD EPYC	4 GB	Docker, Moodle, MariaDB, MongoDB PHP Driver
Server MongoDB *Primary*	Amazon EC2 T3.large	64-bit (x86) 2 vCPU Intel Xeon Scalable processor	8 GB	Docker, MongoDB
Server MongoDB *Secondary* -1	Amazon EC2 T3.large	64-bit (x86) 2 vCPU Intel Xeon Scalable processor	8 GB	Docker, MongoDB
Server MongoDB *Secondary* -2	Amazon EC2 T3.large	64-bit (x86) 2 vCPU Intel Xeon	8 GB	Docker, MongoDB

Scalable processor

MongoDB PHP Driver digunakan agar Moodle yang menggunakan PHP mampu berkomunikasi dengan *node-node server* MongoDB Cluster sebagai MongoDB Client. Amazon EC2 C6a.large digunakan sebagai *server* utama karena kekhususannya pada kemampuan komputasi (Amazon EC2 tipe *compute optimized*) sebagai dukungan bagi kebutuhan komputasi dari aplikasi Moodle. Selain itu processor AMD EPYC yang digunakan pada Amazon EC2 C6a.large memiliki kemampuan komputasi *multi-threaded* yang tinggi untuk kebutuhan komputasi yang dibutuhkan Moodle (AMD, 2023; Moodle, 2023c).

Amazon EC2 T3.large digunakan sebagai *server* pada ketiga *node server* MongoDB Cluster dalam penggunaan umum (general purpose) dengan kekhususannya dalam menyesuaikan kemampuan komputasi dengan tingkat kepadatan dan kelonggaran trafik akses pada *server* (instans Amazon EC2 sub tipe *burstable general purpose*) sebagai dukungan bagi kebutuhan dari aplikasi MongoDB yang digunakan pada mekanisme *caching*. Selain itu processor Intel Xeon Scalable processor yang digunakan pada Amazon EC2 T3.large memiliki kemampuan komputasi *single-threaded* yang tinggi untuk kebutuhan komputasi yang dibutuhkan MongoDB (Spherex, 2022). Berdasarkan kebutuhan aplikasi dan perangkat yang telah dijabarkan, rancangan arsitektur yang diusulkan penelitian ini adalah sebagaimana berikut.



Gambar 2 Arsitektur Sistem Dengan Mekanisme *Caching* MongoDB Cluster

Service Moodle dan MariaDB berjalan pada instans *server* virtual Amazon EC2 C6a.large sebagai *server* utama yang digunakan. Ketiga *node server* MongoDB dalam *replica set* digunakan Moodle dalam mekanisme *caching* yang dilakukan pada seluruh data yang tercakup pada *cache definition* Moodle. Ketiga *node server* pada *replica set* MongoDB menggunakan instans *server* virtual Amazon EC2 T3.large. Dalam mekanisme *caching* yang dilakukan ketika membutuhkan penulisan data (operasi *write*), Moodle akan mengirimkan *request* ke *server* MongoDB *primary*. Alamat IP yang diakses oleh pengguna menggunakan *elastic IP* yang memungkinkan alokasi alamat IP konstan untuk memudahkan akses pengguna. Seluruh instans *server* virtual pada sistem ini berjalan pada private subnet yang sama dan saling terhubung menggunakan IP private dari instans *server* virtual masing-masing.

Dengan *role server* MongoDB *primary* sebagai *primary node* dalam *replica set*, *request* berupa operasi *write* hanya akan dikirim oleh Moodle ke *server* MongoDB *primary*. *Read/write splitting* diterapkan untuk mengkhususkan kinerja *server* basis data pada operasi *read* dan *write* dalam mekanisme *caching* yang dilakukan menggunakan pengaturan *read preference* bernilai *secondary preferred*. Dalam proses replikasi data dari *server* MongoDB *primary*, *server* MongoDB *secondary*-1 dan *server* MongoDB *secondary*-2 mengirim *oplog request* secara rutin kepada *server* MongoDB *primary*. *Oplog* (*operation log*) pada MongoDB merupakan catatan *log* yang berisi riwayat operasi yang dilakukan oleh MongoDB. Kedua *server* MongoDB *secondary* memperbarui basis datanya secara rutin dengan mekanisme replikasi berdasarkan *oplog* yang dikirimkan dari *server* MongoDB *primary*.

Sehingga *request* berupa operasi *read* hanya akan dikirim oleh Moodle ke *server* MongoDB *secondary* jika selama terdapat *server* MongoDB *secondary* yang tersedia pada *replica set*. Jika tidak ada di antara kedua *server* MongoDB *secondary* yang tersedia, sebagai mekanisme *replica set* operasi *read* dapat dikirimkan oleh Moodle ke *server* MongoDB primary. Dengan begitu *server* MongoDB primary bisa meminimalisir operasi *read* untuk memaksimalkan seluruh kapasitas sumber dayanya untuk operasi *write* dan mengirim *oplog* ke *server* MongoDB *secondary* untuk replikasi data. Sedangkan *server* MongoDB *secondary* dapat memaksimalkan seluruh kapasitas sumber dayanya untuk operasi *read*, dan melakukan replikasi data berdasarkan *oplog* yang diterima dari *server* MongoDB primary (Huang, et al., 2019). Dengan demikian diharapkan sistem dapat memberikan kinerja yang lebih baik meski mengalami trafik *request* yang tinggi dalam

operasi penulisan dan pembacaan data dalam mekanisme *caching* yang dilakukan. (Ramadhan, I. & Wulandari, L., 2022).

4. IMPLEMENTASI

Pada *server* utama, *service* Moodle dan MariaDB dijalankan menggunakan docker compose untuk otomasi konfigurasi dan fleksibilitas penggunaan *script*-nya. Dengan begitu konfigurasinya dapat digunakan berulang secara praktis pada *server* lain dengan menyalin dan menjalankan *script* konfigurasi yang telah dibuat sebelumnya. Docker compose digunakan untuk membuat container Moodle sebagai aplikasi utama dan container MariaDB sebagai basis data yang digunakan oleh Moodle. Container dibangun dan dijalankan dengan script docker-compose.yml melalui perintah `docker compose up`.

Script docker-compose.yml merupakan file konfigurasi yml yang dijalankan oleh docker compose untuk membangun docker container pada *server* utama. *Script* ini menggunakan *image* MariaDB dan Moodle yang dikustomisasi melalui Dockerfile dari repositori Docker Hub akun penulis untuk membangun container yang menjalankan Moodle dan MariaDB serta instalasi MongoDB PHP Driver agar *container* tersebut dapat menjalankan MongoDB Client. Melalui kredensial *database host, port, user, password*, dan nama *database* sebagai *environment* di docker-compose.yml, Moodle dimungkinkan untuk terhubung dengan MariaDB.

MongoDB Cluster diimplementasikan dalam sebuah *replica set* yang terdiri 3 *node* yang *server* menggunakan instans Amazon EC2 T3.large. Konfigurasi dan inisiasi *replica set* MongoDB Cluster dilakukan dengan *script* yang menjalankan perintah `rs.initiate()` pada *shell* mongosh. Dengan perintah `rs.initiate()` *shell* mongosh akan mencoba menghubungi seluruh *node server* melalui alamat IP dan *port* yang didefinisikan pada parameter alamat IP dan *port* dari setiap *member*. Oleh karenanya sebelum perintah `rs.initiate()` dijalankan dengan *script* tersebut, seluruh *node server* yang akan menjadi *member* dari *replica set* yang akan dibuat harus berjalan terlebih dahulu. Dengan begitu maka *server* MongoDB *secondary* diinisiasi terlebih dahulu sebelum diinisiasikannya *server* MongoDB *primary* yang bersamaan dengan inisiasi dan konfigurasi *replica set* melalui *script* yang berisi perintah `rs.initiate()`.

Kedua *node server* MongoDB *secondary* masing-masing diinisiasikan menggunakan *script* docker-compose.yml dengan perintah `docker compose up`. Kedua *script* docker-compose.yml tersebut berisikan bagian konfigurasi `entrypoint: [ "/usr/bin/mongod", "--bind_ip_all", "--replSet", "myReplicaSet" ]` yang membuat kedua node *server* MongoDB *secondary* yang baru saja diinisiasikan tersebut menunggu dijalankannya perintah `rs.initiate()` pada *server* MongoDB *primary* untuk menginisiasi sebuah *replica set* dengan nama myReplicaSet untuk menjadi *member* dari *replica set* tersebut.

Setelah kedua *node server* MongoDB *secondary* diinisiasikan, maka *node server* MongoDB *primary* telah bisa diinisiasikan. *Script* docker-compose.yml akan menginisiasikan *server* MongoDB *primary* yang bagian konfigurasi `entrypoint: [ "/usr/bin/mongod", "--bind_ip_all", "--replSet", "myReplicaSet" ]` membuat *server* MongoDB *primary* berada pada status menunggu diinisiasikan dan dikonfigurasikannya *replica set* bernama myReplicaSet untuk menjadi *member* dari *replica set* tersebut. Setelah *service* MongoDB pada *server* MongoDB *primary* diinisiasikan, selanjutnya *replica set* dari MongoDB Cluster dikonfigurasikan yang terdiri dari *node server* MongoDB *primary* dan kedua *node server* MongoDB secondary sebagai *member*-nya pada *port* 27017. Dengan begitu *replica set* dari MongoDB Cluster bernama “myReplicaSet” telah berhasil diimplementasikan.

Setelah *replica set* MongoDB Cluster telah diinisiasi, kinerja pembacaan data pada MongoDB Cluster ditingkatkan melalui pengaturan *read preference* pada file /bitnami/moodle/cache/stores/mongodb/lib.php pada *container* Moodle. Secara *default*, nilai dari variabel *read preference* yang digunakan pada MongoDB PHP Driver adalah `RP_PRIMARY`. Untuk meningkatkan kinerja pembacaan data pada MongoDB Cluster, pengaturan *read preference* bernilai *secondary preferred* digunakan. Dengan pengaturan tersebut pembacaan data terutama akan menggunakan *node server* MongoDB *secondary*. Akan tetapi sebagai antisipasi jika karena berbagai faktor ternyata seluruh *node server* MongoDB *secondary* tidak tersedia atau mengalami kegagalan, maka dengan pengaturan *read preference* bernilai *secondary preferred* pada situasi seperti itu *server* MongoDB *primary*

dapat digunakan untuk melayani operasi *read* dari *client*. Pengaturan *read preference* dilakukan dengan mengubah dari nilai defaultnya menjadi `RP_SECONDARY_PREFERRED` pada file /bitnami/moodle/cache/stores/mongodb/lib.php pada container Moodle.

Dengan telah diinstalasinya MongoDB PHP Driver menggunakan PECL, Moodle dapat menggunakan MongoDB sebagai instans bagi *cache store* yang akan digunakan pada mekanisme *caching*. Setelah itu konfigurasi *caching* dilakukan melalui antarmuka Moodle pada *web browser*. Pada halaman cache configuration, instans *cache store* baru menggunakan MongoDB ditambahkan dengan melakukan *input* pada *field* yang dibutuhkan. Pada *field* "Server", *connection string* yang di-*input*-kan adalah yang terhubung dengan *server* MongoDB *primary* sebagai *node primary* pada *replica set*. Selanjutnya *cache store* tersebut dipetakan bagi data dengan *cache definition* mode *application cache* pada tabel "Stores used when no mapping is present". Dengan begitu mekanisme *caching* pada data dengan *cache definition* mode *application cache* dapat dilakukan menggunakan ketiga *node server* pada *replica set* MongoDB Cluster yang telah dikonfigurasikan.

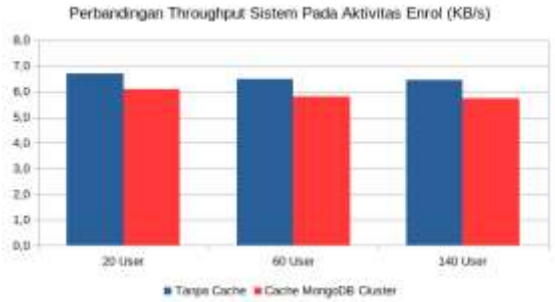
5. PENGUJIAN DAN PEMBAHASAN

Kinerja mekanisme *caching* MongoDB Cluster dievaluasi menggunakan Apache JMeter melalui pengiriman *request* dengan *sampler* dan perekaman data dengan *listener*. Penelitian ini memantau *throughput* dan *latency* sistem untuk meninjau dampak penerapan mekanisme *caching* MongoDB Cluster pada peningkatan kinerja Moodle. Semakin tinggi *throughput* dan semakin rendah *latency*, maka semakin efektif mekanisme caching yang diterapkan (Paschos, et al., 2020). Perekaman pada data *throughput* dilakukan menggunakan *listener Summary Report*, sedangkan perekaman pada data *latency* dilakukan menggunakan *listener View Results in Table* (Rodrigues, et al., 2019).

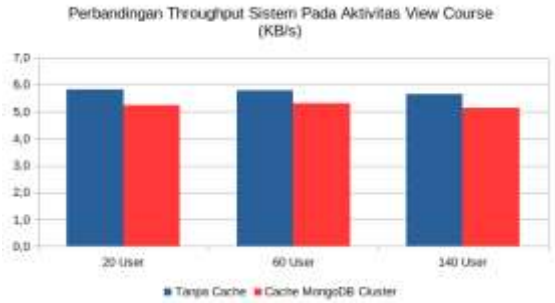
Guna melihat seberapa kadar mekanisme *caching* MongoDB Cluster berpengaruh terhadap kinerja Moodle, pengujian dilakukan dengan memperbandingan kinerja antara (a) sistem yang dijalankan tanpa cache dan (b) sistem yang dijalankan dengan cache MongoDB Cluster. Kedua skenario tersebut diuji dengan tiga skenario *thread group*, yakni 20, 60, dan 140 *user* yang mengirimkan mengirimkan *HTTP request* secara serentak. Adapun *request* yang dilakukan oleh setiap user-nya meliputi (a) *enrol*, (b) *view course*, serta (c) *quiz* menggunakan *sampler HTTP request* dengan *method* GET dan POST. Apache JMeter dijalankan pada Amazon EC2 dengan availability zone yang sama dengan *server* utama. Pengujian dilakukan dengan ketiga skenario *thread user* yang mengirimkan *request* per aktivitas *user* kepada alamat *IP private* dari *server* utama.

Gambar 3. Perbandingan *Throughput* Sistem

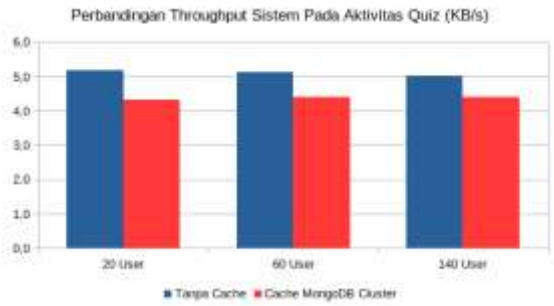
(a) Pada Aktivitas *Enrol*



(b) Pada Aktivitas *View Course*



(c) Pada Aktivitas *Quiz*

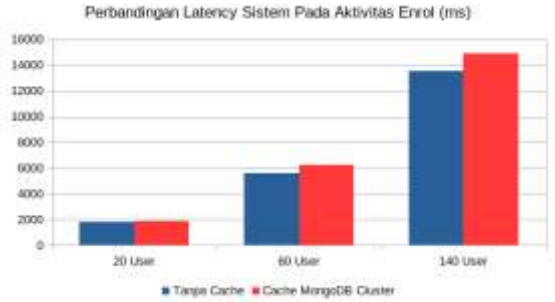


Pengujian menunjukkan menurunnya *throughput* saat mekanisme *caching* MongoDB Cluster diterapkan pada ketiga aktivitas *user*. Dengan beragam fluktuasinya, penggunaan MongoDB Cluster dalam mekanisme *caching* Moodle tidak meningkatkan *throughput* sistem. Penurunan *throughput* terendah yang didapati pada aktivitas *view course* dengan skenario 60

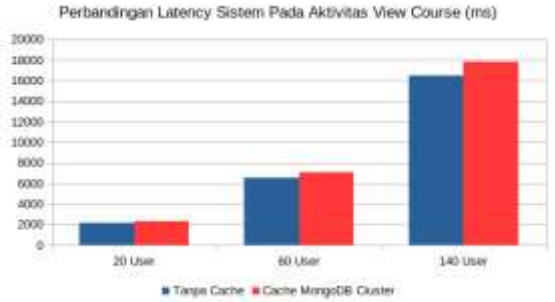
user (-8,26%), sedangkan penurunan *throughput* tertinggi berada pada aktivitas *quiz* dengan skenario 20 *user* (-16,96%).

Gambar 4. Perbandingan *Latency* Sistem

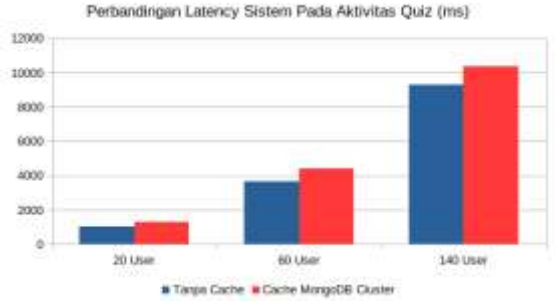
(a) Pada Aktivitas *Enrol*



(b) Pada Aktivitas *View Course*



(c) Pada Aktivitas *Quiz*



Begitu juga dari sisi *latency*. Dengan beragam fluktuasinya, penggunaan MongoDB Cluster dalam mekanisme *caching* Moodle tidak menurunkan *latency* sistem. Peningkatan *latency* terendah didapati pada aktivitas *enrol* dengan skenario 20 *user* (-1,70%), sedangkan peningkatan *latency* tertinggi berada pada aktivitas *quiz* dengan skenario 20 *user* (-27,54%).

6. KESIMPULAN

Pada penelitian ini didapati bahwa mekanisme *caching* MongoDB Cluster pada Moodle dapat diimplementasikan dengan MongoDB PHP Driver pada *server* utama yang menjalankan service Moodle. Dari hasil pengujian didapati implementasi mekanisme *caching* MongoDB Cluster pada penelitian ini berimplikasi negatif terhadap peningkatan kinerja Moodle. Dengan kata lain, mekanisme *caching* MongoDB Cluster yang diimplementasikan belum berhasil meningkatkan kinerja Moodle. Berdasarkan *throughput*, angka penurunan *throughput* terendah yang ditemukan berada pada aktivitas *view course* dengan skenario 60 *user* (-8,26%), sedangkan angka penurunan *throughput* tertinggi berada pada aktivitas *quiz* dengan skenario 20 *user*.(-16,96%). Sedangkan berdasarkan *latency*, angka peningkatan *latency* terendah yang ditemukan berada pada aktivitas *enrol* dengan skenario 20 *user* (-1,70%), sedangkan angka peningkatan *latency* tertinggi berada pada aktivitas *quiz* dengan skenario 20 *user* (-27,54%).

Salah satu penyebabnya adalah penggunaan *storage engine* WiredTiger pada MongoDB Cluster tanpa konfigurasi lebih lanjut tidak cukup memadai untuk bisa memberikan peningkatan kinerja untuk kebutuhan mekanisme *caching*. Hal ini disebabkan pengaturan *default storage engine* WiredTiger ditujukan untuk penyimpanan data secara persisten di *storage disk server* (MongoDB, 2023d). Sedangkan semakin banyak *storage disk I/O*, semakin lambat pula kecepatan akses data yang bisa diberikan MongoDB Cluster kepada Moodle yang dalam hal ini berada pada posisi *client* dari MongoDB Cluster (MongoDB, 2023b).

7. SARAN

Kinerja mekanisme *caching* MongoDB Cluster pada Moodle dapat diperbaiki pada penelitian selanjutnya dengan beberapa pertimbangan sebagai berikut:

1. Frekuensi *storage disk I/O* pada *server* MongoDB bisa lebih diminimalisir menggunakan pengaturan `storage.syncPeriodSecs` dengan nilai 0 pada konfigurasi MongoDB. Hal itu dapat dilakukan agar MongoDB tidak melakukan sinkronisasi *mapped files* dalam memori kepada *storage disk* untuk meningkatkan kinerja mekanisme *caching* yang bersifat lebih volatil.
2. Penelitian selanjutnya dapat mempertimbangkan penggunaan *storage engine* In-Memory alih-alih menggunakan WiredTiger. Pengaturan *default* WiredTiger menyajikan

kekuatan persistensi data yang memperlambat kecepatan akses dalam mekanisme caching, sedangkan *storage engine* In-Memory dikhususkan untuk penggunaan MongoDB secara volatil dan non persisten untuk menyajikan kecepatan akses yang tinggi seperti dalam kasus penggunaan untuk mekanisme *caching*.

3. Selain itu mekanisme *caching* MongoDB mendukung aspek *data guarantee* pada Moodle, namun penelitian ini tidak melakukan analisis dan elaborasi lebih lanjut pada aspek tersebut. Penelitian selanjutnya dapat mempertimbangkan analisis pada aspek *data guarantee* yang didukung oleh mekanisme *caching* MongoDB pada Moodle.

8. DAFTAR PUSTAKA

AMD. 2019. AMD EPYC™ and NAMD® Powering the Future of HPC. [online] Tersedia di: <https://www.amd.com/system/files/documents/namd-gets-high-performance-with-amd-epyc.pdf> [Diakses 20 Juli 2023]

AWS. 2023. Instance types - Amazon Elastic Compute Cloud. [online] Tersedia di: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html> [Diakses 2 Mei 2023]

Bernardo, M. & Bontà, E. 2020. Facing the COVID-19 pandemic: Massive distance learning and on-line exams with moodle, collaborate, smowl, meet. *Proc. 2nd International Conference on Higher Education Learning Methodologies and Technologies Online (HELMeTO)*, 81-84.

David, A., Mihai, D., Mihailescu, M.E., Carabas, M. & Tapus, N. 2022. Scalability through Distributed Deployment for Moodle Learning Management System. *Procedia Computer Science*, 214, 34-41. doi:10.1016/j.procs.2022.11.145.

Espinoza-Guerrero, M. & Bayani-Abbasy, M. 2019. Implementation of a Mini-cloud E-learning Supplementary Tool by Using free Tier AWS. *Memorias del I Congreso Internacional de Ciencias Exactas y Naturales*. doi:10.15359/CICEN.1.79.

Findik-Coşkunçay, D., Alkiş, N. & Özkan-Yildirim, S. 2018. A structural model for students' adoption of learning management systems: An empirical investigation in the higher education context. *Journal of Educational Technology & Society*, 21(2), 13-27. doi:10.1037/t70573-000.

Garone, A., Pynoo, B., Tondeur, J., Cocquyt, C., Vanslambrouck, S., Bruggeman, B. & Struyven, K. 2019. Clustering university teaching staff through UTAUT: Implications for the acceptance of a new learning management system. *British Journal of Educational Technology*, 50(5), 2466-2483. doi: 10.1111/bjet.12867.

Huang, C., Cahill, M., Fekete, A. & Röhm, U. 2019. Data Consistency Properties of Document Store as a Service (DSaaS): Using MongoDB Atlas as an Example. *Dalam:* NAMBIAR, R. & POESS M. (editor) *Performance Evaluation and Benchmarking for the Era of Artificial Intelligence, TPCTC 2018, Programming and Software Engineering (LNPSE)*, 11135, 126-139. Springer, Cham. doi:10.1007/978-3-030-11404-6_10.

Ibrahim, M. H. 2019. A study of the use of Docker compose and dockerhub images. *Master Thesis*. Queen's University Canada.

McCabe, J. D. 2007. *Network Analysis, Architecture, and Design*. 3rd ed. Morgan Kaufmann, Burlington.

Moodle. 2023a. Cache definitions - MoodleDocs. [online] Tersedia di: <https://docs.moodle.org/400/en/Cache_definitions> [Diakses 11 Juli 2023]

Moodle. 2023b. Moodle statistics. [online] Tersedia di: <https://stats.moodle.org/> [Diakses 24 Juli 2023]

Moodle. 2023c. Performance recommendations - MoodleDocs. [online] Tersedia di: <https://docs.moodle.org/400/en/Performance_recommendations> [Diakses 24 Juli 2023]

MongoDB. 2023a. MongoDB Clusters | MongoDB. [online] Tersedia di: <https://www.mongodb.com/basics/clusters> [Diakses 30 Juli 2023]

MongoDB. 2023b. In-Memory Storage Engine — MongoDB Manual. [online] Tersedia di: <https://www.mongodb.com/docs/manual/core/inmemory/> [Diakses 24 Agustus 2023]

MongoDB. 2023c. Replication — MongoDB Manual. [online] Tersedia di: <https://www.mongodb.com/docs/manual/replication/> [Diakses 30 Juli 2023]

MongoDB. 2023d. Storage Engines — MongoDB Manual. [online] Tersedia di: <https://www.mongodb.com/docs/manual/core/storage-engines/> [Diakses 24 Agustus 2023]

Mwakisole, K. F., Kissaka, M. M. & Mtebe, J. S. 2019. Cloud computing architecture for elearning systems in secondary schools in Tanzania. *The African Journal of Information Systems*, 11(4), 299-313.

Ramadhan, I. & Wulandari, L. 2022. Infrastruktur High-Available Learning Management System Universitas Menggunakan Least-Connected Load Balancer. *JURNAL MASYARAKAT INFORMATIKA*, 13(2), 99-100. doi:10.14710/jmasif.13.2.49176.

Rodrigues, A. G., Demion, B. & Mouawad, P. 2019. *Master Apache JMeter-From Load Testing to DevOps: Master performance testing with JMeter*. Packt Publishing Ltd, Birmingham.

Paschos, G. S., Iosifidis, G. & Caire, G. 2020. Cache Optimization Models and Algorithms. *Foundations and Trends in Communications and Information Theory*, 16(3-4), 156-345. doi:10.1561/0100000104.

Roy, R., Potter, S. & Yarrow, K. 2008. Designing low carbon higher education systems: Environmental impacts of campus and distance learning systems. *International Journal of Sustainability in Higher Education*, 9(2), 116-130. doi:10.1108/14676370810856279.

Roy, S., Williamson, C. & McLean, R. 2018. LMS performance issues: a case study of D2L. *International Journal of Computers and Their Applications*, 25(3), 113-122.

Sathiyamoorthi, V., Suresh, P., Jayapandian, N., Kanmani, P. & Janakiraman, S. 2020. An Intelligent Web Caching System for Improving the Performance of a Web-Based Information Retrieval System. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 16(4), 26-44. doi:10.4018/IJSWIS.2020100102.

Spherex. 2022. Taking a look at benchmark results for AWS instances - Spherex. [online] Tersedia di: <https://www.spherex.dev/taking-a-look-at-geekbench-results-for-aws-instances/> [Diakses 20 Juli 2023]

Xin, N. S., Shibghatullah, A. S. & Abd Wahab, M. H. 2021. A systematic review for online learning management system. *Journal of Physics: Conference Series*, 1874(012030). doi:10.1088/1742-6596/1874/1/012030.

Zulfa, M. I., Hartanto, R. & Permanasari, A. E. 2020. Caching strategy for Web application–a systematic literature review. *International Journal of Web Information Systems*, 16(5), 545-569. doi:10.1108/ijwis-06-2020-0032.